# Le formalisme objet appliqué à l'étude de l'édifice patrimonial : Problèmes de modélisation et d'échanges de données sur le réseau Internet

Jean-Yves Blaise

UNIVERSITE DE DROIT D'ECONOMIE ET DES SCIENCES D'AIX-MARSEILLE
(AIX-MARSEILLE III)

TITRE

# Le formalisme objet appliqué à l'étude de l'édifice patrimonial : Problèmes de modélisation et d'échanges de données sur le réseau Internet

THESE
Pour obtenir le grade de :

DOCTEUR DE L'UNIVERSITE DE DROIT D'ECONOMIE ET DES SCIENCES D'AIX-MARSEILLE

DISCIPLINE : Mathématiques et Informatique

Présentée et soutenue publiquement
par
Jean-Yves BLAISE
le 6 Mars 2003

---

***Directeur de thèse : Michel FLORENZANO***

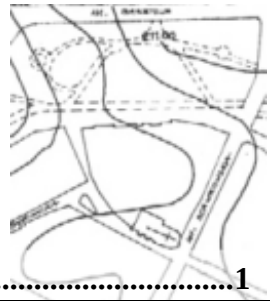
---

**JURY :**

**M. BERTRAND Jean-Claude ;**
**Mme. COUREL Marie-Françoise;**
**M. LE MAITRE Jacques,**
**M. LE MEN Hervé;**
**M. FLORENZANO Michel,**
**M. MELOT Michel.**

ANNEE : 2003

*Plan résumé*





*Plan détaillé*

# 1.INTRODUCTION

## 1.1. LE PROBLEME POSE

Le travail que nous rapportons ici vise à évaluer la pertinence et la portée d'un ensemble de formalismes informatiques récents sur l'étude du patrimoine architectural, champ d'application dont on montrera les spécificités et leurs implications. Le problème que pose ce travail peut être présenté ainsi : comment un ensemble de solutions informatiques peuvent-elles contribuer à mieux étudier[1] l'édifice patrimonial, et comment ces formalismes peuvent en retour être évalués et questionnés ? Nous ferons référence à plusieurs problématiques transversales, la première portant sur l'élaboration de modèles et les suivantes sur l'élaboration d'applications[2].

Nous aurons comme objectif central la définition d'une méthodologie de description interdisciplinaire de l'édifice. Si notre volonté sera de formaliser un modèle de l'édifice, elle s'accompagnera de la volonté de développer des outils exploitant et validant (ou invalidant) ce modèle. Nous confronterons un formalisme de représentation des connaissances - l'approche objet - à notre domaine d'application particulier.
Nous devrons donc répondre à la question suivante : le formalisme objet est-il adapté à notre problème ? Plus précisément, cette question peut-être reformulée ainsi : le raisonnement par classifications que le formalisme objet autorise est-il pertinent pour décrire un modèle de l'édifice patrimonial; les concepts classifiés peuvent-ils être le maillon fédérateur d'une chaîne d'applications relevant de points de vues bien distinct sur l'édifice patrimonial ?
Nous justifierons ce choix dès à présent en relevant cinq points :

1. La réification est, dans le champ d'application qui nous intéresse, une opération qui à elle seule pose des questions difficiles, et lorsque [Ducournau et al, 1998, p 31] parlent de *réifier à bon escient*, on est forcé de reconnaître qu'en effet la question se pose. Nous ne l'éluderons pas, dût elle être la seule abordée.
2. Comment pourrions-nous nous interroger sur d'éventuelles alternatives en matière de représentation des connaissances quand nous ne savons ni si cette approche est pertinente ni même comment décrire la connaissance architecturale propre à l'édifice construit. Nous parlons ici non pas des multiples descriptions littéraires ou proprement architecturales, mais d'une description apte à faire l'objet d'une formalisation informatique opérationnelle. Il nous faut simplement reconnaître la nécessité de commencer quelque part.
3. Le travail que nous menons inclut une démarche de développement d'applications exploitant le modèle sous-jacent dans laquelle le formalisme choisi apporte des réponses concrètes efficaces, permettant de souligner les faiblesses possibles du modèle.
4. Le formalisme objet a fait l'objet d 'applications au sein de l'UMR MAP, laboratoire d'accueil de mon travail de thèse, sur le processus de production de bâtiment. Ce processus déterministe se situe par là même à l'opposé de ce qu'est l'étude du patrimoine construit. Rien ne permet dès lors d'affirmer que le formalisme objet est adapté à notre domaine, il est pourtant dans le cadre de l'UMR MAP important de statuer sur ce point.
5. Concluons enfin en citant [Ducournau et al, 1998, p5] : *« nous prendrons comme point de départ la technique d'implémentation. C'est là procéder au rebours de tout un mouvement de pensée, assez répandu, qui cherche à dégager des concepts et des*

[1] Donnons à ce mot le sens précis qui est le sien dans le champ de la conservation architecturale : rassembler, organiser et interpréter un jeu d'informations brutes pertinentes vis à vis de l'édifice étudié ; jeu d'information qui comprend en amont des données d'archives et en aval des observations (relevés, fouilles, inventaires).
[2] -Formalisation des connaissances sur l'édifice aux différentes échelles,
-Représentation tridimensionnelle et gestion de données qualitatives,
-Information à références spatiales à l'échelle architecturale s'appuyant le modèle,
-Relevé architectural dirigé.
-Développements d'outils sur la plate-forme Internet.

*formalismes supposés indépendants de l'implémentation »*. Nous pensons légitime de nous interroger sur comment décrire l'édifice patrimonial en prenant comme point de départ un formalisme de représentation des connaissances choisi.

Disons en résumé que, malgré l'éventail de problématiques abordées, nous posons en définitive une question peut-être très étroite puisque liée à un formalisme. Mais si nous parvenons à y répondre au moins partiellement, nous aurons tracé une voie possible que d'autres pourront évaluer comparativement.

## 1.2. LES DISCIPLINES EN JEU

L'édifice construit est à l'intersection de plusieurs disciplines, et impose de proposer un modèle des connaissances répondant à un faisceau d'usages ou d'angles d'analyse. Il ne s'agit pas ici de prendre en compte de façon exhaustive ces usages mais de mettre en œuvre un modèle architectural qui puisse répondre à un ensemble, même restreint, d'usages. Les points expérimentés sont:

- La mesure.
- La représentation graphique.
- La gestion de données non-graphiques.
- L'accès distant des outils développés pour Internet.

Le modèle proposée s'inscrit dans une démarche visant à répondre à des questions posées sur le champ patrimonial. Pourtant, il relève clairement d'un ensemble de thèmes liés aux formalismes informatiques expérimentés:

- *La représentation des connaissances*, la définition d'un modèle sur lequel s'appuie le raisonnement du chercheur est le cœur de cette démarche.
- *La visualisation* intervient dans toutes les étapes ou presque de l'étude de l'architecture urbaine et constitue l'outil privilégié de communication sur l'édifice.
- *La gestion de données sur le Web* est un vecteur d'échange de connaissances et d'expériences pour la communauté scientifique concernée qui est à la fois facile d'accès, portable et fortement évolutif.
- *L'information référencée spatialement à l'échelle de l'architecture urbaine*, une modélisation pertinente de l'objet étudié permet d'attacher à sa morphologie tridimensionnelle un ensemble de données et d'informations.
- *La simulation d'hypothèses de restitution d'édifices partiellement ou totalement détruits* est une activité importante dans la compréhension de la forme architecturale et urbaine mais souvent traité sans prendre en considération le problème du rapport des maquettes produites à un modèle théorique de(s) l'objet(s).
- *La collecte de données et d'informations, le relevé architectural*, sont des étapes souvent essentielles dans le processus de renseignement et de documentation des objets étudiés.

### 1.2.1. La représentation des connaissances

Nous faisons l'hypothèse qu'il est possible de définir un modèle interdisciplinaire de l'édifice, structuré par le biais de l'approche objet ; modèle apte d'une part à produire des visualisations et, d'autre part, à donner accès (y compris au travers des maquettes 3D produites) à un ensemble de données ou d'informations dans le contexte du réseau Internet. Ce modèle, bâti à partir d'une analyse architecturale et urbaine, et issu du travail de recherche autour de notre sujet entrepris dans le cadre du programme ARKIW à l'UMR MAP, s'appuie notamment sur deux hypothèses :

- L'univers de la connaissance architecturale est modélisable,
- L'approche objet fournit un formalisme adéquat à la représentation de cet univers.

L'approche objet est un formalisme de représentation des connaissances mais aussi une démarche méthodologique de programmation. Le *"paradigme de l'approche objet est essentiellement la solution de problèmes par recours à l'abstraction*" nous indique par exemple [Durnota, 1995]. "*L'objectif premier de la conception Objet* [au sens de la Programmation Orientée Objet] *est d'améliorer la productivité, la qualité et la maintenance de grosses applications logicielles"* indique pour sa part [Wu, 1996]. Le travail que nous présentons correspond bien à cette double utilisation puisque :

- la connaissance relative à un objet architectural est représentée par un ensemble d'éléments [du corpus] et de relations, objets informatiques, dans une structure hiérarchique.
- l'implémentation des interfaces Web, quel que soit l'outil concerné, s'appuie sur un ensemble de modules à forte réutilisabilité, objets informatiques organisés dans une structure hiérarchique.

### 1.2.2. Les outils de visualisation

L'étude du patrimoine bâti fait référence à un ensemble d'opérations menées à différentes échelles (urbaine, architecturale, corpus) qui chacune appelle un type de représentation adéquat au niveau d'abstraction considéré. La représentation pose donc un ensemble de problèmes liés au sens donné à la maquette produite et au résultat que l'on attend de son analyse. On s'intéressera en particulier aux liens sémantiques à établir entre l'objet représenté et un ensemble d'informations selon que cet objet est vu, par exemple, au travers des filtres de l'urbanité ou de la logique structurelle de l'édifice. On s'intéressera également à la notion de gestion des niveaux de détails qui doit permettre à l'objet d'être vu différemment selon que l'on en est proche ou loin. La production de représentations pose également des problèmes de codage et de charge sémantique de l'image largement traités dans la production graphique 2D traditionnelle mais qui n'ont pas d'équivalents sur les plates-formes de visualisation 3D. On verra là notamment les questions de figuration de niveaux de certitude dans les hypothèses de restitution ou encore de différentiation entre l'original et le reconstruit / réemployé. Cette adaptabilité de la maquette n'est possible que si celle-ci est produite à partir d'un travail de modélisation a priori des objets étudiés : la visualisation de l'édifice, quel que soit l'outil choisi, est dès lors une expression graphique du modèle architectural.

### 1.2.3. La gestion de données et les technologies du Web

La nécessité de documenter l'édifice est un souci essentiel dans notre travail, que la documentation concerne sa forme ou ses usages. Pourtant, la documentation de l'édifice n'est pas traditionnellement centrée sur les concepts architecturaux de notre modèle (objets physiques élémentaires). Il faut distinguer ici trois questions complémentaires :

- *Documentation des concepts / des instances* : le modèle proposé est une collection de concepts à lier aux sources documentaires qui la justifient, l'exploitation de ce modèle sur tel ou tel terrain d'expérimentation donne lieu à création d'instances du modèle qui à leur tour doivent être documentées.
- *Hétérogénéité des données et informations* : les sources documentaires à attacher à l'édifice sont diverses dans leur forme, leur origine, leur localisation et leur portée, un mécanisme d'URL permet de prendre en compte cette diversité et d'assurer une meilleure évolutivité au jeu d'informations référencées.
- *Navigation par la représentation tridimensionnelle* : la représentation de l'édifice peut être utilisée comme un mode d'accès aux sources documentaires recueillies au cours de l'étude de celui-ci. La scène VRML produite figure des instances du modèle, référencées spatialement, qui portent un mécanisme de requête permettant d'interroger un ensemble d'informations.

### 1.2.4. L'information référencée spatialement à l'échelle de l'architecture urbaine

Faisons l'hypothèse que la représentation de l'édifice peut être utilisée comme une interface de navigation dans un ensemble d'informations architecturales, comme la carte l'est dans le champ de la géographie. La question posée est dès lors la suivante : à quel concept spatial attacher les données que la maquette interface ? L'ensemble de concepts architecturaux modélisés selon le processus que nous décrirons peut naturellement jouer ce rôle. Une modélisation pertinente de l'objet étudié permet d'attacher à sa morphologie tridimensionnelle un ensemble de données et d'informations. La question abordée ici est donc celle d'un système d'informations référencées spatialement à l'échelle de l'architecture dans lequel la représentation tridimensionnelle de l'édifice peut servir d'interface privilégiée de navigation.

### 1.2.5. La simulation d'hypothèses de restitution

Par manque de rigueur dans l'utilisation de l'image, la restitution en images de synthèse d'édifices partiellement ou totalement détruits a acquis chez bien des spécialistes du patrimoine la réputation d'un exercice essentiellement commercial, inutile du point de vue scientifique, voire contre productif. Pourtant, la visualisation d'une hypothèse permet à son auteur de la vérifier puis dans un cycle d'essai-erreurs de la valider ou de l'invalider. Elle est donc un des outils au service de l'étude de l'édifice dont il ne faut pas sous-estimer le rôle. La production d'images figurant telle ou telle hypothèse de restitution est un travail long et fastidieux, même avec des logiciels de modélisation géométrique puissants. Il nous semble important d'élaborer un outil de modélisation qui permette de manipuler non plus de la géométrie mais des objets architecturaux dans le cadre de représentations peu détaillées permettant des vérifications rapides d'hypothèses. Cet outil, basé sur la plate-forme Internet (VRML), pose la question du rapport entre géométrie et morphologie architecturale. Là encore, seul un travail de modélisation des connaissances a priori permet de disposer de primitives architecturales paramétriques dédiées à la production de maquettes numériques 3D.

### 1.2.6. La collecte de données et d'informations, le relevé architectural

La conservation du patrimoine architectural est elle-même une discipline qui recouvre plusieurs spécialités. L'objet architectural est support de connaissances dont chacune engendre un jeu d'informations qu'un modèle de l'édifice se doit de prendre en compte.
Au-delà même de cette notion de gestion de points de vues sur l'édifice se pose le problème de l'évolution dans le temps de l'édifice lui-même, de la connaissance que l'on en a, du modèle enfin. Le travail présenté n'a pas ici d'ambition autre que celle d'évaluer la faisabilité de l'intégration de mécanismes de collecte de données, et ce tout particulièrement dans le cadre du relevé architectural. En effet, la contribution que nous défendons ici s'inscrit dans la continuité de travaux menés autour de la question du relevé architectural au laboratoire MAP-GAMSAU et y fera donc tout particulièrement référence.

## 1.3. DES INSTRUMENTS POUR ETUDIER L'EDIFICE

Notre contribution peut être résumée par trois aspects complémentaires : la définition d'un modèle de l'édifice construit, des applications dédiées à l'exploitation de ce modèle et enfin des terrains d'expérimentation sur lesquels ces instruments seront expérimentés.

### 1.3.1. Modèle architectural

Nous faisons l'hypothèse que l'édifice est un objet qui peut être décrit de façon non ambiguë, et qu'il est modélisable par le biais d'un ensemble de concepts et de relations s'appuyant sur le formalisme Objet. Le modèle aura comme seul objectif de représenter des connaissances relatives à l'édifice permettant d'appliquer un raisonnement d'expert humain. C'est dans les

modes d'exploitation de ce modèle que se fera jour l'idée de points de vue correspondant à diverses spécialités. La logique de division et de structuration de ce modèle correspond à une analyse finalisée de l'édifice, celle de l'architecte, posée en intermédiaire potentiel entre ces diverses spécialités.
Loin de se vouloir exhaustive, l'analyse proposée se veut plutôt pragmatique, proche du formalisme de représentation des connaissances choisi, et tournée vers la production d'instruments pouvant contribuer utilement à l'étude de l'édifice.

### 1.3.2. Applications

Les outils développés pour exploiter ce modèle ont pour point commun d'utiliser ce qu'il est convenu d'appeler "les technologies du Web". Concrètement, trois directions sont privilégiées :le relevé architectural, la gestion d'informations localisées spatialement et la visualisation. Les applications que nous avons développées ou expérimentées ont pour objectif d'appuyer les tâches auxquelles elles correspondent sur un travail mené a priori : la définition d'un modèle. L'apport de ce modèle dans le cadre d'applications dédiées à telle ou telle tâche particulière (mesure, restitutions, documentation, etc..) peut par conséquent être vu comme un indicateur de la pertinence du modèle, et en retour servir à l'améliorer.

### 1.3.3. Terrains d'expérimentation

***Des édifices :***
Les édifices ou groupes d'édifices choisis comme terrains d'expérimentation traduisent une des priorités que nous nous fixons : celle de prendre en compte les différents niveaux de lecture de l'édifice (détails, corpus, architecture urbaine, etc…). Ils doivent nous permettre d'évaluer sur des cas concrets l'aptitude du modèle proposé à représenter des particularités du domaine d'application, comme la notion de réemploi, d'incomplétude, etc…Ces terrains d'expérimentation sont situés dans le centre ancien de la ville de Cracovie (Pologne).

***Des objectifs :***
Les terrains d'expérimentation choisis doivent permettre de balayer l'ensemble des questions que pose ce travail : modèle, interfaces réseau, gestion de données, visualisation, etc…Les objectifs liés à ces terrains d'expérimentation sont divers:
- référencement bibliographique,
- simulation d'hypothèses de restitution,
- mesure photogrammétrique,
- maquette de montage-reconstitution des plafonds,
- reconstitution des évolutions architecturales et localisation ou inventorisation d'édifices.

Leur diversité correspond à la diversité des applications que nous souhaitons expérimenter. Elle correspond en définitive à notre démarche de modélisation : il nous semble important de procéder à une évaluation du modèle au travers d'un faisceau d'usages disjoints.

## 1.4. ORGANISATION DU DOCUMENT

Le présent document est organisé en six parties d'inégales longueurs. Dans cette section 1, nous avons situé le contexte de ce travail et les expériences que nous nous proposons de rapporter. Une courte section 2 intitulée *Généricité, portée et expérimentations* nous permettra de situer le niveau de généricité/réutilisabilité des développements proposés, l'apport attendu et le choix d'expériences concrètes à mener. Dans la sixième et dernière section nous tenterons d'établir une synthèse des apports de cette contribution et situerons les chantiers sur lesquels il sera nécessaire de travailler encore. Les sections 3 (*Etat de l'art*), 4 (*Méthodologie*) et 5 (*Implémentations et applications*) formeront le corps de ce document.

# 2.GÉNÉRICITÉ, PORTÉE ET CADRE EXPÉRIMENTAL

## 2.1. DU MODELE A L'OUTIL

Notre travail sera présenté dans la suite de ce mémoire au travers de deux aspects dont les liens pourtant essentiels nous seront peut-être difficile à établir :

- Une démarche de construction d'un modèle théorique, modèle que nous posons en pré-requis au développement d'outils. Nous considérons cette modélisation de connaissances architecturales comme nécessaire à toute évaluation qualitative ou quantitative de l'édifice. Cette démarche se veut générale, elle sera présentée dans ce sens.
- L'élaboration d'outils exploitant ce modèle, outils devant permettre d'évaluer le modèle et d'en vérifier la portée. Ces outils seront quand à eux décrits par le détail.

Nous allons en conséquence dans cette section commencer par cerner avec précision les liens à établir entre *une méthodologie d'analyse de l'édifice générique*, le cœur de notre travail, et *un ensemble d'outils d'évaluation et d'exploitation du modèle*¸ complément qui nous semble indispensable. Nous procèderons en donnant d'abord une réponse brève à six questions :

1. A quel problème notre contribution apporte t'elle une réponse méthodologique ?
2. Pour quels usages notre contribution introduit t'elle des développements concrets originaux ?
3. Dans quelle mesure la réponse méthodologique proposée est-elle générique ?
4. Quelle est la portée du modèle et de la partie de celui-ci implémentée ?
5. Quels sont les éléments de ce modèle effectivement mis en œuvre dans nos développements, et à quels autres éléments ces développements font-ils fait appel ?
6. Quel est le niveau de réutilisabilité de ce travail ?

Nous présenterons alors nos terrains d'expérimentation, en situant pour chacun d'eux les objectifs d'étude particuliers dont ils relèvent. Disons dès à présent que l'étude de chacun exploitera un même modèle, mais étendra si nécessaire le corpus que ce modèle fixe ou ne fera appel qu'à une sous-partie des comportements des éléments de corpus formalisés.

Devant balayer notre travail dans son entier, il nous semble utile de préciser dès à présent mon rôle particulier, bien que j'ai la conviction que cela soit clairement établi dans le corps du document. Nous donnons ici une liste des travaux dans lesquels je n'ai pas eu de rôle :

- Implémentation du module de mesurage photogrammétrique.
- Interfaces des trois outils développées en JAVA.
- Choix et documentation des terrains d'expérimentation.
- Etude et restitution de Kramy Bogate.
- Utilisation du logiciel MAYA pour la restitution de l'ancien hôtel de ville de Cracovie.

### 2.1.1. Analyse de l'édifice : une approche méthodologique globale.

Force est de constater que les analyses de l'édifice telles qu'elles se pratiquent aujourd'hui se font le plus souvent au travers des fourches caudines de leurs disciplines d'origine: la mesure produit une géométrie qui pourrait être celle d'un baril de lessive, la documentation référence des ouvrages sans relation avec la morphologie de l'édifice, la production de scènes tridimensionnelles produit des avatars souvent plaisants mais vides de sens, etc..
Nous pensons qu'il est nécessaire de formuler un modèle de l'édifice apte à intégrer ces points de vue pour recentrer sur l'édifice lui-même des données qui après tout servent à le caractériser. Nous nous proposons d'introduire une méthodologie globale d'analyse et de description de l'édifice permettant de fédérer autour de son *corpus*[3] un jeu d'informations aujourd'hui dispersées.

[3] Nous donnons au mot corpus pour signification celle de [Pérouse De Montclos, 1988] :un jeu d'objets physiques.

Notre travail s'appuie sur une analyse de sources documentaires en vertu d'un raisonnement par classifications rendu possible par le choix technologique opéré. Nous formulons une liste de règles génériques permettant d'isoler des éléments du corpus architectural significatifs d'une manière univoque. Ces concepts doivent alors servir de pivots autour desquels s'organisent des mises en exploitation du modèle. Le premier apport de notre travail sera donc l'élaboration d'une *méthode d'analyse globale* de l'architecture patrimoniale.
Nous souhaitons toutefois relativiser cette affirmation. Notre approche est générale, sa portée globale, dans le seul cadre de l'étude du corpus d'objets physiques constituant la forme de l'édifice. Ni l'architecture ni même l'édifice ne sauraient être décrits par ce seul biais. Notre modèle général ne l'est par conséquent que sur un angle d'analyse bien défini.

### 2.1.2. Des outils pour mieux comprendre l'édifice.

Si notre méthode d'analyse se veut générique, il n'en reste pas moins qu'elle doit être validée par un jeu de cas concrets dans lesquels se poseront des problèmes de réutilisation du modèle. Nous avons donc mené à bien un ensemble d'expériences se basant sur la méthodologie proposée, ensemble reflétant une grande diversité de problématiques liées à l'étude finalisée de l'édifice. Cette diversité d'études et d'outils doit démontrer le caractère générique de notre méthode d'analyse, et le caractère opérationnel de son implémentation. Enfin, second apport de notre travail, nous verrons ce que les outils que nous développons apportent en terme de lisibilité de l'édifice, ce en quoi ils permettent de mieux le comprendre.

### 2.1.3. Une approche méthodologique globale, un modèle en partie générique

Croire qu'un modèle de l'édifice peut recouvrir une réalité aussi peu régulière dans ses détails que l'architecture patrimoniale, c'est nier, de fait, le caractère *artisanal* de cette production, et tout simplement son *invention*. C'est donc pour nous prendre ses désirs pour des réalités. Cela condamne t'il notre effort ? Non, car dans l'édifice coexistent l'idée de régularités fonctionnelles, structurelles ou morphologiques et celle de dérivation de types. Il est par conséquent possible d'isoler un jeu de concepts communs, et d'en dériver un sous-type seulement lorsque cela s'avère nécessaire.
Dans chacune des expériences menées, le corpus générique de concepts architecturaux est étendu par la définition d'un jeu de concepts spécifiques à l'étude concernée. En effet, et c'est là un apport méthodologique de notre travail, nous pensons que si une structuration générique des connaissances liées à l'édifice patrimonial est possible, elle s'accompagne cependant d'éléments de variabilité irréductibles. Nous sommes par ailleurs en face d'édifices dont bien souvent nous ne connaissons la forme que partiellement, ou de façon ambiguë. Nous sommes en face d'une architecture du cousu-main, faite par des artisans, pour laquelle les propositions de classifications valides dans le contexte de la production industrielle de bâtiments sont inopérante. Comme la carte ne recouvre pas le territoire, un modèle architectural ne saurait recouvrir l'architecture construite. Il peut, au mieux, la rendre intelligible : voilà ce que nous nous proposons de faire.

### 2.1.4. Portée du modèle

Nous fixons à notre travail un cadre précis : nous étudierons le corpus architectural à l'échelle de l'édifice et de sa structure, avec pour premier objectif concret d'en faciliter la mesure en appuyant celle-ci sur un modèle des objets observés. Ce modèle définit des catégories sans lien avec la notion de style : les expériences que nous avons menées ici sur un corpus antique et là sur un corpus médiéval ne donnent pas conséquent pas lieu à la définition de modèles disjoints. Le modèle proposé suit en effet une logique de dérivation qui définit deux niveaux successifs de raffinement des concepts : dérivation structurelles et dérivation morphologiques. C'est à l'intérieur du second niveau de dérivation que l'on va si nécessaire étendre le corpus en fonction de tel ou tel cas. Les niveaux précédents sont eux génériques.

L'implémentation que nous avons mené à bien définit un ensembles de classes, le plus souvent abstraites, qui sont la partie générique du modèle. Elle définit par ailleurs un ensemble de classes, le plus souvent instanciables, qui sont la partie du modèle liée à une expérimentation particulière. Mais nous voulons lever une ambiguïté : formaliser un jeu de concepts relatifs à l'architecture antique c'est effectivement développer une partie du modèle liée à une expérimentation particulière. Cela ne veut pas dire que cette partie n'est pas réutilisable : elle l'est sur toute expérience relative à l'architecture antique, voire classique ou néo-classique. Il ne faut donc pas confondre non-généricité des concepts et non-réutilisabilité. Il ne nous semble pas pertinent de dire d'un élément du modèle partagé pour la seule architecture classique qu'il est générique. Pourtant il serait faux de croire qu'à chaque expérimentation il nous faudra réinventer la roue.

### 2.1.5. Eléments mis en œuvre

Dans le cadre fixé au point précédent, nous avons identifié quatre catégories de concepts architecturaux. Nous avons implémenté ces quatre catégories, chacune donnant lieu à la construction d'une hiérarchie plus ou moins étendue. A l'intérieur de chaque hiérarchie sont implémentés des concepts génériques et des concepts plus spécifiques, tels que définis plus haut. Au delà, nous avons aussi et surtout développé un ensemble de modules permettant d'exploiter le modèle dans le cadre des six outils que nous présenterons dans le corps de ce document. Ces modules ont en fait la même réalité concrète que les concepts architecturaux : un jeu de classes au sens de la POO. Pourtant nous les nommons différemment, traduisant par là une des ambiguïtés qui font selon nous une des richesses de l'approche objet : formalisme de représentation des connaissances mais aussi méthodologie de programmation.
Chaque concept est doté d'un jeu de procédures liées à des contextes d'exploitation du modèle ; c'est à dire en fait à des points de vue. Chaque module exploite du modèle le jeu de procédures pertinent, dans la logique du choix technologique opéré.

### 2.1.6. Réutilisations du modèle et des applications

Faisons le point de ce qui vient d'être dit, -une méthode d'analyse et de description générique -Une implémentation en partie générique - Des modules exploitant le modèle,
et regardons comment cela se traduit en terme de réutilisabilité.
La méthode proposée est à l'évidence réutilisable pour autant que l'on s'intéresse au corpus des objets physiques formant l'édifice, et ce à l'échelle qui correspond en gros à son relevé. Les concepts de têtes de hiérarchies sont réutilisables quelque soit le terrain d'expérimentation choisi. Les méthodes de sur-définition morphologique dont sont dotés les concepts architecturaux sont réutilisables puisque basés sur des concepts tiers. Les concepts isolés et implémentés pour un terrain d'expérimentation particulier ne sont réutilisables que dans un contexte d'étude comparable[4]. Les procédures dont chaque concept est doté sont liées à un mode d'exploitation particulier, mais sont réutilisables entre applications. Les applications elle-mêmes sont beaucoup trop embryonnaires pour prétendre à un quelconque degré de réutilisabilité. Pourtant les modules auxquels elles font appel sont eux réutilisables et d'ailleurs réutilisés au delà du travail que nous présentons.
Disons enfin que si la réutilisabilité des développements n'a pas été un de nos objectifs, celle-ci est néanmoins largement facilitée par le choix technologique opéré et que nous n'avons en la circonstance qu'exploité.
En résumé, la réutilisation de ce travail est limitée par :

- Le cadre que nous nous sommes fixé (notion d'échelle).
- La diversité et la variabilité de la forme architecturale.
- L'étroitesse des objectifs auxquels correspondent les outils que nous avons développé.

[4] Disons par exemple que le corpus développé pour l'architecture antique est en grande partie réutilisable pour les périodes classiques et néo-classique. Il est par contre pratiquement inutile aux périodes romanes voire baroque.

## 2.2. EXPERIMENTATIONS

Nous nous intéressons ici d'abord au contexte de nos expérimentations, le programme de coopération ARKIW établi entre l'UMR MAP et l'institut HaiKZ de Cracovie (Pologne). Nous détaillons ensuite chaque cas en examinant les objectifs poursuivis et les particularités architecturales des objets examinés. Nous conclurons enfin en signalant d'une part les questions soulevées sur le modèle et sur son implémentation et d'autre part les perspectives que ces travaux ouvrent.

### 2.2.1. Contexte de travail

ARKIW est un programme de coopération et d'échanges scientifiques entre le laboratoire MAP-GAMSAU UMR CNRS 694 et l'institut HAiKZ de la faculté d'architecture de Cracovie (Pologne). Le projet ARKIW a pour objectif le développement, sur le réseau Internet, d'instruments d'investigation scientifique sur le patrimoine bâti. Sa préoccupation centrale est donc l'intégration des méthodes informatiques et des problématiques patrimoniales, double ancrage auquel correspondent les spécialités respectives du laboratoire MAP-GAMSAU et de l'institut HAiKZ.

#### 2.2.1.i) Un programme de coopération pluridisciplinaire

Le laboratoire MAP-GAMSAU se charge à l'occasion de ce programme des questions relatives à la formalisation des connaissances patrimoniales, à la maîtrise des technologies mises en œuvre sur le réseau Internet et au relevé architectural.
L'institut HAiKZ est un des neufs instituts fédérés au sein de la faculté d'architecture de Cracovie (Pologne). Il est spécialisé dans l'enseignement et la recherche sur les domaines de l'histoire de l'architecture et de la conservation des édifices patrimoniaux. A l'occasion de ce programme, il se charge de l'extraction des connaissances relatives aux édifices traités, du choix de ces édifices et de l'expérimentation des outils développés.
Ce programme de coopération et d'échanges scientifiques a pu être entamé fin 1997 grâce à deux bourses TEMPUS, et a depuis été soutenu au travers d'un Programme d'Actions Intégré POLONIUM (MAE/CNRS/KBN). Au vu des résultats obtenus à ce jour, tant en terme de développements que d'apports à leurs problématiques de recherche respectives, les institutions partenaires ont décidé de poursuivre leur coopération.
Un des trois axes de recherche du programme scientifique du laboratoire MAP UMR CNRS 694 s'intitule *outils numériques et patrimoine architectural*. Le laboratoire peut au travers de cette collaboration bénéficier de compétences reconnues en terme de connaissance du patrimoine bâti, compétences qui viennent questionner avec acuité le travail d'élaboration d'outils et de méthodes que le laboratoire mène autour du patrimoine architectural.

#### 2.2.1.ii) Objectifs généraux

ARKIW était à l'origine centré sur l'étude de l'ancien hôtel de ville de Cracovie. Il se fixe pour objectif aujourd'hui, dans un esprit de généralisation de cette démarche, d'instrumenter, par un effort de modélisation des connaissances patrimoniales, la documentation des évolutions architecturales des édifices situés sur la place centrale de Cracovie. Notre démarche s'appuie sur l'élaboration d'un modèle architectural médiateur entre plusieurs vues sur l'édifice : objet renseigné (par la mesure, par l'histoire), objet représenté (en imagerie de synthèse, en réalité virtuelle), objet interfacé (navigation 3D donnant accès aux sources documentaires). Ce travail, à l'évidence fortement interdisciplinaire, a débouché sur un ensemble de maquettes opérationnelles. Ces résultats témoignent de l'interdépendance des problématiques de recherche en informatique et de leurs domaines d'application. La pertinence de solutions méthodologiques ou logicielles dépend en effet ici non seulement des qualités des interfaces ou de leur adaptabilité mais aussi de l'intégration des questions posées par le domaine d'application dès l'étape de formalisation du modèle, modèle sur lequel s'appuient à la fois le raisonnement du conservateur et les traitements informatiques (gestion

de données, simulations-visualisations, etc.). La complémentarité des équipes française et polonaise se lit au travers des caractéristiques des outils développés:

- Formalisation informatique d'un modèle architectural orienté objet s'appuyant sur l'analyse du corpus menée par une équipe de conservateurs du patrimoine.
- Interfaçage Web (permettant accès distant, indépendance des plates-formes matérielles, disponibilité et gratuité des plates-formes logicielles).
- Développements collaboratifs autorisant partage du modèle à distance et intervention sur celui-ci par chaque chercheur impliqué, aussi bien côté français que côté polonais.
- Modularisation des développements permettant de proposer dans le cadre de la formation doctorale polonaise une assistance à la recherche.
- Expérimentation d'un processus de relevé architectural informé, s'appuyant sur la technique photogrammétrique, sur le beffroi de l'ancien hôtel de ville de Cracovie.

Le travail effectué a été centré sur, d'une part, l'instrumentation informatique de la plate-forme et sur, d'autre part, l'analyse du corpus architectural et des sources notamment bibliographiques qui le sous tendent. Autour de la problématique initiale du projet, nous avons choisi de nous intéresser aux terrains d'expérimentation suivants, détaillés plus loin :

- L'ancien hôtel de ville de Cracovie.
- L'ancien marché aux draps (Kramy Bogate).
- Le corpus des plafonds en bois des maisons urbaines.
- Les édifices publics situés sur la place centrale (Rynek Główny).

### 2.2.2. Le cas de l'ancien hôtel de ville

#### 2.2.2.i) Les objectifs

Cet édifice, dont ne subsiste aujourd'hui que le beffroi, est étudié avec pour objectif d'en restituer les évolutions architecturales depuis sa fondation (XIVéme siècle) jusqu'à aujourd'hui. De nombreuses études architecturales ou archéologiques ont été publiées sur ce sujet depuis deux siècles et servent de références au travail de restitution engagé [Dudek et al, 1999b]. Le beffroi de cet édifice a également fait l'objet d'un relevé architectural dans le cadre de ce programme [Drap et al, 1999a]. Deux objectifs distincts ont été établis :

- D'une part, tester grandeur nature sur le cas du beffroi notre processus de mesurage.
- D'autre part, autoriser la création de maquettes numériques figurant diverses hypothèses de restitutions de l'édifice. Ces maquettes doivent s'appuyer sur le modèle sous-jacent et renvoyer aux sources bibliographiques et iconographiques en situant l'origine.

#### 2.2.2.ii) L'édifice

A la fondation de la ville probablement construit en bois, l'ancien hôtel de ville de Cracovie fut réédifié en briques et pierres à partir de 1383 sous le règne de Casimir le Grand. Il comprenait alors la tour que l'on peut encore voir aujourd'hui et, accolée à elle sur son mur Nord, une aile rectangulaire typiquement gothique, avec pente de toit forte et gâbles à arcatures aveugles. Une cour fermée bordait la partie Ouest de l'édifice et donnait accès aux sous-sol utilisés notamment comme prison. Ce premier édifice fut largement agrandi à la période renaissance, dans l'influence stylistique de Sukiennice.

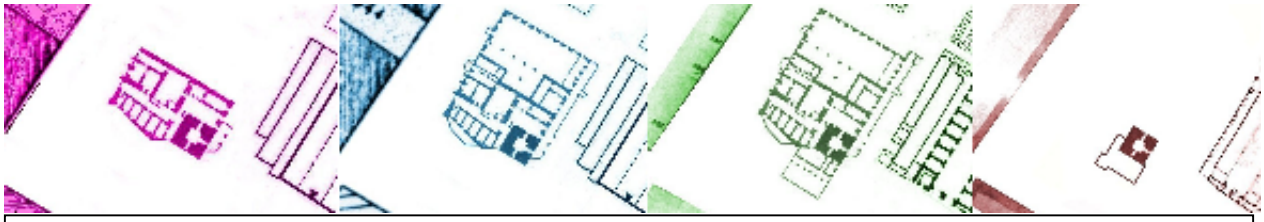


*Figure 1 : Vue en plan de l'ancien hôtel de ville de Cracovie, époques XIVème, XVIème, XVIIème et XIXéme siècles*

L'ancien hôtel de ville de Cracovie fut démoli entre 1817 et 1820 à une époque où le contexte économique local ne permettait pas de procéder aux travaux de restauration que l'état de l'édifice rendait nécessaires. Plusieurs éléments de corpus récupérés pendant la démolition furent réemployés dans d'autres bâtiments de la ville, notamment dans les locaux en transformation du Collegium Maius. Une petite extension néo-classique fut conservée, accolée aux murs ouest et sud de la tour, puis remplacée par une extension néo-gothique utilisée comme poste de police et détruite après la deuxième guerre mondiale.
Les oriels prolongeant en plan le premier étage de la tour sur ses faces est, sud et ouest ainsi que d'autres changements moins visibles furent apporté en 1962 dans la dernière phase importante de transformation du *Ratusz Krakowski*, menée par le Professeur W.Zin, intervenant toujours au sein de l'institut HAiKZ, notre partenaire dans le cadre du programme POLONIUM .

### 2.2.2.iii) Les expérimentations

#### *2.2.2.iii.1. La mesure*

Dans le cadre d'une expérimentation du processus de mesurage, nous avons choisi de nous intéresser aux Oriels du beffroi et à leurs consoles. Ces oriels sont en fait des reconstructions à l'identique d'éléments découverts pendant les fouilles de la campagne de restauration menée au début des années soixante. Ces éléments de corpus sont bien documentés et nous intéressent en particulier parce qu'ils posent le problème du relevé et de la représentation des profils moulurés, problème jusqu'alors pudiquement tu dans nos précédentes expérimentations sur la mesure. Nous avons donc effectué un relevé des consoles de l'oriel Sud dont nous rapporterons les principaux enseignements.

*Figure 2 :Ancien hôtel de ville, le beffroi, photographie des oriels depuis le niveau de la place*

Nous le verrons, l'entité architecturale, pivot de notre modèle, contient un ensemble de primitives géométriques, correspondant à des spécificités morphologiques des entités, évaluées par le mécanisme des EGO[5]. Une entité dont la morphologie comprendrait un profil fait de moulures successives pourrait être considérée comme un jeu de moulures indépendantes auxquelles des primitives géométriques correspondantes seraient rattachées (cercles ou plans dans le cas des consoles). Nous mesurerions alors des points à la surface de chaque moulure jusqu'à ce que l'ensemble du profil soit renseigné. Cette approche pourtant imposerait une durée de relevé très importante, et poserait des problèmes de reconnaissance de points en cas d'érosion du profil. Nous avons donc choisi dès l'étape de mesurage de prendre en compte une connaissance à priori du profil, principe déjà développé dans le cadre de la représentation des profils. Autrement dit nous avons là encore considéré qu'il fallait définit un modèle sur lequel appuyer la phase de mesure. Un mécanisme de mesurage spécifique aux profils moulurés a été développé, qui prévoit deux étapes :

- Relevé des points de contrôle des moulures correspondant aux points d'inversion de leurs courbes.
- Qualification des moulures en fonction d'une typologie donnée.

Les instances de la classe ProfilMoulure, participant à la définition des entités (les consoles dans notre expérimentation), sont chargées des mécanismes de mesurage. Là encore, nous mesurons des instances d'un modèle architectural : les objets présents dans la scène résultant du processus correspondent donc à une définition théorique d'objets architecturaux. Dès lors seront représentées dans l'espace défini par les points de contrôle du relevé photogrammétrique à la fois les entités mesurées et de nouvelles instances du modèle dont les

[5] mécanisme d'auto-evaluation en charge de retrouver dans un nuage de points les propriétés de la primitive.

caractéristiques dimensionnelles pourront s'appuyer sur les résultats de la mesure. C'est en fait un cycle mesure - visualisation/validation - instanciation de nouvelles entités qui a été expérimenté, cycle dans lequel seront ajoutées aux entités mesurées (en partie ou en totalité) des entités non mesurées.
Le relevé a été effectué en novembre 1998 à Cracovie avec un appareil numérique Kodak DCS260. Notre objectif n'était pas un relevé exhaustif du beffroi mais une expérimentation du processus de mesurage. Les contraintes de précision n'étaient donc pas prégnantes, nous avons utilisé un appareil non métrique et pris des paires de clichés à une distance d'environ vingt mètres du beffroi[6].
Le modèle photogrammétrique a été mis à l'échelle à partir de points de contrôle obtenus sur les géométraux établis pendant la campagne de restauration menée au début des années soixante. Chaque point mesuré a été identifié comme appartenant à une primitive géométrique propriété d'une entité architecturale en cours de mesurage.
Nous avons relevé un jeu de cinq consoles en mesurant :

- un jeu de points pour définir le profil,
- un jeu de points pour définir une boîte englobante de l'entité,
- un jeu de points pour définir le plan de projection du profil, et en établissant leur description formelle.

A la suite de la phase de calcul intégrant ces données, la position et l'orientation des consoles est établie ainsi que le plan de projection. Le profil est alors généré en fonction de sa définition formelle établie. Les instances du modèle étant générées, une seconde phase commence dans laquelle de nouvelles entités, purement théoriques cette fois, sont positionnées dans le même espace de la mesure. Dans cette expérimentation nous avons ainsi ajouté à la scène le sol de l'oriel que les consoles soutiennent. Après recompilation du script à la fois les éléments mesurés et les éléments théoriques sont visualisés dans la sortie VRML. Celle-ci est toujours liée au modèle photogrammétrique, il est donc possible de vérifier la validité de ces additions en mesurant de nouveaux points.

*2.2.2.iii.2. La représentation*

Deux types de travaux ont été menés dans le cadre de la simulation d'hypothèses de restitution de l'ancien hôtel de ville de Cracovie :

- Calcul d'images sur la plateforme MAYA pour la production de stéréogrammes, travail effectué par C.Radi dans le cadre de sa formation d'architecte.
- Elaboration de maquettes VRML servant d'interfaces à la base documentaire SOL[7] et à la production de stéréogrammes à l'échelle urbaine.

*Figure 3 : Reconstitution de l'ancien hôtel de ville sur la plateforme MAYA, vue sur l'angle Nord-Ouest*

Dans les deux cas, notre effort a porté sur deux phases de l'évolution de l'édifice, la phase dite gothique (1302-1565) et la phase dite renaissance (1565-1720). Pour chacune de ces phases nous avons étudié deux hypothèses divergentes et tenté d'illustrer à la fois leurs différences et le cas échéant leurs incohérences. Ces hypothèses, réputées comme les plus abouties, s'appuient sur le relevé de l'édifice effectué en 1802 par l'Autrichien Schmaus Von Livonegg.
Nous avons établi pour la génération de stéréogrammes un ratio de 2% entre la distance entre les points de visée et la distance au point visé.

*Figure 4 : Anaglyphe VRML incrusté dans une photographie, reconstitution de la place centrale de Cracovie à l'échelle urbaine (XVIIIème siècle)*

[6]Voir pour la description du processus de mesurage [Drap et al, 1999a] et [Drap et al, 2000].
[7] SOL est l'acronyme donné à un des outils que nous avons développé (voir section Applications)

Dans les maquettes VRML, un routage d'événement approprié se charge d'effectuer les opérations de rotation de la scène depuis le plug-in par le biais d'actions utilisateur[8].

Les stéréogrammes calculés en MAYA ou en VRML ont été utilisés dans les formes courantes (anaglyphes et filtres polarisants) et constituent, comme les représentations plus classiques, un outil apprécié des spécialistes de l'édifice.

Pour nous, l'utilisation privilégiée de la

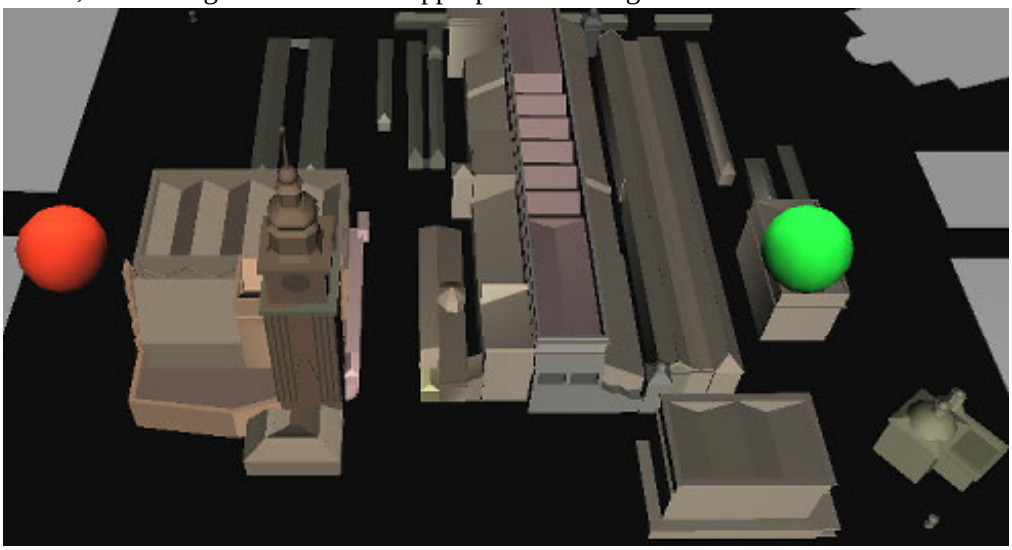


*Figure 5: Maquette VRML utilisée pour la capture d'images stéréos, les déclencheurs d'animation (mouvement de la scène par rapport à l'observateur) sont symbolisés par les deux sphères*

[8] Le principe en est le suivant : deux sphères sont utilisées comme déclencheurs (node TouchSensor) et positionnées autour du point d'observation initial (node Transform) :

```
DEF photoGauche Transform{
…
children [
DEF photoGauche_GoTouch TouchSensor{}
DEF photoGauche_GoTime TimeSensor{…}
Shape{…}
]}
###Une sphere verte fait tourner la scene de cinq degres sur
y positif
DEF photoDroite Transform{…}
]}
```

La scène est orientée à l'origine par un node Transform mais celui-ci contient des interpolateurs de position qui définissent le déplacement que doit subir la scène en rotation et en translation pour simuler la vue stéréoscopique :

```
DEF orientationScene Transform{
….
children [
…
DEF orientationScene_MovableRotate1
OrientationInterpolator{…}
DEF orientationScene_MovableRotate2
OrientationInterpolator{…}
DEF orientationScene_MovableTranslate1
PositionInterpolator{…}
]}
```

Enfin, le fichier contient les instructions de routage des évènements (Déclencheur -> Interpolateur -> Node visé ) :

```
##faire tourner a gauche
ROUTE photoGauche_GoTouch.touchTime TO
photoGauche_GoTime.set_startTime
ROUTE photoGauche_GoTime.fraction_changed TO
orientationScene_MovableRotate1.set_fraction
ROUTE photoGauche_GoTime.fraction_changed TO
orientationScene_MovableTranslate1.set_fraction
ROUTE orientationScene_MovableRotate1.value_changed TO
orientationScene.set_rotation
ROUTE orientationScene_MovableTranslate1.value_changed TO
orientationScene.set_translation
```

représentation tridimensionnelle de l'édifice reste la recherche de documents ou d'informations relatives à l'édifice au travers d'une maquette témoignant d'une analyse du corpus architectural en jeu.

*2.2.2.iii.3. La documentation*

Les maquettes figurant tel ou tel état de l'ancien hôtel de ville ont été utilisées comme interface de navigation dans la base documentaire SOL à des niveaux bien différents. En effet, seul le beffroi a été étudié en vue de recenser les documents relatifs à son corpus. Autrement dit, seul le beffroi renvoie à une requête spécifique à l'élément de corpus choisi par l'utilisateur : nous adaptons une scène à des utilisations distinctes en terme d'interface de requête pour qu'un même objet puisse référencer selon le cas l'édifice dans son entier ou un élément isolé de son corpus.
L'objectif de SOL est de capitaliser les lectures critiques de la bibliographie relatives aux édifices traités. Ces lectures sont faites à l'occasion de travaux de recherche approfondis sur un édifice, par exemple à l'occasion de thèses encadrées par l'institut HAiKZ. En l'absence de telle source, c'est aux données relatives à l'édifice dans son ensemble que vont faire référence par défaut les éléments de corpus représentés dans une maquette. Dans notre cas toutes les sources bibliographiques ou iconographiques ont été décrites comme traitant de tel ou tel édifice. Cependant, seuls l'ancien marché au drap et le beffroi de l'hôtel de ville ont fait l'objet d'une analyse architecturale approfondie permettant d'établir des références entre chaque objet du corpus et des sources documentaires. Les éléments de corpus représentés dans les hypothèses de restitution de l'ancien hôtel de ville renvoient donc, à l'exception de ceux du beffroi, à une requête générale, requête délivrant en retour l'ensemble des sources traitant de l'hôtel de ville. Rien n'empêche pourtant, puisque la maquette figure des instances du modèle architectural, de réinstancier les objets représentés et de leur affecter une nouvelle requête si notre connaissance sur l'édifice progresse.

## 2.2.3. Le Rynek Główny

### 2.2.3.i) Les objectifs

Les édifices publics situés sur la place centrale (Rynek Główny), existants ou disparus, sont concernés par le travail de localisation de ressources documentaires sur le Web proposés dans l'outil SOL [Dudek et al, 1999b].
La base documentaire gère une information localisée spatialement, elle est interfacée sur le Web pour la consultation comme pour la mise à jour. Les interfaces 3D (scènes VRML) ont ici pour vocation de rendre compte de l'analyse du corpus architectural sous jacente et de retracer l'évolution des édifices.
L'objectif poursuivi ici est donc de construire un ensemble de maquettes VRML de la place centrale qui permette l'interrogation de la base documentaire SOL à partir de deux critères imbriqués :

- La constitution de l'édifice en terme de corpus architectural.
- L'évolution de l'édifice dans le temps.


*Figure 6 : Leski N., Plan de localisation initiale de la ville de Cracovie, in [Jamka,1963]*

En traitant de l'édifice dans son milieu urbain, le travail entamé sur le *Rynek Główny* fait par ailleurs naître un certain nombre de problématiques jusqu'alors peu présentes dans notre projet, résumables par l'expression "gestion des échelles et niveaux de détail".

### 2.2.3.ii) Les édifices

Le dessin de la place centrale de Cracovie et de la trame urbaine qui l'entoure remonte à 1257, quand la ville etait encore capitale du royaume de Pologne. Son tracé est très exemplaire des cités fondées sous la charte de Magdebourg sous laquelle le roi Bolesław le Prude décida de placer Cracovie. L'orientation de la place centrale reprend l'ancien tracé des routes est-ouest et nord-sud qui s'y croisaient. Durant les sept siècles et demi d'histoire de cet ensemble urbain, de nombreux bâtiments d'interêt public ont occupé l'espace carré de la place dont les côtés mesurent environ 200 mètres (ce qui correspond a quatre blocs de parcellaire). N'en subsiste aujourd'hui que trois: le beffroi de l'ancien hôtel de ville (Wieża Ratuszowa), le marché aux draps (Sukiennice) et la petite église romane Saint Adalbert (kościoł św. Wojciech). Les bâtiments préservés comme les édifices disparus ont fait l'objet de nombreuses études de la part de conservateurs, d'architectes ou d'archéologues dont il nous a semblé utile de faire état dans le cadre d'un système d'information dont le principe est à rapprocher du paradigme des SIG : donner une localisation a un ensemble de données.
La place a compté jusqu'à dix-sept bâtiments ou groupes de bâtiments essentiellement dédiés au commerce, dont les positions nous sont données par les plans dressés au cours des dix-sept et dix-huitième siècles et dont la forme architecturale peut être simulée soit à partir de représentations d'époque soit par analogies avec des édifices de même type observés ailleurs en Pologne. Intervenant à l'échelle architecturale, nous avons souhaité ajouter la représentation tridimensionnelle des édifices comme moyen de retrouver les données afférentes à telle ou telle partie de l'édifice.

*Figure 7: Rynek główny aux XVI, XVII, XVIII, XIX (1802, 1834, 1874) émes siécles et état actuel.*

### 2.2.3.iii) Les expérimentations

#### *2.2.3.iii.1. La représentation*

Dans le cadre de la représentation des évolutions architecturales et urbaines du *Rynek Główny*, deux questions complémentaires sont posées :

- Figurer des états antérieurs d'édifices pour la plupart détruits (simulation d'hypothèses de restitution).
- Utiliser la représentation tridimensionnelle des édifices comme outil de navigation dans la base documentaire SOL, et ce en rapportant la scène à une période historique.

Deux périodes ont fait l'objet d'une représentation tridimensionnelle exhaustive : la place au XVIIIéme siècle (correspondant au nombre maximal de bâtiments implantés) et la place aujourd'hui. Les édifices ont été modélisés avec l'outil VALIDEUR[9] et renvoient selon le cas à des requêtes spécifiques à chaque éléments de corpus ou générale à l'édifice.
Chaque bâtiment figuré dans la première scène a fait l'objet d'une étude des

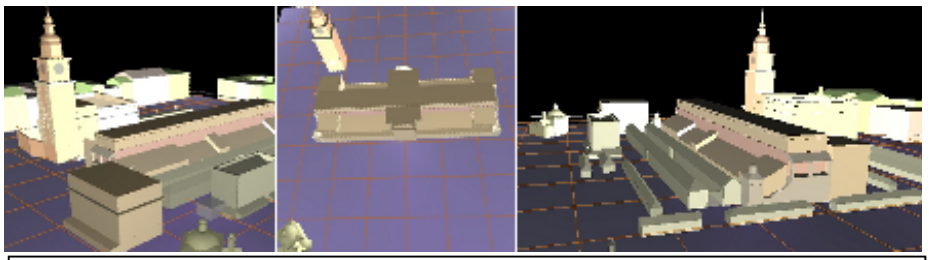


*Figure 8 : Les maquettes VRML utilisées dans la base documentaire SOL, période actuelle et période de plein développement (im auteur)*

[9] VALIDEUR est l'acronyme donné à un des outils que nous avons développé (voir section Applications)

sources bibliographiques et iconographiques citées dans l'outil SOL. Ceci a permis d'élaborer pour chacun d'eux une hypothèse de restitution ancrée dans une réalité documentaire.
Notre objectif ici s'est limité à la création de représentations tridimensionnelles des édifices à fins d'utilisation dans l'outil SOL : il n'a pas été fait de calcul d'images dites photo-réalistes. La création de ces maquettes à partir du jeu de concepts architecturaux que nous avons formalisé a permis de relever certains manques du modèle, comme par exemple la notion de réseau dédié, la gestion des niveaux de détail, etc…

*Figure 9 : Volets rabattants in [Viollet Le duc, 1996]*

Les représentations du modèle mises en œuvre dans le cadre de l'étude du *Rynek Główny* nous fournissent au-delà d'un résultat utile pour l'outil SOL un terrain d'évaluation du modèle particulièrement riche. En effet, la plupart des édifices traités ont été plusieurs fois transformés, à la fois en terme de filiation stylistique et en terme d'organisation physique des éléments. C'est donc la capacité du modèle à représenter le caractère particulier de l'architecture patrimoniale qui est évaluée. Nous considérons donc que le travail que nous avons entamé en matière d'étude et de restitution des édifices implantés sur le *Rynek Główny* est de nature à valider ou invalider notre démarche de définition et d'organisation des concepts architecturaux. La représentation permet par exemple de mettre le doigt sur deux types d'incohérences ou d'incomplétude du modèle :

– **Une définition inadéquate d'objets identifiés quant à leurs contextes d'utilisation.** C'est le cas en autres des fermetures de la galerie couverte de Kramy Bogate (marché aux draps). Une fermeture est l'attribut de la baie identifiant l'objet servant à clore la baie temporairement ou définitivement. Or dans le cas de cette galerie, les fermetures sont des volets de bois se rabattant horizontalement pour servir pendant la journée de surface d'exposition et de vente des commerces qu'abritait la galerie. La fermeture devient donc un concept lié à la surface de vente et par exemple susceptible d'être caractérisé par un taux de taxation. Bien sûr, l'impôt napoléonien sur les ouvertures est un exemple bien connu et souvent cité sur dans ce genre de cas. Mais la situation est ici différente puisque l'impôt napoléonien était calculé sur l'ouverture de la baie, et nous parlons ici de sa fermeture. Autrement dit, si le point de vue de notre taxinomie nous semble un bon médiateur entre divers points de vue sur les objets, sa mise en œuvre sur des cas concrets nous permet d'en mieux lister les manques.

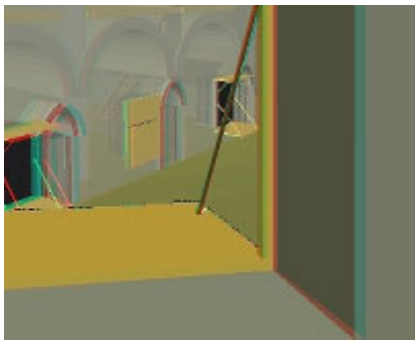


*Figure 10 : Restitution de Kramy Bogate en VRML, fermeture des baies sur la rue intérieure*

– **Une surdéfinition d'instances** (i.e une surabondance de leurs attributs) **ne correspondant pas à l'état des connaissances sur l'objet étudié.** De nombreux édifices peu documentés sont présents à telle ou telle époque sur le *Rynek Główny*. Comment dès lors instancier des objets complètement définis ? Pourquoi figurer dans la représentation de l'édifice un objet complètement défini alors que rien ne permet de justifier le choix effectué ? Pourquoi de plus lui affecter un code graphique identique quel que soit l'état des connaissances sur l'objet (c'est à dire une représentation tridimensionnelle qui est celle de toutes les instances issues d'une classe) ? Ces questions peuvent sembler anecdotique dans bien des disciplines, elles sont pourtant au cœur des préoccupations que les conservateurs évoquent quand ils font face à un travail de restitution numérique d'édifice patrimonial. Nous n'avons là encore

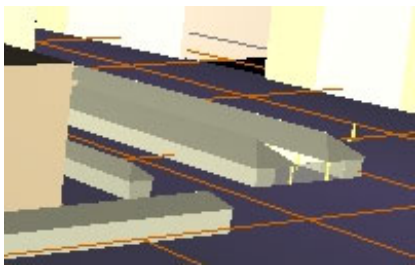


*Figure 11 : Restitution des jatki olejne, powroźnicze, krupnicze,owsiane, śledziowe en VRML, incertitude sur la définition du portail symbolisée par un effet de transparence*

que peu de réponses, bien que ce point fasse partie de nos développements en cours. L'outil VALIDEUR ne permet pas de représenter sous des morphologies différentes des instances du modèle puisque les classes ne disposent que d'une méthode de représentation générale. L'outil permet néanmoins de jouer sur la texturation de l'objet ou sur sa propriété nSeg[10] pour signifier un degré d'incertitude quant à la définition réelle de l'objet représenté. La problématique du codage de la représentation, naturelle chez les géographes et chez de nombreux architectes, nous semble un terrain de recherche particulièrement important dans le champ d'application qui est le nôtre mais aussi intéressant dans la discipline dont relève ce mémoire de thèse. Pourtant elle dépasse largement le cadre de notre travail, nous nous contenterons de cette simple citation.

Le travail d'élaboration de maquettes, fastidieux à bien des égards, joue pourtant selon nous un rôle non anecdotique : il s'inscrit dans cette démarche par essais-erreurs typique de l'activité de l'architecte que décrit [Lebahar,1983]. Il interroge la pertinence du modèle et en retour la rigueur de l'hypothèse. Nous pensons donc utile de nous appuyer sur une évaluation régulière du modèle par la simulation d'hypothèses de restitution.

*2.2.3.iii.2. La documentation*

Certains documents relatifs aux édifices implantés à telle ou telle époque sur le *Rynek Główny* sont des témoignages d'époque, autrement dit des observations: plans ou métrés anciens, textes descriptifs, illustrations, relevés , etc… D'autres sont au contraire des travaux de recherche menés a posteriori dans lesquels les auteurs analysent les premiers. Leur diversité nous a conduit à distinguer la référence (ce que le SGBD gère) et la citation (les pages URL additionnelles). On peut rapidement rappeler ou situer les grandes familles de documents considérés en dressant la liste indicative qui suit :

- Plans d'ensemble aux époques successives.
- Géométraux d'époque sur certains édifices.
- Illustrations (croquis, peintures, gravures, etc…)
- Relevés de détails d'époque (écussons, voûtements, etc..).
- Photographies (dès la seconde moitié du XIXème siècle).
- Descriptifs quantitatifs des interventions sur certains édifices.
- Dossiers complets de restauration sur certains édifices.
- Etudes archéologiques et compte-rendu de fouilles
- Projets d'interventions architecturales ou urbaines (depuis la première moitié du XIXème siècle).
- Extraits de cadastre actuels.
- Etudes historico-architecturales sur certains édifices (depuis la première moitié du XIXème siècle).
- Etudes urbaines sur la trame urbaine du centre historique.
- Hypothèses de restitution sur certains édifices (depuis la seconde moitié du XIXème siècle).
- Publications diverses dans les domaines de l'histoire et de la conservation des édifices.

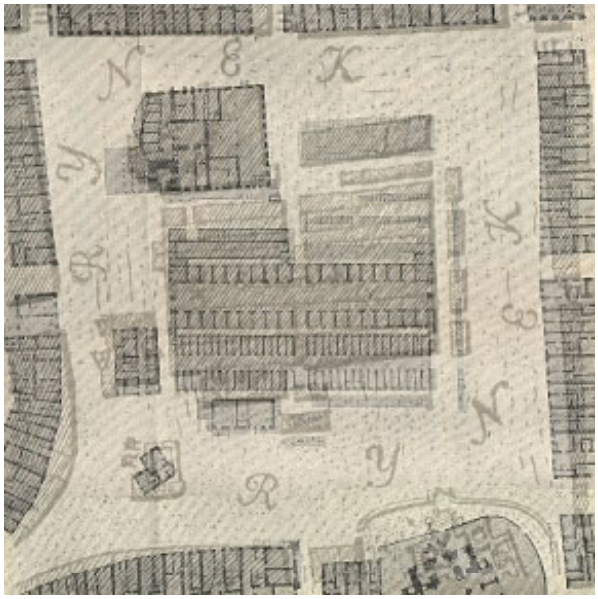


*Figure 12 : Superposition des plans Kołłątajowski [Odlanicki, 1978] et Pucka [Tomkowicz, 1907]*

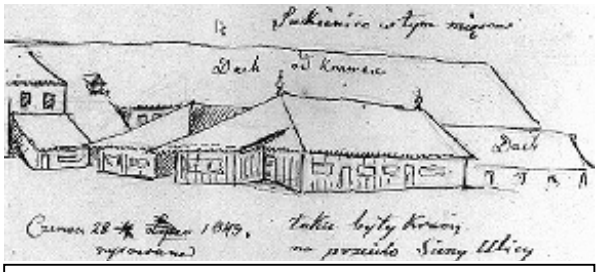


*Figure 13 : Illustration datée de 1848 (Kramy Bogate et bâtiments attenants)*

[10] propriété définissant le nombre de segments a afficher pour représenter une courbe

Un travail sur les modes d'accès à cette documentation fait partie des préoccupations affichées par ceux qui au jour le jour s'investissent dans la pratique ou l'enseignement de la conservation. En cela, nous avons avec SOL souhaité expérimenter une approche de la gestion d'information centrée sur l'objet étudié : l'édifice.

### 2.2.4. Kramy Bogate

#### 2.2.4.i) Les objectifs

L'ancien marché aux draps (Kramy Bogate), regroupant quatre bâtiments le long d'une allée centrale couverte, a été détruit dans la deuxième moitié du XIXéme siècle. Il fait l'objet d'une étude visant à en restituer les états successifs. Peu de références sont ici disponibles pour engager le travail de restitution, notre premier objectif était donc sur cette expérience de proposer un outil d'élaboration d'hypothèses intervenant dans l'étape de validation d'une reconstitution [Drap et al, 1999b], [Dudek et al, 1999a].
Des représentations de l'édifice ont été proposées à l'échelle de l'architecture urbaine dans le cadre de l'outil SOL. Dans ce cas le structure interne du bâtiment n'est pas représentée. D'autres représentations, plus analytiques, et cette fois à l'échelle de l'édifice, ont été construites sur l'outil VALIDEUR dans le cadre d'un travail de doctorat de l'institut HAiKZ[11].

#### 2.2.4.ii) L'édifice

Kramy Bogate etait un marché aux draps construit parallèlement à Sukiennice et modifié au cours de sa période de présence sur la place[12]. Son implémentation sur la place est attestée dès 1358, date correspondant au règne du "roi bâtisseur" Kazimierz Wielki. L'édifice était utilisé pour la vente de tissus et d'habits. Il est possible qu'un édifice en bois ait été présent avant 1358. Le bâtiment était caractérisé par deux rangées parallèles d'échoppes s'ouvrant sur la rue intérieure qu'elles formaient.

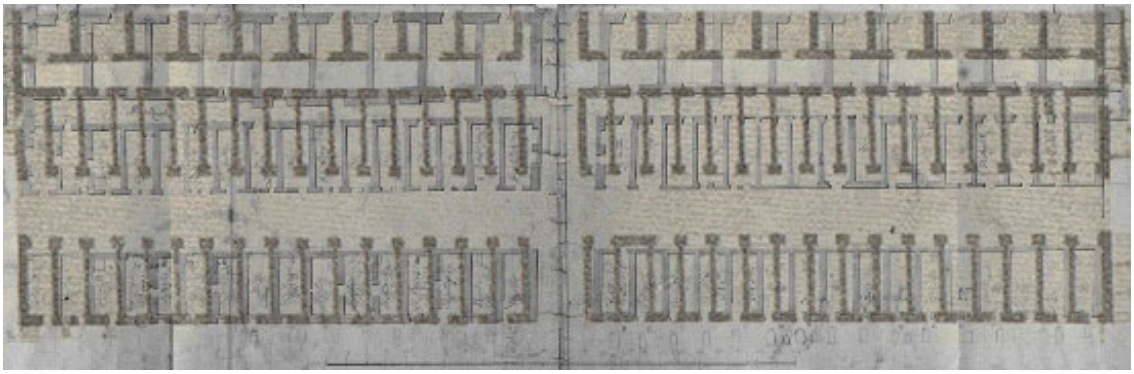


*Figure 14 : Kramy Bogate, plan de l'édifice*

Cette rue intérieure, découverte à l'origine, servait au stockage et au déchargement des marchandises : elle était close à ses deux extrémités par deux portails fermés pendant la nuit. Les échoppes comprenaient deux étages, la pièce de vente en rez-de-chaussée et un grenier - réserve à l'étage. L'édifice ne comprenait au départ pas de sous-sol.
Pendant la période gothique, le niveau de la place va être régulièrement rehaussé par accumulation de gravats et déchets divers. Pendant les premières décennies, quelques marches descendantes placées devant l'accès au niveau du sol suffisaient à rattraper la différence de niveau. Dès le seizième siècle pourtant, la pièce

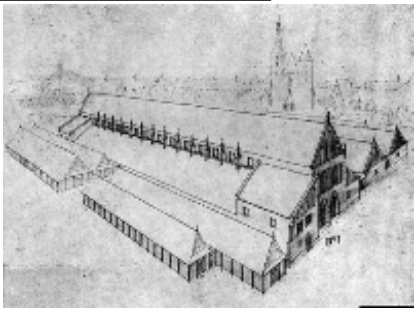


*Figure 15 : Sukiennice et Kramy Bogate, XVème siècle,*

[11] Un des objectifs pour nous est ici de réagir aux observations relatives aux outils développés, outils que nous souhaitons interdisciplinaires, que formulent les spécialistes du domaine d'application dont nous prétendons traiter.
[12] notamment en raison du rehaussement global du niveau du Rynek et du percement d'une allée transversale reliant parties Est et ouest de la place à travers Sukiennice et Kramy Bogate.

inférieure sera réutilisée comme cave et la pièce de vente transférée à l'étage. En parallèle, un nouvel étage est ajouté pour garder une surface de stockage en élévation et la couverture de toutes les échoppes est rehaussée. Vers le milieu du seizième siècle, une allée est percée perpendiculairement à l'édifice, en son centre, afin de relier les parties Est et Ouest de la place. Cette allée utilise la surface de deux échoppes. L'exhaussement continu du niveau de la place rend bientôt l'accès au niveau inférieur, devenu cave, impossible depuis la rue intérieure (surface occupée par l'escalier d'accès trop importante). L'entrée ne peut même plus se faire par une porte sous linteau traditionnelle, une lunette doit être percée cette fois dans le mur extérieur et la fermeture doit être dégagée du mur sous la forme d'une trappe s'ouvrant en oblique. Par la suite, les changements observés sont minimes, principalement des jonctions de plusieurs échoppes par percements des refends. La dégradation de la situation économique et politique en Pologne au dix-neuvième siècle entraîne peu à peu le départ d'un grand nombre de propriétaires d'échoppes. Le bâtiment sera finalement détruit en raison de son mauvais état à la suite d'un écroulement partiel de son toit en février 1867. Seule la partie alors en élévation sera effectivement détruite : les sous-sols restent aujourd'hui enfouis et pratiquement non fouillés. Les grandes étapes[13] de l'évolution de l'édifice ont été étudiées et représentées à la fois en terme de volumétrie urbaine et en terme d'organisation des échoppes.

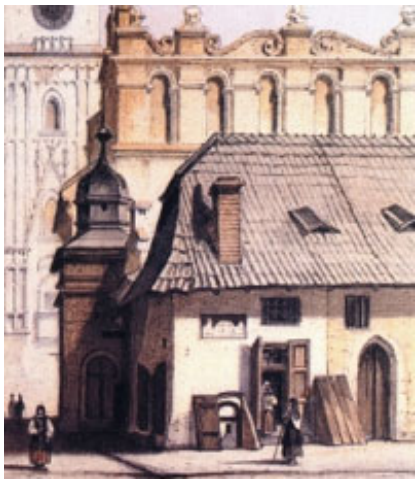


*Figure 16 : Extrémité Sud de Kamy Bogate, on note la trappe s'ouvrant en oblique pour desservir le sous-sol*

### 2.2.4.iii) Les expérimentations

#### *2.2.4.iii.1. La représentation*

Les outils HUBLOT[14] et VALIDEUR ont successivement été utilisés dans la mise en œuvre des représentations du bâtiment à l'échelle de l'architecture urbaine. Ces représentations ont dans le second cas été intégrées à l'outil SOL. A cette échelle, seuls les éléments de corpus visibles depuis l'extérieur ont été représentés. Les hypothèses de restitution figurées ont été élaborées par I.Dudek à l'occasion de son doctorat à l'institut HAiKZ à partir des documents recensés dans l'outil SOL.

Aucune hypothèse de restitution n'avait été élaborée jusqu'alors, justifiant l'intérêt porté à cet édifice et à sa représentation par nos partenaires polonais. Plusieurs phases du développement

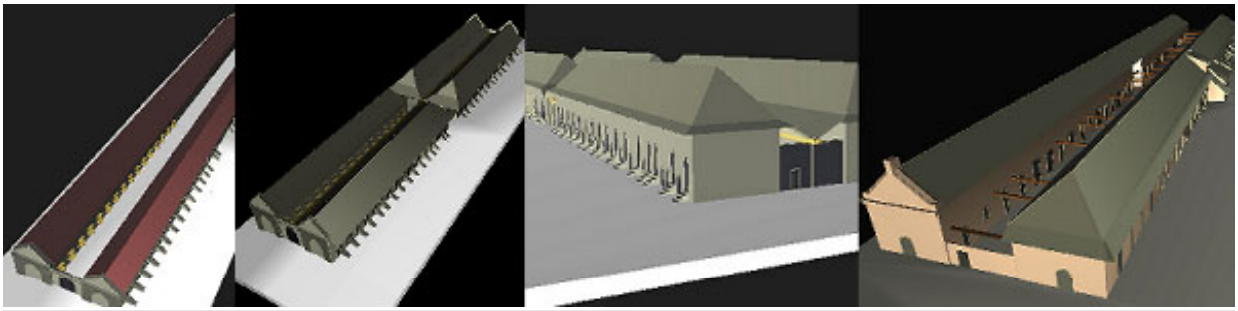


*Figure 17: Restitution de Kramy Bogate à l'échelle urbaine, quatre phases*

[13] Quatorzième - quinzième siècles : L'édifice de style gothique comporte deux bâtiments parallèles séparés par une galerie non couverte fermée aux deux extrémités par de hautes portes. L'édifice de style gothique n'est pas traversant, ses murs pignons sont typiques de l'époque (gâbles à redents).
Milieu du seizième siècle : Une allée perpendiculaire à la grande dimension des deux bâtiments est percée en leur centre.
Milieu et fin du seizième siècle : Le bâtiment est globalement rehaussé : Les couvertures sont modifiées, la rue intérieure est couverte, la surface de vente transférée à ce qui était à l'origine l'étage mais est alors de plain-pied.
Première moitié du dix-neuvième siècle : Dégradations successives du bâtiment (en particulier dépose de la couverture de la rue intérieure) et ajouts ponctuels (rehaussements de toitures, etc..)
De 1867 à 1869 : destruction des parties en élévation.

[14] HUBLOT est le nom donné à un des outils que nous avons développé (voir section Applications)

de l'édifice ont été traitées, dont les plus importantes sont les suivantes :

- L'édifice de style gothique est caractérisé par deux bâtiments parallèles séparés par une galerie non couverte fermée aux deux extrémités par de hautes portes. La galerie est utilisée pour décharger les marchandises.
- L'édifice comporte des murs pignons à gâbles, les échoppes sont accessibles depuis l'extérieur par des escaliers individuels, le bâtiment devient traversant.
- L'édifice est transformé : un seul mur pignon reste à gâbles, les trois autres deviennent à croupes. Les échoppes restent accessibles depuis l'extérieur par des escaliers individuels, la vente se fait au rez de chaussée (compte tenu de l'exhaussement de la place), la galerie est couverte.
- XIXéme siècle - Le bâtiment se délabre rapidement, la couverture de la galerie s'écroule.

A l'échelle de l'édifice, le travail a porté sur l'organisation des espaces intérieurs en fonction des évolutions du bâtiment lui-même et du niveau de la place. Il faut à ce propos situer l'étendue du problème : entre la fin du moyen âge et le début du vingtième siècle, le niveau de la place a été relevé de trois à cinq mètres selon les zones considérées. Autrement dit, chaque édifice a "perdu" un à deux étages, les sous-sols actuels des bâtiments étant en réalité des plain-pied ensevelis. Sur le cas de Kramy Bogate, l'accès aux caves a dû être fait par l'extérieur en ajoutant des trappes. Une étude en détail a été menée sur un des 64 *kram* (échoppes) que comptait l'édifice, étude reprenant les étapes déjà évoquées ci-dessus.

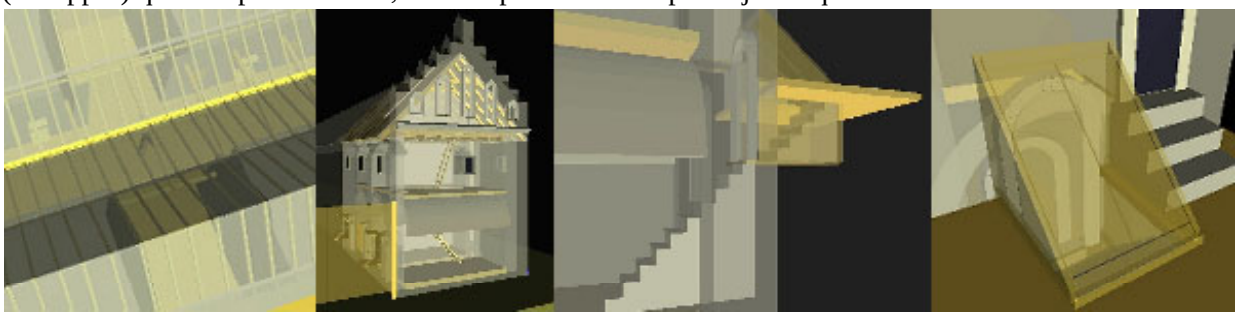


*Figure 18: Restitution de Kramy Bogate à l'échelle architecturale, étude en détail du passage transversal et du Kram 64*

La différence de traitement entre ces représentations témoigne de leurs objectifs divergents : dans le premier cas la maquette est un moyen d'interroger la base documentaire et figure uniquement la logique d'organisation de l'édifice dans la place, dans le second la représentation est un outil utilisé à fins d'analyse d'une hypothèse. Dans ces deux cas elle résulte d'un choix de modélisation et n'est pas vue comme une fin en soi.

### *2.2.4.iii.2. La documentation*

Le travail d'élaboration d'hypothèses de restitution s'est fait ici à partir d'un nombre restreint de sources, les résultats de ce travail avaient donc d'autant plus vocation à être interprétatifs, position que notre contribution défend. Ces sources se contredisent par ailleurs quelquefois les unes les autres, par exemple dans le cas illustré ci-contre de copies d'une même illustration difficiles à dater. L'interprétation de ce document précis est pourtant décisive puisqu'il s'agit de la seule illustration permettant d'établir quel type de couverture était en place sur l'allée percée au seizième siècle. Les documents utilisés dans le cas de Kramy Bogate sont en gros des même types que ceux déjà évoqués pour le cas du *Rynek Główny*.

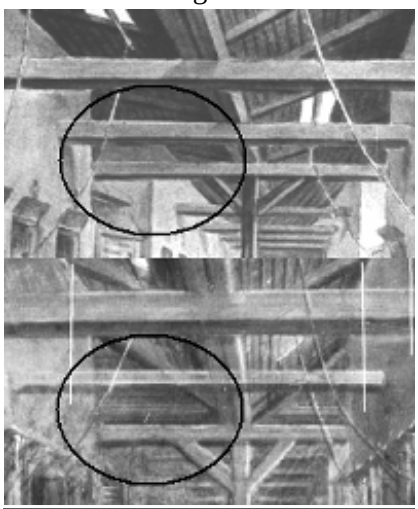


*Figure 19 : Vue sur la rue intérieure de Kramy Bogate au XVIème siècle. Image originale et copie .*

Deux points sont cependant à noter :

- Les observations sur l'édifice s'arrêtent en 1869. Plusieurs photographies antérieures montrent l'édifice mais aucune fouille n'a été entreprise depuis.

– Contrairement au cas de l'ancien hôtel de ville, pratiquement aucun travail de compilation d'informations et d'élaboration d'hypothèses n'a été mené sur Kramy Bogate.

Les documents relatifs à l'édifice utilisés dans la phase d'élaboration des hypothèses de simulation sont indexés dans le cadre de la base documentaire SOL. Par ailleurs, de nombreuses URL additionnelles référencées dans ce même cadre renvoient aux analyses et aux choix faits dans la construction des hypothèses. Le cas de Kramy Bogate doit donc nous servir dans la phase de travail que nous entamons à tester une structuration XML des documents additionnels.

### 2.2.5. Les plafonds en bois

#### 2.2.5.i) Les objectifs

Le corpus des plafonds en bois des maisons urbaines de la vieille ville est le sujet d'une expérimentation visant à offrir aux enseignants chercheurs de l'institut HAiKZ un outil de représentation et de connaissance de ce corpus disponible sur Internet [Dudek et al, 1999a]. Il s'agit donc d'un travail d'analyse du corpus en jeu et de formalisation d'un modèle le représentant. Trois questions principales sont abordées :

– Simulation d'hypothèses d'agencements de plafonds à partir de données partielles (issues de fouilles ou d'observations).
– Présentation pédagogique des principes d'agencements des plafonds.
– Définition d'un formalisme de gestion des profils.

*Figure 20 : Un plafond en bois restauré, on reconnaît solives, murières, entretoises, plancher.*

Le cas des plafonds en bois apporte des perspectives de travail inédites pour nous puisque le modèle est ici utilisé à fins d'explicitation théorique d'un corpus particulier, et devrait à terme servir également à assister le spécialiste dans son travail de reconstitution physique d'un plafond partiellement ou entièrement détruit.

#### 2.2.5.ii) Le corpus

La ville compte plus de trois cents plafonds recensés dont la mise en valeur est une priorité dans l'action de conservation. La demande qui nous a été adressée dans un premier temps consiste à faire connaître ce patrimoine et d'autre part mettre en place au-delà de l'outil de modélisation un système d'information sur ce corpus.

Nous parlerons du corpus des anciens plafonds en bois en illustration à notre définition des entités architecturales (section "***Définition du modèle architectural***"). Disons ici simplement que les éléments principaux d'un plafond sont une structure porteuse (solives et entretoises) ancrée sur des murs et portant un plancher plus ou moins complexe.

Des typologies existantes ont catalogué chacun de ces trois groupes d'éléments. A l'intérieur de ces ensembles fonctionnels, les entités architecturales sont caractérisées par des spécificités morphologiques importantes du point de vue de l'analyse historique et plus ou moins indépendantes les unes des autres : mouluration des profils, terminaisons des profils, typologie des montages, présence ou non de murières, etc… Au-delà de la représentation des éléments du corpus se posait donc ici le problème d'une représentation autonome (en terme d'apparence à l'affichage) des éléments constitutifs des entités. Si l'entité reste l'objet propriétaire de la méthode de visualisation, ses composants devaient dans le cadre du corpus des anciens plafonds en bois être identifiables isolément. Nous allons ci-dessous détailler ce point.

#### 2.2.5.iii) Les expérimentations

##### *2.2.5.iii.1. La représentation*

Le corpus des anciens plafonds en bois a été étudié et décrit avec exhaustivité et précision, nous permettant de proposer à travers l'outil VALIDEUR une plate-forme de visualisation

didactique. Dans l'implémentation précédente, accessible par l'outil HUBLOT, l'entité seule disposait d'attributs fixant l'apparence de l'objet dans la scène VRML. A partir de ce travail nous avions dans un premier temps tenté de rendre explicite l'analyse du corpus en proposant des animations VRML (routage d'évènements pour isoler tel ou tel objet par modification de sa position). Toutefois, cette solution avait deux inconvénients :

- Les animations devaient être prédéfinies, l'interaction avec l'utilisateur se limitant aux scenarii prévus par l'auteur de la scène.
- Il n'était pas possible de rendre compte de l'analyse intra-entité du modèle : les attributs des objets considérés n'étaient pas autonomes en terme d'apparence dans la visualisation.

A partir de ce constat, nous avons choisi de nous orienter vers une visualisation des scènes à l'intérieur de l'outil VALIDEUR. Ce faisant, chaque objet modélisé est une instance du modèle, modifiable côté client attribut par attribut. Par ailleurs, le modèle inclut des attributs permettant de gérer indépendamment l'apparence de l'entité et celle de ses attributs. Sont donc donnés pour modification éventuelle côté client :

- La localisation de l'entité dans la scène.
- Son apparence.
- L'apparence de ses attributs significatifs du point de vue de l'analyse historique.

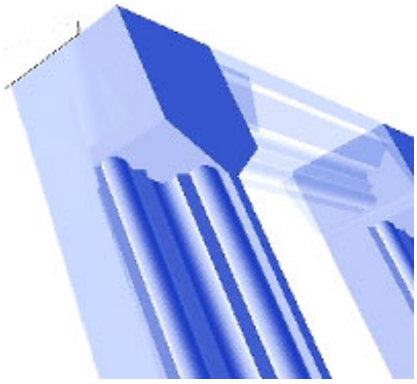


*Figure 21 : Restitution VRML d'un ensemble solives -entretoise (traitée en transparence)*

Nous pensons que cette réponse est assez appropriée au problème posé puisqu'elle rend compte d'une façon exhaustive des spécificités de ce corpus sans être incohérente avec la logique de classification de notre modèle.

Les maquettes produites aujourd'hui sont relatives à des plafonds théoriques en terme de dimensions globales mais correspondant à des observations rapportées par [Tajchman, 1989] en terme de proportions

*Figure 22 : Restitution VRML d'éléments de plafonds (Kraków XVème et XVIéme siècles, voir [Tajchman, 1989, pp79-82])*

des profils, de type de montage et de type de terminaisons. Des cas concrets (plafonds relevés dans les maisons urbaines du centre historique de Cracovie) doivent être à leur tour traités dans un avenir proche. Le corpus des plafonds en bois nous intéresse particulièrement car il renvoie à des échelles dont nos autres terrains d'expérimentation ne relèvent que peu, celles du décor ou des liaisons physiques entre objets par exemple. Il s'agit donc d'un terrain d'expérimentation très complémentaire de ceux que nous avons cités jusqu'à présent.

Nous croyons que chacun de ces terrains d'expérimentation est susceptible d'apporter dans l'élaboration de notre modèle un ensemble de questions permettant d'en mieux évaluer la pertinence au regard de la pratique de la conservation des édifices.

# 3.ETAT DE L'ART.

## 3.1. CHOIX ET ORGANISATION DES SUJETS TRAITES

Nous commencerons dans cette section par situer rapidement les références sur lesquelles nous nous sommes appuyés pour analyser notre domaine d'application. Mais c'est dans notre **annexe 1** que nous détaillerons cette analyse. En effet, nous n'avons pas eu l'ambition de rassembler un état de l'art sur l'architecture patrimoniale mais simplement de faire une lecture finalisée de ressources documentaires afin de justifier notre description du corpus architectural. Il faut néanmoins souligner que cette analyse constitue selon nous en elle-même un des apports importants de notre travail. Fixer des règles de relecture et d'interprétation de la documentation architecturale à fins d'exploitation par un formalisme informatique ne nous semble pas trivial. Nous concevons néanmoins que cet apport là soit trop proche du domaine d'application pour figurer dans le corps de ce mémoire.
La plus grande partie de cette section sera consacrée au formalisme de représentation des connaissances choisi; l'approche objet[15].

D'autres domaines, tenant cette fois non plus à la formalisation d'un modèle mais à son exploitation; devront être abordés pour replacer les propositions que nous faisons dans leur contexte :

- *Sur les problématiques de la conservation : Relevé architectural, représentation, gestion de données historiques et bibliographiques, reconstruction.*
- *Sur le contexte technique des développements que nous proposons : Systèmes de gestion de bases de données, informatique graphique et formats 3D pour le Web, interfaces pour le réseau Internet.*

Nous renverrons pour l'essentiel en annexe le contenu de ces sous-sections afin de ne pas surcharger notre état de l'art, mais nous souhaitons cependant signifier clairement le rôle important qu'ont tenu dans notre travail les problèmes de développement d'outils et leur rapport au modèle sous-jacent.
Notons enfin que chaque question abordée sera accompagnée par un ou plusieurs encarts, placés en annexe, signalant un développement typique ou une application dans le domaine de l'architecture patrimoniale.

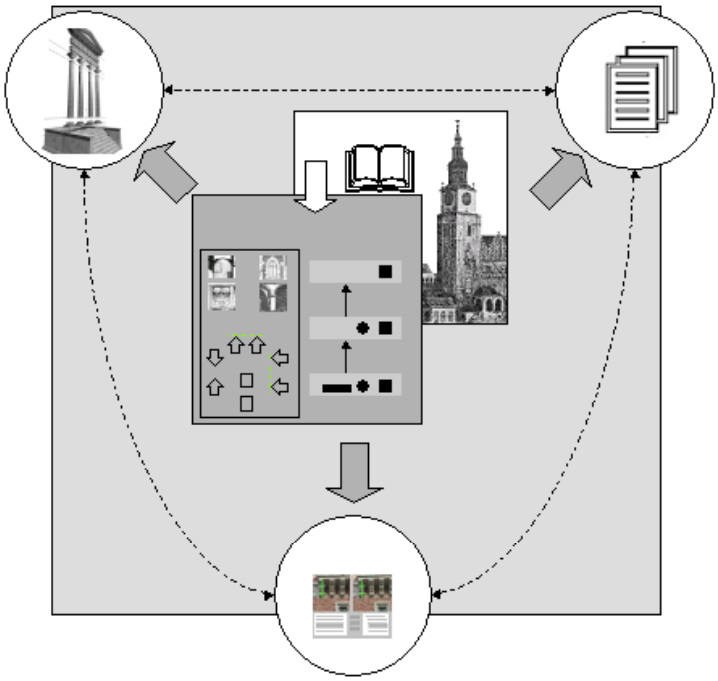


*Figure 23 : Le modèle, à l'articulation du domaine d'application et d'outils d'exploitation*

[15] -Nous positionnerons d'abord l'approche objet dans le champ de la représentation des connaissances.
-Nous aborderons ensuite les principes de la modélisation objet, puis étudierons sa pertinence au regard du domaine d'application de ce travail. Nous tenterons de cerner les limites intrinsèques de ces principes et les problèmes d'implémentation qu'ils posent aujourd'hui.
-Nous verrons notamment comment se pose la question de l'évolutivité du modèle.
-Nous décrirons enfin avec plus de détail un langage de programmation reprenant ces principes.

## 3.2. ELEMENTS DE DEFINITION SUR LE PATRIMOINE BATI ET SA CONSERVATION

Il existe bien des façons d'analyser l'architecture patrimoniale, voire l'édifice construit[16]. Notre rôle n'est ni de le démontrer ni de faire un état de l'art sur ce sujet. Nous croyons cependant inévitable de devoir statuer sur trois points:

1. Quel est le cadre, quelle est la portée de notre travail à l'intérieur de son champ d'application?
2. Sur quel jeu de références pouvons-nous nous appuyer pour, à l'intérieur du cadre défini au point précédent, réfléchir à un modèle de l'édifice patrimonial?
3. Quelles sont les exigences de modélisation particulières à notre champ d'application?

### 3.2.1. Terminologie et portée

A quelles réalités, et à quelles connaissances est il fait référence lorsque nous parlons de patrimoine architectural ? L'objet patrimonial sera pour nous la conjonction d'un édifice bâti, observable ou détruit, et des connaissances qu'il révèle tant dans le domaine de l'architecture que sur les domaines de l'histoire de l'art, de l'archéologie, de la construction, etc..[17] Cette complémentarité est déjà en filigrane des écrits de Palladio [Palladio, 1738/1965] ou de Choisy [Choisy, 1899/1991] lorsque ces auteurs appuient leur analyse d'édifices particuliers par recours à des types et à une histoire. Leur attitude n'est d'ailleurs pas contredite par l'ouvrage de référence aujourd'hui en matière d'analyse du corpus architectural, celui de J.M Pérouse de Montclos[18], qui illustre son vocabulaire par un jeu de cas réels.
Pourtant, il est utile de rappeler cette définition car elle met en lumière la différence que nous devons faire entre :

- L'édifice, un lieu que nous souhaitons pouvoir modéliser de façon exhaustive. Cet édifice témoigne d'un savoir-faire, il est décrit par un corpus d'objets, il est le sujet de notre effort de modélisation.
- Des connaissances auxquelles nous souhaitons qu'il donne accès, connaissances évolutives, souvent incomplètes.

Elle illustre notre propos puisque qu'elle présente un édifice bâti, modélisé depuis le point de vue de l'architecte, comme le médiateur entre divers champs d'activité, hypothèse fondatrice et cadre principal de notre travail.
A cette définition doit naturellement s'ajouter une explicitation des règles choisies pour élever un édifice et son corpus d'objets au rang de "patrimoine architectural". Il n'existe dans l'expression elle-même aucune trace d'exclusion. Tout ce qui est bâti est donc potentiellement patrimoine architectural. Les règles de qualification sont donc du ressort de la coutume locale, et varient très largement d'un pays à l'autre. De nombreuses recherches menées dans le cercle des architectes-conservateurs ont eu pour objectif de cerner des critères "objectifs" de classement. Il ne nous appartient pas dans le contexte de ce travail de prendre position sur une démarche qui relève de l'expertise de spécialistes, spécialistes non seulement du domaine de l'architecture mais encore de son développement local ou régional. L'expérience de l'équipe de Sint-Lukasarchief (se reporter pour les détails de ce travail à [Apers et al, 1979]) en est cependant une illustration très intéressante, que nous relatons brièvement en **annexe 4**.

[16] On peut se reporter à [Boudon, 1977] qui à travers l'étude de la ville nouvelle de Richelieu caractérise de façon claire cette diversité.
[17] Les modalités d'analyse de l'édifice patrimonial à fins de modélisation sont décrites ainsi dans [Coste, 1997, p27] :
"*Aujourd'hui lorsque nous étudions ces édifices dans leur dimension physique, la prise en compte de leur histoire […] permet de modéliser non un objet théorique mais un bâtiment […] qui nous parvient marqué par le temps et les hommes. Il s'agit alors de considérer toutes ces données simultanément, afin de ne pas étudier […] un modèle idéal mais chacune de ses matérialisations « imparfaites »* "
[18] "Architecture – vocabulaire - principes d'analyse scientifique [Pérouse De Montclos, 1988]

A quelles études, à quelles pratiques est-il fait référence lorsque nous utilisons le mot "conservation", quel cadre fixe t'il à notre travail ? La notion de conservation du patrimoine architectural est apparue en France au cours du dix-neuvième siècle. Les pratiques qu'elle recouvre aujourd'hui sont très diverses, depuis la fouille archéologique jusqu'à la restauration d'art. Cette diversité nous intéresse car elle indique une fois encore que l'édifice peut être vu comme intégrateur de connaissances.
En Pologne, où se situe notre terrain d'expérimentation, la protection du patrimoine architectural relève des compétences complémentaires de l'Etat et du gouvernement local. L'Etat protège un ensemble d'édifices, le gouvernement local prend l'initiative de compléter ce dispositif. A Cracovie, l'ensemble du secteur ancien, délimité par le tracé de fortifications aujourd'hui détruites, est placé sous la protection d'un service de conservateurs. Toute intervention sur un édifice érigé dans ce périmètre est soumise à un ensemble de règles visant à la fois à inventorier l'existant et à juger du projet[19]. Ce dispositif de protection a pour conséquence dans le contexte de mon sujet de mettre l'accent à nouveau sur les deux points précédemment évoqués :

- Le corpus architectural comme outil privilégié d'étude de l'édifice.
- La mise en perspective de ce corpus dans le cadre de champs d'étude complémentaires.

Il va cependant au-delà : il implique pour le conservateur de rassembler des indices concrets ou documentaires afin de comprendre l'édifice et son histoire, et de transmettre aux générations futures non pas le même édifice, mais le même témoin. Cette doctrine, longtemps discutée dans la communauté concernée, s'est aujourd'hui stabilisée au moins en Europe, comme l'établit l'ouvrage encyclopédique récent de R.Dinkel [Dinkel, 1997].

*Figure 24: La place centrale de Cracovie à la fin du XIXème siècle et aujourd'hui*

Précisons enfin pour conclure que nous restreignons notre étude aux édifices immobiliers et à un corpus d'objets architecturaux dont nous espérons qu'il soit représentatif du domaine. Nous avons choisi comme terrain d'expérimentation l'architecture des bâtiments construits sur la place centrale de la ville de Cracovie.Le corpus concerné est donc celui d'un nombre très restreint d'édifices. Néanmoins, chaque édifice localisé sur cet ensemble urbain a été transformé à de nombreuses reprises. Il s'agit donc d'un corpus architectural complexe dont nous ne prétendons pas rendre compte de manière exhaustive.
Rappelons enfin pour bien fixer les limites de notre travail que nous nous intéressons exclusivement à l'édifice comme structure bâti[20].

[19] -Une intervention est précédée par le mandatement d'un architecte-conservateur qui va étudier l'édifice et formuler un cahier des charges pour le projet. Pour élaborer le cahier des charges, l'architecte-conservateur étudie l'édifice à la fois du point de vue historique (identification d'une typologie en relation avec le lieu et la fonction) et du point de vue formel (sondage des murs, des sols, des plafonds à la recherche d'éléments de corpus anciens masqués par l'état présent). Le cahier de charges résultant de cette étude fixe notamment quelles parties de l'édifice (quels éléments de corpus) sont à conserver ou à restaurer. L'architecte-conservateur en charge de ce travail peut faire appel selon le cas aux compétences d'experts extérieurs (archéologues, restaurateurs d'art, chimistes des matériaux, etc..).
-Le projet architectural peut être confié soit à un architecte-conservateur soit à un architecte disposant d'une autorisation d'intervention en secteur sauvegardé. Dans les deux cas l'intervention se fait dans le cadre du cahier des charges évoqué plus haut.

[20] Pour [Beckaert, 1997] l'objet patrimonial doit être vu comme un *état social*, position que nous ne nous hasarderons pas à remettre en cause mais que notre contribution ne reprend pas compte tenu des ambitions plus étroites qui sont les nôtres.

### 3.2.2. Situation de notre jeu de références

A l'intérieur même du cadre que nous venons de fixer, de nombreux documents inventorient, classent ou décrivent le patrimoine architectural. Nous ne prétendons pas en avoir fait le tour, pour autant que cela soit possible. Sans anticiper sur la section 4 *méthodologie* nous pensons nécessaire de situer nos références au travers de trois grandes familles de travaux :

1. Analyses stylistiques
2. Etablissement de typologies et inventorisations
3. Identification d'un corpus et du vocabulaire le repérant

Ces familles se recouvrent souvent, et le premier ouvrage à citer, le traité d'architecture de Vitruve, le démontre clairement. On retrouve cette convergence à la renaissance chez Palladio, ou au dix-neuvième siècle chez Viollet Le Duc ou Choisy. Pourtant F.Fichet [Fichet, 1979] établit clairement que les écrits des architectes du classicisme français rompent avec cette vision globale. Blondel ou Félibien organisent leur discours autour de l'idée de style comme élément constitutif de ce qu'est l'édifice construit. L'académisme du dix-neuvième siècle reprend avec vigueur cette pensée, et initie la grande période des traités d'architecture établissant des classifications par style, comme ceux de G.Gromort ou P.Esquié. Sur notre terrain d'expérimentation, l'ouvrage plus récent de Z.Dmochowski s'inscrit directement dans cette lignée. En parallèle à cette démarche d'analyse stylistique nombreux sont les auteurs qui recherchent par l'établissement de typologies à organiser leur discours autour de l'idée qu'un édifice peut se lire par réduction à un jeu de types fini. C'est le cas du travail de [Baculo Giusti et al, 1995] sur un cas concret, mais c'est aussi le cas de [Pérouse De Montclos, 1988] qui isole, par chapitres successifs, un ensemble de concepts génériques qui correspondent soit à une analyse par la fonction (structurelle) du corpus architectural (Couvrement, Baie, Support, etc..) soit à une analyse par la fonction d'usage de l'édifice (Architecture religieuse, publique, etc..). Cette différence s'explique par la différence d'échelle et de type de vocabulaire concernés. Dans le premier cas, l'auteur s'attache à déterminer un ensemble de concepts relatifs aux seuls objets physiques et à leur positionnement, dans le deuxième cas il propose une définition de concepts relatifs à l'organisation de l'espace, à son utilisation, et s'adresse à l'édifice plus qu'aux objets qui le composent.Enfin, disons que nous appuyons notre travail d'identification du vocabulaire architectural sur les contributions de J.Cuisenier et de J.M Pérouse De Montclos qui nous donnent les outils méthodologiques pour procéder à une analyse finalisée des termes décrivant l'édifice. Nous pourrons alors effectuer une relecture d'autres ouvrages ayant trait cette fois au vocabulaire spécifique à nos terrains d'expérimentations[21].

Le jeu de références que nous venons de citer dessine les contours d'une approche de l'édifice patrimonial et de sa description. Mais quels travaux récents relèvent t'ils de cette approche, et quels travaux engagent t'ils une démarche de modélisation et d'instrumentation informatique?

Disons d'abord que l'idée même de modèle comme élément de compréhension d'un édifice est discutée dans le cercle des architectes, comme l'indique [Coste, 1997]. [Beckaert, 1997] rapporte ce propos de l'architecte italien Aldo Rossi[22] : "l'architecture, ce sont les architectures". Si l'on s'en tient là, un *exposé théorique abstrait,* dans les mots de G.Beckaert, ne peut suffire à rendre compréhensible l'édifice. Cette déclaration sonne pour nous comme une mise en garde, et nous la prenons comme telle[23].

[21] [Adam, 1989], [Barberot, 1922], [Choisy, 1899/1991], [Ginouvès et al , 1985], [Hébrard, 1897], ], [Łukacz, 1998], [Tajchman, 1989], [Viollet Le Duc, 1854/1996].

[22] Aldo Rossi, théoricien, auteur, artiste, architecte et professeur influent, lauréeat du prix Pritzker pour l'architecture en 1990, éléments de bibliographie à l'adresse http://isbn.nu/cgi-bin/design-books.

[23] Notons néanmoins qu'il n'y a pas contradiction denfond entre *modèle* et *architectures* de Rossi, mais plutôt d'une part la nécessité de ramener la notion de modèle à l'idée d'instrument de raisonnement dans un processus de création, et d'autre part à l'expression du soupçon qu'entretiennnent les architectes à l'égard d'explications de l'édifice dans lesquelles n'apparaitrait pas la notion d'invention.

Pourtant nous ne pensons pas devoir revenir sur la nécessité d'établir un modèle sur lequel appuyer l'étude de l'édifice, point établi clairement par [Stenvert, 1991] et [Alkhoven, 1993] sur notre champ d'application ; et au delà sur l'acte de conception lui-même par [Leclercq, 1998]. La thèse de P.Alkhoven éclaire ce que peuvent être les apports d'un travail de modélisation et d'instrumentation informatique pour l'étude de l'objet patrimonial tel que nous l'avons défini. Il ne nous sert par contre peu quand aux implémentations proposées puisque celles-ci s'appuient entièrement sur un outil ne permettant pas d'exploiter une logique de constitution de modèle tirant parti du principe de classifications des concepts.
Il n'existe en réalité que peu de travaux de développement ou de recherche qui d'une part s'intéressent spécifiquement aux questions soulevées par la gestion du patrimoine architectural et d'autre part ont fait le choix de s'appuyer sur les technologies objet. Nous allons donc dans un premier temps, pour chasser toute ambiguïté, très brièvement renvoyer à une liste succincte de ressources bibliographiques traitant de l'application des technologies objet au bâtiment en général, avant de recentrer notre propos sur l'architecture patrimoniale.
Tant en terme de formalisme de représentation des connaissances que de technologie de développement logiciel, l'approche objet est aujourd'hui largement utilisée dans le domaine de la construction ou du bâtiment en général. Dans le premier cas, on peut se référer aux nombreux articles publié par la revue ITCON (Electronic Journal of Information Technology in Construction)[24] et en particulier à [Ekholm, 1996], ou encore à [Branki et al, 1997]. Dans le second cas on trouvera dans la même revue plusieurs contributions centrées sur le développement d'applications visant à documenter et coordonner le projet de construction, en particulier [Brown et al, 1996], [Line, 1997] ou encore [Rezgui, 1998] qui présentent une expérience d'utilisation d' UML. On peut également consulter les sites Web des associations ACADIA (association for computer aided design in architecture)[25] et ECAADE (Education in Computer Aided Architectural Design in Europe). Plusieurs travaux de recherche mené au laboratoire GAMSAU-MAP se sont intéressés à l'application des technologies objet à la conception et à la communication du projet d'architecte, travaux dont on trouvera l'ensemble des références sur le site Web du laboratoire, et en particulier les thèses de S.Hanrot, M.Benmahbous et F.Ameziane.
Les problématiques de recherche dans le domaine du patrimoine architectural sont pour la plupart déjà anciennes et fortement liée à des spécialités parallèles: histoire, histoire de l'art, archéologie, ethnologie, chimie des matériaux, etc... L'application de l'informatique en général au domaine du patrimoine architectural a souvent suivi ce morcellement: bases de données de l'inventaire, relevé architectural, diagnostics sur la stabilité ou la dégradations des édifices ont par exemple fait l'objet de projets déjà cités précédemment. Les actes des rencontres annuelles de CAA (Computer Applications in Archaeology) ou des journées d'études de la SFIIC[26] en témoignent: les questions rangées sous le terme "patrimoine" sont nombreuses et très différentes. Il n'est dès lors pas très surprenant que peu de contributions puissent être citées sur la question que pose notre travail.
Nous avons déjà fait état des apports considérables que représentent les travaux de description et de classification des objets architecturaux comme [Pérouse De Montclos, 1988] ou [Cuisenier, 1991]. Ces travaux se placent pourtant résolument en dehors de la sphère informatique, et a fortiori en dehors de celle de l'approche objet. A l'intersection des champs de l'architecture patrimoniale, du relevé et de l'approche objet on peut néanmoins citer les contributions de Dirk Donath et Frank Petzold [Donath et al, 1997a] [Donath et al, 1997b], dont nous présentons les points forts en **annexe 5**.

### 3.2.3. Exigences de modélisation propres au champ de la conservation

Notre objectif sera ici d'établir quelles sont les principales différences à observer en terme de modes de description ou d'analyse de l'édifice dans les champs de l'architecture patrimoniale

[24] http://itcon.org/
[25] http://www.acadia.org/ ; http://www.ecaade.org .
[26] SFIIC (Section française de l'International Institute of Conservation) http://www.fnet.fr/sfiic/

et de la conception architecturale. Nous verrons donc ce qu'une intervention dans le champ de la conservation recouvre en terme d'exigences pour le modèle architectural.
[Cha et al, 1999] donnent une définition du processus de conception *par intégration dans un style* qui serait la sélection d'une solution parmi un ensemble de possibles. Leur travail est centré sur l'idée d'un processus de conception qui serait l'application d'un acquis (savoir lié au processus de conception) à un nouveau problème de conception[27].
En quoi ce principe général explicitant la genèse du projet architectural rejoint-il l'activité de l'architecte, conservateur ou archéologue lorsqu'il tente de comprendre l'évolution d'un édifice détruit ou transformé ? Voilà résumée la question à laquelle il nous faut tenter de répondre maintenant. De façon plus précise, il nous importe dans le cadre de l'étude dont ce document rend compte de cerner les problèmes de modélisation qui se posent spécifiquement dans le domaine du patrimoine architectural. Nous devons donc ici mettre en lumière les raisons pour lesquelles nous établissons cette distinction. Nous le ferons en nous appuyant principalement sur deux travaux de références : le compte-rendu de recherches archéologiques sur un site médiéval écrit par Gabrielle Démians d'Archimbaud [Démians d'Archimbaud, 1987] et la thèse de R.Stenvert [Stenvert, 1991]. Nous pouvons néanmoins avant de citer ces auteurs établir d'après les sources déjà citées une première liste de problèmes de modélisation liés spécifiquement au patrimoine architectural :

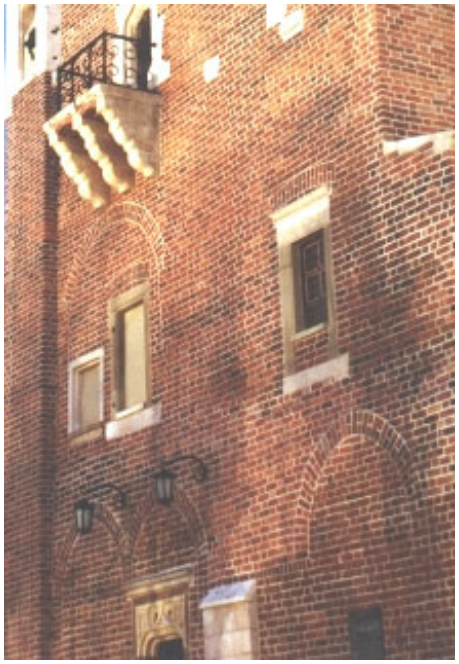


*Figure 25: Beffroi de l'hôtel de ville de Cracovie, traces sur la face Nord*

- L'objet patrimonial, au sens ou nous l'avons défini plus haut, est un ensemble de formes bâties les unes sur les autres, les unes à la place des autres, sur un laps de temps important. La première conséquence de cet échelonnement des phases de construction / transformation est la présence sur l'édifice observé aujourd'hui d'incohérences fonctionnelles au sens de la fonction d'un objet physique que définissent Viollet Le duc et [Pérouse De Montclos, 1988]. Ces incohérences peuvent être liées au réemploi d'un objet élémentaire dans un contexte incongru. On peut citer en exemple la construction en 1962 d'un balcon sur une baie nord du beffroi de l'hôtel de ville de Cracovie qui s'appuie sur des consoles supportant autrefois le hourd de la partie gothique de l'édifice détruite en 1817. Elle peuvent également résulter d'un *désemploi* de l'objet élémentaire, c'est à dire d'une utilisation de cet objet dans un contexte ou sa fonction n'est pas utilisée. On peut citer ici en exemple les doubleaux engagés encore visible sur la face Nord du beffroi et correspondant aux lunettes de la voûte en berceau qui couvrait le premier niveau de la partie gothique de l'édifice. Il s'agit bien ici d'objets ayant eu une fonction, donc relevant de l'analyse que propose [Pérouse De Montclos, 1988]. Si une fonction a justifié leur utilisation à la construction de l'édifice, il ne reste aujourd'hui qu'une forme dont la fonction théorique nous permet de renseigner le travail de formulation d'hypothèses de restitution.
- La forme de l'objet patrimonial s'apparente quelquefois au célèbre jeu de poupées russes. En effet, il est souvent plus simple d'ajouter que de remplacer. L'exhaussement constant de nos routes d'aujourd'hui est là pour rappeler cette évidence. On observe ainsi, bien entendu sur le rehaussement des sols ou planchers intérieurs ou extérieurs, mais également sur l'ensemble de l'édifice, un phénomène de "masquages successifs d'états

[27] Définissant trois types d'inférences (déduction, induction, réduction), ils apparentent le processus de conception à une réduction produisant une instance particulière à partir de notions générales (règles du domaine) et de données spécifiques (caractéristiques particulières auxquelles le résultat doit correspondre). Le néologisme projettation (ou projection) est donc bien pour eux l'action par laquelle le concepteur étend le champ des solutions possibles, ou pour reprendre la terminologie des auteurs, créé "*a new state space from existing vocabulary elements and general concepts*". Ces concepts généraux sont nommés par les auteurs *design knowledge.* Leur travail est centré sur le processus d'apprentissage de ce savoir du projet à partir d'un jeu de formes (au sens morphologies primaires et transformations) et de connaissances a priori.

cohérents". Il s'agit là d'un des problèmes les plus délicats qui se pose au conservateur sur un tel édifice. En effet, chaque masque successif révèle un état cohérent de l'édifice, autrement dit une phase de son évolution. Pourtant, on ne peut par définition en montrer qu'une ! Un choix doit donc être opéré pour établir quelle phase présente le plus d'intérêt (mot dont la définition est ici volontairement ambiguë). Les autres phases sont alors documentées, ouvrant ainsi la voie à un possible travail de modélisationet de simulation.

- Le point précédent met en lumière un constat : l'objet patrimonial n'est pas un édifice mais plutôt *des* édifices qui partagent une localisation dans la cité, éventuellement repéré par un numéro de cadastre, et une histoire[28].
- Un nouveau parallèle s'impose alors de fait : si l'activité du concepteur est celle par laquelle un couple concepts généraux / caractéristiques particulières donne forme à un nouvel objet-solution, l'étude de l'édifice patrimonial va quant à elle formaliser un objet-concept à partir d'un trio objets observés / caractéristiques particulières incomplètes / concepts généraux.
- Nous venons de mentionner une incomplétude de données. Cette idée correspond ici plus précisément à deux problèmes connexes : la présence en parallèle de données exhaustives sur telle ou telle partie de l'édifice et de données peu nombreuses ou hypothétiques sur telle ou telle autre partie de l'édifice, et le degré de validité de chaque donnée utilisée. Un modèle de l'objet patrimonial doit donc inclure un ensemble d'objets bâtis mais aussi pour chacun un niveau d'exactitude (ou d'incertitude).
- Quelles sont les données sur lesquelles se fonde l'étude des évolutions de l'édifice ? Jean Cuisenier rapporte le propos suivant : "*chercher les tableaux, aquarelles, dessins, gravures* […] *photographies qui nous rendraient l'image de ce qui n'est plus*" [Cuisenier, 1991, p248]. Ce faisant, il introduit l'idée (que l'on retrouve également chez Jean-Pierre Adam et Jean Marie Pérouse de Montclos) selon laquelle aux savoirs théorisés s'ajoute pour renseigner l'étude *l'observation des spécimens*. Viollet Le Duc nous donne par ailleurs deux indications d'importance. Il nous rappelle d'abord comment l'étude de l'architecture en géométral, mode de représentation hérité des auteurs de la renaissance, masque les effets perspectifs de l'exécution [Viollet Le Duc, 1863/1977, p274]. En dénonçant ce mode de penser l'espace, en l'opposant à ce qu'il nomme l'école laïque française des XIIème et XIIème siècles, Viollet Le Duc signale de fait que la figure, ou *anticipation graphique* pour reprendre le terme de Jean Cuisenier, n'a pas toujours eu un rôle prédominant dans l'art de bâtir. Autrement dit, Il nous prévient qu'à un pan entier de l'architecture patrimoniale ne correspond aucun document graphique systématique sur lequel s'appuyer. Viollet Le Duc rappelle en outre que des données théoriques ne nous permettront pas de remplir le vide crée : la théorisation de l'architecture ne trouve de forme écrite qu'à partir du seizième siècle. Le savoir-faire des artisans tient lieu jusqu'alors de théorie architecturale, et l'auteur s'attache à en retrouver les traces par l'observation d'édifices spécimens. Les données sur lesquelles se fonde l'étude des évolutions de l'édifice seront donc à rechercher à la fois dans l'acquis théorique résultant du travail des historiens, archéologues, architectes, dans les témoignages écrits ou figurés sur l'édifice, mais aussi dans la confrontation d'observations sur l'édifice et de connaissances sur un savoir-faire non écrit.
- Enfin, nous nous devons de signaler, puisque la question du relevé architectural sera posée dans ce travail, combien les unités de mesure locales ont eu d'influence dans la construction à toutes les époques. En effet, on ne peut légitimement tenter de donner d'hypothétiques mesures à un édifice aujourd'hui détruit sans faire référence au mode de discrétisation de la longueur en vigueur dans la cité lorsque l'édifice fut construit.

[28] . Autrement dit, si le processus de la conception architecturale vise comme nous l'indique [Cha et al, 1999] à la sélection d'une solution parmi un ensemble de possibles, l'étude de l'édifice patrimonial semble elle viser à la sélection d'un ensemble possible de solutions (souvent hypothétiques).

Nous venons d'identifier un ensemble de problèmes de modélisation qui nous semblent relever exclusivement du domaine patrimonial. Voyons maintenant en quoi cela se traduit sur un cas d'étude : l'élaboration d'une hypothèse de restitution.
Le processus d'élaboration d'une hypothèse de restitution comprend selon [Démians d'Archimbaud, 1987, p78] trois phases que nous explicitons dans le tableau ci-après :

| |
|---|
| **Observations**: Vestiges en élévation et relevé des marques d'objets manquants, vestiges au sol, fouilles, vestiges mobiliers et leur répartition, sources documentaires relatives au corpus et à l'histoire locale |
| **Interprétation**: Datation des observations, élaboration / utilisation de modèles architecturaux, comparaisons par sources documentaires relatives au corpus et à l'histoire locale, détermination des phases successives de construction et d'usage, recoupement vestiges immobiliers / vestiges mobiliers. |
| **Restitution**: Elaboration d'une ou plusieurs maquettes tridimensionnelles restituant phase par phase de possibles états antérieurs et en montrant les incertitudes, établissement de documents graphiques complétant cette description par la présentation des vestiges mobiliers, de coupes stratigraphiques et de schémas de répartition des observations. |

Pendant la phase d'observations, nous dit l'auteur, on va recenser outre des éléments d'architecture complets, en usage, un ensemble de vestiges ou de marques laissées sur un objet physique par la présence à une époque donnée d'un autre objet physique[29].
La phase d'interprétation est marquée par l'utilisation combinée des observations sur l'édifice et de modèles architecturaux soit théoriques soit établis par comparaisons avec d'autres spécimens. Elle va établir des solutions possibles en leur affectant un degré de vraisemblance qualitatif[30].

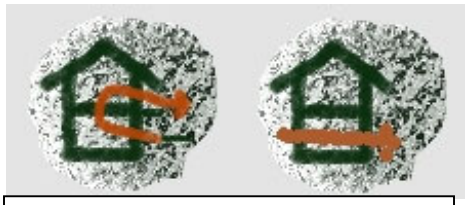


*Figure 26: Phase d'interprétation, recherche de solutions possibles.*

Enfin, la phase de restitution est pour l'auteur l'exercice permettant de rendre compte des interprétations faites sur la base des observations. Les restitutions proposées s'inscrivent pour Gabrielle Démians d'Archimbaud dans une démarche d'explicitation d'un constat, autrement dit elles sont un outil du dialogue que l'auteur de l'hypothèse entend avoir avec celui à qui il les destine, elles n'ont pas valeur de « projet architectural ».
Après le cas concret qu'expose une archéologue en position de praticienne de la conservation, citons maintenant le travail de doctorat de R.Stenvert. La question qu'entend poser Ronald Stenvert dans sa thèse d'université [Stenvert, 1991] est celle de la pertinence et de l'efficacité des moyens informatiques de traitement de l'image dans la compréhension des processus complexes d'évolution de la production architecturale et plus largement artistique. Ronald

[29] Elle définit notamment les réemplois et désemplois : "L'éclairage était assuré […] par une très petite fenêtre percée dans le mur gouttereau donnant sur la rue (ouverture actuellement murée)" [Démians d'Archimbaud, 1987, p59]. Elle établit également les antériorités ou origines de dispositifs arcchitecturaux: "[…] l'étroit passage subsistant prés du donjon contraignait les assaillants éventuels à se présenter à découvert, suivant des principes empruntés à Végèce et à l'antiquité classique [l'attaquant présente son flanc droit, non protégé, aux coups de la garnison] : survivances étonnantes à retrouver dans cette fortification petite mais complexe"[Démians d'Archimbaud, 1987, p38].

[30] Je cite ici l'auteur : "Il est plus difficile d'imaginer les salles hautes. Deux étages au moins ont pu exister, accessibles par des échelles et délimités par des planchers supportés par des corbeaux et des retraits de maçonnerie visibles, à 4m et 8 m environ de hauteur. Il est possible aussi qu'une terrasse dallée ait été aménagée au sommet pour faciliter la protection contre le feu et la circulation de la garnison : c'est du moins ce que suggère la découverte de dalles de tuf amassées à la base des décombres qui remplissaient toute la tour, au-dessus des sols conservés en place"[Démians d'Archimbaud, 1987, p38].
Cette phase va également tenter de restituer l'évolution d'un édifice en cernant des états cohérents successifs. L'auteur donne en exemple l'évolution d'un îlot en ces termes : " L'on assiste ainsi au lotissement des cours latérales, au resserrement et à la multiplication des étages dans le bâtiment F1[…] De ces mutations l'îlot F sort transformé. Trois demeures de petites superficies (30,40 ou 34.5 m² environ) sont créées. L'une au sud est formée d'une des pièces de l'ancienne habitation et de la cour attenante. La seconde se compose du bâtiment F1 désormais exhaussé et de l'ancienne salle centrale transformée en cour. La troisième occupe purement et simplement la cour Nord et le couloir de circulation, désormais cloisonné et charpenté, prés de l'enceinte" [Démians d'Archimbaud, 1987, p63].

Stenvert nous livre quelques réflexions qui rendent plus intelligibles les particularités du domaine d'application qu'est le patrimoine architectural. Il évoque d'abord une relation qu'il nomme *problématique* entre les champs de l'histoire de l'architecture et de la *science informatique* (i. e computer science dans le texte) [Stenvert, 1991, p15]. Il explique cette situation par les caractéristiques opposées des deux champs : approche globale et ambiguïté d'un côté, rigueur et précision de l'autre. Au delà de ce constat un peu manichéen, l'auteur aborde plusieurs points importants :

- Les données visuelles sur lesquelles s'appuient les conservateurs ou historiens de l'architecture n'ont souvent à l'origine pas vocation à représenter un édifice : il faut analyser leur contenu en dehors du contexte du propos qu'elles illustrent.
- Ces données visuelles ne sont informatives qu'après interprétation du matériel brut.
- Elles ne peuvent être interprétées légitimement qu'en relation à une grammaire picturale contextuelle, autrement dit aux conventions de représentations propres à une culture donnée (ou à couple lieu / période) [Stenvert, 1991, p28].
- L'auteur définit pour chaque donnée visuelle quatre groupes d'informations[31]. Il ajoute, et c'est là l'élément que nous souhaitons mettre en lumière, qu'une description exhaustive des données visuelles dans chaque groupe s'avère dans la pratique un objectif plus qu'une réalité. Ce faisant, il situe la donnée visuelle comme une source active d'informations, et par conséquent les hypothèses qui résultent de son utilisation comme évolutives

L'objet patrimonial lui-même comme les données (théoriques ou contextuelles) sur lesquelles s'appuie son étude sont donc à considérer non comme un jeu de réduction produisant une instance particulière, mais comme un ensemble de paramètres qui donnent lieu à la production d'un ensemble de possibles. Chaque arrangement particulier de ces paramètres peut-il néanmoins être rapproché de la sélection d'une solution dont parle [Cha et al, 1999] à propos du processus de la conception architecturale ? On serait bien sûr tenté de le croire. Il nous faut cependant concéder que deux restrictions s'imposent :

- Les objectifs et les données sont distincts : édifier un bâtiment c'est tendre vers une solution unique à partir d'un ensemble de contraintes en dernier ressort fixes, étudier un bâtiment existant ou disparu c'est tendre vers des solutions possibles à partir de contraintes en constante évolution.
- Comment justifier la diversité du patrimoine architectural si l'on dénie au processus de la conception architecturale le droit, tout simplement, d'inventer ? Tout ce qui est patrimoine a été conception, c'est vrai. Mais les artisans du moyen âge nous ont prouvé que l'acquis seul ne peut justifier le créé.

- *Objet patrimonial vu comme un ensemble de formes bâties successivement sur un même lieu,*
- *caractéristiques particulières lui correspondant en constante évolution,*

voilà nommés deux problèmes que pose une modélisation de l'édifice bâti. Le modèle architectural dont nous présentons une première s'appuiera sur une étude typo-morphologiques d'éléments atomiques d'architecture qui interviennent dans le raisonnement du conservateur. Nous verrons que nous tentons, en nous centrant sur la forme architecturale, d'opérer ce que l'on peut maladroitement nommer une double discrétisation : sur l'espace (éléments atomiques d'architecture) et sur le temps (arrangements spécifiques de ces éléments).

[31]Information primaire (information factuelle sur la donnée), information secondaire (interprétation), information opérationnelle (discrétisation de son contenu en mots clés), information additionnelle (relation à d'autres données)

## 3.3. REPRESENTATION DES CONNAISSANCES ET APPROCHE OBJET

Nous positionnerons ici l'approche objet dans le champ de la représentation des connaissances. Nous aborderons ensuite les principes de la modélisation objet, puis étudierons sa pertinence au regard du domaine d'application de ce travail. Nous tenterons de cerner les limites intrinsèques de ces principes et les problèmes d'implémentation qu'ils posent aujourd'hui. Nous aborderons ainsi en particulier la question de l'évolution du modèle.
Nous décrirons avec plus de détail un langage de programmation reprenant ces principes : le langage JAVA, en le replaçant dans la famille des LOO.

### 3.3.1. Le formalisme objet

T.Froese [Froese, 1992] définit le comportement d'un système basé sur l'utilisation d'objets comme résultant de l'agrégation des comportements contextuels des objets qui le composent. Ayant ainsi introduit l'idée d'objets individuels mis en relations, il donne trois points de vue différents au nom desquels, selon lui, l'approche objet trouve ses raisons d'être : le point de vue du génie logiciel, celui de la représentation des connaissances, celui de la gestion de données [Froese, 1992, p35]. Il nous semble utile de rapporter ce que ces points de vue recouvrent en terme d'exigences :

- Du point de vue du génie logiciel, la notion d'objet est sous-tendue par l'idée que l'important est de définir quelles opérations peut prendre en charge chaque objet, et pas comment il le fait ou quelles données il encapsule. Par conséquent, l'objet comprend d'une part une description d'opérations publiques, et d'autre part une représentation et une implémentation privées. En réduisant la "*surface visible*" du code, l'approche objet du point de vue du génie logiciel réduit la complexité et le temps de développement des applications, accroît la modularité et la réutilisabilité du code, et enfin simplifie les opérations de maintenance et de mise à jour des développements.
- Du point de vue de la représentation des connaissances, la notion d'objet est utilisée comme un ersatz de l'objet réel. Elle permet de produire un modèle fonctionnel de la réalité autorisant la mise en œuvre de traitements basés sur la connaissance. T.Budd écrit à ce propos les mots suivants :"*la façon de résoudre un problème que sous-tend l'approche objet est souvent celle que la méthode utilisée pour résoudre les problèmes de la vie quotidienne*" [Budd, 1996, p3]. Nous verrons que de nombreuses difficultés attendent celui qui espère pouvoir prendre au mot le principe objet réel -> objet informatique dès qu'il s'agit d'en passer par l'utilisation d'un LOO.
- Du point de vue de la gestion de données, un système de gestion de base de données objet peut appliquer de façon uniforme un traitement à chaque objet, quel que soit le contenu encapsulé par l'objet.

L'utilisation de la notion d'objet est donc du point de vue du génie logiciel un moyen efficace d'abstraction et d'encapsulation de données, du point de vue de la représentation des connaissances, un moyen de représenter avec un haut niveau d'abstraction des objets du monde réel, et enfin, du point de vue de la gestion de données, un moyen de traitement uniforme de données hétérogènes [Froese, 1992, p36]. Nous allons dans les sections suivantes avoir l'occasion de vérifier d'une part comment se traduit en termes pratiques cette notion d'objet, et quels problèmes concrets cette définition ne mentionne pas.
Nous commencerons par un rapide examen des origines de l'approche objet afin de replacer le modèle à classes et instances, celui qui va en définitive nous intéresser au premier chef, dans une perspective plus large. Nous proposerons une liste des principaux concepts en jeu dans ce formalisme et fixerons un ensemble de termes les représentant. Nous reviendrons alors sur la notion de modélisation objet et sur les principaux problèmes qu'elle soulève selon le sources bibliographiques citées. Enfin, nous conclurons cette section par un aspect important, celui de l'évolutivité et de la persistance du modèle.

### 3.3.1.i) Origines

La séparation données / actions est à la base des premiers langages informatiques (aucune structuration, très liés aux machines, pas de types structurés ) comme de leurs successeurs les langages de programmation structurée. Avec l'accroissement de la taille des programmes, la richesse des environnements, la multiplicité des matériels, la généralisation des réseaux; elle devient un défaut rédhibitoire. L'objectif vers lequel tendre alors tourne autour des notions d'abstraction de types, de modularité, de réutilisabilité.
L'approche objet est un formalisme de représentation des connaissances mais aussi une démarche méthodologique de programmation. Le nombre de ses objectifs est en fait à la mesure de la diversité de ses utilisations[32]. En effet, l'émergence de l'approche objet tient aux exigences renouvelées formulées sur le champ des formalismes informatiques à la suite des progrès techniques permettant de réaliser des systèmes dits complexes. Ces exigences incluaient pour [Ducournau et al., 1998, p6] de confier un rôle de plus en plus important au spécialiste du domaine.

L'approche objet est donc un mode de raisonnement, qu'accompagne une méthodologie de programmation: c'est en fait une démarche d'analyse de problème qui s'appuie sur la très ancienne démarche logique Aristotélicienne. L'approche objet répond selon [Ducournau et al., 1998, p6-9] à trois préoccupations :

- Unicité: l'approche objet part du postulat que les objets du monde réel existent, et que, moyennant quelques précautions quant à leur contexte d'utilisation par exemple, ils sont manipulables, modélisables. La correspondance directe monde réel / objets informatiques est pour les auteurs le moyen offert au programmeur de s'exprimer conformément à son intuition, moyen dont ils notent l'efficacité.
- Abstraction: l'objet concret tel que décrit ci-dessus pose bien sûr la question du rapport concret / abstrait, ou classe / instance dans la terminologie adoptée in fine. Elle place également l'individu devant la nécessité d'organiser, de classer les concepts isolés.
- Hiérarchisation: établir des taxinomies est un besoin que l'on retrouve dans beaucoup d'activités humaines. L'organisation hiérarchisée de systèmes de classes est souvent citée comme la caractéristique majeure de la programmation orientée objet, situant clairement l'importance du mécanisme d'héritage.

A ces trois préoccupations, à ces trois besoins, le modèle à classes et instances apporte une solution que [Ducournau et al., 1998] rappelle approximative mais aussi efficace. Pour ces auteurs, le modèle à classes et instances (tel que l'implémentent les Langages de programmation Orientés Objet, dont que nous citerons plus loin quelques exemples et qui fondent la base de notre implémentation) marche car des besoins, évoqués ci-dessus, trouvent là une solution technique.
L'approche objet établit une correspondance directe entre un objet du monde réel, isolé comme pertinent sur un domaine d'application, et un objet informatique. Elle a bien sûr des points communs avec la démarche des naturalistes, démarche qui tend à proposer une hiérarchie de concepts censés représenter de façon exhaustive et raisonnée l'état des connaissances sur le domaine. [Ducournau et al., 1998], reconnaissant ces points communs, retournent cependant la proposition en arguant des théories évolutionnistes récentes apparues en biologie pour illustrer la fragilité de cette démarche intellectuelle par classifications. Mais n'anticipons pas sur ce point, et revenons aux fondements de l'approche objet. Celle-ci permet donc de traduire en structures informatiques des opérations intellectuelles d'identification et de classement de concepts qui relèvent du bon sens, et de la connaissance particulière d'un

[32] Le *"paradigme de l'approche objet est essentiellement la solution de problèmes par recours à l'abstraction*" nous indique par exemple [Durnota, 1995] chez qui l'approche Objet est utilisée comme démarche de modélisation d'un domaine d'application.
"*L'objectif premier de la conception Objet* [au sens de la Programmation Orientée Objet] *est d'améliorer la productivité, la qualité et la maintenance de grosses applications logicielles"* indique pour sa part [Wu, 1996]. Il n'y a d'ailleurs entre ces deux contributions pas d'oppositions mais des différences de vocabulaire que la genèse des technologies Objet peut contribuer à expliquer.

domaine d'application[33]. Deux tendances se sont fait jour dans ce qu'il est convenu d'appeler l'approche objet pour organiser de telles structures, à savoir les frames basées sur le travail de Minsky et les objets autonomes auxquels on adresse des messages, principe sur lequel se fonde l'implémentation des Langages de programmation Orientés Objet. Ces tendances partagent deux points communs qui justifient leur rattachement au même courant langages à objets :

- L'idée d'objet -entité de base qui encapsule données et procédures de manipulation des données. La définition de l'objet pose la question de la distinction entre le concept et l'instance, et pose le problème de l'existence réelle de ce que l'on nomme concept, existence selon [Ducournau et al., 1998] défendue par Platon alors qu'Aristote faisait du concept un attribut des instances.
- L'idée d'héritage -créer entre les descriptions des objets des liens de particularisation et de généralisation. Le besoin de classer est inhérent aux sociétés humaines. Or, les concepts sont naturellement dotés d'une relation d'ordre: leur degré de généralité. Pour [Ducournau et al., 1998], le mécanisme d'héritage implémenté dans les Langages de programmation Orientés Objet se charge effectivement de répondre au besoin évoqué mais ne correspond plus au sentiment intuitif qui voudrait que la spécialisation implique l'inclusion des concepts.

Ces deux points communs sont en outre deux moyens de particulariser un énoncé général: abstraction / instanciation dans le premier cas, héritage / spécialisation dans le second. Langages de frames et langages de classes sont les deux tendances incluses dans l'approche objet[34]. Nous allons essentiellement en annexe tenter d'illustrer les différences concrètes entre ces deux démarches, puis ferons une rapide citation du formalisme des patterns de Christopher Alexander et de son influence sur la modélisation objet. Nous revenons plus loin sur les Langages de programmation basés sur le modèle à classes et instances tel que le définissent [Ducournau et al., 1998], et qui nous intéresse plus directement.

*3.3.1.i.1. Objets, Frames*

"L'approche objet tente de modéliser les modes de comportements de collections d'objets physiques en relation dans le monde réel. La programmation Orientée Objet fournit une meilleure façon de définir les données et les procédures [actions] qui sont associées à ces objets physiques que ne le font les langages impératifs classiques comme C ou Pascal [Wu, 1996, p1]". En introduisant son travail de comparaison entre Objets et Frames par ces termes, X.Wu définit la genèse de l'approche objet comme résultant d'un constat de manque, lié au développement de grosses applications logicielles, et qu'il fait remonter à la fin des années soixante. L'auteur précise ensuite comment l'approche objet a basculé d'un problème de définition de langage à un problème de méthodologie de travail. Nous revenons sur sa contribution en **annexe 6**.

*3.3.1.i.2. Objets, patterns*

L'influence du formalisme des patterns de Christopher Alexander sur la modélisation objet est attestée par de nombreux auteurs, notamment [Rising, 1996], [Lea, 1997], [Dodani, 1999], cités ici, mais aussi par un ensemble important de publications auxquelles on peut se

[33] Un objet ("structure") en représentation des connaissances est, pour [Ducournau et al., 1998], un réceptacle de connaissances déchargeant le programmeur de choix et d'opérations.

[34] ***Langages de frame.*** Issus de la filière représentation des connaissances, ils sont basés sur le constat d'une inadéquation des langages classiques à représenter des concepts et des relations entre ces concepts. Les langages de frame sont avant tout destinés à représenter et à gérer des connaissances. Ils sont plus ou moins largement inspirés des idées de Minsky et présentent un fort caractère déclaratif (KRL, ...). Minsky propose un formalisme général de représentation des connaissances qui repose sur la réutilisation dynamique des structures appelées frames, qui représentent des modèles de «situations stéréotypiques / objets typiques», se traduisant par un réseau hiérarchisé de nœuds et de relations (voir [Masini et al, 1989]). Une bibliographie de Marvin Minsky est proposée à l'adresse : http://www.ai.mit.edu/people/minsky/bibliography.html

***Langages de classes.*** Issus de la filière génie logiciel, ils sont avant tout des langages de programmation cherchant à répondre aux problèmes posés par l'accroissement de la taille et de la complexité des programmes. Ils se focalisent sur les notions de composabilité, de réutilisabilité, de fiabilité, etc..(Smalltalk, C++, JAVA, etc..).

reporter[35]. Nous nous appuyons sur l'analyse que propose [Lea, 1997] des idées introduites par Christopher Alexander, et sur l'influence de ces idées sur la modélisation objet telle que la démontre l'auteur, pour effectuer en **annexe 7** un rapide survol des principes des *patterns languages*.

*3.3.1.i.3. Le modèle à classes et instances*

La Programmation Orientée Objet est issue du modèle à classes et instances que [Ducournau et al., 1998, p5] qualifient de "*remarquablement simple*" voire "*rustique*". Les auteurs reconnaissent néanmoins l'efficacité de cette technique et portent leur attention sur les raisons de ce succès. Le premier indice qu'ils rapportent au lecteur est l'ambiguïté de la classe définie en intension ou en extension, ambiguïté que les auteurs juge comme source de la fécondité de cette technique. Dès l'introduction de leur ouvrage, ils explicitent leur démarche: "*Nous estimons qu'à vouloir examiner [l'approche objet] sans référence aux vicissitudes opérationnelles, on est conduit à prendre ses désirs pour des réalités*"[Ducournau et al., 1998, p5].

Dans le cadre du présent document il nous faut néanmoins commencer par situer les principes de base de l'approche objet. Il nous faudra surtout chercher à établir un vocabulaire non flou, vocabulaire qui ne se voudra valide dans ses détails que dans les limites de ce document. Les définitions et la terminologie sur lesquels nous revenons en annexe montreront qu'il n'y a pas toujours unanimité des auteurs. Nous pourrons dès lors adhérer à la démarche pragmatique de [Ducournau et al., 1998] et faire référence "*aux vicissitudes opérationnelles*".

Dans le chapitre de leur ouvrage consacré aux concepts de base des langages et systèmes à objets, [Blair et al, 1991] donnent un aperçu très clair des principes de l'approche objet. Les principes qu'ils décrivent restent aujourd'hui valides, nous nous appuierons sur leur contribution pour introduire ces trois notions fondamentales que sont l'objet, la classe et l'héritage.

•Un système basé sur l'utilisation d'objets est un système dans lequel des entités, les objets, interagissent afin de produire tel ou tel résultat. Si le terme "orienté objet" reste sujet à interprétations, l'objet est cependant unanimement défini comme le nœud de base d'un tel système. Mais qu'est ce que l'objet reste un point à éclaircir. Pour [Blair et al, 1991], la réponse à cette question est plus aisée à trouver en considérant l'objet d'une part au point de vue conceptuel et d'autre part du point de vue pratique (de l'implémentation)[36].

L'état de l'objet à un temps est fixé par ses données (propriétés), les opérations reconnues dans le contexte de l'objet forment l'interface entre l'objet et le monde extérieur. Cette interface, réglant le comportement de l'objet, est seule visible pour l'utilisateur. L'interaction entre objets est également limitée à cette interface, l'interaction entre objets est donc équivalente à l'appel à des méthodes associées à d'autres objets (par envoi de message pour reprendre un terme souvent utilisé). [Blair et al, 1991] rappellent le contrecoup de ce principe : l'objet requérant ne sait pas si l'objet requis peut effectivement répondre, ou comment il le

[35] Une introduction aux concepts et à la terminologie des patterns dans le contexte de la POO est proposée par B. Appleton à l'adresse suivante , avec plusieurs publications: http://www.enteract.com/~bradapp/docs/patterns-intro.html
Une liste de publications relatives au formalisme des patterns présentées à la conférence OOPSLA 95 (workshop on design patterns for concurrent, parallel, and distributed object-oriented systems) est proposée à l'adresse suivante: http://www.cs.wustl.edu/~schmidt/OOPSLA-95/html/papers.html
Nikos A. Salingaros (Professeur de Mathématiques de l'Université du Texas, mais aussi théoricien de l'architecture), donne pour sa part une biographie rapide et une bibliographie choisie des œuvres de Christopher Alexander à l'adresse suivante: http://www.math.utsa.edu/sphere/salingar/Chris.text.html#PUBLICATIONS ,
Linda Rising a publié un ouvrage de compilation centré autour de l'influence des patterns sur le développement d'application logicielles: "*Patterns Handbook*" par L.Rising, editions Cambridge University Press, 1998, ISBN 0-521-64818-1

[36] - Du point de vue conceptuel, l'objet est utilisé pour représenter une entité du monde réel : c'est l'opération de réification [Gaillard, 1994]. Les auteurs notent toutefois que ceci n'est qu'approximativement vrai puisqu'il est souvent dans la pratique nécessaire d'introduire dans le modèle des objets dont le rôle n'est lié qu'à tel ou tel formalisme de développement. L'objet apparaît néanmoins comme une structure abstraite proche du domaine de l'application par opposition à la programmation impérative où les structures abstraites sont proches de la machine.
- Du point de vue pratique, la notion d'objet se plaque pour [Blair et al, 1991] directement sur le concept d'encapsulation. L'objet est ici l'encapsulation d'un jeu de méthodes qui peuvent être invoquées par le monde extérieur et d'un jeu de données qui retiennent les effets de ces méthodes.

fera. Ceci implique naturellement de mettre en œuvre un mécanisme de gestion d'exceptions dont nous verrons un exemple en présentant le langage JAVA.

•La classe est un formalisme de classification des objets qui permet de rassembler dans un même groupe des objets de comportements identiques. La classe est plus précisément un moule à partir duquel des objets particuliers peuvent être produits. Elle contient une définition des propriétés de l'objet et des méthodes qui lui sont associées, et par conséquent définit à la fois l'interface de l'objet avec le monde extérieur et sa description interne. Avec la notion de classe apparaît naturellement le besoin de produire des objets particuliers : c'est le mécanisme d'instanciation. L'instance est l'objet produit par appel à une méthode d'instanciation de la classe, indiquant que la classe elle-même est un objet du système[37].

•L'héritage permet de s'appuyer sur la définition d'une classe existante pour définir une nouvelle classe qui la modifie. La nouvelle classe partage le comportement de la classe existante, ou plus exactement le partage et le modifie, le partage et y ajoute de nouveaux comportements. L'utilisateur créant une nouvelle classe spécialise la classe existante mais, on peut dès maintenant le noter, seulement par ajout ou modification de propriétés. [Blair et al, 1991] définissent trois modes de spécialisation : ajout de comportement, changement de comportement, suppression de comportement. Ils notent cependant que si les deux premiers modes sont implémentés dans la plupart des Langages de programmation Orientés Objet, il n'en est pas de même du troisième. Nous reviendrons sur les conséquences à tirer de ce constat dans la section consacrée aux problèmes de modélisation.

Conséquence du mécanisme d'héritage, une hiérarchie de classes est construite, dont les nœuds sont les classes et les arcs représentent les relations de spécialisation (sorte-de). Une classe de base sert de racine à l'arbre d'héritage, dont le rôle et le contenu varient en fonction de l'implémentation[38]. Autre conséquence du mécanisme d'héritage, la description du comportement d'un objet n'est plus accessible exhaustivement à l'intérieur de la classe l'ayant produit mais se retrouve disséminée le long de l'arbre d'héritage. En particulier, les méthodes gérant le comportement de l'objet peuvent être redéclarées n'importe où le long de cet arbre. Pratiquement, lorsqu'un message est adressé à un objet, la méthode appelée est recherchée le plus souvent de façon ascendante dans l'arbre d'héritage, et le plus souvent également de façon dynamique. Troisième conséquence du mécanisme d'héritage, une méthode peut être redéfinie à l'intérieur de plusieurs classes ou plutôt être implémentée de façon différente et par conséquent produire des résultats différents. Souvent appelé polymorphisme, cette capacité d'interprétation du message correspond pour [Blair et al, 1991] au constat qu'il peut y avoir un rapport d'un à plusieurs entre le nom d'une méthode et ses implémentations.

[Blair et al, 1991] étudient les rapports entre le modèle objet à classes et instances et un ensemble d'objectifs de l'approche objet en général en citant trois points de repère :

- Abstraction: la notion d'objet permet effectivement de disposer d'entités abstraites. En autorisant modularité des développements et encapsulation elle offre un moyen de mieux gérer des systèmes complexes.
- Partage de comportement: en premier lieu la notion de classe garantit la similitude de comportements de toutes ses instances. L'héritage raffine ce mécanisme en permettant à une classe de partager et d'amender un comportement. Le polymorphisme (au sens de ce mot que donne [Blair et al, 1991]) autorise également le partage de comportements entre classes hors hiérarchie.
- Evolution: le mécanisme d'héritage s'adapte bien à une programmation évolutive pour autant que l'on se limite au paradigme d'une programmation par différentiations. L'évolution se fait ici par comparaison d'un jeu de concepts existants et d'un problème à modéliser. On doit en conclure qu'évolution n'est pas un terme adéquat : il s'agit en fait concrètement plutôt d'extension, ce qui ne va pas sans poser des problèmes de modélisation sur lesquels nous reviendrons.

[37] En résumé, la classe définit un groupe d'objets dont les comportements sont identiques, elle est apte à générer de nouvelles instances, elle est elle-même objet auquel sont passés des messages invoquant la méthode d'instanciation de son interface.

[38] Concrètement, la plupart des Langages de programmation Orientés Objet fournissent une bibliothèque de classes dont la nécessaire lisibilité pour l'utilisateur est un point important à souligner.

Ayant défini dans leurs grandes lignes les principes d'objet, de classe et d'héritage nous pouvons désormais étudier avec plus de détail les définitions du formalisme objet qui ressortent de la bibliographie rapportée ici.
Ces définitions doivent nous permettre de limiter la portée de notre contribution dans la démarche de modélisation par objets, démarche qui dépasse bien sûr très largement le contexte des sujets abordés ici, comme en témoigne par exemple l'imposant site de référence *Object-Oriented Programming and Systems*[39] auquel on peut se reporter.

### 3.3.1.ii) Eléments de définition

Nous insistons sur la nécessité d'établir un vocabulaire architectural précis pour désigner les concepts que nous souhaitons manipuler. L'occasion de faire le rapprochement avec la terminologie des technologies objet est trop belle pour que nous ne la saisissions pas. En effet il apparaît clairement que l'approche objet a généré un ensemble de vocables dont les définitions ne sont pas toujours consensuelles. Il nous semble par conséquent naturel, ici aussi, de limiter l'effort d'interprétation demandé au lecteur en fournissant une définition, relative à ce document, des concepts utilisés. Le mot objet, concept commun à l'ensemble du domaine de *l'orienté objet* est lui-même comme l'indiquent [Blair et al, 1991] notoirement difficile à définir. On se reportera à l'introduction de [Ducournau et al, 1998, p3-4] pour une discussion plus approfondie sur le sens des termes utilisés dans la "*planète des objets*". Loin de se vouloir une compilation exhaustive, notre contribution se borne à établir ce à quoi tel ou tel vocable fait référence ici, et par conséquent sur quelles bases nous avons travaillé.
Nous allons donc dans cette section recenser les principaux concepts à la base des langages de programmation orientés objets, puis établirons une terminologie qui identifiera le champ d'utilisation du mot que nous allons utiliser pour désigner ces concepts[40]. Nous avons introduit la section intitulée "le modèle à classes et instances" par les mots "La Programmation Orientée Objet". C'était là déjà constater que, compte tenu de nos *vicissitudes opérationnelles*, nous nous intéresserons avant tout aux implémentations des principes évoqués ici dans le cadre des langages de programmation orientés objet.

#### *3.3.1.ii.1. Principaux concepts*

Nous avons d'ores et déjà introduit trois notions fondatrices de la Programmation Orientée Objet (objet, classe héritage). Nous allons maintenant préciser leurs définitions et compléter cette présentation. Oscar Nierstrasz notait [Nierstrasz, 1989] une prolifération de définitions et d'interprétations du terme "orienté objet", et tentait d'en borner l'usage. Nous partirons de sa contribution pour examiner les définitions qu'il propose, rapporter quelques interprétations plus récentes et introduire des compléments.

*L'IDEE D'OBJET*

Pour [Nierstrasz, 1989], tout langage de programmation exploitant le principe d'encapsulation de données et de méthodes peut être considéré comme un langage orienté objet. Pour Oscar Nierstrasz, l'objet regroupe un jeu d'opérations (son interface) et un jeu de traitements (implémentations cachées à l'utilisateur). L'auteur définit un autre critère de jugement des langages orientés objet : la cohérence du modèle objet supporté.
A ce titre, le langage Smalltalk reste une référence, mais nous verrons plus loin que le langage JAVA n'en est pas très loin. Pour [Gaillard, 1994], la notion d'objet est fondée sur trois propositions et un concept fondamental, celui de la réification[41].

---

[39] Computer Science Bibliographies, Bibliographies on Object-Oriented Programming and Systems, site à l'adresse suivante: http://liinwww.ira.uka.de/bibliography/Object/

[40] ainsi qu'éventuellement des synonymes et des utilisations alternatives relevées dans les sources bibliographiques citées.

[41] -Un objet conceptuel est décrit par l'ensemble des attributs qui le caractérisent et des opérations qui permettent de le manipuler.
-Tout peut se représenter sous la forme d'objets conceptuels.
-Les objets sont dans un rapport d'adéquation avec le monde.
-La réification est l'opération par laquelle un objet réel (chose physique, relation, événement, etc..) est représenté de façon informatique sous la forme d'un objet. C'est l'opération par laquelle sont choisis les objets que l'on souhaite manipuler et comment ils seront représentés dans le modèle [Gaillard, 1994].

Cette définition du concept d'objet complète la précédente en le replaçant comme mode de représentation des connaissances, et en insistant sur la question essentielle du processus de réification, "*clef de voûte de toute l'approche objet*" [Ducournau et al, 1998, p29]. En effet, c'est là l'étape essentielle de modélisation d'un problème, et, nous le verrons dans la partie de ce document consacrée au projet, une étape au cours de laquelle un choix de point de vue sur le problème doit être fait par le modélisateur.
Fabrice Gaillard le note bien : le plus souvent, écrit il, "*le processus de réification dépend du point de vue sur l'objet, c'est à dire de façon plus générale du contexte dans lequel s'opère cette réification*". L'auteur distingue deux méthodes de conception objet; d'une part, celle par laquelle est associé à un objet réel plusieurs objets informatiques en fonction du point de vue à prendre en compte, d'autre part celle par laquelle à l'objet réel correspond un unique objet informatique associant plusieurs points de vue sur l'objet réel. C'est dans cette deuxième approche que nous nous situons, comme selon [Gaillard, 1994] le plus souvent en ingénierie.
Pour F.Gaillard [Gaillard, 1994] comme pour T.Froese [Froese, 1992], déjà cité, l'idée d'objet s'inscrit dans des démarches complémentaires : génie logiciel, gestion de données, représentation des connaissances en particulier. [Ducournau et al, 1998], on l'a évoqué, définissent dans le même sens l'objet (informatique) comme un substitut de l'objet réel. Mais ils lui donnent une définition en trois points qui soulève des questions importantes :

- L'objet est une entité individuelle repérée par une adresse unique. Ceci pose bien le problème de la persistance du modèle que nous aborderons plus loin.
- L'objet est caractérisé par un état qui varie au cours de son existence. Cet état correspond à un ensemble de couples champ / valeur auxquels l'interface de l'objet peut donner accès.
- L'objet a "*une capacité réactive qui donne l'illusion de son existence concrète*"[Ducournau et al, 1998, p12] : son comportement. Ce comportement est réglé par un jeu de procédures prédéfinies qui sont appelées par envoi de message.

L'envoi de message ne dicte pas les opérations à effectuer : chaque objet réagit au même message de façon indépendante. Les auteurs notent que ceci a pour conséquence et inconvénient d'empêcher un raisonnement sur les procédures, donc sur le comportement de l'objet. Les procédures ne sont pas considérées comme des valeurs, l'objet comprend donc en définitive un ensemble de couples champ / valeur (son état) et un ensemble de couples nom de procédure / corps de procédures (son comportement)[42].
En résumé, au-delà des principes bien établis de l'objet comme structure encapsulant données et méthodes, construit comme artifice de remplacement de l'objet réel, nous avons en citant [Gaillard, 1994] et [Ducournau et al, 1998] mis à jour trois problèmes de modélisation inhérents au formalisme choisi :

- Si le modèle se veut cohérent, tout doit (en principe) être objet.
- Le processus de réification est l'opération fondatrice du modèle, elle doit intégrer la notion de points de vue sur l'objet réel.
- L'objet est doté d'un comportement qui se prête mal au raisonnement.

*LA REUTILISABILITE DU CODE*

La réutilisation du code qu'autorise ou que favorise l'approche objet est citée par [Nierstrasz, 1989] avant même de parler de classes ou d'héritage. Préoccupation à l'évidence dictée par une démarche de génie logiciel, cet aspect est mis en exergue par beaucoup d'auteurs et nous semble par conséquent justifier sa présence ici.
On peut par exemple se reporter à [Masini et al, 1989, pp 408-439] ou à [Budd, 1996; pp 2-15] pour une discussion sur le style de programmation induit par les langages de classes. [Wu et al, 1995] résument très simplement l'apport de l'approche objet en terme de développement d'applications logicielles en écrivant : "*La technologie objet* […] *rend plus facile le développement, la maintenance et la réutilisation d'un grand nombre d'applications*". Oscar Nierstrasz rappelle qu'encapsulation de procédures et librairies ont bien avant le

[42] [Gaillard, 1994] note qu'il est intéressant de pouvoir considérer les méthodes comme des objets à part entière sans pour autant en donner d'implémentation concrète.

développement de l'approche objet déjà amélioré la réutilisation du code d'applications. Il considère les langages de programmation orientés objet comme permettant de franchir un nouveau pas en encapsulant données et méthodes[43].

*LA NOTION DE CLASSE*

[Nierstrasz, 1989] associe la notion de classe à celle d'instanciation, considérée comme le mécanisme le plus basique de réutilisation du code. L'auteur explique qu'un langage de programmation orienté objet permet à l'utilisateur de définir "son" objet en sus des types de données fournis par le système, se plaçant ici encore clairement dans une perspective de génie logiciel.

En plaçant la notion de classe comme une des conséquences du principe d'instanciation d'objets, l'auteur peut introduire la notion d'objet-prototype. Ici l'objectif est de disposer d'un patron pour l'objet à instancier, lui-même instance, et d'en modifier ou raffiner le contenu. Cette approche, que l'auteur juge adaptée aux systèmes dans lesquels les objets évoluent rapidement, présente plus de différences que de similitudes avec l'approche classes et instances. Elle ne permet en particulier pas d'appliquer de mécanismes d'héritage comparables, écrit l'auteur. Nous renvoyons à [Ducournau et al, 1998, pp 227-256] pour une discussion approfondie sur les langages à prototypes.

Thomas Froese [Froese, 1992] définit plusieurs acceptions du terme classe sur lesquelles nous revenons plus loin. Dans le cadre de notre travail, le mot classe est utilisé dans le sens que lui confèrent les implémentations proposées dans les langages de programmation orientée objet tels que Java, C++, Smalltalk, ou même Perl.

La classe définit un patron pour ses instances. Elle définit les attributs et méthodes assignés à un type d'objet. Toute instance d'une classe partage une même définition: la classe est de façon générale chargée de représenter les concepts.

Concrètement, [Ducournau et al, 1998, p 14] rappellent que la classe est un texte qui rassemble une liste de noms (les attributs) et une listes de méthodes fixant un comportement. L'instance est l'objet constitué de ces champs et procédures. Si la valeur des champs évolue, les procédures sont elles fixées. Les messages adressés à l'instance correspondent à l'exécution en contexte local d'une procédure. La notion de classe est comme l'indiquait [Nierstrasz, 1989] indissociable de l'opération d'instanciation. Celle ci provoque selon les termes de [Ducournau et al, 1998, p 15] outre l'allocation d'une zone mémoire échappant au programmeur, l'affectation à chaque champ d'une valeur initiale par le biais d'une procédure d'initialisation. [Ducournau et al, 1998, p 16] notent ici encore une question importante : Un objet ne change pas de classe au cours de son existence[44].

*Figure 27 : Le Portique nord du forum d'Arles [Sintès et al, 1989; p42]*

Ceci pose bien entendu un problème de modélisation évident, qu'illustre cet exemple caricatural choisi dans notre domaine d'application : Qu'est ce que le portique du forum d'Arles par rapport au mur qu'il contribue à former aujourd'hui ?

Si la classe est définie par intension, il n'en reste pas moins que l'implémentation des Langages de Programmation Orientés Objets en fait une représentation de l'extension de ses instances. [Ducournau et al, 1998, p 17] concluent en notant la difficulté qu'il y a à vouloir concrètement décrire une entité singulière sans décrire automatiquement une famille d'entités isomorphes, difficulté qui pour eux demeure un obstacle majeur à la réalisation en informatique de langages à prototypes.

En résumé, au-delà des principes bien établis de la classe comme moule de fabrication d'objets singuliers appelés instances, nous avons en citant [Ducournau et al, 1998] mis à jour deux nouveaux problèmes de modélisation inhérents au formalisme choisi :

– Une instance ne peut pas changer de classe.

[43] On peut se reporter à [Menzies et al, 1995] pour une discussion critique sur l'encapsulation comme facteur d'une meilleure réutilisation du code. Il est clair que la préoccupation réutilisation du code n'est pas le centre du travail que je présente, bien que, on le verra, cette question soit abordée dans le projet.

[44] Même si l'implémentation d'un tel mécanisme est possible comme en atteste le principe de raffinement des entités architecturales que nous proposons dans notre projet.

- Une classe est définie en intension, elle représente le pôle abstrait de l'opposition concepts / instances. Or toute définition d'un individu est définition de sa famille. L'approche objet comme technique de modélisation de connaissances sur des édifices patrimoniaux pose donc nommément le problème de l'élaboration d'une description intensionnelle de phénomènes observés en grande partie par individus.

*LES RELATIONS D'HERITAGE*

[Nierstrasz, 1989] considère l'héritage comme un mécanisme de réutilisation du code permettant à des objets de partager un comportement. Il propose une liste de critères permettant d'établir les différences entre implémentations de la notion d'héritage :

- Qui hérite ? (Classes ou instances ?)
- Quelles propriétés sont héritées ? (Variables, méthodes, ..)
- Quelles propriétés héritées sont visibles ?
- Peut-on surcharger ou supprimer des propriétés héritées ?
- Comment sont résolus les conflits d'héritage ?

L'héritage simple permet à une sous-classe d'hériter des variables d'instances et des méthodes de la classe dont elle hérite et d'ajouter les siennes propres. L'héritage multiple apparaît pour l'auteur comme une extension naturelle de ce principe, autorisant une sous-classe à hériter de propriétés héritées de plusieurs classes différentes. Plusieurs points sont ici à noter :

- Tous les langages orientés objet n'autorisent pas l'héritage multiple (Java ne le permet pas par exemple).
- La surcharge de méthodes doit être autorisée.
- L'accès aux variables héritées peut être autorisé ou ne pas l'être.
- Les conflits de nom entre propriétés héritées sont résolus soit par défaut par le système soit laissés à l'initiative du programmeur.

L'héritage, note ensuite l'auteur, pose avant tout le problème de l'évolution de la hiérarchie, particulièrement dans le cadre des Systèmes de Gestion de Base de Données Orientés Objet dans lesquels les modifications doivent être propagées non seulement aux classes mais également aux instances déjà présentes. Placé dans le contexte de la représentation des connaissances, l'héritage est un mécanisme de spécialisation à coupler à un mécanisme d'agrégation. L'héritage est alors utilisé non seulement comme un mécanisme de réutilisation du code mais aussi et surtout comme un mécanisme de structuration des connaissances. L'héritage partiel implémenté en C++ permet à un objet d'hériter de certaines propriétés et d'en supprimer d'autres. A priori confortable pour le programmeur, ce mécanisme pose un évident problème de cohérence dans la hiérarchie de classe construite.
[Froese, 1992] décrit en résumé chaque objet (classe) comme une spécialisation de celui dont il hérite et une généralisation de celui qui hérite de lui. [Menzies et al, 1995] décrivent séparément l'héritage et la classification des objets en un nombre fini de groupes (classes) comme deux exigences de base de tout Langage de programmation Orienté Objet.
Ici encore, nous nous tournons vers [Ducournau et al, 1998, pp17-23] pour tenter d'établir les limitations intrinsèques de la notion d'héritage. Pour ces auteurs, l'héritage est une relation d'ordre représentée par un graphe d'héritage. Une classe A hérite d'une classe B si tous les attributs de A se retrouvent dans B. Du point de vue intensionnel, le contenu définissant la sous-classe contient celui définissant la super-classe: l'extension des instances de la sous-classe est au plus égale à l'extension de sa super-classe. Concrètement, un objet doit appartenir à l'extension d'une classe s'il appartient à l'extension de cette classe ou d'une de ses sous-classes. Ce mécanisme d'analyse de l'extension d'une classe est à implémenter en ajoutant un champ contenant l'adresse de la super-classe.
La notion de classe abstraite présente par exemple en JAVA recoupe la distinction genre / espèce : elle correspond à la nécessité de définir un concept dont on ne rencontre jamais d'instances (i.e. "le véhicule").

Pour [Ducournau et al, 1998], la relation d'héritage est un outil particulièrement commode pour le programmeur, mais les implémentations qui en sont données par la plupart des Langages de Programmation Orientés Objets posent un problème essentiel : elles imposent qu'une sous-classe soit définie par ajout de propriétés. Ils posent à l'appui de leur propos cet exemple : prenons une classe Homme. Il est aisé à partir de cette classe d'imaginer construire une nouvelle classe Barbu. Mais comment décrire par des sous-classes les notions d'unijambiste ou de manchot ? L'absence d'un champ peut pourtant être considérée comme justifiant de spécialiser un concept. Ce n'est donc pas le principe d'une classification par différences qui pose problème mais plutôt l'incapacité des Langages de Programmation Orientés Objets à exprimer ce type de spécialisation.

En résumé, au-delà des principes bien établis de l'héritage comme moyen à la fois d'organiser hiérarchiquement des concepts en procédant par ajout de propriétés, et de partage de comportement entre objets, nous avons en citant [Nierstrasz, 1989] et [Ducournau et al, 1998] mis à jour deux problèmes de modélisation inhérents au formalisme choisi à ajouter à ceux déjà cités plus haut :

- La définition d'une hiérarchie de classes pose la question de la propagation des modifications apportées à une classe en particulier en gestion de données.
- L'implémentation du mécanisme d'héritage offerte par la plupart des Langages de Programmation Orientés Objets induit une spécialisation des concepts non par différences mais par ajout de propriétés[45].

*L'ENVOI DE MESSAGE*

Pour [Nierstrasz, 1989], l'envoi de message au sens de la programmation orientée objet est un paradigme de communication qui reflète l'indépendance des objets communicants. Un message adressé à une instance d'une classe est mis en correspondance avec les procédures écrites dans celle-ci. Si la procédure n'apparaît pas dans cette classe elle est recherchée de façon ascendante dans le graphe (arbre) d'héritage. On l'a évoqué précédemment, [Blair et al, 1991] rappellent le contrecoup de ce principe : l'objet requérant ne sait pas si l'objet requis peut effectivement répondre, ou comment il le fera. Ce mécanisme se complique lorsque l'héritage est multiple : il faut alors définir un parcours dans l'arbre [Ducournau et al, 1998, p19]. Pour ces auteurs, l'envoi de message donne l'illusion que l'objet modélisé a un comportement à rapprocher de celui de l'objet réel. Or, ajoutent-ils, dans un univers de connaissances, il existe des objets immuables. Cette distinction objets / valeurs est d'ailleurs prise en compte dans la plupart des Langages de Programmation Orientés Objets qui implémentent de façon atypique des classes outils (chaînes de caractères, booléens, etc..)[46].

L'envoi de message est en résumé cet appel de procédure sur un objet requis par un requérant dans lequel l'implémentation réelle de l'opération requise est cachée au requérant.

A partir des propos que nous relevons dans [Blair et al, 1991] et [Ducournau et al, 1998], un problème de modélisation supplémentaire apparaît :

- L'appel à une méthode d'un objet requis ne signifie par que cette méthode soit définie ou redéfinie par cet objet, posant un problème de gestion d'erreurs comme de modélisation

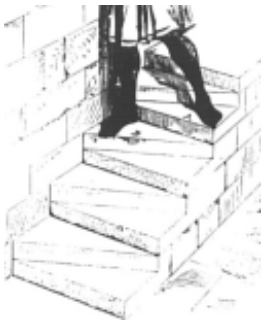


*Figure 28 : problème de spécialisation du concept de volée d'escalier, l'escalier à marches en coin d'après [Noël, 1994, p154]*

[45] Illustrons ce point par un exemple. Pierre Noël [Noël,1994, p154] donne de la volée d'escalier la définition suivante : "*suite de degrés qui […] servent à monter ou à descendre*". Il isole les propriétés essentielles d'un tel objet : une montée (distance entre les deux niveaux reliés), un emmarchement, un giron et une hauteur de marches. Comment spécialiser ce concept pour définir un *escalier à marches en coin*, c'est à dire dans lequel les marches sont alternées pour ne porter à chaque marche qu'un seul pied ? C'est une différence d'usage qui distingue ces deux concepts. Il n'y a pas ajout de propriété mais, au mieux, interprétation géométrique différente des mêmes propriétés en fonction d'un usage différent.

[46] Nous renvoyons à [Ducournau, 1997] pour une discussion approfondie sur la compilation de l'envoi de message dans laquelle l'auteur détaille les techniques de tables développées pour obtenir une sélection de méthode en temps constant pour les Langages de Programmation Orientés Objets typés dynamiquement.

*LE POLYMORPHISME*

Pour [Nierstrasz, 1989] et [Froese, 1992], le polymorphisme se définit comme la propriété d'un message à correspondre à des implémentations différentes selon l'objet auquel il est adressé. En conséquence, il s'agit pour eux d'une caractéristique permettant à une procédure de s'appliquer uniformément à un ensemble d'objets. Dans le cas de hiérarchies de classes, le polymorphisme est à rapprocher de l'idée de surcharge de méthodes. En C++ le polymorphisme nécessite une opération d'édition de liens spécifique pour les fonctions déclarées "virtual". L'invocation de ces procédures sur une instance sera résolue à l'exécution selon la classe de laquelle dérive l'instance.

Le problème du choix à effectuer à l'exécution entre procédures redéfinies, ou liaison dynamique, est évoqué dans [Ducournau, 1997].

En ce qui nous concerne, nous citerons [Ducournau et al, 1998, p26] pour évaluer les conséquences en terme de modélisation de la redéfinition de méthodes dans une hiérarchie. Pour ces auteurs, si l'intuition veut que la spécialisation d'une méthode redéfinie dans une sous-classe s'exprime en restreignant son domaine de définition (covariance), c'est en théorie la règle de contravariance qui s'applique dans le mécanisme d'héritage : pour être plus "spéciale", une procédure doit avoir un domaine de définition plus général (les nouveaux types des arguments doivent contenir les types déjà stipulés).

En pratique, la plupart des concepteurs de Langages de Programmation Orientés Objets ont choisi d'opter pour la première règle, c'est à dire type des arguments de la sous-classe inclus dans le type des arguments de la super-classe. JAVApropose une troisième voie : il interdit la modification du profil d'une méthode redéfinie par rapport à sa définition initiale.

[Ducournau et al, 1998] nous signalent donc un problème de modélisation, cette dois-ci lié à l'implémentation que nous avons expérimentée :

– Le profil des méthodes redéclarées ne peut pas être modifié en JAVA

*HERITAGE ET SOUS-TYPAGE*

Nous renvoyons à [Ducournau et al, 1998, pp 24-27] pour une introduction au rapport d'induction entre relation d'héritage et relation de sous-typage. On peut également se reporter à [Plaindoux et al, 1998] pour une discussion dans laquelle les auteurs dressent un inventaire des problèmes liés au contrôle de type dans les Langages de Programmation Orientés Objets et évaluent un ensemble de solutions.

Nous considèrerons ici acquis le constat auquel [Ducournau et al, 1998] arrivent : l'héritage n'est pas du sous-typage. Pourtant, les auteurs soulèvent un point qu'il est bon de rapporter dès à présent. En séparant les notions de classe et de type, on est amené à les relier par un lien d'implémentation, le type se réduisant alors à une signature de méthodes. Une classe qui implémente un type fournit un ensemble d'attributs et des corps de procédures qui doivent réaliser la signature. Le langage JAVA sur lequel nous reviendrons introduit ce principe par le biais de ses classes "*interface*".

*METACLASSES*

La question posée ici est en première approche celle du statut de la classe comme objet au sein du système. La classe est-elle présente sous forme d'objet manipulable ou n'est-elle qu'un texte plus ou moins compilé ? Autrement dit, quel statut pour les concepts abstraits? Sont ils doté d'une existence réelle dans le monde ou n'apparaissent ils que dans notre discours sur le monde ? En approche objet, ce problème posé depuis longtemps par ailleurs se trouve renouvelé : la classe est elle un outil qui fabrique des instances ? C'est alors un objet du monde réel, lui même fabriqué: c'est la notion de métaclasse, dont les instances sont des classes. Le niveau Méta de la classe est la métaclasse comme le niveau Méta de l'instance est la classe. Concrètement les langages à compilation séparée comme C++ n'autorisent pas de façon aisée ce mécanisme. JAVA, au départ dans ce cas, a basculé, avec le package java.reflect de sa version 1.1, du côté des réalistes qui affirment que les concepts existent réellement, pour reprendre les termes de [Ducournau et al, 1998, p 28]. Bien sûr, avec cette démarche en niveaux Méta successifs se pose le problème de l'amorçage. De nombreuses contributions peuvent être citées sur le thème de la génération de classes, ou plus

généralement sur le développement de méta-outils objets . On peut se reporter par exemple à [Maugham et al, 1998] pour la présentation d'un méta-modèle Objet destiné à l'analyse d'applications ou à [Sunyé et al, 1997] qui implémentent un environnement de génération d'outils de développement logiciel en Smalltalk.
[Ducournau et al, 1998] ouvrent en parallèle au principe de métaclasses un deuxième champ de recherche : le statut des messages ou des méthodes comme objets eux-mêmes. Il est clair que nous pouvons voir ces entités comme des objets sur le comportement desquels nous voudrions pouvoir agir. Cette réification des messages ou des métaclasses est aujourd'hui plus ou moins disponible selon les systèmes.
En Smalltalk par exemple, un message refusé par son destinataire est réifié en objet-message. De façon générale, métaclasses, classes et objets sont activés en Programmation Orientée Objets de la même manière; réification et réflexion (l'opération inverse) posent donc avant tout un problème de modélisation: *une modélisation réussie réifie à bon escient* [Ducournau et al, 1998, p 31]. Pour ces auteurs, la modélisation par objets trouve sa pertinence lorsque le domaine d'application peut s'analyser par le "sens commun". Autrement dit, la modélisation Objet a pour limites, en l'état actuel des outils disponibles, celles qu'a l'Aristotélisme: les concepts généraux peuvent ils être induits par l'observation d'individus identifiables par un faisceau de traits caractéristiques ?
Se pose à travers ce constat un problème de modélisation inhérent au formalisme choisi à ajouter à ceux déjà cités. Pourtant, dans le cadre de notre étude, le processus de réification doit être replacé dans un contexte précis :

- Nous ne parlons pas de conception architecturale (au sens de la définition d'un parti architectural).
- Nous traitons d'édifices bâtis du point de vue de leur description en terme de corpus, concepts issus du vocabulaire architectural existant.

Nous considérerons donc qu'eu égard à ces limitations une modélisation Objet d'édifices patrimoniaux peut être légitimement expérimentée. Néanmoins, nous devons retenir après [Ducournau et al, 1998] deux perspectives :

- La métaclasse peut être un outil qui fabrique des classes.
- Les méthodes peuvent être réifiées à l'instar des classes[47].

*ENVIRONNEMENTS DE PROGRAMMATION*

Nous terminons cette section par la question des environnements de programmation Objet soulevée par [Nierstrasz, 1989] comme par [Blair et al, 1991]. La Programmation Orientée Objet a clairement soulevé l'espoir d'une réduction des coûts de construction d'applications logicielles. Encore faut-il pour cela que les objets implémentés soient réellement réutilisables. Oscar Nierstrasz propose plusieurs critères d'évaluation des environnements de développement ou des packages de classes :

- Sont-ils décomposés en objets d'une façon qui exploite au mieux les paradigmes de l'approche objet ?
- Proposent-ils une classification des objets déjà présents, facilement assimilable par l'utilisateur ?
- Comment est gérée la question de l'évolution des classes ?

Bien que centrée sur une problématique de génie logiciel affichée, cette première liste de critères est dans le cadre de notre étude loin d'être anodine. En effet, les questions que pose la lisibilité des environnements de programmation Objet sont à rapprocher des questions que posent l'accessibilité d'un modèle de l'édifice tirant parti de l'approche objet pour un interlocuteur du projet. Loin de négliger cet aspect, nous devons garder à l'esprit que nous procédons à une analyse du domaine d'application à faire partager à une communauté d'interlocuteurs peu informés sur le sujet (voire franchement sceptique). L'expression *classification […] facilement assimilables par l'utilisateur* prend donc pour nous un poids

[47] On peut se reporter à [Castellani, 1998] pour une discussion sur ce dernier point.

tout particulier. Il apparaîtra rapidement que se pose ici un problème de modélisation inhérent au formalisme choisi à ajouter à ceux déjà cités plus haut :

– Exploiter au mieux les paradigmes de l'approche objet ne se traduit pas nécessairement par une classification des objets facilement assimilables par un utilisateur non-spécialiste.

*3.3.1.ii.2. Terminologie*

Nous proposons en **annexe 10** une sélection de définitions pour les principaux termes qui sont utilisés dans ce travail[48]. Cette terminologie est présentée ainsi :

| **Terme à définir** | Définition et référence(s) |
|---|---|
| | *Termes utilisés avec le même sens* références |
| | Autres utilisations de ce terme références |

### 3.3.1.iii) Modélisation Objet

Notre démarche de description des édifices architecturaux s'appuie sur l'approche objet comme formalisme de représentation des connaissances et vise à établir un modèle architectural relatif au patrimoine bâti. Pourtant, les implémentations de ce modèle que nous proposerons dans la partie projet de ce document se basent sur l'utilisation de Langages de Programmation Orientés Objets. Quelles sont les implications de ce basculement ? Quelles sont les vicissitudes opérationnelles, au sens de [Ducournau et al, 1998], auxquelles ce modèle sémantique devra être soumis ? C'est ce que nous tentons d'établir ici en comparant démarche de modélisation objet et méthodologie de programmation dans les Langages de Programmation Orientés Objets.

En effet, nous souhaitons au travers de cet état de l'art mettre à jour les principaux problèmes de modélisation inhérents au formalisme choisi. Nous avons au chapitre "Eléments de définition, principaux concepts" déjà mis à jour un ensemble de questions. Nous cherchons ici à les compléter en comparant logiques de génie logiciel et de représentation des connaissances dans le contexte de l'approche objet.

*3.3.1.iii.1. L'approche objet comme formalisme de représentation de connaissances*

Nous avons déjà évoqué la genèse et les attendus théoriques de l'approche objet comme « *new programming paradgim* » dans les mots de T.Budd [Budd, 1996, p3], ou mode de représentation des connaissances, aux sections précédentes. Nous ne reviendrons donc pas ici sur ce point, et sur la position de l'approche objet dans les disciplines de l'intelligence artificielle pour laquelle nous renvoyons à [Schneider, 1994] ou encore [Gaillard, 1994]. En effet, notre problématique est à considérer comme une problématique d'application logicielle dans laquelle l'accent est mis sur des objectifs comme la mesure de l'édifice ou sa visualisation. Nous travaillons de fait notamment avec deux Langage de Programmation Orienté Objet: JAVA et Perl. Pourtant, nous tentons de formaliser un modèle relatif à un domaine de connaissances spécifique, modèle associant une "*dimension de comportement*" et *une "dimension structurelle-fonctionelle*" [Schneider, 1994, chapitre 7.3.1]. Et en effet, pour [Ducournau et al, 1998, p293], le but d'une représentation des connaissances est de "*rendre compte d'un domaine particulier de telle sorte que cette représentation soit manipulable* […]". Il nous semble donc nécessaire ici, partant du travail de compilation des recherches sur le domaine de la représentation de connaissances par Objets que proposent [Ducournau et al, 1998, chapitre 10], de cerner quels rapports nous pouvons établir entre notre démarche applicative assez étroite et les questions que posent ce travail.

Historiquement, la représentation de connaissances par Objets relaye d'autres modèles généraux de représentation de connaissance, notamment réseaux sémantiques et frames[49]. Vers la fin des années soixante-dix, des systèmes adoptant la notion de classe (distinguant

[48] Cette sélection présente un ensemble de vocables, leur définition et des vocables utilisés avec le même sens, ainsi qu'éventuellement en partie basse des utilisations alternatives relevées dans les sources bibliographiques citées. Cette sélection est présentée par ordre alphabétique, seuls les vocables qui apparaissent le plus fréquemment dans la bibliographie sur laquelle nous nous appuyons sont présentés comme "termes à définir".

[49] On peut, pour une discussion sur ces formalismes, se reporter à [Masini et al, 1989, chapitres 7 et 8]

classes et instances) apparaissent, s'inspirant à la fois des modes de représentation de connaissances existants et des Langages à Objets [Ducournau et al, 1998, p295]. Les systèmes de représentation de connaissances par Objets ne partagent pas de principes inflexibles et uniques mais peuvent être décrits par deux fonctions communes :

- Organisation de la connaissance. Les objets sont décrits d'une manière assez proche de celle adoptée par les Langages de Programmation Orientés Objets, c'est à dire à l'aide de classes organisées dans une hiérarchie de spécialisation. Les classes regroupent des traits communs aux objets. Un ensemble de descripteurs associés aux attributs sert deux objectifs principaux, typage des attributs et mécanismes d'inférence.
- Services inférentiels. Messages et méthodes dans les représentations de connaissances par Objets fonctionnent comme dans les Langages de Programmation Orientés Objets mais peuvent ne pas être proposés. Par contre, Langages de Programmation Orientés Objets et représentations de connaissance par Objets diffèrent par l'existence dans ces dernières de mécanismes d'inférence. Ces mécanismes permettent de calculer (par le biais de procédures, de règles, etc..) une valeur pour un attribut lorsque celle-ci n'est pas présente dans l'objet. Ils sont attachés aux classes sous forme de facettes d'inférence dont les plus communes sont relevées par [Ducournau et al, 1998, p301][50].

Ces mêmes auteurs rappellent que dans le cadre de la représentation de connaissances par Objets aussi se pose la question des conséquences de l'héritage multiple, que celle-ci fait l'objet de nombreux travaux de recherche. Ils distinguent enfin mécanisme d'inférence et mécanisme utilisant la notion de connaissance implicite. Dans ce dernier cas, l'inférence de connaissance est dite implicite, elle a lieu à la création de l'objet pour en compléter la détermination sans que l'utilisateur ait à le faire. [Ducournau et al, 1998] proposent une présentation du système de représentation des connaissances TROEPS[51] sur laquelle ils s'appuient pour poser trois problèmes distincts. Il nous importe ici de voir en quoi ces problèmes peuvent peu ou prou être posés dans notre travail.

*PROBLEMES D'EVOLUTION*

La représentation de connaissance par Objets pose le même problème d'évolution que les Langages de Programmation Orientés Objets : les objets attachés à une classe ne peuvent en changer. La question posée ici est celle du "point de vue" sur l'objet. Changer de classe n'a de sens que dans la mesure ou l'objet est décrit en fonction d'un point de vue dominant relatif à un de ses aspects[52].

[Ducournau et al, 1998] distinguent ici trois problèmes, celui de l'enrichissement de la connaissance sur l'objet (complétion de la représentation), celui de son évolution (modification de la représentation), celui de la perspective sur l'objet (représentations alternatives). Le système TROEPS contraint à distinguer l'aspect ontologique des classes et leur aspect taxinomique. Une même notion peut alors être attachée à plusieurs classes pourvu que celles-ci n'appartiennent pas à la même taxinomie.

*DEFINITION DES CLASSES*

La question posée ici est celle de la classe comme description (conditions nécessaires pour qu'un objet soit instance d'une classe) ou comme prescription (conditions nécessaires et suffisantes). [Ducournau et al, 1998] notent qu'il est souvent difficile d'établir des conditions

[50] -Valeur par défaut fixant une valeur donnée en l'absence d'autre valeur.
-Passage de valeur contraignant la valeur à celle d'un autre attribut.
-Héritage latéral (valeur du même attribut d'un objet référencé par l'objet courant).
-Filtrage (donner pour valeur d'un attribut un ensemble d'objets satisfaisant certaines contraintes).
-Attachement procédural (exécution d'une fonction retournant la valeur demandée et prenant en paramètres d'autres attributs de l'objet).

[51] Pour une présentation plus détaillée, on peut se reporter à l'adresse http://www.inrialpes.fr/sherpa/ , ou pour l'historique de ce projet à [Euzenat et al, 1995]

[52] Par exemple, la fermeture d'une baie peut être considérée par l'architecte comme un objet dont les caractéristiques essentielles sont ses dimensions, les coefficients d'isolation phonique et thermique, l'aspect et éventuellement l'histoire à l'occasion de notre projet. Pour l'industriel de la menuiserie par contre, à cette même fermeture sont avant tout attachés un numéro dans son catalogue, un prix de vente, une disponibilité, un atelier de fabrication, etc…

*suffisantes* et défendent l'interprétation descriptive contre l'interprétation définitionnelle. Il est à noter que le caractère descriptif ou définitionnel des Langages de Programmation Orientés Objets n'a pas été clairement établi.

*NOMMAGE*

Le problème des conflits de noms (d'attributs) que pose les systèmes à héritage multiple trouve dans TROEPS une solution qui consiste à remonter la définition et le nommage des attributs au niveau des concepts. Nous ne sommes pas directement confrontés à cette question puisque nous utilisons en priorité un Langage de Programmation Orienté Objet n'implémentant pas la notion d'héritage multiple.

En conclusion, nous notons ici deux nouveaux points :

- Un système de représentation de connaissances par Objets répond à deux exigences : organiser la connaissance et proposer des services inférentiels destinés à compléter l'information disponible. Si les Langages de Programmation Orientés Objets permettent eux aussi d'organiser la connaissance autour de la notion d'objet, ils n'offrent pas de solutions natives pour inférer des connaissances, implicites ou non.
- Des problèmes liés notamment à l'évolution des classes ou à l'héritage multiple se posent à la fois dans le cadre de Langages de Programmation Orientés Objets et des représentations de connaissances par Objets.

### *3.3.1.iii.2. La technologie objet comme méthodologie de programmation*

Pour [Tsichritzis et al, 1992], le formalisme objet n'est pas seulement une technique mais une approche qui peut potentiellement changer la façon dont nous construisons, pensons et utilisons des systèmes informatiques. Pourtant des notions de bases de l'approche objet (i.e. pour les auteurs encapsulation et réutilisabilité) sont reconnues bien au-delà de ses limites, mais ce qui aux yeux des auteurs rend l'approche objet si spécifique est qu'elle conduit à repenser le développement d'une application non pas en terme de travail fait sur mesure mais en terme de création de modules réutilisables.

[Tsichritzis et al, 1992] entendent démontrer successivement comment l'approche objet change la manière de programmer des applications, comment elle change leur cycle de vie, comment enfin le produit final est perçu. Les auteurs s'attachent dans un premier temps à définir sept *dimensions* qui leur servent à établir un système de classification de langages de programmation [Tsichritzis et al, 1992, p2][53].

Ce système de classification prend pour nous tout son sens quand les auteurs écrivent qu'à chaque dimension correspond un conflit fondamental de l'approche objet dont la source est à rechercher dans une application incomplète du paradigme de l'approche objet. Ce faisant, ils nous rappellent que les Langages de Programmation Orientés Objets peuvent n'implémenter qu'en partie le formalisme de représentation de connaissances qu'est l'approche objet. Nous verrons plus loin quelles conséquences cela a sur le travail de modélisation que nous proposons. Nous rapportons ici brièvement la nature de ces conflits, tels que les analysent [Tsichritzis et al, 1992].

[53] Sept dimensions :

1. Un langage basé objets permet l'encapsulation d'objets.
2. Un langage de classes permet l'instanciation d'objets à partir de classes.
3. Un Langage de Programmation Orienté Objet est un langage de classes dans lequel de nouvelles classes (sous-classes) sont produites à partir de classes existantes par héritage.
4. Un Langage de Programmation entièrement Orienté Objet est homogène, c'est à dire que tout est objet, y compris les types de base (entiers par exemple).
5. Un Langage de Programmation Orienté Objet fortement typé fournit des interfaces abstraites (types) pour tous les objets et garantit la cohérence de types des expressions.
6. Un Langage de Programmation Orienté Objet concourant permet l'utilisation de multiples processus de contrôle simultanés (que les objets soient ou non distribués).
7. Un Langage de Programmation Orienté Objet persistant permet l'utilisation d'objets qui ont une durée de vie supérieure au processus qui les a créés.

*CONFLIT ENCAPSULATION / HERITAGE.*

Les auteurs formulent ainsi la question posée ici: les méthodes des sous-classes doivent-elles avoir accès aux variables d'instance et aux méthodes héritées de la super-classe? Ce problème devient critique s'il y a évolution de la super-classe, provoquant un conflit héritage / encapsulation. La solution la plus "*commode*" est bien sûr de ne donner accès aux variables d'instances que par l'intermédiaire de méthodes publiques d'entrées-sorties. Dans ce cas toutefois toute classe a visibilité sur ces variables. Les classes ont deux rôles distincts : être instanciées ou être dérivées. Pour [Tsichritzis et al, 1992], rendre explicite les différences entre ces deux types de contrats est la seule façon de résoudre le conflit. La plupart des Langages de Programmation Orientés Objets proposent outre les qualificatifs *public* et *private* (accès ouverts à toutes les classes pour le premier, à la seule classe propriétaire pour le second) un ou des modes d'accès intermédiaires. En Java par exemple, Le mot clé *protected* désigne un membre visible depuis la classe et toutes ses sous-classes [Flanagan, 1997, p73][54].

*CONFLIT CLASSE / TYPE*

Nous ne reviendrons pas ici sur les objectifs distincts que servent les définitions d'une classe et d'un type, déjà illustrés plus haut par [Ducournau et al, 1998, pp 24-25], et qui font dire à ces auteurs que "la classe n'est pas un type". On peut se reporter à la présentation d'un Langage de Programmation Orienté Objet fortement typé, Eiffel, que ces auteurs proposent [Ducournau et al, 1998, pp 35-72]. [Tsichritzis et al, 1992] relèvent deux difficultés liées à l'intégration des notions de type et de classe dans Eiffel. En premier lieu, deux classes non reliées dans la hiérarchie de classes peuvent partager le même type. En second lieu, une sous-classe peut introduire une modification de l'interface dont elle hérite de sa super-classe, entraînant une incompatibilité de leurs types.

*CONFLIT STRUCTURE / COMPORTEMENT*

[Tsichritzis et al, 1992] formulent ainsi la question posée ici: les objets des Langages de Programmation Orientés Objets et ceux des Systèmes de Gestion de Bases de Données (Orientés) Objets ? Ces auteurs répondent par l'affirmative si l'on considère les attributs d'un objet comme des services rendus par cet objet. Autrement dit, les objets sont à considérer comme un jeu de services correspondant à des opérations à invoquer par envoi de messages ou à des attributs représentant un état de l'objet. On peut se reporter pour une discussion sur ce sujet, discussion étendue aux SGBDR ,à [Chen et al, 1995].

*CONFLIT CLASSES PREMIERES / CLASSES SECONDAIRES*

[Tsichritzis et al, 1992] font ici référence à un problème que nous avons déjà évoqué : les classes sont-elles des objets, les messages sont-ils des objets, les types primitifs sont-ils des objets ? Les auteurs appellent classes secondaires des classes outils utilisables seulement dans un contexte particulier, restriction faite pour des raisons le plus souvent pragmatiques. [Tsichritzis et al, 1992] défendent une position "tout objet" sur laquelle [Ducournau et al, 1998] émettent des réserves.

Au-delà de cette analyse, [Tsichritzis et al, 1992] s'intéressent aux bénéfices de l'approche objet en terme de développement d'applications modulaires. Ils posent comme axiome de départ que le domaine d'application soit suffisamment maîtrisé pour en autoriser l'abstraction de ses mécanismes principaux dans ce qu'ils nomment un Generic Application Framework (GAF). Cette hypothèse ne saurait être prise trop à la légère. En effet, dans une démarche de génie logiciel centrée sur un problème bien analysé, elle ne pose sans doute pas trop de

[54] La même interprétation est donnée en C++ [Stroustrup, 1996, p211]. D'autres modes peuvent être ajoutés, comme par exemple le mode *default* (visibilité sur tout le package) en Java [Flanagan, 1997, p73], ou les méthodes *friend* (accès à la partie privée d'une classe indépendante) en C++ [Stroustrup, 1996, p163].

problèmes. Il n'en va pas de même dans une démarche de représentation de connaissances plus évolutive, comme nous tenterons de le montrer par la suite[55].
La démarche des auteurs nous intéresse plus particulièrement sur deux points. En premier lieu, le processus qu'ils décrivent correspond en large partie avec l'architecture du langage JAVA, postérieur à leur article (packages génériques orientés tâches, base d'informations logicielles évolutive javadoc [Flanagan, 1997, pp233-234]). En second lieu, la notion de réutilisabilité qu'ils défendent va au-delà de la production de modules génériques individuels puisqu'elle intègre l'idée de problèmes génériques dont la solution passe par l'utilisation d'un jeu de modules. Cette idée est donc à rapprocher de la contribution de [Castellani, 1998].
En résumé, bien que la production d'applications fortement réutilisables ne soit pas au cœur de notre problématique, il est clair que la détermination d'une méthodologie de programmation tirant parti du formalisme choisi ne peut en rester absente. En particulier, il convient à la lumière de ce qui vient d'être dit de retenir deux points clés :

- La nécessité d'évaluer les choix présentés par le Langage de Programmation Orienté Objets.
- La notion de réutilisabilité sous l'aspect du couple problème générique / application spécifique.

#### 3.3.1.iv) Problèmes de modélisation

Nous entendons ici résumer les éléments de discussion sur lesquels nous devrons revenir en présentant notre travail. En effet, nous avons d'ores et déjà identifié, en nous appuyant sur les travaux de recherches cités, un ensemble de problèmes de modélisation liés à l'approche objet. Nous reviendrons plus en détail sur les questions d'évolutivité du modèle et de persistance dans la section suivante. Nous procèderons ici en rapportant derrière les notions d'objets de classe et d'héritage à la base du formalisme objet les problèmes identifiés pas à pas dans notre discussion.

##### *3.3.1.iv.1. Objets*

L'objet informatique présent dans un modèle tel que nous le formalisons est conçu comme réactif, doté d'un comportement qui tend à le rapprocher de l'objet réel. On l'a évoqué, [Tsichritzis et al, 1992] défendent une position "tout objet" qui dit que si le modèle se veut cohérent, tout (connaissance du domaine, objets outils de l'application, types primitifs) doit être objet. A l'inverse, pour [Ducournau et al, 1998, p24] dans un univers de connaissance, il existe des objets immuables comme par exemple les nombres. Ces auteurs notent donc que dans le cadre des Systèmes de Gestion de Bases de Données (Orientés) Objets, il faut prendre garde à la distinction objets / valeurs présente dans la plupart des Langages de Programmation Orientés Objets (qui implémentent de façon atypique des classes outils pour les nombres, les chaînes de caractères, etc..). Ceci pose par l'exemple la question du processus de réification, l'opération fondatrice du modèle. On l'a vu, cette opération doit intégrer la notion de point de vue sur l'objet réel [Gaillard, 1994]. Plus précisément, à l'objet réel peut correspondre un unique objet informatique associant plusieurs points de vue sur l'objet réel, ou plusieurs objets informatiques construits en fonction du point de vue à prendre en compte. Dans ce second cas, on peut cependant se demander où se situe l'aspect ontologique de l'objet. Il faut donc dans le contexte de cette étude distinguer point de vue de la modélisation (celui de l'architecture au sens de la détermination d'un corpus d'objets physiques, aspects ontologiques du modèle) et points de vue de l'usage du modèle (gestions de données hétérogènes relevant de spécialités différentes). Schématiquement, nous réifions au nom du premier et nous composons au nom du second.

[55] [Tsichritzis et al, 1992] définissent leur concept de Generic Application Framework comme suit : un GAF a pour vocation de rendre le travail du développeur d'applications aussi facile que possible. Le GAF encapsule des modules réutilisables factorisant des problèmes génériques de développement d'applications. Les auteurs décrivent ensuite un cycle de travail qui inclut :
-Factorisation dans un GAF de composants réutilisables à partir d'applications existantes et de connaissances du domaine.
-Détermination d'un Specific Application Frame (SAF) à partir du GAF et des spécifications de l'application.
-Evolution du SAF en fonction des évolutions des exigences de l'application.
-Evaluation et réécriture du GAF en fonction de l'expérience.

Troisième élément de réflexion, un modèle objet associe une "*dimension de comportement*" et *une "dimension structurelle-fonctionelle*" [Schneider, 1994]. L'objet est ici l'encapsulation d'un jeu de méthodes qui peuvent être invoquées par le monde extérieur et d'un jeu de données qui retiennent les effets de ces méthodes. Cette interface, seule visible pour l'utilisateur, règle le comportement de l'objet, et l'interaction entre objets. Ceci pose de fait deux problèmes : la validité de l'appel de méthode adressé à l'objet (à son interface) et le statut du message[56].

*3.3.1.iv.2. Classes*

Nous avons présenté la classe comme moule de fabrication d'objets singuliers appelés instances. Nous avons également établi le constat selon lequel une instance ne peut pas changer de classe. Ce problème est à rapprocher d'un problème d'évolution du modèle puisque la demande est en fait de modifier les éléments de définition d'un objet a posteriori. Nous détaillons cette question dans la section suivante. La notion de classe pose néanmoins d'autres problèmes dont nous allons discuter ici. En premier lieu, une classe est définie en intension. Or toute définition d'un individu est également définition de sa famille. Comment dès lors réifier à bon escient, pour reprendre les termes de [Ducournau et al, 1998], si chaque individu observé peut définir une famille d'objets isomorphes ? Le passage de l'observation (d'objets physiques ou de descriptions théoriques) à une description intensionnelle d'abstractions les représentant (en exploitant au mieux les paradigmes de l'approche objet) est bien sûr le cœur de notre démarche.

Par ailleurs, la notion de métaclasse introduit une autre question : la classe est elle un outil qui fabrique des instances? On doit ici clairement distinguer les notions de méta-classe [Ducournau et al, 1998, pp28-29], de classes génériques [Stroustrup, 1996, pp 257-293] et de génération de classes [Drap et al, 1999b]. Dans le premier cas, la classe est une instance d'une métaclasse, manipulable comme telle à l'exécution. Dans le second cas, implémenté en C++, la classe générique définit un module paramétré. Dans chaque instance de cette classe, un type spécifique est obtenu en substituant dans la classe générique chaque paramètre par le type voulu. Enfin, dans le troisième cas, sur lequel nous reviendrons dans la partie projet de ce document, la nouvelle classe construite l'est avant la compilation, et prend naturellement la forme d'une classe normale du système (un texte qui rassemble une liste de noms, les attributs, et une listes de méthodes).

Enfin, Oscar Nierstrasz posait dès 1989 (voir [Nierstrasz, 1989]) un problème concret qu'il nous faut prendre en compte : l'accessibilité d'un modèle objet pour un interlocuteur du projet. La lisibilité pour l'utilisateur à la fois de la définition comportementale et structurelle des objets présents dans le système et de l'arbre de classes construit reste un point clé pour évaluer le modèle. En effet, si encapsulation et héritage sont des éléments fondateurs du paradigme objet, ils sont également par essence créateurs d'ambiguïtés (pour l'utilisateur des classes). A quels messages l'instance d'une classe sait-elle répondre ? Quel ancêtre d'une classe se charge d'y répondre ? Comment l'objet répond-il à un message ? Nous considérons que la validité d'un modèle objet ne saurait être évaluée sans prendre en compte la question de la réutilisabilité des classes produites[57].

*3.3.1.iv.3. Héritage*

Nous avons établi les principes du mécanisme d'héritage comme moyen à la fois d'organiser hiérarchiquement des concepts en procédant par ajout de propriétés, et de partage de

[56][Blair et al, 1991] définissent ainsi le premier : l'objet requérant ne sait pas si l'objet requis peut effectivement répondre, ou comment il le fera. Le second fait l'objet de nombreuses recherches dans des directions très variées comme en témoignent par exemple [Ducournau, 1997] ou [Seinturier et al, 1998]. La question posée est celle de la réflexivité du système. Cette réification des messages est aujourd'hui plus ou moins disponible selon les systèmes, et son absence empêche tout raisonnement aisé sur le comportement de l'objet.

[57] Nous entendons ici par réutilisabilité la capacité à rendre plus facile la compréhension, l'extension et la maintenance du modèle et des applications développées pour le manipuler. Une notion de réutilisabilité étendue est présentée par [Tsichritzis et al, 1992] sous l'aspect du couple problème générique / application spécifique. Cet aspect rejoint pour nous la question de l'abstraction de modèles de solutions sur laquelle nous reviendrons dans la partie méthodologie, et pour laquelle on peut se reporter à [Castellani, 1998] ou [Maughan et al, 1998].

comportement entre objets. La définition d'une hiérarchie de classes pose bien sûr, on l'a vu, la question de la propagation des modifications apportées à une classe, problème d'évolution du modèle que nous traitons plus loin. Au delà de ce point, l'implémentation du mécanisme d'héritage offerte par la plupart des Langages de Programmation Orientés Objets induit une spécialisation des concepts non par différences mais par ajout de propriétés. Il s'agit comme le notent [Ducournau et al, 1998, p22] d'une particularisation provenant uniquement d'une complication des concepts. Ces auteurs, on l'a vu, établissent clairement les limitations de ce mécanisme d'héritage. Mais ils établissent également que si cette solution reste partielle, elle correspond néanmoins à un besoin de hiérarchisation que l'on retrouve dans toutes les sociétés humaines. Une attitude pragmatique devra ici être adoptée pour contourner la difficulté en substituant par exemple des relations d'agrégations aux relations de spécialisation là ou cela ne remet pas en cause la sémantique du modèle. L'héritage est donc utilisé comme un mécanisme de structuration des connaissances. L'héritage simple permet à une sous-classe d'hériter des variables d'instances et des méthodes de la classe dont elle hérite et d'ajouter les siennes propres. L'héritage multiple autorise une sous-classe à hériter de propriétés héritées de plusieurs classes. L'héritage partiel (implémenté en C++) ou l'héritage multiple semblent au premier abord confortables pour le programmeur. Pourtant, en relation avec le problème de la lisibilité des classes évoqué précédemment, ces mécanismes posent un évident problème de cohérence dans la hiérarchie de classes construite.
Par ailleurs, on l'a vu, pour [Ducournau et al, 1998] l'héritage n'est pas du sous-typage. Rappelons ici rapidement que le langage JAVA sur lequel nous reviendrons introduit le principe d'une séparation classe / type par le biais de ses classes "*interface*". Une classe qui implémente un type fournit un ensemble d'attributs et des corps de procédures qui doivent réaliser la signature. Avec la notion d'héritage se pose le problème de la redéfinition de méthodes dans une hiérarchie, c'est à dire du rapport d'inclusion des types des signatures de méthodes héritées. Notons ici que le langage JAVA interdit tout simplement la modification du profil d'une méthode redéfinie par rapport à sa définition initiale. Enfin, nous renvoyons à [Castellani, 1998] pour une discussion sur la notion d'héritage de construction dans un ensemble de catégories d'objets distinguant objets sémantiques et objets de servitude.

#### 3.3.1.v) Evolutivité et persistance

Nous avons isolé la question de l'évolution du modèle des autres problèmes de modélisation cités ci-dessus. Ce choix s'explique par trois constats :

- La question de l'évolutivité est transversale par rapport aux principes de l'approche objet : elle se décline dans les notions de classe, d'héritage, de modularité et en définitive dans l'idée même d'objet, réinterprété comme entité encapsulant données et comportement *à un instant t* de l'effort de modélisation.
- L'évolution des classes rejoint l'exigence de réutilisabilité citée par [Nierstrasz, 1989] et [Wu et al, 1995] comme un des bénéfices essentiels à attendre de l'approche Objet.
- Notre projet présente par ailleurs trois aspects pour lesquels cette question se pose de façon aiguë :
  - Le domaine de connaissance traité (le patrimoine architectural) est encore peu abordé au travers du formalisme objet, nous donnant peu d'indices sur comment réifier.
  - Les objectifs (ou points de vue utilisateur) de notre travail recoupent des spécialités diverses, nous imposant une remise en cause des résultats liée à des demandes évolutives.
  - Notre collaboration avec une équipe de conservateurs nous pousse à prendre en compte un ensemble de connaissances évolutives puisque liées à un *état courant* de la recherche sur les édifices traités.
    Nous devons par conséquent nous efforcer de proposer un formalisme et des outils permettant l'enrichissement et la remise en cause ponctuelle du modèle, par exemple un générateur de classes sur les détails duquel nous reviendrons.

On l'a évoqué, [Ducournau et al, 1998] distinguent trois problèmes d'évolution, celui de l'enrichissement de la connaissance sur l'objet (complétion de la représentation), celui de son

évolution (modification de la représentation), celui de la perspective sur l'objet (représentations alternatives). Nous tentons ici de les cerner avec plus de détail. Nous revenons également à la fin de cette section sur la question de la persistance dans les Langages de Programmation Orientés Objets.

*3.3.1.v.1. Enrichissement de la connaissance*

L'enrichissement de la connaissance sur l'objet correspond au besoin qui apparaît lorsque l'on dispose d'informations plus spécifiques sur l'objet représenté. Cet enrichissement nécessite un affinement de la représentation qui se traduit par exemple par une spécialisation du concept (notion de migration d'instance par raffinement expérimentée dans notre projet). Dans ce cas, l'instance d'une classe A dont hérite la classe B devient instance de B pourvu que le programmeur prévoie une procédure d'initialisation des attributs définis dans la classe B. Le problème de la compatibilité des signatures des méthodes ne se pose pas dans notre cas puisque nous avons expérimenté ce mécanisme en JAVA, langage qui interdit la redéfinition du profil des méthodes re-déclarées. Bien que ce mécanisme soit une première réponse au problème de la migration d'instance, il faut bien reconnaître que cette solution n'est que très partielle puisqu'elle ne permet que de descendre dans l'arbre d'héritage. Restent donc à implémenter des solutions plus globales traduisant l'enrichissement de la connaissance par une migration latérale d'instances par exemple. Reste également, comme le notent [Blair et al, 1991], que la plupart des Langages de Programmation Orientés Objets implémentent l'ajout et la modification de comportements mais pas leur suppression. Cela, traduit dans notre contexte, signifierait qu'un enrichissement ne peut passer que par une surdétermination de l'objet. Ce point nous paraît essentiel à rapporter car il introduit une interprétation de la notion d'enrichissement : évolution par différentiations. Et si l'amélioration d'un modèle passait par son expurgation? Le mot évolution n'est donc pas dans le sens que lui donnent [Blair et al, 1991] un terme adéquat : il s'agit en fait plutôt d'extension du modèle.

*3.3.1.v.2. Evolution de la connaissance*

L'évolution de la connaissance sur l'objet correspond au besoin qui apparaît lorsque l'objet change et nécessite une modification de sa représentation. On est donc ici dans le cas d'une redéfinition de l'objet et non dans le cas de la détermination d'un nouvel objet à insérer dans l'arbre d'héritage. Cet objet se situe donc n'importe où dans l'arbre d'héritage (arbre en cas d'héritage simple, ce qui est notre cas).
Cette question est soulevée par Oscar Nierstrasz [Nierstrasz, 1989] sous le terme de *propagation des modifications* dont l'auteur rappelle qu'il se pose avec d'autant plus d'acuité dans le cadre des Systèmes de Gestion de Bases de Données (Orientés) Objets où les modifications doivent être propagées non seulement aux classes mais aussi aux instances. [Tsichritzis et al, 1992] explorent, on l'a vu précédemment, une voie qui consiste à abstraire un jeu de problèmes génériques encapsulés dans des modules réutilisables. Leur démarche s'appuie néanmoins sur un axiome dont nous devons comprendre les conséquences : le domaine d'application est suffisamment maîtrisé pour autoriser l'abstraction de ses mécanismes principaux. Cette hypothèse se traduit par l'idée que le domaine d'application est non seulement maîtrisé mais également stable, autrement dit peu évolutif. Autrement dit, le découplage problème générique / application spécifique ne s'impose comme solution partielle au problème de l'évolution de la connaissance que si celle-ci est … peu évolutive[58].
Enfin, il faut nous rappeler de ce qu'est la notion d'objet : entité représentant un objet du monde réel et encapsulant données et méthodes. La question posée ici est donc la suivante:

[58] Concrètement, il n'est pas toujours, loin s'en faut, impossible de faire évoluer la représentation d'un concept par modification de sa détermination, quelle que soit sa position dans l'arbre d'héritage. Les données ajoutées ne posent souvent pas de problème: on rejoint ici le point précédent. Les données supprimées posent des questions plus difficiles qui peuvent néanmoins être résolues partiellement en faisant appel à des classes intermédiaires ou à des relations d'agrégations. La modification du type d'une donnée nous replace dans le cadre de ce travail "fait main" auquel l'approche objet était censée selon [Nierstrasz, 1989] mettre un terme. Des solutions partielles sont proposées par certains langages de programmation pour ajouter un niveau de généricité à la définition des classes comme les patrons de classe de C++ [Stroustrup, 1996, pp 257-293] ou pour découpler types et classes comme les interfaces de JAVA [Flanagan, 1997, pp77-81]. On peut également se reporter à [Bellahsène et al, 1998] pour une introduction à la notion de vue utilisée pour masquer un attribut à supprimer.

l'objet du monde réel évolue t'il ou est-ce notre perception, notre connaissance sur cet objet qui évolue? Ainsi nous en revenons au constat fait par [Ducournau et al, 1998]: la réification est la clef de voûte de l'approche objet, et l'implémentation de mécanismes destinés à redéfinir un objet s'apparente dès lors à un ensemble de béquilles fournies à un modèle boiteux. A la différence des notions d'enrichissement de la connaissance et de perspectives sur l'objet, inhérentes au fait même de traiter d'un domaine d'application complexe, la notion d'évolution par redéfinitions des objets identifiés dans un système d'objets pose d'abord un problème de validité du modèle. Il nous semble donc relever d'une problématique de méthodologie de modélisation plutôt que d'une problématique de langages et d'implémentations.

*3.3.1.v.3. Perspective sur l'objet*

Une perspective sur l'objet correspond au besoin qui apparaît lorsque l'objet peut être vu sous différents aspects. Ce point de vue sur l'objet, que l'on préfèrera nommer perspective sur l'objet, peut nécessiter un changement de représentation. [Ducournau et al, 1998, p305] note que ce point est mécaniquement résolu dans les systèmes de représentation de connaissance par l'utilisation de la multi-instanciation ou de la multi-généralisation. Les méthodes d'analyse et de conception par objets donnent une interprétation de la notion de points de vue comme par exemple les *modules* de OMT[59] qui regroupent des classes pour matérialiser une perspective sur une situation [Ducournau et al, 1998, p110]. Concrètement, dans le cas des langages de Programmation Orientés Objets, chaque langage peut fournir des moyens différents de prendre en compte la notion de points de vue. On pourra se reporter à [Ducournau et al, 1998, pp49-53] pour une discussion sur son interprétation en Eiffel par exemple. Nous reviendrons plus en détail sur cet aspect dans la section de ce document consacrée aux méthodes de modélisation.
En effet, la notion de perspective sur l'objet est à considérer sous le double aspect de l'implémentation (qui dépend du langage) et de la nécessité exprimée au travers de ces méthodes. Une perspective sur l'objet sera donc une représentation partielle de cet objet. Une représentation *multiple* [Bellahsène et al, 1998] de l'objet se traduit par la possibilité donnée à un objet d'être manipulé de façon différente en fonction du contexte. Données et comportement de l'objet sont sélectionnés en fonction de l'utilisation attendue. Partant d'une comparaison entre notions de points de vue et de vues en base de données, [Bellahsène et al, 1998] démontrent comment simuler des perspectives sur l'objet (représentations multiples) à partir de vues dans les systèmes de gestion de bases de données.

*3.3.1.v.4. Persistance*

Nous citons ici la notion de persistance car celle-ci ne peut être évoquée dans le seul contexte des Systèmes de Gestion de Bases de Données (Orientés) Objets. En effet, [Ducournau et al, 1998, pp130-131] rappellent qu'on trouve dans l'approche objet des systèmes qui assurent simplement la gestion d'objets persistants (relevant de la famille des Langages de Programmation par Objets persistants) par opposition aux systèmes intégrant gestion des objets et programmation d'applications (relevant de l'approche Systèmes de Gestion de Bases de Données).
Damian Conway définit ainsi une notion de programmation persistante ("*persistent programming*" [Conway, 2000, p387]), qui pour lui signifie la capacité des objets à conserver leur nature, leur identité, leur état entre les exécutions du programme qui les a crée. L'auteur rappelle qu'un Langage de Programmation doit pour assurer la persistance des objets fournir quatre services: identité des objets, sérialisation, stockage, synchronisation. Il détaille ensuite l'implémentation Perl de la notion de persistance à laquelle nous renvoyons [Conway, 2000, pp 387-428]. Nous reviendrons par ailleurs sur cette implémentation en comparaison avec d'autres Langages de Programmation Orientés Objets dans la section suivante.

[59] Nous reviendrons sur OMT et les méthodes de modélisation plus loin dans ce document

### 3.3.2. Les langages de programmation Orientés Objets.

Nous avons largement discuté des principes fondamentaux de l'approche objet et de certains problèmes de modélisation qu'elle peut poser. Il est donc temps d'en revenir aux "*vicissitudes opérationnelles*" dont [Ducournau et al, 1998, p5] nous indiquaient qu'elles nous éloignent des concepts et formalismes supposés indépendants de l'implémentation. Venons en donc dans cette section à ces implémentations, et vérifions comment sont traduits par les Langages de Programmation Orientés Objets du projet les principes établis jusqu'à présent.
Nous nous appuierons pour ce faire sur une présentation du langage JAVA. Nous détaillerons quelques choix d'implémentation faits et en tirerons les conséquences en terme de contraintes de modélisation. Nous nous essaierons à quelques comparaisons avec les langages C++ et Perl, respectivement utilisés antérieurement à JAVA et en parallèle (notamment applications pour le Web). Nous commencerons en conséquence par établir une grille d'analyse comparative. Nous n'avons bien sûr pas ici pour ambition d'émettre des critères d'évaluation ou de qualification tels que les proposent part exemple [Nierstrasz, 1989] ou [Tsichritzis et al, 1992]. Nous nous contenterons d'établir les principales implémentations de caractéristiques "objet" communes à ces langages, leurs points de divergence essentiels, et quelques aspects liés aux problèmes de modélisation que nous venons d'évoquer et qui nous intéressent plus directement.

#### 3.3.2.i) Généralités

Nous avons isolé un ensemble de concepts fondamentaux communs aux Langages de Programmation Orientés Objets implémentant le modèle à classes et instances, et en premier lieu l'idée d'objet, les notions de classe et d'héritage, ainsi que, pour [Conway, 2000, p2], le polymorphisme et l'envoi de message. Ceci ne constitue pas une grille d'analyse comparative mais plutôt un ensemble d'exigences de bases pour un Langage de Programmation Orienté Objet. Nous allons donc devoir déterminer avec plus de précision ce qui différencie ces langages, au-delà de leurs interprétations de ces cinq points. Nous nous appuierons sur une bibliographie dont nous allons donner les références (ouvrages uniquement) sous forme de liste. Ceci nous évitera de surcharger le texte par des références pour chaque point abordé. Les ouvrages que nous citons sont centrés sur un langage ou sur un aspect de ce langage, tel qu'explicité ci-dessous:

| *Langage* | *Référence de l'ouvrage* | *Thèmes principaux de l'ouvrage* |
|---|---|---|
| *JAVA* | *[Flanagan, 1997]* | *Guide de référence du langage, didacticiel pour le passage de C++ à JAVA, description des packages et des classes des API (Application Programming Interfaces) JAVA (JDK 1.1).* |
| | *[Englander, 1997]* | *Ouvrage consacré à la programmation des Java Beans, modèle à composants propre à Java. Il détaille notamment le modèle évenementiel du langage et le mécanisme de sérialisation des objets.* |
| | *[Harold, 1997]* | *Livre consacré à JAVA pour le développement d'applications réseau (gestionnaires de protocoles, applets, API, et RMI).* |
| *C++* | *[Stroustrup, 1996]* | *Ecrit par l'auteur du langage C++, l'ouvrage inclue une présentation des choix d'implémentation, une discusssion sur la conception par objets ainsi qu'un manuel de référence du langage.* |
| *Perl* | *[Christiansen et al, 1999]* | *Guide de référence du langage Perl, exemples et solutions, accès à des bases de données, communication entre processus, services internet.* |
| | [Conway, 2000] | Premier ouvrage dédié aux seuls aspects "objet" du langage Perl, incluant également des comparaisons succintes avec les langages Smalltalk, Eiffel, C++ et Java. |
| | *[Brenner et al, 1996]* | *Programmation perl pour les communications Internet client / serveur basées sur les protocoles HTTP et CGI.* |

Nous n'entendons pas ici établir des comparaisons de syntaxe mais bien de mettre en lumière des choix d'implémentation. En le faisant, nous souhaitons en fait anticiper sur la détermination des raisons qui nous ont poussé à privilégier les langages JAVA puis Perl, l'héritage simple, etc…Nous établirons donc une grille d'analyse, et l'appliquerons dans

l'**annexe 11** au langage JAVA, puis en comparaison à C++ et Perl. Nous précisons que cette grille ne présente pas de façon exhaustive les caractéristiques de ces langages mais reprend, ou en tout cas s'efforce de reprendre, les points essentiels à noter dans le cadre de notre effort de modélisation.

*3.3.2.i.1. Proposition pour une analyse comparative*

Le tableau ci-dessous donne les termes de notre grille d'analyse comparative, avec en partie gauche l'intitulé des termes de l'analyse (classés par thème), en partie droite les questions correspondantes.

| | | |
|---|---|---|
| Objets | Encapsulation des variables | Quelles alternatives de contrôle d'accès sont offertes pour les variables d'instances ? |
| | Encapsulation des méthodes | Quelles alternatives de contrôle d'accès sont offertes pour les méthodes ? |
| | Déclaration de variables | Ou doivent être déclarée les variables ? |
| | Déclaration de méthodes | Ou doivent être déclarée les méthodes ? |
| | Allocation –désallocation | Comment sont gérées les allocations et désallocation ? |
| | Constructeurs | Comment sont écrits les constructeurs et quelles sont les conséquences de l'appel à un constructeur ? |
| | Destructeurs | Comment se fait la destruction de l'objet, est-elle nécessaire ? |
| | Valeurs de retour (méthodes) | Doit-on fixer une valeur de retour pour toute méthode de l'objet ? |
| | Type des variables | Les variables sont-elles typées ? Quels types systèmes le langage offre t'il ? Comment sont traités les types primitifs ? |
| | Agrégations | Quels mécanismes pour l'agrégation d'objets ? |
| | Persistance | Quelles solutions pour rendre les objets persistants ? |

| | | |
|---|---|---|
| Classes | Déclaration de classe | Quelle forme doit prendre une déclaration de classe? |
| | Variables de classe | Quelles formes et quelles règles d'accès pour les variables de classe (données collectives) ? |
| | Méthodes de classe | Quelles formes et quelles règles d'usage pour les méthodes de classe ? |
| | Classes abstraites | Une implémentation de la notion de classe abstraite est-elle proposée ? |
| | Méthodes virtuelles | Existe t'il un formalisme de méthodes virtuelles (redéfinition de méthodes) ? |
| | Typage et généricité | Un mécanisme de spécification d'une famille de classes sans définition du type des variables de cette classe est-il offert ? |
| | Portée | Comment les classes, variables et méthodes peuvent-elle être partagées en dehors du fichier les contenant ? |

| | | |
|---|---|---|
| Héritage | Simple / multiple | Quel forme d'héritage le langage implémente t'il ? |
| | Polymorphisme d'héritage (Redéfinition de méthodes) | Les modifications de profil des méthodes redéfinies sont-elles autorisées ? |
| | Polymorphisme d'interface (Redéfinition de méthodes) | Comment sont gérées les exceptions liées à la redéfinition de méthodes dans le cas de polymorphisme d'interface ? |
| | Surcharge d'opérateurs | Les opérateurs peuvent-ils être redéfinis dans le contexte d'une classe ? |

| | | |
|---|---|---|
| Types | Typage | Quelle forme de typage des variables le langage propose t'il ? |

| | |
|---|---|
| Gestionnaire d'exceptions | Comment le système gère t'il les exceptions ? |

### 3.3.2.ii) Le langage JAVA

JAVA est un langage objet plus ou moins proche de C++, mis au point au début des années quatre vingt dix par James Gosling et son équipe chez Sun Microsystems. JAVA se voulait un langage orienté objet reprenant les caractéristiques principales du C++ mais à la fois portable (indépendance des plates-formes matérielles) et moins encombrant. C'est un langage presque purement objet, comme on le verra. En 1994, compte tenu de l'importance grandissante du réseau Internet et du besoin de portabilité des applications qui en résultait, l'équipe de conception de JAVA décida de mettre au point un navigateur (qu'ils appelèrent HotJava) intégrant Java et capable de faire fonctionner des applets (des petites applications fonctionnant sous un navigateur). Rendu public en 1995, Le langage JAVA a connu un important succès du en partie au soutien de Sun Microsystems et de Netscape. Ce succès est également lié au fait que plusieurs caractéristiques critiques (du point de vue de l'utilisateur) de C++ ont été supprimées en JAVA[60]. JAVA est néanmoins réputé moins rapide que C++[61]. Enfin, le langage autorise à la fois la création d'applications classiques (programmes typiques .exe), et la création d'applets, applications dédiées au Web (fichiers .class) dont l'appel peut être inséré dans du code HTML standard. Nous allons dans un premier temps passer en revue les aspects les plus spécifiques de ce langage :

- machine virtuelle Java,
- applets,
- kit de développement,
- packages et API,
- gestion des exceptions,
- gestion d'évènements,
- multithreading,
- programmation réseau.

Nous présenterons ensuite dans leurs grandes lignes les choix d'implémentation de ce langage. Cette présentation sera complétée par l'application de la grille d'analyse du langage introduite ci avant.

#### *3.3.2.ii.1. Aspects fondamentaux du langage*

Le fichier source d'un programme écrit en Java est un simple fichier texte dont l'extension est par convention "*.java*". Ce fichier source doit être un fichier texte non formaté, à compiler. La première particularité de JAVA est que contrairement aux langages compilés comme C++, pour lesquels le compilateur créer un fichier binaire directement exécutable par un processeur donné, le code source Java est compilé en un langage intermédiaire (appelé pseudo-code ou bytecode). Ce pseudo-code est exécutable sur une machine virtuelle JAVA (parfois appelée interpréteur Java) tournant sur une plate-forme donnée, et capable d'interpréter le code intermédiaire. C'est par cette particularité que les auteurs du langage réalisent sa portabilité.
On l'a évoqué, Java permet de créer deux types de programmes : les applications (s'exécutant dans le système d'exploitation à condition d'avoir installé une machine virtuelle); les applets (applications destinées à fonctionner sur un navigateur). Une applet a dès lors un champ

[60] -Plus de pointeurs.
-Pas de surcharge d'opérateurs.
-Pas héritage multiple.
-La libération de mémoire est transparente (il n'est plus nécessaires de créer de destructeurs).
-Une meilleure gestion des erreurs.
-Les chaînes et les tableaux sont désormais des objets faisant partie intégrante du langage.

[61] Voir par exemple sur ce point la position de P.A Péclier *: "comparées aux programmes C+ +, les applets sont environ vingt fois plus lentes*" (http://www.chez.com/webpap/), celle de Y. *Bergeron "les temps d'exécution d'applications Java sont plus élevés que ceux des mêmes applications écrites en C ou C++*" (http://www.virtuel.collegebdeb.qc.ca/info/yvesb/inf803-n/c12_803.htm) ou celle de M. Rémon. "*L'ouverture à tous les systèmes d'exploitation n'a pu se faire qu'en divisant le rôle du compilateur en deux[…]. Cela s'est fait au détriment d'une rapidité d'exécution et d'une efficacité que seul un compilateur dédicacé à un type précis de machine et de système d'exploitation peut avoir*" (*http://www.fundp.ac.be/~mremon/java/notes_cours/cours_java_1.html*) ou encore les contributions mentionnées en note de bas de page, p111.

d'action plus réduit qu'une application traditionnelle Java (une applet ne peut par exemple pas accéder au système sur lequel elle s'exécute).
En soutien à l'expansion du langage, la firme Sun Microsystems distribue un ensemble d'outils de développement appelé JDK (Java Development Kit). Ce Kit de développement comprend un compilateur Java, un interpréteur d'applications (machine virtuelle), un interpréteur d'applets (applet Viewer), un débogueur , un décompilateur, un générateur de documentation et un compresseur de classes.
Langage encore récent, Java a subi plusieurs évolutions[62]. Sur chaque plate-forme matérielle coexistent aujourd'hui plusieurs versions du JDK, plus ou moins compatibles avec les navigateurs standards. Ceci ne va pas bien sûr sans poser un problème de portabilité renouvelé.
Le compilateur javac transforme le code source en bytecode, un fichier binaire intermédiaire interprétable par la machine virtuelle sur n'importe quelle plate-forme (dans les limites évoquées précédemment). L'interpréteur java est une machine virtuelle fonctionnant en mode texte, c'est-à-dire sans interface graphique L'interpréteur d'applets permet de visualiser l'exécution d'un applet sans avoir recours à un navigateur. L'utilitaire Javadoc permet de créer une documentation automatisée au format HTML à partir de la définition des classes, grâce aux commentaires placés dans le code source. Enfin, le compresseur de classes permet de réduire la taille de classes Java à télécharger.
Les classes JAVA sont contenues dans des packages organisés sous forme de hiérarchie. L'emplacement des fichiers sources correspondant à la hiérarchie des packages doit être précisé dans une variable d'environnement appelée classpath, qui donne la liste des chemins d'accès aux classes. L'instruction import permet de spécifier le chemin d'accès relatif à la classe. Java définit deux packages par défaut, toutes les classes ne faisant pas partie explicitement d'un package et situées dans les répertoires accessibles, le package java.lang contenant les classes standards telles que System ou Math. Le principe d'organisation des packages est lié à la possibilité que souhaite offrir le langage d'enrichir son jeu de classes par des modules indépendants, téléchargés sur Internet. Autre conséquence de ce souhait, JAVA n'autorise pas la définition de variables ou de méthodes globales. Vingt trois packages de base forment l'Applications Programming Interface (API) standard livré dans la version JAVA 1.1. Ces packages contiennent les classes définissant aussi bien les objets de base du langage que les objets images, évènements, applets, etc.. Nous renvoyons à la bibliographie citée précédemment pour plus de détail. Il est important toutefois de noter ici que l'organisation des classes adoptées par JAVA a ouvertement pour vocation de faciliter un travail de développement en équipe.
Pour traiter les erreurs, Java propose un mécanisme d'exception consistant à effectuer les instructions dans un bloc d'essai (le bloc try) qui surveille les instructions. Lors de l'apparition d'une erreur, celle-ci est interceptée et lance un bloc de traitement d'erreur (le bloc catch, avec pour paramètre le type pour l'objet Exception).
JAVA est un langage autorisant l'exécution simultanée de plusieurs processus, ou threads. Le package java.lang contient une classe nommée Thread qui par exemple fournit des méthodes de démarrage et d'arrêt de threads ou encore définit leur priorité. Ce mécanisme est très largement utilisé en JAVA, en particulier dans toutes les applets animées.
Les interfaces graphiques en JAVA s'appuient sur les classes du package AWT (Abstract Window Toolkit). Ces classes étaient dans la version 1.0 du langage fournies avec des gestionnaires d'évènements. La version 1.1 du langage introduit un changement majeur puisque la gestion des évènements est déléguée à un ensemble de classes spécifiques, d'une part classes événement (sous-classes de Event) et des classes récepteurs des événements (sous-classes de EventListener). Les événements renseignement l'application sur d'une part leur existence et d'autre part leur contenu éventuel (position de la souris par exemple). Les évènements gèrent les actions ou sélections de l'utilisateur, le comportement des fenêtres, ou encore le focus sur un composant graphique. Les Listeners, récepteurs de ces événements sont des classes interfaces dont les méthodes sont chargées d'analyser l'événement. Les objets

[62] Ou peut-être devrait-on dire plutôt "les programmeurs JAVA ont subi plusieurs évolutions du langage"…

graphiques sont donc considérés comme sources d'évènements, ils sont dotés de méthodes permettant d'ajouter ou de retirer un récepteur correspondant aux évènements. Les classes définissant des objets graphiques sources d'évènements implémentent l'interface des classes récepteur.
Enfin, nous abordons avec la programmation réseau en JAVA un des points clés qui expliquent le succès actuel de ce langage. Nous allons en citer ici brièvement quatre raisons, qui recouvre en fait quatre catégories de classes servant d'interface de programmation pour le réseau Internet: les classes URL, Socket, Servlet et RMI.
La classe URL propose une interface de haut niveau pour accéder aux informations du Web en utilisant des URLs. Les constructeurs de la classe URL permettent de créer un objet de la classe URL à partir du protocole, de la machine, du port et du fichier ou à partir d'une chaîne de caractères brute. La classe URLConnection permet alors de construire des objets pouvant accéder aux informations contenues dans l'URL (Un objet de cette classe est construit par l'appel à la méthode openConnection() sur un objet de la classe URL). La Classe URLConnection est une classe abstraite qui propose une interface simple pour manipuler les entêtes et le contenu d'une URL, d'une façon indépendante du protocole.
Au delà de l'accès à des URL, la classe Socket de JAVA fournit tous les outils nécessaires à l'ouverture d'une connexion TCP/IP depuis une applet ou une application, donnant accès à un flux d'entrée et un flux de sortie, comme avec un fichier.
En troisième lieu, les servlets de JAVA, composants de serveur, indépendants du protocole, sont l'équivalent sur le serveur des applets sur le client. Les interfaces que fournit le langage peuvent être dérivés pour écrire les méthodes service appelées à chaque requête dans une nouvelle thread.
Enfin, le package java.rmi contient des mécanismes de RMI (Remote Method Invokation) permettant d'invoquer de façon simple des méthodes d'objets distribués sur le réseau Internet. JAVA fournit un ensemble d'outils et de classes prédéfinies qui rendent l'implantation d'appels de méthodes et d'objets possibles en utilisant le mécanisme standard de sérialisation de JAVA. Les RMIs sont basées sur la notion de Stub/Skeleton: Du coté client, le Stub, substitut de l'objet distant, doit implémenter la même interface que celle définie par l'objet distant. Il transforme l'appel de méthode en une suite d'octets à envoyer sur le réseau (Marshaling). Le résultat reçu est reconstruit sous le même format (Unmarshaling). Le format d'un appel de méthode contient l'identificateur de l'objet distant, l'identificateur de la méthode et les paramètres sérialisés. La valeur de retour contient le type (valeur ou exception) et la valeur sérialisée. Du coté serveur, le skeleton d'un objet distant est l'entité qui reconstruit les paramètres, trouve l'objet appelé, appèle la méthode et retourne le résultat[63].
En ce qui concerne JAVA, nous avons désormais situé les principaux aspects qui font l'attrait de ce langage. Mais nous citerons également, dans la conclusion de cette, quelques autres aspects de nature à modérer notre enthousiasme.

#### *3.3.2.ii.2. Choix d'implémentation*

**Les objets et les types primitifs :** En JAVA tous les éléments manipulés sont des objets, c'est-à-dire des instances de classes. Tout ou presque puisque JAVA est un langage fortement typé qui dispose de types de données systèmes qui ne sont pas des objets : short int et long (entiers), float et double (réels), boolean, char (caractère). Le langage fournit pour contourner la difficulté des classes dites enveloppeurs définissant des objets pouvant contenir un type primitif et auxquels sont associées des méthodes permettant de les manipuler [64]. Les chaînes de caractères sont instances d'une classe particulière appelée String. La chaîne de caractère est donc un objet possédant des attributs et des méthodes.
**Le typage :** Les conversions de type de données (ou transtypage) peuvent être implicites (modification du type de donnée effectuée par le compilateur) ou explicite (appelée aussi

[63] On pourra également s'intéresser à la norme CORBA édictée par l'Object Management Group (OMG)[63] qui vise à spécifier les fonctionnalités d'un bus logiciel pour les applications distribuées.
[64] En voici la liste correspondante: Short (short) Integer (int) Long (long) Float (float) Double (double) Boolean (boolean) Character (char) Void (void).

opération de cast). Dans ce dernier cas JAVA dispose d'opérateurs dit de cast pour spécifier la conversion.
**Les structures conditionnelles :** Les structures conditionnelles classiques sont disponibles en JAVA, ainsi que l'opérateur ternaire (condition) ? instruction si vrai : instruction si faux.
**L'encapsulation :** L'accessibilité aux propriétés de l'objet est organisée en quatre niveaux (reprenant les principes d'encapsulation de C++). Un élément déclaré public est accessible sans restriction. Un élément déclaré protected est accessible uniquement aux classes d'un package et à ses sous-classes. Un élément déclaré private est accessible uniquement dans sa classe. Ces éléments ne peuvent alors être manipulés qu'à partir de méthode accesseurs. Enfin, Un élément est déclaré par défaut avec un niveau d'accessibilité appelé "package visibility" (accessible uniquement à sa classe et aux classes du même package).
**Les variables :** La déclaration de variables spécifie bien sûr leurs types et peut être accompagnée de leurs initialisations. L'opérateur de résolution de portée :: permet d'accéder à des variables globales plutôt qu'à des variables locales, lorsqu'il en existe portant le même nom.
**Les méthodes :** Pour peu que des méthodes aient des arguments différents (en type et/ou en nombre), JAVA autorise leur redéfinition. Une déclaration de méthode doit obligatoirement spécifier un type de retour (sauf pour les constructeurs et destructeurs), elle peut bien sûr comporter des paramètres, auxquels peuvent être affectées des valeurs par défaut.
**L'instanciation :** L'instanciation d'objets se fait en JAVA par appel à l'opérateur new sur la classe à instancier (sur son constructeur). L'opérateur new fait appel au constructeur de la classe qui peut initialiser les variables de la classe, et permettre différentes actions définies par le programmeur lors de l'instanciation. Un constructeur peut avoir des arguments, il doit porter le même nom que sa classe mais n'a pas de type de retour (pas même void). La définition d'un constructeur n'est pas obligatoire puisqu'un constructeur par défaut est défini par le compilateur Java si la classe n'en possède pas. JAVA autorise la définition de plusieurs constructeurs ayant des nombres ou types d'arguments différents. Chaque objet instancié est identifié par un élément appelé handle, concept proche de celui des pointeurs en C++, et permettant d'accéder aux propriétés des objets. Le mot clé this permet de désigner l'objet dans lequel on se trouve. L'objet courant this est en réalité une variable système, passée en tant que paramètre caché de chaque méthode de l'objet.
**L'héritage:** De premier abord , l'héritage en JAVA est un héritage simple. Une classe ne peut hériter que d'une superclasse[65].. Le mot clé super permet de désigner la superclasse comme this permet de faire référence à l'objet courant. La redéfinition de méthodes n'est possible qu'en respectant la signature de la méthode héritée (même nombre et même type d'arguments que la méthode de la superclasse).
La grille d'analyse présentée en **annexe 11** explique ou complète ces choix d'implémentation.

### 3.3.2.iii) Comparaisons avec C++ et Perl

C++ et Perl (version 5) ont en commun d'avoir été conçus comme Langages de Programmation Orientés Objets à partir d'un langage procédural existant. Pour le reste, il nous faut commencer ici par introduire en quelques mots les principes qui ont guidé la définition de ces langages avant de leur appliquer notre grille d'analyse. Dans la section suivante, nous mettrons, en terme de conclusion à ce chapitre sur les Langages de Programmation Orientés Objets, l'accent sur les avantages et inconvénients comparés de JAVA, C++ et Perl.
Conçu par Bjarne Stroustrup au début des années quatre-vingts, C++ est devenu un langage de référence tant dans l'industrie que dans l'enseignement. Son succès est sans doute du en partie au souci de compatibilité avec le langage C sur lequel il s'appuie : C++ ne devait pas obliger les programmeurs C à changer leurs habitudes. C++ est donc du C auquel sont

[65] On le verra, le mécanisme d'interface permet souvent de simuler l'héritage multiple par un lien d'implémentation: la classe implémentant un interface définit l'implémentation des méthodes abstraites contenues dans la définition de l'interface. Une classe JAVA ne peut donc hériter de l'implémentation de méthodes que de sa superclasse, mais peut également hériter de méthodes abstraites de ses interfaces

ajoutées des facilités autorisant l'abstraction de données et la programmation par objets. C++ a été conçu comme un langage sans vocation à devenir un système intégrant un environnement de programmation comme Smalltalk. C++ est un langage à typage statique fort : les variables, paramètres et résultats de méthodes doivent faire l'objet de déclarations précisant leur type. La déclaration d'une classe en C++ est assimilée à la déclaration d'un type utilisateur s'ajoutant aux types prédéfinis. Elle permet au compilateur de traiter l'allocation mémoire et le contrôle de types.
Plusieurs exemples peuvent être rapportés pour illustrer le style de programmation hybride que peut autoriser C++. En premier lieu, une application C++ peut inclure des fonctions extérieures à tout objet (bloc main() par exemple). En second lieu, la définition d'une classe ne constitue pas forcément une unité syntaxique : elle peut typiquement être répartie sur plusieurs fichiers source.
En terme d'encapsulation de propriétés, C++ permet la modification par n'importe quelle méthode de toute variable public d'une classe, et propose un mécanisme de classes amies permettant d'exporter toute la partie privée d'une classe vers une autre classe. L'héritage multiple est implémenté en C++ (avec désactivation manuelle de la duplication de données en cas d'héritage en diamant), tout comme est proposée une forme de généricité des classes à travers la notion de patrons de classe. C++ est construit en deux couches: le langage C et les classes. La notion de classe n'est en définitive qu'un outil parmi d'autres en C++, au même titre que la surcharge des opérateurs, l'héritage multiple, la généricité, etc… Pour [Ducournau et al, 1998, p101], il s'agit là à la fois d'un avantage et d'un inconvénient car masquant les principes fondamentaux de la programmation par objets.

Inventé par un linguiste, Larry Wall, au milieu des années quatre-vingt , le langage Perl est appelé par son auteur un langage "humble". Il le décrit comme un langage à plier au besoin de son utilisateur. Il était au départ conçu pour répondre exclusivement au problème de traitement des chaînes de caractères. Perl (Practical Extraction and Report Language) est un langage interprété, donc a priori plus lent que du code compilé.
Perl a été décrit comme un langage de haut niveau, indépendant de la plate-forme, pour la programmation système. Il a aujourd'hui largement débordé de ce cadre et sert notamment dans un bon nombre d'applications phare pour le réseau Internet (Altavista, Yahoo, Amazon, Lycos, etc..).
Perl n'a pas été conçu comme un Langage de Programmation Orienté Objet mais en implémente les principes dans sa version 5. C'est un langage à typage dynamique fournissant un mécanisme de ramasse-miettes automatique, pour les objets à désallouer. Une classe Perl est définie par un ensemble de méthodes et de variables, mais celles-ci sont généralement déclarées à l'intérieur du constructeur de la classe. D'autres points clés peuvent être notés pour situer ce langage :

- Pas de contrôle statique de type (pas de classes génériques).
- Héritage multiple.
- Règles d'encapsulation laissées aux soins du programmeur.
- Polymorphisme implicite.
- Variables et méthodes de classes, surcharge d'opérateurs.

Perl n'est pas un langage tout objet, loin s'en faut. La couche objet ajoutée à Perl rend l'appel de méthodes plus lent. La surcouche objet de Perl est selon [Conway, 2000] à l'image du langage lui-même : souple et peu respectueux des conventions.

Les grilles d'analyse comparative des langages C++ et Perl sont placées en **annexes 12** et **13**.

### 3.3.2.iv) Conclusion

On doit d'abord ici remarquer que les deux Langages de Programmation Orientés Objets les plus utilisés pour la programmation d'applications sur Internet, JAVA et Perl, se situent bizarrement à l'opposé l'un de l'autre en terme de position face au formalisme objet. JAVA est

un langage récent, "tout objet" (ou presque) dans lequel les principes du formalisme objet sont appliqués avec rigueur. Perl est tout l'inverse: un langage dans lequel les principes du formalisme objet sont appliqués avec peu de rigueur. Entre ces deux extrêmes sur l'emploi desquels nous allons revenir, C++ propose sa démarche à la fois hybride et contraignante. Elle est hybride car la syntaxe de C++ s'inspire largement de C, et le langage autorise par exemple la présence de blocs d'instructions en dehors de tout contexte d'objet. Elle est contraignante car par exemple l'héritage répété impose au programmeur de déclarer des classes virtuelles, ou encore parce qu'il faut définir explicitement des destructeurs pour récupérer l'espace libéré par les objets obsolètes (pas de ramasse-miettes).

Plus important sans doute dans notre contexte, C++ fournit un ensemble complexe d'outils concrets ou conceptuels (classes génériques, surcharge d'opérateurs, opérations de conversion de types utilisateurs, héritage multiple, etc..) qui enrichissent le langage mais nuisent à sa lisibilité et peut encourager les programmeurs à ignorer les principes de base de l'approche Objet. Notre objectif, rappelons le, est de nous appuyer sur le formalisme objet à la fois en temps que moyen de représenter des connaissances et en temps que moyen de produire des applications. Il ne faut par conséquent pas négliger dans une telle approche la clarté du modèle élaboré au profit d'une écriture cryptique efficace mais conduisant à la production de code peu réutilisable ou peu extensible. On peut se reporter pour une discussion sur ce point à [Ducournau et al, 1998, p102].

Nous avons choisi en 1997 de basculer l'ensemble des développements élaborés jusqu'alors en C++ vers JAVA. Ce choix était au départ lié essentiellement à notre souhait d'une part de travailler sur un langage multi plate-forme et d'autre part de porter nos applications sur Internet. Parallèlement, nous avons mené un certain nombre de travaux en utilisant le langage Perl (version 4.0) pour de petites applications sur Internet. Ce langage, lui aussi indépendant des plates-formes, nous a servi plus tard (version 5.0) dans le cadre de travaux en collaboration avec des partenaires extérieurs. Il a donc été utilisé pour évaluer ou développer des applications alternatives à l'intérieur du projet. Nous allons ici à la lumière de notre (petite) double expérience tenter de dresser un bref bilan des utilisations optimales à attendre de ces langages.
Nous souhaitons d'abord rapporter le travail de comparaison effectué par L. Bois et rapporté sur le site Web de l'ENSTA[66]. Il distingue :

- Langages interprétés travaillant sur les objets au niveau du système, facilement portables, en général beaucoup plus lents que tout autre langage machine natif ou byte code interprété.
- Langages tels que Perl, présentant des caractéristiques communes avec JAVA comme la robustesse, le comportement dynamique et la neutralité de l'architecture.
- Enfin, langages compilés de haute performance tels que C et C++, dont l'auteur donne comme inconvénient les coûts de débuggage et la difficulté d'implémenter et d'utiliser les possibilités de multithreading.

Le langage JAVA se positionne ici en compromis entre les langages de haut niveau, les langages scripts portables, mais lents, et les langages compilés, non portables mais rapides. Ce travail est conclu par le tableau comparatif proposé sur le document http://www.ensta.fr/java/java_perf.html, recensant les caractéristiques globales de plusieurs langages (notamment des trois cités précédemment), tableau dont nous donnons ci-après une version résumée.

| | JAVA | Perl | Smalltalk | Shell | C++ |
|---|---|---|---|---|---|
| Simple | ■ | ▲ | ■ | ▲ | |
| Orienté Objet | ■ | ■ | ■ | | ▲ |
| Robuste | ■ | ■ | ■ | ■ | |
| Sécurisé | ■ | ■ | ▲ | ▲ | |

[66] Ecole Nationale Supérieure des Techniques Avancées, http://www.ensta.fr/java/java_perf.html

| | | | | | |
|---|---|---|---|---|---|
| Interprété | ■ | ■ | ■ | ■ | |
| Dynamique | ■ | ■ | ■ | ▲ | |
| Portable | ■ | ■ | ▲ | ▲ | ▲ |
| Neutre | ■ | ■ | ▲ | ▲ | |
| Threads | ■ | ■ | | | |
| Garbage Collection | ■ | | ■ | | |
| Exceptions | ■ | ▲ | ■ | | ▲ |
| Performances | Haute | Moyenne | Moyenne | Basse | Haute |

| | |
|---|---|
| ■ : La caractéristique existe | ▲ : La caractéristique existe parfois |

Il n'est pas certain que la forme que prend ce tableau ne reflète pas autant les préférences de son auteur que les réelles différences entre ces langages. Le langage C++ reste dans la littérature une référence en terme d'outil de programmation[67]. Néanmoins ce tableau nous apporte une indication intéressante par les intitulés des caractéristiques comparées. Ceux-ci recoupent en fait un ensemble de critères d'usages assez liés aux problématiques du développement d'applications multi-plateforme pour le réseau Internet. Il ne constitue pas pour autant une justification du choix de tel ou tel langage. Pour ce faire il faut dans notre cas repenser notre projet sous un double aspect:

- D'une part, une problématique de modélisation objet du corpus architectural.
- D'autre part, un besoin de développement d'applications portables pour le réseau Internet.

Les langages Java et Perl répondent de façon assez complémentaire à ces deux problèmes. En terme de modélisation, Perl pèche par une trop grande liberté laissée au programmeur (modes d'encapsulation par exemple). Cette liberté peut se traduire sur un gros projet par une moins bonne modularité puisqu'elle repose sur la seule rigueur des programmeurs. Une bonne programmation Perl implique discipline et rigueur de la part du programmeur pour qui la souplesse du langage peut s'avérer un piège autant qu'un avantage. A l'échelle de ce projet, ceci ne constitue pas un réel frein. Moins souple que Perl, Java impose un style de programmation tout objet qui en facilite la lecture. Par contre, les choix d'implémentation fait par ce langage (héritage simple par exemple) imposent au programmeur de penser son problème pour Java.
Java comme Perl autorisent une forme de modélisation objet du corpus architectural.
En terme de développement d'applications pour le réseau, Perl et Java offrent des solutions assez différentes. Perl est le langage de référence pour l'écriture de scripts CGI. Java dispose en plus d'un ensemble de classes (API) qui permettent de construire des interfaces graphiques pour le Web ou indépendamment, et autorise le multithreading. On peut globalement dire que ce qui est fait en Perl peut l'être en Java mais pas l'inverse (mais globalement seulement). [Benett, 1997] résume en une formule le choix Perl: "*Si des programmes sont écrits pour aider les gens à faire les choses plus vite, mieux et moins cher, alors Perl est le langage de programmation pour les programmeurs*". Il s'agit bien d'une formule que l'auteur s'attache à dépasser en expliquant d'abord que Perl, loin d'être *un "toy language*" a rejoint en terme de caractéristiques dans une large mesure Java. Il rappelle ensuite que la programmation en Perl demande rigueur et sophistication, et que leur absence a des conséquences d'autant plus graves que le projet grossit. Enfin, il donne une indication rentrant dans le cadre de notre interrogation: de grands projets ou des processus de gestion d'évènements complexes ne doivent sans doute pas être implémentés en Perl. Cela laisse un champ d'application qui comprend en particulier l'écriture de scripts CGI ou les tâches administratives. Java comme Perl autorise le développement d'applications réseau pour Internet, avec tout de même une grande différence quand aux moyens mis à la disposition du programmeur en terme d'interfaçage, de gestion des évènements et de multithreading, bien supérieurs en Java.

[67]Voir par exemple l'article d'Eric Galyon "C++ vs Java Performance" à l'adresse suivante: http://www.cs.colostate.edu/~cs154/PerfComp/, ou celui de Carmine Mangione "Performance tests show Java as fast as C++" à l'adresse http://www.javaworld.com/javaworld/jw-02-1998/jw-02-jperf.html

Nous avons établi ici d'abord comment trois Langages de Programmation Orientés Objets différents implémentent les notions essentielles de l'approche objet. Nous avons ensuite évoqué quelles sont les raisons pour lesquelles nous avons choisi Java et dans une moindre mesure expérimenté Perl. Il nous restera dans la partie de ce document consacrée au projet à voir concrètement ce que ces choix ont impliqué en terme de modélisation et de production d'applications pour le Web. Pour l'heure nous allons aborder la question qui vient naturellement en contrepoint à l'implémentation de l'approche objet par telle ou telle plateforme de programmation: les Langages de Modélisation Objet dont la vocation est de formaliser un problème en amont de son implémentation.

### 3.3.3. Les langages de modélisation Orientés Objets.

La diversité des implémentations que les Langages de Programmation Orientés Objets proposent pour les concepts fondamentaux de l'approche objet conduit naturellement à ré-interroger le propos de [Wu et al, 1995]: "*La technologie objet* […] *rend plus facile le développement, la maintenance et la réutilisation d'un grand nombre d'applications*". Qu'en est-il si les acteurs d'un projet, auteurs de modules de code à réutiliser, n'ont pas comme bagage commun un Langage de Programmation Orienté Objets particulier ? Qu'en est-il s'ils doivent échanger non seulement une analyse du projet mais également intégrer dans cette discussion les spécificités d'un langage qu'ils n'utilisaient pas jusqu'alors ? Les méthodes d'analyse et de conception Objet, introduites ici sous le terme "langages de modélisation Objet" s'inscrivent en réponse à cette question. Nous allons dans un premier temps établir à la fois les exigences de bases des méthodes d'analyse et de conception par Objets et les champs de recherche ouverts sur ce thème, puis discuterons plus en détail un certain nombre de formalismes faisant aujourd'hui référence.

#### 3.3.3.i) Analyse et conception par Objets : introduction

La phase d'analyse d'une application logicielle orientée objet a pour [DeChampeaux et al, 1993] comme objectif de clarifier le cahier des charges de l'application à partir de trois évaluations et d'une notion:

- Intrants: spécification d'objectifs qui peut être imprécise.
- Extrants: description exhaustive des caractéristiques et des comportements attendus.
- Technique: un choix d'implémentation.
- Objets: la notion clé à intégrer dans les descriptions.

Une analyse complète et cohérente permet pour ces auteurs de détecter et de réparer dès cette phase les erreurs dans la spécification du cahier des charges avant qu'il ne devienne trop coûteux d'y remédier. A partir de ce constat, les méthodes d'analyse et de conception par objets tentent de concilier besoin de représentation de haut niveau et capacités des Langages de Programmation Orientés Objets. Les méthodes d'analyse et de conception par objets sont à rapprocher des techniques de réutilisation (design patterns, frameworks, composants logiciels)[68] sur lesquelles nous ne reviendrons pas, compte tenu de l'échelle de notre projet. Elles sont issues de méthodes de génie logiciel antèrieures et leur ajoutent des traits spécifiques. Pour [Ducournau et al, 1998], la complexité croissante des applications a motivé l'élaboration de méthodes de conception, et avec elles l'établissement d'un cycle de vie du logiciel décrit ainsi [Ducournau et al, 1998, p104] :

- La définition des besoins (domaine de l'application, fonctionnalités indispensables, etc…)
- La phase d'analyse (organisation et décomposition des informations issues de la phase précédente sans référence à une technique d'implémentation). Pour [DeChampeaux et al, 1993] , elle fournit une description de ce que l'application doit faire.

[68] On peut se reporter pour cette question par exemple à [Dodani, 1999], [Borne et al, 1999], à l'article de Douglas C. Schmidt "Using Design Patterns to Develop Reuseable Object-Oriented Software" à l'adresse http://www.cs.wustl.edu/~schmidt/OOWG-statement.html ou à la bibliographie présentée à l'adresse http://www.hillside.net/patterns/papers/

- La phase de conception (détermination des éléments à implémenter pour réaliser l'application en prenant en compte la technique d'implémentation).
- Le codage de l'application (réalisation dans un langage de Programmation).

Partant de cet acquis, les méthodes d'analyse et de conception par objets apparues à la fin des années quatre vingt intégraient plusieurs points nouveaux, notamment la notion d'objet et celle de spécialisation.
La phase dite de définition des besoins s'appuie souvent dans les méthodes d'analyse et de conception par objets sur d'autres méthodes[69]. Les phases d'analyse et de conception dans les méthodes d'analyse et de conception par objets se trouvent rapprochées par l'utilisation des mêmes concepts. Pour [DeChampeaux et al, 1993] le paradigme objet tend à intégrer les phases d'analyse, de conception et d'implémentation, avec pour risque d'apporter une confusion masquant leurs différences.
Une méthode d'analyse et de conception par objets produit un ensemble de *modèles* selon un *processus* [Ducournau et al, 1998, p105]:

- Le *modèle* est une vue du système décrite par une notation (diagramme et formulaires), il est le résultat de l'activité d'analyse et de conception.
- Le *processus* décrit la démarche à adopter pour produire différents modèles.

Un nombre important de méthodes d'analyse et de conception par objets ont été proposées et sont décrites dans la littérature relative à ce sujet[70]. Nous présentons en **annexe 14** les principes d'une méthode d'analyse et de conception par objets de référence, OMT (Object Modelling Technique), et indiquons ci-dessous le constat à partir duquel a été spécifié le standard UML (Unified Modelling Language). Nous renvoyons aux chapitres 1.6 et 5 de [Oussalah, 1997] pour notamment une présentation synthétique de plusieurs méthodes d'analyse et de conception par Objets citées ici en note de bas de page.

### 3.3.3.ii) La notation UML

UML (Unified Modeling Language) est un langage d'analyse et de conception par objets basé principalement sur les méthodes OMT, Booch et OOSE. Son développement a commencé en 1994 avec pour objectif de gommer les différences superficielles de notation ou de terminologie entre les méthodes dont il est le successeur[71].
UML fait l'objet de nombreuses recherches pour les références desquelles on peut se reporter à [Hitz et al, 1998] ou [Clark et al, 1998] par exemple. Nous allons ici rapporter les principaux éléments de la notation UML et en conclusion en évaluer l'importance dans le cadre de notre projet. Les notations qu'offrent les méthodes d'analyse et de conception par objets fournissent des mécanismes aptes à représenter les objets et leurs interactions pour atteindre un objectif commun, les fonctionnalités d'un système. UML, comme suite à des contributions antérieures, apporte trois avantages :

[69] Voir sur ce point la méthode d'analyse et de conception par objets BON http ://www.eiffel.com/products/bon.html
[70] OMT (Object Modelling Technique) " Object-Oriented Modeling and Design" , Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., , Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1991.
OOD (Object Oriented Design ) "Object-Oriented Analysis and Design with Applications" Booch, G., The Benjamin/Cummings publishing Company Inc., Redwood City, California, USA, 1994.
OOAD (Object Oriented Analysis and Design), "Object Oriented Analysis (2nd Edition)", Coad, P., and Yourdon, E., Yourdon Press, Englewood Cliffs, New Jersey, USA, 1991; "Object Oriented Design", Coad, P., and Yourdon, E., Yourdon Press, Englewood Cliffs, New Jersey, USA, 1991.
FUSION, "Object-Oriented Development : The Fusion Method", Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., Jeremaes, P., édité Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1994.
BON (Business Object Notation) : "Seamless Object-Oriented Software Architecture - Analysis and Design of Reliable Systems", Kim Walden et Jean-Marc Nerson, édité par Prentice Hall 1994, ISBN 0-13-031303-3
OOSE (Object-Oriented Software Engineering), " Object-Oriented Software Engineering", Jacobson, I., Christerson, M., Jonsson, P., Övergaard, G., édité par Addison-Wesley, 1992.
[71] Voir http://www.rational.com/uml

- La standardisation des notations permet de se focaliser sur les problèmes essentiels de la modélisation que sont la sémantique du modèle, le processus de développement et les outils adéquats.
- UML intègre les use cases, c'est à dire la définition des spécifications utilisateurs du système.
- Les différentes vues d'un modèle qu'autorise UML permettent d'appréhender un système complexe en terme de caractéristiques essentielles.

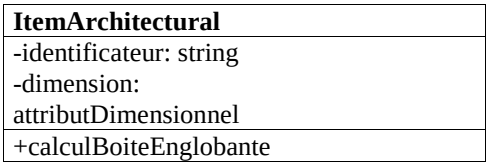
UML offre des notations diverses pour les phases principales du développement d'applications. Par contre, il laisse de côté les problèmes d'applications distribuées et de gestion de données. On peut se reporter à [Hitz et al, 1998] pour des propositions sur ces domaines. L'objectif d'UML est de fournir un langage standard qui pourrait être utilisé en remplacement des méthodes telles qu'OMT. Cet effort d'unification est selon [Evans et al, 1998] proche d'aboutir tant dans le domaine de la littérature sur la modélisation objet que chez les fournisseurs d'outils de génération de code (CASE tools). UML est par ailleurs un standard de l'OMG (Object Management Group, déjà cité).

*3.3.3.ii.1. Modélisation des données en UML*

Nous présentons ici très brièvement quatre éléments de la notation UML avant d'en discuter l'utilisation dans le cadre de notre travail.

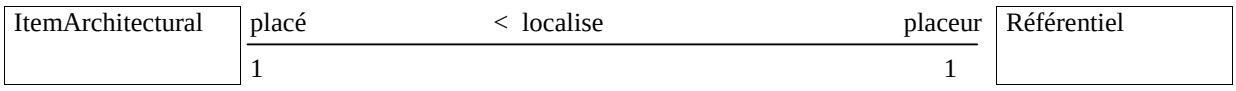
*DIAGRAMMES DE CLASSE*

La classe décrit un ensemble d'objets dont les attributs, représentés par des chaînes de caractères, ont une visibilité particulière (+ Public, # Protégé, - Privé)



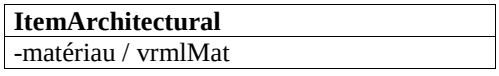

*ASSOCIATIONS*

Les associations notent les relations entre classes en indiquant les classes associées, le nom d'association et le nom des rôles d'association, y compris une cardinalité. Les association binaires sont représentées par des lignes, les associations ternaires (et plus) par un diamant.



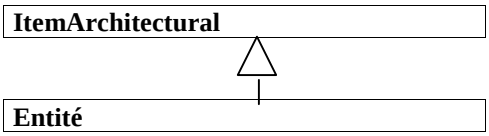

*ÉLEMENTS DERIVES*

L'élément dérivé, calculé à partir d'un autre, est représenté par le signe slash devant le nom de l'élément qui sert à le calculer.




*SPECIALISATION*

Les relations d'héritage s'expriment classiquement par une flèche reliant l'élément fils à l'élément père.



*3.3.3.ii.2. Perspectives d'utilisation*

UML comme ses prédécesseurs s'inscrit dans une démarche visant à fournir une base méthodologique à l'application des concepts de l'approche objet : modéliser une application selon une vision objet. UML a pour vocation d'être à la fois un langage de modélisation et le support d'une analyse objet, avec pour objectifs notamment de représenter des concepts abstraits, de limiter les ambiguïtés et de faciliter l'évaluation comparative de solutions. UML est donc à la fois un langage de modélisation objet et un cadre méthodologique qui permet d'élaborer des modèles objet indépendamment de tout langage de programmation. Les solutions sont exprimées sous forme de diagrammes expliquant visuellement les choix d'organisation du modèle. Ces diagrammes représentent un aspect précis du modèle, ils permettent d'interroger le modèle selon différentes perspectives. Ces apports indéniables ne doivent cependant pas cacher quelques réalités qu'il est bon de rappeler ici :

- UML est un langage qui permet de représenter des modèles, mais il ne définit pas le processus d'élaboration de ces modèles.
- UML n'apporte pas de support dans la phase d'analyse d'un problème (définitions des besoins).
- Les notations graphiques des modèles UML peuvent être sujettes à interprétation (voir à ce propos [Evans et al, 1998] et plus généralement les travaux du groupe precise UML - pUML[72]).
- UML est en particulier utile comme support de communication au sens ou il permet à différents acteurs de mettre en discussion des solutions avec un minimum d'ambiguïtés de langage. Cet avantage a néanmoins un revers: les acteurs doivent conjointement faire le choix d'UML. Lorsque ces acteurs échangeaient déjà sur la base d'une connaissance partagée d'un langage de programmation, ce choix induit un surcroît d'investissement.
- Enfin, La disponibilité des outils de génération de code vers tel ou tel langage de programmation Orienté Objet est un point dont on doit surveiller l'évolution.

Dans le cadre de ce projet, la nécessité d'avoir recours à UML ne s'est pendant longtemps pas clairement faite sentir. A cela nous pouvons fournir quatre explications:

- L'utilisation d'un Langage de Programmation Orienté Objet unique rendait inutile le recours à une notation comme vecteur de communication du projet.
- La thématique assez étroite de ce projet à son origine rendait peu probable la nécessité d'exprimer au delà du cercle de ses concepteurs les principes de modélisation adoptés.
- L'échelle de ce projet rendait tout à fait improbable l'acquisition d'un outil de génération de code et par conséquent nous laissait avec la double tâche d'exprimer le modèle en UML et de l'implémenter dans un Langage de Programmation particulier.
- Notre projet s'intéresse à trois directions pour lesquelles UML n'apporte pas encore aujourd'hui de solutions valides : applications distribuées, gestion de données et de définition des exigences attendues du modèle.

En résumé, on peut dire que nous avons dans un premier temps jugé le coût d'UML supérieur aux avantages attendus. L'évolution de notre projet nous conduit désormais à remettre en cause cet a priori et à nous appuyer "au coup par coup" sur la notation UML. Solution intermédiaire, elle traduit clairement la nécessité d'inscrire ce point comme une possible perspective pour la suite de notre travail.

### 3.3.4. Formalisme objet : Bibliographie

Nous proposons, classée par ordre alphabétique, une liste des références citées dans cette partie de l'état de l'art à la fin du présent document, section "bibliographie thématique" (section 7.2.2).

[72] The precise UML group, adresse web : http://www.cs.york.ac.uk/puml/

## 3.4. PROBLEMATIQUES DE LA CONSERVATION

### 3.4.1. Relevé architectural

Nous avons expérimenté une approche du relevé architectural dans laquelle un processus de mesure photogrammétrique dédié vient informer un modèle architectural sous-jacent. Pour replacer cette contribution dans le contexte plus général de la mesure en architecture nous proposons en **annexe 15** d'aborder les méthodes de relevé des édifices par un bref historique puis un état des techniques actuelles. Nous reportons donc en annexe 15 la présentation de quatre techniques de relevés utilisées en architecture : les GPR, la topographie, le balayage laser et la photogrammétrie. Nous portons bien sûr une attention particulière aux outils développés aujourd'hui pour exploiter le relevé photogrammétrique d'édifices, et discuterons ainsi dans cette section seulement de la pratique de la photogrammétrie architecturale aujourd'hui.
Le relevé architectural s'inscrit dans une démarche intentionnelle : éclairer un questionnement sur l'édifice. Placé au centre d'une documentation textuelle et figurative, son objectif sera d'aider le chercheur à dresser l'état descriptif, reconstituer la chronologie et replacer éventuellement l'édifice dans l'histoire des formes et des processus culturels. Analytique, cette mise en forme reflète ainsi l'état des connaissances et reste soumise à des révisions ou des approfondissements ultérieurs.
Deux attitudes s'opposent dans la constitution d'un dossier documentaire:

- collecte d'informations amorphes, diffuses, dont la masse même constitue le handicap.
- collecte ciblée, méthodique, autour de critères choisis pour décrire et expliquer l'objet, dont l'adéquation au niveau de connaissances d'un temps T constitue la limite.

Nous devons cependant commencer ici par un point sur lequel nous ne reviendrons plus ultérieurement : les objectifs d'une campagne de mesure. En effet, comme le rappelle Jean-Paul Saint Aubin, tout relevé commence par l'établissement d'un cahier des charges en fixant la portée : "*La fonction du relevé doit être explicite dès avant la réalisation* [StAubin, 1992,p18]" . Ainsi les relevés qu'effectue Andrea Palladio [Palladio, 1738/1965, livre 4] au seizième siècle sur les monuments de la Rome Antique n'ont-ils pas pour vocation d'en établir un inventaire ou même de les décrire toujours exhaustivement. Ils lui permettent de montrer par l'exemple comment les anciens concevaient tel ou tel arrangement d'éléments du corpus, comme par exemple l'entablement du temple de la paix érigé par l'empereur Claude et terminé par l'empereur Vespasien de retour de Judée [Palladio, 1738/1965, livre 4, page 86][73].
En citant ici Palladio nous souhaitons avant d'aborder le relevé comme mesure d'une morphologie le replacer dans le contexte de son utilisation. En effet, il peut devoir rapporter au-delà du dimensionnement d'objets leur provenance, leur composition (type de matériau), le sens d'un motif, etc.. Cette préoccupation est bien sûr à l'origine de notre démarche visant à lier la mesure et le modèle sous-jacent.
Jean-Paul Saint Aubin donne du relevé ces définitions [StAubin, 1992, p32] :

- Collecte de mesures visant à connaître les dimensions et formes d'un objet et les relations de ses diverses parties, et à fixer numériquement dans un espace de référence les points mesurés sur l'objet. Cet espace de référence est un choix de plans sur lesquels sera projetée graphiquement l'image de l'objet. Fixer numériquement un point correspond à déterminer les distances de ce point à ses projections sur les plans.

[73] L'auteur note que figurent notamment dans sa décoration des éléments empruntés au temple de Jérusalem. Dès Palladio donc apparaît dans le relevé de l'édifice l'idée d'adjoindre à la représentation des éléments du corpus leur origine, leur réemploi. De même lorsque l'auteur décrit le célèbre Panthéon de Rome, il prend position sur son origine :*"Il fût édifiéselon l'opinion de certains par M.AGRIPPA autour de l'an 14. Mais je crois que le corps du temple fut construit à l'époque de la république, et que M.AGRIPPA lui ajouta seulement le portique, ce qui peut être compris par les deux frontispices qui sont placés sur le devant* [Palladio, 1738/1965, livre 4, page 99]. Palladio pratique le relevé architectural non seulement en notant quels éléments de corpus lui apparaissent comme de réemploi mais également en émettant sur leur datation ou appartenance des hypothèses.

- Analyse quantitative et qualitative de l'objet à partir de données objectives visant à établir, entre points, des relations quantitatives (mesure de la distance les séparant) ou qualitatives (forme de la ligne joignant les points et valeur des zones que la ligne partage).
- Production éventuelle d'un document graphique qui soit, projeté dans un plan déterminé, l'image réduite de l'objet dans un certain rapport.

L'auteur ajoute que les techniques traditionnelles du relevé sont strictement quantitatives, alors que les techniques photogrammétriques établissent à la fois un positionnement relatif des points mesurés sur l'édifice et les relations formelles qui les lient. Enfin, il introduit ainsi les rapports relevé / objectifs : "*la pratique du relevé* […] *se situe au lieu commun résultant de la définition de l'objet - c'est à dire de sa théorisation dans un système pertinent d'analyse- et de la définition du degré de précision de l'information* […]". Des définitions que donne Jean-Paul Saint Aubin ressortent clairement trois idées maîtresses. La première positionne en deux étapes successives et disjointes l'établissement de mesures (dans les trois dimensions) et l'établissement de documents graphiques (en plan). L'auteur prend comme axiome qu'un relevé, pourtant dans les trois dimensions, a pour représentation naturelle la projection dans un plan de l'image de l'édifice. Ne plus perdre cette troisième dimension dans la représentation de la mesure est, on le verra, un point clé de notre démarche. La deuxième idée défendue par l'auteur est qu'un relevé s'appuie nécessairement sur une analyse préalable de l'objet étudié, autrement dit que la mesure ne peut se passer d'une étude théorique de l'édifice. Nous avons repris ce point à notre compte. Enfin, Jean-Paul Saint Aubin considère la pratique du relevé comme à la fois un travail de mesure des dimensions de l'objet mais aussi comme une pratique documentaire pour laquelle le relevé photogrammétrique apporte naturellement un plus.

Nous reportons en **annexe 16** notre discussion sur les différents champs applicatifs de la technique du relevé photogrammétrique dans le domaine de l'architecture

### 3.4.2. La représentation

Nous abordons ici le thème de la représentation en architecture dans un double souci: celui d'étudier comment elle sert aujourd'hui le travail de documentation de l'édifice ou de recherche effectué par l'architecte-conservateur, et celui de définir en quoi elle peut être utilisée comme outil d'appréhension et d'exploitation d'un modèle. Nous évoquerons également les questions que pose l'utilisation de l'imagerie de synthèse en l'absence de ces codifications de fait qu'ont fait naître dans le dessin géométral plusieurs siècles de pratique graphique des architectes. Nous verrons plus loin d'abord quel type d'image nous souhaitons produire, et comment elles sont liées au modèle sous-jacent. Nous verrons également comment nous tentons de tirer parti de la représentation tridimensionnelle pour servir d'interface de navigation dans un ensemble de données relatives aux édifices. Afin d'alléger cette partie du mémoire, nous renvoyons en **annexe 18** d'une part l'analyse proposée par R. Stenvert [Stenvert, 1991] sur la signification de l'image dans l'étude du patrimoine et d'autre part un rapide historique de la représentation en architecture.

#### 3.4.2.i) L'image dans le contexte du patrimoine bâti : expériences

La thèse de P.Alkhoven [Alkhoven, 1993], bien que datant déjà un peu en terme d'outils informatiques, donne une bonne idée du rôle que l'on souhaite voir rempli par la représentation dans un travail sur l'évolution de l'architecture. Elle fixe ainsi l'objectif de son étude: étudier les dynamiques d'évolution du paysage urbain de Heusden (Pays-bas) en vérifiant en quoi les techniques de représentation par ordinateur peuvent servir à mieux comprendre les processus de mutation de la ville. L'auteur mène dans un premier temps une analyse formelle du tissu urbain de Heusden dont elle dérive une classification typologique à rapprocher du travail de [Baculo et al, 1995], déjà cité. Elle s'appuie pour ce faire sur les trois représentations conventionnelles de l'édifice (plan de toiture, masses, façades) sur lesquelles se fondent aux Pays-Bas les décisions relatives aux espaces protégés. Elle s'intéresse en priorité aux évolutions formelles (morphologiques) de la ville, en indiquant qu'elle considère

cet aspect comme un préalable indispensable à toute interprétation plus globale des processus de transformation de la ville. Elle ajoute : "*a city can be regarded as a historical document from which the subsequent phases of its development can be read"* [Alkhoven, 1993, p16]. La méthode choisie par l'auteur pour analyser les évolutions de Heusden et tenter d'en simuler des états antérieurs identifie deux sources de données différentes : d'une part les plans de la ville à différentes périodes qui fournissent une implantation aux édifices, et d'autre part une banque de modèles architecturaux (masses et façades des édifices) qui leur fournissent une volumétrie hypothétique.

*Figure 29 : Représentation du tisu urbain à partir d'une banque de modèles architecturaux, Heusden, Pays-Bas, in [Alkhoven, 1993]*

On retrouve cette approche dans la méthode utilisée pour produire les maquettes numériques. L'auteur s'est placée au niveau utilisateur d'un logiciel de CAO. Compte tenu de leur poids, six maquettes numériques différentes ont été produites, chacune représentant une période historique particulière. Les scènes ont donc été construites sur la base de plans remis à l'échelle les uns par rapport aux autres. La définition des volumes (hypothétiques pour certains) des édifices s'appuie sur une documentation historique pour une bonne part photographique puisque la période traitée est la première moitié du vingtième siècle. Pour chaque édifice est établie une fiche définissant un ensemble de données sur ses dimensions et son histoire particulière.

Ces données ne sont pas accessibles au travers de la plateforme utilisée. Les objectifs de l'étude sont explicites: analyser les transformations spatiales par plans volumes et apparence. L'auteur donne pour rôle à la représentation celui de traduire par une expression graphique pertinente (motifs typologiques 2D) le niveau d'abstraction de l'étude. Le choix fait en définitive en terme d'expression graphique a été de représenter des volumes ombrés (Phong) en combinaison avec un plaquage de textures pour les seules façades répertoriées. La superposition des maquettes tridimensionnelles s'est avérée dans cette expérience moins d'une moins bonne lisibilité que la superposition de plans. Les textures plaquées ne sont pas des photographies mais des élévations (au sens de l'architecte) dessinées en fonction d'un type[74]. Ce faisant, l'auteur introduit une idée qu'il nous semble essentiel de mettre en lumière: la maquette numérique, utilisée pour générer des représentations géométrales, perspectives, immersives ou animées, a vocation à être interprétative dans le cadre d'études patrimoniales.

A l'opposé, de nombreux travaux tels que Virtual Stonehenge, présenté en **annexe 19**, s'attachent à produire une représentation proche du réel dont l'incontestable pouvoir de séduction ne doit pas faire oublier que leur application est le plus souvent cantonnée à une pédagogie grand public du projet architectural qui s'inscrit là dans la lignée des vues perspectives de Claude Perrault.

### 3.4.3. Gestion de données historiques et bibliographiques

Un travail autour du patrimoine architectural ne se conçoit pas sans s'appuyer sur un ensemble de sources documentaires et historiques. Documenter l'édifice, c'est aussi le replacer dans le cadre plus général de l'évolution des formes architecturales et des usages du bâti. Il convient par conséquent non seulement de s'appuyer sur ces références dans l'élaboration du modèle architectural mais également que celui-ci, en retour, donne accès aux sources documentaires. Pourtant, nous allons le voir, la documentation de l'édifice n'est pas traditionnellement centrée sur les concepts à partir desquels nous élaborons notre modèle

[74] Ce point mérite d'être approfondi. En effet, une façade est ici définie par un ensemble mur + ouvertures, chacun des deux termes variant indépendamment de l'autre. A partir de cette typologie de murs et de baies, chaque façade est restituée manuellement en deux dimensions (représentation géométrale sans effets de texture). Une coloration globale est ajoutée à la définition de chaque façade. Chaque façade est alors utilisée comme texture à plaquer dans la maquette tridimensionnelle. Le type de rendu qu'autorise cette méthode, évidemment coûteuse en temps, correspond pour l'auteur au champ de préoccupation de la conservation où la maquette est un instrument d'analyse. Elle l'écrit en ces termes: "*Because this project deals with visualisation of architecture for a special purpose rather than the creation of measured drawings or photorealistic renderings, this level of abstraction is certainly acceptable*"[Alkhoven, 1993, p72].

architectural (des objets physiques élémentaires). Nous devons prendre acte de cet état de fait afin d'établir en quoi les nécessités de la documentation de l'édifice peuvent peser sur notre effort de modélisation.
Nous ferons donc ici un tour d'horizon du type de données dont se sert le conservateur, précisant également quelles sont les méthodes utilisées aujourd'hui encore pour documenter l'édifice et gérer l'archivage de l'information qui lui est attachée :

- Forme des données: textes, plans, dessins, etc…
- Origine des données: sources historiques, relevés, etc…
- Localisation et disponibilité des données: archivage en bibliothèques, etc..
- Portée des données: informations relatives à l'édifice, au genre d'édifice, à l'usage de tels édifices, etc…

En résumé, nous devons établir ici la grande diversité des sources documentaires à prendre en compte dans une étude de l'édifice patrimonial. Cette diversité recouvre une question plus fondamentale dans le cadre de notre travail: à quels concepts les sources documentaires sont-elles attachées? En effet, notre démarche de modélisation s'appuie sur une classification typo-morphologique des éléments de corpus de l'édifice. Il nous faudra vérifier, en conséquence des points évoqués ici, quelles solutions le formalisme de représentation des concepts utilisé (l'approche objet) nous permet de mettre en œuvre pour lier notre hiérarchie de concepts et un ensemble de données non morphologiques elles mêmes sujets potentiels à modélisation.
Nous proposons en **annexe 20** une section intitulée "Méthodes de Documentation et d'archivage" qui complète les éléments rapportés ici. Nous ne nous intéresserons ici qu'au thème de la construction de systèmes d'information dans le contexte de l'étude des édifices bâtis.

### 3.4.3.i) La recherche de systèmes d'information

L'archéologie est le champ disciplinaire privilégié où s'est fait jour depuis plusieurs années la nécessité d'organiser, autour de fouilles d'un édifice par exemple, un système d'informations utilisé à l'échelle du bâti. Se plaçant en utilisateurs de plates-formes logicielles existantes, de nombreux chercheurs ont tenté d'adapter les Systèmes d'Information Géographiques commerciaux tels qu'Arc Info[75] aux besoins spécifiques de l'archéologie.
De tels systèmes offrent bien entendu l'intérêt de disposer d'un outil de référencement spatial d'informations sur un site, mais leur succès tient également à la qualité de leurs interfaces pour des non informaticiens, et à leur capacité à interpréter par exemple sur une échelle de temps les données stockées. On peut se reporter pour une discussion sur ce sujet à [Ioannidis et al, 1999], déjà cités.
De nombreuses contributions ont pourtant permis de mieux cerner le problème central qu'il convient de citer pour mieux comprendre en quoi les Systèmes d'Information Géographiques tardent à s'imposer dans le domaine de l'architecture patrimoniale: la forme architecturale se lit en trois dimensions. Partant de ce principe, plusieurs projets récents tentent de rapprocher principe général des SIG et représentation tridimensionnelle de l'édifice. Ils posent dès lors une nouvelle question: à quel concept spatial attacher les données que la maquette interface? Nous présentons en **annexe 21** en fin de document deux projets de recherche qui illustrent cette difficulté.

### 3.4.3.ii) Gestion d'informations et points de vue

Dans le travail de Philippe Boudon [Boudon, 1971] se fait jour avec force l'idée que l'architecture est le lieu d'une rencontre de contraintes essentiellement différentes. Il détaille notamment un ensemble d'échelles correspondant à autant de points de vue s'entrecroisant dans l'analyse du bâti. Cette proposition vaut bien entendu également dans le domaine du patrimoine architectural où par ailleurs la notion du temps intervient :

[75] On peut se reporter par exemple à l'article en ligne de Gary L. Christopherson, D. Phillip Guertin, et Karen A. Borstad intitulé "*Gis and archaeology: using arcinfo to increase our understanding of ancient jordan*" sur le site internet de la société ESRI, éditeur du logiciel ArcInfo: http://www.esri.com/library/userconf/proc96/TO150/PAP119/P119.HTM

- Sur la forme architecturale, en troublant la cohérence du projet si tant est qu'elle existe (Un édifice comme l'église St Marie de Cracovie fût construit sur une période de 97 ans, passant d'un plan d'église-halle à un plan basilical, entamé gothique tardif et terminé aux débuts de l'époque baroque).
- Sur notre connaissance de l'édifice, en masquant quelquefois les facteurs notamment socio-économiques intervenus à chaque phase dans son évolution.

La proposition de P.Boudon, présentée avec plus de détail en **annexe 22**, vise à organiser autour de l'édifice réel un ensemble de points de vue plus ou moins indépendants entre eux dont l'addition seule permet de décrire justement l'édifice. Prenons trois exemples mentionnés en annexe 2 pour illustrer l'importance de cette démarche dans le cadre de notre étude :

- La longueur inhabituelle pour l'époque et le style du chœur de l'église St André est due au statut de Capitale qu'avait alors la ville
- Les attiques de Cracovie symbolisant dans cette ville l'époque renaissance mais sont le résultat d'une législation adoptée à la suite de l'incendie qui détruisit une partie de la ville au XVIème siècle. L'évacuation des eaux pluviales ne pouvait plus se faire comme jusqu'alors sur le mur mitoyen.
- Enfin, l'église pré-romane située dans l'enceinte du château royal est une petite rotonde qui était flanquée d'une tour semi-circulaire dont ni la forme exacte ni la fonction ne sont connues à ce jour.

Ces trois exemples montrent que si une morphologie peut souvent être décrite, inventoriée ou déduite de façon non-équivoque et exhaustive, il n'en est pas de même des facteurs notamment sociaux-culturels qui interviennent dans la genèse d'un édifice architectural. L'étude de l'édifice consiste alors, en s'appuyant sur faisceau de présomptions, à formaliser une hypothèse quant à l'évolution de l'édifice dont la forme architecturale, la morphologie, est le principal «amer». Cette morphologie peut être un outil de rationalisation qui va permettre de fixer un ensemble d'échelles et d'en vérifier la cohérence.
Nous avons donc choisi de considérer la forme architecturale comme un médiateur possible entre les points de vue que proposent les études que nous avons mentionnées. C’est notamment la démarche adoptée dans l’élaboration de l’outil SOL décrit en fin de document.

### 3.4.4. Reconstruction

En citant le relevé architectural, la représentation et les données historiques et bibliographiques, nous avons cité trois instruments essentiels dans le travail de formalisation d'hypothèses de restitution qu'entreprennent architectes ou archéologues, soit dans le cadre d'études sur des édifices disparus, soit dans le cadre d'études sur des édifices dont l'état actuel, lui-même à conserver, masque un ou des états antérieurs intéressants. La reconstruction graphique d'hypothèses de restitution apparaît à y bien regarder avec les premières traductions des dix livres d'architecture de Vitruve. En effet, seul le texte de ces livres nous est parvenu, les traducteurs de l'architecte romain ont donc de fait exécuté un travail de reconstruction graphique à partir de la description textuelle que celui-ci donne de tel ou tel édifice ou ordonnancement. On l'a vu notamment avec la traduction française des dix livres que Claude Perrault illustre au XVIIème siècle. Parallèlement, Andrea Palladio propose une pratique du relevé architectural dans laquelle non seulement les éléments de réemploi sont explicitement désignés, mais également dans laquelle il émet des hypothèses sur la datation du corpus observé. La reconstruction graphique est alors utilisée à l'appui d'un texte fixant sur la forme d'un édifice une hypothèse qu'elle veut illustrer.
Le développement de la reconstruction graphique coïncide à l'évidence avec celui de la représentation de façon générale. Dans son neuvième entretien sur l'architecture [Viollet Le Duc, 1863/1977, pp 385-448], Viollet Le Duc apporte au travail de reconstruction graphique une nouvelle dimension, en émettant l'hypothèse de tracés régulateurs fixant les proportions de tel ou tel édifice. Il utilise la reconstruction graphique pour vérifier ou démontrer la

présence, dans les proportions d'édifices variés[76], de schémas de composition basés sur des figures géométriques simples (triangles, pyramides à base carrées, cercles, etc..). Il introduit donc l'idée d'une *reconstruction interprétative* dont la *représentation* graphique permet *l'explicitation*.
Les termes de notre discussion sont dès lors fixés : la reconstruction est bien une hypothèse, fondée sur une analyse de l'édifice et de son histoire, sa représentation est une formalisation dont l'objectif doit être de rendre compte cette analyse. Nous avons choisi d'aborder cette discussion en comparant la reconstruction ou la restauration des édifices physique et son équivalent virtuel : la simulation d'hypothèses en imagerie numérique.
En effet, la question que nous devons nous poser peut s'écrire ainsi : comment signifier ce qui est reconstruit dans la modélisation et la représentation d'une hypothèse de reconstruction, comme c'est le souhait dans la reconstruction ou la restauration des édifices. Ce souhait, aujourd'hui largement partagé dans le monde de la conservation et de la restauration des édifices patrimoniaux, est d'ailleurs érigé en une doctrine consignée par la Charte de Venise (Charte Internationale sur la Conservation et la Restauration de Monuments et Sites)[77].

### 3.4.4.i) Documentation des hypothèses

René Dinkel, déjà cité, décrit le choix du restaurateur d'un édifice comme devant reposer sur *une "véritable connaissance du monument*" et non sur un "*catalogue abstrait de coutumes généralement admises*"[Dinkel, 1997, p665]. Il décrit l'édifice comme première source documentaire sur lequel le restaurateur devant intervenir prendra soin de délimiter explicitement son apport (par exemple par une frontière incluse dans la maçonnerie tels que des rangs de tuiles). S'opposant par principe à l'idée d'une reconstruction, même partielle, d'éléments de structure comme un mur, il encourage par contre le complètement de décors peints fragmentaires. La doctrine du conservateur, on le voit, a ses méandres. René Dinkel justifie par exemple l'anastylose partielle du temple de Glanum par le souci de permettre aux visiteurs de se rendre compte de l'échelle des constructions initiales [Dinkel, 1997, p169]. L'auteur donne une liste des sources nécessaires à l'élaboration d'hypothèses de reconstruction qui nous concerne plus directement :

- Images anciennes, photographies et gravures.
- Relevés et dessins.
- Archives d'interventions.
- Documents historiques relatifs aux techniques utilisées par exemple.

Il ajoute à propos de l'action du restaurateur une consigne que nous pouvons appliquer également à notre champ de préoccupation : "[…] *proposer des solutions pour conserver au monument sa lisibilité* […]". Nous allons évoquer ci-dessous deux cas concrets, choisis sur notre terrain d'expérimentation, qui illustrent à la fois la logique d'intervention qui guide le restaurateur et le travail de documentation et d'analyse qui précède et accompagne la restauration.

*Figure 30 : Le beffroi de l'ancien hôtel de ville avant restauration (photographié à la fin du XIXème siècle)*

[76] Temple de Corinthe, Temple de la concorde à Agrigente, Temple de Khons à Karnak, Parthénon de l'Acropole d'Athènes, Arc de Titus à Rome, Arc de Saint-Chamas, façade de Notre Dame de Paris, etc…
[77] Voir le site de l'ICOMOS (International Council On MOnuments and Sites) http://www.icomos.org ou [Dinkel, 1997, p 529-539, 601-604]

Le beffroi de l'ancien hôtel de ville de Cracovie a fait l'objet d'une campagne d'étude et de conservation menée entre 1962 et 1966 par le professeur Wiktor Zin [Zin, 1966], ancien conservateur général des monuments historiques de Pologne. Le premier objectif de ces travaux était de remédier à un certain nombres de désordres observés sur la structure de l'édifice (fissurations et tassements en particulier)[78].
Effet du temps, les dégradations de l'édifice peuvent également être liées à des évènements ponctuels. Ainsi les bâtiments de Cracovie inscrits au patrimoine mondial ont-ils survécu au cours de leur histoire à 25 incendies, 5 tremblements de terre, 13 inondations et 17 sièges de la ville !
La campagne de fouilles précédant ou accompagnant ces travaux a permis de retrouver notamment des encadrements de fenêtre gothiques correspondant en taille aux ouvertures du troisième niveau, ainsi qu'un ensemble de marques de tâcherons rassemblées en une seule pièce. La reconstruction au second niveau d'oriels de facture gothique est le signe le plus marquant de cette intervention. Elle s'appuie sur plusieurs données :

- Des consoles retrouvées au cours des fouilles, et dont les dimensions correspondaient aux marques de leur présence laissées sur le mur après la destruction des oriels originels à l'époque renaissance.
- Des hypothèses existantes sur les états antérieurs de l'édifice.
- Des marques diverses sur l'édifice lui-même.
- Des analogies (techniques et formelles) avec des bâtiments comparables.

Des éléments de style baroque ont été déplacés aux niveaux 3 et 4 afin de montrer les murs gothiques originels et leurs percements bouchés en 1680. Ce travail s'est accompagné de l'établissement :

- En amont, d'un inventaire (relevé, descriptif, analyse)
- En aval, d'une documentation détaillée de l'intervention permettant de retracer précisément sa mise en œuvre.

Dans ce projet, antérieur à la rédaction de la charte de Venise, l'accent a été mis sur la reconstitution d'un état cohérent de l'édifice, dit période gothique, à partir d'une part de vestiges retrouvés au cours des investigations et d'autre part d'hypothèses que nourrissent à la fois des connaissances génériques et des analogies techniques ou formelles.
Le travail, plus récent, de restauration des plafonds mené par le Dr.arch M.Łukacz[79] sur la maison dite Szara Kamienica située sur la place centrale de Cracovie témoigne d'un souci plus manifeste de lisibilité de l'édifice[80].
Retenons deux points :

- La documentation (inventaire ou intervention) est la base du travail du conservateur.
- La lisibilité historique de l'édifice est sa priorité en tant que concepteur du projet de restauration.

[78] Des tirants métalliques terminés par des ancres, noyés dans la maçonnerie, ont été posés sur toute la hauteur de l'édifice et deux planchers remplacés. Des travaux d'adaptation du sous-sol ont également été engagés pour réutiliser celui-ci comme salle de spectacle. Les reprises de maçonnerie en briques ont été signalées par l'utilisation de joints au mortier de forme concave (par opposition aux joints gothiques originaux en pointe).
[79] travaux menés entre 1997 et 1999, auteur déjà cité [Łukacz,1998].
[80] En effet, sur l'existant ont pu être retrouvés des plafonds dont l'état variait considérablement selon le cas. Un premier plafond polychrome quasi-complet a pu être restauré. Sur ce plafond, seules quelques solives neuves ont du être ajoutées. Il comprend une grosse poutre médiane servant de point d'appui intermédiaire à des solives placées perpendiculairement mais dont la longueur est celle de la pièce. Les solives neuves, construites dans le même bois et avec le même profil, sont deux fois moins longues, utilisant la poutre médiane comme point d'appui latéral. Ainsi, la différence de longueur entre les solives originelles et leurs copies est le moyen par lequel M.Łukacz signifie dans ce premier cas son intervention.
D'un second plafond ne subsistaient que la poutre médiane et quelques solives démontées. Ces solives originelles montraient que le sens de pose du plancher devait être le même que celui des solives (encoches visibles sur les solives). Par ailleurs, des marques visibles sur les murs indiquaient notamment la distance entre axes de pose des solives. Dans ce second cas, le plancher reconstitué a été posé à l'inverse du sens original de pose lu sur les solives afin de signifier l'absence complète d'informations sur le plancher original.

### 3.4.4.ii) La simulation d'hypothèses de restitution

Le travail de formalisation d'hypothèses de restitution a pour objectif de proposer une reconstitution plausible d'un édifice partiellement ou entièrement détruit ou transformé. Cette représentation graphique précède naturellement une éventuelle intervention sur l'édifice comme dans tout projet d'architecture. Elle trouve dans le cadre de simulations d'hypothèses à fins d'étude un rôle particulier : celui de représenter une alternative. L'élaboration d'une hypothèse de restitution s'accompagne de la production de maquettes tridimensionnelles plus ou moins détaillées qui s'utilisent dans un processus par essais-erreurs pour affiner l'hypothèse. Le résultat final simule un état antérieur possible de l'édifice à un temps donné[81].
Pourtant le rôle potentiel de la maquette tridimensionnelle ne se limite pas à cette utilisation. La Charte de Venise stipule que "*le monument est inséparable de l'histoire dont il est témoin et du milieu où il se situe*" et que "*les éléments de sculpture, de peinture ou de décoration qui font partie intégrante du monument ne peuvent en être séparés que lorsque cette mesure est la seule susceptible d'assurer leur conservation*".
La maquette tridimensionnelle peut dans ce cadre également être utilisée afin de présenter des états masqués d'un édifice dont plusieurs strates successives seraient à conserver. En effet, le conservateur est très souvent dans l'intervention sur un tissu ancien aux prises avec des édifices dont plusieurs états successifs s'avèrent intéressants, mais dont en respect de la charte de Venise seul le plus récent peut effectivement être montré.
La demande, formulée aujourd'hui par les conservateurs ou historiens de l'architecture , et à laquelle répond la simulation d'hypothèses de restitution, est donc triple :

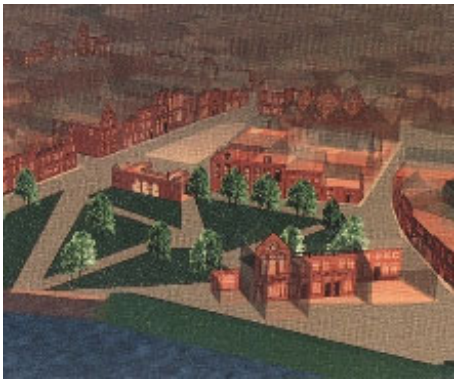


*Figure 31: Simulation d'hypothèses de restitution à l'échelle de l'architecture urbaine : le tissu urbain de Heusden, Pays-Bas, in [Alkhoven, 1993]*

- Evaluer des hypothèses concernant l'utilisation du site et de l'édifice, la structure de celui-ci, son contenu, l'évolution de ses rapports avec son environnement immédiat comme avec les courants culturels dont il témoigne.
- Présenter les états successifs de l'édifice que masque au visiteur une strate plus récente de son évolution.
- Appliquer à la reconstruction graphique la déontologie de la reconstruction physique.

Par ailleurs, nous avons au chapitre précédent évoqué plusieurs enjeux liés à l'utilisation de la représentation graphique dans le champ de la conservation en architecture : l'image comme rendant compte de connaissances sous-jacentes, l'image comme code interprétatif de ces connaissances. La simulation d'hypothèses de restitution ne se limite donc pas à représenter un édifice mais doit l'ancrer par des références et faire par exemple une différence entre le connu et l'hypothétique[82].
Nous pensons qu'en situant la production d'une maquette simulant des hypothèses de restitution en étape ultime d'un processus de modélisation des connaissances sur l'édifice, nous proposons une possible réponse à ce problème.

[81] De multiples exemples de tels travaux peuvent être cités, comme [Alkhoven, 1993] et [Ogleby, 1997] qui s'appuient sur des maquettes numériques[81] ou [Démians d'Archimbaud, 1987] qui ne le fait pas.
[82] Nous citons en **annexe 23** une expérience de simulation d'hypothèses en imagerie de synthèse menée à l'instigation du conservateur général du site de Wawel (ancienne résidence des rois de Pologne de 990 à 1550) à Cracovie, expérience citée dans [Kadłuczka, 2000]. Nous aurions pu en citer bien d'autres qui posent toutes peu ou prou le problème de la réutilisation de l'image produite et de son rapport à un travail de modélisation en amont. En effet, cette expérience a été saluée pour la qualité des rendus. Elle n'a pourtant pas fait l'unanimité chez nos interlocuteurs quand :
-d'une part, aux moyens de générer de nouvelles maquettes à partir de celles produites
-d'autre part, au message ambiguë que ces maquettes portent en terme de connaissance sur les édifices montrés (rendus réalistes soignées pour des édifices dont on ne connaît pratiquement rien).

# 3.5. CONTEXTE TECHNIQUE DE NOS DEVELOPPEMENTS

## 3.5.1. Systèmes de Gestion de Bases de Données et interfaces Web

La gestion d'informations relatives à l'édifice patrimonial, notamment dans sa dimension historique, est bien sûr un des problèmes qu'aborde ce projet. Il reste cependant en arrière plan par rapport à la question centrale de notre travail. En précisant ceci nous souhaitons expliquer la place relativement limitée consacrée ici à l'état de l'art des Systèmes de Gestion de Bases de Données. En effet, il nous semble à la lecture des références que nous citerons ci-après que quelle que soit la technologie choisie (Relationnel ou Objet), l'évolution du modèle sous-jacent reste une difficulté importante dans le développement d'un schéma de données.
Par conséquent, nous avons souhaité mettre l'accent sur la formalisation du modèle architectural plus que sur la question de la gestion de données, question qui dépasse par ailleurs largement le contexte de ce projet. Nous avons choisi ici de situer et de comparer les principales caractéristiques des Systèmes de Gestion de Bases de Données relationnels et des Systèmes de Gestion de Bases de Données (Orientés ) Objets, puis de présenter les techniques d'interfaçage SGBD / Web aujourd'hui opérationnelles, avant de citer quelques exemples de réalisation dans le domaine du patrimoine architectural. Nous verrons en conclusion à cette section quelles perspectives ces différentes solutions ouvrent à notre projet.
Les fondements des deux formalismes présentés étant largement établis aujourd'hui, nous renvoyons en **annexe 25** la présentation des principes de représentation des données dans les SGBDR et dans les Systèmes de Gestion de Bases de Données (Orientés ) Objets et ne traiterons ici que des questions liées à la gestion des requêtes et à l'interfaçage pour le Web.

### 3.5.1.i) SGBD relationnels

#### *3.5.1.i.1. Langages de requêtes et de développement*

Avec Les SGBD relationnels, les langages de définition de données (LDD) et de manipulation des données (LMD) ont fait l'objet d'une standardisation (SQL, norme ANSI de 1986). Ils permettent la formulation de contraintes d'intégrité au niveau des schémas relationnels impliqués dans un schéma conceptuel et l'expression de requêtes plus ou moins complexes, tout en respectant l'indépendance des données [Ducournau et al, 1998, p133]. Les SGBD relationnels fournissent donc leur propre jeu de types et langage de requête, SQL (Structured Query Language) , pour accéder aux données contenues dans les tables, et le langage de programmation est utilisé pour accéder à la mémoire.
SQL a sa propre syntaxe pour formuler une requête mais aussi des variables et opérations spécifiques. Les requêtes SQL sont écrites en utilisant les variables SQL et non celles du langage de programmation ou de développement.
Les SGBD relationnels fonctionnent sur le principe de deux processus indépendants, application et Base de données, dont les interactions requièrent des appels nommés IPC [Loomis, 1995, p114]. L'auteur décrit la succession d'évènements en jeu lors d'un accès par le programme à une table de données[83].
Tous les SGBD relationnels supportent SQL mais le plus souvent ils étendent le langage. SQL peut être vu comme un habillage syntaxique de l'algèbre relationnelle avec quelques

[83] -Le SGBD relationnel utilise un indice pour déterminer quelle page contient l'enregistrement recherché.
-Il lit cette page sur le disque et la copie dans son buffer.
-L'application ne pouvant accéder à ce buffer, Le SGBD relationnel copie les champs indiqués par la requête SQL dans des variables du programme, dans l'espace de celui-ci.
-L'application peut alors manipuler ces données dont le type est celui des variables du programme.
-Si l'application prévoit une mise à jour de la base, alors une requête SQL UPDATE est adressée au SGBD qui copie les valeurs des variables du programme dans les champs d'un enregistrement localisé dans son buffer.
-Le SGBD copie alors l'enregistrement modifié soit à sa position d'origine soit ailleurs en fonction des circonstances de l'événement. Si une nouvelle position est attribuée à l'enregistrement ou si les valeurs d'indices ont été modifiées, la base modifie également les indices affectés.

éléments empruntés aux langages prédicatifs (les quantificateurs) et quelques extensions (agrégats et GROUP BY). Les opérateurs SQL forment un système fermé agissant et produisant des tables [Loomis, 1995, p38], autorisant le programmeur à construire des requêtes imbriquées complexes.

*3.5.1.i.2. Limitations du modèle*

Les SGBD relationnels sont aujourd'hui largement utilisés, en particulier couplés avec les Systèmes d'Information Géographiques tels qu'ARC INFO. Pourtant, [Ducournau et al,1998, p134] en rappellent ainsi les limites:

- Le modèle de données ne permet de manipuler qu'un seul type de données, la table, qui correspond à un ensemble de n-uplets. Ceux-ci sont eux-mêmes une concaténation de types primitifs (entiers, réels, chaînes de caractères, etc…). Ce modèle de données impose au concepteur d'applications à objets une étape de transformation de données rendues persistantes *à plat* [Loomis, 1995, p21]. De plus ces étapes de transformation sont laissées à la charge du programmeur, alourdissant d'autant le développement d'applications. Les objets doivent non seulement être *aplatis* mais aussi scindés en autant de tables qu'il en faut pour assurer leur persistance, et assurer les mécanismes de requêtes formalisés en SQL[84].
- L'incompatibilité entre les langages de définition et de manipulation de données (LDD et LMD) et les langages de programmation pose des problèmes de développement non triviaux. Les types que manipule SQL et ceux que manipulent les Langages de programmation sont incompatibles.

La représentation des objets dans une base de données relationnelle est discutée dans [Loomis, 1995] qui consacre le chapitre 5 de son ouvrage à l'utilisation d'une Base de Données relationnelle avec un Langage de Programmation Orienté Objets. On peut également se reporter à [Rahayu et al, 1999] qui décrivent une structure d'index dédiée à la représentation des objets composés.
De nombreux travaux de recherche s'intéressent par ailleurs à l'enrichissement du modèle Relationnel. Nous renvoyons ici à [Ducournau et al, 1998], qui en situent les perspectives. Enfin, entre les SGBD relationnels et les SGBD objet apparaissent de nouveaux produits qualifiés de relationnel-objet comme ceux proposés par Informix, IBM, Oracle ou Sybase pour lesquels nous renvoyons à ces éditeurs.

### 3.5.1.ii) Systèmes de Gestion de Bases de Données (Orientées) Objets

Les Bases de Données orientées Objets (BDO) sont nées de la convergence de deux domaines, celui des bases de données et celui des langages de programmation orientés objets. [Loomis, 1995, p31] résume succinctement la question posée par cette phrase: un Système de Gestion de Bases de Données (Orientées) Objets doit supporter un modèle de données orienté objet plutôt qu'un modèle de données orienté tables.
Dans leur article "The object oriented database system manifesto" [Atkinson et al, 1990], M.Atkinson, F.Bancilhon, D.DeWitt, K.Dittrich, D.Maier et S.Zdonik définissaient en 1989 les caractéristiques qu'un système devait proposer pour être qualifié de Base de Données Objet. Les auteurs séparent ces caractéristiques en trois groupes :

- Caractéristiques obligatoires - être des systèmes à objets permettant la gestion d'objets complexes, l'identité d'objet, l'encapsulation, les notions de classes et/ou de types, l'héritage et la surcharge et être des Systèmes de Gestion de Bases de Données permettant la persistance des données, la concurrence et l'accès aux données via un langage de requêtes.
- Caractéristiques optionnelles visant à améliorer le système - notamment héritage multiple, extensibilité du modèle, versions, adaptabilité.

[84] [Loomis, 1995, p20] rappelle que la persistance des objets peut avec les mêmes contraintes de transformation être assurée par le système de fichiers. L'auteur cite en priorité deux des inconvénients de telles solutions: complexité des transformations à implémenter, invalidation des formats de fichiers ou de table lorsque les spécifications des classes évoluent

– Caractéristiques ouvertes permettant au concepteur d'effectuer un certain nombre de choix - paradigme de programmation, système de représentation, système de types et uniformité.

*3.5.1.ii.1. Langages de requêtes et de développement*

Les applications développées pour accéder à la base de données sont programmées dans un langage similaire aux Langages de Programmation Orientés Objets, les accès et mises à jour de la base sont réalisés par envoi de messages. Il faut adjoindre des opérations de manipulation de collections pour que ce langage de programmation devienne un LMD, langage de manipulation des données. Des efforts de standardisation ont été entrepris dans le cadre de l'OMG et de l'ODMG[85] pour unifier à la fois le modèle objet utilisé dans divers Systèmes de Gestion de Bases de Données Objets , mais aussi les LMD (standard OQL) Et LDD (standard ODL). Nous renvoyons à [Loomis, 1995, pp87-110] pour une présentation plus détaillée de ces initiatives.
Les SGBD Objets offrent également la possibilité d'interroger la base à travers un langage de requêtes. Ce langage spécifique est utilisable dans un environnement particulier ou immergé dans le langage de programmation. Il autorise la formulation de requêtes de base qui, pour ne pas venir en contradiction du principe d'encapsulation, impose au programmeur de définir des méthodes d'entrée / sortie pour chaque champ de l'objet. Il autorise également la formulation de requêtes dites associatives qui réalisent des sélections d'objets appartenant à des collections à partir de conditions relatives aux valeurs de leurs attributs. Il autorise enfin la formulation de requêtes dites constructives dont le résultat est une nouvelle structure construite. Les requêtes peuvent être insérées dans le langage de programmation, leur résultat est alors affecté à une variable de même type. Un nouveau standard du langage de requêtes SQL, SQL3, intègre aujourd'hui des aspects objets (identité d'objet, héritage, etc…) et divers aspects de programmation (structures itératives et opérations récursives en particulier)[86].

### 3.5.1.iii) Interfaçage Web

Nous nous appuyons sur une passerelle d'interfaçage Web standard SGBD relationnel - serveur WWW appelée Internet Database Connector (Voir [Israel, 1997]). Il nous importe ici de situer ce choix dans le contexte plus large des solutions de passerelles entre un système de gestion de bases de données relationnel et un serveur WWW. L'objectif est, à partir d'un navigateur WWW standard, d'exécuter les programmes permettant la consultation ou la mise à jour d'informations contenues dans la Base de Données.
Du côté client, WWW s'appuie sur le langage HTML (Hypertext Markup Language) décrivant à la fois la structure et la présentation des informations, interprétées par le navigateur. Du côté serveur, WWW met en œuvre le protocole HTTP (Hypertext Transport Protocol) qui régle le dialogue entre un navigateur et un serveur WWW. Enfin le serveur WWW et le SGBD communiquent via une passerelle qui peut être propre au SGBD ou reposer sur le standard CGI (Common Gateway Interface)[87].
Le SGBD assure le stockage des données, leur intégrité, l'exécution de traitements ou de requêtes principalement de type SQL. Les traitements sont effectués en principe sur le

[85] Object Management Group, Object Database Management Group
[86] L'objectif affiché des standards ODMG et SQL3 est d'offrir des langages intégrant les aspects langages de programmation par objets et LMD - LDD. [Ducournau et al, 1998, p156] s'interrogent néanmoins sur l'avenir de ces standards. Nous verrons plus loin qu'est également apparu plus récemment, dans le sillage du standard XML, un nouvelle initiative pour la définition d'un langage de requêtes, XQL. Je renvoie également au chapitre 9 de [Loomis, 1995], pour une présentation plus détaillée des formalismes d'interrogation des Systèmes de Gestion de Bases de Données Objets.
[87] On peut schématiser le fonctionnement de ces passerelles en distinguant quatre étapes :
-Le navigateur WWW présente une interface à l'utilisateur pour la saisie des requêtes. Il peut s'agir de pages HTML standard ou de l'envoi d'une requête par le biais d'informations saisies dans un formulaire.
-Le serveur WWW est à l'écoute des messages reçus et transmet les requêtes au SGBD via une passerelle qui a pour fonction de traduire le message en requêtes SQL ou traitements exécutables par le SGBD.
-Les réponses fournies par le SGBD sont restituées de la même façon sous forme de pages HTML au serveur WWW.
-La visualisation des résultats sous forme de pages HTML est assurée par le navigateur WWW.

serveur mais le développement de langages de script tels que Javascript permet d'introduire des pré-traitements ou des post-traitements côté client.
Pour implémenter la passerelle de communication entre le serveur WWW et le SGBD les premières solutions s'appuyaient sur le standard CGI utilisé dans le traitement des formulaires HTML. Plus récemment, certains éditeurs de SGBD ont proposé des solutions propriétaires plus performantes telles qu'Oracle WebServer 2.0, solution propriétaire Oracle ou Internet Database Connector, solution Microsoft s'appuyant sur Internet Information Server et le standard ODBC (Open Database Connectivity). ODBC est une interface de programmation permettant de transmettre des requêtes SQL à un SGBD et de récupérer les résultats. L'équivalent JAVA d'ODBC est JDBC, interface de programmation permettant d'adresser des requêtes à n'importe quel SGBD depuis le langage Java ainsi que de récupérer le résultat.
Ces architectures de passerelle restent assez contraignantes puisqu'elles imposent d'une part de gérer un ensemble de pages HTML servant à présenter sous forme graphique les requêtes à adresser au SGBD, et d'autre part de les traduire en requêtes SQL. Cela a pour conséquence d'imposer au programmeur un travail de définition des requêtes a priori assez lourd, sauf à introduire entre ces deux étapes un ensemble de traitements côté client (Javascript par exemple) ou côté serveur (Perl par exemple). Ces traitements ont notamment pour objectif de libérer le programmeur de la création systématique de pages HTML.
En technologie Objet, les solutions de passerelle entre un système de gestion de bases de donnée et un serveur WWW s'appuie sur le développement du Langage JAVA déjà évoqué précédemment, et dont un des API permet de réaliser des applications pour le Web. Les éditeurs de Systèmes de Gestion de Bases de Données (Orientés ) Objets cherchent donc à offrir un support complet de Java, par exemple Object Design et la version 5.1 de son SGBDO ObjectStore.
L'interface Java intègre désormais le support de la collecte de "déchets" persistants et permet d'exploiter les fonctions d'ObjectStore telles que la notification des événements, le contrôle de la concurrence de plusieurs versions, la restauration en cas d'incident, la réplication. Cette interface utilise la même API qu'ObjectStore PSE pour Java, version légère et monoposte du SGBDO.

#### 3.5.1.iv) Exemples et perspectives d'utilisation

Dans le cadre des Systèmes de Gestion de Bases de Données Objets comme dans celui des Systèmes de Gestion de Bases de Données relationnels, la problématique de l'évolution du système est présente dans de nombreux travaux de recherche.
On se reportera à [Ducournau et al, 1998, pp156-162] pour une présentation des perspectives qui s'ouvrent dans ce domaine[88]. S'il n'entre pas dans le cadre de ce document de détailler ce point, il nous faut néanmoins en terme de conclusion essayer d'établir en quoi les modèles objets ou relationnels peuvent s'avérer plus ou moins pertinents dans le cadre de ce projet.
Les SGBD Objets offrent aujourd'hui une solution permettant notamment:
- D'assurer la représentation de modèles de données complexes.
- De bénéficier des apports de la technologie objet en terme de modularité et de réutilisabilité des applications.

En contrepartie, ils posent un certain nombre de problèmes résumés ainsi par [Ducournau et al, 1998, p162] :
- Un manque de formalisation pour les modèles proposés.
- L'absence du concept simple de relation tel que présent dans modèle relationnel.
- Le manque de déclarativité des langages de requêtes.

Les Systèmes de Gestion de Bases de Données relationnels, de leur côté, bénéficient notamment

[88] Les auteurs distinguent en particulier quatre axes de recherche principaux : les problèmes d'évolution de schéma, les problèmes d'évolution des données, les problèmes d'évolution globale schéma-données (propagation des mises à jour depuis le niveau schéma jusqu'aux instances) et les problèmes de multi-expertise (points de vue divergents sur un même ensemble de données).

- D'un modèle de données simple.
- D'une base formellement définie qui a permis à la fois de définir des méthodes de conception de schémas et des langages de manipulation standardisés.

En contrepartie, ils posent un certain nombre de problèmes déjà évoqués plus haut :
- Le modèle de données ne permet de manipuler qu'un seul type de données, la table.
- Les types que manipule SQL et ceux que manipulent les Langages de programmation sont incompatibles.

Le succès des couplages OO-relationnel qu'évoquent [Ducournau et al, 1998, p162] montre bien que le choix entre ces solutions n'est pas trivial[89].
Nous avons à l'origine travaillé en modèle objet et base de données relationnelle. Ce choix sera discuté plus loin. Nous souhaitons cependant nous inscrire dans une possible double perspective : celle de l'intégration d'un Systèmes de Gestion de Bases de Données (Orientés ) Objets déjà réalisée dans le module mesure photogrammétrique développé par P.Drap [Drap, 1997], mais plus encore dans celle du développement des standards XML et XQL qui ont pour avantage de rester à l'échelle du problème que notre projet traite.
En l'absence d'éléments nous indiquant un choix définitif, nous avons pour l'instant privilégié une solution avant tout pratique et disponible qui nous permettait d'avancer sur le reste du projet, et qui est à l'évidence plus adaptée aux collections de données existantes que nos partenaires nous soumettent.
Nous présentons par ailleurs en **annexe 24** un exemple de référence sur l'utilisation de Systèmes de Gestion de Bases de Données dans le cadre d'études patrimoniales.

#### 3.5.1.v) Gestion de données: Bibliographie

Nous proposons, classée par ordre alphabétique, une liste des références citées dans cette partie de l'état de l'art à la fin du présent document, section "bibliographie thématique".

### 3.5.2. Informatique graphique et formats 3D pour Internet

La représentation joue un rôle majeur dans le travail de l'architecte ou du conservateur. J.C Lebahar [Lebahar,1983] la présente comme un outil principal dans le processus de réduction d'incertitudes qui permet à l'architecte de formuler son projet. La représentation donne dans ce cadre à l'architecte un moyen de prendre des instantanés de son projet. Elle ne permet pas d'omissions en terme de morphologie mais peut être utilisée dans un travail d'essais-erreurs pour autant que la géométrie figurée s'affine en même temps que le projet. Par ailleurs, le développement des modeleurs dédiés ou des modeleurs paramétriques apporte en phase finale de projet des solutions permettant de typer morphologiquement les entités géométriques utilisées. La place de la représentation dans le domaine de la pratique architecturale est discutée dans [Tweed, 1997], nous ne pensons pas utile d'en faire la démonstration ici.
A l'occasion d'études visant à simuler une hypothèse de restitution, la représentation géométrique exhaustive de l'édifice disparu se traduit essentiellement par la mise en évidence d'incohérences ou d'impossibilités dans les choix théoriques faits par les architectes, archéologues ou historiens auteurs de l'hypothèse. Elle se traduit également par la nécessité pour l'auteur de l'hypothèse de dimensionner chaque élément de l'hypothèse et par conséquent de disposer de nouveaux éléments de comparaison avec d'autres édifices de même type, travail essentiel à l'occasion d'études architecturales à caractère historique[90]. Nous ferons ici un bref sommaire des formalismes de représentation géométrique ou de rendu et des outils

[89] Dans le cas de notre projet, deux paramètres essentiels sont également à considérer : le choix d'un développement "tout Internet" (Applications plateforme-indépendantes, interfaces gérées par les navigateurs WWW, accès et mise à jour des données par le biais de formulaires standards, représentation tridimensionnelle VRML) et la profondeur relativement restreinte du modèle.
[90] On peut pour s'en convaincre se reporter aux expériences menées par le laboratoire MAP- équipe GAMSAU, notamment sur l'amphithéâtre d'Arles ou l'ancien hôtel de ville de Cracovie ou par le laboratoire MAP- équipe CRAI, notamment sur plusieurs édifices de la cité grecque de Delphes, http: //www.map.archi.fr

applicatifs correspondant, et étudierons ensuite quelles familles de solutions sont aujourd'hui envisageables pour représenter via la plate forme Internet une scène tridimensionnelle.
Nous souhaitons en particulier mettre l'accent sur l'utilisation de l'image comme étape nécessaire dans l'élaboration d'une hypothèse de restitution ou dans la représentation d'un corpus. Nous entamerons donc ce chapitre par une synthèse des différents modèles de représentation géométrique et de simulation de l'éclairage sous-jacents aux outils de visualisation choisis pour ce projet. Nous souhaitons par ce biais montrer qu'en terme de plate-forme de visualisation de nombreuses perspectives restent ouvertes. Nous conclurons en signalant que le problème que pose un projet comme le nôtre en terme de visualisation est celui du sens qu'une représentation suggère. Autrement dit, nous verrons que la problématique ouverte par ce projet en terme de visualisation est une problématique de sémantique de l'image indépendamment de tout choix technique.
Nous devons néanmoins en premier lieu insister sur la nécessité de lier la représentation géométrique de l'édifice au modèle architectural que nous proposons. En effet, si ce principe est au cœur même des Systèmes d'Information Géographique, où la carte sert d'interface de navigation dans un ensemble d'informations, il ne l'est pas ou peu dans le monde de la représentation en architecture patrimoniale où l'image est le plus souvent au mieux une des facettes du modèle, au pire sa seule justification. Pourtant, la définition d'un corpus tridimensionnel nous donne une réelle possibilité de nous servir de la représentation de l'édifice comme d'une interface de navigation dans l'ensemble d'informations qui lui sont relatives. Nous nous situons donc dans une démarche qui est celle des SIG, une information localisée dans l'espace, avec cependant deux exigences nouvelles : la troisième dimension, celle de l'architecture en volume, et la "quatrième" dimension, celle de l'histoire de l'édifice[91].
Nous revenons plus loin sur les deux questions que nous considérons soulevées par notre travail sur le champ de la représentation:

- Quelles règles déontologiques pour la simulation d'hypothèses de restitution en images de synthèse?
- Quelles solutions expérimenter pour lier la représentation géométrique de l'édifice au modèle architectural sous-jacent dans un hypothétique Système d'informations sur l'édifice patrimonial dont l'interface serait spatio-temporel ?

### 3.5.2.i) La représentation géométrique

Les principes des modèles volumiques et leur genèse sont présentés avec détail dans les ouvrages de référence [Péroche et al ,1988] et [Péroche et al ,1997] (seconde édition du premier cité)[92]. Nous allons rapidement situer, en **annexe 26**, les principes de deux schémas de représentation: l'arbre de construction (CSG, Constructive Solid Geometry), et la représentation par frontières (BREP, boundary représentation)[93].

### 3.5.2.ii) Modèles de rendu

Dès les années soixante, avec le problème de l'élimination des faces cachées, de nombreux travaux de recherche se sont focalisés sur les méthodes et modèles dédiés à la synthèse d'images pour produire des images réalistes. [Péroche et al, 1988, pp145-159] présentent un historique des recherches menées dans le domaine de l'élimination des parties cachées et un certain nombre d'algorithmes dédiés à cette tâche.
Nous ne détaillons pas ce point désormais bien connu mais revenons rapidement en **annexe 27** et **28** sur trois autres points, les modèles de couleur et d'éclairement, le lancer de rayon et la vision stéréoscopique humaine. Nous consacrerons enfin un bref encart placé également en

[91] On pourra sur ce dernier point se reporter aux travaux du laboratoire LIMOS (Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes) sur les systèmes d'information spatio-temporels : http://libd2.univ-bpclermont.fr/
[92] Après avoir défini les notions de base des mathématiques pour la synthèse d'images, les auteurs présentent les algorithmes 2D fondamentaux et traitent notamment des problèmes d'affichage et d'antialiasage. Les auteurs introduisent ensuite la notion de modèle volumique en présentant différentes alternatives dans une perspective historique, notamment instanciation de primitives et énumération spatiale pour lesquels je renvoie à leur ouvrage.
[93] Ce choix s'explique ici par le fait que ces deux modèles sont sous jacent aux outils de visualisation que nous utilisons, lancer de rayon POV-Ray dans le premier cas, VRML2.0 dans le second.

**annexe 29** à la présentation de POV-ray et ne nous intéresserons dans cette section du document qu'aux formats 3D pour le Web.

### 3.5.2.iii) Formats 3d pour le Web

Nous ne ferons pas ici un recensement exhaustif des formats 3D utilisés sur le Web. En effet, leurs applications sont trop diverses[94] pour intéresser de façon générale notre projet. Nous allons donc tenter en introduction à cette section de définir ce que nous attendons d'un format 3D, avant de nous intéresser plus longuement au format VRML (Virtual Reality Modelling Language).

#### *3.5.2.iii.1. Type et objectifs*

Commençons ici par établir les caractéristiques essentielles que nous recherchons dans un format 3D pour le Web:

- Qu'il permette la visualisation sur le Web, sans investissement matériel ou logiciel, de scènes 3D.
- Qu'il permette la visualisation de ces scènes 3D sur n'importe quelle plate-forme.
- Qu'il permette à l'utilisateur distant d'interagir avec cette scène, soit en s'y déplaçant, soit en l'interrogeant, soit en la modifiant.
- Que cette scène soit décrite dans un langage standard manipulable par un Langage de Programmation.
- Qu'elle puisse servir d'interface avec un Système de Gestion de Bases de Données.
- Qu'elle puisse servir de lien vers d'autres scènes ou d'autres représentations du modèle.

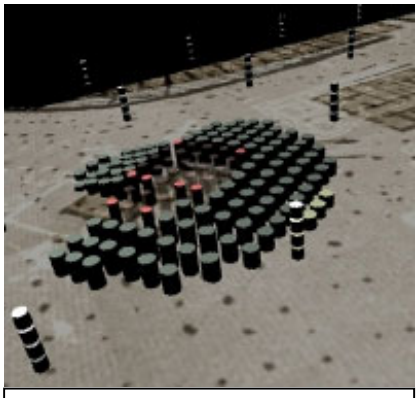
On voit dès lors que l'éventail de solutions se rétrécit considérablement. Les géographes ont été les premiers à se rechercher des systèmes disposant de caractéristiques proches de celles-ci, avec cependant comme différence notable une moins grande priorité donnée à la troisième dimension. On pourra bien sûr se reporter au site Web de la société ESRI[95], éditeur du très répandu système ARCInfo, site sur lequel on trouvera de nombreuses publications relatives à l'application des Systèmes d'Information Géographiques dans le cadre de solutions Web.
On est cependant ici dans le cadre d'un produit commercial à l'opposé des points 1, 2 et 4 cités ci-dessus. On peut également se reporter au travail mené par l'équipe du *Centre for Landscape Research* de l'Université de Toronto (Canada)[96] qui a développé un client Web spécifique appelé CLRMosaic.
Le couple SDML / navigateur CLRMosaic permet donc notamment :

- D'attacher des informations non-graphiques à des objets référencés spatialement.
- De déplacer les attributs des objets côté client-accélérant d'autant les requêtes sur le modèle.
- D'effectuer des requêtes non sur les objets de la scène mais sur leurs attributs.

Si le langage VRML dont nous parlerons ci-après permet de faire la même chose en ce qui concerne le premier point, il n'offre en revanche pas de solution aisée pour les deux suivants. CLRMosaic apporte donc des solutions intéressantes (fonctionnalités d'interrogation, format SDML ASCII libre) mais pose de façon générale un problème de standardisation tant en terme de langage de modélisation de scènes (basé sur l'utilisation de polygones ) que de fonctionnalités du navigateur. Reste alors pour répondre aux exigences que nous nous

[94] Quick TimeVR par exemple, format Apple, est une simulation de la 3D utilisant des images 2D collées les unes aux autres pour restituer un lieu photographié exhaustivement. Bien que dédié au Web et plus ou moins 3D la plate-forme Quick TimeVR n'a pas d'application concrète dans notre cas ou l'accent est mis sur la restitution d'un édifice à partir d'un modèle architectural sous-jacent.
[95] Voir http://www.esri.com
[96] CLRMosaic interprète un langage de description de scènes baptisé SDML (Spatial Data Modelling Language), langage ASCII permettant à la fois de décrire morphologiquement une scène et d'attacher aux objets ainsi décrits notamment des liens de type URL soit vers d'autres pages soit vers des requêtes adressées à des Systèmes de Gestion de Bases de Données. Voir http://www.clr.toronto.edu/CLRMOSAIC/

sommes fixées essentiellement deux solutions : le format VRML et les modules 3D du Langage de Programmation JAVA.
JAVA propose un API (Application Programming Interface) appelé JAVA 3D dont l' objectif est en particulier d'autoriser l'accès collaboratif à des modèles 3D sur le Web[97]. Comme tous les développements JAVA, Les applets ou applications 3D JAVA sont censés indépendants des plates-formes matérielles. La définition de géométrie peut de faire par la définition de chaque arête de chaque triangle de chaque objet (!..) ou par le biais de modèle importés (Lightwave3D, Alias .obj, VRML, 3DstudioMax, etc..). Couleur et textures des objets, sources de lumières ambiantes, directionnelles et ponctuelles sont les attributs à spécifier dans une scène. Java 3D permet d'inclure des capacités d'animation proches de VRML: détection de collisions, boîtes englobantes, déclenchement d'évènements, jeu d'interpolateurs de mouvements. L'affichage des scènes est indépendant de la définition de celle-ci, autorisant par exemple la définition de deux caméras pour l'affichage en stéréo.
JAVA 3D est une API assez récente, et c'est là la première raison pour laquelle nous n'avons pas développé d'applicatif l'utilisant. En terme de performance et de qualité de rendu, JAVA 3D ne se distingue pas de façon notable des plugins VRML. JAVA 3D est une API dont l'évolution rapide rend plus difficile la mise en place d'une plate-forme de visualisation stable, ce que nous recherchons en priorité.
Enfin, travailler en JAVA 3D pose le problème de l'exportation du modèle vers un standard et de l'utilisation de la scène en dehors du contexte d'un applet JAVA, problèmes résolus avec VRML.
Toutes ces raisons nous ont conduit à privilégier la plate-forme VRML jusqu'à présent. Notons enfin qu'en terme de visualisation native en JAVA, des solutions alternatives existent comme la librairie JView3D, beaucoup plus économique, compatible avec le JDK JAVA 1.0.2, et basée sur l'algorithme d'élimination des parties cachées dit algorithme du peintre[98].



*Visualisation VRML, exemple.*

*3.5.2.iii.2. Langage VRML*

Nous allons procéder ici en deux temps: nous présenterons un historique du langage VRML et de ses principales caractéristiques, puis détaillerons quelques points essentiels de la création de scènes sous la forme d'un encart en **annexe 30**. En effet, il importe de vérifier d'une part comment ce langage permet d'exprimer notre modèle architectural sur le Web, mais également de décrire plus précisément quelques caractéristiques (comme par exemple le mécanisme de gestion des niveaux de détails) qui vont au-delà de la simple visualisation du modèle mais ouvrent des perspectives sur sa définition même.
La version 2.0 du langage[99], sortie en 1996, propose une description des scènes 3D permettant notamment une grande interaction client Web- scène 3D, la possibilité de lier des objets

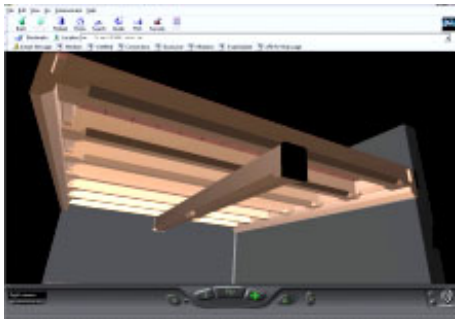


*Figure 32 : Reconstitution d'un plafond en bois en VRML , visualisation dans le plugin CosmoPlayer 2.1, navigateur Netscape 4.5 (im.auteur)*

[97] . La hiérarchie d'une scène comprend une première branche appelée *content branch* et une seconde appelée *viewing branch*. La première permet de définir les objets par leur géométrie et leur apparence ainsi que les conditions d'éclairage. La seconde permet, indépendamment de la première, de modifier les paramètres de la plate-forme de visualisation.
[98] On en trouvera une utilisation dans le cadre du projet ARPENTEUR [Drap et al, 2000], on peut également se reporter au site Web de son auteur http://visualbeans.com/View3D/
[99] La première version du langage VRML (VRML 1.0, Virtual Reality Modeling Language ou Langage de Modélisation de Réalité Virtuelle) était au départ un sous-ensemble du format ASCII des fichiers Open Inventor de Silicon Graphics. Cette extension du savoir-faire de Silicon Graphics en graphique 3D pour le web proposait une description de scènes 3D assez statique, avec une interaction très limitée. VRML 1.0 permettait de décrire et d'organiser des ensembles d'objets (nœuds) permettant de construire des scènes 3D stockées dans des fichiers ascii. Comme dans OpenInventor, les nœuds étaient organisés selon une arborescence (SceneGraph) qui permettait de structurer et d'ordonner les objets.

à un son, une animation ou un script, notamment une applet Java, mais également le déclenchement d'évènements en cascade. La notion de nœud script permet de manipuler une scène VRML depuis une application ou une applet Java[100].
VRML 2.0 a fait l'objet en 1997 d'une standardisation sous le nom de VRML 97 (International Standard ISO/IEC 14772-1:1997)[101]. [Bridges,1999] propose une discussion sur la genèse des formats HTML et VRML à laquelle on peut par exemple se reporter[102].
Les browsers permettant de visualiser le scènes VRML sont des applications externes aux clients Web (navigateurs tels que Netscape) ou des surcouches de ceux-ci, dites plug-ins, qui interprètent les fichiers au format VRML[103]. Une scène VRML est transférée depuis le serveur au client, c'est donc le machine cliente qui, une fois le transfert achevé, est chargée de prendre en charge, en local, par exemple les mouvements que l'utilisateur déclenche. Ce principe permet une fois le chargement de la scène effectué de ne plus faire d'appel réseau. En contrepartie, le comportement de la scène (vitesse de déplacement en particulier) n'est pas totalement maîtrisable par le programmeur.
Depuis l'origine, VRML est donc un langage de modélisation de scènes (devenu une norme) permettant de définir des objets.
Ces objets peuvent contenir des figures géométriques bien sûr mais également des données multimédia par l'association de fichiers externes à l'objet. Nous revenons dans un encart placé en annexe 30 sur les aspects les plus significatifs du langage VRML[104]. Le principe de base d'un document VRML est la notion de *node* (nœud)[105]. Les nœuds peuvent être groupés, avec la notion de nœud parent et enfant: ils sont organisés dans des

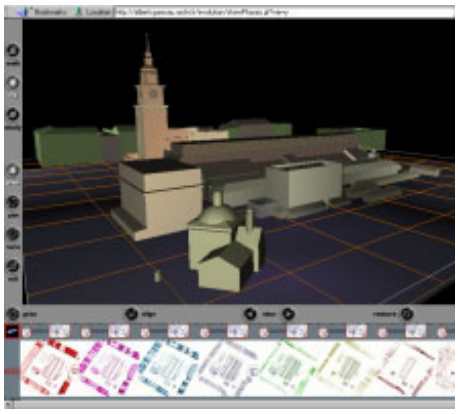


*Figure 33 : Maquette VRML utilisée dans le cadre de l'interface d'accès à la base documentaire Sol [Dudek et al, 1999b], visualisation dans le plugin Cortona VRML Client, , navigateur Netscape 4.5 (im auteur).*

[100] On peut se reporter pour une description exhaustive des méthodes de programmation des scènes VRML en JAVA à l'ouvrage de référence "VRML 2.0 with Java" [Roehl et al, 1997].
[101] Voir : http://www.web3d.org/Specifications/
[102] Les objectifs fixés pour ce standard y sont décrits de la façon suivante: "*VRML est un format de fichier destiné à décrire des mondes 3D interactifs. VRML est conçu pour être utilisé sur Internet, en intranet et en local. VRML est aussi destiné à être un format d'échange universel pour des productions graphique 3D multimédia ….. VRML est capable de représenter des objets multimédia statiques ou animés comprenant des hyperliens vers d'autres media ( textes, sons, films, images). Des browsers VRML et des outils de création de scènes sont largement disponibles …* ".
[103] Voir notamment le site de Silicon Graphics consacré à VRML: http://vrml.sgi.com/intro.html et l'adresse suivante pour, entre autres, une liste plug-ins adaptés à VRML 2.0: http://www.vrml.ch/
[104] -Texte ASCII interprété -Primitives géométriques combinables sans opérations booléennes -Apparence et éclairage (Modèle de Phong) et texture par références à des URL externes -Fonctionnalités d'animations - Affectation d'URL - Gestion des niveaux de détail (représentations alternatives) -Insertion de scripts et communication VRML-JAVA.
[105] . Un nœud est défini par un type (cube, sphère, texture, transformation, etc..) et des paramètres (rayon de la sphère, images à importer pour servir de texture, etc..), il peut être caractérisé par un nom Plus précisément, un node est caractérisé exhaustivement par les éléments suivants :
-Un type d'objet, que l'on peut classer en quatre catégories principales de nœuds (géométries, propriétés, évènements et groupes). Les nœuds de la catégorie géométries permettent de définir des objets géométriques dans la scène. Ce sont nommément les nœuds Text, Cone, Box, Cylinder, IndexedFaceSet, IndexedLineSet, PointSet, Sphere, ElevationGrid, Extrusion. Les nœuds de la catégorie propriétés affectent la façon dont les nœuds géométries sont affichés. Ce sont nommément par exemple les nœuds Appearance, FontStyle, Material, LOD, etc…Les nœuds de la catégorie évènements permettent d'associer aux objets géométriques des comportements définis par des objets récepteurs (SphereSensor, TimeSensor, etc…) , des interpolateurs (OrientationInterpolator, NormalInterpolator, etc…) et une instruction de routage (ROUTE). Enfin, les nœuds de la catégories groupes permettent de gérer des collections de nœuds comme un seul objet auquel lier tel ou tel traitement. Chaque nœud parent peut contenir zéro ou plusieurs child nodes (nœuds enfants), et peut lui-même être un child node.
-Un ensemble de paramètres spécifiques à ce nœud, appelés fields (champs) . En VRML on distingue les champs qui ne contiennent qu'une seule valeur (nom commençant par "SF", single field), et les champs qui contiennent des valeurs multiples (nom commençant par "MF", multiple field). Les champs multi-valeurs sont écris comme une série de valeurs séparés par des virgules, le tout entre crochets ("[]"). A titre d'exemple, citons quelques-uns de ces types de champs: SFBool (Un champ de ce type contient une valeur booléenne 0 / FALSE ou 1 / TRUE) SFColor (Un champ de ce type permet de définir une couleur RGB par trois valeurs allant de 0.0 à 1.0.), SFFloat (Pour les champs de type flottants), SFRotation (Un champ de ce type permet de définir une rotation sous forme de 4 valeurs séparées par des espaces), MFVect2f (Un champ de ce type permet de stocker une liste de vecteurs 2d), MFVect3d (Un champ de ce type permet de stocker une liste de vecteurs 3d), etc…
-Un nom qui permet d'identifier le nœud et de faire référence à l'objet plus loin dans le document VRML par exemple.

structures hiérarchiques appelées *scene graphs*. Les *scene graphs* définissent des états qui se propagent à l'intérieur de la scène. Un mécanisme de séparateur est défini de manière à borner les effets de cette propagation à certaines parties de la scène[106]. La modélisation de scènes en VRML consiste à décrire cette scène sous la forme de hiérarchies de nodes, dans le cas le plus simple sous la forme d'un fichier ASCII. On peut disposer aujourd'hui également de nombreux outils de créations de scènes commerciaux. Par exemple, Silicon Graphics offre avec Cosmo Worlds un éditeur de fichiers VRML. La plupart des logiciels de DAO permettent également l'exportation de scène au format VRML.

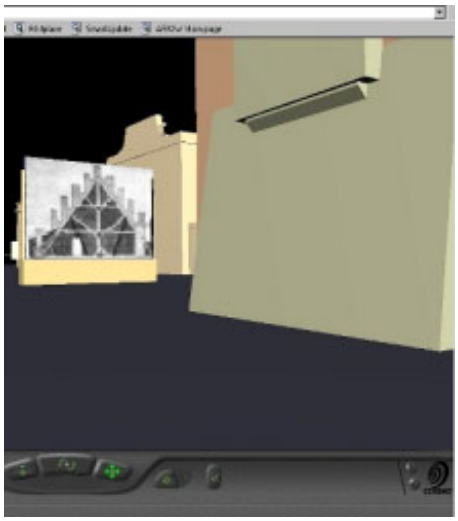


*Figure 34 : Placage de texture par URL distant en VRML , visualisation sur le plug-in CosmoPlayer2.1*

Pourtant, on l'a vu, VRML permet d'aller bien au-delà d'un simple représentation géométrique d'objets en basant la représentation des objets sur deux concepts importants: nodes enfants / nodes parents et propriétés associées aux nodes. En transformant systématiquement les objets d'une scène en node du seul type IndexedFaceSet, les modules d'exportation au format VRML que nous connaissons à ce jour nient à l'évidence les capacités réelles de ce langage à représenter une sémantique plus complexe[107]. VRML est aujourd'hui dans le domaine de l'architecture au cœur de nombreux travaux de recherche sur la conception en collaboration distante, on peut se reporter par exemple à [Oxman, 1999] qui intègre une problématique de typologies architecturales pour lesquelles VRML doit servir d'interface[108]. Nous nous devons cependant d'en rappeler les inconvénients :

- Pas d'opérations booléennes.
- Temps d'affichage long (selon la complexité de la scène, et en particulier en cas de texturation des surfaces faisant appel à des fichiers image distants).
- Temps de calcul long lorsque le modèle se complexifie, et calcul effectué côté client donc sans maîtrise par le concepteur de la scène.
- Modèle de rendu simple.

Conséquence de ces points, les scènes VRML aujourd'hui construites dans le cadre de la recherche ou de l'entreprise restent le plus souvent sommaires[109]. Il faut pourtant bien noter que VRML n'a pas pour vocation de remplacer ou de concurrencer les outils de DAO ou d'imagerie. Avant tout, VRML est un

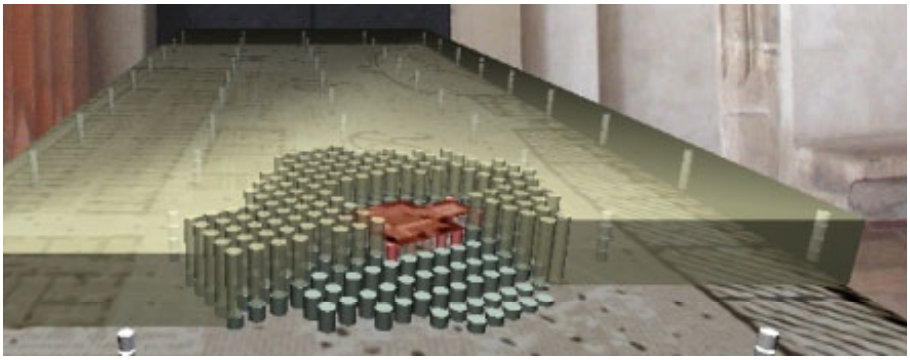


*Figure 35 : Eventail de rendus d'une maquette au format VRML 2.0 (placage de textures, transparence, etc..) insérée dans un cliché photographique (im auteur).*

[106] On trouvera une description détaillée de chaque nœud sur le site Web du Consortium VRML à l'origine des spécifications du langage : http://www.web3d.org/Specifications/VRML97/part1/nodesRef.html

[107] Nous verrons que ce point est à l'origine de notre choix visant à produire directement une représentation en VRML du modèle architectural. Nous reviendrons également sur certains avantages considérables de ce format (représentation des niveaux de détails, facilités d'animation, portabilité, mécanismes de groupages) qui permettent de prendre en compte la notion de points de vue sur le modèle dès sa visualisation.

[108] On peut également se reporter à [Bourdakis et al, 1999] ou [Roberts, 1999] qui évaluent le rôle possible de la plateforme VRML dans l'enseignement de l'architecture. On trouvera enfin dans [Geroimenko, 1999], une expérience centrée sur le mécanisme de texturation interactive en VRML.

[109] On peut se reporter aux travaux menés à l'Ecole Polytechnique fédérale de Zurich (http://caad.arch.ethz.ch/) pour un aperçu (en forme de démenti?).

langage de description de scènes statiques ou animées pour le web, dédié notamment à la mise en œuvre d'hyperliens. Le problème posé ici est donc avant tout celui de la prise en compte par le concepteur de la logique de ce langage.
En gros, il est généralement reproché au format VRML de ne pas savoir faire ce pour quoi il n'a pas été conçu. L'économie de la scène est en définitive du ressort de son modélisateur. Nous considérons que VRML est un langage adapté à la génération de scènes conçues comme des interfaces permettant, au travers de représentations correctes de l'édifice, de naviguer entre différentes vues sur celui-ci. Nous l'employons dans ce seul cadre et soutenons que ce langage apporte en terme d'interface tridimensionnelle de requête sur le Web un ensemble de solutions aujourd'hui sans réel équivalent. Nous tenterons d'illustrer ce propos dans la partie de ce document consacrée au projet. Nous considérons enfin qu'au-delà de ces services, VRML apporte à la visualisation de scènes des formalismes de représentation d'éléments de sémantique du modèle (niveaux de détail et représentations alternatives, etc..) qu'aucun outil de CAO commercial n'est aujourd'hui capable d'exprimer à notre connaissance.

#### 3.5.2.iv) Informatique graphique et formats 3D pour Internet : Bibliographie

Nous proposons, classée par ordre alphabétique, une liste des références citées dans cette partie de l'état de l'art à la fin du présent document, section "bibliographie thématique" (section 7.2.4).

### 3.5.3. Interfaces pour le réseau Internet

Nous présentons en **annexe 31** les choix que nous avons fait en terme d'architecture client-serveur sur le réseau Internet. Un des objectifs de notre travail est le développement d'une plate-forme de travail collaboratif sur le Web dont un modèle architectural de l'édifice serait l'objet principal et l'interface privilégié.
Nous présentons en annexe les éléments suivants :

- Aspects essentiels du réseau Internet (sous la forme de définitions terminologiques)
- Etat actuel et perspectives de développement des langages de formatage de données
- ou de programmation
- Interfaçage client-serveur et communication avec des applications
- Portée et utilisation du langage Javascript pour le traitement de choix utilisateur
- Tour d'horizon des solutions récentes ou en devenir, avec en particulier un regard sur XML

Si ces points ne constituent pas le centre de notre travail, ils n'en reste pas moins que nous nous sommes fixé comme objectif de confronter notre modèle à des outils dont ces points forment l'arrière plan : nous pensons donc qu'il est légitime de situer au moins en annexe un contexte technique pour nos applications.

#### 3.5.3.i) Interfaces réseau : Bibliographie

Nous proposons, classée par ordre alphabétique, une liste des références citées dans cette partie de l'état de l'art à la fin du présent document, section "bibliographie thématique" (section 7.2.5).

# 4.MÉTHODOLOGIE DE CONSTITUTION DU MODÈLE

## 4.1. OBJECTIFS

Le projet de recherche auquel ce travail faisait initialement référence avait pour objectif de proposer "*un outil de formalisation des connaissances architecturales et d'exploitation de la mesure photogrammétrique"* [Florenzano et al, 1997a]. Ce projet, nommé PAROS, se situait donc à l'articulation des domaines de la modélisation de la connaissance et de la photogrammétrie architecturale[110]. Trois axes de recherche étaient proposés :

- Formalisation centrée objet des modèles théoriques : classer par spécialisation et agrégation les objets étudiés (architecturaux et géométriques).
- Confrontation Modèle / Mesure : une étude par lecture comparative où l'aide au diagnostic architectural est basée sur l'évaluation des écarts entre le modèle théorique et le résultat du traitement des données photogrammétriques.
- Mécanismes de simulations / reconstitution d'édifices : utiliser les définitions typo-morphologiques répertoriées pour assembler, «au mieux », les éléments d'architecture.

Nous considérions donc le modèle à constituer comme support nécessaire de la mesure, permettant d'en interpréter les résultats. Dans ce cadre, les jeu de concepts à formaliser relevait naturellement d'une analyse morphologique de l'édifice, et qui plus est d'une analyse visant à coupler des formes théoriques et des outils permettant d'en observer des instances.
La méthode proposée et pérennisée dans les suite du projet initial rend compte de deux contraintes liées aux spécificités du domaine d'application :

- Domaine de production normatif dans ces principes mais localement irrégulier (par suite de destructions partielles, de réemplois, etc..).
- Relevés dont les cahiers des charges privilégient ici, l'économie de mesurage, là, la précision ou l'exhaustivité.

A ses débuts orienté vers un processus de dimensionnement du modèle architectural par le biais d'un relevé photogrammétrique adapté [Florenzano et al, 1996a], le projet s'est étendu et diversifié vers de nouveaux outils d'analyses du bâti :

- Outil d'aide à la gestion du patrimoine architectural au travers d'un modèle synthétisant un ensemble de connaissances hétérogènes. Les informations manipulées par les différents acteurs du processus de gestion du patrimoine, historien, archéologue, géologue, sociologue, architecte sont à représenter au sein d'un même modèle fédérateur.
- Outil d'analyse des pathologies, soit par confrontation Modèle / Mesure, soit par l'interfaçage du modèle théorique dimensionné par la mesure avec des outils de calcul numérique utilisés par les mécaniciens
- Outil interdisciplinaire de mise en scène d'hypothèses archéologiques. L'élaboration d'un langage de description du modèle théorique de l'édifice permet de mettre en scène diverses hypothèses archéologiques autour de quelques entités architecturales mesurées et de représenter ainsi plusieurs états antérieurs possibles de l'édifice.

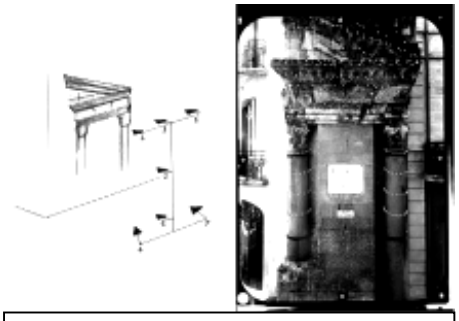


*Figure 36 : Le relevé photogrammétrique initié dans le projet PAROS, schéma de prise de vue, cliché intégrant les points mesurés à la surface des objets relevés*

[110] Le programme PAROS (Photogrammétrie Architecturale et Restitution par Outils de Synthèse) proposait donc un outil de production de relevés architecturaux s'appuyant sur une formalisation de données patrimoniales. On s'attachait à rechercher une méthode dans laquelle la problématique du relevé (traitement d'informations géométriques) intégrait celle du patrimoine architectural (gestion de connaissances). PAROS est un programme de recherche soutenu par la MRT (Mission de la Recherche et de la Technologie) du ministère de la culture français

Notre travail interroge donc à la fois les domaines liés aux techniques du relevé architectural (outil photogrammétrique, modélisation des connaissances...) et les domaines liés à l'analyse et la représentation du patrimoine bâti (étude de pathologies, simulation d'hypothèses,..). Il pose enfin et surtout le problème de l'existence d'un modèle de l'édifice capable de "*se substituer à lui pour l'étudier*" [Randall et al, 1993].
La stratégie que nous avons développée s'appuie sur l'apport des langages informatiques orientés objet en matière de formalisation des domaines de connaissance complexes. Les disciplines abordées à chaque étape du processus de traitement de la mesure comme en gestion de données seront analysées en fonction du postulat d'un raisonnement par classification rendu possible par cette technologie [Florenzano et al, 1997b].
Au regard des objectifs initiaux, le projet dans son état d'avancement actuel permet de valider l'approche défendue, celle d'un relevé s'appuyant sur une connaissance a priori du corpus de l'édifice à mesurer. Il est cependant à noter que le développement du projet aujourd'hui, s'il reprend les principes mentionnés ci-dessus, introduit de nouvelles contraintes et de nouveaux objectifs qui ont justifié une refonte complète des développements[111].

Nous avons choisi de construire autour de la plate-forme Web un ensemble d'applications dédiées à des points de vue sur l'édifice, et qui s'appuient sur notre modèle.
Un pas essentiel dans la construction d'un outil à l'intersection des disciplines respectives des intervenants consiste à définir un ensemble commun de concepts architecturaux, aptes à représenter l'édifice étudié comme à faire l'objet d'une formalisation informatique. Notre contribution est centrée sur l'élaboration de ce *modèle architectural* sur lequel s'appuie le projet, et sur ses applications aux disciplines et préoccupations des architectes-conservateurs, et sur la mise en œuvre d'outils utilisant les technologies du Web. Nous avons souhaité, dans ce travail, mettre l'accent sur les terrains d'expérimentation abordés à l'occasion d'un programme de coopération franco-polonais dans lequel est posé le problème de l'interdisciplinarité, et de la pertinence du modèle architectural que nous proposons au regard de leurs points de vue.
Nous allons dans cette section procéder en deux temps. Nous introduirons d'abord notre proposition méthodologique, puis nous détaillerons le jeu de concepts effectivement formalisés. Les implémentations et applications seront décrits dans le chapitre suivant.

*Figure 37 : Eléments du modèle visualisés en VRML et terrain d'expérimentation (corpus des plafonds en bois, im auteur/ P.Drap)*

[111] En effet, l'outil ou les outils sur lesquels nous travaillons se veulent :
- Interdisciplinaires.
- Indépendant des plates-formes matérielles.
- Complémentaires entre eux.
- Evolutifs.

## 4.2. PROPOSITION POUR UNE DESCRIPTION DE L'EDIFICE PAR ANALYSE ET CLASSIFICATION MORPHO-STRUCTURELLE

Nous proposons une description de l'édifice patrimonial qui s'appuie sur le formalisme objet.
[Blair et al, 1996] proposent comme définition transversale de ce qu'est l'approche objet "*l'identification de la notion d'objet comme l'élément fondamental autorisant la mise en œuvre de traitements*". Ils détaillent deux points de vue sur ce qu'est un objet :

- L'objet au niveau conceptuel (ce qu'il est pour l'utilisateur)
- L'objet tel que formalisé dans une application (ce qu'il est pour le programmeur).

Partant de là, ils proposent comme définition d'un objet au niveau conceptuel "n'importe quelle entité identifiée au sein du système en développement". Du point de vue de l'implémentation, ils considèrent un objet comme recouvrant complètement le concept d'encapsulation.
Le domaine de l'architecture sera pour nous étudié au travers d'édifices patrimoniaux vus comme une combinatoire d'objets ainsi définis dont les propriétés décrivent morphologie, position et fonctions dans l'édifice ainsi que les informations non graphiques qui s'y rattachent. Le langage architectural s'analyse donc par une discrétisation de l'édifice en éléments univoques. L'approche objet est utilisée pour formaliser ces modèles et les organiser hiérarchiquement par l'observation de similitudes. L'édifice patrimonial est ainsi décrit par l'observation de régularités qui tiennent autant à des typologies qu'à des observations structurelles.

La classification des éléments proposée revient à construire une hiérarchie de concepts (un ensemble de classes au sens de la programmation orientée objet) et à formaliser les relations qui les lient [Turk et al, 1993]. Nous nous appuyons sur des modèles canoniques d'architecture dans un raisonnement de type intensionnel.
L'édifice est considéré comme un assemblage de ces entités au moyen de relations de composition prédéfinies, thème auquel plusieurs travaux de recherche font référence comme [Eastman, 1994] ou [Watanabe, 1994]. Mais quelle définition donner à ces entités, quel point de vue sur l'édifice représentent t'elles ?
Nous proposons une description du domaine dans laquelle trois grandes hiérarchies de concepts architecturaux sont implémentées avec pour objectif de représenter un objet physique mesurable. L'architecture ne se confond en effet pas avec les limites physiques (murs, baies, poutraison, ..) qui peuvent caractériser l'édifice. Elle peut également être décrite par un coût de maintenance, par un ensemble d'usages, etc.. Nous devons dès lors en préambule caractériser l'échelle (au sens de [Boudon, 1977]) que nous privilégions pour notre description. Nous nous intéressons aux éléments de corpus du bâtiment, objets physiques mesurables. Les grandes hiérarchies de concepts architecturaux que nous proposons correspondent donc à une étude typo-morphologique et structurelle de l'édifice:

- Entités architecturales (morphologie et fonction univoques).
- Réseaux (jeu de relations univoque).
- Attributs (éléments de description morphologique univoques participants à la description des entités architecturales).

Nous présentons ci-après, dans la section intitulée "définition du modèle architectural", les différences faites entre ces concepts et leurs hiérarchies respectives[112].
Notre approche vise à reconnaître au sein du système entités-relations [Florenzano et al, 1998a] qu'est l'édifice les éléments de base du raisonnement de l'expert des domaines allant de l'histoire de l'architecture au relevé photogrammétrique. Nous avons choisi de nous

[112] Nous le ferons d'une part au travers des règles d'identification des concepts que nous avons identifiées et d'autre part par l'illustration, repoussée en annexe, de l'application de ces règles à des cas concrets.

appuyer notamment sur le travail de [Pérouse De Montclos, 1988] qui répertorie un vocabulaire "partagé" du domaine. Sur la base de ce vocabulaire, nous proposons (voir ci-après) un canevas de règles visant à isoler au sein du corpus correspondant à chaque expérimentation les concepts architecturaux à modéliser. De nombreuses classifications des éléments d'architecture ou du vocabulaire qui s'y rapportent sont mentionnées dans la bibliographie. Leur nombre témoigne de la nécessité d'une description univoque des concepts que nous souhaitons manipuler mais aussi de la difficulté de la mettre en œuvre. En effet, si nous avons présenté l'édifice patrimonial comme régulier, modélisable, il n'en reste pas moins que l'Architecture construite ne peut être considérée comme répondant d'une norme stable dans l'espace et dans le temps. Le modèle architectural que nous proposons est donc une ossature qui fixe des grandes hiérarchies de concepts architecturaux mais aussi qui doit s'enrichir dynamiquement en fonction du corpus traité.
Le passage "du terme au concept" peut s'analyser à travers la grille que propose [Rastier, 1995] et qui distingue quatre étapes :
(i) La nominalisation donne pour forme canonique du terme le nom substantif (…). La nominalisation est fort utilisée pour créer un effet d'objectivation (…).
(ii) La lemmatisation permet ensuite, outre sa commodité lexicographique, de dépouiller de ses variations accidentelles la substance que le terme est censé représenter (…). Felber [Felber, 1987, p. 82] affirme ainsi : "le terminologue [...] ignore les déclinaisons et la syntaxe".
(iii) La décontextualisation permet de définir le terme par lui-même, indépendamment des variations qui pourraient affecter ses occurrences (…) un terme n'a pas de contexte, il n'a que des pères, des frères, et des fils ; et les relations qui le lient à ces termes apparentés sont seules recevables et pertinentes.
(iv) La constitution du mot en type, et l'affirmation corrélative que toutes ses occurrences sont subsumées sous ce type -- ou du moins que celles qui ne le sont point témoignent d'un emploi incorrect. La définition est le moyen principal de cette constitution en type : elle énonce, conformément au principe du positivisme logique , les conditions nécessaires et suffisantes pour que le terme soit pourvu de sa dénotation correcte. La notion (ou concept) est la signification du symbole (ou mot).

La problématique de représentation des concepts architecturaux à laquelle nous sommes confrontés croise celui de la terminologie dans la définition qu'en donne Felber «Domaine du savoir interdisciplinaire et transdisciplinaire ayant trait aux notions et à leurs représentations (termes, symboles, etc.) » [Felber, 1987]. En effet, les notions, entités conceptuelles, priment en terminologie sur leurs expressions [Rastier, 1995]. L'élaboration du corpus de concepts architecturaux à partir des apports lexicographiques dont nous nous servons se fait par filtrage des termes rencontrés pour en extraire des concepts correspondant aux trois hiérarchies retenues. Ce filtrage peut par commodité être illustré par la grille rappelée ci-dessus. Notre première étape , comparable à la nominalisation, sera l'étape de lecture des lexicographies. Une deuxième étape consiste à défaire chaque terme ou expression des déclinaisons morphologiques du terme (la soliveAProfil = une solive + un profil, le Profil= un jeu de moulures renvoyant à une typologie de la mouluration). Une autre étape consiste alors à sortir l'élément d'architecture que désigne le terme (représentant le concept que nous cherchons à isoler) des conditions matérielles de son emploi (la soliveAncrée= une solive + un ancrage, l'arcature aveugle [Pérouse De Montclos, 1988] = un jeu d'arcs et de remplissages). Un terme univoque peut alors désigner à la fois les propriétés typo-morphologiques de l'élément considéré et son usage au sein de l'édifice.
L'artefact se décrit en définitive par l'usage d'un vocabulaire non ambigu correspondant aux concepts représentés et organisés au sein de hiérarchies.

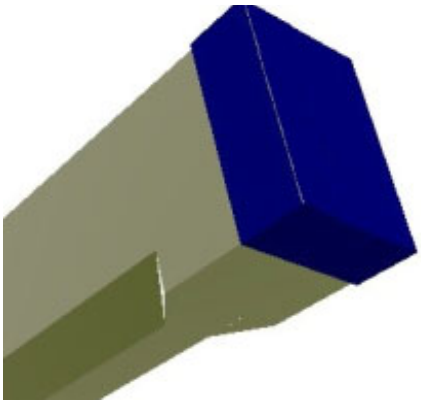


*Figure 38 : La solive et son engravure, un concept architectural et un mode d'assemblage (maquette VRML, im auteur)*

### 4.2.1. Définition du modèle architectural

Nous abordons la définition d'un modèle de l'édifice patrimonial en plaçant une échelle de référence, celle de l'entité architecturale, élément du corpus des objets physiques. Cette échelle est choisie car elle est pertinente au regard des besoins essentiels exprimés dans notre projet : la mesure de l'édifice et la gestion de données relatives à ses évolutions morphologiques. Nous situons en rapport avec cette échelle des notions gigognes (réseaux, attributs) et des règles d'assemblage (relations). Puis, nous fédérons dans la représentation de ces notions un ensemble de données que l'on peut appeler points de vue sur le modèle par opposition à la ligne de division fondatrice de celui-ci qu'est l'identification non ambiguë d'objets physiques du corpus architectural.

Nous allons dans cette section d'abord discuter des quatre concepts essentiels du modèle que nous présentons : entités, attributs, réseaux et relations. Nous en donnerons une définition aussi claire que possible puis en annexe analyserons quelques exemples illustrant notre démarche. Nous présenterons également en annexe de courts extraits de la structuration hiérarchique de ces concepts traduite sous forme d'arbre de classes. Rappelons que nous nous détaillerons le jeu de concepts effectivement formalisés dans la section suivante.

Nous terminerons cette section par une discussion sur l'idée de norme en architecture patrimoniale et sur ses répercussions sur la constitution du modèle, avant de conclure par des remarques sur des limitations volontaires à ce modèle. Nous verrons dans la section intitulée *interfaces* quelles caractéristiques supplémentaires qualifient nos quatre concepts afin de leur associer des points de vue comme par exemple la mesure.

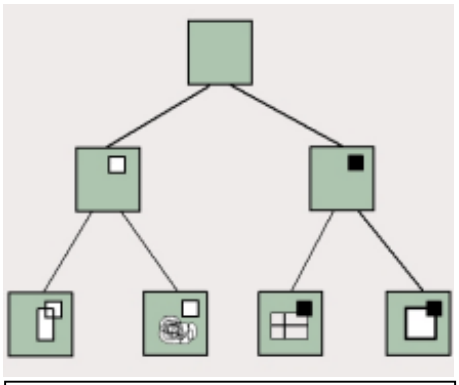


*Figure 39 : Structuration hiérarchique des concepts architecturaux, une logique de spécialisation*

#### 4.2.1.i) Les entités

La logique d'identification des entités architecturales, échelle de base de notre modèle, a déjà été évoquée au travers d'un détour par l'étude du vocabulaire. Ce biais est explicité par Jean-Claude Golvin [Golvin, 2000] dans la présentation de son intervention intitulée "Modèle versus Maquette". L'auteur rappelle la définition suivante d'un modèle : " Nous avons assemblé un certain nombre de données en un tout qui constitue *l'image mentale* de la chose. Nous appelons modèle la structure signifiante dans son ensemble à laquelle nous rattachons un sens. Or toutes les données qui la constituent ne sont pas traduisibles spatialement. En effet si l'on peut représenter en volume une colonne, on ne peut exprimer ainsi sa masse ou citer sa date de construction. Des évènements, des noms propres et de multiples données devront être enregistrées sous d'autres formes si bien que la ou les *maquettes* possibles d'un monument ne constituent en réalité qu'une partie du modèle correspondant."

A l'appui de sa démonstration, l'auteur indique que le raisonnement hypothético-déductif de l'archéologue ou de l'architecte est un raisonnement par induction dans lequel un ensemble d'observations non démenties permet d'établir une catégorisation des objets observés. C'est bien dans ce cadre que nous avons travaillé puisque la définition des entités architecturales que nous proposons spécifie des règles d'observation devant guider le choix des concepts à isoler. Enfin, nous nous situons clairement dans la démarche qu'évoque Jean-Claude Golvin, "construire le modèle c'est construire le concept et montrer les sources". Le modèle architectural que nous proposons, et la définition de catégories d'entités, correspondent à l'édiction de règles générales fixant un jeu de symboles dont une des représentations est traduisible spatialement.

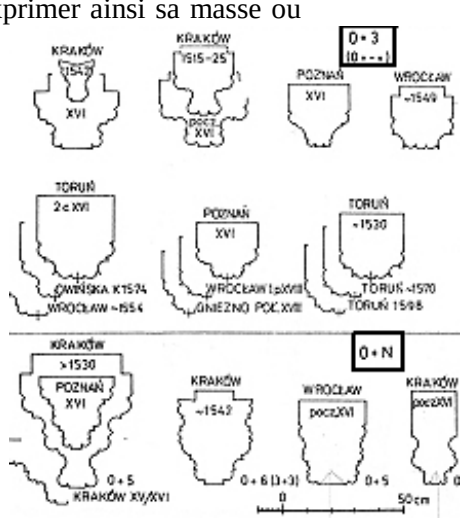



*Figure 40 : Catégorisation des concepts, par observation d'un ensemble d'objets, le cas des plafonds en bois [Tajchman, 1989]*

*4.2.1.i.1. Identification*

Nous avons reconnu une demande : disposer de concepts aptes à représenter l'édifice. Restons ici dans le champ de l'architecture patrimoniale et illustrons par l'exemple de l'oculus ci-contre photographié dans la ville de Brignoles la nécessité d'isoler des concepts proprement architecturaux. Tout intervenant dans le domaine l'architecture patrimoniale reconnaîtra l'élément figuré comme étant un "oculus". Pourtant, par l'action du temps, l'objet originel n'est plus identifiable :

- Par sa fonction d'ouverture (de baie) car il est rempli.
- Par sa fonction structurelle (intrados déplacés).
- Par sa morphologie (ronde ou elliptique).
- Par les dimensions relevables à sa surface (deux courbes aujourd'hui, un diamètre ou les petit et grand diamètres à l'origine).

*Figure 41 : Détermination d'un objet dégradé, le cas de l'oculus de la place des comtes de Provence, Brignoles, Var*

Comment alors expliquer que la détermination de la catégorie à laquelle appartient cet objet ne souffre d'aucune ambiguïté ? Aucune de ces quatre lignes de division prises isolément ne permet à l'évidence d'établir l'appartenance de cette instance à un modèle d'oculus. Il faut selon nous avoir recours à une définition en creux pour s'assurer que l'objet observé n'a pas à la fois une forme et une fonction autre que celles, hypothétiques, que nous déduisons de son état actuel, et pour s'assurer que l'élément n'est pas décomposable en sous-éléments sans perdre son identité (c'est à dire sans laisser vide de sens le terme le désignant). Nous ne sommes pourtant même dans ce cas pas à l'abri de réutilisations incongrues ou d'enfouissements dans des structures ultérieures. Dans les deux cas illustrés ci-contre, les baies originales n'ont plus leur fonction d'ouverture. A gauche, la baie est encore visible par ses piédroits et son arc, intacts, en appareillage de blocs (Palais des comtes de Provence, Brignoles, Var). A droite, la structure dégagée par de récents travaux de réfection ne correspond plus à la définition canonique de notre modèle de baie dans lequel figurent linteau, appui et deux piédroits (Budynek Copernicus, Ulica Kanonicza, Kraków, Pologne). La structure est pourtant reconnaissable à l'œil comme étant le vestige d'une baie de période renaissance.

*Figure 42 : Problèmes de reconnaissance d'objets réemployés, baie remplie du Palais des comtes de Provence, Brignoles, Var, et baie enfouie de la façade sur rue du Budynek Copernicus, Cracovie, Pologne*

Le modèle que nous proposons peut gérer le premier cas avec une relative facilité. La baie a été dotée d'une propriété nommée remplissage qui dans le cas général est constitué d'une liste d'objets (menuiseries) mais peut également contenir une instance de la classe mur correspondant au remplissage observé ici. Par contre, le second cas pose plus de problèmes puisque la morphologie de la structure observée ne correspond plus à celle du modèle de baie pertinent. Dans l'état actuel de nos développements, nous allons pouvoir instancier une baie dont les propriétés (qualitatives et quantitatives) seront renseignées mais dont la représentation ne figurera pas l'incomplétude de l'objet réel observé. Ce point constitue bien entendu une perspective pour notre projet.

Il nous semble néanmoins que la définition que nous donnons de l'entité architecturale ci-dessous permet d'isoler avec assez d'acuité des catégories d'objet. Le patrimoine architectural est un domaine dans lequel des concepts stables (le concept de couverture, de couvrement, etc..) s'accompagnent d'éléments de variabilité historiques ou morphologiques. Une description fine du domaine s'impose donc pour isoler d'une part des objets non-ambigus, nos entités architecturales, et d'autre part leurs éléments de variabilité contextuelle (interrelations,

moulurations, réutilisations, etc.). Le modèle proposé devra par conséquent intégrer l'exigence de variabilité morphologique ou fonctionnelle que requiert notre domaine d'étude particulier. Premier élément de notre proposition, nous proposons pour l'entité architecturale, notion pivot de notre modélisation, la définition suivante :

Une entité est reconnue comme telle pour autant que :
1. L'entité soit un objet identifié par un élément singulier du vocabulaire architectural.
2. L'entité ait un rôle permanent et indivisible dans la structure de l'édifice,
3. L'entité intervienne de façon autonome dans le système de relations topo-morphologiques de l'édifice.

Autrement dit, les entités architecturales répondent à deux questions :
- L'objet étudié a t'il à la fois une forme et une fonction univoques (la colonne, par exemple, a une fonction univoque mais une forme qui ne l'est pas puisqu'elle peut contenir indifféremment une base et un chapiteau : ce n'est pas une entité).
- L'objet s'utilise t'il indépendamment du rapport à un autre objet (le chaperon, par exemple, couronne un mur et seulement un mur, ce n'est pas une entité).

Nous isolons donc des catégories d'entités pour lesquelles des similitudes de comportement ou de forme sont observables, et gérons les spécificités morphologiques intra- entités (profils par exemple) au travers des catégories d'attributs décrites plus loin. Dans la hiérarchie d'entités architecturales, chaque entité porte potentiellement un ensemble d'informations graphiques ou non graphiques permettant par exemple :
- de coupler sa représentation tridimensionnelle avec un ensemble de références bibliographiques la concernant,
- de coupler sa description morphologique théorique avec un outil de visualisation spécifique, etc.

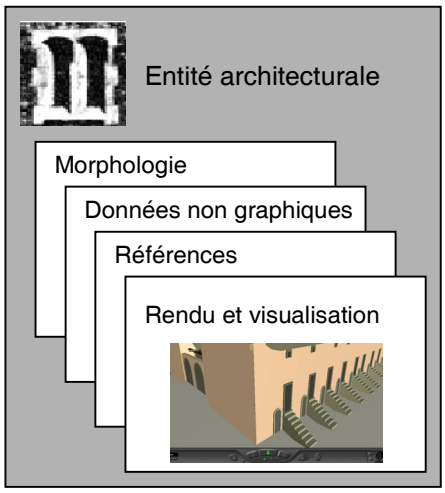



*Figure 43 L'entité, réceptacle d'informations quantitatives et quantitatives*

Le travail d'analyse du corpus effectué à ce jour a porté dans un premier temps sur l'architecture antique[113] puis sur les terrains d'expérimentation dont nous ferons état par la suite[114], à savoir l'ancien hôtel de ville de Cracovie, *Kramy Bogate*, les anciens plafonds en bois des maisons urbaines et la place du marché (Rynek Główny) de cette même ville.
Les entités sont définies par un ensemble de propriétés, dont en particulier des éléments fixant leur morphologie, et par un ensemble de méthodes qui correspondent à leur interfaçage, comme en particulier les méthodes d'écriture au format VRML. Naturellement, la hiérarchie d'entités architecturales est produite par ajout de propriétés aux entités les plus génériques. L'hypothèse fondatrice de ce travail est donc qu'il est possible de représenter les éléments du corpus architectural dont font état nos sources bibliographiques au travers de hiérarchies de classes capables d'exprimer un modèle de l'édifice.
Nous avons souhaité en **annexe 32** illustrer notre démarche d'identification des concepts avec l'exemple d'une partie du corpus des plafonds en bois[115] en nous appuyant sur le travail de Jan Tajchman[116] . L'auteur distingue dans l'ouvrage qu'il consacre à ce sujet plusieurs classifications pour rendre compte des caractéristiques du corpus. Pour nous, même si cette

[113] Voir notamment [Florenzano et al, 1998] , [Florenzano et al, 1997b], [Florenzano et al, 1997a], [Florenzano et al, 1996b], [Florenzano et al, 1996a],[Drap et al, 1995].
[114] Voir notamment [Drap et al, 1998b], [Czubinski et al, 1998a], [Drap et al, 1999b], [Dudek et al, 1999b], [Drap et al, 1999a], [Dudek et al, 1999a] .
[115] Précisons que nous parlons du corpus des plafonds en bois bâtis en Pologne entre les treizième et dix-huitième siècles (dits de période gothique, renaissance et baroque), et utilisés en majorité dans les maisons urbaines.
[116] Voir [Tajchman, 1989], habilitation en architecture, université de Torun (Pologne)

contribution constitue une formidable base de travail, la logique de classification proposée ne peut être absolument pas prise en compte telle quelle, ce que nous démontrons en annexe 32. Le pas que nous avons franchi pour construire une hiérarchie cohérente de concepts à partir d'une part du travail de J.Tajchman et M.Łukacz et d'autre part de nos règles d'identification nous semble suffisamment explicite et exemplaire pour être rapportée. Nous renvoyons aussi en **annexe 33** une présentation de la hiérarchie des principaux concepts architecturaux en jeu pour représenter le corpus des plafonds en bois.

*4.2.1.i.2. Organisation des concepts*

Nous avons tenté d'illustrer la méthode utilisée pour identifier cette échelle de l'entité architecturale dont nous faisons le pivot de notre modèle au travers d'un exemple qui situe bien la place prise dans ce processus d'identification d'objets par la documentation. En bref, nous pouvons remarquer que si cette documentation nous permet d'identifier les concepts architecturaux correspondant à l'échelle des entités, elle ne nous donne que peu d'indices sur :

- D'une part, les relations entre les entités et les concepts relevant d'échelles différentes (le décor par exemple).
- D'autre part, une logique d'organisation hiérarchique des entités tirant parti du formalisme de représentation des connaissances choisi, l'approche objet.

Si nous avons avec l'exemple du corpus des plafonds en bois partiellement répondu à la première question, nous devons maintenant nous intéresser à la seconde. Notre démarche d'organisation des concepts aboutit à la formation d'une hiérarchie d'entités dans laquelle l'élément le plus haut dans la hiérarchie, la classe entité, classe abstraite, fédère un certain nombre de propriétés communes à tous les concepts de cette échelle. Les sous-classes de la classe Entité sont déterminées par ajout de propriétés, non sans contradictions[117].
Mais nous ne revenons pas ici sur ces points. Nous allons plutôt expliciter sur la base de quelles propriétés les entités se spécialisent, étant entendu que leur définition fait à la fois référence à leur forme et à leur fonction.
Il s'agit donc ici de dire si l'ajout d'une propriété morphologique suffit à former un nouveau concept, ou bien encore si le changement de fonction (structurelle) d'un objet conduit à la création d'une nouvelle classe (par exemple, la baie murée présentée précédemment).

Deuxième élément de notre proposition, nous proposons pour l'entité architecturale, notion pivot de notre modélisation, trois logiques de spécialisation complémentaires :

Une entité architecturale est dérivée peut être dérivée par :
1. Spécialisation par fonction structurelle
2. Spécialisation par propriétés morphologiques
3. Spécialisation par dimensions observables

Quelques compléments doivent cependant être ajoutées à cette définition :

- Spécialisation par fonction structurelle : nous isolons ici des catégories d'entités auxquelles nous attachons des propriétés qualitatives ou quantitatives exprimant le rôle que jouent les entités de ces catégories dans la structure de l'édifice. Par exemple, la classe Support pourra être dotée de propriétés permettant de gérer la charge portée par les entités en dérivant et d'éventuels porte-à-faux, ou la classe Couvrement de propriétés établissant leur portée et leur ancrage indépendamment des dimensions globales de l'entité. Cette logique de spécialisation nous permet d'établir des catégories d'entités en tête de la hiérarchie, le plus souvent classes abstraites sans définition morphologique

[117] L'hagioscope, par exemple, ne se distingue de la baie que par l'angle de vue particulier qu'il permet sur le maître-autel d'une église, propriété difficile à prendre en compte. De même la chantepleure est une sorte de baie qui se distingue des autres par le fait que, placée dans un mur de terrassement pour permettre l'évacuation des eaux, elle ne reçoit pas de fermeture (identification par soustraction de propriété).

propre. On retrouve donc sous la classe Entité un ensemble de concepts spécialisé en référence aux seules questions fonctionnelles.

- Spécialisation par propriétés morphologiques : nous isolons à l'intérieur des catégories d'entités citées au point précédent des concepts dérivant les uns des autres par ajout de spécificités morphologiques ou comportementales. Nous entendons ici par spécificités morphologiques l'ajout à une classe de base de propriétés dimensionnelles ou encore d'objets gérant des aspects morphologiques (profils par exemple).
  L'ajout de spécificités morphologiques correspond donc au point de vue formel sur la collection d'entités. Il permet d'exprimer par exemple qu'un arc tiers point se distingue d'un arc plein cintre par la nécessité d'ajouter à la définition du second (un rayon, un centre) notamment un deuxième centre. Cette logique de spécialisation intervient donc à l'intérieur de catégories indistinctes du point de vue de la morphologie des objets.
  Il peut être tentant de définir, pour chaque élément de modénature ajouté, une classe correspondante. Il faut toutefois ici à nouveau faire référence à la notion d'échelle : nous devons distinguer les modifications morphologiques pertinentes à l'échelle de l'entité (arc plein cintre, arc tiers point, arc brisé, etc..) et celles qui relèvent de l'échelle du décor (dessin de l'archivolte par exemple).
  En effet, une modification de la morphologie d'une entité n'indique pas forcément un changement de modèle mais peut renvoyer également à la notion de style. L'oculus polylobé n'est pas une spécialisation de l'oculus, jour dont le tracé est circulaire, ovale ou polygonal, mais un élément de décor d'un réseau, ensemble des éléments de remplage formant les divisions dans les baies à tracé curviligne.
  De la même façon, une morphologie quasi-identique n'indique pas que deux objets soient instances d'une même classe : l'arc brisé est morphologiquement proche du formeret, mais ce dernier n'est qu'un des éléments entrant dans la définition de la voûte ("arc formant nervure sous le front d'une voûte en berceau ou d'un quartier de voûte" [Pérouse De Montclos, 1988, p153] ).

- Spécialisation par dimensions observables : nous avons à l'origine du projet centré notre démarche de modélisation sur la question de la mesure. Le point de vue de la mesure sur la collection d'objets était traduit par la définition, à l'intérieur de chaque entité, de primitives géométriques observables à sa surface. Dès lors la logique de spécialisation intégrait l'idée qu'en disposant de nouvelles primitives à mesurer, un objet se spécialisait de fait. Cette forme de spécialisation a depuis lors été remise en cause.

Nous présentons en **annexe 34** un schéma reprenant la tête de la hiérarchie d'entités telle que proposée dans notre modèle, et figurant le niveau fonctionnel de la classification puis le premier niveau morphologique. Nous détaillerons dans la partie "Interfaces" le contenu de la classe de base Entité et de façon plus globale les ensembles de propriétés isolées pour chaque classe en dérivant.

#### 4.2.1.ii) Les relations

Une question mérite d'être soulevée avant même de mener une réflexion sur un formalisme pour les relations : faut-il organiser les relations en une hiérarchie de concepts ou ne les considérer que comme devant servir au montage dans l'espace des entités architecturales ? En effet, une relation ne saurait être considérée comme un concept à formaliser si elle n'intervient que dans une relation du type :

n (entités) + n (relations) = réseau

La relation n'est alors pas signifiante en tant que telle mais peut être ramenée au rang de propriété basique du réseau. Cela serait le cas si les relations dont nous discutons n'indiquaient qu'un rapport topologique simple entre des entités au sein d'un réseau figé morphologiquement. Mais la relation devient un concept à part entière dès lors qu'elle porte une signification indépendante de la nature des éléments qu'elle associe, comme par exemple

la notion de trame qui peut s'appliquer à un jeu d'entités, de réseaux ou d'attributs et plus encore est déterminante de propriétés spécifiques (ordre, type, rythme, etc..). La relation devient alors un élément déterminant du raisonnement et non une méthode (ou un outil pour rester dans un vocabulaire plus général) associée à la hiérarchie de concepts représentant les objets physiques. De plus, nous devons nous garder de limiter le terme relation à une interprétation centrée sur le positionnement relatif des objets physiques. En effet, derrière le terme de relation il faudra prendre en compte à la fois la gestion de données relatives au positionnement des objets physiques, à la typologie de ces objets, à l'évolution historique de la morphologie d'un élément structurel, aux propriétés physiques de ces objets, et enfin à leur rapport à l'histoire (présence - absence, style, bassin géographique d'emploi). Du moins identifions-nous là quelques-unes une des interprétations possibles d'une notion de relation signifiante en terme de modélisation. Il nous faudrait en réalité dès à présent constater qu'un formalisme générique de gestion des relations entre éléments physiques s'impose. Nous allons décrire les choix faits à l'heure actuelle dans ce projet et placerons cette problématique au centre des perspectives de développement du projet.

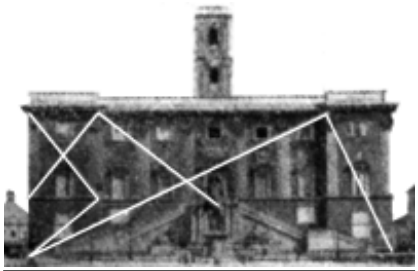


*Figure 44 : Une relation déterminante de la forme de l'édifice, analyse du tracé du Palais Farnese à Rome, par Le Corbusier [Corbusier,1958]*

Dans une première série d'expériences (voir notamment [Florenzano et al, 1998], [Florenzano et al, 1997b], [Florenzano et al, 1997a]), la notion de relation était portée par un langage de manipulation du modèle architectural, décrit dans un script lu et interprété par un analyseur syntaxique. Cette solution ne répondait pas à l'exigence d'expressivité du modèle que nous nous fixons puisque les relations n'apparaissent que sous la forme de tokens de la grammaire formelle. Nous expérimentons à l'heure actuelle une autre solution, dans laquelle une hiérarchie de concepts représentent les relations. Cette expérimentation, nous l'avons indiqué plus haut, se limite aujourd'hui aux relations de dépendances topologiques entre deux entités architecturales. C'est cette autre solution que nous présentons dans les deux sections suivantes.

*4.2.1.ii.1. Identification*

Nous nous limiterons ici à identifier, à partir de nos sources documentaires principales, les relations topologiques sur lesquelles nous travaillons. Jean-Marie Pérouse de Montclos [Pérouse De Montclos, 1988, pp22-26] recense un ensemble de vocables utilisés pour expliciter le rapport de positionnement entre un objet référent et l'objet à localiser : contrefort au droit du pilier, porte à l'aplomb de la fenêtre, etc... Cette première catégorie de vocables met en rapport un objet censé déjà positionné et un autre objet dont le localisation est donnée par référence au précédent. Jean-Marie Pérouse de Montclos comme Jean Cuisenier [Cuisenier, 1991] établissent également l'existence de relations non-unaires entre objets: l'édifice radio-concentrique [Pérouse De Montclos, 1988, p11], l'axialité, la trame, la composition modulaire [Cuisenier, 1991, p244-245] fixent un tracé dans lequel s'insèrent des groupes d'entités. On doit dès lors distinguer relations topologiques entre deux objets, dont l'un sert de référent, relations que nous nommerons par la suite dépendances, et relations entre un groupe d'entités et un tiers, modèle de composition, relations que nous nommerons par la suite compositions.

Nous avons souhaité jeter les bases d'une notion de relation, limitée à l'heure actuelle aux dépendances, mais dont le statut de concept à part entière dans notre modèle permet d'entrevoir des utilisations plus avancées. Nous ne parlerons dans la suite de cette section que de ce qui a été réalisé, laissant le reste aux perspectives. Nous nous situons donc dans le cadre de relations d'entité à entité, relations dans lesquelles nous cherchons à positionner une entité en référence à une autre, supposée correctement localisée. En nous appuyant sur la terminologie établie par Jean-Marie Pérouse de Montclos, nous pouvons proposer une définition de chaque relation sous la forme d'un vocable désignant le type de relation. Mais nous devons affiner cette définition en donnant du vocable une définition indiquant une transformation et un ensemble de paramètres pertinents.

En effet, les termes recensés par Jean-Marie Pérouse de Montclos ne constituent pas une base suffisante pour assurer un positionnement non ambiguë des objets les uns par rapport aux autres. Par exemple, le terme *au droit de* indique *un élément dont l'axe en plan coïncide avec celui d'un autre élément* [Pérouse De Montclos, 1988, p23]. Il n'est pas fait mention dans cette définition de :

- La distance entre ces deux éléments (en *plan*).
- La référence implicite à l'horizontale qu'indique le terme *plan*.
- La position relative des deux éléments sur l'axe que ne figure pas le *plan* (vertical).

On découvre ici au travers du vocabulaire utilisé la présence d'un modèle de représentation de l'édifice qu'utilise l'architecte: plans, coupes, élévations. L'édifice, et les relations entre entités qui le figurent, sont en effet le plus souvent présentés sous la forme de géométraux, même aujourd'hui puisqu'on notera que seuls les géométraux sont contractuels. Plans, coupes et élévations aplatissent le volume des éléments représentés et s'accompagnent de descriptions textuelles qui trahissent cet a priori. Le mot surplomb désigne par exemple une relation sur la verticale qui n'est figurée que sur une coupe ou une élévation. Autrement dit, une définition exhaustive de la relation entre objets passe non par un vocable mais par un croisement de vocables, chacun faisant référence à une projection en deux dimensions de l'édifice. Encore dans ce cas faut-il avoir recours à la lecture des éléments de cotation des géométraux pour donner une dimension aux relations.

Il nous fallait ici lever ces ambiguïtés ; nous avons donc réinterprété les définitions des relations topologiques que nous appelons de dépendance de façon à ce que chaque vocable choisi désigne de façon univoque une transformation. Nous adjoignons donc au vocable un ensemble nécessaire et suffisant de paramètres rendant la relation non ambiguë, et permettant de déterminer complètement la position d'une entité par rapport à une autre.

Ce choix n'était pas, nous souhaitons le dire, le seul possible. Une autre piste pourrait conduire à la détermination de relations univoques entre entités par un croisement de relations, croisement plus proche de la définition originale des vocables utilisés. Il nous a semblé pourtant que cette piste compliquerait de façon trop importante la création de maquettes numériques.

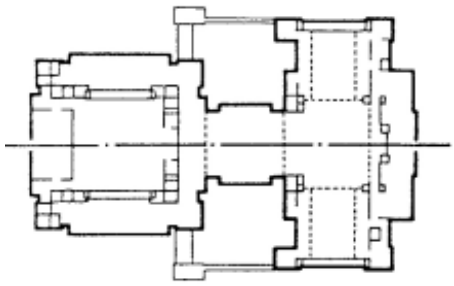


*Figure 45 : Relations entre objets définie en plan, le cas de l'Unity Temple (FL Wright)*

Nous illustrons la notion de dépendance présentée ici en **annexe 35**, en rappelant qu'il s'agit d'une première expérience ne témoignant que très partiellement de l'apport potentiel d'un formalisme de relation encore à développer.

*4.2.1.ii.2. Organisation des concepts*

La tête de la hiérarchie des relations est une classe abstraite dont les propriétés essentielles sont un identificateur et un pivot, objet représentant la ligne selon laquelle les entités architecturales seront comparées (ou plus généralement, mises en relation). Dans le cas de la hiérarchie des dépendances topologiques, le pivot est un être architectonique, axe principal de la relation par exemple. Le concept de dépendance, dérivant de la classe Relation, représente une relation ayant pour propriété une entité architecturale référente. La dépendance est donc la mise en relation d'une entité requérante et d'une entité référente. Le concept de dépendance de Position, dérivant du précédent, restreint son champ d'application au cas des relations entre une entité requérante et une entité dépendante, établies en rapport à une règle d'assemblage canonique simple comme l'aplomb. Le pivot de cette relation est un axe orienté (en fait, un référentiel), elle a pour propriété spécifique un type d'assemblage. L'établissement d'une dépendance topologique entre une entité requérante et une entité référente se limite dès lors à une suite d'appels de méthodes :

- Renseignement du pivot sur l'entité référente.
- Renseignement du référentiel de l'entité requérante par le pivot.
- Positionnement de celui-ci eu égard aux paramètres des assemblages (axialités ou alignements).

L'organisation, bien qu'embryonnaire, de la hiérarchie des relations indique les choix de modélisation que nous expérimentons. Troisième élément de notre proposition, nous introduisons une interprétation de la notion de relation qui définit celle-ci par :

1. Un concept générique de relation faisant référence à une notion de pivot, concept tiers liant les objets liés. Les liens entre entités architecturales sont donc rapportés explicitement à un point de vue au nom duquel les entités sont comparées.
2. Un concept générique de dépendance qui implique la mise en relation d'un ou de plusieurs individus avec une instance du modèle.
3. Un concept de dépendance limité aux relations unaires rendant compte des vocables désignant de telles relations. Chaque instance de ces dépendances est caractérisée par un type (le vocable désignant la relation dans le vocabulaire architectural) et par des paramètres complémentaires en fixant de manière univoque la transformation que subit l'entité requérante.

Reste aujourd'hui à expérimenter des dépendances de groupe (une entité référente, des entités requérantes). Par ailleurs, la notion de relation serait à étendre à des assemblages qui ne soient pas fondés sur des critères fixant la localisation des entités.

#### 4.2.1.iii) Les réseaux

La notion de réseau recouvre l'idée que, placées ensemble en fonction de telle ou telle logique, les entités architecturales acquièrent un statut de concept (architectural ou urbain) indépendant de ce qu'elles sont. Autrement dit, le réseau définit par ses seules propriétés (et non par la lecture des propriétés des entités qu'il contient) un élément de connaissance non ambiguë (temple diptère, Etage attique, etc..).
Dans l'état actuel du développement que nous présentons, au delà de cette définition générale, le réseau sert un peu (il faut l'avouer) de fourre-tout puisque, par exemple, il ne distingue pas d'échelle (au sens architectural du terme). Le réseau est pour nous concrètement un formalisme permettant d'analyser des agencements d'éléments du corpus, et reste donc générique. Cette approche s'est justifiée par le souci initial du projet de s'attacher à la mesure de l'édifice. Nous devions donc disposer de concepts d'assemblage neutres sémantiquement nous permettant de générer des jeux d'objets physiques libres autorisant des analyses dimensionnelles par comparaisons, indépendantes de la logique de division du modèle architectural. Nous allons néanmoins présenter cette logique en en rappelant dès à présent l'aspect prospectif. En effet, dans l'optique d'un système dit d'informations, le formalisme de réseau que nous proposons à l'heure actuelle manque d'expressivité.

*4.2.1.iii.1. Identification*

Sous le terme générique de divisions, Jean-Marie Pérouse de Montclos [Pérouse De Montclos, 1988, pp40-53] regroupe des vocables désignant soit des parties d'un édifice (le vaisseau, la galerie, etc..) soit des logiques d'agencement (l'ordonnance, l'oriel, etc…). Dans le premier groupe, majoritaire, les vocables désignent des parties d'un édifice identifiées soit par leur localisation (l'étage-attique, le porche, etc…) soit par leurs spécificités :

- dimensionnelles, en rapport avec l'édifice (le vaisseau, espace intérieur caractérisé par son développement dans le plus grande partie de la hauteur d'un bâtiment…).
- proportionnelles (la galerie, espace habitable plus long que large délimité dans un étage…).
- morphologiques (l'abside, espace intérieur de plan cintré ou polygonal …).

Dans le second groupe les vocables désignent des modes d'assemblage d'entités ou de groupes d'entités. On notera ici que la plupart des modèles d'assemblage canoniques sont présentés dans l'ouvrage cités de façon transversale par rapport aux catégories (divisions, couvrements, supports, etc…) qu'isole l'auteur. La colonnade est ainsi présentée dans le chapitre support, l'entablement dans le chapitre consacré au décor, etc…
Il nous semble que cette dualité de définition recouvre une idée déjà évoquée plus haut: l'édifice est un ensemble d'objets physiques déterminant les limites entre des espaces

intérieurs et l'extérieur. Autrement dit, les vocables utilisés pour qualifier un lieu correspondent dans la pratique à une qualification d'usage (ce qu'est l'espace délimité) ou à une qualification de bornage (comment cet espace est délimité). Or, notre modèle s'attache à représenter des objets physiques, et non les espaces que ceux-ci contribuent à clore.
Nous allons donc donner des divisions une interprétation qui ne coïncide pas avec celle que propose Jean-Marie Pérouse de Montclos mais en reprend une idée maîtresse: des concepts correspondant à des parties d'édifices (pour nous, du point de vue de leurs limites physiques) et des concepts correspondant à des modèles d'assemblage reconnus (l'arcade, le portique, etc..). Cette distinction se justifie de notre point de vue par le fait qu'elle correspond à deux préoccupations différentes :

- D'un côté, on va chercher à affecter à un groupe d'individus des propriétés essentiellement qualitatives permettant d'établir par exemple une concordance stylistique ou une stratification chronologique.
- De l'autre côté, on va chercher à tirer parti de notre connaissance théorique d'un modèle d'agencement d'objets canoniques pour par exemple compléter le jeu d'instances d'une colonnade ou d'un plafond partiellement détruit.

Même si ce choix constitue une première étape dans l'élaboration d'un formalisme de réseau, elle ne doit néanmoins pas être considérée comme autre chose que comme une direction de recherche qu'il nous reste à évaluer.
Un réseau détient un ensemble d'entités architecturales et de relations; il permet notamment :

- d'appliquer à des entités des relations,
- d'accéder aux propriétés des unes et des autres.

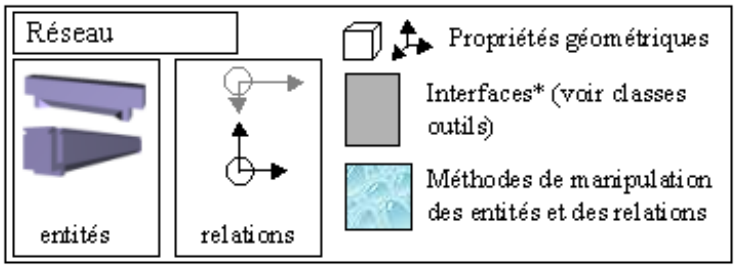



*Figure 46 : Le concept de réseau, un jeu d'entités et de relations*

En effet, dans l'implémentation que nous proposons seule l'instanciation d'un réseau (dit neutre parce qu'il ne détient pas d'éléments de sémantique propres) permet d'instancier entités ou relations et de les manipuler. Le réseau dispose de propriétés géométriques propres (référentiel, boîte englobante) utilisées notamment dans les méthodes de représentation. L'invocation de méthodes sur une entité architecturale est, dans les outils développés aujourd'hui, traité par le réseau qui se charge d'une part d'appeler la méthode en question et d'autre part de procéder à d'éventuelles mises à jour. Il n'y a donc pas encore dans notre modèle de distinction entre le réseau, mécanisme autorisant le manipulation d'un groupe d'entités et de relations, et le réseau, concept autonome dépositaire d'éléments de sémantique propres.

*4.2.1.iii.2. Organisation des concepts*

Comme indiqué précédemment, nous n'avons pas à ce jour développé une possible hiérarchie de réseaux, et n'avons proposé dans le mécanisme actuel qu'un outil. Cela fera partie des perspectives importantes de ce travail. Nous partirons du principe évoqué plus haut, d'une part des agencements et d'autre part des concordances. Autrement dit, nous traduirons par là l'idée qu'un réseau est un outil d'analyse d'un groupe de concepts architecturaux ayant pour objectif soit de déterminer des objets soit d'en comparer les états. Il faut toutefois noter que le concept d'agencement doit être conçu dans le même esprit que celui de Profil. En effet, la constitution d'une hiérarchie d'agencements pose le même type de problèmes: comment limiter l'extension d'une hiérarchie en dégageant de l'extension du concept des invariants de nature à autoriser une description générique de sa constitution. Ceci reste donc un axe de travail à développer.

### 4.2.1.iv) Les attributs

Les attributs sont représentés par une hiérarchie de concepts proches (du point de vue des propriétés contenues) de ceux des entités mais qui diffèrent sémantiquement. Un attribut va, comme l'entité, contenir par exemple des éléments de définition morphologique, des méthodes de représentation, des mécanismes de mesurage, etc.. Mais dans le modèle que nous proposons l'attribut n'est manipulable que par l'intermédiaire de l'entité qui le contient. Perte de lisibilité et de performance (en gestion de données), ce choix s'explique par le souci de disposer d'un interface simple au modèle, ne multipliant pas les types de concepts manipulables. Ce choix n'est ni définitif ni totalement assumé, il introduit de fait des limitations importantes en terme de gestion de données sur lesquelles nous nous pencherons ultérieurement. Concrètement, les attributs servent aujourd'hui d'outils de définition morphologique pour les entités architecturales. Si la géométrie globale de l'arc est définie dans les classes pertinentes de la hiérarchie des entités, leurs profils sont eux manipulés par une classe pertinente de la hiérarchie des attributs. Nous allons maintenant tenter de donner une définition des attributs puis l'illustrerons par quelques cas concrets.

#### *4.2.1.iv.1. Identification*

Une entité architecturale est le plus souvent un assemblage de formes ou plus généralement d'éléments de modénature. On l'a vu, elle est intégrée à la hiérarchie d'entité non sur la base de sa forme mais sur le croisement de trois indices: forme, fonction, autonomie. Pourtant, il convient de pouvoir manipuler chacune des formes constituant l'entité, et donc d'isoler des concepts permettant de le faire. L'attribut est le formalisme choisi pour répondre à cette attente. Un attribut devrait en théorie porter des informations qualitatives, une inscription stylistique par exemple, et des informations quantitatives. Dans l'état actuel de notre travail, seul le deuxième groupe d'information a été traité. Nous nous ramenons donc à un problème de vocabulaire de formes, formes pour la définition desquelles nous proposons la formulation suivante :

1. L'attribut caractérise une forme qui n'est employée que dans la détermination morphologique d'une entité (pas d'existence autonome dans l'édifice).
2. Un même attribut est le plus souvent employé dans la détermination morphologique de diverses entités.
3. La morphologie d'un attribut est complètement décrite par le renseignement de ses propriétés dimensionnelles propres.

Cette définition restrictive exclut par exemple les éléments de statuaire, et ce, à dessein. En effet, les attributs dont nous avons traité jusqu'à présent sont des attributs morphologiques, c'est à dire ayant trait à la forme et à la structure des entités, et non des qualifieurs pour son décor. Nous introduisons donc dès à présent la distinction sans doute la plus difficile à établir dans le domaine d'application dont nous parlons: celle qui sépare la morphologie structurelle des élements de décor rapportés. En effet, dès l'époque romaine sont apparus des modes de construction dans lesquels des structures rapportées imitaient des éléments de morphologie structurelle existants. Nous nous bornerons ici à décrire un jeu de concepts représentant des éléments déterminant la morphologie des entités architecturales sans préjuger de leur mode de construction.

Par ailleurs, nous ne pourrons gérer de façon générique les attributs morphologiques sans en déterminer un jeu de morphèmes: tore, scotie, quart de rond, etc… Nous proposons donc un modèle d'attribut raffiné en trois catégories:

- attributs morphologiques,
- primitives morphologiques,
- qualifieurs,

dont seules les deux premières ont pu jusqu'à ce jour être expérimentées.

La classe Attribut, tête de la hiérarchie, regroupe des propriétés génériques, notamment les trois dimensions d'une boîte englobante, et se distingue de la classe AttributDimensionnel

(attributs monodimensionnels), utilisée de façon transitoire dans l'implémentation Java pour gérer les propriétés dimensionnelles des entités.
Nous avons en **annexe 36** choisi d'illustrer le processus d'identification des attributs en présentant trois classes dérivant de la classe Attribut, classes qui nous semblent représentatives des différents problèmes traités par les attributs. Le premier attribut présenté, la scotie, est une moulure utilisée largement aux périodes antique, renaissance et néo-classique. Sa particularité tient à un tracé dont il nous semblait important de rappeler le mode de définition dans la pratique: une succession d'arcs dont les centres se décalent. Le second attribut présenté, la moulure, propose un formalisme générique de moulure permettant de gérer le troisième attribut présenté, le profil, comme une succession d'objets de type "moulure" dont les tracés deviennent dès lors indépendants les uns des autres. Il faut rappeler qu'ici notre objectif est double :

- D'une part, intégrer ces notions au modèle.
- D'autre part, faciliter leur représentation tridimensionnelle en définissant un formalisme de points de contrôle / liste de segments intermédiaire. Ce formalisme permet de prendre en compte les contraintes liées à la visualisation (niveau de détail en particulier) mais il permet également de représenter le mode de description théorique des moulures tel que la bibliographie nous le transmet.

*4.2.1.iv.2. Organisation des concepts*

L'organisation hiérarchique des attributs suit une logique directement issue de l'application du formalisme objet: spécialisation par ajout de propriétés. En fait, cette logique est avant tout une logique d'implémentation, nous n'avons pas ici identifié de nécessité plus impérieuse. On remarquera par ailleurs dans le schéma placé en **annexe 37** que le nombre de concepts en jeu est relativement faible. Notre objectif est en effet d'en limiter l'extension en proposant une détermination générique des attributs morphologiques des entités. De plus, ainsi que nous l'avons mentionné plus haut, les attributs non morphologiques n'ont pas encore été traités. On notera que les attributs morphologiques ne sont pas nécessairement formés à partir de collections de primitives morphologiques. Le classe piedProfil par exemple représente des objets dimensionnellement déterminés par leurs seules propriétés. Ces propriétés correspondent en effet à des éléments de modénature dont nous ne trouvons pas dans les sources documentaires de réutilisations.

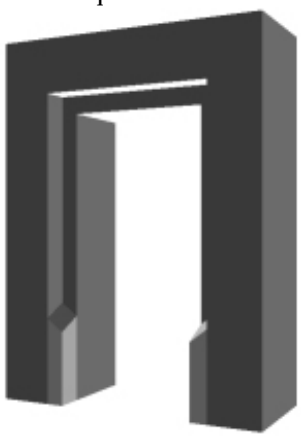


*Figure 47 : Visualisation de l'attribut PiedProfil (VRML, im auteur)*

**4.2.1.v) La norme**

En introduisant l'idée de norme nous faisons référence à un sujet qui reste encore dans les développements actuels de notre projet essentiellement prospectif. En effet, si cette idée a été reprise dans le cadre de nos terrains d'expérimentation sur l'architecture antique, elle n'a pas été étendue au-delà. Pourtant, il nous semble important de citer cette question car elle fait référence à un pan des connaissances manipulées par les architectes, en particulier par les bâtisseurs de ces édifices que nous appelons aujourd'hui le patrimoine architectural construit.
Cette idée de norme recouvre aujourd'hui celle des processus d'industrialisation des bâtiments, et intervient dans la construction de ce que nous appelons dans notre modèle les réseaux. Elle détermine dans ce cadre un jeu de relations entre objets. Mais elle intervient également dans le dimensionnement des entités architecturales. Il n'entre pas dans le contexte de ce travail de discuter du contenu et de l'évolution du concept de norme en architecture. Il nous semble nécessaire cependant d'en fixer quelques grandes lignes et d'insister sur la nécessité d'inscrire ce point comme une des limitations importantes de notre modèle dans sa forme actuelle.
Pour tenter d'illustrer l'importance donnée à ce sujet dans le domaine d'application de notre modèle, nous citerons en introduction à cette section le propos d'Eugène Viollet Le Duc dans son neuvième entretien [Viollet Le Duc, 1863/1977, p395]:
"*Il est utile de connaître les proportions que les anciens ont donné aux ordres, mais il l'est davantage probablement de chercher quels ont été les principes générateurs de proportions*

*dans les édifices antiques, du moyen âge et même de la Renaissance… Ce serait faire illusion si l'on croyait que les proportions en architecture sont le résultat d'un instinct. Il y a des règles absolues, il y a des principes géométriques, et si ces principes sont d'accord avec le sentiment des yeux, c'est que la vue est un sens, comme l'ouïe, qui ne peut se faire à une dissonance sans être choquée… Il serait étrange, on l'avouera, que l'architecture, fille de la géométrie, ne pût démontrer géométriquement pourquoi il se fait que l'œil est tourmenté par un défaut dans les proportions d'un édifice…*"

Il nous appartiendra de rechercher un formalisme apte à représenter le concept de norme sous ses différentes formes :

- La proportion entre les différents objets physiques en jeu dans une composition et son éventuel rapport à une dimension de référence.
- Les dimensions relatives des attributs morphologiques d'une entité entre eux et leur rapport aux points précédents.
- Les tracés régulateurs explicitant le positionnement relatif des objets physiques et leur dimensionnement.
- Le rôle des contraintes techniques ou administratives dans la formulation de solutions architecturales.
- Enfin, les dispositifs de corrections visuelles particulièrement important aux époques dont nous parlons.

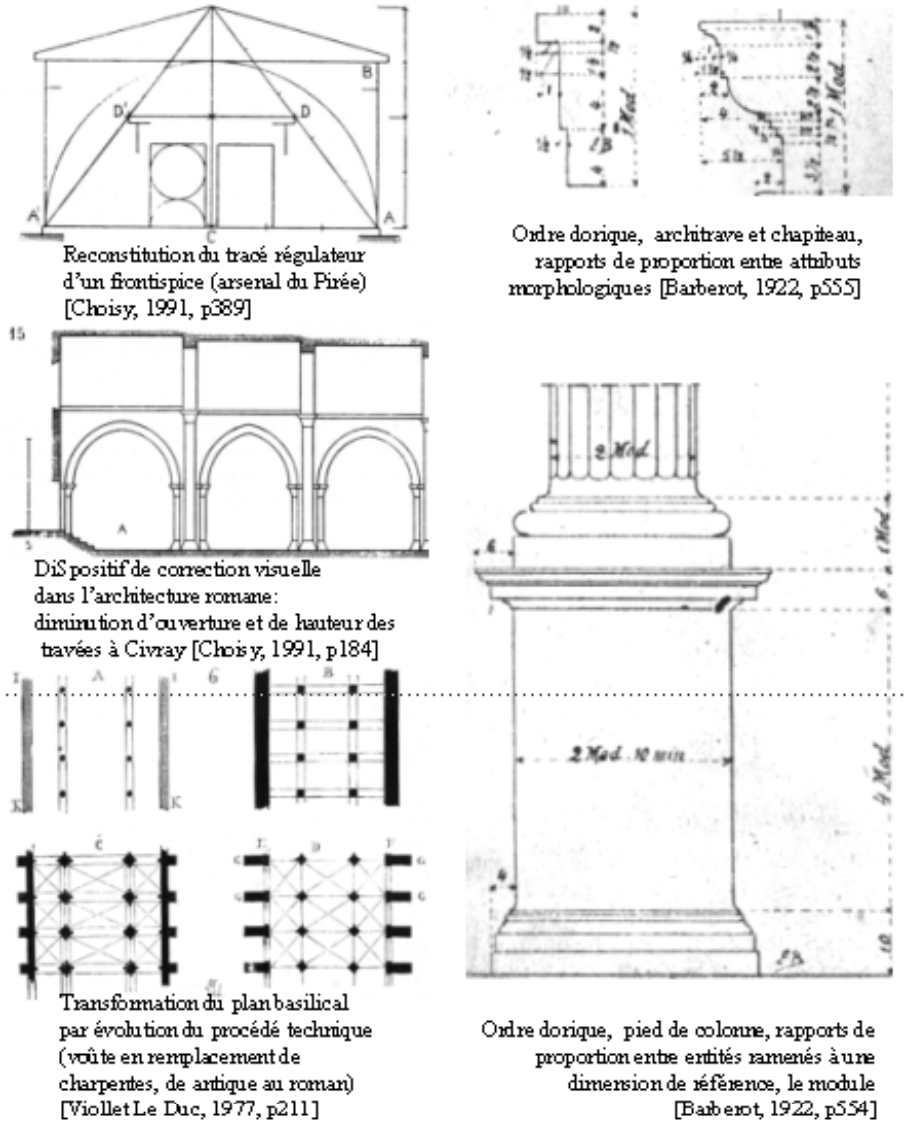



*Figure 48 : tracés régulateurs et principes de compositions à différentes périodes*

Pour l'heure, nous nous proposons de présenter sous le thème de proportion la démarche que nous avons choisie dans le cas de l'architecture antique et d'en citer les perspectives dans le cas d'un corpus architectural moyenâgeux. Nous verrons ensuite sous le thème de rythme les répercussions de cette notion sur la constitution de notre modèle.

### *4.2.1.v.1. La proportion*

Nous ne souhaitons pas ici donner du terme proportion une définition argumentée ou d'en mener une étude approfondie. Nous en donnons seulement notre interprétation afin de rendre plus clair les points soulevés dans les sous-chapitres suivants. Nous appellerons donc proportion le rapport de dépendance des grandeurs existant entre divers objets architecturaux, c'est à dire entre entités architecturales ou entre attributs morphologiques et entités

architecturales[118]. Nous nous contentons de constater l'existence de rapports arithmétiques entre propriétés permettant d'en évaluer une par rapport à une autre. Nous avons en **annexe 38** proposé deux illustrations qui indiqueront peut-être mieux l'usage possible de cette notion de proportion dans le cadre de notre effort de modélisation[119].

*4.2.1.v.2. Le rythme*

Situons pour commencer le sens que nous donnons à ce terme. Le rythme désignera pour nous la règle d'alternance d'éléments de morphologie, qu'il s'agisse d'entités architecturales ou d'attributs morphologiques. Dans l'exemple ci-contre (ordonnance de baies de la façade du palazzo Pitti, Florence), un rythme simple règle l'alternance de baies et de trumeaux de l'étage mais aussi la superposition des baies. Jean-Marie Pérouse De Montclos décrit des ordonnances de baies de rythme binaire ou ternaire [Pérouse De Montclos, 1988, p93]. Nous ne détaillons pas plus avant ce point et renvoyons à la littérature si nécessaire. En effet, nous n'introduisons cette notion ici que pour en expliciter les répercussions sur la constitution de notre modèle. Nous n'avons pas à ce jour formalisé de mécanisme autorisant l'instanciation d'entités architecturales liées par une relation de rythme. Nous avons par contre mis en œuvre un formalisme de manipulation des profils qui s'appuie sur cette notion de rythme puisqu'il permet de définir d'un côté les éléments en alternance (type de moulure) et un pas variable (points de contrôle). Cette approche peut être la base d'une solution à l'échelle des entités. Toutefois, il faut rappeler que dans leur cas l'objectif est double[120]:

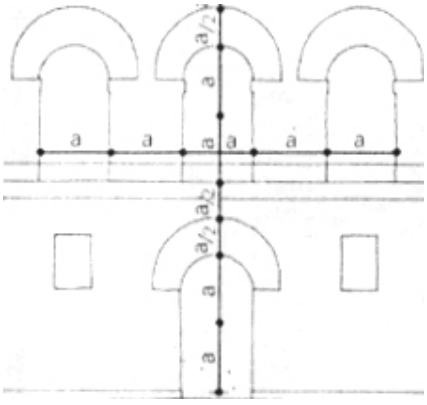



*Figure 49 : Ordonnance de la façade du Palazzo Pitti, Florence*

- Positionner des entités dans l'espace en fonction d'une règle établie.
- Manipuler cette règle comme un concept à part entière du modèle architectural.

## 4.2.2. Evaluation du modèle

### 4.2.2.i) Limitations du modèle

La première limitation au modèle que nous proposons, et celle qui nous semble la plus significative, est une conséquence directe du point de vue initial qui a présidé à son élaboration. En effet, nous nous étions fixé comme objectif de travailler sur le relevé des objets physiques constituant l'édifice. Le modèle que nous présentons ne gère donc que des concepts relatifs à l'édifice en temps qu'artefact. Il ne fait en rien référence à l'usage fait de ses espaces. Ainsi, la gestion des espaces clos, fermés par des objets physiques (entités) mis en relations (réseau), n'est pas aujourd'hui prise en considération dans notre modèle.
Bien que cette limitation soit avant tout à replacer dans le contexte de la genèse de ce projet, il faut aussi considérer qu'il s'agit là d'un problème de continuité dans le temps : un espace clos peut changer radicalement en terme de limites comme de destination (voire devenir non clos), et ce indépendamment des éléments qui le ferment. Il ne peut exister qu'un lien du type relation éphémère entre entités et espaces clos. Nous renvoyons notamment à [Donath et al, 1997a] [Donath et al, 1997b] qui distinguent deux grandes hiérarchies d'objets: les espaces ou vides (correspondant aux subdivisions de l'édifice) et les objets bâtis. La problématique initiale de notre projet était centrée sur le relevé quantitatif de l'édifice, et donc le souci de prendre en compte la nature les lieux n'apparaissait pas justifié. Dans l'optique d'un système plus général de relevé informationnel, la notion d'espace doit trouver une place au sein de notre dispositif d'étude. Cela fait partie des directions de recherche que nous souhaitons

[118] Nous ne considérons pas le rapport de ces grandeurs à une dimension de référence, comme le module dans l'architecture antique, comme nécessaire à notre interprétation.
[119] L'architecture modulaire de l'édifice antique, les convenances constructives et tracés régulateurs dans l'édifice moyenâgeux.
[120] Dans le cas des profils, l'instance porte elle-même les deux informations puisque le rythme est caractérisé par un jeu de propriétés de la classe Profil Mouluré. Pour les entités, l'information relative au positionnement est détenue par un réseau. Nous devrons donc exprimer la notion de rythme indépendamment de l'entité, par le biais des concepts gérant les relations entre entités.

explorer. Mais au delà de ce point, le modèle architectural que nous proposons souffre de plusieurs limitations, dont la plupart ont déjà évoquées plus haut dans cette section :

- La typologie des assemblages et la répercussion de ceux-ci sur la morphologie des entités n'est pas explicitée.
- Les relations d'une entité à un groupe d'entités ne sont pas implémentées.
- Les relations traitent seulement de la localisation des entités.
- Les réseaux d'agencements, ou assemblages prédéfinies d'entités et de relations, ou les réseaux de concordance visant à établir des comparaisons entre propriétés qualitatives restent à développer.
- Les éléments de statuaire ou du décor peint ne sont pas traités.
- Les attributs dits *qualifieurs* ne sont qu'un principe non implémenté.
- Les attributs ne sont manipulables que par l'intermédiaire de l'entité.
- Les modes de construction sont représentés comme propriété des entités et ne figurent pas comme concept à part entière dans notre modèle.
- La notion de norme, et plus généralement de méthodologie de tracé et de proportionnement des assemblages, est présente de façon ponctuelle (module, profils).

Il doit donc être souligné que de nombreux points restent à développer avant de pouvoir évaluer les manques du modèle proposé de façon exhaustive. Reste également en terme de conclusion à cette présentation du modèle architectural à en rappeler l'objectif: servir le raisonnement du conservateur. Les développements en cours ou en perspective doivent donc être ramenés à cette ambition, et les limitations du modèle dont nous faisons état doivent se comprendre dans ce sens. Nous présentons dans les sections suivantes les outils développés pour exploiter ce modèle et ainsi en pourrons-nous mesurer l'aptitude à servir l'étude de l'édifice patrimonial et plus généralement à servir d'instrument de connaissance sur le bâti.

#### 4.2.2.ii) De l'approche méthodologique à l'implémentation

Nous avons au début de ce document présenté plusieurs terrains d'expérimentation sir lesquels nous avons souhaité évaluer notre modèle. Nous l'avons fait au travers de six outils s'appuyant sur ce modèle, outils dont nous présenterons les implémentations dans la section suivante. Mais quel est le rapport à établir entre notre approche méthodologique visant à définir une *logique de constitution* du modèle, et l'ensemble d'implémentations que nous allons détailler ?

Pour fixer un ordre d'idées disons que dans notre implémentation la plus récente[121] nous avons isolé 8 catégories de classes, trois catégories représentant les classes chargées de manipuler les concepts architecturaux, les interfaces et et la géométrie, cinq catégories représentant des modules à forte réutilisabilité en charge d'exploiter les CGI[122]. Dans les trois premières catégories on retrouve quatre hiérarchies de concepts architecturaux, une hiérachie de concept gérant l'interfaçage des classes architecturales pour les formats supportés sur le réseau Internet, et enfin une hiérarchie d'objets représentant des outils géométriques (points, vecteur, …). Dans les cinq autres catégories on retrouve des outils de réalisation d'interfaces, à savoir les hiérarchies d'objets chargés du contrôle d'accès au serveur Web, du formatage HTML de listes, etc…Nous avons instancié à partir du jeu de classes architecturales 1682 objets, figurés à l'intérieur de 27 scènes tridimensionnelles VRML.Nous avons établi pour ce jeu d'instances un jeu de 203 instances prototypes[123] réparties dans les scènes, portant le nombre total d'objets dont les états sont persistants à 1885. Nous avons par ailleurs archivé 57 scènes tests ne comportant qu'un petit nombre d'instances. L'exécution des traitements dans l'application VALIDEUR est contrôlé par en tout et pout tout deux fichiers exécutables PERL. La base de données SOL contient environ 500 entrées documentées ou illustrées par 957 fichiers HTML (pour 580 fichiers image), son interfaçage s'appuie sur 27 scripts PERL..

[121] Implémentation PERL que j'ai développé en totalité et qui est mise en œuvre sur les outils SOL, VALIDEUR et DIVA.
[122] Nous renvoyons en **annexe 49** une évaluation quantitative des hiérarchies correspondantes.
[123] L'instance prototype est un objet auquel pour une scène donnée est affecté un jeu de valeurs d'attributs prototypique.

## 4.3. EXPLOITATION DU MODELE : INTERFACES

Nous allons ici définir les éléments relatifs à quatre dimensions du modèle, et présenterons en parallèle les intrants et les extrants relatifs à ces quatre dimensions. En effet, le modèle va servir à nourrir un raisonnement centré sur l'édifice. Ce raisonnement nécessite d'une part la prise en compte de paramètres (objectifs ou liés à des hypothèses) fixant les propriétés des éléments du modèle et d'autre part la mise à disposition de résultats associés à ces paramètres, résultats dépendant de l'activation de méthodes propres aux objets du modèle. Le mot Interface recouvrera donc ici l'idée de propriétés et méthodes incluses dans la définition des objets du modèle non pour en justifier les lignes de division (la classification) mais pour exploiter des points de vue sur celui-ci, c'est à dire pour fournir propriétés et méthode pertinentes dans le seul cadre de ces points de vue. Autrement dit, Les aspects du modèle présentés ici n'interviennent pas dans l'organisation de celui-ci mais permettent de l'exploiter Notre objectif dans cette section sera donc d'une part d'établir quels sont les éléments déterminants relatifs à chacun de ses points de vue et comment nous les intégrons au modèle, puis de préciser à la fois quels traitements nous autorisons sur ces éléments et quelles interfaces leur correspondent.
Nous ne présentons pas les détails des implémentations (cela sera fait pour certains à plus loin) mais les problèmes auxquels font référence ces différents aspects. Nous évoquons ici des aspects applicatifs du modèle, à savoir les questions de la mesure, des interfaces Web, de la documentation et de la représentation. Nous le faisons essentiellement sous l'angle des services que le modèle est censé rendre. Un aperçu des principes de modélisation des connaissances relatives à ces aspects est présenté dans la section 4.3.4.

### 4.3.1. La mesure

La mesure a été le point de vue de départ de notre projet. Plusieurs publications rendent compte de cet aspect[124], et notamment la thèse de P.Drap [Drap, 1997]. Nous ne reviendrons donc pas en détail sur ce point mais présenterons notre démarche et ses répercussions sur la constitution du modèle.

#### 4.3.1.i) Un relevé renseigné

Le relevé architectural tel qu'il se pratique couramment consiste à fixer un ensembles de grandeurs relatives à l'édifice observé et à produire à partir de ces données quantitatives un ensemble de représentations dessinées. Il est clair que notre démarche s'inscrit en rupture avec cette pratique puisque nous mettons en œuvre une technique de relevé s'appuyant sur une connaissance a priori du corpus de l'édifice à mesurer. Autrement dit, nous ne relevons que des instances d'un modèle formalisé a priori. Cette mise au point nous semble nécessaire pour clarifier la discussion à venir : notre objectif n'est pas d'utiliser telle ou telle grandeur mesurée comme intrant à une démarche de représentation de l'édifice, autrement dit à une démarche visant à construire une *maquette* de l'édifice.
Commençons par préciser le problème tel qu'il se pose pour nous : mesurer une instance du modèle. Dans notre approche du relevé, mesurer une instance du modèle architectural est un processus dont l'objectif est le renseignement des propriétés morphologiques d'une entité architecturale, de sa position et de son orientation. L'entité est dans ce cadre cet objet élémentaire prédéfini sur lequel portera le relevé. La diversité des objets architecturaux et l'importance de leur modénature nous ont conduit à rechercher sur ces entités des particularités morphologiques stables sur lesquelles portera la mesure. Une hiérarchie d'objets géométriques capables de rendre compte de la morphologie des entités architecturales est destinée à faire le lien entre la mesure et la forme.

[124] Voir notamment [Florenzano et al, 1996a], [Florenzano et al, 1996b], [Florenzano et al, 1997a], [Florenzano et al, 1997b], [Drap et al, 1999a]

Les entités étant définies, le relevé de leur morphologie revient en fait à trouver le moyen de passer du nuage de points relevés à leur surface à la valeur de leurs attributs morphologiques. Cette étape utilise des primitives géométriques simples décrivant des parties significatives de la surface des entités. Ces primitives géométriques sont le lien entre le résultat de la mesure (un nuage de points étiquettés) et les attributs morphologiques des entités.

#### 4.3.1.ii) Le processus : principe général

La mesure de la forme des entités est gérée par des objets capables de recalculer une géométrie parfaite en s'appuyant sur une série d'observations. La photogrammétrie ne fournit qu'un nuage de points mesurés sur la surface de l'objet observé et chaque point est entaché d'une erreur aléatoire centrée (car supposée exempte de systématisme). La forme des entités sera évaluée au travers de primitives géométriques simples. Elles seront calculées par les objets des classes EGO (Etre Géométrique Optimal) à partir des nuages de points mesurés. Les EGO sont en fait des modèles fonctionnels décrivant les relations liant le modèle théorique visé (la primitive géométrique) et les quantités observées (les points mesurés en photogrammétrie)[125].

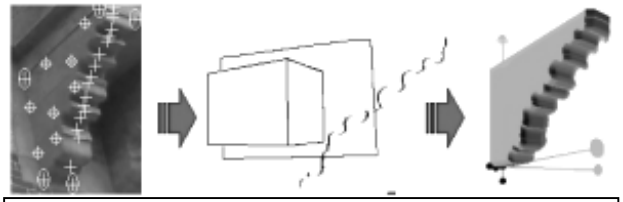


*Figure 50 : Trois étapes du relevé d'une console: points étiquettés, boîte englobante plan de projection et profil, console (Expèrience de l'ancien hôtel de ville de Cracovie, voir [Drap et al, 1999a] )*

Les EGO se chargent donc de l'interface entre la mesure et la géométrie. La méthode consiste à minimiser la somme des carrés des distances dans l'espace entre les observés (les points) et le modèle théorique (la primitive géométrique). En d'autres termes la primitive géométrique calculée par l'EGO passe au mieux (au sens des moindres carrés) par le nuage de points observé. On le voit encore une fois le calcul s'appuie sur la connaissance qu'a l'opérateur du domaine lors de la phase de mesurage. En effet les points ont été non seulement affectés à des entités architecturales mais aussi à des primitives géométriques. Il appartient à l'opérateur de vérifier a posteriori la pertinence du modèle géométrique proposé lors de la phase de mesurage. Une étude attentive des résidus (écart entre les points mesurés et la primitive calculée) et de la répartition des points sur la primitive doit valider le modèle.

Un script "*LaDemarche*" résultant du processus de mesurage est généré, script dans lequel les entités mesurées sont décrites et qui peut être amendé pour ajouter par exemples de nouvelles instances issues d'une analyse théorique de l'édifice (Par exemple, pour ajouter des entités clones de l'entité mesurée dans une composition répétitive comme un

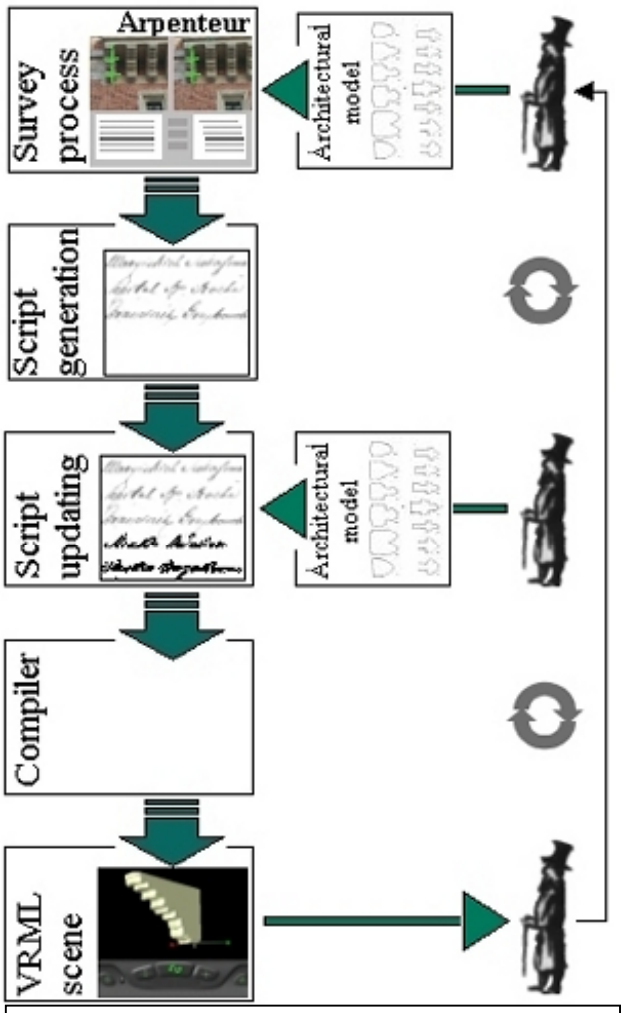



*Figure 51.Phases de mesure et de mise à jour des objets architecturaux (Expèrience de l'ancien hôtel de ville de Cracovie, voir [Drap et al, 1999a] )*

[125] Ainsi, pour chaque primitive géométrique supportée :
-il existe, dans la classification décrivant les primitives, une classe apte à effectuer toutes les opérations qui s'y rattachent (affichage, calcul dans l'espace ...),
-par ailleurs, une classe homologue dans l'arbre des EGO décrit son modèle fonctionnel et assure le lien entre la mesure et la primitive géométrique. Chaque classe de l'arbre des EGO a pour membre une primitive homologue de l'arbre des primitives géométriques.

encorbellement partiellement détruit ou enfoui). Les éléments relevés étant instances d'un modèle architectural théorique, leur représentation correspond à l'activation des méthodes appropriées des entités, et correspond donc à l'expression d'un modèle théorique. Par conséquent, les entités mesurées comme les entités instanciées pour compléter la représentation du réseau sont représentées dans un même espace, celui caractérisé par les points de contrôle du relevé photogrammétrique[126].

#### 4.3.1.iii) Le processus : répercussion sur la constitution du modèle

La description de l'entité comprend donc un ensemble de mécanismes de calcul géométrique susceptibles de rentrer en jeu dans le processus de mesurage. L'entité est alors positionnée, dimensionnée et orientée par ses primitives géométriques.

Chaque entité architecturale dispose en fait d'un jeu de primitives géométriques redondant correspondant à sa spécificité morphologique. La redondance doit permettre à l'opérateur d'adapter sa saisie en fonction de l'état de l'objet observé. Chaque primitive dispose d'un mécanisme d'auto-évaluation, membre de la hiérarchie des EGO, chargé de retrouver dans un ensemble de points donné les propriétés de la primitive géométrique.

Chaque point relevé à la surface d'un objet doit être étiquetté comme appartenant à une instance donnée du modèle théorique. Un relevé architectural mené selon le processus décrit ci-dessus commence par l'indication d'un choix d'entité à mesurer (en fait, choix de son type). L'utilisateur choisit dès lors dans la définition de l'entité quelle particularité morphologique il souhaite relever (et donc la primitive géométrique correspondante).

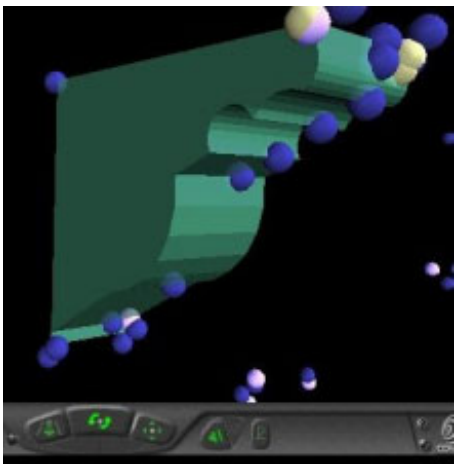


*Figure 52 : Points mesurés à la surface d'une Entité (Console) visualisés en VRML*

Le concept d'entité architecturale réunit dans une même classe les données architecturales décrivant l'entité, les mécanismes d'interfaçage avec la mesure et les méthodes de représentation. Ceci nous a conduit vers un modèle générique d'entité, construit autour d'objets hétérogènes et de méthodes de communication vers d'autres outils.

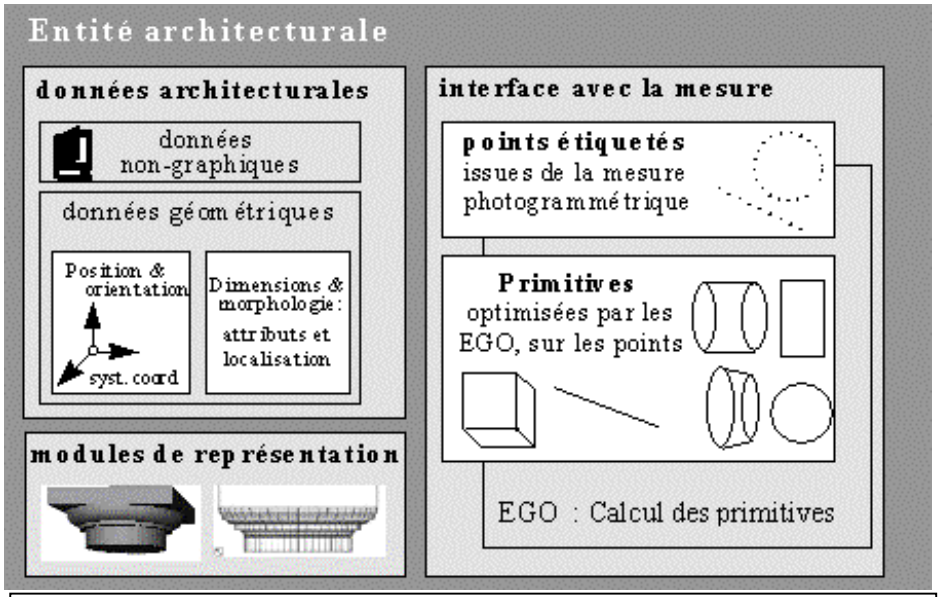



*Figure 53 : Le modèle générique d'entité*

[126] Une intervention sur le jeu d'instances mesurées est possible dès qu'un script "*LaDemarche*" est généré : de nouvelles entités peuvent être ajoutées et visualisées, et le résultat global évalué. Un nouveau cycle de mesurage peut alors être lancé dans lequel les objets ajoutés comme les objets déjà mesurés peuvent être relevés.

Ces objets structurant l'entité peuvent être classés en trois grandes familles :
- Objets décrivant les propriétés non graphiques des entités.
- Objets appartenant à l'univers de la géométrie ( points, vecteurs, matrices, référentiels).
- Ceux dédiés à l'interfaçage avec la mesure (EGO pour Etres Géométriques Optimaux).

Le concept d'entité, défini ainsi, est bien un élément d'architecture qui dépasse le cadre pur d'une description de l'édifice en entités élémentaires et prend en compte la problématique de la mesure. Dès lors l'objet architectural véhicule des informations sur sa nature et sa typo-morphologie, sa position et son orientation dans le référentiel du bâtiment, ses dimensions, et la méthodologie du mesurage (fiabilité des grandeurs caractéristiques, incertitude de mesure).

### 4.3.2. L'échange de données sur Internet: Interfaces Web

Rappelons ici d'abord le problème posé: il s'agit pour nous de donner accès à l'exploitation du modèle au travers de la plateforme Internet. En parlant d'exploitation du modèle, nous excluons de fait l'idée de redéfinition ou d'amendement de celui-ci. Nous verrons plus loin qu'une tentative allant dans ce sens a été menée à bien. Mais restons en pour l'instant à l'idée d'un accès au modèle (propriétés et méthodes associés aux concepts) par le biais d'interfaces web.

Il faut ici distinguer trois développements parallèles :
- Un interface dédié au relevé photogrammétrique, autonome, appelé ARPENTEUR, développé exclusivement par P.Drap, et qui utilise un certain nombre de classes architecturales (entités ou attributs) telles que définies à la section précédente. Nous ne l'avons cité et ne le citerons qu'en référence aux expérimentations de cet interface menées conjointement et centrée sur l'exploitation du modèle architectural.
- Un interface de consultation et d'interrogation d'une base de données documentaire utilisant la représentation tridimensionnelle (maquettes numériques VRML) du modèle pour lier tel ou tel élément de corpus à un ensemble de données. Nous revenons sur ce développement dans la section documentation suivant immédiatement celle-ci.
- Enfin, un interface d'instanciation et de manipulation d'instances du modèle au travers de formulaires basiques CGI, autorisant à la fois.
  - L'appel aux méthodes de représentation des instances.
  - L'établissement d'un lien vers la base documentaire.
  - Le travail collaboratif sur une scène.

Nous allons maintenant discuter de ce troisième développement. Nous verrons dans la section implémentations que d'autres interfaces ont été développés dans le cadre d'applications expérimentales comme le Classeur. Nous les présenterons en temps utile, mais ne souhaitions ici citer que les grandes familles d'Interfaces, dont les architectures se veulent plus indépendantes des applications.

L'expression d'un jeu d'instances du modèle sur la plateforme Internet est assurée par une hiérarchie d'objets outils gérant pour chaque instance un compte rendu de son état en HTML. L'intervention sur un jeu d'instances est autorisée par le biais de formulaires CGI gérant l'accès à leur état et à leurs méthodes. Concrètement, un réseau doit être instancié en préalable, il se charge d'appeler en fonction de sélections utilisateur les méthodes appropriées des entités, qui font de même avec leurs attributs morphologiques. Une question importante était ici la mise en œuvre d'un mécanisme séparant clairement données issues de la consultation des états des instances et données relatives à la mise en forme grahique de ces données dans une page lue par un navigateur classique. Notre réponse est explicitée ci-dessous.

#### 4.3.2.i) Interfaces Web: propriétés communes

Un formalisme générique d'interface Web a été développé pour regrouper les propriétés communes des interfaces dédiés à l'instanciation ou à la modification de réseaux ou d'entités architecturales. La classe outil implémentée permet essentiellement, à partir d'un objet requérant et de données génériques relatives à la mise en page, de fournir un résultat en

HTML. On notera que le recours à un tel formalisme d'interface n'est pas requis dans le cas de scènes VRML : celles ci sont détectées par un plug-in qui se charge de les afficher telles quelles (pas de balisage), l'expression VRML d'un jeu d'instances ne pose pas un problème générique d'Interfaçage mais seulement un problème d'affichage du ressort de l'application concernée.

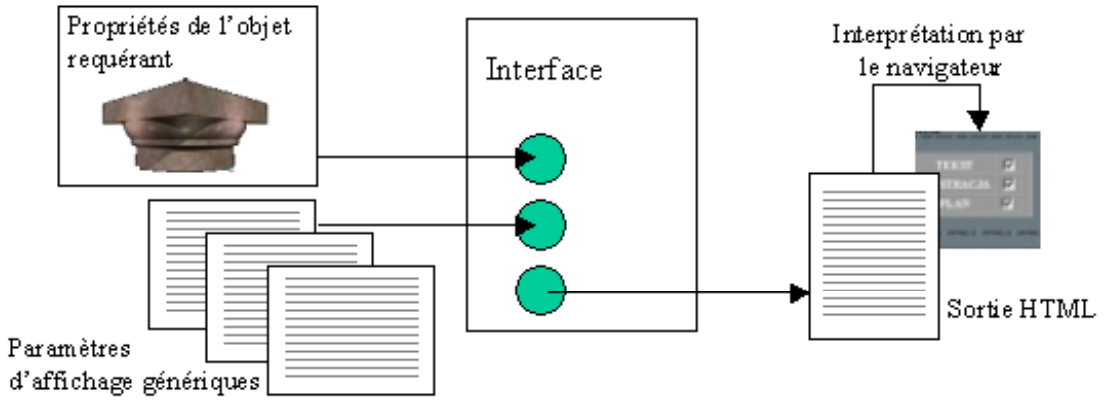



*Figure 54 : Formalisme d'interface Web, données génériques et spécifiques côté serveur, sortie HTML interprétée côté client.*

#### 4.3.2.ii) Interfaces Web : spécialisations

Nous distinguons un formalisme d'affichage des entités et un formalisme d'affichage des réseaux. Dans le premier cas, le service attendu est l'accès à l'état de l'entité architecturale. Cela suppose d'autoriser l'accès à ses propriétés comme à celles de ses attributs morphologiques. Un formalisme de génération de formulaires CGI a été développé pour décharger l'utilisateur de l'écriture des balises adéquates et de la mise en page. La classe InterfaceEntité gère donc des objets capables à partir des intrants de la classe parente de générer un résultat HTML intégrant le formulaire pertinent pour chaque entité architecturale considérée. C'est à l'application au sein de laquelle ces formalismes d'Interfaces et de Formulaires génériques sont utilisés de déterminer l'action de mise à jour précise à exécuter lors de la validation du formulaire. Le formalisme d'interfaçage des réseaux a des objectifs proches du précédent[127].. Toutefois, il s'en distingue par la nécessité d'autoriser plusieurs actions de mise à jour complémentaires :

- Instanciation d'entités.
- Suppression d'entités.
- Génération de rapports (y compris VRML).
- Appel à l'interface des instances d'entités sélectionnées.
- Exécution de méthodes d'assemblage définies dans les relations.


*Figure 55 : Une réutilisation du formalisme de gestion des interfaces Web, le cas de la base documentaire UIA*

### 4.3.3. La documentation

Dans le domaine d'application dont nous traitons, la documentation de l'édifice joue un rôle fondamental, que ce soit dans le cadre du relevé, dans celui de la formulation d'hypothèses de restitution ou dans celui de la gestion de données localisées. A.Bilgin, dans la présentation de son travail sur l'application d'un SIG (Système d'Informations Géographique) dans le champ de l'architecture patrimoniale [Bilgin, 1997, p216], rappelle que "*la première étape dans*

[127] Nous reviendrons sur un exemple d'utilisation de ces formalismes dans le cadre de notre projet avec la présentation de l'outil VALIDEUR pour lequel ils ont été initialement développés On notera que certaines de ces classes outils (ainsi que celles chargées de l'écriture des balises de formulaires) sont également utilisées dans le cadre de développements Web très différents au sein du site marseillais de l'UMR MAP, le GAMSAU. C'est notamment le cas dans le cadre du projet "Patrimoine Architectural du XXème siècle" (centré gestion de données sur le Web) mené en collaboration avec l'UIA (Union Internationale des Architectes), projet pour lequel on peut se reporter à l'adresse Web suivante : http://www.archi.fr/UIA.

*l'étude et la compréhension des évolutions urbaines est la compilation de diverses données, qu'elles proviennent de relevés, d'études archéologiques ou de sources documentaires recueillies dans les archives ou les bibliothèques*". L'auteur ajoute ces quatre points :

- les interventions sur le tissu urbain ont un caractère pluridisciplinaire.
- Les informations relatives au tissu urbain doivent donc être facilement partagées et disponibles.
- La mise à jour des données existantes doit être autorisée.
- Seule une classification adéquate permet de rendre lisible les différentes sources.

C'est dans cet esprit que nous avons abordé le problème de la gestion des données documentaires relatives aux édifices nous servant de terrains d'expérimentation. Nous pouvons néanmoins ajouter un autre point, plus prospectif : les données documentaires doivent également être rapportées à la constitution du modèle, autrement dit les concepts que celui-ci véhicule doivent être ancrés dans une documentation en situant les sources, en explicitant la constitution.

### 4.3.3.i) Constitution de la base documentaire

Dans le cadre du travail mené en collaboration avec l'institut HAiKZ[128] sur les édifices de la place centrale de Cracovie (Rynek Główny), expérience relatée en détail plus loin dans ce document, la constitution d'une base documentaire nous est apparue dès l'origine comme un passage obligé. Nous nous étions pourtant fixé un certain nombre de contraintes dans la constitution de la base et dans l'élaboration de son schéma:

- L'outil à développer devrait autoriser la mise à jour à distance de la base, un des objectifs étant de capitaliser les expériences des doctorants déjà engagés dans l'étude des édifices du Rynek Główny ou devant s'y engager. L'outil devait donc au-delà de la gestion de données documentaires s'imposer comme un outil d'information collaboratif pour les chercheurs, enseignants et doctorants concernés.
- L'outil serait centré sur la plateforme Internet et utiliserait une architecture logicielle entièrement portable, gratuite, et maîtrisée par les deux parties (HTML / VRML / Perl).
- Les documents référencés devraient faire l'objet d'une lecture critique permettant de dégager au-delà des classiques critères descriptifs des bases de données documentaires des critères propres à notre démarche de modélisation ou à la discipline architecture patrimoniale. Ces critères, (maladroitement) nommés points de vue, devraient ainsi autoriser la prise en compte de notions propres à l'analyse du conservateur (évolution historique, références d'un auteur à un auteur) et celles qui nous intéressent directement, la constitution physique de l'édifice.
- Les informations répertoriées, données textuelles ou graphiques, seraient interrogeable à partir d'interfaces adaptées au type de critère sélectionné: plan 2D pour les angles de vues des illustrations, maquette numérique 3D pour le corpus des entités, etc…


*Figure 56: Plan de la place centrale de Cracovie dressé par K.Bąkowski en 1785* [*Tołwiński, 1939]*

Ces contraintes ont été prises en compte et le système actuel répond effectivement au cahier des charges que nous venons d'expliciter[129]. Les documents répertoriés dans la base documentaire aujourd'hui sont des sources bibliographiques, cartographiques et iconographiques rassemblées dans le cadre de ce projet pendant ces trois dernières années. Des critères descriptifs usuels dans ce genre de travail sont renseignés : auteur, éditeur, etc.. D'un autre côté, dix-neuf bâtiments sont recensés et référencés dans le système. Ces

[128] Instytut Historii Architektury i Konserwacji Zabytków, Wydział Architektury, Politechniki Krakowskiej, Kraków, Pologne, voir notamment [Czubinski et al, 1998a], , [Dudek et al, 1999b], [Dudek et al, 1999a].

[129] Nous reviendrons sur l'architecture logicielle mise en place (Système de Gestion de Bases de Données Relationnelles SQLServer, serveur Web IIS, Interfaces CGI/Perl, HTML et VRML) dans le détail en présentant l'outil SOL (nom de cette base documentaire) en fin de document

bâtiments sont aujourd'hui détruits, à l'exception de trois d'entre eux partiellement transformés (les évolutions des édifices sont ainsi un des critères descriptifs de chaque enregistrement). Un ensemble de critères nommés points de vus dédiés permet au système de prendre en compte des éléments d'information issus d'une étude approfondie des sources répertoriées. Le renseignement de ce groupe de champs pour chaque enregistrement permet d'effectuer sur la base documentaire des requêtes centrées sur des problématiques patrimoniales ou de modélisation, comme par exemple :

- Objet architectural -> en rapport au modèle d'entité architecturale.
- Période historique-> en rapport à la période de présence ou à l'évolution d'un édifice.
- Orientation -> en rapport à l'angle de vue sur la place de chaque illustration.
- Médium-> référence le type de media concerné (texte - illustration - plan, etc…).
- Groupe thématique-> en rapport à des problématiques générales (matériaux, techniques, conservation, etc...).
- Disponibilité-> Localisation et référence de la source dans les archives ou bibliothèques.
- Edifice-> référence des problématiques spécifiques à tel ou tel édifice (au sens de l'usage des lieux).
- A propos de-> référence à des auteurs ou des praticiens que la source cite.
- URL-> adresse URL contenant des informations additionnelles.

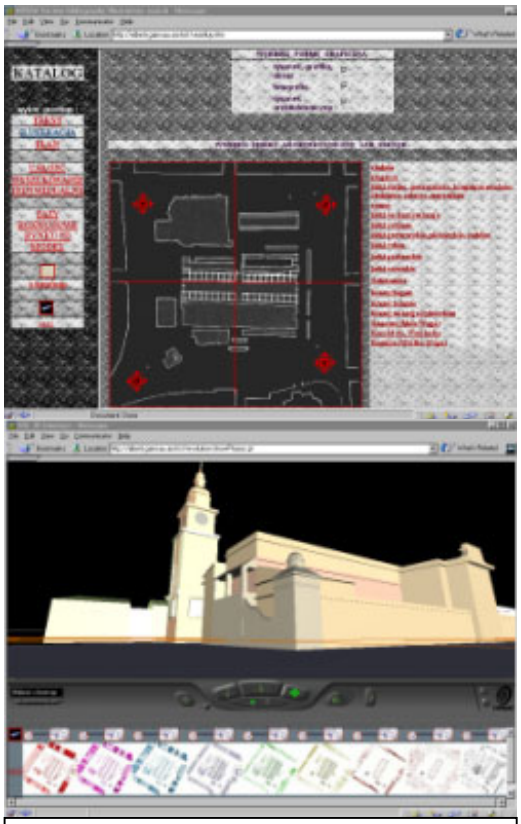



*Figure 57: interfaces de requêtes 2D et 3D sur la base documentaire (im auteur)*

On le voit, cette discussion nous éloigne largement des propos tenus jusqu'à présent dans le cadre de ce document. Pourtant, on le constatera ci-dessous, la notion de modèle architectural n'est pas loin puisque chaque enregistrement y fait référence. Toutefois, il convient de rappeler que l'objectif poursuivi ici était la constitution d'une base documentaire et non la validation d'une démarche de modélisation dont les objectifs sont bien différents. Il nous semble donc qu'il est juste de considérer que l'une comme l'autre sont des analyses parallèles d'un domaine de connaissance, analyses dont nous allons pouvoir évaluer la compatibilité. En effet, la démarche de modélisation que nous avons adopté souffre de limitations, déjà présentées, qui dans le cadre de la constitution d'une base documentaire sont rédhibitoires. Conscient de cet état de fait, notre approche a donc consisté à dégager entre le modèle et la base deux concepts pivot, points de contact entre l'un et l'autre, un critère attachant la source au corpus et un autre lui adjoignant une adresse URL commune.

#### 4.3.3.ii) Mises à jour et attachement des documents au modèle

En premier lieu, Les sources référencées dans la base documentaire sont des ouvrages ou documents graphiques, des travaux de recherche, etc…D'un autre côté, nous nous intéressons au tissu urbain, aux objets architecturaux et à leurs évolutions morphologiques. Nous proposons un formalisme d'attachement des premiers aux seconds à travers la définition de champs des enregistrements et d'une interface de requête. Cela a signifié pour nous d'introduire la notion d'éléments de morphologie dans les champs des enregistrements, et de les interroger au travers d'une maquette tridimensionnelle figurant l'édifice.
Par ailleurs, nous avons mis l'accent sur la nécessité d'autoriser les mises à jour distantes de la base. Ceci se traduit non seulement par la mise à jour des enregistrements, mais aussi par la mise à jour des critères appelés points de vues dédiés. Ainsi, il est possible d'ajouter par exemple à la liste de problématiques spécifiques à tel ou tel édifice de nouveaux éléments, ou encore de rajouter de nouveaux thèmes aux groupes thématiques. Cela correspond à notre

souhait d'expérimenter un outil évoluant à la fois par le contenu de la base et par les critères de requêtes. Les interfaces de consultation sont de trois types : interfaces textuelles classiques (formulaires CGI), plans (tissu urbain, orientation, évolutions et illustrations) et maquettes numériques 3D (VRML, corpus architectural). Les procédures de mises à jour tant du contenu de la base que des critères descriptifs sont textuels, le système se chargeant de mettre à jour les listes de critères et les interfaces de requêtes concernées.

#### 4.3.3.iii) La représentation

Les représentations du bâti qu'autorisent outils et de techniques numériques actuelles servent à l'évidence une forme de communication autour de l'édifice, mais qu'apportent t'elles en matière d'analyse de celui-ci ? En effet, la représentation de l'édifice peut, au-delà du champ de l'imagerie virtuelle, s'intégrer dans un dispositif d'étude du patrimoine bâti alliant restitutions en images de synthèse et gestion d'informations; dispositif d'étude apte à rendre compte de la complexité tant formelle qu'historique de l'objet architectural. La représentation ne peut dès lors être abordée sans interroger le rapport de l'image et d'un modèle de l'édifice qu'elle figure. C'est dans cette démarche que nous nous situons puisque la représentation de l'édifice correspondra pour nous à l'activation d'un ensemble de méthodes des instances en jeu dans une scène donnée : l'image figurera pour nous une expression du modèle et seulement cela.
Reste à discuter de la plateforme matérielle et logicielle utilisée. Le développement de modeleurs géométriques dédiés intégrant par exemple moteurs de rendus, modules paramétriques ou graphes hiérarchiques permet de typer morphologiquement les entités géométriques utilisées, accélérant d'autant la production de maquettes numériques figurant l'édifice. La modélisation géométrique de la forme architecturale ou urbaine reste pourtant - même avec des logiciels puissants - une opération longue et souvent fastidieuse.
La formalisation d'un modèle architectural apte à générer sa représentation dessinée facilite cette opération. C'est par exemple le cas du modeleur de scènes VRML développé pour le projet ARKIW[130], modeleur dans lequel la définition géométrique de l'objet est sous jacente (intégrée au modèle), développement dont nous reparlerons en fin de document.

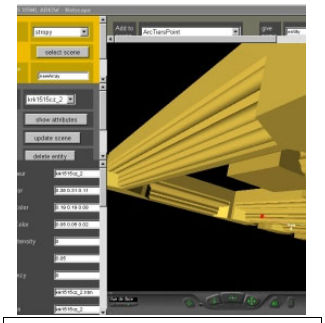


*Figure 58 : Un modeleur architectural (scènes VRML , projet ARKIW, im auteur)*

Aujourd'hui, la production de maquettes numériques répond à des cahiers des charges distincts :

- Les maquettes en formats propriétaire (outils de CAO-DAO), produites à partir d'interfaces utilisateur faciles à prendre en main, permettent de produire des images de qualité mais n'intègrent pas ou peu la notion de modèle.
- Les maquettes en formats dits d'échange ne restituent pas la richesse du modèle.
- Les maquettes en format standard pour le Web VRML sont potentiellement fidèles au modèle, indépendantes des applications de CAO DAO mais produisent des images de qualité moindre.
- Enfin, quel que soit le choix de plate-forme effectué se pose le problème du lien graphique / non graphique.

Une représentation sert donc bien un objectif, déterminant dans le choix de la méthode et de l'outil à privilégier, et constitue un résultat ponctuel à réintégrer dans un dispositif d'étude plus large de l'édifice. A titre d'exemple, dans le cadre d'études visant à simuler une hypothèse de restitution, la représentation géométrique exhaustive de l'édifice disparu se traduit essentiellement par la mise en évidence d'incohérences ou d'impossibilités dans les choix faits par les auteurs de l'hypothèse. Elle se traduit également par la nécessité pour

[130] ARKIW est un programme de coopération franco-polonais (UMR MAP-GAMSAU, iHAiKZ) soutenu par un Programme d'Actions Intégrées POLONIUM (MAE-CNRS / KBN), http://alberti.gamsau.archi.fr.

l'auteur de l'hypothèse de dimensionner chaque élément de l'hypothèse et par conséquent de disposer de nouveaux éléments de comparaison avec d'autres édifices de même type.
Utilisée comme moyen de simuler une hypothèse de restitution, la maquette numérique pose à l'auteur de l'hypothèse deux grandes familles de questions :

- Validité de la représentation. On distingue ici exactitude géométrique de la maquette et validité architecturale de l'hypothèse. Dans le premier cas les problèmes posés seront par exemple la gestion des niveaux de détail ou encore la gestion du décor. Dans le second, c'est l'analyse architecturale de l'édifice figuré qui est en jeu : observation des incohérences, comparaisons, etc.
- Codification de la représentation. Le problème posé ici s'apparente à la définition de règles d'usages pour la simulation d'hypothèses de restitution en images de synthèse : comment signifier des notions telles que l'incertitude, l'incomplétude, etc., quelles solutions adopter pour lier la représentation géométrique de l'édifice au modèle architectural sous-jacent, pour la partager en réseau ?

Nous ne développerons pas ici en détail ces points, très présents dans le domaine d'application qui est le notre mais débordant trop largement du cadre de ce travail. Nous allons centrer notre discussion d'abord sur les objectifs attendus des méthodes de représentation que nous proposons, puis sur les choix techniques qui en découlent. Nous commencerons par rappeler les choix qui étaient les nôtres au démarrage de ce projet, et tenterons de, justifier leur remise en cause éventuelle. En effet, nous croyons l'avoir déjà évoqué, l'ensemble de l'architecture logicielle de ce projet a été modifié entre 1997 et 1998, les migrations de C++ vers JAVA s'accompagnant par exemple également de la migration de Autocad vers MicroStation ou de VRML 1.0 vers VRML 2.0. Le processus que nous avions mis au point alors traduisait la morphologie, le positionnement et l'orientation vers diverses plates-formes de visualisation. Chaque plate-forme correspondait à un objectif et des développements particuliers brièvement présentés en **annexe 39**.

*4.3.3.iii.1. Méthodes de visualisation*

L'objectif poursuivi est donc simple : représenter une instance de notre modèle sur la plateforme Internet en décrivant des scènes en VRML figurant un jeu d'entités architecturales, et se servir de ces scènes comme d'une interface vers des données non graphiques. Les éléments figurés dans la scène étant issus du modèle, ce dernier point consiste en fait à affecter à chaque entité une propriété telle qu'une sélection utilisateur sur sa représentation appelle des données non graphiques. Le mécanisme d'ancrage d'URL aux objets géométriques que permet VRML répondra à cette attente. La classe abstraite Entité, tête de la hiérarchie des entités, dispose d'une telle propriété. En conformité avec leur définition, les attributs ne sont pas accessibles directement mais par l'intermédiaire des entités. C'est en cela que nous avons parlé plus tôt de limitation au modèle en disant : *Les attributs ne sont manipulables que par l'intermédiaire de l'entité*. La propriété URL n'étant pas caractéristique de l'entité mais du mécanisme par lequel s'établit le lien graphique / non graphique, celle-ci est encapsulée dans un objet outil de l'entité appelé VrmlOut, sur la description duquel nous reviendrons ultérieurement.
Deux autres propriétés sont ajoutées à la définition de la classe abstraite Entité en rapport avec ses méthodes de visualisation :

- La propriété niveau de détail reprend le principe des RangeLOD de VRML : plus l'objet est loin de l'observateur moins on le représente avec détail. La propriété fixe des seuils au passage desquels une représentation de l'objet en remplace une autre. Si cette propriété s'utilise aujourd'hui exclusivement sur la plate-forme de visualisation VRML, elle est d'une pertinence telle en regard du domaine d'application[131] qu'il nous a semblé nécessaire de l'extraire de la classe outil VrmlOut et de la prendre en compte comme un élément caractéristique de l'entité.

[131] On pourra si nécessaire s'en convaincre à la lecture de [Viollet Le Duc,1977] ou de [Choisy, 1899/1991].

- La propriété nSeg a une vocation proche de la précédente : elle permet en VRML de définir le nombre de segments à afficher pour représenter une courbe. Bien que plus proche d'une problématique exclusive de visualisation, elle traduit néanmoins également l'idée que l'affichage de la morphologie d'une entité doit être rapporté à la perception qu'en a l'observateur, et a donc également été prise en compte comme un élément caractéristique de l'entité.

Le fichier VRML résultat est constitué de sections imbriquées : en-tête et clôture gérés par le réseau puis sections entités comprenant l'expression de leurs attributs morphologiques. Un ensemble de méthodes est associé aux propriétés de la classe abstraite Entité :

- Les méthodes initWrl et endWrl permettent d'écrire les en-têtes et clôtures de chaque section entité. Elles appellent les méthodes relatives à l'écriture du node Transform positionnant et orientant l'entité dans l'espace de la scène, puis les méthodes relatives à la définition de la texturation générique (node DEF) appliquée à l'ensemble des formes géométriques dessinées dans cette entité. Chaque node, conformément à la logique de VRML, est nommé de façon à autoriser sa manipulation a posteriori.
- Les méthodes relatives à l'écriture du node Transform (writeRef et closeRef) positionnant et orientant l'entité font partie de la classe de base Entité. C'est le cas également de la méthode drawRefWrl qui permet d'afficher sous la forme de trois axes (IndexedLineSet) le référentiel de l'entité.
- La classe de base Entité dispose également de méthodes dédiées à l'animation de la scène, appelées writeSensorsWrl et closeSensorsWrl. Ces méthodes définissent dans le corps de la section entité du fichier VRML, un ensemble de Sensors, de DirectionalLight et d'interpolateurs prédéfinis. Pour autant qu'un routage adéquat soit défini par le réseau, chaque Sensor va ainsi être attaché à une action à effectuer sur l'entité architecturale. Les Sensors utilisés sont de deux types : timeSensor ou TouchSensor. Ils sont en nombre suffisant pour autoriser l'activation indépendante de chaque type d'action (éclairage, mouvements de translation, de rotation ou d'homothétie). Un node Switch écrit par ces mêmes méthodes permettra de basculer d'une représentation donnant accès à des données non graphiques par le biais d'une ancre URL à une représentation utilisant la morphologie de l'objet comme un déclencheur d'événements (Touch Sensor). Cet ensemble de mécanismes de gestion d'animations est écrit par défaut comme inactif, le node Switch étant initialisé pour lire la section Anchor (node URL) de la définition de l'entité.

L'affichage d'une scène fait intervenir, au-delà des méthodes propres à la classe abstraite Entité et non surchargées par ses descendants, un ensemble de méthodes propres à l'objet VrmlOut que chaque entité possède par le même héritage :

- Affectation d'une texture à l'objet par le biais des méthodes setDefaultMat et setMat dont le résultat est l'affectation de valeurs aux champs correspondant au modèle d'éclairement (diffuseColor, ambientColor, etc..)
- Ecriture d'un node Shape nommé comprenant la définition exhaustive de la texture, node qui sera référencé (mot clé USE du langage) par les différentes formes géométriques dessinées pour cette entité.

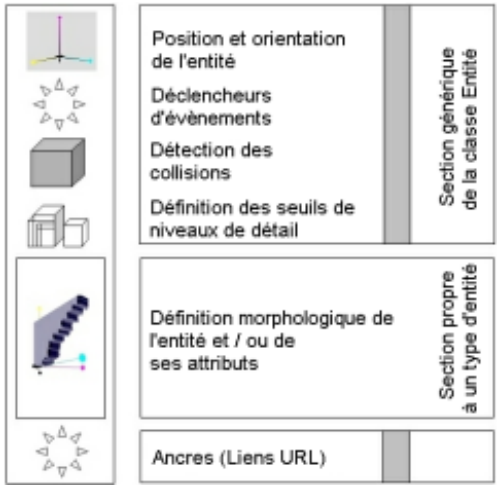



*Figure 59 : Méthode d'écriture au format VRML, partie générique (classe de base Entité) et parties spécifiques à chaque entité en dérivant*

La description de la morphologie de l'entité elle-même (ou de ses attributs) est assurée par une méthode surchargée dans les classes dérivant de la classe abstraite Entité. Cette méthode, appelée ecritureWrl, se charge d'effectuer les opérations nécessaires au renseignement de l'ensemble des attributs morphologiques (valeurs par défaut par exemple). Par ailleurs, cette méthode fait appel aux méthodes adéquates des

attributs morphologiques lorsque l'entité en contient. Elle gère l'écriture de la partie morphologie de la description d'une entité, partie nommée par un node Transform. L'imbrication à l'intérieur de ce node Transform de l'ensemble des données morphologiques assure la cohérence de la représentation des attributs morphologiques[132].
Dans le cas d'entités dont la propriété Profil est affectée, la méthode ecritureWrl se charge d'effectuer les opérations nécessaires à l'évaluation de chaque moulure. La figure ci-avant récapitule l'ordre dans lequel les sections écrites par la classe abstraite Entité et les sections relevant de chaque entité apparaissent à l'intérieur de chaque section Entité d'un fichier VRML. Pourtant, ce fichier comprend également des données qui ne sont pas relatives à telle ou telle entité mais à l'ensemble, comme l'éclairage de la scène ou les points de vue. Ces renseignements sont placés en première partie de fichier après lecture par le réseau de ses propriétés et de celle de ses entités. Le réseau se charge donc d'écrire les en-têtes et clôtures du fichier VRML résultat[133].

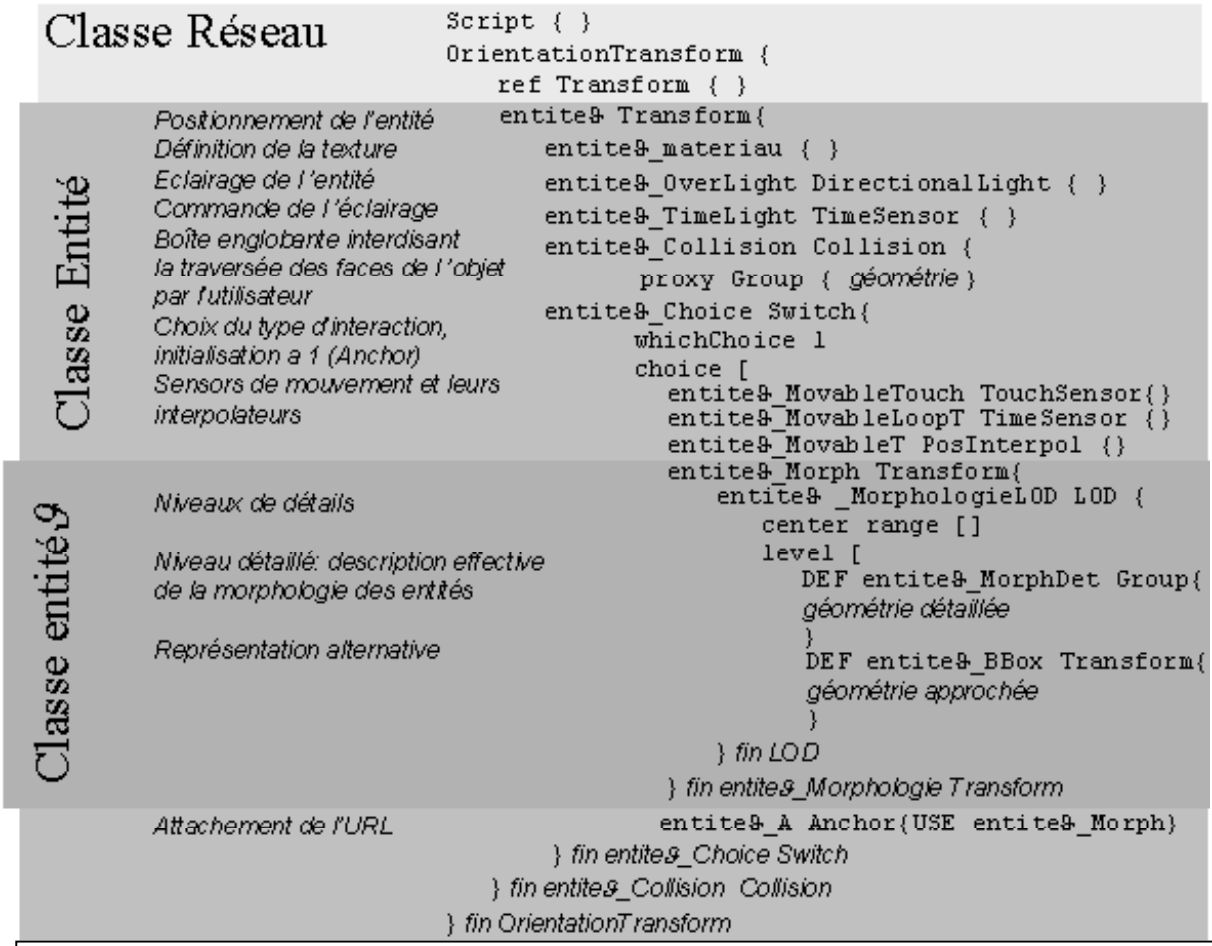



*Figure 60 :Contenu des différentes sections du fichier VRML généré par le réseau, section écrites par le réseau, section écrite par la classe de base Entité, section écrite par chaque entité.*

Le schéma résultant de cette organisation laisse apparaître nos principaux choix :

[132] Voir éventuellement l'annexe consacrée à VRML en fin de document pour un résumé des éléments de la syntaxe du langage utilisés.
[133] Les méthodes chargées de ces tâches sont les suivantes :
-Les méthodes InitAllWrl et endAllWrl n'écrivent que des éléments de syntaxe du langage VRML nécessaires mais sans intérêt pour nous.
-La méthode writeScriptWrl est utilisée pour calculer des points de vue standard sur la scène : six vues frontales et quatre axonométries. Un script VRML évalue à partir de couples point observateur / point visé les positions et orientations des caméras. Cette évaluation est faite au chargement de la page dans le navigateur par appel à la fonction réservée initialize (). La détermination des points observateur est menée à bien par le réseau qui détermine sur ses entités une boîte englobante servant de base à cette détermination.

- Un concept outil, vrmlOut, qui rassemble les propriétés spécifiques à la plateforme de visualisation
- Des propriétés proches de la plateforme de visualisation mais pertinentes sur le domaine d'application, intégrées à la définition du concept d'entité
- Des méthodes génériques regroupées dans les classes Entité et Réseau chargées du positionnement, de l'orientation, de l'accès aux données non graphiques et du routage d'évènements.
- Des méthodes spécifiques à chaque entité limitées à l'écriture d'une morphologie imbriquant aspects propres aux entités et aspects liés à ses attributs morphologiques.

#### 4.3.3.iv) Choix techniques associés

Nous allons ici voir comment les procédures d'écriture de scènes VRML s'intègrent dans notre dispositif global de modélisation et d'interfaçage de données patrimoniales. Il faut ici distinguer les phases d'élaboration d'une maquette et d'exploitation de cette maquette dans le cadre par exemple de l'interface d'accès à la base documentaire. En effet, l'objectif poursuivi en représentant en VRML un jeu d'instances n'est pas forcément lié à une problématique de gestion de données. La maquette numérique sert également, dans un processus par essais-erreurs, à argumenter la formulation d'une hypothèse de reconstruction. Celle ci va trouver sa source dans une étude documentaire de l'édifice, et utiliser le jeu de concepts architecturaux mis en place pour valider ou invalider un scénario ponctuel d'assemblages d'objets. C'est typiquement l'approche suivie dans [Dudek et al, 1999a], et illustrée par la maquette ci-contre. Deux problématiques se dégagent :

- L'élaboration d'une interface permettant l'instanciation et la représentation au format VRML des concepts dont le modèle autorise la manipulation. Il s'agit ici de la création de maquettes numériques à des fins de simulation d'hypothèses de restitution, mais également de la création de scènes figurant tel ou tel état du tissu urbain sans ambition de réalisme.
- L'établissement d'un lien entre d'une part la représentation de l'édifice et d'autre part la base documentaire telle que décrite à la section précédente.

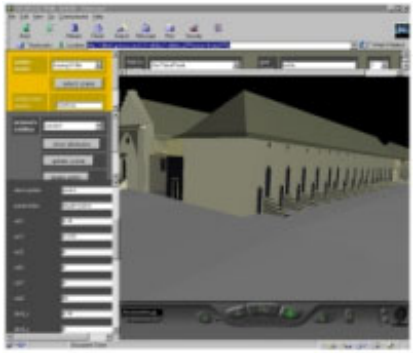


*Figure 61 La maquette VRML, un scénario d'assemblages d'objets du modèle (im auteur)*

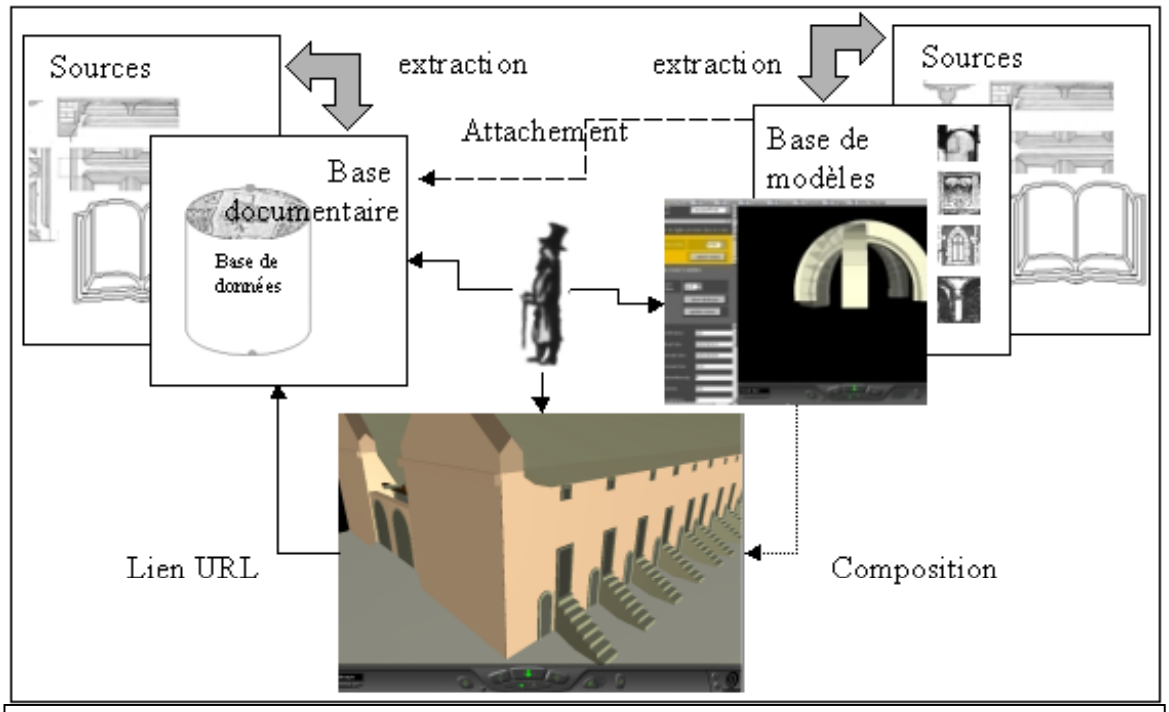



*Figure 62 : La maquette numérique, générée à partir du modèle, liée à la base documentaire*

Nous considérons que la scène VRML produite par instanciation du modèle constitue le lien naturel entre ces deux problématiques ; problématiques qui font par ailleurs l'objet de développements autonomes. Ainsi, comme tente de l'illustrer le schéma ci-dessous, la maquette numérique va servir à l'explicitation de la base de modèles comme à l'interrogation de la base de données. Une architecture d'interface commune nous a semblé s'imposer, nous avons donc développé des outils allant dans ce sens en privilégiant des accès Internet, applets JAVA pour le mesurage et HTML standard pour le reste. Nous travaillons aujourd'hui (hors mesurage) sur le principe suivant :

- Ecriture au vol des pages interfaces utilisateurs (HTML) par le biais des classes outils InterfacesWeb propriétés des réseaux et des entités. Ceci est vrai tant dans l'interface de création de scènes que dans l'interface de consultation de la base de données où les mêmes modules sont utilisés de façon autonome.
- Interfaces utilisateurs par formulaires CGI : les sélections utilisateurs sont traitées par un serveur Web (IIS) et décodées par une classe outil Decodage qui s'appuie sur le module de [Brenner et al, 1996] et par les classes outils InterfacesWeb. Ces sélections correspondent dans le cas de l'interface de création de scènes à l'activation de méthodes du réseau courant, dans le cas de l'interface de consultation de la base documentaire à l'écriture au vol d'une nouvelle page, à l'appel d'une scène VRML ou a une requête IDC adressée au SGBD.
- Scènes VRML dans lesquelles chaque entité architecturale représentée dispose d'une ancre dont l'URL associée est une page Web standard, une page écrite au vol ou une requête IDC adressée au SGBD. Les scènes utilisées dans le cadre de l'interface de consultation de la base documentaire sont des photographies du modèle : elles ne sont pas recalculées lors de leur appel. Par contre, les scènes VRML affichées dans le cas de l'interface de création de scènes sont écrites au vol.

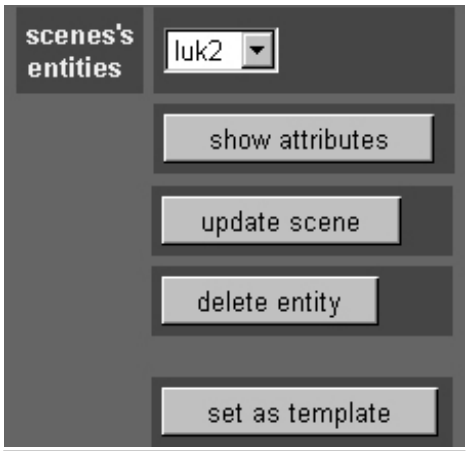



*Figure 63 : Activation de méthodes du réseau courant par le biais d'un formulaire HTML standard (outil VALIDEUR, voir [Dudek et al, 1999a])*

Il y a donc autonomie de la scène (elle peut être visualisée indépendamment de son contexte de fabrication) ou dépendance de celle-ci au modèle, et ce en fonction de son contexte d'utilisation. Ce choix n'est qu'en contradiction apparente avec notre souci de considérer la représentation comme l'expression d'un jeu d'instances du modèle. En effet, rien n'empêche la mise à jour des scènes utilisées dans l'interface de consultation de la base documentaire. Toutefois les modifications à y apporter ne nous semblent pas du tout relever du travail d'essai-erreurs dans lequel un lien constant du modèle à la maquette est nécessaire. Nous considérons donc que la séparation des deux interfaces est une approche pragmatique qui se justifie, d'autant que les formalismes d'outils mis en place de part et d'autre sont eux conjoints.

Nous présentons en **annexe 40** quelques exemples des représentations produites en les replaçant dans le cadre des applications à l'intérieur desquelles elles sont utilisées[134].

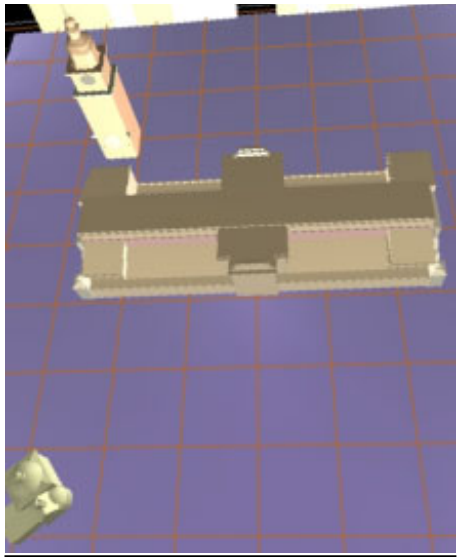


*Figure 64 : Les scènes VRML utilisées de façon autonome comme photographies ponctuelles d'instances du modèle*

[134] En conclusion, rappelons quels types de représentations sont associés au modèle dans l'état actuel de notre travail :
-Sortie vers le modeleur MicroStation, uniquement dans le cadre de la plate-forme de mesurage développée par P.Drap, et nommée ARPENTEUR.
-Sorties VRML 2.0 dont nous venons de parler, dans le cadre des outils ARPENTEUR, HUBLOT, SOL et VALIDEUR, applications présentées en détail à la fin de ce document
-Rendus POV en cours de (re)développement dans le cadre de l'outil VALIDEUR.

### 4.3.4. Connaissances associées au modèle

Il sera ici question non plus de modélisation architecturale mais du traitement des connaissances relatives à l'exploitation du modèle élaboré, en d'autres termes relevant d'autres disciplines. Nous pensons en priorité à la géométrie, mais aussi aux concepts isolés dans le cadre de notre approche du relevé architectural. Nous présentons assez brièvement comment sont prises en compte ces connaissances associées, et distinguons leur définition et leur organisation de leur intégration au modèle. Nous n'introduisons ici que les aspects généraux des notions associées au modèle, leur implémentation sera discutée dans la section suivante.

#### 4.3.4.i) Aspects morphologiques : géométrie

La formalisation de notions de géométrie est déterminante dans plusieurs aspects de l'étude de l'édifice:

- La géométrie intervient dans le processus de mesurage : la forme des entités est évaluée au travers de primitives géométriques simples. Les points relevés sont affectés à des entités architecturales mais aussi à ces primitives géométriques.
- Elle intervient dans la manipulation des entités en fixant notamment la position et l'orientation de chaque instance du modèle en rapport au référentiel du réseau en cours.
- Elle sert d'intermédiaire entre des formalismes théoriques fixant une morphologie (profils par exemple) et leur visualisation effective sur telle ou telle plate-forme matérielle et logicielle.
- Elle est déterminante enfin dans la définition des concepts qui, intégrés aux entités architecturales, en permettent une figuration approchée (référentiel et boîte englobante).

Une hiérarchie de classes outils géométriques a été formalisée afin de répondre à ces différents besoins, hiérarchie dont nous allons présenter le contenu avant de préciser quels concepts contribuent à la définition du modèle architectural.

##### *4.3.4.i.1. Outils géométriques*

Nous avons distingué quatre groupes de notions intervenant ici ou là dans le processus de mesurage ou d'exploitation graphique du modèle architectural: les points, les primitives géométriques, les OPPI (pour *Objet defini Par des Points Interpoles*) et les polyèdres. Chaque groupe intervient comme outil du modèle à des phases et en fonction d'objectifs indépendants. Leurs propriétés communes (booléens de contrôle d'état) sont factorisées au sein d'une classe nommée ItemGeometrique, elle même dérivée d'une superclasse dans laquelle sont factorisées les propriétés d'identification des objets communes aux classes outils et aux classes architecturales. Nous allons ici expliquer le rôle de ces quatre groupes de notions, sans détailler les hiérarchies correspondantes mais en situant les contextes d'utilisation :

- Une classe Point sert de tête de hiérarchie au groupe de notions représentant les points. Elle est constituée d'un triplet de réels et de propriétés liées au contexte du relevé (seuil d'égalité, nombre de décimales, etc…). Elle factorise un ensemble de méthodes de redéfinition d'opérateurs sur ce triplet : addition ou soustraction membre à membre, produit scalaire, produit vectoriel, etc… D'autres méthodes de la classe Point ont été écrites pour évaluer par exemple la distance d'un point à une droite, la coplanarité de quatre points, etc… Ces méthodes ont été développées par P.Drap dans le contexte du relevé. La hiérarchie de concepts représentant les points est développée à la fois dans ce cadre du relevé et dans celui de la représentation. Dans le premier cas, les points, résultat de la saisie opérateur, forment l'ensemble des observations à partir desquelles les attributs morphologiques des entités seront évalués. Des propriétés relevant de la problématique de la mesure (coefficient de corrélation par exemple) et adjointes à la définition des points justifient les dérivations de la classe Point que propose P.Drap [Drap, 1997]. Par ailleurs, la visualisation des points mesurés se fait au travers de méthodes spécifiques aux points issus de la mesure, points qui forment donc les spécialisations de la classe Point. Dans le cas des méthodes de représentation des entités,

les points interviennent comme outil dans la définition des moulures et des profils, cette fois-ci sans visualisation autonome.

– On l'a dit, les attributs morphologiques des entités sont évalués dans le processus de relevé que nous avons expérimenté au travers de primitives géométriques calculées à partir du nuage de points mesurés. La primitive géométrique est un modèle théorique à renseigner à partir des observations. Des primitives géométriques simples doivent donc décrire des parties significatives de la surface des entités comme le plan du front d'Arc, le cercle de son Intrados, etc... Ces primitives géométriques assurent le lien entre le résultat de la mesure et les attributs morphologiques des entités puisqu'il est possible d'établir un rapport direct ou indirect entre la primitive instanciée et tel ou tel attribut morphologique. Dans le cas du fût de colonne ci-contre, un ensemble de cercles permettent d'évaluer les attributs Module et Module diminué de l'entité Fût, correspondant à ses rayons supérieur et Inférieur. Il n'est pas possible ici de d'établir un rapport direct entre les cercles instanciés et ces deux attributs puisque d'une part le fût est incomplet et que d'autre part sa morphologie réelle est encore inconnue (galbe à déterminer). Un ensemble de modèles théoriques (Plan, Cercle, Droite) dérivent d'une classe Primitive tête de la hiérarchie de concepts associés à l'évaluation de la morphologie des entités. Dans le schéma de la base ci-contre nous figurons à gauche un sous-ensemble des Primitives calculables à partir d'observations sur l'objet construit et à droite les attributs morphologiques correspondant.

*Figure 65 : Points relevés à la surface de fûts de colonne (voir [Florenzano et al, 1996a] )*

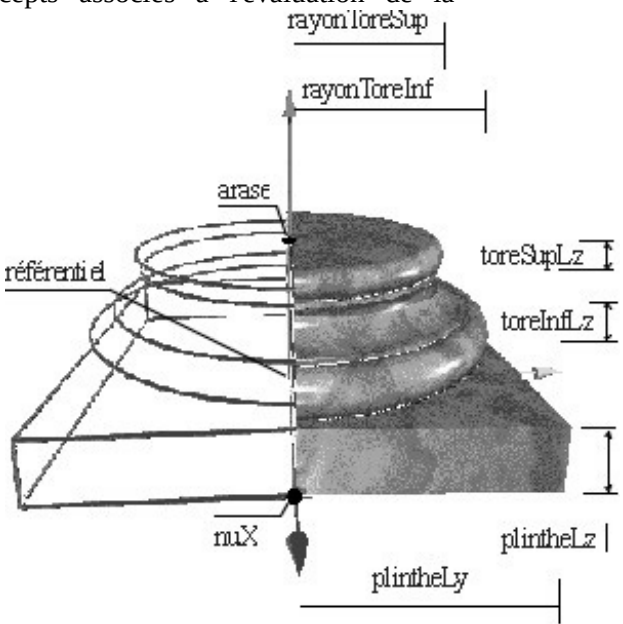



*Figure 66 : Rapport des primitives géométriques aux attributs morphologiques d'une entité dans le cas de la base toscane (voir [Florenzano et al, 1996a]*

Faut-il utiliser cette hiérarchie de primitives dans le cadre de la représentation ? Notre réponse est négative, et nous souhaitons en indiquer la raison. Les primitives géométriques décrivent de façon redondante des spécificités observables à la surface des entités architecturales mais ne recouvrent que rarement leurs attributs morphologiques. Il n'y a donc aucun lien systématique entre les classes de Primitives formalisées dans le cadre du relevé et la représentation de l'entité par une série d'opérations booléennes sur des solides (POV-RAY ou AUTOCAD) ou par appel à une bibliothèque de primitives géométriques simples (VRML). Dans le cas des représentations, le rapport entre l'attribut morphologique à représenter et l'éventuelle primitive géométrique (VRML) susceptible d'être utilisée pour le faire est explicitement écrit dans le corps de la méthode de représentation. Autrement dit, nous n'établissons volontairement aucun rapport entre un ensemble de concepts destinés à l'observation de l'objet construit et un ensemble de méthodes de représentation. Pourtant, il peut arriver qu'il y ait ressemblance entre le jeu de Primitives issues de la mesure et le jeu de Primitives utilisées pour représenter l'objet. Toutefois, le non-systématisme de cette ressemblance condamne par avance toute tentative pour considérer que l'on peut représenter l'objet au travers de formes géométriques simples. Nous voyons à cela deux raisons essentielles : la morphologie des entités n'est souvent pas assez simple pour le permettre (moulurations complexes), le jeu de primitives des plateformes de visualisation varie trop pour asservir la représentation de nos entités à la présence de telle ou telle primitive sur telle ou telle

plateforme. Nous insistons donc sur le fait que les concepts regroupés dans la hiérarchie des Primitives ne nous servent que dans le cadre du relevé architectural et n'ont pas vocation à servir de boîte à outils dans la représentation des entités.

- La hiérarchie des OPPI (pour *Objet defini Par des Points Interpoles*) a été développée pour représenter des objets caractérisés par une liste de points. Son application principale est pour moi le concept de LigneBrisee, objet utilisé dans la gestion des Profils. La classe LigneBrisee définit un objet caractérisé par une liste de points et doté de méthode d'accessions, de mise à jour et de gestion de liste.
- Le développement de la hiérarchie des polyèdres par D.Merad s'inscrit dans le cadre d'applications de ce projet dans lesquelles je ne suis pas personnellement impliqué, je n'en dirai donc que peu de choses. L'objectif poursuivi ici était de simuler le comportement mécanique des grands édifices patrimoniaux en appareillage de blocs en utilisant les méthodes développées par les mécaniciens pour la simulation du comportement de structures homogènes (modèles des éléments finis ou discrets). Une modification de la structure interne des entités architecturales devait permettre de représenter leurs composants, à savoir des appareillages de blocs simples dans l'expérience concernée. Une représentation géométrique compatible avec les outils de simulation par les éléments finis fournis était à développer, outils qui impliquait de gérer des objets définis par une liste de sommets, une liste d'arêtes et une liste de faces, caractéristiques des objets de la hiérarchie des polyèdres. Pour plus de détail sur cette application particulière nous renvoyons à la lecture de [Acary et al, 1999].

Nous avons introduit des groupes de notions formalisées pour servir à telle ou telle exploitation du modèle. Nous allons maintenant indiquer quelles propriétés des concepts architecturaux sont définis dans ces groupes.

*4.3.4.i.2. Propriétés géométriques intégrées au modèle*

Nous décrivons ici sous la forme d'une liste commentée les propriétés géométriques intervenant dans la définition des concepts architecturaux (réseaux, entités, relations et attributs). On retrouvera d'abord dans cette liste des notions déjà introduites au point précédent (points, primitives) et qui servent d'outils également dans le processus de mesurage. Mais on introduira également des notions utilisées sous une forme particulière dans le seul cadre de ces concepts architecturaux (Référentiel).

- Les points sont utilisés dans la définition de la Moulure et définissent les extrémités d'une instance de cette classe. Ils sont également utilisés dans la gestion des moulures au travers du formalisme de ligneBrisee. Enfin, une des représentations des sols intègre une liste de points permettant de prendre en compte les irrégularités de surface de la face exposée.
- Une ligneBrisee est utilisée dans la définition de la Moulure pour gérer la représentation de celle-ci. Une ligneBrisee est également utilisée dans la définition du Profil pour la même raison, la première renseignant la seconde.
- Un référentiel est utilisé pour positionner les entités dans le réseau et pour donner une origine (objet Point) à celui-ci. Ce référentiel est propriété de le classe ItemArchitectural qui factorise l'ensemble des propriétés communes aux concepts architecturaux.

### 4.3.4.ii) Connaissances liées à la mesure

Ainsi que brièvement évoqué au chapitre "exploitation du modèle, la mesure" chaque entité architecturale modélisée détient un jeu de primitives redondant, jeu de primitives correspondant à ses spécificités morphologiques. Chaque primitive dispose d'un mécanisme d'auto-evaluation appelé EGO (Etre Géométrique Optimal), en charge de retrouver dans un nuage de points les propriétés de la primitive. Les EGO sont en fait des modèles fonctionnels décrivant les relations liant le modèle théorique visé (la primitive géométrique) et les quantités observées (les points mesurés en photogrammétrie). Ainsi, pour chaque primitive géométrique supportée :

- il existe, dans la classification décrivant les primitives, une classe apte à effectuer toutes les opérations qui s'y rattache (affichage, calcul dans l'espace ...)
- par ailleurs, une classe homologue dans l'arbre des EGO décrit son modèle fonctionnel et assure le lien entre la mesure et la primitive géométrique. Chaque classe de l'arbre des EGO a pour membre une primitive homologue de l'arbre des primitives géométriques.

Les EGO se chargent donc de l'interface entre la mesure et la géométrie. La méthode consiste à minimiser la somme des carrés des distances dans l'espace entre les observés (les points) et le modèle théorique (la primitive géométrique). En d'autres termes la primitive géométrique calculée par l'EGO passe au mieux (au sens des moindres carrés) par le nuage de points observé. Le calcul s'appuie donc sur la connaissance qu'a l'opérateur du domaine lors de la phase de mesurage. En effet les points ont été non seulement affectés à des entités architecturales mais aussi à des primitives géométriques. Il appartient à l'opérateur de vérifier a posteriori la pertinence du modèle géométrique proposé lors de la phase de mesurage. Une étude attentive des résidus (écart entre les points mesurés et la primitive calculée) et de la répartition des points sur la primitive doit valider le modèle. Nous renvoyons pour une présentation plus détaillée de ces classes à [Drap, 1997], [Drap et al, 1999a].

#### 4.3.4.iii) Connaissances liées aux représentations graphiques

La problématique de la représentation recouvre deux questions complémentaires : celle de la géométrie à figurer, déjà évoquée, et sur laquelle nous ne ferons ici que des rappels, et celle du modèle d'éclairage (ici dans le cas de VRML) sur laquelle nous nous pencherons plus en détail.
Nous avons dans les sections précédentes introduit deux aspects liés à la géométrie du modèle architectural :

- En illustrant la définition des entités et des attributs nous avons présenté un formalisme générique de moulure permettant de gérer le profil comme une succession de moulures dont les tracés deviennent dès lors indépendants les uns des autres. Nous avons rappelé qu'un des objectifs poursuivi était de faciliter leur représentation tridimensionnelle en définissant un formalisme de points de contrôle / liste de segments intermédiaire. Ce formalisme permet en effet de prendre en compte les contraintes liées à la visualisation (niveau de détail en particulier) : la figuration des objets architecturaux tient compte de connaissances liées ici à leur facettisation.
- Dans la section consacrée à la représentation comme expression du modèle nous avons largement discuté des méthodes de visualisation que nous privilégions et avons évalué leur répercussion sur le modèle.

Nous avons également indiqué l'existence d'un objet outil de l'entité appelé VrmlOut, objet appelé par les méthodes d'écriture au format VRML. Nous allons brièvement en présenter la définition. La classe vrmlOut définit des objets contenant trois types d'informations :

- Un matériau dont les propriétés sont définies dans un objet vrmlMat. Ces propriétés correspondent au modèle empirique d'éclairement tel que le définissent [Péroche et al, 1988, p171].
- Une source lumineuse associée à l'entité dont les propriétés définissent un objet vrmlLight. Chaque source lumineuse est décrite par des propriétés relevant du modèle d'éclairement (couleur, intensité, etc..) mais dispose également d'une propriété appelée status qui permet d'en manipuler l'état à l'intèrieur de la scène (voir mécanismes de routage d'évènements en VRML dans l'annexe sur le sujet).
- Enfin, le mécanisme d'ancrage URL de VRML est géré par un objet vrmlAnchor qui dispose de propriétés fixant l'adresse URL retenue, un texte descriptif du lien et un champ paramètres permettant de spécifier la fenêtre du navigateur cible du lien.

Chacun de ces objets dispose d'une méthode d'écriture en VRML dont le résultat est concaténé par la méthode d'écriture en VRML de l'entité. Chacun de ces objets dispose

également de méthodes d'initialisation à partir de valeurs par défaut. L'interface des entités donne accès à l'ensemble des propriétés décrites ci-dessus[135].

#### 4.3.4.iv) Connaissances liées aux données non graphiques

Nous distinguerons ici les données relatives à la définition du modèle et celles relatives au renseignement des instances. En effet, les sources documentaires interviennent à ces deux niveaux, nous nous attacherons à en expliquer les conséquences en terme de modélisation dans l'état actuel du projet, et en rappellerons le caractère encore prospectif dans une large mesure. Nous introduisons dans un premier temps le formalisme retenu à l'heure actuelle pour donner à chaque concept une référence bibliographique. Nous présentons ensuite plus en détail le mécanisme de référencement bibliographique des instances.

##### *4.3.4.iv.1. Documentation des concepts*

Nous appelons documentation des concepts la définition dans les classes décrivant ces concepts d'une propriété permettant de lui attacher un ensemble de sources documentaires. Le premier point à soulever est la diversité des sources en question. Il peut en effet a priori s'agir de sources bibliographiques classiques, mais aussi d'illustrations, de géométraux, de maquettes numériques, etc… De plus, nous travaillons dans le cadre du projet ARKIW en trois langues : français, anglais et polonais. Chaque concept architectural a fait l'objet d'un travail de définition et de traduction que nous présenterons plus loin (application DIVA[136]). Une base documentaire a ainsi été réunie et portée sur le Web au travers d'une interface de consultation et de mise à jour d'un SGBDR.
Nous disposons donc déjà d'un ensemble de sources décrivant les concepts architecturaux. Le principe retenu tient compte de cet acquis puisque nous attachons à la classe une variable de classe dont le type est une adresse URL, par défaut requête idc sur le SGBDR SQLServer. Cette ébauche de solution nous semble être bien adaptée à la diversité des sources documentaires potentiellement rassemblées sur chaque concept. En effet, le concept isolé est un invariant du modèle, sa justification peut par contre évoluer. Dans l'implémentation que nous développons actuellement seuls les concepts décrivant des entités architecturales sont concernés. Les attributs, manipulables uniquement au travers de leurs entités propriétaires, ne sont présents dans les interfaces développés que sous la forme d'instances. Quant aux réseaux, l'organisation des concepts n'est pas suffisamment avancée pour permettre un tel mécanisme. Compte tenu de l'importance de la documentation dans le domaine d'application qui nous occupe, nous considérons que ce point doit être inscrit dans les perspectives à court terme du projet. Nous résumons en **annexe 41** l'implémentation actuelle du mécanisme de documentation des concepts, ainsi cas des tables de rapports modulaires fixant les proportions relatives des attributs morphologiques d'entités architecturales du corpus antique qui relève de la même problématique.

##### *4.3.4.iv.2. Documentation des instances*

La question dont nous discutons ici a déjà été évoquée dans notre présentation des choix techniques associés à la représentation du modèle. Il s'agit pour nous d'associer des données qualitatives (bibliographie, repères historiques ou stylistiques, données sur l'auteur ou sur la datation) aux instances du modèle que figure telle ou telle scène. Citons, pour rendre ce point plus clair, quelques questions auxquelles peut devoir répondre une instance d'un modèle de l'édifice :

- A quelle affiliation stylistique correspond-elle ?
- De quelle influence témoigne t'elle ?

[135] Deux remarques doivent être formulées ici. En premier lieu, la classe vrmlOut ne rend pas compte de l'ensemble des éléments de la syntaxe VRML qui pourraient nous intéresser, comme par exemple la définition de textures plaquées (images fixes ou animées), le suivi des déplacements de l'observateur dans le cas des réseaux, etc... En second lieu, il n'est établi aucun lien entre les données qualitatives sur l'entité à représenter (période, type d'opus, etc…) et les paramètres gérant l'éclairement de l'entité. Ces deux directions n'ont pas été développées faute de temps, mais peuvent constituer des améliorations importantes au formalisme actuel.
[136] Voir également [Czubinski et al, 1998a]

- S'inscrit t'elle dans une interprétation régionale d'un style ?
- Quel est son commanditaire ?
- Quelle est sa date de construction, son ou ses concepteurs, son ou ses constructeurs ?
- Quelles ont été les méthodes de mise en œuvre des matériaux utilisées ?
- Quelles ont été ses transformations ?
- Qui l'a décrite ?
- Qui l'a représentée ?
- Est-elle une copie ?
- Quelles ont été ses localisations successives ?

Une première approche consiste à doter les concepts architecturaux de variables d'instances, supports de ces données. Ces propriétés viendraient en complément de la définition des concepts architecturaux présentée plus haut, réseaux, entités, attributs, relations, essentiellement centrée sur des données quantitatives (dimensionnement des objets, positionnement, représentation). Et effectivement, des propriétés qualitatives sont intégrées à la définition du modèle architectural, son identificateur par exemple. Mais cette approche pose deux problèmes qui nous semblent la remettre en cause de façon importante :

- Le positionnement de ces propriétés dans l'arbre d'héritage des concepts architecturaux n'est pas, loin s'en faut, une question triviale. Notre logique de modélisation conduirait naturellement à les factoriser au sein d'une classe chapeautant les concepts d'entités, d'attributs, de réseaux et de relations. Pourtant on s'apercevra très vite que la plupart d'entre elles ne sont pas partagées par ces quatre groupes de concepts: il n'existe par exemple pas de copie d'une relation, ni d'affiliation stylistique des attributs morphologiques. Nous serons donc conduits à choisir, pour chaque propriété, un ou plusieurs groupes de concepts pertinents. Admettons que nous ayons pu reclasser nos quatre groupes pour intégrer ces propriétés à bon escient, ici une telle, là une autre, et ce dans les classes chapeautant les hiérarchies des réseaux, des entités, des attributs et des relations. Reste à prouver que ces propriétés sont pertinentes à l'intérieur même de ces quatre arbres. Prenons le cas de la datation d'un élément construit. A l'intérieur de la hiérarchie des réseaux, cette propriété peut perdre tout sens puisque les réseaux seront des outils d'analyse de l'édifice: l'ensemble des plafonds restaurés d'un édifice n'est pas caractérisé par une date. C'est déjà le cas de nos réseaux neutres actuels dans lesquels cette notion de datation n'est pas viable. Prenons un autre exemple, à l'intérieur de la hiérarchie des entités. La présence d'un descripteur fixant les localisations successives d'instances des classes représentant les Murs n'aurait aucun sens, mais elle en a pour les classes représentant les baies dont on sait qu'elles sont souvent déplacées ou réemployées. Il ne nous semble pas utile d'aller plus loin dans cette discussion, dont deux faits nous paraissent ressortir: certaines propriétés peuvent être assimilées à des points de vue, à représenter par le biais de réseaux comme indiqué aux sections précédentes; d'autres ne seraient représentables par des variables d'instances qu 'en soustraction de propriétés. Les données qualitatives dont nous souhaitons faire état ne seront donc pas incluses dans la définition des concepts architecturaux sans une remise ne cause du modèle proposé et de sa logique d'organisation des concepts.
- Une deuxième question vient à son tour remettre en cause une approche visant à représenter les données qualitatives par le biais de variables d'insistance. Elle ne tient pas cette fois au modèle mais au caractère des données en question. En effet, nous sommes en face de deux inconnues: le type de ces données (au sens des langages de programmation à typage) et leur évolution. Rappelons que notre objectif est d'associer aux instances des données aussi variées que le sont les sources de documentation à prendre en compte: iconographies diverses, références bibliographiques, textes bruts, etc.. Rappelons également que nous ne pouvons pas figer un jeu de types dans la mesure ou nous souhaitons faire évoluer les données attachées aux instances. Le type des données à attacher au modèle nous est donc inconnu. De la même façon, nous ne sommes pas en mesure de déterminer l'évolution à attendre des données en question, que ce soit en terme

d'évolution du support technique ou en terme d'évolution des descripteurs eux-mêmes dont nous ne pouvons fournir qu'une liste réduite. Nous rappellons à ce sujet que nous avons ainsi dans le cadre de l'outil de consultation de la base documentaire SOL autorisé les mises à jour distantes des enregistrements comme des critères descriptifs. L'objectif est donc clairement de ne pas figer un jeu de descripteurs. Les données qualitatives dont nous souhaitons faire état ne seront donc pas incluses dans la définition des concepts architecturaux sans perdre de vue l'exigence d'évolutivité de ces données.

Pour associer des données aux instances du modèle que figure telle ou telle scène, nous avons en conséquence choisi une approche rustique mais plus à même de supporter les évolutions nécessaires du modèle : l'adjonction d'un descripteur générique (adresses URL) en tête de la hiérarchie des concepts architecturaux. Cette propriété est renseignée à l'instanciation en fonction de l'état des connaissances sur l'instance soit par sa seule représentation soit par une requête http vers tel ou tel applicatif Internet[137].

Ce mécanisme constitue à première vue une non-réponse : le problème de la documentation des instances est renvoyé aux hypothétiques réponses à des requêtes http. Il n'y a pas de modélisation des connaissances relatives à la documentation de l'édifice mais simple report vers des applications autonomes. Pourtant, elle nous semble être une solution tenant à la fois du pragmatisme et de l'analyse des services attendus. En effet, la bonne question à se poser nous semble plutôt être : pourquoi traiter dans la modélisation du corpus architectural de problèmes dont d'autres peuvent mieux rendre compte? Autrement dit, pourquoi autoriser notre modèle à répondre à des questions qui relèvent d'autres disciplines ? L'élaboration d'une base documentaire ne répond pas aux mêmes exigences que celles qui ont dicté notre modélisation. Il nous semble par conséquent plus juste de considérer l'interfaçage de travaux complémentaires comme la bonne réponse au problème posé.
Nous proposons un formalisme de lien URL permettant d'attacher à chaque instance un ensemble non contraint de données qualitatives. Ce formalisme est expérimenté sur un seul groupe de concept, les entités architecturales. Il a été testé pour attacher à l'entité des requêtes vers la base documentaire SOL ou vers des pages Web standard.

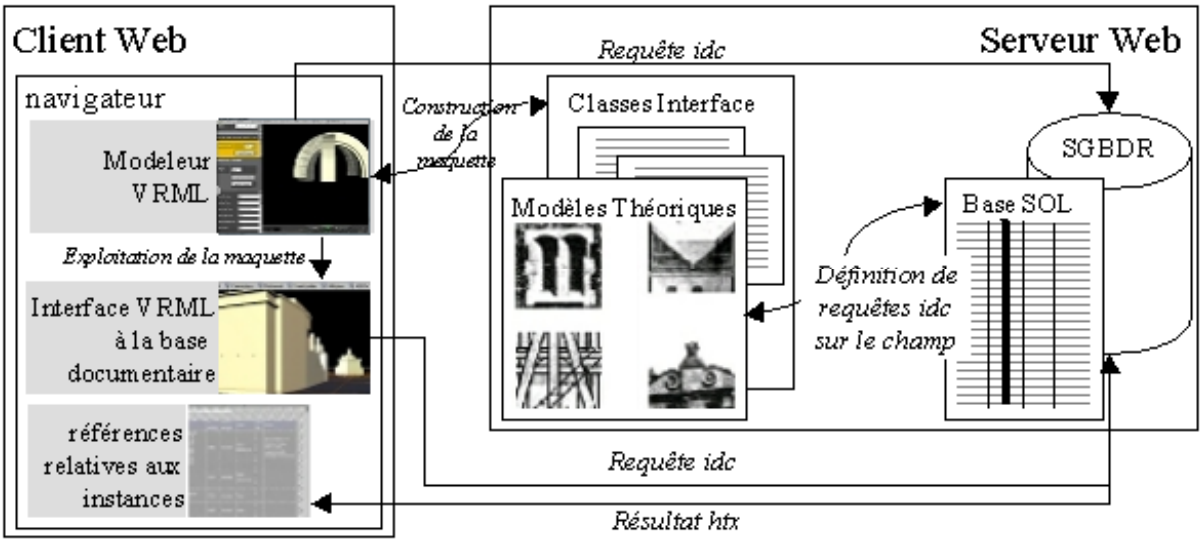



*Figure 67 : Architecture Client-Serveur dans l'interface de consultation de la base documentaire par la représentation tridimensionnelle d'instances du modèle (voir [Dudek et al, 1999a])*

La base documentaire contient en retour pour chaque enregistrement une liste de concepts architecturaux concernés. Précisons enfin que le mécanisme décrit ici s'applique aux données qualitatives relevant de classifications parallèles et non de données relatives au modèle.

[137] Dans l'état actuel de notre développement, ces requêtes http sont adressées au serveur du projet et font référence soit à des pages HTML spécifique (cas de Kramy Bogate) soit à des requêtes idc sur la base documentaire SOL.

Le schéma ci-après introduit les rapports que nous établissons entre différentes applications afin de lier instances du modèle, représentations et base documentaire. Les principales phase de travail sont les suivantes :

- Instanciations d'objets du modèle dans une scène représentée au format VRML dans l'application VALIDEUR.
- Affectation ou modification dans cette même application de requêtes sur la base documentaire par le biais de node Anchor, syntaxe standard du langage VRML.
- Insertion de scènes ponctuelles dans l'interface de consultation de la base documentaire. Les objets que contiennent ces scènes servent par défaut à interroger la base documentaire.

La représentation tridimensionnelle des instances peut donc servir à rechercher des documents dans la base. Elle n'est pour autant pas la seule méthode d'accéder à ces informations. En effet, ce mode d'accès ne vient pas se substituer aux méthodes habituelles de recherche de documents (par auteur, par date de publication, etc..) mais apporte une solution assez radicalement différente: l'utilisateur n'a pas besoins d'avoir connaissance d'informations partielles sur les documents recherchés mais seulement de s'intéresser à l'édifice en tout ou partie. Autrement dit, la recherche de documents par le biais de la représentation tridimensionnelle de l'édifice ne fait plus référence à la nature de ce qui est recherché (documents), mais à la nature de ce dont ces documents parlent (édifices et connaissances associées).

La représentation tridimensionnelle des instances sert d'interface de requête sur les données propres à chaque édifice. Par ailleurs, d'autres critères, plus généraux mais toujours attachés à un seul édifice, et non représentés dans notre modèle, sont présents dans le schéma de données. Dans la figure ci-contre par exemple, une liste de problématiques spécifiques à l'ancien marché aux draps peut servir de mode de requête sur la base.

Ce point rappelle la pauvreté de notre formalisme de réseau actuel puisque nous avons dit que l'interaction avec l'édifice dans son ensemble doit être possible au travers des réseaux. Ce point montre également qu'un travail de documentation des instances même relativement indépendamment de l'élaboration du modèle comme c'est le cas ici peut contribuer à interroger et enrichir ce modèle.


*Figure 68 : Interrogation de la base documentaire sur des problématiques spécifiques à un édifice*

# 5.IMPLÉMENTATIONS ET APPLICATIONS

## 5.1. IMPLEMENTATIONS

Dans cette section nous discuterons des implémentations réalisées à ce jour. Nous présenterons les hiérarchies de classes relatives à la formalisation des concepts évoqués aux chapitres précédents. Nous commencerons par une description des hiérarchies de classes implémentées pour représenter les concepts architecturaux dans les diverses applications développées. Nous verrons ensuite comment sont implémentées les classes dites outils présentées dans la section précédant immédiatement celle-ci. Il s'agira en définitive de cerner les limitations de l'implémentation actuelle et d'une certaine façon son degré de représentativité du modèle théorique.

### 5.1.1. Les hiérarchies de classes architecturales

Nous avons identifié au chapitre précédent les quatre groupes de concepts architecturaux de notre modèle et en avons établi les logiques d'organisation. Nous allons voir ici quels choix ont été faits dans leurs implémentations en introduisant d'abord leur représentation en JAVA puis le portage en PERL. Nous ne reviendrons pas ici sur l'implémentation C++, très incomplète par rapport à l'implémentation JAVA et abandonnée depuis plus de trois ans. Nous situerons par contre l'implémentation Perl, elle aussi un peu incomplète par rapport à l'implémentation JAVA, mais qui s'en distingue de façon notable sur certains points et nous sert par exemple dans les classes gérant les interfaces des outils de consultation de la base documentaire.

#### 5.1.1.i) Les classes JAVA

Nous allons commencer cette section en situant les principaux aspects relatifs à l'organisation des classes architecturales, puis nous attarderons plus longuement sur quelques détails d'implémentation concernant l'ensemble des classes architecturales avant de terminer par la présentation d'aspects liés à tel ou tel classe particulière.

*ORGANISATION*

La définition de classes JAVA est faite par le mot clé class suivi du nom de la classe puis d'une description de ses propriétés et méthodes mise entre parenthèses. Contrairement à C++, la définition de la classe et sa mise en œuvre sont jointes, définition placée dans un fichier ASCII qui prend le nom de la classe et l'extension ".java". Les classes travaillant conjointement sur un même domaine sont regroupées dans des packages: c'est le cas de l'API JAVA, mais cela peut également être le cas de packages utilisateur. Un package "architecture" rassemble les quatre groupes de concept dont nous discutons. Cela implique dans l'écriture de chaque définition de concept de préciser le package auquel il appartient en plaçant le mot clé package en tête de fichier:

Le nom du package doit correspondre à un emplacement sur le système de fichiers. Ce mécanisme de package est décrit comme permettant de regrouper les classes selon des liens sémantiques. Cette affirmation nous semble singulièrement tenir de l'auto-congratulation, mais il faut reconnaître qu'il nous permet en tout cas d'assurer une meilleure lisibilité de l'organisation des classes sur le système de fichiers. L'importation de classes d'un package vers d'autres classes se fait par le biais du mot clé import. La machine virtuelle JAVA (JVM) charge le code concerné en recherchant en référence à la variable d'environnement CLASSPATH localisant l'ensemble des fichiers ".class" (bytecode)[138].

[138] La syntaxe :

```
import moma.architecture.entite.*;
```

importe l'ensemble des fichiers situés en CLASSPATH/moma/architecture/entite dans le fichier courant.

Soit l'ensemble d'un package est chargé, soit des fichiers individuels. Mais de toute façon les fichiers ne sont pas tous chargés dès le début de l'exécution du programme, ils le sont dynamiquement en fonction du besoin.
Le choix du langage JAVA était pour nous justifié par notre souhait de travailler sur des applets. Ces programmes Java avaient donc vocation à circuler sur le Web, posant le problème de la lenteur des transferts de données. Pour limiter la taille des données à transférer, JAVA dispose d'un formalisme de compression de données appelé archive JAVA. Les fichiers d'archives (extension .jar) sont des fichiers zippés contenant les exécutables (.class) et les packages correspondants. Un seul fichier transite dès lors entre le serveur et son client.
Pourtant, cette solution a son revers : quand le code de la classe se trouve dans une archive, l'archive complète est transférée sur le réseau. Une fois l'archive transférée sur la machine cliente, la JVM n'en extrait que le fichier requis. Dans le cas où le code d'une classe n'est pas dans une archive, la classe est chargée lors de sa première utilisation, et non pas au commencement du programme.
Autrement dit, lorsqu'il y a transfert des classes sur le réseau, l'initialisation d'un programme JAVA avec archives est beaucoup plus lente que l'initialisation d'un programme JAVA sans archives. Inversement, l'exécution d'un programme JAVA avec archives est beaucoup plus rapide que l'exécution d'un programme JAVA sans archives. Le mot "beaucoup" utilisé ici rend bien peu compte des problèmes concrets rencontrés : en effet, une démarche d'analyse comparative serait à mener pour évaluer le seuil au-delà duquel une solution doit être préférée à l'autre. Ceci n'entre pas directement dans le cadre de ce travail, nous nous bornerons à signaler la nécessité de prendre ce point en compte dans l'évaluation de l'efficacité de l'applet. Il nous semble que c'est là que se trouve l'indice permettant de faire un choix. Dans le cadre de nos implémentations, le caractère itératif des tâches pilotées (mesurage ou élaboration d'hypothèses de restitution) nous a conduit à privilégier la création et le transfert d'archives JAVA.

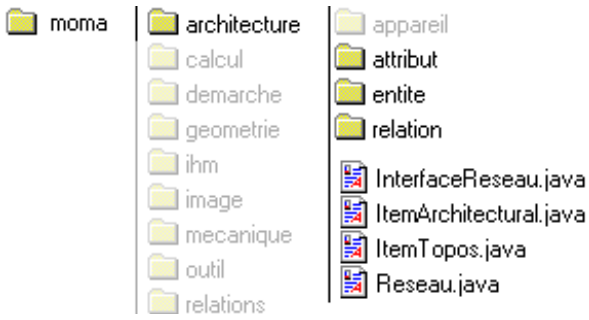



*Figure 69 : Le package "Architecture"*

Le package moma rassemble outre les classes architecturales regroupées dans le package architecture d'autres packages de classes relatives à un même domaine, comme par exemple les outils géométriques ou de calcul. La figure ci-contre récapitule cette organisation. A l'intérieur du package architecture se retrouvent nos quatre groupes, trois sous la forme de hiérarchies de classes et un, le réseau, sous la forme d'une seule classe puisque seul le concept de réseau neutre a été implémenté en JAVA. Deux classes abstraites factorisent les propriétés communes aux concepts architecturaux : les classes ItemTopos et ItemArchitectural.

La première est la tête de la hiérarchie des concepts architecturaux, elle factorise des éléments génériques de connaissance sur leur morphologie et sur leur représentation, comme par exemple :

- Un objet Référentiel.
- Une boîte englobante.
- Le niveau de détail de la représentation graphique.
- Un pointeur sur le propriétaire de l'Item.

Elle a en charge en particulier l'initialisation par défaut de toutes les propriétés liées à la représentation en VRML, ainsi que l'écriture des sections génériques de tête et de fin correspondantes.
La seconde, la classe ItemArchitectural, factorise une liste d'attributs morphologiques desquelles elle gère les accesseurs.
La classe Réseau représente un réseau neutre, c'est à dire un assemblage libre d'entités. Elle est caractérisée essentiellement par une liste d'entités architecturales, par une liste d'entites

mesurées, et par une liste de réseaux. Elle détient également les adresses des différents rapports à lire ou écrire (y compris le script LaDemarche fixant l'agencement des entités dans le réseau). Elle définit les méthodes de propagation d'appels de méthodes aux entités du réseau, ainsi que par exemple les méthodes d'évaluation et d'écriture des doublons point observateur / point observé utilisés pour gérer les points de vue par défaut dans les représentations en VRML 2.0.
Nous avons déjà discuté de l'organisation des autres classes en jeu et n'y revenons pas ici. Nous signalerons néanmoins que dans l'organisation actuelle des concepts architecturaux c'est la classe Attribut qui définit une liste d'EGO. Chaque attribut présent dans la définition d'une entité architecturale dispose donc de son propre jeu de mécanismes de mesurage. Le renseignement des propriétés de l'entité est donc laissé à la charge des attributs eux-mêmes.

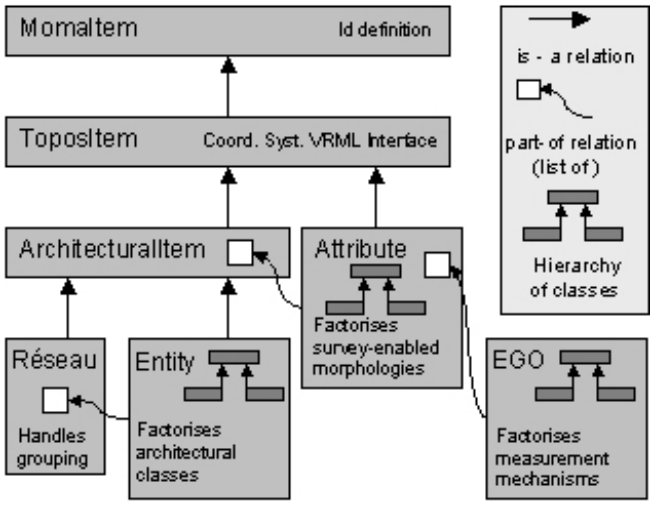



*Figure 70 : Hiérarchie des concepts architecturaux dans l'implémentation JAVA*

Afin d'alléger la lecture de ce document, nous renvoyons en **annexe 42** un ensemble de détails liés à l'implémentation des classes JAVA, annexe dans laquelle nous traiterons des points suivants :

1. Héritage multiple et polymorphisme
2. Aspects généraux de la définition des classes
   - -Constructeurs / destructeurs d'objets
   - -Collections d'éléments
   - -Duplication d'objets
   - -Exceptions
3. Aspects spécifiques à certaines classes
   - -Méthodes finales
   - -Gestion des entrées-sorties
   - -Variables de classes
   - -Variables finales (constantes)
   - -Surcharge
   - -La classe AttributDimensionnel
   - -Méthode de facettisation des profils

### 5.1.1.ii) Le portage PERL

Les classes architecturales ont été également modélisées sous la forme de hiérarchies PERL. Ce choix peu répandu à notre connaissance (voire peu orthodoxe) correspondait à un souci de compatibilité avec nos partenaires dans les terrains d'expérimentation polonais. Pourtant, au-delà de cette contrainte, nous pensons utile d'avoir fait face aux mêmes problèmes d'implémentation d'un modèle à partir de deux plateformes logicielles très différentes. Nous allons ici commencer par établir un rapide comparatif des deux approches afin de mieux mettre en perspective le portage PERL des classes architecturales.

#### *5.1.1.ii.1. Comparatif pratique*

Nous avons, dans la partie de ce document consacrée à l'état de l'art, déjà procédé à un comparatif des deux langages. Nous nous plaçons ici davantage du point de vue de la pratique de la programmation pour le Web. Nous pensons de cette façon préciser les avantages et inconvénients des deux approches au regard d'une problématique de développement spécifique, celle des applications client /serveur pour le Web.
Rappelons d'abord que JAVA est un vrai langage de programmation et une machine virtuelle permettant de faire à peu prés tout ce que l'on veut. La réutilisabilité de JAVA est forte, les

packages destinés à la production d'applets permettent de substituer au navigateur une interface graphique de l'application JAVA quasi-autonome. En gros, en JAVA on va recréer son propre pseudo-navigateur. Bien sûr, les objets à manipuler sont censés être faciles à utiliser et à réutiliser, ce travail n'est donc pas fait en vain. L'applet java, sous réserve de disposer des packages correspondants, va pouvoir gérer de la vidéo, inclure des images animées, inclure du son, etc… Les fichiers VRML seront eux affichés par le biais de plug-ins pilotables par des objets appropriés.
Pourtant, dans bien des cas, toutes ces fonctionnalités sont l'essence même des navigateurs Web. La programmation d'applets JAVA pose donc un premier problème, caricaturable par cette expression : pourquoi refaire ce qui existe déjà ? Par ailleurs, second écueil, un applet dispose une fois placé sur une page Web d'un espace fixe et définitif.
Il ne s'agit pas ici de discuter de ce que sait faire JAVA *en plus* de ces fonctionnalités d'un navigateur que nous utilisons, mais bien d'évaluer ce qu'un navigateur sait faire en standard afin de ne pas induire une pratique de programmation par redondance. Autrement dit, nous nous sommes posés la question suivante: que reste t'il de l'implémentation JAVA une fois les fonctionnalités d'interfaçage de l'application pour le Web remplacées par les fonctionnalités standard des navigateurs?
De façon figurée, dans la gestion des interfaces Web, là ou le programmeur JAVA sera tenté de prévoir jusqu'aux poignées de portes en étendant des packages de classes existants ou bâtis de toute pièce, le programmeur PERL sera lui poussé à en faire le moins possible en s'appuyant sur les fonctionnalités standards du navigateur.
De façon plus concrète, commençons par situer quelque différences essentielles :

- En PERL, l'interface Homme-Machine est laissée à la charge du navigateur en terme d'affichage à l'opposé des applets JAVA.
- En PERL il y a par essence distinction entre le modèle et ses représentations résultantes (ces dernières étant à la charge du navigateur et devant se conformer aux formats que celui-ci supporte). EN JAVA il y a distinction volontaire entre le modèle et ses représentations.
- En PERL, les entrées-sorties sont d'un maniement très simple, mais ne gèrent pas d'exceptions au sens JAVA.
- EN PERL, il n'y a pas de mécanisme d'échange d'objets comparables à ceux qu'autorise JAVA.

La formalisation de nos classes architecturales en PERL est centrée uniquement sur les problèmes liés au domaine d'application. C'est une programmation de la seule sémantique du domaine là où JAVA est une programmation universelle (du domaine comme de l'ensemble de l'application). Bien sûr, la modularisation des packages JAVA décharge le programmeur de la rédaction d'un grand nombre de classes. Il n'en reste pas moins qu'en JAVA tout est programmé, notions liées au domaine comme gestion des IHM, alors qu'en PERL seules les notions liées au domaine sont programmables[139]. Programmer un applet JAVA requiert de la part du programmeur non seulement une connaissance du domaine traité mais également sinon d'écrire du moins de connaître un ensemble de packages se rapportant à la gestion de l'affichage des applets et des protocoles de communication. Ces packages évoluent par ailleurs rapidement, impliquant une remise en cause à courte échéance de la partie non-sémantique du développement.
JAVA est sans conteste un langage plus riche que PERL, c'est une évidence que rappelle [Conway, 2000, pp449-455], et doit être privilégié dès que l'application à implémenter est tant soit peu complexe. Pour autant, JAVA n'est pas la seule alternative en matière de programmation d'applications sur le Web, et ne devrait a priori pas être considéré comme la solution à privilégier de façon systématique dans ce type de développement.
Notre expérience de portage des classes architecturales en PERL a comme conséquence, en dehors de répondre aux contraintes évoquées plus haut, de mieux évaluer la faisabilité d'un développement objet sur le Web sans JAVA. Nous ne prétendons pas avoir obtenu de

[139] Ceci introduit bien entendu à la fois plus d'économie et moins de liberté.

réponses globales mais pensons avoir démontré qu'il est possible de travailler à partir du même modèle objet sur des interfaces de bas niveau (navigateurs standard) sans perdre en route la sémantique du domaine traité.
Ce n'est qu'en réduisant l'implémentation JAVA à ce qui est propre à la formalisation du modèle que nous pouvons tenter de dresser un parallèle avec l'implémentation PERL. Quelques constats s'imposent donc :

- La création d'applets se substituant au navigateur introduit un niveau de complexité supérieur et une charge importante pour le programmeur, charge sans rapport avec le domaine traité.
- PERL est un langage interprété dont la lenteur par rapport à JAVA est évidente, et pénalisante dès que le nombre d'opérations que l'application réclame est important. Par contre, là où le téléchargement initial d'un applet JAVA peu s'avérer long[140], une application PERL ne pose pas ce genre de problème puisqu'il n'y a en PERL ni machine virtuelle ni échange de classes compilées mais seulement interprétation d'une série d'opérations relatives à la seule page chargée.
- PERL gère mal les opérations mathématiques.
- PERL ne bénéficie pas d'outils d'aide à la programmation (interfaces graphiques et / ou debuggeurs interactifs).
- La stabilité du langage PERL est supérieure à celle de JAVA, en particulier car il n'inclut pas de fonctionnalités d'interfaçage graphique.
- PERL gère de façon simple et robuste les entrées-sorties.
- Nous considérons qu'il est important d'évaluer ces alternatives en fonction de quatre aspects distincts d'un développement comme le nôtre, c'est à dire dans lequel s'ajoute à la formalisation d'un modèle une problématique "applicative" de visualisation de représentations hétérogènes de ce modèle :
- Formalisation du modèle.
- Traitements sur ce modèle.
- Gestion d'IHM pour le Web.
- Représentations d'instances du modèle.

Nous pensons qu'une analyse comparative des réponses des deux langages cités au regard de chacun de ces points peut mieux justifier l'adoption de l'un d'eux. Dans notre cas, s'y ajoutait le souhait de limiter le poids de la programmation d'applications au bénéfice de formalismes plus portables, plus proches des standards du Web. C'est dans cet esprit que notre choix de Perl doit être replacé.

*5.1.1.ii.2. La hiérarchie de classes PERL*

Les classes architecturales portées en PERL constituent un sous-ensemble du modèle architectural implémenté en JAVA. Si les classes dérivées de la classe Entité ou de la classe Attribut sont en nombre supérieur ou égal dans le portage PERL, la tête de la hiérarchie diffère cependant sensiblement.

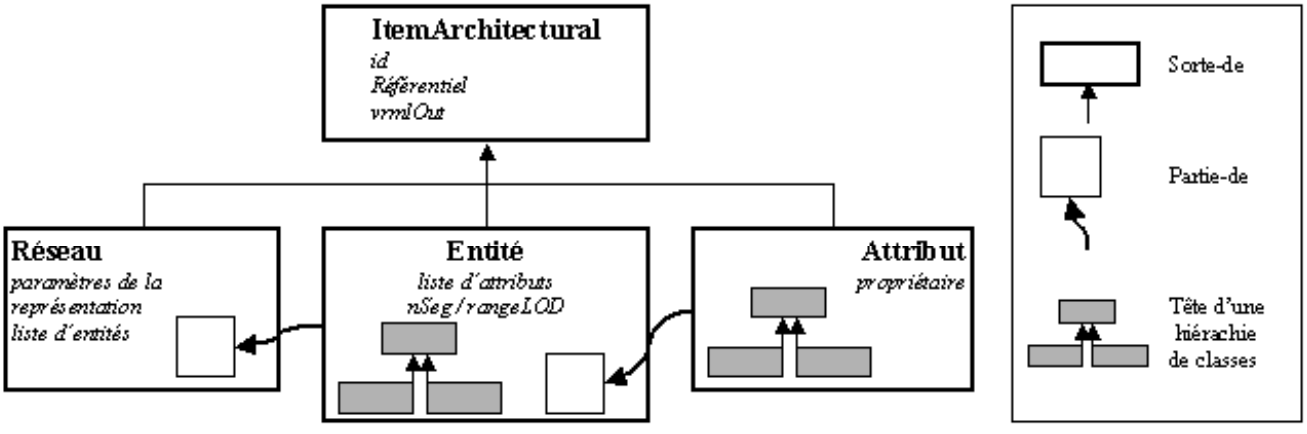



*Figure 71 : Tête de la hiérarchie des concepts architecturaux dans l'implémentation PERL*

En effet, la classe ItemTopos, tête de la hiérarchie JAVA, correspondait a une problématique d'abstraction des mécanismes de mesurage des objets. Or les objectifs poursuivis dans l'implémentation PERL sont uniquement centrés sur les questions de représentation et de gestion de données. Dans ce cadre, la classe ItemArchitectural redevient tête de la hiérarchie. Elle sert de mécanisme d'abstraction pour les propriétés communes aux entités, aux attributs et aux réseaux.
Nous introduisons en **annexe 43** quelques différences notables entre l'implémentation PERL et l'implémentation JAVA de la hiérarchie des classes architecturales.

### 5.1.2. Les classes outils

Nous allons dans cette section présenter trois hiérarchies de classes utilisées pour représenter des aspects importants du modèle, les classes dédiées aux calculs de mesurage, celles dédiées à la représentation du domaine de la géométrie et celles, enfin, en charge de l'interfaçage client/serveur. Nous n'en situerons pour rester brefs que les principes d'organisation et les domaines d'utilisation.

#### 5.1.2.i) Calcul

Le hiérarchie de classes dérivant d'ItemCalcul a été mise en place par P.Drap dans le cadre de l'implémentation JAVA. Cette hiérarchie représente des objets intervenant dans le processus de mesurage des objets architecturaux, à commencer par les EGO (Etres géométriques Optimaux), supports de la mesure dans les entités architecturales. Les classes définissant ces objets sont organisées de la façon suivante :

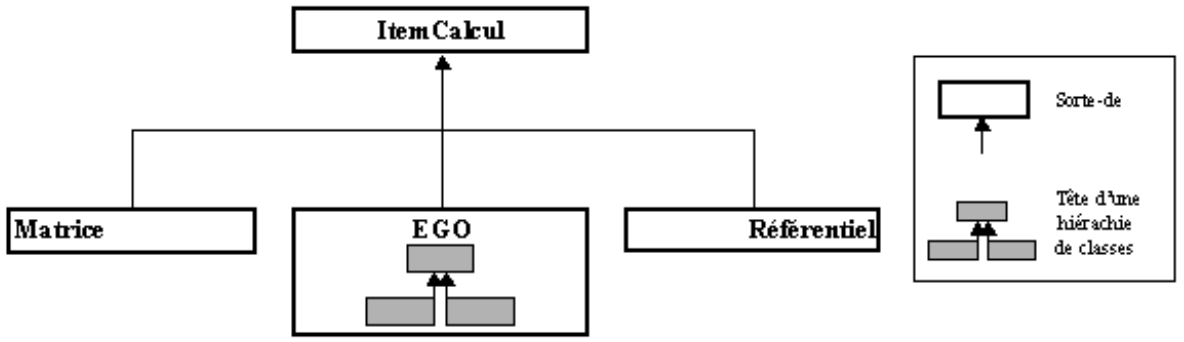



*Figure 72 : Hiérarchie des classes dérivant de ItemCalcul dans l'implémentation JAVA*

La classe EGO et ses dérivées représente des objets capables de retrouver les paramètres géométriques de primitives dites homologues dans un nuage de points. La classe Référentiel définit des objets caractérisés notamment par un point origine, des angles, une matrice de rotation et dotés de méthodes permettant par exemple de normer le référentiel ou d'exprimer un point dans un référentiel local. La classe Matrice définit un package de calculs matriciels.

#### 5.1.2.ii) Géométrie

La formalisation de notions de géométrie a déjà été. Nous n'en parlerons donc ici seulement pour indiquer quels choix d'organisation ont été faits pour représenter des objets géométriques intervenant dans le processus de mesurage ou dans la visualisation des objets architecturaux.
Quatre hiérarchies de classes dérivent de la classe ItemGéométrique: la hiérarchie des Points, celle des primitives géométriques, celle des OPPI (pour *Objet defini Par des Points Interpolés*) et celle des polyèdres. Ces deux dernières avaient dans l'implémentation JAVA été développées en modules autonomes par D.Gras et D.Merad. Dans l'implémentation PERL, seules les classes définissant les Points et les OPPI ont été traduites de JAVA. En effet, la notion de Point intervient à l'évidence un peu partout (à commencer par l'origine du

[140] Voir la section organisation de la hiérarchie de classes architecturales JAVA en début de chapitre.

référentiel de chaque Réseau ou Entité), et la notion d'OPPI a permis de mettre en œuvre le mécanisme de calcul de moulures (calcul des coordonnées x et y de tous les points à représenter entre les extrémités d'une moulure à partir des coordonnées x et y de ces extrémités et du type de la moulure).
Par contre, les classes définissant les primitives géométriques et les polyèdres n'étaient utilisées dans l'implémentation JAVA que dans le cadre du mesurage, elles n'ont donc pas été reprises dans l'implémentation PERL. Nous revenons ici rapidement sur les deux notions

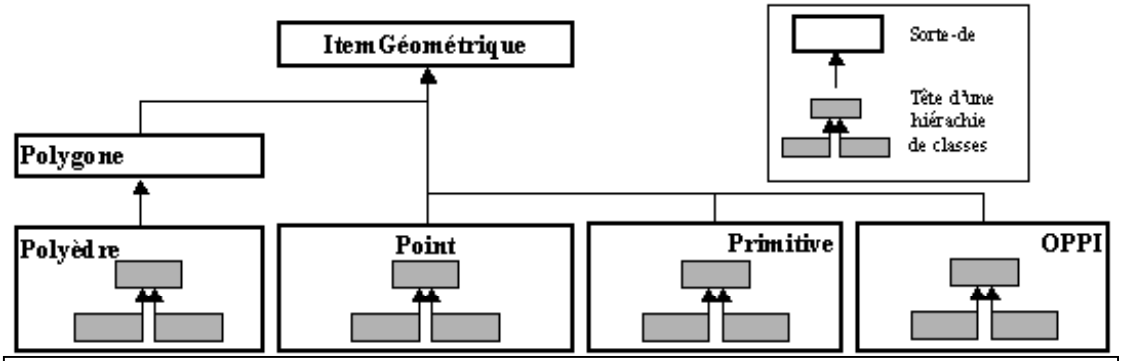



*Figure 73 : Hiérarchie de concepts représentant les notions de géométrie utilisées dans l'implémentation JAVA*

communes aux deux implémentations: Point et OPPI.
La classe Point représente des objets caractérisés par un triplet de réels et des constantes relatives à leur détermination dans le processus de mesurage. Ces objets sont dotés d'un ensemble de méthodes facilitant leur usage dans des classes tiers:

- des constructeurs de profils divers,
- des opérateurs sous la forme de méthodes dont le nom renvoie à la logique de surcharge des opérateurs de C++[141].
- des méthodes relatives au processus de mesurage comme la projection du point sur une primitive géométrique, le contrôle de la coplanarité de quatre points, etc…

Dans l'implémentation JAVA, plusieurs classes dérivant de Point sont formalisées pour représenter des notions spécifiques au domaine de la photogrammétrie, points de calage par exemple. Ces derniers sont incidemment dotés de méthodes d'affichage VRML au même titre que les objets architecturaux puisqu'ils sont déterminants du réseau créé : leur visualisation est considérée comme un moyen d'estimation visuelle du réseau d'entités mesurées. Classe Dérivée d'OPPI, la classe LigneBrisee définit des objets caractérisés par une liste de points dont la gestion doit être dynamique. C'est le cas de la ligne brisée attribut des objets Moulure[142].. Dès lors le renseignement de cet objet LigneBrisee par une méthode adéquate de l'objet Moulure va permettre à celle-ci de

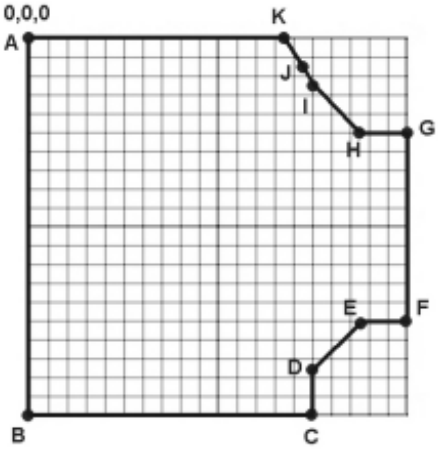



*Figure 74 : Suite de moulures de type plat définissant un profil*

[141]
```
public void moinsEgal (Pt p) {
x(x() - p.x ()) ;
y(y() - p.y ()) ;
z(z() - p.z ()) ;
}
```

[142] objets dont le constructeur s'écrit :

```
sub new {
…
$self->{ptDepart} =Point->new();
$self->{ptArrivee} = Point->new
$self->{type} = undef; ###  type de la moulure
$self->{ligne} = LigneBrisee->new();# liste des points
…
}
```

s'afficher sous la forme d'une courbe représentée par cette ligne brisée[143].
Connaissant les points de départ et d'arrivée de la moulure ainsi que son type, la ligneBrisee est renseignée en fonction d'un niveau de détail paramétrable puisque la liste de points qui la caractérise est modifiable.

### 5.1.2.iii) Interfaces

Dans l'implémentation PERL, la gestion de l'interfaçage client / serveur est à la charge d'un ensemble de cinq modules de fonctions et d'importances variés: *Connections*, *Codage*, *Interfaces*, *HtmElements* et enfin *Inputs*.
Ceci correspond à la diversité des tâches à effectuer :

- Assurer un contrôle et un suivi autonome des accès au serveur.
- Délivrer côté client des pages formatées correctement (c'est à dire lisibles pour un navigateur standard).
- Délivrer côté client un ensemble d'éléments permettant d'interroger l'utilisateur (scènes VRML, formulaires HTML).
- Traiter côté serveur les sélections et les actions utilisateur et ré-informer le client.

Nous allons dans un premier temps préciser le contenu de chacun de ces modules puis verrons comment ils interagissent dans la gestion effective des interfaces.

#### *5.1.2.iii.1. Les cinq modules développés*

Nous aborderons ici en premier lieu les modules dédiés à l'écriture des documents délivrés au client. Nous distinguons ici deux catégories :

- Les classes liées au langage interprété par le navigateur (HTML) : hiérarchies *Inputs* et *HtmElements*.
- Les classes liées à la représentation du modèle : hiérarchie *Interface*.

Nous présenterons ensuite les classes dédiées au suivi des connections utilisateur, à savoir les hiérarchies *Codage* et *Connections*. Précisons dès à présent que le décodage des données saisies par le client est assuré par un module indépendant, *Decodage.pm*, qui reprend pour l'essentiel le *cgi-lib.pl* de [Brenner et al, 1996].

*LA HIERARCHIE INPUTS*

Un formulaire standard HTML comprend des entrées (inputs) standard et seulement cela. Autrement dit, il est possible de prévoir de façon générique sous quelles formes des interactions utilisateur peuvent être autorisées. La hiérarchie Inputs correspond à cette approche : définir des objets gérant l'interaction utilisateur au travers de formulaires utilisateur. Ces objets doivent être aptes à écrire en HTML standard des entrées de tous types.
Par ailleurs, il est rare qu'une seule question soit posée à l'utilisateur. La classe Inputs définit donc des objets caractérisés par :

- Une liste d'entrées (nom, valeur, type, etc.., correspondant aux champs des balises HTML en jeu).

[143]
```
sub renseigneLigne {
  my $self = shift;
  my @tabDeop= (@_[0], @_[1],@_[2]);
  my @tabArr= (@_[3], @_[4],@_[5]);
  $self->ptDepart->Pt(@tabDeop);
  $self->ptArrivee->Pt(@tabArr);
  $self->type(@_[6]);
  my $nbPtsATrouver= (@_[7]/2)-1;
  …
  …
  $self->ligne->addElement($self->ptArrivee()->xyz());
  $self->ligne->nbElements(6+($nbPtsATrouver *3));
  $self->ligne->nbPoints(2+$nbPtsATrouver);
}
```

- Une liste de descripteurs textuels affichés (ou non) en regard des entrées.
- Une liste d'actions Javascript éventuelles.

L'objet *Inputs* ne détermine pas une logique d'affichage mais un contenu : il ne dispose d'aucune méthode d'écriture. Cette écriture effective des balises correspondant aux entrées est gérée par un objet HtmModuleForm décrit plus loin. En effet, il nous a semblé nécessaire de séparer ce qui représente la liste de questions posées à l'utilisateur de la manière effective dont ces questions sont affichées sur son navigateur. Ces objets n'ont en définitive pour méthodes propres que celles gérant l'accès à leurs attributs : ils se contentent de rassembler les questions adressées à l'utilisateur.
L'objet *CharsetInputs*, dérivé du précédent, a pour caractéristique additionnelle un jeu de caractères spécifique, utilisé dans le cas des formulaires en langue polonaise (Latin 2). Ils disposent de méthodes de traduction des caractères accentués Latin 2 en codes HTML correspondants.

*LA HIERARCHIE HTMELEMENTS*
L'affichage des formulaires est géré par une hiérarchie de classes représentant des objets chargés de l'écriture de balises HTML valides relatives à des formulaires permettant à l'utilisateur de saisir des données. Ces objets réalisent donc l'écriture effective du code HTML, écriture commandée par l'objet *Interface* décrit plus loin.

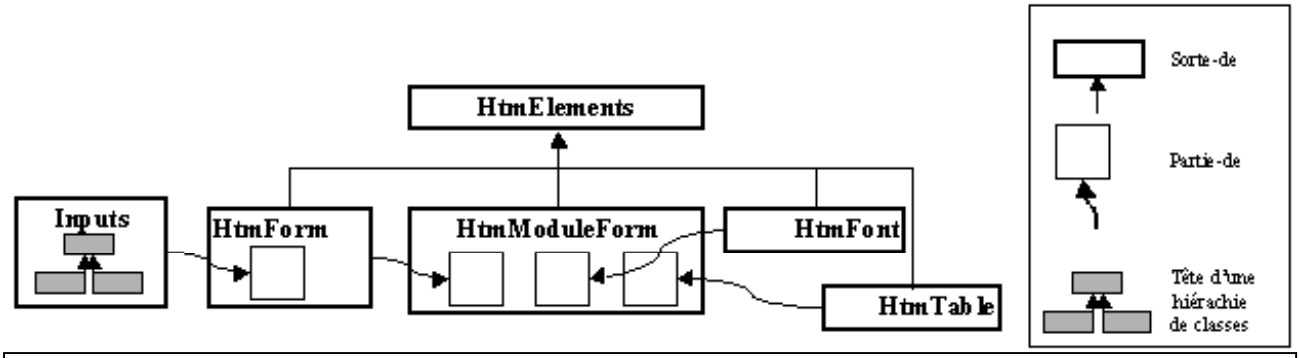



*Figure 75 : Tête de la hiérarchie des classes dédiées à l'interfaçage Web (implémentation PERL)*

La classe *HtmModuleForm* définit des objets aptes à afficher sous forme de tableau du langage HTML (défini génériquement par la classe *HtmTable*) un formulaire (défini génériquement par la classe *HtmForm*) contenant une liste d'entrées (définie génériquement par la classe *Inputs*) avec une police de caractère donnée (définie génériquement par la classe *HtmFont*).
Ce mécanisme très simple permet de gérer de façon indépendante la liste d'attributs d'une entité à faire modifier par l'utilisateur et son affichage effectif comme un formulaire standard du langage HTML. Les instances de la classe HtmModuleForm disposent de plusieurs méthodes d'écriture qui parcourent la liste d'entrées de *Inputs* et formatent un tableau HTML correspondant.
Du point de vue de l'entité architecturale ou du réseau, l'écriture d'un fichier HTML présentant à l'utilisateur côté client l'ensemble de propriétés qu'il est censé modifié se résume à une instruction simple commandée par son *Interface*[144]. ette solution nous semble révéler deux points importants de notre formalisme d'interfaçage :

- L'écriture effective du code interprété par le navigateur est encapsulée dans une méthode (allant dans le sens d'une meilleure évolutivité de l'interface).
- Le résultat se conforme au standard HTML en cours (allant dans le sens d'une meilleure portabilité).

*LA HIERARCHIE INTERFACE*

[144] `$self->form->writeLeftAlignedModuleForm($fic);`

La classes *Interface* et ses dérivées délivrent côté client une représentation du modèle apte à interroger l'utilisateur. Elles assurent côté serveur le traitement de sélections et/ou ordres utilisateur. La classe *Interface* définit des objets capables d'écrire au vol un fichier d'interfaçage HTML délivré au client à partir d'un jeu de données évolutives :

- Des sélections utilisateur (création ou mises à jour d'instances du modèle).
- Des informations génériques sur la présentation du modèle (correspondant aux balises d'en-tête des fichiers HTML). Ces informations, stockées indépendamment des classes *Interface*, représentent la configuration graphique des interfaces. Elle sont partagées par l'ensemble des instances du modèle dans un souci de facilité de maintenance mais déportées hors de la définition des classes dans un souci de plus grande évolutivité.

Les objets ainsi définis sont utilisés comme propriétés d'éléments du modèle pour lesquels ils réalisent en fait un formatage au standard HTML, autrement dit une représentation du modèle.
La classe *InterfaceEntité*, dérivée de la précédente, a pour caractéristique additionnelle un objet de type *HtmModuleForm*. Chaque Entité dispose donc d'une *Interface* apte à délivrer côté client une représentation qui, sous l'aspect d'un formulaire dont les entrées correspondent aux attributs morphologiques de l'entité, permet à l'utilisateur d'agir sur l'instance. La classe *InterfaceRéseau* est bâtie sur le même principe, mais dispose cette fois de caractéristiques additionnelles permettant à l'utilisateur d'accéder à la liste d'instances du réseau et à la liste d'entités disponibles au moment de l'appel pour instanciation. Enfin, une classe *InterfaceValideur* est chargée de délivrer une interface par défaut à l'initialisation d'une connection pour l'outil VALIDEUR décrit plus loin.

*LA HIERARCHIE CODAGE*

Parmi les classes dédiées au suivi des connections utilisateur, la hiérarchie de classes dérivées de *Codage* a pour objectif de représenter des objets capables de réaliser l'authentification de codes entrés par l'utilisateur côté client. La première application de ce mécanisme est bien entendu l'identification du requérant, mais nous allons voir que son rôle ne s'arrête pas là. La classe *Codage* définit des objets ayant pour caractéristiques deux tableaux (identifiants et codes associés) et une source de données. L'identification d'une entrée utilisateur correspond alors à la recherche de son occurrence dans le tableau concerné, tableau initialisé par la lecture de la source de données. La classe *PassCodage*, dérivée de la précédente, correspond à une utilisation dans le cadre de l'identification de l'utilisateur : elle permet de garder une trace personnalisée des tentatives de connections (source de donnée utilisateurs et informations additionnelles comme leurs adresses Mèl).
La classe *AddCodage*, dérivée de *Codage*, définit des objets capables de générer un code lexical à partir de la requête et d'accéder en écriture aux sources de données. Ces objets sont utilisés dans le cadre de l'outil SOL pour générer de nouveaux codes relatifs aux groupes thématiques des informations détenues dans la base de données. Autrement dit, lorsque l'utilisateur met à jour la base en ajoutant non seulement un enregistrement mais aussi un nouveau critère descriptif, ce dernier est codé puis ajouté à la source de données par l'intermédiaire d'un objet *AddCodage*[145].
Au delà d'une authentification autonome des connections, la hiérarchie de classes dérivées de Codage intervient donc partout où se pose la question suivante : comment traiter en amont du SGBD des entrées utilisateur pour évaluation/modification sur des sources de données gérant l'interfaçage.

[145] Dans la suite d'instructions ci-dessous, un nouveau code est ajouté à ceux concernant le groupe thématique "*t*" dont les sources de données sont les fichiers $fic et $ficadd:

```
my $codeTruc = AddCodage->new();
$codeTruc ->fic($fic);
$codeTruc ->ficAdd($ficad);
$codeTruc ->root("t");
$codeTruc->makeCode("test2");
```

*LA HIERARCHIE CONNECTIONS*

Cette hiérarchie de classes est chargée d'assurer le suivi des connections entre les clients et le serveur. Elle est constituée de quatre classes liées par des relations d'héritage successives correspondant à des niveaux de contrôle et de rapport sur la connection allant en se renforçant. La tête de cette hiérarchie (*Connection*) est une classe définissant des objets capables de retrouver les informations de base sur la connection: date, numéro IP du requérant et URL requérant et demandé. La classe dérivée de *Connection*, *IndexedConnection*, définit des objets capables de rapporter sur une source de données générale au système ces informations ainsi que de maintenir un suivi quantitatif des connections. De cette classe dérive la classe *ProtectedConnection* qui introduit la notion d'authentification du requérant par le biais d'un objet de type *PassCodage*. Enfin, la classe *RegisteredConnection* définit des objets capables de rapporter sur des sources de données utilisateur les informations détenues par l'objet de type PassCodage dont ils héritent. Le schéma ci-dessous récapitule les types d'information et de rapport en jeu selon la position dans l'arbre d'héritage des classes dérivées de Connection.

Cette solution permet d'assurer un suivi autonome des connections par rapport à d'éventuels

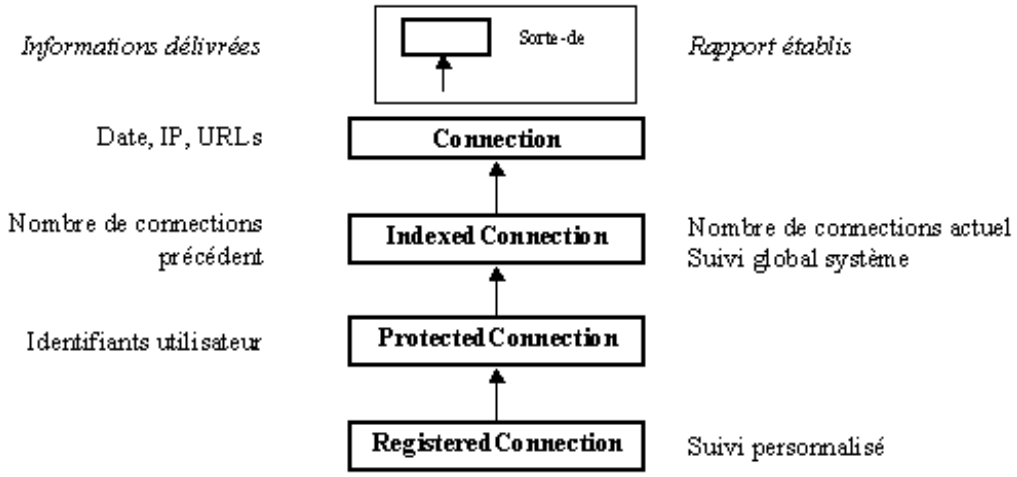



*Figure 76 : La hiérarchie de classes dérivées de Connection, informations lues par le objets et type de rapports générés (implémentation PERL).*

changements de plateforme matérielle du système. Elle permet en outre de maîtriser le format et le type d'informations contenues dans les rapports aussi bien au niveau suivi global qu'au niveau suivi utilisateur.

### *5.1.2.iii.2. Les tâches réalisées*

Pour une implémentation censée déporter les tâches d'interfaçage sur le navigateur, on ne peut pas dire que le résultat semble convaincant puisque cinq modules dédiés à cet interfaçage ont du être développés. Pourtant il faut prendre en compte deux aspects importants avant d'aboutir à une telle conclusion :

- En premier lieu, il est bien évident que le navigateur est effectivement en charge de la seule interprétation du contenu HTML que lui transmettent les programmes CGI. La rédaction de ce contenu incombe naturellement au programme et si le formatage des fichiers a donné lieu au développement de modules dédiés, c'est dans une perspective de meilleure réutilisation qui ne remet pas en cause le bénéfice de confier l'affichage effectif au navigateur.
- En second lieu, ces modules sont d'une grande simplicité à la fois en terme de syntaxe et en terme d'organisation des classes. Leur évolution est ainsi à la portée de programmeurs peu expérimentés, souci constant dans le cadre du projet pour lequel ces modules ont été développés.

Nous pensons en conséquence que loin de remettre en cause l'objectif de l'implémentation PERL, ces modules s'adaptent à la diversité des tâches à réaliser dans une logique de

meilleure réutilisabilité. Leur utilisation dans le cadre de la manipulation du modèle architectural s'effectue à deux niveaux: celui de l'application (classes dérivées de *Connection* et de *Codage*) et celui de la représentation des concepts architecturaux (classes *Interface*, *HtmElements* et *Inputs*). Les tâches que réalisent ces classes sont les suivantes :

- Suivi des connections.
- Ecriture des fichiers HTML rapportant l'état des objets architecturaux.
- Transmission et organisation de ces fichiers côté client.

Le suivi des connections est assuré au niveau des applications (SOL et VALIDEUR) sans rapport au modèle. Ceci ne constitue qu'un choix intermédiaire, la constitution d'une banque de maquettes numériques s'appuyant sur le modèle architectural pouvant nécessiter d'introduire dans la représentation même des instances un identifiant explicite. Nous considérons cependant qu'il s'agit là d'une perspective et n'avons pas souhaité livrer un formalisme définitif de cette question.

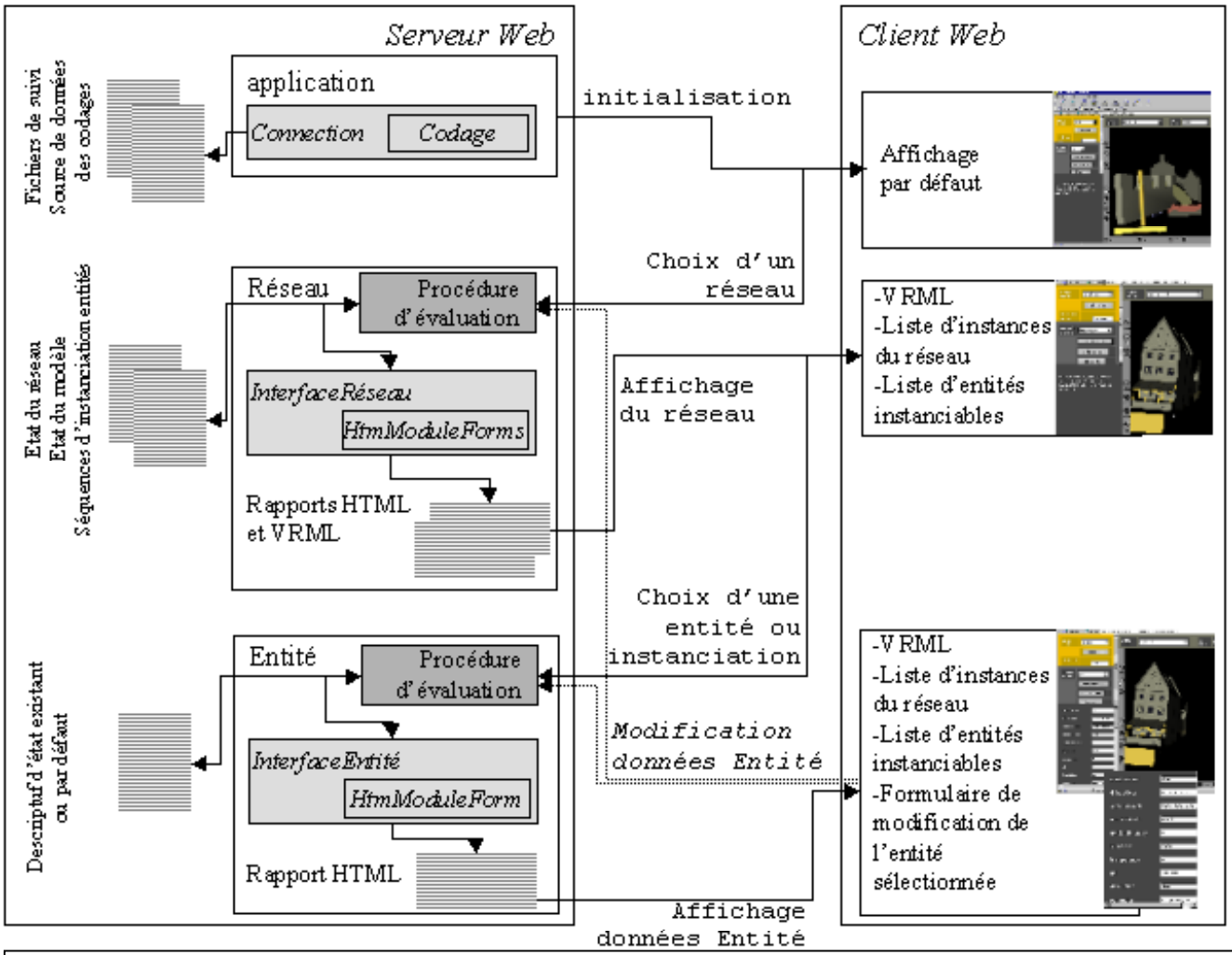



*Figure 77 : Utilisation des modules d'interfaçage dans l'architecture Client-Serveur Web choisie*

En effet, nous devrons distinguer deux notions différentes : l'auteur de l'hypothèse, qu'il peut être utile de voir figurer dans la représentation de la maquette, et l'auteur du modèle sous-jacent qu'il est impératif de pouvoir identifier si ce modèle se veut évolutif.

Dans le schéma ci-avant, qui récapitule la façon dont les modules d'interfaçage s'articulent au modèle architectural et aux applications dans le cas de l'outil VALIDEUR, nous détaillons un peu plus les rôles respectifs des modules et leurs séquences d'utilisation. Nous précisons également à quelles sources de données ces modules sont susceptibles d'avoir recours soit directement (cas des connections) soit indirectement au travers des classes Entité et Réseau.

### 5.1.3. Le langage de description des scènes

Le langage de description des scènes utilisé dans l'implémentation JAVA, nommé LaDemarche, a été développé par P.Drap et utilisé dans plusieurs expérimentations sur le modèle architectural dont nous avons déjà fait mention (voir par exemple [Drap et al, 1999a] et [Drap et al, 1999b]). Nous en rappelons le principe en **annexe 44** et en signalons un aspect important dans le cadre du modèle architectural, le raffinement d'instances. Nous renvoyons par ailleurs à [Drap, 1997] pour une présentation plus détaillée de ce développement.

### 5.1.4. Les formats d'expression du modèle

Dans l'une comme l'autre des implémentations citées il est fait usage de rapports textuels et graphiques relatifs à une instance de la classe Réseau et à ses entités architecturales. Nous allons ici brièvement établir une liste de ces formats d'expression. Par rapports textuels nous ferons référence à des fichiers texte ASCII non interprétés côté clients et par rapports graphiques des fichiers texte ASCII interprétés côté client ou côté serveur.

#### 5.1.4.i) Rapports textuels

Dans l'implémentation JAVA, la séquence d'instanciation et le descriptif d'état des instances d'objets architecturaux contenus dans le réseau sont décrits dans le script LaDemarche relatif à ce réseau. L'écriture du script (ou sa ré-écriture en cas de modification) est du ressort du réseau, les entités ne disposant pas de rapports textuels autonomes. Par ailleurs, la table de détermination des rapports par défaut entre propriétés morphologiques des entités est également lue par le réseau, elle peut être choisie en fonction par exemple d'une époque ou d'un auteur. Ces deux rapports textuels sont délivrables au client, affichés par l'applet, et modifiables directement par l'utilisateur qui a accès au texte lui-même.
Dans l'implémentation PERL, la séquence d'instanciation et le descriptif d'état des entités sont des rapports textuels autonomes concaténés par le réseau. La séquence d'instanciation de celui-ci comprend à la fois son instanciation propre et celle des entités qui le composent. Le formalisme de relations expérimenté n'est pas persistant. Chaque instanciation d'une nouvelle entité au sein du réseau est faite sur la base d'un descriptif d'état par défaut qui est local au réseau, il réalise une initialisation de l'entité avec des valeurs par défaut locales au réseau, choisies par l'utilisateur. Ces rapports textuels ne sont pas délivrables au client, ils sont ainsi qu'indiqué à la section précédente formatés en langage HTML standard pour affichage dans un navigateur. Ils sont modifiés indirectement par l'utilisateur qui n'a pas accès au texte lui-même mais à des formulaires décodés par le programme CGI avant réécriture des rapports.

#### 5.1.4.ii) Rapports graphiques

Dans l'implémentation JAVA, les rapports graphiques font non seulement référence aux fichiers ASCII interprétés par le navigateur ou ses plug-ins (HTML et VRML) mais également aux fichiers MicroStation générés par le processus de mesurage[146].
Dans l'implémentation PERL, les rapports graphiques sont des fichiers HTML et VRML résultant de l'évaluation du réseau et affichés aux bons soins du navigateur. Nous avons déjà évoqué la méthode d'écriture des fichiers HTML dans la section précédente, et celle des fichiers VRML dans la section "*Méthodes de visualisation*"[147].
Au travers des solutions adoptées pour l'écriture de rapports textuels et graphiques on retrouve bien les différences qui sépare un langage de programmation universelle, JAVA, et un langage de programmation conçu au départ pour manipuler des chaînes de caractères, PERL.

[146] Pour ces derniers, je renvoie à [Drap et al, 2000]. Pour les premiers, les rapports sont écrits sous la forme de flux de données redirigés vers l'applet et affichés par lui.
[147] L'écriture de fichiers de rapports au format POV (format ASCII également) est en cours de développement. L'objectif est toutefois ici légèrement différent : le fichier crée n'a pas vocation à être distribué au client pour interprétation mais doit servir à un calcul d'image sur le moteur de rendu POV-Raytracer. Si l'écriture de ce troisième rapport graphique se fera côté serveur, deux solutions sont envisageables quand au calcul d'image lui-même : exécution côté serveur (avec les restrictions qui s'imposent) ou transfert au client.

### 5.1.5. Gestion des données attachées au modèle

Dans le cadre de la documentation du modèle et des édifices traités, nous avons souhaité mettre en œuvre un Système de Gestion de Bases de Données relationnel (SGBDR) interfacé pour le Web. Il est important de préciser dès à présent que le rôle dévolu au SGBD n'est pas d'assurer la persistance du modèle mais de données rattachées au modèle et à ses instances. Ces données, rappelons le, sont des données qualitatives (bibliographie, repères historiques ou stylistiques, données sur l'auteur ou sur la datation) associées aux instances du modèle que figure telle ou telle scène. Ce sont aussi des données de ce même type associées cette fois-ci aux concepts que le modèle représente. Les bases construites n'ont donc qu'un rapport de citation avec le modèle: elle n'en reprenne en rien l'organisation objet. Des enregistrements de ces bases documentent tel ou tel instance ou concept mais n'ont pas vocation à figurer les liens qu'entretiennent les instances ou concepts en terme de modélisation du domaine. Les bases en question sont dédiées à une utilisation autonome par rapport au modèle, utilisation centrée sur une approche de "*documentaliste spécialisé*". En nous plaçant dans cette perspective, nous considérons de fait que si le modèle doit pouvoir interroger ces bases, il n'est pas pour autant justifié de les concevoir à l'image du modèle. En effet, nous pensons que le travail de documentation d'une maquette numérique ne peut pas systématiquement impliquer la construction d'une base documentaire spécifique : nous nous plaçons dans une situation où le modèle et ses instances sont *informés* par une base documentaire existante. Il est donc justifié de parler de données attachées au modèle puisqu'elles peuvent être organisées et délivrées en toute indépendance par rapport à celui ci.
Fermons donc la parenthèse de façon claire : l'utilisation d'un SGBDR dans le cadre du projet n'est pas liée à un souci de persistance du modèle. Il est lié à l'élaboration d'une base documentaire à laquelle les concepts et instances du modèle doivent pouvoir accéder.
Une fois ce point éclairci, nous espérons que l'utilisation dans ce projet d'un SGBDR, de préférence à un Systèmes de Gestion de Bases de Données Objets (SGBDO), trouve une première explication. Précisons par ailleurs que l'utilisation du second peut toujours faire partie des perspectives possibles du projet.
Nous allons maintenant pouvoir discuter en détail du travail fait autour des données attachées au modèle. En quels termes le problème se pose t'il ?
Nous avons d'un côté un modèle architectural dans lequel sont isolés et organisés des concepts relatifs au domaine, concepts servant par exemple à l'élaboration d'hypothèses de restitution sur des édifices détruits en partie ou en totalité. De l'autre côté, nous avons des bases documentaires répertoriant selon leur logique propre des références citant ces mêmes concepts, soit pour les définir (définir le terme sous lequel est reconnu le concept) soit pour en signaler des instances sur tel ou tel édifice. Les références contenues identifient des documents de types variés, ouvrages relatifs à un édifice, à un groupe d'édifices, à une technique de construction, à une affiliation stylistique, ou bien encore illustrations, documents graphiques, compte rendus de travaux de restauration, etc…
Deux points essentiels sont donc à relever :

- Une grande hétérogénéité des données à référencer.
- La nécessité d'inscrire dans leurs références la notion de modèle architectural telle que nous l'avons définie.

Le référencement des données attachées au modèle est contraint à deux niveaux, il doit pouvoir être apte à délivrer des documents hétérogènes, il doit inclure un ensemble de descripteurs relatifs au modèle architectural à partir duquel nous souhaitons interroger la base. Le choix que nous avons opéré s'appuie sur une architecture logicielle basique, celle d'un SGBDR interfacé pour le Web. Nous distinguons deux bases : la base de définition et de référencement des concepts, la base documentaire des instances. Peu de différences existent entre les implémentations liées à ces deux bases, nous en parlerons donc de façon générale. Une base est utilisée comme intégrateur entre deux types de documents servis sur le Web: des interfaces de requêtes et des URL référencés dans la base. Ces derniers supportent le

caractère hétérogène des documents référencés et peuvent à leur tour être structurés par le biais de formalismes comme XML.

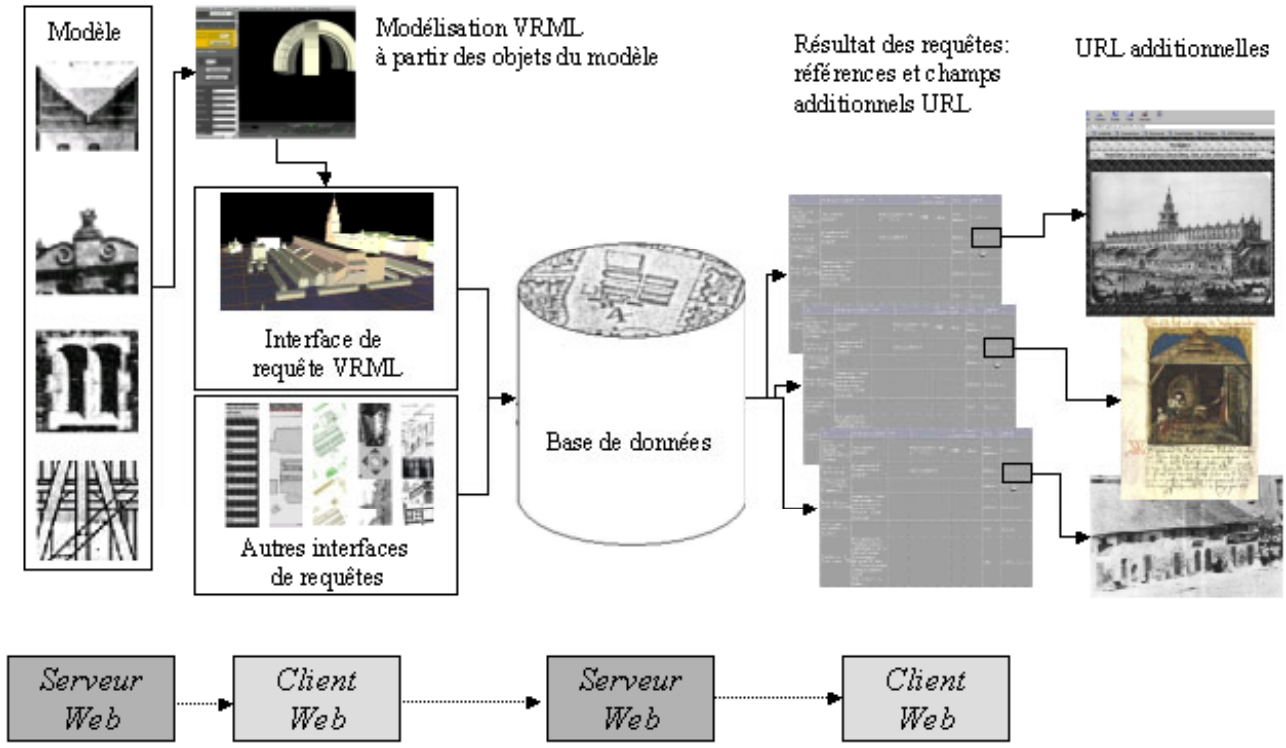



*Figure 78 : Schéma synoptique illustrant le rapport modèle - maquette - système de gestion de base de données - interfaçage Web dans le cas de la base documentaire SOL*

Le modèle est pris en compte à deux niveaux puisqu'il sert à élaborer des interfaces de requêtes graphiques (format VRML) mais aussi est référencé dans les champs concernés des tables de la base. Dans cette section nous allons présenter deux aspects de cette implémentation : les caractéristiques principales du SGBDR utilisé puis le mode d'interfaçage pour le Web qu'il autorise.

*LE SGBD UTILISÉ*

L'implémentation proposée fait appel aux éléments suivants :

- Un Système de Gestion de Base de Données Relationnel (Microsoft SQL Server).
- Un Serveur Web (Microsoft IIS).
- Une interface pour Navigateurs (CGI-Perl / Javascript).

Ce développement est conduit sur un ordinateur de type PC sur lequel a été installé le système d'exploitation Microsoft Windows NT livré avec le serveur Web IIS (Internet Information Server). Le pilote ODBC SQL Server autorise l'interfaçage du SGBD SQL Server[148] avec le système d'exploitation et son serveur Web (IIS) .

Deux modèles d'accès à des données sont possibles : le modèle *pull* dans lequel la requête passée au SGBD est construite dans une page Web puis génère en résultat une autre page Web, et le modèle *push* dans lequel le SGBD se charge de créer une page Web correspondant à une requête prédéfinie. Dans notre cas nous utilisons pratiquement exclusivement des

[148] Les fonctionnalités de SQLServer satisfont aux normes SQL-92 de l'American National Standards Institute (ANSI) et aux Federal Information Processing Standards (FIPS 127-2) établies par le National Institute of Standards and Technology (NIST). SQL Server était à l'origine un produit de SYBASE tournant sur VMS et Unix. En 1992, Microsoft propose sa version du produit qui dès lors ne tourne plus que sous le système d'exploitation Windows NT. La version 6.5 de SQLServer avec laquelle notre travail a débuté propose une intégration aux standards SQL et une ouverture vers les systèmes hétérogènes, notamment réplication vers des bases de données compatibles ODBC.

interfaces de requêtes dynamiques. Le développement tourne sur un PC de performances limitées déclaré dans un domaine NT spécifique.
Comme on l'a vu, les pages Web constituant l'interface sont créées contextuellement aussi bien en phase d'indexation qu'en phase de requêtes par le biais de scripts PERL et incluent quelquefois des scripts Javascript. Il y a de fait à la fois pré-traitement des requêtes (problèmes d'accentuations, formatage de champs spécifiques, création de fichiers IDC et HTX contextuels) et post - traitement des réponses (problèmes d'accentuations, etc..).
Dans le premier cas, l'ensemble des tâches d'interfaçage est modularisé sous forme de modules objet Perl.
Dans le second, des procédures Javascript sont intégrées aux pages délivrées en réponse à une requête par le SGBD. Sur la machine concernée par ce développement sont installés le serveur IIS, SQL server version 6.5 ou 7.0 et Perl IIS. Les fichiers physiques correspondants sont ici de plusieurs types :

- L'espace disque (unité de base de données contenant bases et journaux de transactions) que gère SQL server lors de la création d'une nouvelle base. La taille de ces fichiers est choisie lors de l'initialisation de la base, elle est réajustée, si besoin est, ultérieurement.
- Les fichiers de script Perl appelés par le navigateur soit en phase de requête sur la base soit en phase d'indexation d'un nouveau document.
- Les fichiers de requête aux formats idc et htx (formats d'interface SQL Server) écrits dynamiquement lors de la formulation des requêtes par les scripts Perl.
- Les fichiers au format texte contenant l'intitulé affiché dans l'interface et son codage pour chaque champ ou ceux répertoriant les comptes d'accès restreint à la phase d'indexation.
- Des fichiers au format HTML notés "URL additionnelles" dans le schéma présenté à la page précédente (figure 104) et qui sont de fait totalement indépendants du reste.

SQLServer propose trois modes de sécurité :

- Sécurité standard SQL Server dans laquelle coexistent un compte SQL et un compte NTServer, mode dit sans confiance.
- Sécurité intégrée dans laquelle l'utilisateur NT est couplé à SQLServer. Dans ce cas, l'identification NT de l'utilisateur suffit puisque SQLServer va chercher les informations nécessaires dans la base de comptes NT et les compare à celles de son gestionnaire de sécurité. Ce mode est dit avec confiance.
- Sécurité mixte : dans ce cas les deux modes sont possibles.

SQL Server fonctionne dans notre cas en mode de sécurité intégré. SQL Server va donc récupérer les informations de la base de comptes de Windows NT. Dans le cadre d'une connexion Internet le compte IIS iusr_nomServeur est mis en correspondance avec l'accès InvitéNT.
Le chargement initial de la base peut se faire par l'utilitaire BCP (Bulk Copy Program) qui permet d'importer des données à partir de tableaux de données aux formats standards.
SQL Server propose un ensemble d'outils de gestion de la base sur lesquels notre développement ne s'appuie pas à proprement parler, et sur lesquels nous ne reviendrons donc pas. On se reportera par exemple à [Israel, 1997] pour une information complète sur les caractéristique du SGBD. Nous n'allons pas ici entrer dans le détail fastidieux du développement mais seulement revenir en **annexe 45** sur le typage des champs qui nous semble un point plus important à évoquer. En effet, plusieurs problèmes se sont posés quant à la façon dont devaient être gérés par exemple les dates, les caractères accentués polonais, la clé primaire, les descripteurs associant la table au modèle. Nous distinguerons dans cette annexe les questions relatives aux mécanismes d'intégrité, à la gestion des index et aux contraintes de configuration initiale du SGBD, questions liées à la plate-forme logicielle choisie, et les questions relatives aux pré et post traitements des contenus, questions liées à notre modélisation. Nous renvoyons également en **annexe 46** une présentation des principes de l'interfaçage Web avec SQLServer.

## 5.2. APPLICATIONS

Nous avons, aux chapitres précédents, présenté les objectifs de notre travail, les principes de modélisation adoptés puis les implémentations mises en œuvre. Il nous reste à décrire les divers outils élaborés pour tirer parti du modèle implémenté puis à rendre compte des expérimentations faites autour de ces outils. Dans cette section, nous discuterons de la mise en œuvre d'applications développées ou utilisées dans le cadre de notre projet. Nous nous attacherons à situer clairement les objectifs de chacune d'entre elles afin de cerner sans ambiguïté leur rapport aux problèmes de modélisation qui forment le fond de notre travail. Nous en présenterons de façon succincte les principes. Nous devons néanmoins commencer par situer les outils en question par rapport d'une part à l'implémentation concernée et d'autre part aux besoins auxquels ils répondent.
L'axe de recherche du laboratoire GAMSAU-MAP MOMA (Mesures Optiques et Modèles Architecturaux) traitait d'un modèle architectural utilisé comme support à la fois de la phase de mesurage de l'édifice et de la phase de représentation ou de formulation d'hypothèses de restitution. Les outils mis en œuvre dans l'implémentation JAVA du modèle s'inscrivent dans ce cadre exact. Les applications développées et leurs interfaces Web avaient pour objectif de donner accès a la fois au modèle lui-même mais aussi aux étapes de mesurage et de formalisation d'hypothèses de restitution. L'implémentation JAVA et les trois outils correspondant s'inscrivent donc dans une problématique qui est celle du renseignement d'un modèle par la mesure. L'application majeure est ici liée à l'instrumentation du processus de mesurage, les soucis de documentation et de représentation étant laissés au second plan[149].

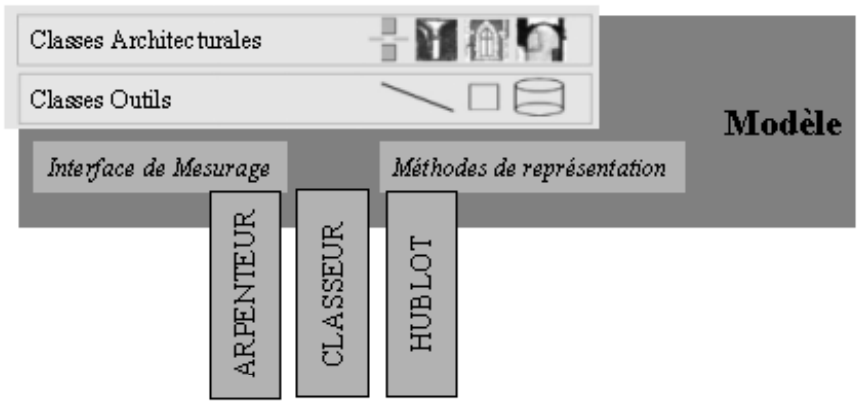



*Figure 79 : Les outils développés à partir de l'implémentation JAVA*

Les outils développés dans le cadre de l'implémentation JAVA répondent à un besoin de détermination quantitative de l'édifice pour lequel la formalisation d'un modèle architectural nous semble permettre une économie de moyens non négligeable.
Le projet ARKIW dans le cadre duquel a été introduite l'implémentation PERL a des objectifs assez différents puisque l'accent est ici mis, par exemple, sur la constitution de maquettes représentant diverses hypothèses de restitution, sur la documentation de ces hypothèses, sur la formalisation d'un modèle architectural s'appuyant sur l'analyse du corpus menée par une équipe de conservateurs du patrimoine, sur un interfaçage Web basique , etc… On le voit, la problématique du mesurage en est absente. Le modèle n'est pas ici utilisé comme "moule à instances" renseignables par la mesure mais comme moyen d'organiser et de documenter des concepts uniquement liés au domaine d'application.

[149] . Les trois outils qui ont ici été développés, et qui sont présentés plus loin, sont les suivants :
-L'ARPENTEUR, plate-forme de relevé photogrammétrique numérique sur le Web, construite à l'origine comme outil dédié au relevé architectural issu du projet PAROS puis largement réorienté et diversifié.
-Le CLASSEUR, outil de génération de nouvelles classes architecturales pour raffiner le modèle existant
-Le HUBLOT, interface Web permettant de visualiser les résultats du processus de mesurage en VRML et d'interagir avec eux.

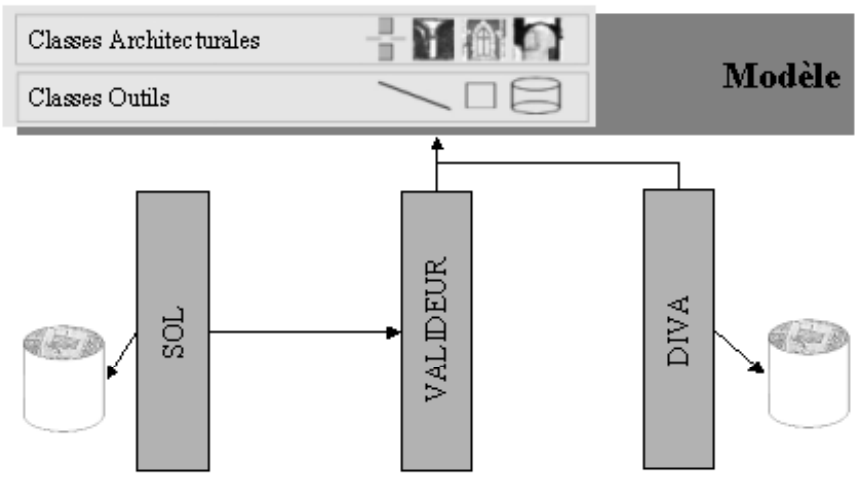



*Figure 80 : Les outils développés à partir de l'implémentation PERL*

Les outils développés dans le cadre de l'implémentation PERL répondent à un besoin de représentation et de documentation de l'édifice pour lequel nous pensons qu'une analyse du corpus architectural telle que celle que nous proposons peut constituer une réponse valable[150].

### 5.2.1. l'ARPENTEUR

ARPENTEUR, "an ARchitectural PhotogrammEtry Network Tool for EdUcation and Research", est une application de photogrammétrie numérique dédiée à l'architecture et accessible à tous via le réseau Internet[151].
ARPENTEUR est basé sur la technologie WEB, serveur HTTP, serveur FTP, communication entre machine par le protocole IP. Les calculs photogrammétriques et les traitements d'images sont développés en Java. Nous n'allons pas ici décrire un travail qui n'est pas le nôtre mais seulement en situer le fonctionnement afin de replacer l'expérience du relevé photogrammétrique de l'ancien hôtel de ville de Cracovie [Drap et al, 1999a] dans une perspective plus claire. En effet, L'ARPENTEUR est l'outil au travers duquel nous avons tenté de valider le processus de renseignement du modèle par la mesure.
Les objectifs de l'ARPENTEUR sont les suivants :

- Un développement à destination des communautés de l'enseignement et de la recherche utilisant Java pour autoriser l'indépendance de plateforme.
- Un outil sur le Web dédié au relevé architectural.
- Un système de photogrammétrie simple pour les archéologues, les architectes et les photogrammètres.

[150] Dès lors, les applications développées se distinguent largement des précédentes :
-SOL est un outil de recherche bibliographique, iconographique et cartographique sur le Web. Il fait référence aux sources bibliographiques et iconographiques traitant des édifices choisis comme terrain d'expérimentation du programme ARKIW. Il s'appuie sur un système de gestion de base de données interfacé pour le Web. Le modèle est présent comme une grille d'analyse des sources : la représentation d'instances de ce modèle (scènes VRML) permettant d'interroger la base.
-VALIDEUR est une interface Web de génération de scènes au format VRML 2.0 permettant d'instancier un jeu d'objets s'appuyant sur le modèle architectural sous-jacent. Il s'agit donc d'un "modeleur architectural" par référence à l'expression "modeleur géométrique". Naturellement accessible à distance, il s'appuie sur une architecture client Web - serveur Web basique au travers de programmes CGI.
-DIVA est un dictionnaire méthodologique trilingue centré sur le vocabulaire architectural, référençant des termes décrivant les édifices choisis comme terrains d'expérimentation. Son objectif est de permettre une désignation commune et non équivoque des objets du corpus à décrire dans l'élaboration d'une hypothèse de restitution.
[151] Elle est le résultat d'un travail de collaboration entre l'UMR MAP-GAMSAU et le laboratoire de photogrammétrie de l'ENSAIS (Ecole Nationale Supérieure des Arts et Industries de Strasbourg), sous la responsabilité de P.Drap côté MAP-GAMSAU et de P.Grussenmeyer côté ENSAIS. Outil de photogrammétrie léger, dont les auteurs maîtrisent complètement la structure et le développement. Il est le prolongement sur Internet du logiciel de photogrammétrie TIPHON développé à l'ENSAIS.

Ce travail trouve aujourd'hui de nombreuses utilisations au-delà du champ de l'architecture. Nous nous bornerons ici à rapporter les aspects de l'application liées à l'interaction entre l'outil de mesure et le modèle architectural.
L'ARPENTEUR est une applet prenant en charge les étapes classiques d'un relevé photogrammétrique (définition du modèle photogrammétrique) et proposant en outre une méthode d'orientation absolue sans points de contrôle, en définissant un système de coordonnées local défini par des points remarquables sur l'édifice. L'ARPENTEUR peut être utilisé localement, mais il est conçu plutôt pour être utilisé comme une applet. Les différences de performance entre l'implémentation JAVA (ARPENTEUR) et l'implémentation C++ (TIPHON) des modules de calcul photogrammétrique sont aux dires des auteurs insignifiantes. Utilisé en applet, ARPENTEUR nécessite que soit installé sur le client un navigateur utilisant une version JAVA de niveau adéquat, à savoir JDK 1.1.7 au moment de l'expérience que nous rapportons[152].
Moyennant l'installation d'une version de navigateur compatible, ARPENTEUR est accessible sur Internet et utilisable dans un simple navigateur Web. Trois serveurs sont utilisés : un serveur HTTP, un serveur FTP et des serveurs RMI. L'étape d'authentification de l'utilisateur ouvre un espace de travail personnel à l'utilisateur enregistré. Un serveur FTP donne accès à cet espace de travail (lecture / écriture de fichiers d'entrée du processus ou de résultat). L'applet tourne côté client, les restrictions de sécurité JAVA s'appliquent, il n'est donc pas possible d'écrire les résultats du processus de mesurage sur le disque client ou sur le disque serveur. Une servlet est dédiée à l'écriture des fichiers de résultats sur le serveur. Parmi les objets manipulés par l'ARPENTEUR, de nombreux implémentent une interface VRML qui permet de visualiser les instances créées pendant le processus de mesurage[153]. L'interaction JAVA / VRML est utilisée pour autoriser l'utilisateur à récupérer des données depuis un objet affiché dans la scène. Le script LaDémarche, créé en résultat de la phase de mesurage initiale, est modifiable côté client dans un processus essai-erreur permettant à l'utilisateur d'intervenir sur les instances mesurées ou d'instancier de nouveaux objets. Nous avons utilisé cet outil dans une expérimentation du processus de mesurage sur le beffroi de l'ancien hôtel de ville de Cracovie. S'il a permis de valider ce processus, l'outil reste néanmoins avant tout un développement à l'intention des photogrammètres, dans lequel le modèle architectural n'intervient que marginalement puisqu'en définitive n'importe quelle forme décrite par le biais de primitives géométriques est étudiée de la même. ARPENTEUR est donc un outil de relevé photogrammétrique dont nous avons montré qu'il pouvait être adapté à la recherche d'informations dimensionnelles sur des objets architecturaux.

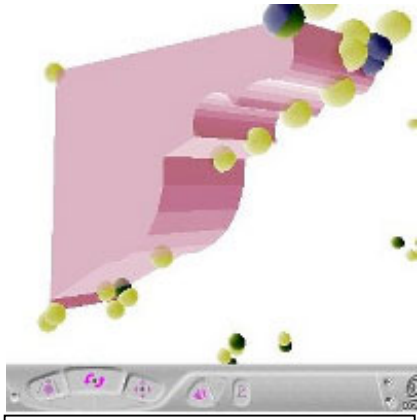


*Figure 81 : Visualisation VRML des instances d'objets architecturaux et de Points dans le processus de mesurage de l'ARPENTEUR*

### 5.2.2. Le HUBLOT

L'interface hublot permet de créer et de visualiser un ensemble d'arrangements spatiaux utilisant le modèle architectural prédéfini, par exemple une hypothèse de restitution. Le hublot permet l'édition de scripts de type 'La demarche', leur compilation et la visualisation d'une expression graphique en VRML liée à une expression plus complète sous forme de texte formaté en HTML. Le langage de description de scènes 'LaDémarche' porté en JAVA permet de décrire textuellement un jeu d'entités architecturales et de relations les liant les unes aux autres. Un compilateur utilise cette description textuelle pour représenter l'édifice sous la forme d'une collection d'instances du modèle sous-jacent.

[152] Ce point, qui peut sembler anecdotique, s'avère dans la pratique bien loin de l'être. En effet, c'est le principe même d'un travail centré sur le Web qu'il remet en cause : la disponibilité sur le réseau pour toute machine connectée à Internet d'un service ou d'un outil quelque soit la plateforme matérielle utilisée côté client. Il nous semble important de relever dès à présent cette ambiguïté : une applet JAVA reste dépendante de la plate-forme matérielle et logicielle du client.
[153] Les points mesurés, les points de contrôle, les primitives géométriques et les objets architecturaux mesurés peuvent ainsi être visualisés pendant les étapes successives du mesurage.

Le HUBLOT est en fait une interface (applet) à ce compilateur permettant de choisir et d'éditer un script. A la suite de la phase de compilation, l'interface donne un accès aux instances créées et, par un mécanisme d'envoi de messages, permet de visualiser le jeu d'objets en VRML ou d'obtenir des informations sur les instances du modèle. Le hublot permet plusieurs interactions avec le modèle[154] ; il a permis de valider l'ensemble du processus de mise à jour du modèle puis d'exploitation de ce modèle suivant le schéma rapporté en **annexe 47** (ainsi qu'une présentation graphqiue de l'Architecture Client / Serveur de l'application HUBLOT et qu'un exemple de scène).

### 5.2.3. Le CLASSEUR

Le formalisme objet, tel qu'utilisé jusqu'ici, nous permet d'étendre en le spécialisant le corpus déjà modélisé. Toutefois, donner à des non-informaticiens la possibilité de générer de nouveaux concepts dans le corpus existant nous a poussé à développer un générateur de code capable d'exprimer le code des classes d'objets.Le Classeur offre donc une interface utilisateur orientée vers l'architecte et l'archéologue. L'utilisateur définit une nouvelle classe, un objet architectural, par ses attributs spécifiques et sa position dans l'arbre d'héritage. Le nouvel élément du corpus est ajouté dans l'arbre et possède ses propres attributs. La génération du code assure son intégration dans le corpus existant et dans la grammaire nécessaire à l'élaboration de scripts définissant un ensemble architectural. Pour utiliser le classeur il faut et il suffit en principe d'être expert de son domaine (architecte, archéologue) et de connaître le point de vue présidant à la taxinomie des objets (en principe seulement : de nombreux points posent des problèmes concrets non triviaux, par exemple faire accepter auprès des experts du domaine en question la simple idée de procéder à une classification hiérarchique, quelqu'en soit la logique, de concepts architecturaux). Les nouveaux concepts élaborés par le chercheur qui utilise le Classeur sont intégrés sans recompilation dans le langage de gestion de scène (LaDemarche). Bien que posant un problème important, celui de la prise en main par des non-spécialistes d'un modèle architectural dont ils sont après tout les inspirateurs voire les co-auteurs, le CLASSEUR reste une application non aboutie. En effet, au-delà de l'idée d'un modèle placé à la portée du non-spécialiste, le CLASSEUR fait référence à un problème évoqué dans [Ducournau et al, 1998,p305], celui de l'évolution des concepts. Deux points sont ici à relever[155] :

- Le CLASSEUR est une application dédiée à l'extension du modèle, elle ne résout pas les problèmes plus courants de la mise à jour du modèle.
- Le CLASSEUR ne gère pas la notion de perspective sur le concept : celui-ci n'est vu que sous un aspect, celui qu'impose la logique de taxinomie principale.


*Figure 82 : Rynek główny 1787*

### 5.2.4. SOL

SOL (Sources On Line) est un outil de recherche bibliographique, iconographique et cartographique sur le Web, censé être facile d'usage, accessible depuis toute machine connectée à Internet. Il a été construit pour rassembler sur une plate-forme partagée les sources bibliographiques et iconographiques traitant des édifices choisis

[154] -ddition ou retrait d'entités aux réseaux,
-modification des caractéristiques morphologiques des entités architecturales (dimensions, orientation, position)
-Calcul et visualisation des résultats VRML et des rapports textuels en HTML (rapports établissant les caractéristiques de chaque entité).
-Action itérative sur chacun de ces trois niveaux.

[155] Par ailleurs, Le CLASSEUR génère un code minimal ne permettant pas d'exploiter entièrement la classe créée. En particulier, la méthode de représentation de l'objet ajouté au modèle est créée vide : l'écriture effective des opérations effectuées par cette méthode reste manuelle. L'outil CLASSEUR semble plus utile par les questions qu'il pose que par les réponses qu'il apporte. Il apparaît clairement que l'évolution du modèle, tant en terme de mise à jour de l'implémentation que de modifications de la taxinomie existante, sera un point central des perspectives à tracer pour notre projet. Si le premier point est intimement lié à l'instabilité du couple JAVA/ navigateurs Web, le second est quand à lui plus fondamental, il dessine un axe de travail particulièrement intéressant par rapport au formalisme de représentation des connaissances choisi.

comme terrains d'expérimentation du programme ARKIW. Il s'appuie sur un système de gestion de base de données interfacé pour le Web. Il comprend a ce jour 450 entrées, chacune en rapport avec un des objets architecturaux organisés autour du *rynek główny*, ou avec un problème relevant du champ de la conservation (édifices, détails, matériaux, ..). Son interface est développée en langue polonaise. SOL est un outil de recherche dans lequel des critères de description issus d'une analyse "orientée domaine"[156] de chaque entrée sont ajoutés aux critères descriptifs traditionnels. Chaque contribution au système (ajout d'une entrée au travers de l'interface de mise à jour sur le Web) est le résultat d'une lecture critique de la ressource à indexer, mise à disposition de la communauté de chercheurs. SOL est donc développé comme un module d'information collaboratif que chaque participant peut enrichir, notamment dans le cadre de recherches doctorales.

#### 5.2.4.i) Documents indexés

Les documents référencés sont des textes, des illustrations, des photographies ou des plans. Chaque document est décrit traditionnellement (auteur, éditeur, etc..). Il est également attaché à un ou plusieurs édifices et à une ou plusieurs bibliothèques de la ville. Chaque document est de plus indexé comme traitant d'une période historique, d'un ensemble de *problèmes* (charpente, clocher, soubassement, ...). Parmi d'autres champs, on trouve notamment pour toutes les illustrations un point de vue qui indique la position de la prise de vue, ainsi qu'une adresse URL permettant de créer un lien vers tout document Web complémentaire, notamment par exemple vers une page présentant l'illustration ou l'extrait de texte cité.

#### 5.2.4.ii) Points de vue thématiques

Les points de vue thématiques permettent au système de prendre en compte des informations ne figurant pas dans un catalogue de ressources classique. Ces éléments d'informations sont le résultat d'une analyse de la ressource et permettent d'interroger le système sur des critères relevant du domaine patrimonial.

| | |
|---|---|
| Contexte urbain | Relation aux édifices de la place centrale. |
| Eléments architecturaux | Relation au corpus architectural . |
| Période | Relation à la période de présence de l'édifice sur la place ou à son évolution. |
| Orientation | Relation à l'angle depuis lequel un édifice est vu et représenté sur une illustration. |
| Media | Média concerné : texte - illustration (type)- plan. |
| Groupe thématique | Relation aux problèmes de conservation (matériaux, techniques, etc..). |
| Disponibilité | Référencement dans les archives et bibliothèques. |
| Édifice | Relation à des objets spécifiques d'un édifice (Horloge de l'hôtel de ville, etc…). |
| À propos de | Référence les auteurs que cite la ressource. |
| URL | Connection à une adresse URL (donne accès à une adresse distante ou locale qui contient une information non pertinente dans le contexte de la base de données comme un texte complet, des images, etc….). |

#### 5.2.4.iii) Interface de recherche

Un document référencé dans la base de données peut être recherché de plusieurs façons :

- Champ par champ dans un interface standard CGI prédéfini (avec mises à jour automatiques des critères).
- Par création interactive d'une interface personnalisée qui comprend la liste de critères que l'utilisateur juge pertinente.
- Graphiquement sur un ensemble de plans 2D qui donnent accès aux recherches sur les critères urbains, historiques, type de média et orientation.
- Graphiquement sur un ensemble de scènes 3D en VRML 2.0 qui donnent accès aux objets architecturaux élémentaires par période d'évolution de la place.
- Accès privilégiés qui autoriseront l'accès en modification et la lecture de notes de travail.

[156] Ou analyse de documents centrée sur le contenu (de quoi parle la ressource) par opposition à une analyse de documents centrée sur le contenant (qu'est ce qui caractérise la ressource du point de vue de l'édition).

Deux schémas placés en **annexe 48** récapitulent les types d'interfaces de requête aujourd'hui disponibles et les placent en rapport aux groupes thématiques identifiés dans SOL.

#### 5.2.4.iv) Procédures de mise à jour

Deux types de mises à jour à distance (par le biais d'une interface Web) sont proposés :
- Ajout d'une ressource : une interface textuelle affiche les champs à compléter et gère les problèmes liés aux données incomplètes et à l'accentuation des caractères polonais.
- Ajout de critères descriptifs : Les critères déjà présents dans le système sont affichés dans une interface textuelle qui autorise l'ajout de nouveaux critères (qui peuvent correspondre par exemple à une spécificité morphologique d'un édifice que mentionne la ressource et qui ne fait pas partie des critères existants). Le système met à jour en temps réel la liste de critères et permet à l'utilisateur de référencer sa ressource en prenant en compte son nouveau critère. L'interface de recherche est bien entendu parallèlement mise à jour.

SOL propose deux interfaces conçues pour répondre à la double exigence d'un accès distant via Internet et d'une meilleure adéquation avec le domaine du patrimoine architectural :
- Une interface de mise à jour à la fois des données et de leurs descripteurs.
- Une interface de recherche graphique 2D et 3D prenant en compte l'importance de la représentation dans le travail de l'architecte.

En instrumentant modifications ou additions d'entrées à la base de données, l'outil développé constitue un premier module d'information spécialisé dédié aux chercheurs impliqués dans le programme de coopération ARKIW, et centré sur les problèmes de conservation dans le contexte du centre de la ville de Cracovie.
Le principe de ce développement est l'interfaçage d'un SGBDR (SQLServer) sur le Web au travers d'un ensemble de modules PERL. Les méthodes et attributs des objets PERL prennent en charges les étapes successives de l'interfaçage client Web / serveur Web : connection, codage, écriture des formulaires CGI, etc...
Les scènes VRML permettant d'interroger la base sont autonomes vis à vis de SOL comme de l'outil de modélisation architecturale sur le Web VALIDEUR, présenté plus loin, pour autant que le node ANCHOR des objets contiennent une URL valide. Ce choix correspond à deux objectifs :
- Adapter une scène à des utilisations distinctes en terme d'interface de requête : un même objet doit pouvoir référencer selon le cas l'édifice dans son ensemble ou un élément isolé de corpus.
- Utiliser la scène non comme un résultat en soi mais comme un moyen d'accéder à une information évolutive.

SOL s'appuie sur le modèle architectural sans pour autant en relever. Ce choix a été justifié précédemment. Pourtant, il est clair que nous considérons la représentation tridimensionnelle de l'édifice comme un mode d'accès privilégié à la base documentaire. Nous considérons en effet que l'implémentation d'un modèle architectural capable d'exprimer spatialement des objets du corpus étudié peut favoriser une meilleure compréhension du tissu urbain, et constitue une réponse possible à la quantité et à la complexité des données assemblées à ce jour autour du patrimoine architectural du lieu considéré.

### 5.2.5. DIVA

En parallèle à SOL, un deuxième outil du même type a été développé : le dictionnaire méthodologique trilingue pour le vocabulaire architectural DIVA, centré autour des termes décrivant les édifices choisis comme terrains d'expérimentation.
Son objectif est de permettre une désignation commune et non équivoque des objets du corpus a décrire dans l'élaboration d'une hypothèse de restitution. En effet, notre analyse du corpus vise à définir un ensemble de concepts architecturaux univoques. Il nous est donc apparu essentiel de mettre en partage un vocabulaire précis désignant ces concepts.

L'interface web de la base de données DIVA permet à l'utilisateur de rechercher les traductions françaises polonaises et anglaises d'un mot écrit dans une de ces trois langues. Les 900 termes dont nous proposons une traduction (accompagnée des références bibliographiques ayant servi à l'établir) font partie du vocabulaire architectural et plus particulièrement de celui utilisé dans le projet ARKIW. Ceci dit l'outil DIVA ne recense pas que des termes relatifs aux concepts architecturaux modélisés, loin de là. En effet, ce développement a été initié indépendamment du travail d'analyse du corpus. Comme SOL, DIVA est un outil autonome vis à vis de l'élaboration du modèle architectural mais qui peut contribuer à le renseigner.

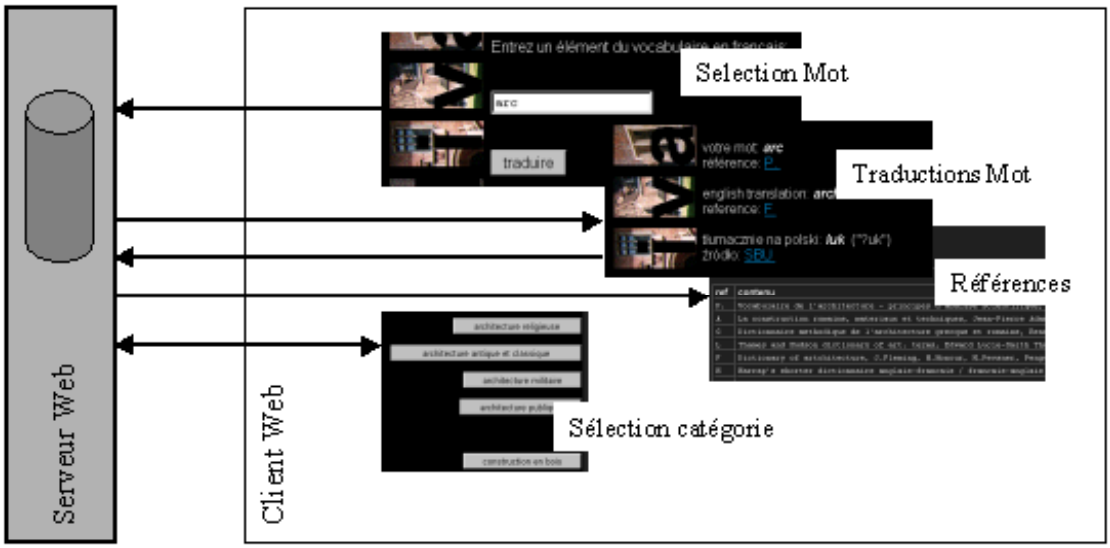



*Figure 83 : Interface Web de l'outil DIVA*

L'interface Web de l'outil DIVA dans son état actuel propose une recherche mot par mot mais aussi une recherche catégorie par catégorie, reprenant pour l'essentiel les grands chapitres de [Pérouse De Montclos, 1988][157].

Travail plus ancien que SOL et VALIDEUR, DIVA ne reprend que partiellement la plateforme d'interfaçage PERL présentée précédemment. Une mise à niveau de l'outil est prévue dans un proche avenir. Au-delà, plusieurs pistes sont ouvertes pour développer cet outil et en élargir la portée, avec deux directions privilégiées :

- Rôle de l'image dans l'interface de requête.
- Référencement d'instances des concepts sous la forme d'un catalogue d'URL locaux ou distants structurés en XML.

### 5.2.6. VALIDEUR

Dans l'interface 3D de recherche de documents que propose SOL comme dans le cadre de l'étude des anciens plafonds en bois, nous nous appuyons sur le langage de description de scènes 3D VRML version 2.0. Nous proposons une interface Web de création de scènes sur Internet qui permet aujourd'hui de créer une scène VRML s'appuyant sur le modèle architectural sous-jacent depuis n'importe quelle machine connectée sur Internet. Les objectifs de cet outil sont les suivants :

- Autoriser la création et la modification de scènes sur le Web.
- Appuyer cette démarche sur le modèle architectural développé pour le projet et par la même représenter celui-ci.
- Proposer une connection depuis chaque objet créé (chaque entité architecturale) vers une URL qui sera dans le cas de SOL une requête vers la base de données.

[157] Une recherche par l'image artisanale (Javascript/Perl) a été expérimentée, reprenant là encore le principe d'utilisation de notre ouvrage de référence en matière de vocabulaire architectural, [Pérouse De Montclos, 1988].

- Autoriser la création interactive de scènes VRML (et accessoirement le travail collaboratif puisque sur une même scène, en temps réel, peuvent travailler par exemple un intervenant en France et un intervenant en Pologne. Ce point ne constitue cependant pas pour nous une priorité).
- Détailler la représentation des entités architecturales pour visualiser par exemple le rapport profil / terminaison d'une solive.
- Alimenter une banque de maquettes du "*Rynek Główny*" indépendantes de tout outil commercial.
- Proposer des modules didactiques d'information sur les anciens plafonds en bois des maisons urbaines.

Un réseau est représenté dans l'outil VALIDEUR sous la forme d'une scène VRML interactive dont les points de vue standards sont calculés par le système. Chaque entité architecturale présente dans le modèle peut être ajoutée à la scène active ou à une nouvelle scène. Chaque entité instanciée dans une scène peut servir de *patron morphologique* pour cette scène aux entités de même type (i.e. instances de la même classe) créées par la suite. Pour chaque entité, l'interface textuelle ou la représentation 3D donnent accès aux attributs morphologiques ou spécifiques à la représentation graphique, mais également à la propriété URL de l'objet entité, autorisant ainsi la création de liens vers une adresse Internet locale ou distante.

Cet outil, dont le développement a été entamé au printemps 1999, n'a pas vocation à faire partie de la famille des modeleurs géométriques. Il a pour vocation de rendre compte de notre analyse du patrimoine architectural. Il intervient comme un outil de visualisation rapide d'une hypothèse de restitution archéologique.
En effet, compte tenu du manque d'informations précises et de l'état des édifices étudiés (souvent partiellement détruits ou largement transformés), toute reconstruction se base sur des comparaisons et des analogies.

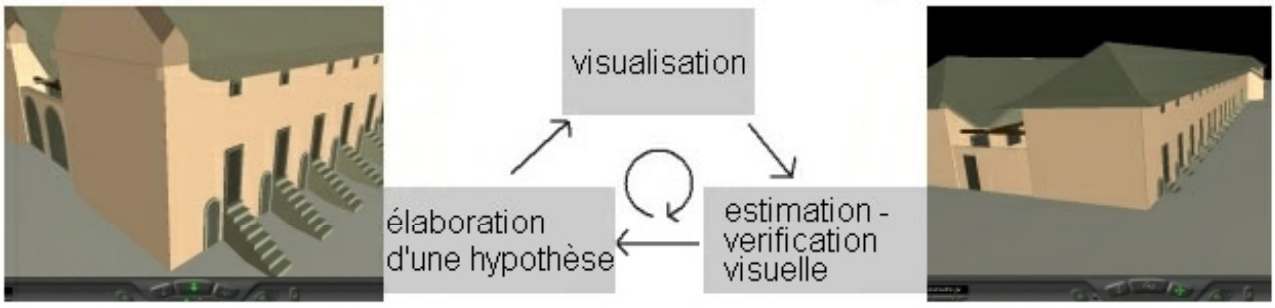



*Figure 84 : La visualisation comme étape de vérification dans le processus de validation d'une hypothèse de restitution*

Ceci suppose bien sûr une place importante laissée au point de vue subjectif de l'auteur d'une proposition de reconstruction, et par conséquent élargit le champ des propositions possibles. Il s'agit donc ici de proposer un outil souple de visualisation et d'estimation graphique par essais-erreurs[158].
Etant développé pour Internet, la création et la modification de scènes VRML à distance sont autorisées. Toutefois, une scène VRML peut s'avérer longue à télécharger. VALIDEUR a donc été conçu avec pour objectif d'en assurer la portabilité. Ce développement pourra donc être installé et mis à jour sur les postes de travail des partenaires du projet.
VALIDEUR met en partage un modèle architectural et ses représentations HTML et VRML. Il est donc un outil didactique privilégié pour rendre compte de notre analyse du corpus. L'interface actuelle de l'outil s'appuie entièrement sur les modules d'interfaçage décrits aux

[158] VALIDEUR a été testé sur les terrains d'expérimentation Kramy Bogate et anciens plafonds en bois, ainsi que dans la création des scènes utilisées comme interfaces de navigation dans la base documentaire SOL.

sections concernées. Les objets créés sont des instances du modèle et intègrent le formalisme de description des moulures par points de contrôle et indications de type de courbure. La notion de dépendance topologique, ou de dépendances de position, y est implémentée. Elle intervient exclusivement dans l'état actuel du développement pour positionner une entité par rapport à une autre en rapport à un axe principal. Il n'est pas pour l'instant gardé trace de cette action.

Nous verrons dans la section consacrée aux développements en cours quelles directions nous prenons pour améliorer cet outil. Plusieurs points, qui nous semble très importants, sont en jeu :

- La gestion des niveaux de détail.
- La persistance des relations.
- La gestion des réseaux prédéfinis.
- L'interface avec un moteur de rendu.
- Les fonctionnalités d'animation.
- La gestion des points de vue sur l'instance. Nous parlons ici des multi-actions dont doit relever l'instance et qu'il s'agit d'interfacer côté client, comme par exemple le déclenchement d'actions versus la connection à une URL, l'asservissement de l'action déclenchée au niveau de détail, etc..
- La structuration et le formatage des URL associées.
- Le transfert des résultats côté client.

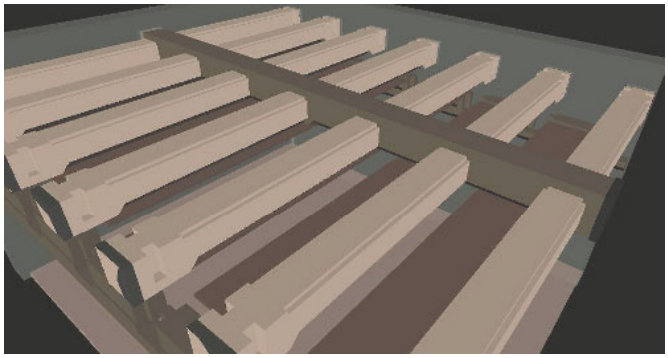


*Figure 85 : Restitution d'un plafond en bois, vue partielle*

VALIDEUR reste pourtant, et doit rester, une application basique. Ce choix, bien que pouvant paraître frustrant au premier abord, nous permet de faire face à l'exigence de portabilité, de stabilité et de disponibilité qu'impose le type de collaboration que nous avons choisi. En effet, le temps du conservateur n'est pas le temps du développeur d'applications. C'est dans la définition du service attendu par le premier que nous avons porté notre effort le plus important.

VALIDEUR s'inscrit ainsi comme une contribution allant dans le sens d'une modélisation architecturale dirigée par la connaissance. Si la plateforme technique choisie en limite par essence les performances, elle n'en constitue pas moins un moyen d'évaluation des choix de modélisation. En cela, elle est de nature à faire progresser la définition du modèle et sa pertinence au regard du domaine de connaissance abordé. En définitive, VALIDEUR renvoie à une problématique de modélisation, celle dont ce travail a souhaité traiter, plus qu'à une problématique d'implémentation pour laquelle l'outil apporte des réponses pragmatiques et de bas niveau.

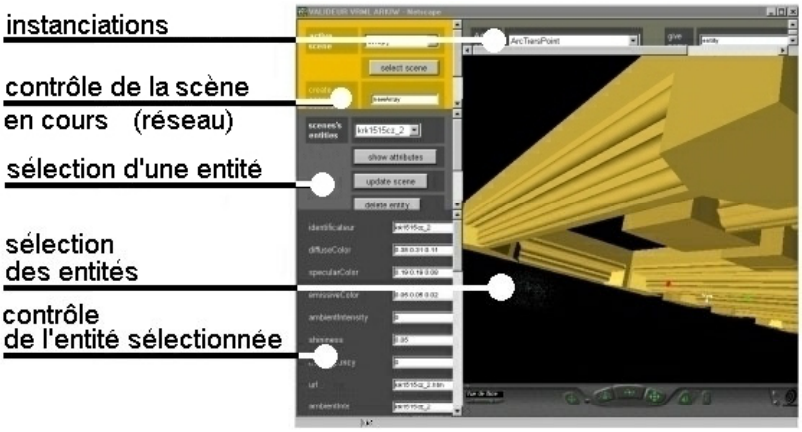



*Figure 86 : L'interface de manipulation des instances.*

# 6.CONCLUSION

## 6.1. LES DEVELOPPEMENTS EN COURS

Notre contribution, telle que rapportée dans ce document, apparaîtra certainement morcelée par bien des aspects[159]. Bien sûr, ainsi qu'il nous semble l'avoir démontré, il existe entre ces travaux un point commun essentiel, à savoir cette hypothèse que nous avons faite selon laquelle il est possible de représenter un jeu de connaissances architecturales par un modèle tirant parti du formalisme objet. Nous avons souhaité récapituler en **annexe 50** quelques-uns des points principaux de notre contribution et introduire les axes de travail que nous nous fixons aujourd'hui[160]. Au-delà de ces points se posent deux questions transversales : celle de l'intégration des diverses échelles de définition des objets dans le modèle (corpus-architecture urbaine - détails) et celle, concomitante, des multi-représentations des instances. Ces questions resteront centrales dans notre travail, elles constituent des obstacles non triviaux dans notre démarche de définition et d'organisation du modèle. En effet, l'objectif que nous devons poursuivre est d'autoriser, pour chaque concept modélisé, la sélection de représentations alternatives de l'objet non pas en fonction d'une logique de points de vue (sélection d'un sous-ensemble des propriétés de l'objet) mais en fonction de lectures alternatives d'un groupe de concepts.

## 6.2. APPORTS DE NOTRE CONTRIBUTION

Le travail que nous présentons dans ce document avait pour objectif premier de confronter un formalisme de représentation des connaissances - l'approche objet - à un domaine d'application particulier - le patrimoine architectural. Il est donc fondé sur l'hypothèse selon laquelle il est possible de représenter un jeu de connaissances architecturales par un modèle tirant parti du formalisme objet, hypothèse que nous nous sommes attaché à vérifier. Ce travail avait comme second objectif d'expérimenter sur des cas concrets un certain nombre d'outils d'exploitation du modèle et d'ainsi évaluer sa pertinence au regard des attentes des spécialistes du domaine d'application.
Nous avons cité les références essentielles sur lesquelles nous appuyer pour effectuer le travail d'identification et de classification des concepts architecturaux que rendait nécessaire le formalisme de représentation des connaissances choisi[161].
Nous avons présenté des travaux de référence sur un certain nombre d'aspects liés à l'exploitation de ces vues : mesure, visualisation-restitutions, documentation. Nous avons en effet choisi de nous limiter à ces trois aspects pour évaluer et expérimenter notre modèle. Les travaux cités font apparaître que, prises isolément, ces types d'études peinent à faciliter la compréhension de l'édifice. A la suite de ces constats, nous avons donné quelques éléments de définition de l'approche objet, et cité les problèmes de modélisation qu'elle pose dans notre cas : principes de réification, évolutivité et lisibilité du modèle, héritage simple / multiple, perspective sur l'objet, etc... Les choix que nous avons fait en terme de plate-forme de développement ont été introduits afin de bien situer l'échelle de ce travail.

[159] : plusieurs implémentations, des applications d'architectures et d'objectifs divers, des terrains d'expérimentation relevant de préoccupations variées, etc…
[160] Nous y aborderons en premier lieu les questions relatives aux avancées actuelles de notre modélisation et aux choix devant lesquels nous nous trouvons. Le modèle sera ainsi replacé au centre de ce travail. Nous établirons ensuite d'autres perspectives, notamment sur le champ de la représentation et de la documentation de l'édifice.
[161] Plusieurs points ont été soulevés:
-Les classifications existantes se basent principalement sur l'étude du vocabulaire décrivant l'édifice aux différentes échelles.
-Les classifications existantes ne se contraignent pas à respecter une logique de division du modèle unique.
-La séparation des notions relatives à l'édifice physique et aux lieux qu'il définit n'est pas souvent faite.
-Les concepts aptes à représenter l'édifice patrimonial ne peuvent être identiques à ceux utilisés dans le processus de production des bâtiments.
-La diversité des études menées sur l'architecture construite appelle la mise en œuvre d'un modèle médiateur entre diverses vues sur l'objet étudié.

Notre contribution avait pour objectif, au-delà des questions liées à la représentation des connaissances, d'évaluer des modes d'exploitation du modèle, et ce au travers d'applications client-serveur sur la plate-forme Internet. Nous avons donc resitué dans leurs cadres respectifs les choix faits en terme de gestion de données, de formats graphiques et d'interfaces réseau.
Nous avons ensuite abordé le compte-rendu de notre travail en expliquant comment nous avons isolé quatre grandes catégories de concepts architecturaux (entités, attributs, réseaux et relations) et quelles logiques d'organisation de ces concepts nous avons adopté. En explicitant les règles de définition du modèle, nous avons vu également quelles sont les limitations actuelles de ce modèle (relations, assemblages, réseaux, décor, attributs, tracés, etc..).Nous avons ensuite décrit les deux principales implémentations menées à bien, l'implémentation JAVA autour de la problématique de la mesure, et l'implémentation PERL, autour de la problématique de la documentation de l'édifice sur Internet. Nous avons insisté sur les objectifs respectifs de ces deux implémentations en justifiant le caractère minimaliste de la seconde notamment par le contexte du projet de coopération dans lequel s'inscrit notre contribution.
En présentant applications et terrains d'expérimentation, nous avons tenté de montrer quels usages pouvaient être faits d'un modèle architectural sous-jacent dans le cadre d'études patrimoniales. Nous avons, il nous semble, montré qu'une analyse a priori du corpus d'éléments physiques formant l'édifice peut grandement en faciliter l'étude. Mesure, représentations et documentation deviennent dans ce cadre des processus visant à renseigner le modèle et ses instances. La compréhension globale de l'édifice bâti et de son évolution est mieux assurée puisque sa logique de constitution est :

- Représentable telle quelle (notions d'entités, d'attributs, de réseaux et de relations).
- Documentée (sources bibliographiques attachées aux concepts et aux instances).
- Décrite antérieurement au processus de mesurage.

Nous avons également largement évalué la faisabilité de développements sur le Web et en avons montré quelques limitations, poids des applets JAVA, choix de standards, gestions des interfaces graphiques ou localisation de l'exécution de traitements. Reste à déterminer si oui ou non nous avons vérifié notre hypothèse de départ. Avons une réponse à la question : le formalisme objet permet-il de représenter les connaissances relatives au domaine d'application abordé ?
Nous pensons avoir démontré qu'une réponse positive à cette question implique le choix d'un point de vue rigoureux sur la classification des objets. Nous pensons avoir fait un tel choix et par conséquent avoir pu valider notre hypothèse de départ. Pour autant, cette conclusion doit être relativisée. En effet, la condition que nous venons d'édicter est loin d'être sans conséquences[162]. Les connaissances véhiculées dans le champ patrimonial ont souvent un caractère partiel, incomplet, mal défini. A cela nous ne pensons pas avoir apporté de réponse aboutie. Nous pensons en définitive avoir démontré qu'il est possible de formaliser un modèle de l'édifice patrimonial cohérent et exploitable : celui qui a été discuté ici et dont nous avons rapporté plusieurs expérimentations.
Mais nous pensons que ce modèle reste pour l'essentiel en chantier tant les questions qu'il pose sont nombreuses. La contribution que nous avons présenté a permis, nous l'espèrons, d'en relever quelques-unes des plus significatives et tenté d'établir un constat de dépendance mutuelle entre domaine d'application et instrumentation informatique.

[162] Quelques problèmes évoqués dans ce document peuvent être cités à l'appui de cette remarque :
-Nous ne savons pas modéliser les usages de l'édifice. Mais il s'agit là d'un problème de modélisation qui n'est pas lié à l'approche objet en particulier.
-Nous ne savons pas gérer l'incomplétude d'un objet (voir section 4.2.1, budynek Copernicus).
-Nous ne savons pas gérer les objets "malformés" (voir section 4.2.1, oculus de la place des comtes de Provence).
-Nous ne savons pas définir un concept par soustraction de propriétés (voir section 4.2.1, le cas de la chantepleure).
-Nous ne savons pas (ou mal) représenter notre incertitude quant à la définition d'un objet hypothétique
-Nous ne savons pas redéfinir une instance en fonction de son contexte d'utilisation.

# 7.BIBLIOGRAPHIE

Nous proposons ci-dessous, section 7.1, une liste exhaustive (par ordre alphabétique) des auteurs cités dans l'ensemble de ce document. Nous proposons à la suite de cette liste, section 7.2, des extraits de cette bibliographie se rapportant aux diverses sections de la partie Etat de l'Art du présent document.
Enfin, nous proposons en fin de section 7.2 la liste des publications relatives au travail présenté.

## 7.1. OUVRAGES ET PUBLICATIONS


[Acary et al, 1999] V. Acary, J.Y. Blaise, P. Drap, M. Florenzano, S. Garrec, M. Jean, D. Merad.
*"NSCD method applied to mechanical simulation of masonry in historical buildings using MOMA"*
XVII CIPA International SymposiumWG3 - Simple methods for architectural photogrammetry Olinda, Brazil, October 3-6 1999.

[Adam, 1989] Jean-Pierre Adam
*"La construction romaine, Matériaux et techniques"*
Édition A. et J. Picard, 82, rue Bonaparte, 75006 Paris, 1980.

[Albitz et al, 1998] P.Albitz, C.Liu
*"DNS and Bind"*
Editions O'Reilly, 3ème édition, 482 pages, 1998, ISBN 1-56592-512-2

[Alkhoven, 1993] Patricia Alkhoven
*"The changing image of the city. A study of the transformation of the townscape using Computer assisted Design and visualisation techniques"*
Thèse de doctorat de l'université d'Utrecht (Pays-Bas), décembre 1993.

[Apers et al, 1979] Jan Apers, Alfons Hoppenbrouwers, Jos Vanderbreeden
*"Inventaire d'urgence du patrimoine architectural de l'agglomération Bruxelloise"*
Sint-Lukasarchief, Paleizenstraat 70, 1030 Brussels, 1979.

[Atkinson et al, 1990] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, et S. Zdonik
*"The object-oriented database system manifesto"*
In Proceedings of the 1st Int. Conf. on Deductive and Object-Oriented Databases, Kyoto, Japan. Elsevier Science Publishers B.V (NorthHolland), 1990, pp40-57.

[Baculo Giusti et al, 1995] A. Baculo Giusti, L.Bucci de Santis, A.di Luggo, R.Florio, F.Rino
*"Napoli Citta' in vista"*
Université de Naples Federico II, edité par Electa Napoli 1995

[Baletti et al, 1999] C.Baletti, F.Guerra, L.Pilot
*"Plan and survey for the recovery and the care of an ancient abandoned nucleus: the castle of Campo of Brenzone"*
in Proceedings ISPRS WG V/5 Workshop, Thessaloniki, Grèce, 7-9 juillet 1999.

[Barberot, 1922] E.Barberot
"Aide mémoire de l'architecte et du constructeur"
Librairie Polytechnique Ch.Béranger, Paris, 1922, seconde édition (première édition 1914).

[Beckaert, 1977] Gert Beckaert
*"Introduction à l'édition de 1977"*
Introduction à [Viollet Le Duc, 1863/1978].

[Behem, 1501] Baltazar Behem
*"Codex Picturalis Baltasaris Behem"*
Manuscrit original, 1501-1506, conservé à la Jagellonian University de Cracovie (Pologne).

[Bellahsène et al, 1998] Zohra Bellahsène, Roland Ducournau
*"Vues en bases de données et points de vue en représentation des connaissances"*
Revue L'objet Volume 4 / n°3, pages 307 à 331, publié par Hermès éditions, 1998, ISSN1262-1137

[Benett, 1997] Gordon Benett
*"Oyster Grit"*

Intranet Design Magazine, 16 Janvier 1997 (http://idm.internet.com/features/perlism.shtml) et dans LAN Times Online, McGraw-Hill Publishers, http://www.lantimes.com/

[Bilgin, 1997] A.G Bilgin
*"An adaptable Model for a systematic approach to conservation works : An introductory study on GIS for the urban archaeological site of Bergama (Pergamon)"*
Informatique et conservation-restauration du patrimoine culturel, Actes des huitièmes journées de la SFIIC, 23-24 octobre 1997, pp211-220, ISBN 2-905430-10-9.

[Blair et al, 1991] Blair,Gallagher, Hutchison and Shepherd
*"Object-Oriented Languages, Systems and Applications"*
Published by Pitman Press ISBN 0-273-03132-5, 1991.

[Borne et al, 1999] Isabelle Borne, Nicolas Revault
*"Comparaison d'outils de mise en œuvre de design patterns"*
Revue L'objet Volume 5 / n°2, pages 243 à 266, publié par Hermès éditions, 1999, ISSN1262-1137

[Boudon, 1971] Philippe Boudon
*"Sur l'espace architectural, essai d'épistémologie de l'architecture"*
Dunod Editeur, collection Aspects de l'Urbanisme, 1971, 138p.

[Boudon, 1977] Philippe Boudon
*"Richelieu, ville nouvelle"*
Dunod Editeur 1977 ISBN 2-04-040270-1

[Bourdakis et al, 1999] V.Bourbakis, D.Charitos
*"Virtual Environment Design - defining a new direction for architectural education"*
in Architectural computing, proceedings Ecaade 1999, pp 403-409, ISBN 0 9523687 5 7

[Branki et al, 1997] C.Branki, A.Wallis, I.Aird, A.Bridges
*"Collaborative design objects as experts agents"*
in CAAD towards new design conventions, TU Białystok, 1997, pp 63-74, ISBN 83-86272-63-5

[Brenner et al, 1996] Steven E.Brenner, Edwin Aoki
*"Introduction to CGI/Perl"*
Publié par M&T books, 1996, ISBN 1-55851-478-3

[Bridges, 1999] A.Bridges
*"Progress ? What Progress ?"*
in Architectural computing, proceedings Ecaade 1999, pp 321-326, ISBN 0 9523687 5 7

[Brown et al, 1996] Brown, A., Rezgui, Y., Cooper, G., Yip, J. , Brandon, P.
*Promoting Computer Integrated Construction Through the Use of Distribution Technology*
ITCON Electronic Journal of Information Technology in Construction, Volume 1, 1996, ISSN 1400-6529.

[Budd, 1996] Timothy Budd
*"Introduction to Object-Oriented Programming"*
Edité par Addison-Wesley Longman, 1996 (seconde édition), ISBN 0201824191

[Burton et al, 1997] N.R Burton, M.E Hitchen, P.G. Bryan
*"Virtual Stonehenge: a fall from disgrace?"*
in Proceedings CAA97 (Computer Applications in Archaeology, Birmingham) 1997.

[Camara et al, 1997] L Camara, P Latorre
*"Information systems on heritage conservation"*
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Castellani, 1998] Xavier Castellani
*"Catégories d'objets d'un système d'objets"*
Revue L'objet Volume 4 / n°1, pages 45 à 72, publié par Hermès éditions, 1998, ISSN1262-1137

[Cha et al, 1999] Myung Yeol Cha, John Gero
*"Style learning: inductive generalisation of architectural shape patterns"*
in Architectural computing, proceedings Ecaade 1999, pp 629-644, ISBN 0 9523687 5 7

[Chattopadhyay, 1997] Swati Chattopadhyay
*"A critical history of architecture in a post-colonial world: A View from Indian History"*
Architronic Vol 6n1 may 1997 ISSN 1066-6516

[Chen et al, 1995] Jian Chen, Qiming Huang
*"Eliminating the impedance mismatch between relational systems and Object-Oriented programming languages"*
In *Database reengineering and interoperability*, proceedings 6th international Hong Kong computer society database workshop, Honk Kong, 1995.

[Choisy, 1899/1991] Auguste Choisy
"Histoire de l'architecture"
Ré-édition Inter-Livres 1991 de l'ouvrage original paru en 1899.

[Christiansen et al, 1999] Tom Christiansen, Nathan Torkington
"Perl en action"
Edité par O'Reilly, 1999, traduit par Sébastien Blondeel, ISBN 2-84177-077-X

[Clark et al, 1998] Tony Clark, Andy Evans
*"Foundations of the Unified Modeling Language"*
in Proceedings 2nd northern formal methods workshop, Ilkley, éd. Electronic Workshops in Computing, Springer-Verlag, 1998.

[Cohen, 1996] I.Cohen
*"CGI/Perl et Javascript"*
Editions Eyrolles, 1996, 294 pages, ISBN 2-212-08918-X

[Conway, 1997] Damian Conway
*"Advanced Object Oriented Techniques In C++ "*
Course Material Monash University , 1997,
Adressse : http://www.csse.monash.edu.au/~damian/Idioms/Topics/10.1.Persistence/html/text.html

[Conway, 2000] Damian Conway
*"Object Oriented Perl"*
Edité par Manning Publications , 2000, ISBN 1-884777-79-1

[Corbusier, 1958] Le Corbusier
*"Le Modulor"*
Editions de l'Architecture d'aujourd'hui, collection Ascoral, 1958

[Coste, 1997] Anne Coste"
"Le Modèle en architecture, entre rétrospective et prospective"
in Les cahiers de la recherche architecturale, n°40, Editions Paranthèses, 1997, ISBN2-86364-841-1

[Cuisenier, 1991] Jean Cuisenier
*"La maison rustique; logique sociale et composition architecturale"*
Presses Universitaires de France, 1991

[Czubinski et al, 1998a] J.Czubinski, P. Drap, I.Dudek, J.Y Blaise
*"Collaborative network tools for the architectural analysis in conservation research"*
CYBER-REAL DESIGN, 1998 23-25 April 1998 Bialystok, POLAND.

[Dalbera, 1997] J.P Dalbera
*"Des bases de données aux applications multimédia"*
Actes des 8èmes journées d'études de la SFIIC, Châlons sur saône, 23 -24 Octobre 1997, ISBN 2-905430-10-9.

[Dedieu, 1997] Olivier Dedieu
*"Java et l'informatique répartie"*
Note OD97A groupe SOR (Systèmes d'Objets Répartis) de l'INRIA.

[DeChampeaux et al, 1993] Dennis de Champeaux, Doug Lea, Penelope Faure
*"Object oriented system development"*
Edité par Addison-Wesley, 1993, ISBN 0-201-56355-X

[Démians d'Archimbaud, 1987] Gabrielle Démians d'Archimbaud
*"Rougiers Village médiéval déserté"*
Ministère de la culture et de la communication / imprimerie national 1987

[Dinkel, 1997] René Dinkel
"Encyclopédie du patrimoine"
Publié par les encyclopédies du patrimoine, Paris, 1997.

[Dmochowski, 1956] Zbigniew Dmochowski
"The architecture of Poland"
The polish research centre Limited, London. Distributed by ALMA Book Co. 9 Lenthal Place, Gloucester Road, London.

[Dodani, 1999] Mahesh Dodani
*"Rules are for fools, Patterns are for cool fools"*
JOOP (Journal of Object Oriented Programming) Octobre 1999, http://www.joopmag.com (SIGS Publications)

[Donath et al, 1997a] Dirk Donath, Frank Petzold
*"Towards a building information system based on computer-supported surveying system"*
in CAAD towards new design conventions, TU Białystok, 1997, ISBN 83-86272-63-5

[Donath et al ,1997b] Dirk Donath, Frank Petzold
"From Digital Building Surveying to an Information System"
ECAADE 97 in Vienna, September 1997

[Doneus et al, 1997] M.Doneus, A.Eder-Hinterleitner, W.Neubauer
*"Combination of geomagnetics and low-altitude aerial photogrammetry in archaeology"*
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Drap et al, 1995] Pierre DRAP, Jean-Yves BLAISE
*"PAROS. De la photogrammétrique à la synthèse d'image. Le cas du forum antique de la ville d'Arles"*
Actes des Troisièmes journées de l'AFIG Association Française d'Informatique Graphique 1995, Marseille, France, 22 - 24 Novembre 1995, pages 303 à 310.

[Drap, 1997] Pierre Drap
*"Contribution au projet PAROS. Photogrammétrie et modèles architecturaux."*
Thèse de troisième cycle de l'Université d'Aix-Marseille III, mars 1997.

[Drap et al, 1998b] P. Drap, I.Dudek, J.Y Blaise
*"Java collaborative interface for architectural simulations A case study on wooden ceilings of Kraków"*
Międzynarodowe Sympozjum Konserwatorskie,V Teoria i praktyka w ochronie zabytków architektury, zespołów i miejsc historycznych , 22-24 November 1998, Kraków, POLAND.

[Drap et al, 2000] P.Drap , P. Grussenmeyer
*"A digital photogrammetric workstation on the WEB"*
In Journal of Photogrammetry and Remote Sensing 55 (2000), pp.48-58.Avril 2000, Elsevier.

[Drap et al, 1999a] P. Drap, J.Y Blaise, P. Grussenmeyer
*"A photogrammetric survey using knowledge representation on the arpenteur web-based photogrammetric workstation"*
XVII CIPA international symposium wg3 - simple methods for architectural photogrammetry Olinda, brazil, october 3-6, 1999

[Drap et al, 1999b] P. Drap, I.Dudek, J.Y Blaise
*"An architectural model compiler dedicated to archaeological hypothesis. An experiment on Krakow's kramy Bogate"*
HCP'99, Human Centered Processes 22 - 24 September 1999 Brest, France.

[Ducournau, 1997] Roland Ducournau
*"La compilation de l'envoi de message dans les langages dynamiques"*
Revue L'objet Volume 3 / n°3, publié par Hermès éditions, septembre 1997, ISSN1262-1137

[Ducournau et al, 1998] Roland Ducournau, Jérôme Euzenat, Gérald Masini, Amedeo Napoli
*"Langages et modèles à objets. Etat des recherches et perspectives"*
Publié par l'INRIA (Institut National de Recherche en Informatique et en Automatique), 1998, ISBN 2-7261-1131-9

[Dudek et al, 1999a] I.Dudek, J.Y Blaise
*"IT applications for architectural intervention and documentation in monuments' ensembles"*
Międzynarodowe Sympozjum Konserwatorskie,V Teoria i praktyka w ochronie zabytków architektury, zespołów i miejsc historycznych ,*22-24 November 1999, Kraków, POLAND.*

[Dudek et al, 1999b] I.Dudek, J.Y Blaise
*"SOL : Spatial and historical web-based interface for On Line architectural documentation of Kraków's Rynek Glówny "*
Turning to 2000,17th conference of eCAADe, September 15-18, 1999 The University of Liverpool, UK

[Durnota, 1995] Bohdan Durnota
*"Complex object models of natural systems"*
Proceedings International conference OOSimulation '95, Las Vegas, USA, 1995.

[Eastman, 1994] Charles M. Eastman
*"A data model for design knowledge."*
Automation in construction, Elsevier juillet 1994

[Ekholm, 1996] Ekholm, A.

*"A Conceptual Framework for Classification of Construction Works"*
ITCON Electronic Journal of Information Technology in Construction, Volume 1, 1996, ISSN 1400-6529.

[Englander, 1997] Robert Englander
*"Java Beans - guide du programmeur"*
Editions O'Reilly, édition française traduite par Bonnie Cupac, 1997, ISBN 1-84177-038-9

[Euzenat et al, 1995] Jérôme Euzenat, François Rechenmann
*"Shika, 10 ans c'est Tropes ?"*
Actes des 2ème journées Langages et Modèles à objets, Nancy, 1-3 octobre 1995, pp13-34

[Evans et al, 1998] Andy Evans, Robert France, Kevin Lano, Bernhardt Rumpe
*"The UML as a Formal Modeling Notation"*
in Proceedings UML' 98 1st Int workshop on UML, Mulhouse, France, 3-4 juin 1998, édité par Springer-Verlag LNCS 1618

[Flanagan, 1997] David Flanagan
"JAVA in a nutshell"
O'Reilly Publishing, deuxième édition, 1997, ISBN 1-56592-262-X

[Florenzano et al, 1996a] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"PAROS, Photogrammétrie Architecturale et Restitution par Outils de Synthèse"*
Revue de CFAO et d'informatique graphique. Volume 11 - N°4/1996,pages 345 à 361.

[Florenzano et al, 1996b] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"Photogrammétrie et modèles architecturaux"*
ISPRS: XVIIIème Congrés de la société Internationale de Photogrammétrie et de télédétection, Vienne, Autriche, juillet 1996.
Proceedings: International archives of photogrammetry and remote sensing, volume XXXI, tome B5, pages 167 a 172.

[Florenzano et al, 1997a] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"PAROS, Modèles objet appliques à l'architecture construite"*
L'OBJET, Logiciel, Bases de Données, Réseaux. Numéro 1, volume 3, pages 29 à 52. Revue publiée par éditions Hermès, avril 1997.

[Florenzano et al, 1997b] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"Photogrammetry and knowledge representation, restitution of archeological hypothesis on the Bigot model of ancient Rome"*
CIPA, Photogrammetry in Architecture, Archeology and Urban Conservation, Göteborg, Suède, les 1,2 et 3 octobre 1997.
Proceedings: Intenational Archives of Photogrammetry and remote sensing, volume XXXII, part 5C1B, pages 90 a 96.

[Florenzano et al, 1998] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"From photogrammetrical survey to architectural models monitoring through a virtual reality interface"*
Complex Systems Intelligent Systems & Interfaces du 26 au 28 mai 1998 Nimes, France.
Publié dans La lettre de l'IA, numéro 134-135-136 mai-août 1998 pages 168, 175.

[Froese, 1992] Thomas Michael Froese
*"Integrated computer-aided project management through standard object-oriented models"*
Thèse de doctorat du département génie civil de l'université de Stanford, soutenue en juin 1992, Université de Stanford, USA.

[Gaillard, 1994] Fabrice Gaillard
*Thèse de doctorat sur la modélisation des connaissances et l'utilisation de base de données objet en productique*
Soutenue le 1er Décembre 1994 à l'UT Compiègne, France.

[Gao, 1997] Libing Gao
*"Archaeological Prospection with GPR Approaches : case studies in Xian and Shangqiu, China"*
in Proceedings CAA97 (Computer Applications in Archaeology, Birmingham) 1997.

[Garshol, 1999] Lars Marius Garshol
*"Introduction to XML"*
Note de travail Department of Informatics University of Oslo (Norvège).

[Geroimenko, 1999] V.Geroimenko
*"Online Photorealistic VR with Interactive Architectural Objects"*
in Architectural computing, proceedings Ecaade 1999, pp 414-417, ISBN 0 9523687 5 7

[Ginouvès et al, 1985] René Ginouvès, Roland Martin
*"Dictionnaire méthodique de l'architecture grecque et romaine"*
Tome 1: Matériaux, techniques de construction, techniques et formes du décor
Ecole française de Rome / Ecole française d'Athènes, 1985
René Ginouvès

Tome 2: Eléments constructifs: supports, couvertures, aménagements intérieurs
Ecole française de Rome / Ecole française d'Athènes, 1992
Tome 3: Espaces architecturaux, bâtiments et ensembles
Ecole française de Rome / Ecole française d'Athènes, 1998

[Gromort, 1960] Georges Gromort
*"Choix d'éléments empruntés à l'architecture classique - Parallèle d'ordres grecs et romains et application des ordres"*
Vincent, Fréal et Cie éditeurs, Paris, 6ème édition, 1960, 80 planches.

[Gucek et al, 1997] M.Gucek, M.Janezic, M.Piccolo, M.Stokin
*"Koper Platea Communis: beneath the medieval square. New approaches in Slovenian conservation policy"*
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Guild et al, 1988] R.Guild, J.Guyon, L.Rivet, M.Vecchione
*"Saint Sauveur d'Aix en Provence. La Cathédrale et le Baptistère"*
Congrés Archéologique de France, 143ème session, 1985, Pays d'Aix, édit. Société Française d'Archéologie / CNRS, 1988, pp 17-64.

[Hanke et al, 1997] K.Hanke, M.A-B Ebrahim
*"A low-cost 3D-Measurement tool for architectural and archaeological applications"*
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Hanrot, 1989] S.Hanrot
*"Une base de connaissances architecturales pour un système de CAO intelligent"*
Thèse de troisième cycle de l'Université d'Aix-Marseille III, mars 1989.

[Harold, 1997] Eliotte Rusty Harold
*"Programmation réseau avec JAVA"*
Edité par O'Reilly Publishing, édition française, traduction de Manuel Makarévitch, 1997, ISBN 2-84177-024-6

[Hébrard, 1897] A.Hébrard
*"Architecture"*
P.Vicq-Dunod et Cie, éditeurs, Paris, 1897.

[Hillier,1996] Bill Hillier
*"The introduction to space is a machine"*
space syntax publications

[Hillier et al,1997] Bill Hillier, Alan Penn, Tim Stonor and Mark David Major
*"Housing design and the virtual community"*
in Proceedings 19th International Conference on Making cities livable, Charleston, 1997

[Hitz et al, 1998] Martin Hitz, Gerti Kappel
*"Developing with UML – Some pitfalls and workarounds"*
in Proceedings UML'98 1st Int workshop on UML, Mulhouse, France, 3-4 juin 1998, édité par Springer-Verlag LNCS 1618

[Huitema, 1994] C.Huitema
*"Le routage dans l'internet"*
Editions Eyrolles, 407 pages, 1994, ISBN 2-212-08902-3

[Ioannidis et al, 1999] C.Ioannidis, H.Chlepa
*"Spatial information system for the geometric documentation and restoration studies of monuments: an application to the wall of ancient Messene"*
in Proceedings ISPRS WG V/5 Workshop, Thessaloniki, Grèce, 7-9 juillet 1999.

[Israel, 1997] Marc Israel
*"Administration SQLServer 6.5. Sécurité, Optimisation, réplication, interface Web"*
Editions Eyrolles, 1997

[Juhasz et al ,1999] P.Juhasz, Z.Kiss, M.Szoboszlai
*"Drawing's dimensions"*
in Architectural computing, proceedings Ecaade 1999, pp 498-502, ISBN 0 9523687 5 7

[Jamka,1963] R.Jamka
*"Kraków w pradziejach"*
Narodowy Zakład Imiena Ossolińskich, Wrocław-Warszawa-Kraków 1963

[Jamroz,1983] Józef St. Jamroz
*"Mieszczańska kamienica Krakowska"*
Wyd. Literackie Kraków 1983 ISBN 83-08-01018-X

[Kappel et al, 1998] Gerti Kappel, Birgit Schroder
*"Distributed light-weight persistence in Java- A tour on RMI- and CORBA- based solutions"*
in Proceedings 9th International Conference on Database and Expert Systems Applications (DEXA 98), Vienne, Autriche, 24-28 Août 1998, pp 411-424, ISSN 0302-9743

[Lammari et al, 1999] Nadira Lammari, Régine Laleau, Mireille Jouve
*"Processus d'optimisation conceptuelle d'un schéma orienté objet"*
Revue L'objet Volume 5 / n°1, "best of OOIS", publié par Hermès éditions, 1999, ISSN1262-1137, ISBN 2-86601-750-1

[Larpin, 1988] Dominique Larpin
*"Le Château de Peyrolles-en-Provrence"*
Congrés Archéologique de France, 143ème session, 1985, Pays d'Aix, édition Société Française d'Archéologie / CNRS, pp 254-276.

[Laurie et al, 1997] B.Laurie, P.Laurie
*"Apache, Installation et mise en œuvre"*
Traduit par R.Debonne, éditions O'reilly, 294 pages, 1997, ISBN 2-84177-036-2

[Lea, 1997] Doug Lea
*"Christopher Alexander: an introduction for Object-Oriented Designers"*
Critique en ligne : http://g.oswego.edu/dl/ca/ca/ca.html
Publié dans "*Patterns Handbook*" par L.Rising, editions Cambridge University Press, 1998, ISBN 0-521-64818-1

[Lebahar,1983] Jean Charles Lebahar
*"Le dessin d'architecte. Simulation graphique et réduction d'incertitude"*
Editions Paranthèses, 1983.

[Leclercq, 1998] P.leclercq
*"A tool supporting the architectural sketch"*
in La lettre de l'IA n° 134-135-136, Proceedings conférence Complex Sytems Intelligent Systems and Interface, Nîmes 26-28 mai 1998, ISBN 2-910085-21X

[Lecomte et al, 1996] C.Lecomte, T.Leduc
*"Programmation Javascript"*
Editions Eyrolles, 1996, 318 pages, ISBN 2-212-08937-6

[Levoy, 1999] Marc Levoy
*"The Digital Michelangelo Project"*
in Proceedings 2nd International Conference on 3D Digital Imaging and Modeling, Ottawa, Canada, October 5-8, 1999

[Line, 1997] Lars Line
*"Virtual Engineering Teams: Strategy and Implementation"*
ITCON Electronic Journal of Information Technology in Construction, Volume 2, 1997, ISSN 1400-6529.

[Loomis, 1995] Mary E.S. Loomis
*"Object databases"*
Addison Wesley publishing company, 1995, ISBN 0-201-56341-X

[Łukacz,1998] Marek Łukacz
*"Metrologia i pierwsza faza zabudowany staromiejskich blokóz lokacyjnego Krakowa"*
Proceedings of the International conference on conservation Kraków 1998, Tom 3, pp 92-100, ISBN 83-7242-072-6

[Major et al,1997] Mark David Major and Tim Stonor
*"Designing for context : the use of space syntax as an interactive design tool in urban developments"*
in Proceedings 14th Inter-schools Conference on Development, Edinburgh, 1997

[Masini et al, 1989] Gérald Masini, Amedeo Napoli, Dominique Colnet, Daniel Leonard, Karl Tombre
*"Les langages à objets. Langages de classes, langages de frames, langages d'acteurs"*
Publié par InterEditions, 1989 (deuxième tirage corrigé 1990) avec le concours du Centre National des Lettres

[Maughan et al, 1998] Glenn Maughan, Jon Avotins
"A Meta-model for Object-oriented Reengineering and Metrics Collection"
Eiffel Liberty Journal, Vol 1, n°4, 1998

[Menzies et al, 1995] Tim Menzies, Philip Haynes
*"Do we really need encapsulation ?"*
in Proceedings TOOLS (Technology Of Object-Oriented Languages And Systems) 1995 conference Melbourne, Australia, 27 November - 30 November 1995

[Nierstrasz, 1989] Oscar Nierstrasz
*"A survey of object oriented concepts"*
Publié dans "Object oriented concepts, databases and applications", éditions ACM Press Addison Wesley , 1989.

[Noël, 1994] Pierre Noël
*"Technologie de la pierre de taille. Dictionnaire des termes couramment employés dans l'extraction, l'emploi et la conservation de la pierre de taille"*
Editeur SEBTP, Paris , 1994 (deuxième édition).

[Odlanicki, 1978] M.Odlanicki-Poczobutt Z.Taranczewska-Białek
*"Plan Kołłątajowski miasta krakowa",*
Kraków, Zeszyty naukowe AGH, 1978.

[Ogleby, 1997] Cliff Ogleby
*"From rubble to virtual reality, a reconstruction of the ancient city of Ayutthaya using modern photogrammetric techniques"*
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Orfali et al, 1996] R.Orfali, D.Harkey, J.Edwards
*"Objets répartis"*
Traduit par F.Leroy, J.P Gout, R.Larsen. International Thomson Publishing France, 618 pages, 1996, ISBN 284180-043-1

[Ousssalah, 1997] Chabane Oussalah (et alii)
*"Ingéniérie objet"*
Publié par InterEditions, 1997, ISBN 2-7296-0642-4

[Oxman, 1999] Rivka Oxman
*"Visual Emergence in Creative Collaboration"*
in Architectural computing, proceedings Ecaade 1999, pp 357-363, ISBN 0 9523687 5 7

[Palladio, 1738/1965] Andrea Palladio
*"The four books of architecture"*
Dover Publications, New York, 1965 (réimpression de l'ouvrage publié par Issac Ware en 1738)

[Péroche et al, 1988] Bernard Péroche, Jacqueline Argence, Djamchid Ghazanfarpour, Dominique Michelucci
"La synthèse d'images"
Hermès Editions, 295 pages, 1988 - ISBN : 2-86601-112-0.

[Péroche et al, 1997] Bernard Péroche, Jacqueline Argence, Djamchid Ghazanfarpour, Dominique Michelucci
"Informatique graphique 2e édition - Méthodes et modèles"
Hermès Editions, 416 pages, 1997 - ISBN : 2-86601-652-1.

[Pérouse De Montclos, 1988] Jean Marie Pérouse De Montclos
*"Architecture vocabulaire - Principe d'analyse scientifique"*
Imprimerie Nationale 1972-88

[Plaindoux et al, 1998] Didier Plaindoux, Jean-Paul Bodeveix, Christian Percebois
*"Types versus Classes"*
Revue L'objet Volume 4 / n°1, publié par Hermès éditions, mars 1998, ISSN1262-1137

[Pollefeys et al, 1999] M.Pollefeys, R.Koch, M.Vergauwen, L.Van Gool
*"An automatic method for acquiring 3D models from photographs: application to an archaeological site"*
in Proceedings ISPRS WG V/5 Workshop, Thessaloniki, Grèce, 7-9 juillet 1999.

[Potier et al, 1998] S.Potier, JL Maltret, J.Zoller
*"L'infographie : une aide a la formulation d'hypothèses archéologiques."*
in Digital Design Studios: Do Computers Make A Difference? Proceedings ACADIA 98, T. Seebohm and S. V. Wyk, eds., ville de Québec, Québec, Canada.

[Rahayu et al, 1999] W.Rahayu, E.Chang
*"Composites indices as a mechanism for transforming multi-level composite objects into relational databases"*
Revue L'objet Volume 5 / n°1, "best of OOIS", publié par Hermès éditions, 1999, ISSN1262-1137, ISBN 2-86601-750-1

[Randall et al, 1993] Randall Davis, Howard Shrobe,Peter Szolovits.
*"What is a Knowledge Representation?"*
AI Magazine, 14 (1):17-33, 1993.

[Rezgui et al, 1998]Yacine Rezgui, Grahame Cooper
*"A proposed open infrastructure for construction project document sharing"*
ITCON Electronic Journal of Information Technology in Construction, Volume 3, 1998, ISSN 1400-6529.

[Rising, 1996] Linda Rising
*"The road, Christopher Alexander, and good software design"*
Proceedings OOPSLA '96, (Eleventh Annual ACM Conference on Object-Oriented Programming Systems, Languages and Applications) 6-10 October 1996, San Jose, USA.

[Roberts, 1999] A.Roberts
*"Virtual site planning"*
in Architectural computing, proceedings Ecaade 1999, pp 442-448, ISBN 0 9523687 5 7

[Roehl et al, 1997] B.Roehl, J.Couch, C.Reed-Ballreich, T.Rohaly, G.Brown
*"Late night VRML 2.0 with Java"*
Ziff -Davis Press / MacMillan Computer Publishing Editeurs, 710 pages, 1997, ISBN 1-56276-504-3

[Rolle, 1998] Jean-François Rolle
*"Mon ordinateur voit double"*
Revue "flash Informatique" du Service informatique central de l'EPFL (Ecole polytechnique Fédérale de Lausanne, Suisse), numéro spécial du 1er septembre 1998 (FI -SP 1998).

[Rosenblatt, 1995] Bill Rosenblatt
*"Persistently C++"*
Edité par Web Publishing Inc., SunWorld / Advanced Systems Magazine, Avril 1995
À l'adresse : http://sunsite.uakom.sk/sunworldonline/asm-04-1995/asm-04-client.html

[Schneider, 1994] Daniel K. Schneider
*Thèse de doctorat sur la modélisation de la démarche du décideur politique dans la perspective de l'intelligence artificielle* présentée à la faculté des sciences économiques et sociales de Genève, 1994.

[Seinturier et al ,1998] Lionel Seinturier, Laurence Duchien, Gérald Florin
*"CAOLAC : un protocole à méta-objets pour la synchronisation d'objets concurrents"*
Revue L'objet Volume 4 / n°3, publié par Hermès éditions, septembre 1998, ISSN1262-1137

[Sintès et al, 1989] Claude Sintès, Jean-Maurice Rouquette
*"Arles antique, Monuments et sites"*
Edition Imprimerie Nationale, 1989.

[Snyder, 1998] Alan Snyder
*"Object-Oriented language design"*
in Eiffel Liberty Journal - Vol.1, No.1, 1998 (http://www.elj.com/elj/).

[StAubin, 1992] Jean-Paul Saint Aubin
*"Le relevé et la représentation de l'architecture"*
Service de l'inventaire général, ed : Documents & méthodes, 1992.

[StAubin, 1996] Jean-Paul Saint Aubin
*"Le relevé des grottes et de leur décor, leur restitution en informatique"*
Conservation des grottes ornées (ouvrage collectif dirigé par J.Brunet et J. Vouvé) CNRS Editions, Paris, 1996

[Stenvert, 1991] Ronald Stenvert
*"Constructing the past: computer-assisted Architectural-Historical Research"*
Thèse de doctorat de l'université d'Utrecht (Pays-bas), décembre 1991.

[Stroustrup, 1996] Bjarne Stroustrup
*"Le langage C++"*
Thomson Publishing, deuxième édition, 1996, traduit par Hervé Soulard, ISBN 2-84180-079-2

[Sunyé et al, 1997] Gerson Suyé, Houari A.Sahraoui, B.Lesueur, G.Blain
*"Chroniques d'implémentation d'un méta-outil en Smalltalk"*
Revue L'objet Volume 3 / n°4, publié par Hermès éditions, décembre 1997, ISSN1262-1137

[Tajchman, 1989] Jan Tajchman
*"Stropy drewniane w polsce. Propozyca systematyki"*
Ośrodek dokumentacji zabytków, Warszawa 1989.

[Tanenbaum, 1997] A.Tanenbaum
*"Réseaux"*
Traduit par J.A Hernandez et R.Joly. Prentice Hall / Dunod éditeurs, 3ème édition, 791 pages, 1997, ISBN 2-10-004315-3

[Tołwiński, 1939] T.Tolwiński

*"Urbanistyka"*
Druk. Anczyca i Sp, T.1, Warszawa 1939.

[Tomkowicz, 1907] S.Tomkowicz,
"*Plan rynku Krakowskiego z roku 1787*",
Kraków, Rocznik krakowski, 1907.

[Tsichritzis et al, 1992] Dennis Tsichritzis, Oscar Nierstrasz, Simon Gibbs
*"Beyond Objects: Objects"*
in International Journal of Intelligent and Cooperative Information Systems (IJICIS), vol 1, n°1, pp 43-60, 1992.

[Tunçer et al, 1999] Bige Tunçer , Rudi Stouffs
*"Computational richness in the representation of architectural languages"*
in Architectural computing, proceedings Ecaade 1999, pp 603-610, ISBN 0 9523687 5 7

[Turk et al, 1993] Turk, @, D.J. Vanier.
*"Classification systems in object oriented modelling of buildings"*
Proceedings DMMI Conference, Maribor, ISBN 86-345-0100-0, Slovenia, 1993.

[Tweed, 1997] C.Tweed
*"The predominance of the visual in Computer Aided Architectural Design"*
in CAAD towards new design conventions, TU Białystok, 1997, pp 269-285, ISBN 83-86272-63-5

[Van Den Heuven, 1998] Frank A. Van Den Heuden
*"3D Reconstruction from a single image using geometric constraints"*
ISPRS Journal of Photogrammetry and Remote Sensing n° 53, 1998, Elsevier Editeur.

[Viollet Le Duc, 1863/1977] Eugène Viollet Le Duc
*"Entretiens sur l'architecture"*
Edition Pierre Mardaga 1977, réédition du texte écrit entre 1863 et 1872 par Eugène Viollet Le Duc.

[Viollet Le Duc, 1854/1996] Eugène Viollet Le Duc
*"Encyclopédie Médiévale"*
Edition Bibliothèque de l'image, 1996., ouvrage basé sur le *dictionnaire raisonné de l'architecture* que Viollet Le Duc a rédigé entre 1854 et 1868

[Viollet Le Duc, 1877/1978] Eugène Viollet Le Duc
"Histoire d'un hôtel de ville et d'une cathédrale"
Edition Pierre Mardaga 1978, réédition de l'ouvrage original édité en 1877.

[Vitruve, 1684/1988] Vitruve,
*"Les dix livres d'Architecture"*
Traduction de Claude Perrault 1684, édition Pierre Mardaga 1988.

[Watanabe, 1994] S. Watanabe
*"Knowledge integration for architectural design"*
Automation in construction, Elsevier juillet 1994

[Whiting et al, 1997] David Whiting, Steve Nickerson
*"Computer aided recording tools automate the creation of a site information system"*
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Wu et al, 1995] Xindong Wu, Sita Ramakrishnan, Heinz Schmidt
*"Knowledge objects"*
Revue Informatica (International journal of computing and informatics) vol 19, n°4, 1995.

[Wu, 1996] Xindong Wu
*"A comparison of Objects with Frames and OODBs"*
in Object Currents Journal, Volume 1, number 1, 1996. SIGS Publications http://www.sigs.com/objectcurrents

[Zin, 1966] Wiktor Zin
*"Kronika prac i problemów konserwatorskich krakowa za rok 1965"*
Rocznik krakowski, tome XXXVIII, 1966, Kraków.

## 7.2. BIBLIOGRAPHIE THEMATIQUE

Dans cette section les différentes références bibliographiques citées dans la partie Etat de l'Art sont reprises thème par thème, dans l'ordre d'apparition des sous-chapitres :
- *Patrimoine et problématique de la conservation*
- *Représentation des connaissances et approche objet*
- *Gestion de données et interfaces Web*
- *Informatique graphique*
- *Interfaces pour le réseau Internet*

Nous présentons également à la fin de cette section la liste de nos publications.

### 7.2.1. Patrimoine et problématiques de la conservation

[Adam, 1989] Jean-Pierre Adam
"La construction romaine, Matériaux et techniques"
Édition A. et J. Picard, 82, rue Bonaparte, 75006 Paris, 1980.

[Alkhoven, 1993] Patricia Alkhoven
"The changing image of the city. A study of the transformation of the townscape using Computer assisted Design and visualisation techniques"
Thèse de doctorat de l'université d'Utrecht (Pays-Bas), décembre 1993.

[Apers et al, 1979] Jan Apers, Alfons Hoppenbrouwers, Jos Vanderbreeden
"Inventaire d'urgence du patrimoine architectural de l'agglomération Bruxelloise"
Sint-Lukasarchief, Paleizenstraat 70, 1030 Brussels, 1979.

[Baculo Giusti et al, 1995] A. Baculo Giusti, L.Bucci de Santis, A.di Luggo, R.Florio, F.Rino
"Napoli Citta' in vista"
Université de Naples Federico II, edité par Electa Napoli 1995

[Baletti et al, 1999] C.Baletti, F.Guerra, L.Pilot
"Plan and survey for the recovery and the care of an ancient abandoned nucleus: the castle of Campo of Brenzone"
in Proceedings ISPRS WG V/5 Workshop, Thessaloniki, Grèce, 7-9 juillet 1999.

[Barberot, 1922] E.Barberot
"Aide mémoire de l'architecte et du constructeur"
Librairie Polytechnique Ch.Béranger, Paris, 1922, seconde édition (première édition 1914).

[Beckaert, 1977] Gert Beckaert
*"Introduction à l'édition de 1977"*
Introduction à [Viollet Le Duc, 1863/1977].

[Behem, 1501] Baltazar Behem
"Codex Picturalis Baltasaris Behem"
Manuscrit original, 1501-1506, conservé à la Jagellonian University de Cracovie (Pologne).

[Boudon, 1971] Philippe Boudon
"Sur l'espace architectural, essai d'épistémologie de l'architecture"
Dunod Editeur, collection Aspects de l'Urbanisme, 1971, 138p.

[Boudon, 1977] Philippe Boudon
"Richelieu, ville nouvelle"
Dunod Editeur 1977 ISBN 2-04-040270-1

[Burton et al, 1997] N.R Burton, M.E Hitchen, P.G. Bryan
"Virtual Stonehenge: a fall from disgrace?"
in Proceedings CAA97 (Computer Applications in Archaeology, Birmingham) 1997.

[Camara et al, 1997] L Camara, P Latorre
"Information systems on heritage conservation"
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Cha et al, 1999] Myung Yeol Cha, John Gero
"Style learning: inductive generalisation of architectural shape patterns"

in Architectural computing, proceedings Ecaade 1999, pp 629-644, ISBN 0 9523687 5 7

[Chattopadhyay, 1997] Swati Chattopadhyay
"A critical history of architecture in a post-colonial world: A View from Indian History"
Architronic Vol 6n1 may 1997 ISSN 1066-6516

[Choisy, 1899/1991] Auguste Choisy
"Histoire de l'architecture"
Ré-édition Inter-Livres 1991 de l'ouvrage original paru en 1899.

[Corbusier, 1958] Le Corbusier
"Le Modulor"
Editions de l'Architecture d'aujourd'hui, collection Ascoral, 1958

[Coste, 1997] Anne Coste"
"Le Modèle en architecture, entre rétrospective et prospective"
in Les cahiers de la recherche architecturale, n°40, Editions Paranthèses, 1997, ISBN2-86364-841-1

[Cuisenier, 1991] Jean Cuisenier
"La maison rustique; logique sociale et composition architecturale"
Presses Universitaires de France, 1991

[Démians d'Archimbaud, 1987] Gabrielle Démians d'Archimbaud
"Rougiers Village médiéval déserté"
Ministère de la culture et de la communication / imprimerie national 1987

[Dinkel, 1997] René Dinkel
"Encyclopédie du patrimoine"
Publié par les encyclopédies du patrimoine, Paris, 1997.

[Dmochowski, 1956] Zbigniew Dmochowski
"The architecture of Poland"
The polish research centre Limited, London.
Distributed by ALMA Book Co. 9 Lenthal Place, Gloucester Road, London.

[Donath et al, 1997] Dirk Donath, Frank Petzold
"Towards a building information system based on computer-supported surveying system"
in CAAD towards new design conventions, TU Białystok, 1997, ISBN 83-86272-63-5

[Doneus et al, 1997] M.Doneus, A.Eder-Hinterleitner, W.Neubauer
"Combination of geomagnetics and low-altitude aerial photogrammetry in archaeology"
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Drap, 1997] Pierre Drap
"Contribution au projet PAROS. Photogrammétrie et modèles architecturaux."
Thèse d'informatique de l'Université d'Aix-Marseille III, mars 1997.

[Gao, 1997] Libing Gao
"Archaeological Prospection with GPR Approaches: case studies in Xian and Shangqiu, China"
in Proceedings CAA97 (Computer Applications in Archaeology, Birmingham) 1997.

[Ginouvès et al, 1985] René Ginouvès, Roland Martin
"Dictionnaire méthodique de l'architecture grecque et romaine"
Tome 1: Matériaux, techniques de construction, techniques et formes du décor
Ecole française de Rome / Ecole française d'Athènes, 1985
René Ginouvès
Tome 2: Eléments constructifs: supports, couvertures, aménagements intérieurs
Ecole française de Rome / Ecole française d'Athènes, 1992
Tome 3: Espaces architecturaux, bâtiments et ensembles
Ecole française de Rome / Ecole française d'Athènes, 1998

[Gucek et al, 1997] M.Gucek, M.Janezic, M.Piccolo, M.Stokin
"Koper Platea Communis: beneath the medieval square. New approaches in Slovenian conservation policy"
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Guild et al, 1988] R.Guild, J.Guyon, L.Rivet, M.Vecchione
"Saint Sauveur d'Aix en Provence. La Cathédrale et le Baptistère"
Congrés Archéologique de France, 143ème session, 1985, Pays d'Aix,
Edition Société Française d'Archéologie / CNRS, pp 17-64.

[Hanke et al, 1997] K.Hanke, M.A-B Ebrahim
"A low-cost 3D-Measurement tool for architectural and archaeological applications"
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Hébrard, 1897] A.Hébrard
"Architecture"
P.Vicq-Dunod et Cie, éditeurs, Paris, 1897.

[Hillier,1996] Bill Hillier
"The introduction to space is a machine"
space syntax publications

[Hillier et al,1997] Bill Hillier, Alan Penn, Tim Stonor and Mark David Major
"Housing design and the virtual community"
in Proceedings 19th International Conference on Making cities livable, Charleston, 1997

[Ioannidis et al, 1999] C.Ioannidis, H.Chlepa
"Spatial information system for the geometric documentation and restoration studies of monuments: an application to the wall of ancient Messene"
in Proceedings ISPRS WG V/5 Workshop, Thessaloniki, Grèce, 7-9 juillet 1999.

[Jamka,1963] R.Jamka
"Kraków w pradziejach"
Narodowy Zakład Imiena Ossolińskich, Wrocław-Warszawa-Kraków 1963

[Jamroz,1983] Józef St. Jamroz
"Mieszczańska kamienica Krakowska"
Wyd. Literackie Kraków 1983 ISBN 83-08-01018-X

[Larpin, 1988] Dominique Larpin
"Le Château de Peyrolles-en-Provrence"
Congrés Archéologique de France, 143ème session, 1985, Pays d'Aix,
Edition Société Française d'Archéologie / CNRS, pp 254-276.

[Lebahar,1983] Jean Charles Lebahar
"Le dessin d'architecte. Simulation graphique et réduction d'incertitude"
Editions Paranthèses, 1983.

[Leclercq, 1998] P.leclercq
*"A tool supporting the architectural sketch"*
in La lettre de l'IA n° 134-135-136, Proceedings conférence Complex Sytems Intelligent Systems and Interface, Nîmes 26-28 mai 1998, ISBN 2-910085-21X

[Levoy, 1999] Marc Levoy
"The Digital Michelangelo Project"
in Proceedings 2nd International Conference on 3D Digital Imaging and Modeling, Ottawa, Canada,
October 5-8, 1999

[Łukacz,1998] Marek Łukacz
"Metrologia i pierwsza faza zabudowany staromiejskich blokóz lokacyjnego Krakowa"
Proceedings of the International conference on conservation Kraków 1998,
Tom 3, pp 92-100, ISBN 83-7242-072-6

[Major et al,1997] Mark David Major and Tim Stonor
"Designing for context: the use of space syntax as an interactive design tool in urban developments"
in Proceedings 14th Inter-schools Conference on Development, Edinburgh, 1997

[Noël, 1994] Pierre Noël
"Technologie de la pierre de taille. Dictionnaire des termes couramment employés dans l'extraction, l'emploi et la conservation de la pierre de taille"
Editeur SEBTP, Paris , 1994 (deuxième édition).

[Odlanicki, 1978] M.Odlanicki-Poczobutt Z.Taranczewska-Białek
"Plan Kołłątajowski miasta krakowa",
Kraków, Zeszyty naukowe AGH, 1978.

[Ogleby, 1997] Cliff Ogleby
"From rubble to virtual reality, a reconstruction of the ancient city of Ayutthaya using modern photogrammetric techniques"
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Palladio, 1738/1965] Andrea Palladio
"The four books of architecture"
Dover Publications, New York, 1965 (réimpression de l'ouvrage publié par Issac Ware en 1738)

[Pérouse De Montclos, 1988] Jean Marie Pérouse De Montclos
"Architecture vocabulaire - Principe d'analyse scientifique"
Imprimerie Nationale 1972-88

[Pollefeys et al, 1999] M.Pollefeys, R.Koch, M.Vergauwen, L.Van Gool
"An automatic method for acquiring 3D models from photographs: application to an archaeological site"
in Proceedings ISPRS WG V/5 Workshop, Thessaloniki, Grèce, 7-9 juillet 1999.

[StAubin, 1992] Jean-Paul Saint Aubin
"Le relevé et la représentation de l'architecture"
Service de l'inventaire général, ed : Documents & méthodes, 1992.

[StAubin, 1996] Jean-Paul Saint Aubin
"Le relevé des grottes et de leur décor, leur restitution en informatique"
Conservation des grottes ornées (ouvrage collectif dirigé par J.Brunet et J. Vouvé) CNRS Editions, Paris, 1996

[Stenvert, 1991] Ronald Stenvert
"Constructing the past: computer-assisted Architectural-Historical Research"
Thèse de doctorat de l'université d'Utrecht (Pays-bas), décembre 1991.

[Tajchman, 1989] Jan Tajchman
"Stropy drewniane w polsce. Propozyca systematyki"
Ośrodek dokumentacji zabytków, Warszawa 1989.

[Tołwiński, 1939] T.Tolwiński
"Urbanistyka"
Druk. Anczyca i Sp, T.1, Warszawa 1939.

[Tomkowicz, 1907] S.Tomkowicz,
"*Plan rynku Krakowskiego z roku 1787*",
Kraków, Rocznik krakowski, 1907.

[Tunçer et al, 1999] Bige Tunçer , Rudi Stouffs
"Computational richness in the representation of architectural languages"
in Architectural computing, proceedings Ecaade 1999, pp 603-610, ISBN 0 9523687 5 7

[Van Den Heuven, 1998] Frank A. Van Den Heuden
"3D Reconstruction from a single image using geometric constraints"
ISPRS Journal of Photogrammetry and Remote Sensing n° 53, 1998, Elsevier Editeur.

[Viollet Le Duc, 1863/1977] Eugène Viollet Le Duc
*"Entretiens sur l'architecture"*
Edition Pierre Mardaga 1977, réédition du texte écrit entre 1863 et 1872 par Eugène Viollet Le Duc.

[Viollet Le Duc, 1854/1996] Eugène Viollet Le Duc
*"Encyclopédie Médiévale"*
Edition Bibliothèque de l'image, 1996., ouvrage basé sur le *dictionnaire raisonné de l'architecture*, rédigé entre 1854 et 1868

[Viollet Le Duc, 1877/1978] Eugène Viollet Le Duc
"Histoire d'un hôtel de ville et d'une cathédrale"
Edition Pierre Mardaga 1978, réédition de l'ouvrage original édité en 1877.

[Vitruve, 1684/1988] Vitruve,
"Les dix livres d'Architecture"
Traduction de Claude Perrault 1684, édition Pierre Mardaga 1988.

[Whiting et al, 1997] David Whiting, Steve Nickerson
"Computer aided recording tools automate the creation of a site information system"
in Proceedings CIPA Symposium 1997, Göeteborg, Suède, 1-3 octobre 1997.

[Zin, 1966] Wiktor Zin
"Kronika prac i problemów konserwatorskich krakowa za rok 1965"
Rocznik krakowski, tome XXXVIII, 1966, Kraków.

### 7.2.2. Représentation des connaissances et approche objet

[Bellahsène et al, 1998] Zohra Bellahsène, Roland Ducournau
"Vues en bases de données et points de vue en représentation des connaissances"
Revue L'objet Volume 4 / n°3, pages 307 à 331, publié par Hermès éditions, 1998, ISSN1262-1137

[Benett, 1997] Gordon Benett
"Oyster Grit"
Intranet Design Magazine, 16 Janvier 1997 (http://idm.internet.com/features/perlism.shtml)
et dans LAN Times Online, McGraw-Hill Publishers, http://www.lantimes.com/

[Blair et al, 1991] Blair,Gallagher, Hutchison and Shepherd
"Object-Oriented Languages, Systems and Applications"
Published by Pitman Press ISBN 0-273-03132-5, 1991.

[Borne et al, 1999] Isabelle Borne, Nicolas Revault
"Comparaison d'outils de mise en œuvre de design patterns"
Revue L'objet Volume 5 / n°2, pages 243 à 266, publié par Hermès éditions, 1999, ISSN1262-1137

[Branki et al, 1997] C.Branki, A.Wallis, I.Aird, A.Bridges
"Collaborative design objects as experts agents"
in CAAD towards new design conventions, TU Białystok, 1997, pp 63-74, ISBN 83-86272-63-5

[Brenner et al, 1996] Steven E.Brenner, Edwin Aoki
"Introduction to CGI/Perl"
Publié par M&T books, 1996, ISBN 1-55851-478-3

[Brown et al, 1996] Brown, A., Rezgui, Y., Cooper, G., Yip, J. , Brandon, P.
Promoting Computer Integrated Construction Through the Use of Distribution Technology
ITCON Electronic Journal of Information Technology in Construction, Volume 1, 1996, ISSN 1400-6529.

[Budd, 1996] Timothy Budd
"Introduction to Object-Oriented Programming"
Edité par Addison-Wesley Longman, 1996 (seconde édition), ISBN 0201824191

[Castellani, 1998] Xavier Castellani
"Catégories d'objets d'un système d'objets"
Revue L'objet Volume 4 / n°1, pages 45 à 72, publié par Hermès éditions, 1998, ISSN1262-1137

[Chen et al, 1995] Jian Chen, Qiming Huang
"Eliminating the impedance mismatch between relational systems and Object-Oriented programming languages"
In *Database reengineering and interoperability*, proceedings 6th international Hong Kong computer society database workshop, Honk Kong, 1995.

[Christiansen et al, 1999] Tom Christiansen, Nathan Torkington
"Perl en action"
Edité par O'Reilly, 1999, traduit par Sébastien Blondeel, ISBN 2-84177-077-X

[Clark et al, 1998] Tony Clark, Andy Evans
"Foundations of the Unified Modeling Language"
in Proceedings 2nd northern formal methods workshop, Ilkley, Edit. Electronic Workshops in Computing, Springer-Verlag, 1998.

[Conway, 2000] Damian Conway
"Object Oriented Perl"
Edité par Manning Publications , 2000, ISBN 1-884777-79-1

[Conway, 1997] Damian Conway
"Advanced Object Oriented Techniques In C++ "
Course Material Monash University , 1997,
Adressse : http://www.csse.monash.edu.au/~damian/Idioms/Topics/10.1.Persistence/html/text.html

[DeChampeaux et al, 1993] Dennis de Champeaux, Doug Lea, Penelope Faure
"Object oriented system development"
Edité par Addison-Wesley, 1993, ISBN 0-201-56355-X

[Dodani, 1999] Mahesh Dodani
"Rules are for fools, Patterns are for coll fools"

JOOP (Journal of Object Oriented Programming) Octobre 1999, http://www.joopmag.com (SIGS Publications)

[Donath et al, 1997a] Dirk Donath, Frank Petzold
"Towards a building information system based on computer-supported surveying system"
in CAAD towards new design conventions, TU Białystok, 1997, ISBN 83-86272-63-5

[Donath et al ,1997b] Dirk Donath, Frank Petzold
"From Digital Building Surveying to an Information System"
ECAADE 97 in Vienna, September 1997

[Ducournau et al, 1998] Roland Ducournau, Jérôme Euzenat, Gérald Masini, Amedeo Napoli
"Langages et modèles à objets. Etat des recherches et perspectives"
Publié par l'INRIA (Institut National de Recherche en Informatique et en Automatique), 1998, ISBN 2-7261-1131-9

[Ducournau, 1997] Roland Ducournau
"La compilation de l'envoi de message dans les langages dynamiques"
Revue L'objet Volume 3 / n°3, publié par Hermès éditions, septembre 1997, ISSN1262-1137

[Durnota, 1995] Bohdan Durnota
"Complex object models of natural systems"
Proceedings International conference OOSimulation '95, Las Vegas, USA, 1995.

[Eastman, 1994**]** Charles M. Eastman
"A data model for design knowledge."
Automation in construction, Elsevier juillet 1994

[Ekholm, 1996] Ekholm, A.
"A Conceptual Framework for Classification of Construction Works"
ITCON Electronic Journal of Information Technology in Construction, Volume 1, 1996, ISSN 1400-6529.

[Englander, 1997] Robert Englander
"Java Beans - guide du programmeur"
Editions O'Reilly, édition française traduite par Bonnie Cupac, 1997, ISBN 1-84177-038-9

[Euzenat et al, 1995] Jérôme Euzenat, François Rechenmann
"Shika, 10 ans c'est Tropes ?"
Actes des 2ème journées Langages et Modèles à objets, Nancy, 1-3 octobre 1995, pp13-34

[Evans et al, 1998] Andy Evans, Robert France, Kevin Lano, Bernhardt Rumpe
"The UML as a Formal Modeling Notation"
in Proceedings UML' 98 1st Int workshop on UML, Mulhouse, France, 3-4 juin 1998, édité par Springer-Verlag LNCS 1618

[Flanagan, 1997] David Flanagan
"JAVA in a nutshell"
O'Reilly Publishing, deuxième édition, 1997, ISBN 1-56592-262-X

[Froese, 1992] Thomas Michael Froese
"Integrated computer-aided project management through standard object-oriented models"
Thèse de doctorat du département génie civil de l'université de Stanford
Soutenue en juin 1992, Université de Stanford, USA.

[Gaillard, 1994] Fabrice Gaillard
*Thèse de doctorat sur la modélisation des connaissances et l'utilisation de base de données objet en productique*
Soutenue le 1er Décembre 1994 à l'UT Compiègne, France.

[Hanrot, 1989] S.Hanrot
*"Une base de connaissances architecturales pour un système de CAO intelligent"*
Thèse de troisième cycle de l'Université d'Aix-Marseille III, mars 1989.

[Harold, 1997] Eliotte Rusty Harold
"Programmation réseau avec JAVA"
Edité par O'Reilly Publishing, édition française, traduction de Manuel Makarévitch, 1997, ISBN 2-84177-024-6

[Hitz et al, 1998] Martin Hitz, Gerti Kappel
"Developing with UML – Some pitfalls and workarounds"
in Proceedings UML' 98 1st Int. workshop on UML, Mulhouse, France, 3-4 juin 1998, édité par Springer-Verlag LNCS 1618

[Kappel et al, 1998] Gerti Kappel, Birgit Schroder

"Distributed light-weight persistence in Java- A tour on RMI- and CORBA- based solutions"
in Proceedings 9th International Conference on Database and Expert Systems Applications (DEXA 98)
Vienna, Autriche, 24-28 Août 1998, pp 411-424, ISSN 0302-9743

[Lea, 1997] Doug Lea
"Christopher Alexander: an introduction for Object-Oriented Designers"
Critique en ligne : http://g.oswego.edu/dl/ca/ca/ca.html
Publié dans "*Patterns Handbook*" par L.Rising, editions Cambridge University Press, 1998, ISBN 0-521-64818-1

[Line, 1997] Lars Line
"Virtual Engineering Teams: Strategy and Implementation"
ITCON Electronic Journal of Information Technology in Construction, Volume 2, 1997, ISSN 1400-6529.

[Masini et al, 1989] Gérald Masini, Amedeo Napoli, Dominique Colnet, Daniel Leonard, Karl Tombre
"Les langages à objets. Langages de classes, langages de frames, langages d'acteurs"
Publié par InterEditions, 1989 (deuxième tirage corrigé 1990) avec le concours du Centre National des Lettres

[Maughan et al, 1998] Glenn Maughan, Jon Avotins
"A Meta-model for Object-oriented Reengineering and Metrics Collection"
Eiffel Liberty Journal, Vol 1, n°4, 1998

[Menzies et al, 1995] Tim Menzies, Philip Haynes
"Do we really need encapsulation ?"
in Proceedings TOOLS (Technology Of Object-Oriented Languages And Systems) 1995 conference
Melbourne, Australia, 27 November - 30 November 1995

[Nierstrasz, 1989] Oscar Nierstrasz
"A survey of object oriented concepts"
Publié dans "Object oriented concepts, databases and applications", éditions ACM Press Addison Wesley , 1989.

[Ousssalah, 1997] Chabane Oussalah (et alii)
"Ingéniérie objet"
Publié par InterEditions, 1997, ISBN 2-7296-0642-4

[Plaindoux et al, 1998] Didier Plaindoux, Jean-Paul Bodeveix, Christian Percebois
"Types versus Classes"
Revue L'objet Volume 4 / n°1, publié par Hermès éditions, mars 1998, ISSN1262-1137

[Randall et al, 1993] Randall Davis, Howard Shrobe,Peter Szolovits.
"What is a Knowledge Representation?"
AI Magazine, 14 (1):17-33, 1993.

[Rezgui et al, 1998]Yacine Rezgui, Grahame Cooper
"A proposed open infrastructure for construction project document sharing"
ITCON Electronic Journal of Information Technology in Construction, Volume 3, 1998, ISSN 1400-6529.

[Rising, 1996] Linda Rising
"The road, Christopher Alexander, and good software design"
Proceedings OOPSLA '96, (Eleventh Annual ACM Conference on Object-Oriented Programming Systems, Languages and Applications) 6-10 October 1996, San Jose, USA.

[Rosenblatt, 1995] Bill Rosenblatt
"Persistently C++"
Edité par Web Publishing Inc., SunWorld / Advanced Systems Magazine, Avril 1995
À l'adresse : http://sunsite.uakom.sk/sunworldonline/asm-04-1995/asm-04-client.html

[Schneider, 1994] Daniel K. Schneider
Thèse de doctorat sur la modélisation de la démarche du décideur politique dans la perspective de l'intelligence artificielle présentée à la faculté des sciences économiques et sociales de Genève, 1994.

[Seinturier et al ,1998] Lionel Seinturier, Laurence Duchien, Gérald Florin
"CAOLAC : un protocole à méta-objets pour la synchronisation d'objets concurrents"
Revue L'objet Volume 4 / n°3, publié par Hermès éditions, septembre 1998, ISSN1262-1137

[Sintès et al, 1989] Claude Sintès, Jean-Maurice Rouquette
"Arles antique, Monuments et sites"
Edition Imprimerie Nationale, 1989.

[Snyder, 1998] Alan Snyder

"Object-Oriented language design"
in Eiffel Liberty Journal - Vol.1, No.1, 1998 (http://www.elj.com/elj/).

[Sunyé et al, 1997] Gerson Suyé, Houari A.Sahraoui, B.Lesueur, G.Blain
"Chroniques d'implémentation d'un méta-outil en Smalltalk"
Revue L'objet Volume 3 / n°4, publié par Hermès éditions, décembre 1997, ISSN1262-1137

[Stroustrup, 1996] Bjarne Stroustrup
"Le langage C++"
Thomson Publishing, deuxième édition, 1996, traduit par Hervé Soulard, ISBN 2-84180-079-2

[Tsichritzis et al, 1992] Dennis Tsichritzis, Oscar Nierstrasz, Simon Gibbs
"Beyond Objects: Objects"
in International Journal of Intelligent and Cooperative Information Systems (IJICIS), vol 1, n°1, pp 43-60, 1992.

[Turk et al, 1993] Turk, @, D.J. Vanier.
"Classification systems in object oriented modelling of buildings"
Proceedings DMMI Conference, Maribor, ISBN 86-345-0100-0, Slovenia, 1993.

[Watanabe, 1994] S. Watanabe
"Knowledge integration for architectural design"
Automation in construction, Elsevier juillet 1994

[Wu et al, 1995] Xindong Wu, Sita Ramakrishnan, Heinz Schmidt
"Knowledge objects"
Revue Informatica (International journal of computing and informatics) vol 19, n°4, 1995.

[Wu, 1996] Xindong Wu
"A comparison of Objects with Frames and OODBs"
in Object Currents Journal, Volume 1, number 1, 1996. SIGS Publications http://www.sigs.com/objectcurrents

### 7.2.3. Gestion de données et interfaces Web

[Atkinson et al, 1990] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, et S. Zdonik
"The object-oriented database system manifesto"
In Proceedings of the 1 st Int. Conf. on Deductive and Object-Oriented Databases, Kyoto, Japan. Elsevier Science Publishers B.V (NorthHolland), 1990, pp40-57.

[Bilgin, 1997] A.G Bilgin
"An adaptable Model for a systematic approach to conservation works : An introductory study on GIS for the urban archaeological site of Bergama (Pergamon)"
Informatique et conservation-restauration du patrimoine culturel, Actes des huitièmes journées de la SFIIC, 23-24 octobre 1997, pp211-220, ISBN 2-905430-10-9.

[Dalbera, 1997] J.P Dalbera
"Des bases de données aux applications multimédia"
Actes des 8èmes journées d'études de la SFIIC, Châlons sur saône, 23 -24 Octobre 1997, ISBN 2-905430-10-9.

[Ducournau et al, 1998] Roland Ducournau, Jérôme Euzenat, Gérald Masini, Amedeo Napoli
"Langages et modèles à objets. Etat des recherches et perspectives"
Publié par l'INRIA (Institut National de Recherche en Informatique et en Automatique), 1998, ISBN 2-7261-1131-9

[Israel, 1997] Marc Israel
"Administration SQLServer 6.5. Sécurité, Optimisation, réplication, interface Web"
Editions Eyrolles, 1997

[Lammari et al, 1999] Nadira Lammari, Régine Laleau, Mireille Jouve
"Processus d'optimisation conceptuelle d'un schéma orienté objet"
Revue L'objet Volume 5 / n°1, "best of OOIS", publié par Hermès éditions, 1999, ISSN1262-1137, ISBN 2-86601-750-1

[Loomis, 1995] Mary E.S. Loomis
"Object databases"
Addison Wesley publishing company, 1995, ISBN 0-201-56341-X

[Rahayu et al, 1999] W.Rahayu, E.Chang
"Composites indices as a mechanism for transforming multi-level composite objects into relational databases"
Revue L'objet Volume 5 / n°1, "best of OOIS", publié par Hermès éditions, 1999, ISSN1262-1137, ISBN 2-86601-750-1

### 7.2.4. Informatique graphique

[Bridges ,1999] A.Bridges
"Progress? What Progress?"
in Architectural computing, proceedings Ecaade 1999, pp 321-326, ISBN 0 9523687 5 7

[Bourdakis et al, 1999] V.Bourbakis, D.Charitos
"Virtual Environment Design - defining a new direction for architectural education"
in Architectural computing, proceedings Ecaade 1999, pp 403-409, ISBN 0 9523687 5 7

[Drap et al, 2000] P.Drap , P. Grussenmeyer
"A digital photogrammetric workstation on the WEB"
In Journal of Photogrammetry and Remote Sensing 55 (2000), pp.48-58.Avril 2000, Elsevier.

[Geroimenko, 1999] V.Geroimenko
"Online Photorealistic VR with Interactive Architectural Objects"
in Architectural computing, proceedings Ecaade 1999, pp 414-417, ISBN 0 9523687 5 7

[Juhasz et al ,1999] P.Juhasz, Z.Kiss, M.Szoboszlai
"Drawing's dimensions"
in Architectural computing, proceedings Ecaade 1999, pp 498-502, ISBN 0 9523687 5 7

[Péroche et al ,1988] Bernard Péroche, Jacqueline Argence, Djamchid Ghazanfarpour, Dominique Michelucci
"La synthèse d'images"
Hermès Editions , 295 pages , 1988 - ISBN : 2-86601-112-0.

[Péroche et al ,1997] Bernard Péroche, Jacqueline Argence, Djamchid Ghazanfarpour, Dominique Michelucci
"Informatique graphique 2e édition - Méthodes et modèles"
Hermès Editions, 416 pages , 1997 - ISBN : 2-86601-652-1.

[Oxman ,1999] R.Oxman
"Visual Emergence in Collaborative Design"
in Architectural computing, proceedings Ecaade 1999, pp 357-363, ISBN 0 9523687 5 7

[Potier et al, 1998] S.Potier, JL Maltret J.Zoller
"L'infographie : une aide a la formulation d'hypotheses archeologiques."
in Digital Design Studios: Do Computers Make A Difference? Proceedings ACADIA 98, T. Seebohm and S. V. Wyk, eds., ville de Québec, Québec, Canada.

[Roberts, 1999] A.Roberts
"Virtual site planning"
in Architectural computing, proceedings Ecaade 1999, pp 442-448, ISBN 0 9523687 5 7

[Roehl et al, 1997] B.Roehl, J.Couch, C.Reed-Ballreich, T.Rohaly, G.Brown
"Late night VRML 2.0 with Java"
Ziff -Davis Press / MacMillan Computer Publishing Editeurs, 710 pages, 1997, ISBN 1-56276-504-3

[Rolle, 1998] Jean-François Rolle
"Mon ordinateur voit double"
Revue "flash Informatique" du Service informatique central de l'EPFL (Ecole polytechnique Fédérale de Lausanne, Suisse), numéro spécial du 1er septembre 1998 (FI -SP 1998).

[Tweed, 1997] C.Tweed
"The predominance of the visual in Computer Aided Architectural Design"
in CAAD towards new design conventions, TU Białystok, 1997, pp 269-285, ISBN 83-86272-63-5

### 7.2.5. Interfaces pour le réseau Internet

[Albitz et al, 1998] P.Albitz, C.Liu
"DNS and Bind"
Edition sO'Reilly, 3ème éditions, 482 pages, 1998, ISBN 1-56592-512-2

[Brenner et al, 1996] Steven E.Brenner, Edwin Aoki
"Introduction to CGI/Perl"
Publié par M&T books, 1996, ISBN 1-55851-478-3

[Christiansen et al, 1999] Tom Christiansen, Nathan Torkington
"Perl en action"
Edité par O'Reilly, 1999, traduit par Sébastien Blondeel, ISBN 2-84177-077-X

[Cohen, 1996] I.Cohen
"CGI/Perl et Javascript"
Editions Eyrolles, 1996, 294 pages, ISBN 2-212-08918-X

[Conway, 2000] Damian Conway
"Object Oriented Perl"
Edité par Manning Publications , 2000, ISBN 1-884777-79-1

[Dedieu, 1997] Olivier Dedieu
"Java et l'informatique répartie"
Note OD97A groupe SOR (Systèmes dObjts Répartis) de l'INRIA.

[Garshol, 1999] Lars Marius Garshol
"Introduction to XML"
Note de travail Department of Informatics University of Oslo (Norvège).

[Harold, 1997] Eliotte Rusty Harold
"Programmation réseau avec JAVA"
Edité par O'Reilly Publishing, édition française, traduction de Manuel Makarévitch, 1997, ISBN 2-84177-024-6

[Huitema, 1994] C.Huitema
"Le routage dans l'internet"
Editions Eyrolles, 407 pages, 1994, ISBN 2-212-08902-3

[Laurie et al, 1997] B.Laurie, P.Laurie
"Apache, Installation et mise en œuvre"
Traduit par R.Debonne, éditions O'reilly, 294 pages, 1997, ISBN 2-84177-036-2

[Lecomte et al, 1996] C.Lecomte, T.Leduc
"Programmation Javascript"
Editions Eyrolles, 1996, 318 pages, ISBN 2-212-08937-6

[Orfali et al, 1996] R.Orfali, D.Harkey, J.Edwards
"Objets répartis"
Traduit par F.Leroy, J.P Gout, R.Larsen
International Thomson Publishing France, 618 pages, 1996, ISBN 284180-043-1

[Oxman, 1999] Rivka Oxman
"Visual Emergence in Creative Collaboration"
in Architectural computing, proceedings Ecaade 1999, pp 357-363, ISBN 0 9523687 5 7

[Tanenbaum, 1997] A.Tanenbaum
"Réseaux"
Traduit par J.A Hernandez et R.Joly
Prentice Hall / Dunod éditeurs, 3ème édition, 791 pages, 1997, ISBN 2-10-004315-3

### 7.2.6. Publications du travail présenté

[Acary et al, 1999] V. Acary, J.Y. Blaise, P. Drap, M. Florenzano, S. Garrec, M. Jean, D. Merad.
*"NSCD method applied to mechanical simulation of masonry in historical buildings using MOMA"*
XVII CIPA International SymposiumWG3 - Simple methods for architectural photogrammetry Olinda, Brazil, October 3-6 1999.

[Blaise , 2000] Jean-Yves Blaise
*"Outils numériques et représentation de l'architecture patrimoniale"*
Culture et recherche n°81, publication de la MRT (Mission de la Recherche et de la Technologie du Ministère de la Culture et de la Communication), novembre-décembre 2000

[Czubinski et al, 1998a] J.Czubinski, P. Drap, I.Dudek, J.Y Blaise
*"Collaborative network tools for the architectural analysis in conservation research"*
CYBER-REAL DESIGN, 1998 23-25 April 1998 Bialystok, POLAND.

[Drap et al, 1995] Pierre DRAP, Jean-Yves BLAISE
*"PAROS. De la photogrammétrique à la synthèse d'image. Le cas du forum antique de la ville d'Arles"*
Actes des Troisièmes journées de l'AFIG Association Française d'Informatique Graphique 1995, Marseille, France, 22 - 24 Novembre 1995, pages 303 à 310.

[Drap et al, 1998b] P. Drap, I.Dudek, J.Y Blaise
*"Java collaborative interface for architectural simulations A case study on wooden ceilings of Kraków"*
Międzynarodowe Sympozjum Konserwatorskie,V Teoria i praktyka w ochronie zabytków architektury, zespołów i miejsc historycznych , 22-24 November 1998, Kraków, POLAND.

[Drap et al, 1999a] P. Drap, J.Y Blaise, P. Grussenmeyer
*"A photogrammetric survey using knowledge representation on the arpenteur web-based photogrammetric workstation"*
XVII CIPA international symposium wg3 - simple methods for architectural photogrammetry Olinda, brazil, october 3-6, 1999

[Drap et al, 1999b] P. Drap, I.Dudek, J.Y Blaise
*"An architectural model compiler dedicated to archaeological hypothesis. An experiment on Krakow's kramy Bogate"*
HCP'99, Human Centered Processes 22 - 24 September 1999 Brest, France.

[Dudek et al, 1999a] I.Dudek, J.Y Blaise
*"IT applications for architectural intervention and documentation in monuments' ensembles"*
Międzynarodowe Sympozjum Konserwatorskie,V Teoria i praktyka w ochronie zabytków architektury, zespołów i miejsc historycznych ,*22-24 November 1999, Kraków, POLAND.*

[Dudek et al, 1999b] I.Dudek, J.Y Blaise
*"SOL : Spatial and historical web-based interface for On Line architectural documentation of Kraków's Rynek Glówny "*
Turning to 2000,17th conference of eCAADe, September 15-18, 1999 The University of Liverpool, UK

[Dudek et al, 2001] I. Dudek, J.Y Blaise
*"Interpretative modelling as a tool in the investigation of the architectural heritage : information and visualisation issues" Proceedings. VIIP (Visualization, Image and Image Processing) 2001 international Conference,* Marbella, Spain, 2001, pp48-54, publié par ACTA Press, ISBN 0-88986-309-1.

[Florenzano et al, 1996a] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"PAROS, Photogrammétrie Architecturale et Restitution par Outils de Synthèse"*
Revue de CFAO et d'informatique graphique. Volume 11 - N°4/1996,pages 345 à 361.

[Florenzano et al, 1996b] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"Photogrammétrie et modèles architecturaux"*
ISPRS: XVIIIème Congrés de la société Internationale de Photogrammétrie et de télédétection, Vienne, Autriche, juillet 1996.
Proceedings: International archives of photogrammetry and remote sensing, volume XXXI, tome B5, pages 167 a 172.

[Florenzano et al, 1997a] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"PAROS, Modèles objet appliques à l'architecture construite"*
L'OBJET, Logiciel, Bases de Données, Réseaux. Numéro 1, volume 3, pages 29 à 52. Revue publiée par éditions Hermès, avril 1997.

[Florenzano et al, 1997b] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"Photogrammetry and knowledge representation, restitution of archeological hypothesis on the Bigot model of ancient Rome"*
CIPA, Photogrammetry in Architecture, Archeology and Urban Conservation, Göteborg, Suède, les 1,2 et 3 octobre 1997.
Proceedings: Intenational Archives of Photogrammetry and remote sensing, volume XXXII, part 5C1B, pages 90 a 96.

[Florenzano et al, 1998] Michel FLORENZANO, Jean-Yves BLAISE, Pierre DRAP
*"From photogrammetrical survey to architectural models monitoring through a virtual reality interface"*
Complex Systems Intelligent Systems & Interfaces du 26 au 28 mai 1998 Nimes, France.
Publié dans La lettre de l'IA, numéro 134-135-136 mai-août 1998 pages 168, 175.

[Florenzano et al, 2000] Jean-Yves Blaise, Michel Florenzano
"Modèles et représentation a l'échelle architecturale : une expérience à Cracovie "
Colloque International "Rome an 2000", Université" de Caen – Basse Normandie, (28-29-30.09.2000) Actes à paraître.

# 8.ANNEXES

## *LISTE DES SUJETS ABORDES EN ANNEXE*

*ANNEXE 1 : DESCRIPTIONS DE L'ÉDIFICE DANS LE CHAMP DE LA CONSERVATION*

Nous nous plaçons ici dans le contexte de l'analyse architecturale par classifications et discutons d'abord des différences qui séparent les deux principales familles de classifications du patrimoine bâti que nous avons identifié :
- Classifications stylistiques.
- Classifications typologiques.

Nous présenterons ensuite les travaux de référence sur le vocabulaire de l'architecte sur lesquels nous nous appuyons dans notre identification du corpus :
- Vocabulaire attaché aux objets.
- Vocabulaire attaché au positionnement des objets.
- Vocabulaire attaché aux fonctions des objets.

Notre position quant à l'utilisation de ce vocabulaire sera explicitée dans la partie de ce document consacrée à l'état d'avancement du projet. Nous observerons les différences qui nous semblent apparaître dans les champs de l'architecture patrimoniale et de la conception architecturale sur les modes de description de l'édifice. Nous verrons notamment ce qu'une intervention dans le champ de la conservation recouvre en terme d'exigences pour le modèle architectural (l'original, le réemploi, etc.).
Nous introduirons par le travail de Philippe Boudon sur l'échelle en architecture [Boudon, 1977] le problème que pose la gestion d'informations hétérogènes attachées à l'édifice en fonction de l'objectif de l'étude (analyse historique, stylistique, projet de restructuration, etc.). Nous mentionnerons ainsi quelques-unes des grandes familles d'études patrimoniales et la diversité des informations qu'elles imposent de prendre en compte.

*ANALYSE ARCHITECTURALE ET CLASSIFICATIONS*
Nous allons ici dans un premier temps introduire l'idée que deux grandes familles de classification président à l'analyse du patrimoine bâti :
- Classifications stylistiques.
- Classifications typologiques.

Nous tenterons de donner une définition plus précise de ces deux familles en nous appuyant sur les méthodes d'analyse données en bibliographie. Nous verrons ensuite par quels choix de modèle se traduit l'appartenance à chacune de ces familles de classification puis en quoi la seconde option sert de base à notre travail.

**Classifications: définitions**
Revenons d'abord sur l'opposition que nous établissons entre ces deux familles. Il s'agit en fait d'une opposition avant tout commode, qu'il est nécessaire de décrire de façon plus approfondie. On observe essentiellement deux modes d'analyse de l'architecture : celui qui, partant d'un tout (l'édifice), en détaille les parties homogènes, notamment en terme de cohérence stylistique ; et celui qui s'attache à la description typo-morphologique d'éléments canoniques d'architecture, et propose in fine un tout (l'édifice) résultant d'un agencement de ces éléments.
A l'origine de ces deux modes d'analyse il n'y a pas opposition de principe mais différence d'objectif : dans le premier cas, l'étude porte sur l'édifice en relation en particulier avec son contexte historique ou artistique. Dans le deuxième cas, l'étude reste centrée sur l'architecture en tant que discipline. Ces deux approches sont donc complémentaires, elles ne sont pas issues de spécialités différentes mais correspondent à des objectifs différents. Ces deux approches sont déjà présentes dans le traité d'architecture de Vitruve [Vitruve, 1684/1988], présentées de façon indépendante l'une de l'autre. En effet, le livre IV décrit un corpus d'objets théoriques sans références à un édifice particulier. Le livre VII, lui, décrit un ensemble de projets, agencements théoriques s'appuyant sur le corpus du livre IV, et les présente dans une approche du tout aux parties.

Jusqu'à la Renaissance, c'est avant tout la pratique, le savoir-faire, et le rapport des parties au tout qui régissent l'art de bâtir.
Ceci est dû en partie à la succession de deux influences :

- D'abord celle des ordres religieux qui essaiment leur école artistique et le savoir-faire de leurs artisans partout en Europe,
- puis en réaction, l'influence combinée du pouvoir ecclésiastique local et de l'organisation sociale des villes où se développent des confréries d'artisans, qui donne naissance à une architecture urbaine, indépendante de forme et d'esprit.

On peut pour une analyse plus détaillée de ces transformations se reporter au septième entretien sur l'Architecture de Viollet Le Duc [Viollet Le Duc, 1863/1977, pp 249-319].
Le savoir-faire technique s'accompagne de règles de proportionnalités des éléments de corpus qui guident le bâtisseur dans la construction d'un tout cohérent. On pourra se reporter au livre de Z.Dmochowski [Dmochowski, 1956] pour voir par l'exemple comment ces règles se traduisent dans le contexte de nos principaux terrains d'expérimentation, et comment elles évoluent localement (proportion 5/8 des piliers des églises gothiques de Cracovie, doublement du pilier par un contrefort remplaçant l'arc boutant, etc..). De nombreux ouvrages traitent de ce point dans le cadre de la production architecturale française, comme par exemple [Choisy, 1899/1991] [Cuisenier, 1991] [Fichet, 1979] [Larpin, 1988] [Viollet Le Duc, 1863/1977], ouvrages que nous aurons l'occasion pour la plupart de citer plus en détail ultérieurement.

La démarche que suivent les premiers architectes-auteurs de la Renaissance est très voisine de celle de Vitruve. Si une étude typo-morphologique des éléments d'architecture est proposée, elle l'est de façon indépendante des bâtiments concrets que ces architectes-auteurs décrivent. Andrea Palladio détaille dans son premier livre [Palladio, 1738/1965, livre 1] un ensemble de recommandations sur la manière de bâtir, dont la typo-morphologie et les proportions des éléments théoriques de corpus. En propos introductif à son premier livre, Palladio cite Vitruve et fait la recommandation suivante[163] : *"[…] great care ought to be taken, before a building is begun, of the several parts of the plan and elevation of the whole edifice intended to be raised…"*

Son premier livre établit des recommandations qui détaillent à la fois la morphologie, les dimensions relatives et la structure de l'élément étudié. La méthode de tracé des éléments moulurés est également décrite. Ce premier livre établit donc bien une relation des parties au tout, dans laquelle chaque élément typo-morphologique est théorisé. Dans son livre deux [Palladio, 1738/1965, livre 2], Palladio peut alors présenter plusieurs édifices dont il est l'architecte sans détailler les éléments du corpus utilisé mais en insistant sur leur agencement global, leurs proportions[164], et passant cette fois du tout aux parties.

Avec les encyclopédistes, ces deux approches vont très clairement se scinder. En effet, ceux-ci définissent de façon indépendante *l'élément* d'architecture atomique (une base, un profil) et un

[163] Il considère donc qu'avant de construire un tout, chacune de ses parties doit être maîtrisée. Son premier livre établit des recommandations sur, dans l'ordre :
- les matériaux de construction,
- la manière de fonder un édifice,
- la structure interne et la construction des murs,
- les cinq ordres (éléments de corpus, proportions et composition),
- les piédestaux,
- la forme des pièces de distribution,
- les planchers et plafonds,
- les couvrements,
- les portes et fenêtres,
- les cheminées,
- les escaliers,
- les toits.

[164] Il est intéressant de noter que l'idée de proportion héritée du *Proportio* [Vitruve, 1988] de Vitruve est aujourd'hui interprétée comme dimensions relatives des éléments composant l'édifice. Vitruve parlait en fait de *Proportio* comme plus généralement du rapport des parties au tout, et ne limitait donc pas ce principe d'ordonnancement aux seules dimensions relatives des éléments du corpus. Cette idée de *Proportio* qu'il présente comme un élément déterminant de la qualité de l'édifice peut être comprise comme la nécessité d'établir un ensemble de rapports (dimensionnels, stylistiques, contextuels, etc…) entre chaque élément du corpus utilisé et l'édifice dans sa globalité.

*agencement* qu'ils désignent comme élément non-divisible de leur classification. L'école académique française fournira essentiellement des classifications stylistiques ou maniéristes dont l'imprécision tranche avec la volonté de théoriser l'ordre dont elle est issue [Fichet, 1979, p21]. Jacques-François Blondel propose ainsi dans son cours d'architecture paru en 1771 une définition de l'architecture « *champêtre, mystérieuse, hardie, terrible, grande, naine, frivole, licencieuse, dissemblable, amphibologique, vague, barbare* », etc…[ Fichet, 1979, p439]. Françoise Fichet parle d'une méthode cartésienne [Fichet, 1979, p22] qui donne naissance au XVIIéme siècle à la doctrine classique en France. Elle la définit comme s'appuyant sur :

- L'ordre : disposer les objets de la pensée de façon à ce qu'ils puissent être représentés.
- La mesure : les ramener à une unité extérieure au contexte, unité de référence pour l'esprit.

Il est frappant de noter que si les livres d'architecture de Vitruve peuvent relever de ce modèle cartésien, on ne peut pas en dire autant des écrits de Jacques-François Blondel, représentant de cette école académique Française qui pourtant s'en réclame.
Citons encore André Félibien [Fichet, 1979, p125] :
*L'architecte… cherche d'abord à concevoir une noble idée de son dessein, et lorsqu'il la possède il établit une mesure qui lui sert de loi et de raison, par laquelle il ordonne (…) les choses qui entrent dans la composition de ce qu'il veut bâtir.*
L'idée de mesure est présente, elle se place comme le moyen de définir harmonieusement les parties à partir du tout : l'approche est celle d'une classification stylistique. Poursuivons: « *Quand il a une fois déterminé ses mesures, et choisi les ordres qu'il veut suivre, il travaille à la proportion des parties et aux ornements qu'elles sont capables de recevoir »…*
Félibien introduit trois idées sur lesquelles il nous faut revenir :

- L'architecte *choisit les ordres qu'il veut suivre*, autrement dit utilise une classification stylistique de la modénature de ces ordres.
- L'architecte *travaille à la proportion des parties*, autrement dit ramène les parties au tout qu'il a déjà défini.
- L'architecte *travaille aux ornements qu'elles sont capables de recevoir*, autrement dit plaque sur celles-ci la quantité nécessaire et suffisante d'ornements en vertu de son bon goût.

Il semble donc qu'au moment où en botanique par exemple l'idée de classification renaît avec vigueur, seuls les encyclopédistes proposent une démarche équivalente en architecture. Bien que se réclamant d'une pensée cartésienne, la théorie architecturale à l'âge classique néglige l'observation des éléments atomiques du corpus architectural au profit d'une classification par similitudes de modénatures et de règles de proportions décontextualisées ; classification dont le dix-neuvième siècle amplifiera l'usage. On verra dans le chapitre consacré au vocabulaire que la théorie architecturale à l'âge classique est pourtant riche en efforts de recensement des termes liés aux objets physiques, mais sans pour autant en dresser de classifications systématiques.
C'est notamment contre cet académisme qu'il nomme une "barbarie" que s'élève Eugène Viollet Le Duc. Son œuvre marque l'évolution de la pratique et de la théorie architecturale en France à la fois par l'influence qu'elle a eue au moment ou naissait la conscience d'un patrimoine architectural à préserver, et par la vision claire et convaincue qu'il donne des principes régissant ce qu'il est convenu d'appeler les styles. Viollet Le Duc, comme Camillo Sitte à l'échelle urbaine, défend une architecture "vraie" contre ce qu'il observe chez ses contemporains : "des principes anciens singés sans conscience".
Il n'entre pas dans le sujet de ce travail de s'attarder sur les positions de Viollet Le Duc ou de montrer en quoi son discours comme la portée de son œuvre dépassent du cadre du pittoresque qu'on leur accorde quelquefois. On pourra se reporter à [Viollet Le Duc, 1863/1977] ou à [Viollet Le Duc, 1854/1996]. Il est cependant important de remarquer que les démonstrations que proposent Viollet Le Duc lorsqu'il analyse l'architecture laïque française des XII et XIIémé siècles comme similaire dans l'esprit, mais opposée dans la forme, à l'architecture de l'antiquité grecque s'appuient sur une description fine du corpus architectural.
Viollet Le Duc décrit les éléments de corpus utilisés et leur arrangement dans l'espace de l'édifice pour expliquer comment naît le sentiment d'un tout cohérent. En cela, Viollet Le Duc reconnaît l'édifice comme résultant des parties qui le composent et le style comme dépendant de la cohérence de leur arrangement. La classification qu'il propose est stylistique parce qu'elle identifie une période historique.

Mais elle est avant tout typologique car elle repose sur une analyse des parties au tout, elle place le corpus architectural au centre des critères d'évaluation de l'édifice.
L'opposition que j'identifie entre classifications stylistiques et classifications typologiques n'a donc pas de rapport avec la présence ou non de l'idée de style dans la qualification de l'édifice. Elle se justifie par l'opposition qui se fait jour entre ceux qui attachent l'idée de style à une apparence, un tout, et ceux qui l'attachent à la morphologie du corpus utilisé et à son arrangement. Une définition de ces classifications peut ainsi être donnée :

- Classifications stylistiques : établies par l'observation de similitudes de modénature, de programme, de chronologie.
- Classifications typologiques : établies par l'observation de similitudes de morphologie et de fonction du corpus.

**Classifications stylistiques**

L'étude de Z.Dmochowski est l'archétype de ces classifications dont nous qualifions la logique de *stylistique,* classifications dont nous venons de voir les origines. On pourra également se reporter à [Chattopadhyay, 1997] pour une discussion sur l'opposition entre classification stylistique (basée sur l'histoire comme critère et la géographie comme division) et classification d'artefacts. Une description plus détaillée du travail de Zbigniew Dmochowski est présentée en **annexe 2**. Cette annexe, outre qu'elle nous permet de présenter le contexte des terrains d'expérimentations de notre travail, nous donnera la possibilité de mettre l'accent sur les nombreuses dérogations à la règle que l'on observe lorsque l'on étudie un édifice du tout aux parties, rendant malaisée toute modélisation. Nous l'avons évoqué ci-dessus, la classification de Z.Dmochowski s'inscrit dans une lignée de points de vue sur l'architecture qui tendent à situer l'édifice, certes comme un objet construit, mais surtout comme résultant d'une conjonction de facteurs: le style, l'usage, l'époque par exemple. Un regard prolongé sur cette classification permet de noter qu'en définitive l'édifice est décrit comme un tout dont les mutations à travers l'histoire, qu'elles résultent de renversements de mode ou de catastrophes, sont traduites par ce qu'il advient des parties de l'édifice. Une description typo-morphologique des éléments constituant l'édifice physique n'en est donc pas absente. Elle intervient cependant non comme déterminant des différences ou relevant des similitudes entre tel ou tel élément de corpus mais comme référent interne à l'édifice.
Une classification de ce type prend comme item de base un édifice évoluant dans le temps. Elle est apte à fédérer un nombre important d'informations hétérogènes se rapportant en particulier à l'histoire de l'édifice.
Mais résumons nous, il ne nous appartient pas de discuter de son bien-fondé mais de discuter de son adéquation avec le formalisme objet, formalisme de représentation des connaissances adopté dans le cadre de ce travail. Illustrés par l'annexe consacrée à la classification de Z.Dmochowski; trois points de rupture avec les principes de ce formalisme tel que les décrit [Ducournau et al, 1998] nous conduisent à en douter :

- La réification des concepts en jeu est malaisée car ceux-ci ne correspondent pas nécessairement à des objets du monde réel
- L'incomplétude ou l'imprécision du contenu exact d'un concept rendent difficile une description autonome de ses propriétés et de son comportement
- L'organisation quasi horizontale des concepts, souvent liée à une étude chronologique, ne permet pas de tirer partie d'un mécanisme clé du formalisme objet: l'héritage.

Le formalisme objet nous semble par conséquent peu adéquat à représenter les classifications stylistiques telles que nous les avons définies. Nous devons noter qu'une classification stylistique procède de l'observation de l'extension des concepts. Ceux-ci sont isolés non par ce qu'ils ont réellement en partage (propriétés et comportement, au sens du contenu physique réel) mais par la désignation, hors contexte de l'objet, d'une étiquette commune relative à leur style, leur usage, etc.. Cette étiquette correspond effectivement à un point de vue. Mais il nous semble que ce point de vue est extérieur à l'objet lui-même[165].. De plus, l'édifice réel recèle le plus souvent des éléments relatifs à plusieurs classes

[165] Ce point de vue traite non pas de l'objet (concept à représenter, à réifier dans le cadre de mon travail) mais des objets. Autrement dit, ce point de vue donne naissance à des concepts non-réifiables qui chez Dmochowski tiennent à la fois du style de l'époque et de l'usage et qui ne permettent pas de décrire exhaustivement un édifice réel. Au contraire, chaque édifice réel est

stylistiques. Sa classification correspond alors à sa forme à la naissance, et sa description fixe les éléments qu'il recèle et qui correspondent à d'autres classes stylistiques. On voit donc apparaître deux nouveaux obstacles à une modélisation objet :

- Concepts isolés selon un point de vue extérieur aux objets du monde réel,
- Concepts isolés, mais instances imbriquées (Autrement dit, l'édifice réel vu comme instance d'une classe héritant de toutes les classes..).

Si le formalisme objet peut sans doute être utilisé pour organiser et représenter les connaissances relatives aux familles stylistiques, il apparaît clairement que cela ne peut se faire qu'en séparant cette connaissances des édifice réels, autrement dit en réappliquant un mécanisme intensionnel aux seuls concepts relatifs aux styles, usages, etc.. Encore se heurtera t'on à la difficulté qu'évoque [Ducournau et al, 1998] : L'idée est elle une chose ?
Notre travail visant à étudier l'édifice réel, nous ne poursuivons pas cette démarche.

**Classifications typologiques**

Nous avons déjà établi l'ascendance historique des classifications typologiques en architecture en incluant dans celles-ci la théorie architecturale de Viollet Le Duc. Dans la droite ligne de son œuvre, Auguste Choisy [Choisy, 1899/1991] définit dans son histoire de l'architecture des éléments de corpus puis en présente des arrangements. La présentation qu'il propose de chaque mouvement architectural cité commence par une description du corpus d'objets élémentaires en jeu et des procédés de construction correspondant. L'auteur décrit ensuite des édifices particuliers puis enfin termine par un aperçu de l'historique et de la géographie du mouvement considéré.
Aujourd'hui, les classifications typologiques en architecture sont nombreuses mais leurs logiques parfois très dissemblables. Nous pouvons cependant cerner deux approches différentes :

- Des classifications typologiques qui procèdent par extraction de caractéristiques partagées observées sur un ensemble d'édifices,
- Des classifications typologiques qui procèdent par définition de modèles théoriques relatifs au corpus architectural et à son vocabulaire.

Dans le premier cas la classification est le plus souvent liée à un travail d'inventorisation au travers duquel se fait jour la nécessité d'introduire une logique de présentation ou de regroupement des observations. Cette démarche est par exemple illustrée par l'expérience de [Baculo Giusti et al, 1995] sur la représentation du tissu urbain de la ville de Naples. Les auteurs décrivent l'objectif de leur ouvrage dans ces termes : *"…investigate whether this city has a form -and if so where, how and why this exists; whether it can be represented; and if it is possible to define one or more identities for it"*.[ Baculo Giusti et al, 1995, p35]
Cette définition de l'identité de la ville, les auteurs la recherchent en proposant une typologie des couvertures et des élévations d'édifices, qu'ils nomment éléments paradigmatiques [Baculo Giusti et al, 1995, p48], rapprochant ainsi leur démarche de la définition d'un jeu de prototypes. La définition d'un prototype est faite sur la base de la présence d'un ensemble d'éléments atomiques d'architecture (fenêtre à corniche, fenêtre à tympan, couverture plane à corps technique, etc…). A chaque spécificité morphologique ainsi isolée correspond une forme graphique iconique utilisée dans la représentation de l'édifice. Celui-ci n'existe pas en tant que tout mais comme l'addition de plusieurs prototypes, à l'exception marquante d'édifices *particuliers* qui sont érigés de fait en prototype unique. Il n'est pas fait référence à une organisation hiérarchique de ces prototypes.
Cette classification introduit donc bien l'idée qu'il existe une morphologie partagée entre plusieurs édifices. Son utilisation, clairement pertinente au vu des objectifs de l'étude, pose cependant dans le cadre plus général d'une classification typo-morphologique des éléments d'architecture quelques questions:

- Le corpus d'éléments paradigmatiques est construit à partir de l'observation d'un ensemble limité d'objets.
- Aucun lien (d'appartenance ou de dérivation) n'est établi entre les éléments.

observé à l'aune de cette classification stylistique puis placé sous une étiquette et enfin quelquefois décrit dans ce qui le différencie de la forme théorique placée sous l'étiquette, sans que cette forme théorique n'ait été réellement définie.

- La présence sur l'édifice de deux éléments différents de même catégorie n'est pas autorisée (corniche continues et dicontinues par exemple).
- Le corpus d'éléments paradigmatiques ne s'applique qu'à une partie des objets représentés.

A l'opposé de la démarche que nous venons de voir, l'ouvrage de Jean-Marie Pérouse de Montclos [Pérouse De Montclos, 1988] décrit un corpus architectural théorique et en propose des illustrations sur tel ou tel édifice. Basé sur une étude du vocabulaire attaché à la description du patrimoine architectural, l'ouvrage propose d'une part une classification, par *distribution dans l'édifice*, des espaces clos; et d'autre part une classification, par *propriétés partagées*, d'éléments formels ou structurels. Nous revenons ultérieurement sur l'utilisation du vocabulaire qu'établit ce travail. Il importe néanmoins ici de décrire la façon dont l'auteur distingue et organise ce vocabulaire. Jean-Marie Pérouse de Montclos isole, par chapitres successifs, un ensemble de concepts génériques qui correspondent soit à une analyse par la fonction (structurelle) du corpus architectural (Couvrement, Baie, Support, etc..) soit à une analyse par la fonction d'usage de l'édifice (Architecture religieuse, publique, etc..). Cette première différence s'explique par la différence d'échelle et de type de vocabulaire concernés. Dans le premier cas, l'auteur s'attache à déterminer un ensemble de concepts relatifs aux seuls objets physiques et à leur positionnement, dans le deuxième cas il propose une définition de concepts relatifs à l'organisation de l'espace, à son utilisation, et s'adresse à l'édifice plus qu'aux objets qui le composent.
L'ouvrage comprend trois parties :

- Vocabulaire commun à tous les édifices (à la fois analyse du corpus architectural, de sa construction et de l'édifice),
- Vocabulaire des différentes parties d'un édifice (analyse du corpus architectural),
- Vocabulaire propre à certains édifices (analyse de l'édifice).

Bien que leur contenu corresponde en gros aux analyses rappelées entre parenthèses ci-dessus, on notera que chaque partie est liée de façon constante aux autres.
Chaque chapitre consacré à une analyse par la fonction (structurelle) du corpus architectural comprend au moins deux sous-chapitres importants : *variétés* et *parties*. Le premier distingue à partir du concept générique tel que défini en tête de chapitre des types d'éléments concrets (par opposition au concept générique que l'on peut par analogie avec le formalisme objet qualifier d'abstrait). Ces éléments sont distingués essentiellement par addition de propriétés morphologiques (variétés d'après le tracé et la forme), et quelquefois également d'après la structure. La plupart des définitions incluent des règles précises permettant de placer l'élément à l'intérieur d'un type ou de le rejeter. Ces règles lient la morphologie observée à la fois à la fonction de l'élément et à sa forme, comme par exemple : *"Le linteau a, par définition, un soffite, ne pas confondre l'arc monolithe avec le linteau délardé en arc"* [ Pérouse De Montclos, 1988, p266]. La forme et le tracé interviennent comme ligne de division de cette classification après (plus bas dans la hiérarchie par analogie avec le formalisme objet) la fonction structurelle. La définition de variétés d'un type s'appuie donc pour Jean-Marie Pérouse de Montclos à la fois sur la forme de l'objet et sur sa fonction (attitude assez rare et pas toujours simple à prendre en compte dans la modélisation du corpus), elle se construit en intension.
Parallèlement aux variétés d'un type, Jean-Marie Pérouse de Montclos décrit ce qu'il appelle les parties (ou quelquefois faces) correspondant globalement au type générique.
Ces parties correspondent selon le type étudié à :

- Des particularités géométriques (l'arasement, ou face supérieure horizontale d'un linteau ou d'une plate-bande, exceptionnellement d'un arc ou d'une voûte [Pérouse De Montclos, 1988, p275]).
- Des éléments de décor (l'archivolte, corps de moulure porté par le front d'un arc ou d'une voussure [Pérouse De Montclos, 1988, p275]).
- Des accessoires fonctionnels, quelquefois décrits dans un sous-chapitre spécifique (la ruellée, chape en mortier posée sur un rampant … [Pérouse De Montclos, 1988, p340]).

D'autres sous-chapitres viennent compléter la description du type générique comme par exemple le vocabulaire de la charge et de la poussée dans le cas des supports.

Point important, il n'est le plus souvent pas établi de lien direct entre concepts *décrit par la variété* et concepts *décrits par parties*. Comme on le verra dans la partie "projet " du présent document, la modélisation du corpus architectural que nous proposons nécessitera ici une forte interprétation.
L'ouvrage de Jean-Marie Pérouse de Montclos fournit plusieurs propositions essentielles dans le cadre du travail que nous présentons. Il décrit d'abord l'édifice comme un ensemble d'éléments d'architecture atomiques mis en relation au travers de règles de composition explicitement nommées et décrites. Il décrit ensuite beaucoup d'éléments de corpus en s'attachant d'une part à les regrouper par type (exprimant une similitude de fonction), et d'autre part à les distinguer par adjonction de propriétés morphologiques. Ce faisant, il résout en tout cas partiellement une contradiction omniprésente dans l'organisation d'un corpus architectural : l'objet est-il fonction ou est il forme? Il décrit également des parties attachées de façon générique aux variétés des éléments de corpus sans toutefois expliciter les liens entre variétés et parties. Enfin, il place clairement l'édifice comme un tout dont un vocabulaire spécifique relate l'usage ou le programme et un autre la morphologie.
Plusieurs travaux de recherche récents, proches du sujet dont nous traitons, se basent également sur une classification typologique des éléments d'architecture par définition de modèles théoriques. Leur logique de division correspond à un double point de vue :

- Structure / fonction d'éléments physiques
- Divisions de l'espace / fonction (usage) des espaces créés

On retrouve cette approche par exemple dans [Donath et al, 1997]. Les auteurs proposent une classification hiérarchique de concepts architecturaux selon deux ordres principaux, la "pièce" et "l'élément". La taxinomie d'éléments distingue au premier niveau quatre types: l'objet maître, l'objet adjoint, l'objet inclus, la surface. La hiérarchie des objets maîtres correspond à une description structurelle de l'édifice (plancher, poutres, etc..). Les objets adjoints sont les équipements (installations électriques et sanitaires, etc..), les objets inclus des éléments de morphologie non structurels (ouvertures, rampes d'escaliers, etc..).
La thèse de S.Hanrot apporte une contribution intéressante puisque l'auteur définit les jeux de propriétés des objets architecturaux dans des concepts tiers appelés vues architecturales et informant chaque élément de la hiérarchie des objets architecturaux sur sa forme, sa fonction, etc..Ce faisant, l'auteur introduit l'idée de familles de propriétés vues comme des outils de la hiérarchie principales des objets architecturaux, idée séduisante au premier abord. Mais en ne donnant aucune clé sur la logique de spécialisation de cette hiérarchie principale, de sorte que se retrouvent par exemple comme héritant d'un même concept, celui de *bâtiment*, deux concepts bien différents, la demeure et l'hôpital [Hanrot, 19.., p157]. Disons d'abord que si l'on comprend que le second est caractérisé par un usage, on ne voit pas vraiment ce qu'il en est pour le premier. Mais disons surtout que l'approche de S.Hanrot, tout à fait logique dans le cadre de la conception architecturale qui était le sien, ne nous semble pas valide dans le cadre du patrimoine architectural : en effet, quelle demeure n'a pas un jour été hôpital ? Bien sûr, un *programme* d'hôpital diffère d'un *programme* de demeure, mais cette notion de programme ne saurait tenir lieu de ligne de division d'un modèle architectural dédié à l'édifice patrimonial. La (Les) classifications de S.Hanrot est à la fois typo-morphologique (notion de forme et de fonction) mais aussi liée à l'usage de l'édifice. Elle pose donc, comme les références citées précédemment, deux problèmes qu'il va nous falloir trancher:

- Les éléments isolés et classés sont ici éléments physiques atomiques, là usages ou qualités du lieu que ces éléments physiques limitent.
- Les éléments isolés et classés, s'il s'agit d'éléments physiques atomiques, sont choisis ici en raison de leur morphologie, là en raison de leur fonction (principalement structurelle) dans l'édifice.

Comme on le verra dans la suite de ce document, notre proposition vise à considérer une hiérarchie des seuls objets physiques, isolés et- classés à la fois par fonction et morphologie, dans une approche dont la parenté la plus proche est encore l'ouvrage de Jean-Marie Pérouse de Montclos précédemment cité. Enfin, citons la présentation des liens entre style et éléments (typomorphes) d'architecture qu'établissent [Cha et al, 1999, p 634]. Ces auteurs définissent le style d'un édifice comme reconnu par l'observation sur

sa morphologie d'éléments ou de qualités répétitifs. Ils entendent par éléments des objets physiques identifiables (isolables), et les qualités comme les relations qui lient ces éléments entre eux[166].

*LE VOCABULAIRE DE L'ARCHITECTURE*

Il n'entre pas dans le contexte de ce travail de discuter des liens souvent établis entre langage et architecture. Ce sujet, particulièrement discuté dans le champ de la conception architecturale, a fait l'objet de très nombreux travaux. On peut se reporter par exemple à [Tunçer et al, 1999][167]. Bill Hillier décrit précisément les limites à fixer aux rapports établis entre langage et architecture [Hillier,1996]. Il rappelle que les structures d'un langage sont celles qui restreignent le jeu de combinaisons entités / relations possibles et produisent du sens par le biais de ces restrictions. De la même façon, écrit-il, les lois gouvernant le champ de l'architecture fixent un champ de possibles à l'intérieur duquel une architecture peut exister.

A travers ces deux travaux récents apparaissent à la fois la nécessité de prendre en compte le vocabulaire de l'architecture comme élément d'analyse de l'édifice mais aussi les rapports imprécis liant langage, architecture, vocabulaire architectural. Nous allons tenter de mieux cerner le rôle du vocabulaire architectural dans la description de l'édifice construit, et comment sa désignation peut servir l'analyse.

Jean Cuisenier rappelle que trois moyens de traitement des faits architecturaux se sont historiquement complétés dans l'analyse du patrimoine bâti [Cuisenier, 1991, p19] :

- Description de la morphologie visible de l'édifice et des blocs qui le composent,
- Recueil de données dialectales,
- Recueil aux fins de comparaison de données architecturales empiriquement observées.

Ces moyens de traitement des faits architecturaux doivent pour l'auteur contribuer à discerner sous la structure patente (l'édifice observé) des structures latentes (la logique de composition de l'édifice) [Cuisenier, 1991, p21].

Nous proposons en **annexe 3** une analyse rapide des aspects de l'ouvrage de Jean Cuisenier qui nous intéresse, mais ne posons ici que les principes sur lesquels nous nous appuierons:

- Le vocabulaire de l'architecture est un outil de caractérisation du signe architectural (objets physiques, composition, valeurs d'usages d'un lieu, détermination de relations spatiales, etc.)
- Un vocable n'est utile à l'analyse que fixé à l'intérieur d'un champ sémantique
- La désignation de concepts architecturaux par un jeu de termes correspond à un objectif de communication entre acteurs. Elle révèle par conséquent leur angle d'analyse spécifique.

Nous avons, au chapitre précèdent, introduit l'idée que s'opposent classifications stylistiques et morphologiques. A la suite de Jean Cuisenier, nous pouvons de la même façon observer que le vocabulaire de l'architecture fait en particulier l'objet d'analyses centrées sur l'agencement des espaces ou sur la définition d'éléments de morphologie. Dans le premier cas, on pourra s'intéresser aux travaux de recherche menées à la Bartlett School of Architecture de Londres par les membres du Space Syntax laboratory [Hillier, 1996] [Hillier et al, 1997] [Major et al,1997]. Ils posent ainsi une définition de leur approche[168] : "*Space syntax […] is a tool founded on the simple idea that the way we design space is fundamental to the way we use it*". En citant ici leur démarche, nous souhaitons avant tout signaler qu'un

[166] Leur proposition fait donc référence d'une part à un corpus d'objets qualifiés et d'autre part à une logique de composition. Leur définition du style prend alors la forme suivante : Style (N) = {(UM),(UF)} ; où F sont les éléments de corpus, M les édifices, N le style. Les éléments du corpus et leur description générique y apparaissent donc comme un élément déterminant dans la qualification stylistique de l'édifice, CQFD?

[167] Les auteurs typent des jeux d'éléments de vocabulaire en fonction de cohérences de point de vue sur la liste d'éléments. Des abstractions, pour reprendre le terme qu'ils utilisent, représentent un ensemble de types et de relations liant ces types. Ces abstractions rassemblent des types pour gérer un aspect, ou point de vue, sur l'édifice (Ce principe, on le verra, peut être rapproché du travail de P.Boudon cité plus loin). Le vocabulaire attaché à chaque abstraction est spécifique au domaine que cette abstraction est censée représentée. Les auteurs s'appliquent à rechercher des similitudes de schéma de relations entre différentes abstractions qu'ils cherchent à exprimer par un métalangage qui serait une composition des langages propres à chaque abstraction. Leur contribution présente le langage de l'architecture comme orienté points de vue, et distingue bien un vocabulaire descriptif permettant d'identifier ce qu'ils nomment un type (au sens typo-morphologie ou style) et une syntaxe utilisée pour qualifier un type et le comparer à d'autres.

[168] Ils proposent un ensemble de mécanismes d'analyse visant à étudier et qualifier l'usage de l'espace en observant à l'échelle de la ville ou de l'édifice les rapports entre positions relatives des lieux et densité d'utilisation. Décrivant chaque lieu par ses relations au système global (la ville ou l'édifice), ils lui attachent un ensemble de valeurs (intégration / ségrégation / profondeur) et le questionne au vu de l'usage social du lieu.

important courant de recherche s'intéresse aujourd'hui non pas aux vocables du langage architectural mais à une ré interprétation de concepts issus du champ de la sociologie urbaine. On remarquera toutefois que ce courant se positionne plus sur des problématiques de conception que de relevé ou d'études patrimoniales à proprement parler. Revenons donc à notre propos principal.
Nous nous attacherons donc ici essentiellement à présenter les travaux de référence sur le vocabulaire de l'architecte comme outil d'analyse de la forme architecturale. Notre position quant à l'utilisation de ce vocabulaire sera explicitée dans la partie de ce document consacrée à l'état d'avancement du projet. Il nous importe dans un premier temps d'établir en quoi, dans notre champ d'application, le vocabulaire peut être utilisé afin de désigner de façon non ambiguë un ensemble de concepts sur lesquels travailler.
L'écriture, *conservation fidèle de la parole*[169], est le moyen par lequel l'être humain a su expliciter, exprimer et pérenniser sa pensée. Son évolution, du pictogramme vers l'orthographe, s'est faite par nécessité d'élaborer un outil de représentation non ambiguë de la pensée. Le vocabulaire, et la grammaire qui en détermine les relations de dépendance, ont pour vocation de donner une solution unique au jeu de signes que représente une construction sémantique écrite.
Le déchiffrement des écritures tel que l'analyse Ernst Doblhofer[170] est le travail grâce auquel un ensemble de signes graphiques est structuré a posteriori comme un langage permettant d'isoler des contenus sémantiques.
Cette approche illustre le travail, spécifique au champ de l'architecture patrimoniale, que mène le conservateur, l'architecte ou l'archéologue confronté par exemple à un édifice partiellement ruiné ou remanié. En effet, l'édifice patrimonial peut être lu à travers un ensemble de signes formels, physiquement présents à l'observation. Ces signes contribuent à révéler une réalité partiellement disparue: les phases de l'évolution de l'édifice. Bien souvent ces signes ne suffisent pas pour proposer un tout issu de la seule observation des vestiges. L'analogie, ou appel à un corpus de connaissances, et les nouveaux signes qu'elle introduit, sert alors de base à l'élaboration d'une hypothèse de reconstruction, à l'image du processus de complètement d'une phrase en cours de déchiffrement[171].
Le vocabulaire de l'architecte correspond à cet ensemble de signes formels auxquels les architectes, par l'écrit depuis Vitruve [Vitruve, 1684/1988], donnent des noms. Il se situe comme un moyen de description de l'édifice, a priori ou a posteriori. La complexité du vocabulaire architectural est lié à ce qu'il décrit plusieurs points de vue :

- L'arrangement spatial global de l'édifice ("parti architectural").
- Ses parties du point de vue de l'usage ("le déambulatoire").
- Ses parties du point de vue de la composition ("le massif-antérieur").
- Ses parties du point de vue structurel ("La charpente").
- Ses composants atomiques du point de vue morpho-structurel ("La voûte- l'arc").
- Ses composants atomiques du point de vue formel ("le tore de la base").
- Les parties de ces composants ("l'intrados de l'arc").
- Les qualificatifs de ces composants ("l'arc aveugle").
- Une typologie des arrangements, composants ou parties ("l'arc plein cintre, le temple diptère").
- L'appartenance à un style ("la colonne dorique").
- La décoration, qu'elle se rapporte à un composant ("corbeau à volutes") ou aux motifs primaires du décor ("la doucine").
- Les matériaux de construction ("la brique"), leur mise en œuvre globale ("appareil un sur deux") et leur arrangement ("boutisse").
- Le positionnement des composants atomiques les uns par rapport aux autres ("à l'aplomb de").
- Le positionnement des parties par rapport au tout ("flanqué de").

Encore cette liste ne fait-elle aucune mention du vocabulaire qualificatif des espaces bâtis, de leur inscription dans un environnement, etc…Cette première liste ne reprend, et de façon non exhaustive, que les principales familles de termes qu'inventorient ou utilisent [Adam, 1989] et [Pérouse De Montclos,

[169] in Ernst Doblhofer; « Le déchiffrement des écritures », éditions Arthaud, 1959, p17.
[170] op cit.
171 On retrouve de nombreux exemples de cette démarche dans [Démians d'Archimbaud, 1987] : traces d'arrachements de voûtes révélant une nef de deux travées voûtées [Démians d'Archimbaud, 1987, p 46], traces d'arcs-diaphragmes conservées jusque dans l'appareil d'une tour d'angle démontrant leur installation précoce [op. cit, p 42].

1988]. Bien entendu un même terme architectural fait référence, a des implications, sur plusieurs de ces familles : "l'appareil à bossages" se rapporte à la technique de mise en œuvre du matériau "Pierre", concerne le composant "Mur", exclue l'appartenance de celui-ci au style "Gothique", se retrouve le plus souvent comme "décor" de la partie "rez-de-chaussée" et "étage en entresol", etc.
Dans l'introduction de son "Principes d'analyse scientifique - Architecture - vocabulaire" [Pérouse De Montclos, 1988], Jean-Marie Pérouse De Montclos situe *les principes de l'architecture* d'André Félibien (1676) comme le premier ouvrage traitant spécifiquement des termes d'art. Le lexique que propose Félibien sera revu, complété en partie par ses successeurs en France et ailleurs. Jean-Marie Pérouse De Montclos fait remonter à la fin du XVIIIème siècle, avec les premiers pas de l'archéologie scientifique, la séparation de fait entre vocabulaire orienté métiers et vocabulaire orienté art.
Cette distinction fallacieuse, en reprenant le terme qu'utilise l'auteur, entre art et technique compromet durablement, nous dit-il, la linguistique avec la stylistique. Au milieu du dix-neuvième siècle, le comité historique des arts et des monuments se voit chargé d'une mission d'inventorisation du vocabulaire architectural attaché au patrimoine bâti, travail qui préfigure le chantier des principes d'analyse scientifiques de l'inventaire général. Le rôle que doit jouer ce vocabulaire est déjà fixé de façon claire : donner un nom aux formes observées qui n'en ont encore pas. Les efforts de compilation du vocabulaire architectural sont nourris pour ce qui concerne la période classique par les nombreux traités d'architecture qui ont succédé aux premières traductions de Vitruve.
Le vocabulaire de la période médiévale est quant à lui plus sujet à inventions lexicales [Pérouse De Montclos, 1988, p3] a posteriori lorsque le vocabulaire traditionnel s'avère insuffisant. En citant les contributions de Viollet le Duc (Dictionnaire raisonné de l'architecture, 1861) et de Chabat (Dictionnaire, 1875), Jean-Marie Pérouse De Montclos précise que ces auteurs, s'ils s'appliquent à recenser le vocabulaire, n'en donnent cependant pas des définitions univoques. Un foisonnement verbal se crée à leur suite avec le développement des études archéologiques dont l'auteur observe deux conséquences :

- D'une part, une spécialisation du langage qui tend à restreindre les acceptions d'un terme et par là même à exclure de fait un jeu de variantes de ce terme ne correspondant pas par exemple à l'époque observée.
- D'autre part, sous prétexte de simplifications didactiques, un appauvrissement du langage qui se traduit par une incapacité du vocabulaire usuel à définir de façon univoque les parties signifiantes de l'édifice.

Ainsi apparaît la nécessité d'opérer une normalisation scientifique du vocabulaire dont l'auteur nous rappelle que le précis d'architecture de Brutails fixe pour la première fois la méthode en 1908. Brutails, en effet, recense comme ses prédécesseurs un ensemble de vocables. Mais il va au-delà, d'une part en opérant des regroupements sémantiques par observation des différences et des ressemblances entres les éléments que ces vocables décrivent, et d'autre part en choisissant les acceptions d'un terme qui se rapprochent le plus de la réalité observée. Jean-Marie Pérouse De Montclos illustre ce propos par un exemple : "le néologisme clocher-mur s'est ainsi substitué à l'ancien clocher-arcade pour désigner un ouvrage dont les baies ne sont pas nécessairement couvertes d'un arc".
Jean-Marie Pérouse De Montclos situe enfin les règles de différenciation qu'il entend utiliser dans la détermination du vocabulaire qu'il recense. Nous le citons, son propos ne pouvant mieux être présenté : "*Les définitions du vocabulaire que nous présentons ici ne s'appuient que sur des différenciations morphologiques indépendantes de toute qualification historique*. […] *L'identification des formes doit précéder l'interprétation historique, les styles ne s'identifiant que par références à ces formes*". L'auteur rappelle à la suite de Viollet Le Duc le rôle essentiel que joue le parti fonctionnel dans la compréhension de l'édifice, et récuse l'idée souvent répandue par les traités d'architecture eux-mêmes (voir chapitre précédent) qu'à un style correspondrait un vocabulaire spécifique. Chaque terme est ainsi défini eu égard à la détermination d'un couple unique forme / fonction que le texte explicite à la fois par ce qu'il est, et par référence à ce qu'il n'est pas (mais lui ressemble du point de vue formel ou fonctionnel).
Ayant ainsi livré la clé d'une méthode d'identification des termes, l'auteur s'attache ensuite à définir quelle règle il a souhaité suivre pour ordonner ces termes. Jean-Marie Pérouse De Montclos propose une classification, que nous avons déjà évoquée, basée sur deux règles:

- Regroupements distincts de notions relatives aux parties de l'édifice et à leur genre (au sens du parti architectural et du programme auquel il répond).

- Définition d'ensembles sémantiques plaçant chaque notion à l'intérieur de cadres de références condamnant "les incertitudes que l'ordre alphabétique des dictionnaires traditionnels dissimule".

L'auteur s'appuie sur une double bibliographie : une liste d'ouvrages dédiés au recensement du vocabulaire (dictionnaires, lexiques, glossaires) ainsi qu'un ensemble de traités plus généraux ou de relevés et d'archives. En effet, il s'attache à recueillir variantes et synonymes des termes explicités dans le corps de son ouvrage et à donner les références justificatives des choix opérés pour valider ou rejeter l'acception d'un terme.
Notre effort de modélisation du corpus trouve donc dans le travail de l'auteur un ensemble d'outils « d'aide à la réification » fondamentaux :

- Identification de concepts univoques basée sur l'observation de couples forme / fonction.
- Mises en relation des parties de l'édifice au tout qu'il constitue.
- Classification hiérarchique (partielle) de ces concepts.
- Recours tant à une bibliographie figurant un ensemble de modèles théoriques qu'à l'observation de spécimens.

Ce dernier point s'accompagne pour nous bien entendu d'une question : classification intensionnelle ou extensionnelle ? Nous en reparlerons dans le chapitre suivant, consacré aux exigences de modélisation propres au champ de la conservation.
Ayant explicité, au-delà d'un ensemble de termes, une démarche de définitions de concepts architecturaux, Jean-Marie Pérouse De Montclos nous livre une méthodologie de relecture des nombreux travaux d'inventorisation du vocabulaire architectural[172]. Les travaux cités dans ce chapitre établissent plusieurs points clés que nous résumons ici :

- Il existe un vocabulaire des types dont une syntaxe qualifie les arrangements
- L'étude du vocabulaire fait historiquement partie des moyens mis en œuvre pour comprendre l'architecture
- L'affectation d'un vocable à un concept ne suffit pas à déterminer sons sens de façon définitive
- Un vocabulaire des objets physiques existe en parallèle à celui qualifiant les espaces créés.
- Depuis le seizième siècle, l'illustration tend à figurer les premiers et le texte à décrire les seconds.
- Au dix-neuvième siècle apparaît une nouvelle ambiguïté : un vocabulaire du style vient en complément de ceux des formes et des usages.
- Concomitamment, le rôle de la représentation graphique évolue vers plus de précision, et devient une expression non ambiguë du projet.
- Depuis le milieu du dix-neuvième siècle, le vocabulaire attaché au patrimoine bâti fait l'objet de recherches visant à nommer le formes observées.
- Cette normalisation scientifique du vocabulaire s'opère par identification de similitudes entre objets physique élémentaires à nommer et par choix d'une acception proche de la réalité observée.
- Une méthode d'identification du terme associé à ces concepts se fait jour, celle que propose Jean-Marie Pérouse De Montclos : ceux-ci sont définis par un couple univoque forme / fonction.
- On peut alors rechercher l'articulation entre objets physiques élémentaires et matériaux mis en œuvre.
- Un travail de modélisation sur ces bases devra prendre en compte notamment la difficulté à fixer des limites entre ces concepts.

L'étude du vocabulaire architectural nous renvoie à une nouvelle question: les concepts en jeu, et leur mode de désignation, sont ils propres au domaine patrimonial ou relèvent t'ils d'un unique et permanent art de bâtir, pour reprendre le terme de Jean Cuisenier ? C'est ce dont discutons en section 3.2.3.

[172] Nous reportons en **annexe 9** un certain nombre d'exemples qui peuvent illustrer ce point, et qui mettent l'accent également sur les problèmes que pose le passage d'un terme même non-ambigu à sa description par une classe au sens de la POO.

*ANNEXE 2 : LA CLASSIFICATION DE Z.DMOCHOWSKI*

Je cite ici une petite moitié des types isolés par l'auteur [Dmochowski, 1956] afin d'illustrer par l'exemple quelles contradictions peuvent apparaître entre l'intitulé d'un type et les édifices réels observés.

**Partie 1 : Roman**

**G1: Pré-roman : les débuts de la maçonnerie en pierres**

La rotonde Notre Dame située sur la colline de Wavel remonte au Xéme siècle. De plan circulaire, elle est couverte par un dôme probablement inspiré de modèles byzantins réintroduits en Italie au Viéme siècle. Deux points sont ici à noter :

- La fonction de la structure semi-circulaire flanquant l'édifice au sud n'a toujours pas été précisée.
- L'arc couvrant la baie sud-est est de type plein cintre, monté en pierres posées côte à côte. Sa clé est composée d'un conglomérat de dallettes montées en chevrons.
  Cette technique, primitive en comparaison au dessin de l'édifice, s'apparente à l'architecture anglo-saxonne de l'époque.

**G2: Eglises basilicales à transept**

Le niveau initial du sol de l'église Św Andrzej (1086) a été repéré à neuf pieds sous son niveau actuel. L'église était le centre d'une petite place forte, ce qu'illustre la présence de meurtrières au niveau du sol. Deux particularités différencient par ailleurs cette église d'un modèle d'église basilicale à transept :

- Les murs intérieurs, construits à l'origine en lits alternés de grés rose et de pierres calcaires blanches, ont été complètement recouverts de stuc à l'époque baroque.
- Le prolongement inhabituel du chœur se retrouve dans beaucoup d'églises de Cracovie. Il était dû au statut de capitale qu'avait alors la ville et par conséquent à l'importance en nombre du clergé qui y séjournait.


*Figure 87 : Un prolongement inhabituel du choeur*

**G3: Eglise basilicales sans transept**

L'église bénédictine de Tyniec est aujourd'hui un édifice très représentatif de l'architecture baroque. Pourtant des fouilles ont permis de découvrir dans l'épaisseur des murs actuels de vestiges de la maçonnerie romane originelle ainsi que des fûts de colonne.

**G4: Eglises circulaires à cellule unique.**

L'église St Procopius de Strzelno fut construite en 1160 en opus spicatum de granit et de briques (Pour la définition des opus, voir [Adam, 1989] ). Une tour dite d'époque gothique fut ajoutée à l'édifice original qui était flanqué de deux apses contiguës au nord dont on a retrouvé des vestiges. Outre sa position très peu usuelle dans la composition de telles églises, cette tour ronde présente deux particularités :

- ✓ Elle est raidie par un contrefort unique, désaxé
- ✓ Elle se termine par un registre de plan carré

**G5: Eglises rectangulaires à cellule unique**

Fondée probablement au XIème siècle, l'église St Adalbert dût être profondément transformée au XVIème siècle pour l'adapter au niveau alors arasé du *Rynek Główny* sur lequel elle est située. Elle fût rehaussée de quatre pieds, sa couverture initiale faite de tavaillons de bois remplacée par un dôme circulaire, et une extension perpendiculaire à l'axe de l'édifice original fut construite pour ouvrir un nouveau portail baroque.

*Figure 88 : Eglise Saint Adalbert, accès d'origine et niveau actuel de la place, plan de l'édifice*

**G6: Eglises Cisterciennes**

A Mogiła, la construction de l'église du monastère cistercien s'acheva en 1243. Les proportions relatives de la nef et des collatéraux sont celles d'une église romane typique (1:2), toutefois le couvrement des collatéraux est de facture gothique (voûtes d'ogives).

**<u>Partie 2: Gothique</u>**

**G9: Eglises des ordres préchants**

La frise décorant le gâble nord du transept est formée d'arcs de réseau entrecroisés se situant comme une évolution du motif roman de la frise d'arceaux.

Une tour érigée à la croisée du transept rattachait de façon plus proche cet édifice aux églises de même type bâties en Europe de l'ouest à la même époque. Elle s'est effondrée en 1484.

**G10: Eglises basilicales**

L'église Ste Marie, située sur le *Rynek Główny* de Cracovie, est un bon exemple d'adaptations successives d'un édifice. Conçue comme église-halle aux débuts de sa construction, elle suit en définitive un plan basilical dont le presbytère, sans déambulatoire, est plus allongé que de coutume. La tour nord de son massif-antérieur fut couverte en plomb selon un modèle issu de l'école tchèque dès 1478. La tour sud, achevée en 1592, fut couverte, renversement de mode oblige, d'un dôme de facture pré-baroque. Aux XVIéme et XVIIèmes siècles des chapelles furent édifiées entre les contreforts, les hautes baies originales étant transformées en portes donnant accès aux chapelles.

Les arcs-boutants, forme gothique par excellence, sont ici remplacés par de petits contreforts flanquant les piliers qui soutiennent les retombées de voûtes. Ce système constructif local avait été expérimenté dans la cathédrale royale de Wavel où les constructeurs, doutant de leur nouvelle technique, avaient doublé ces contreforts intérieurs d'arcs boutant placés exceptionnellement sous la couverture des collatéraux.

L'église Ste Marie appartient au type des "églises basilicales gothiques". On voit pourtant ici que :
- ✓ son plan basilical n'apparaît qu'en cours de constructions
- ✓ un élément essentiel de l'architecture gothique n'y apparaît pas (l'arc boutant)
- ✓ Un élément incongru y est ajouté : un dôme couvrant la tour sud

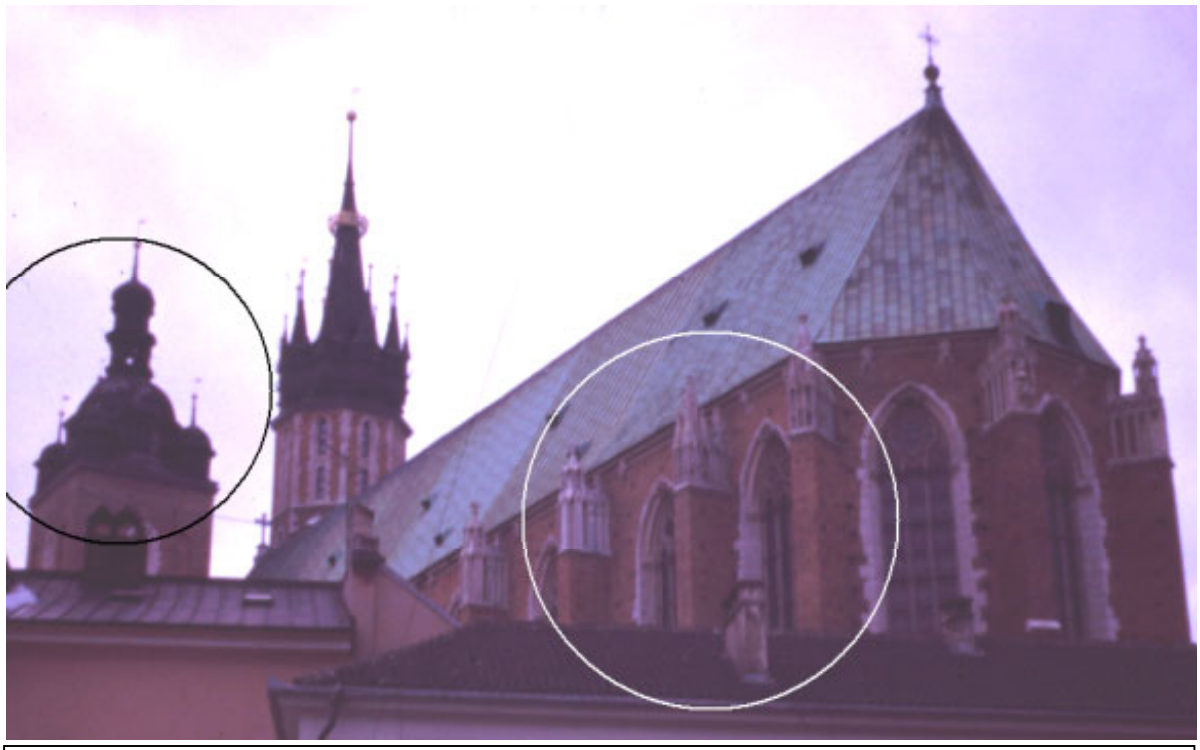


*Figure 89 : L'église gothique Ste Marie de Cracovie, contreforts enlieu et place des arcs-boutants et couverture du clocher Sud par un dôme*

**G11: Eglise -halle à trois nefs**

L'église Ste Croix et St Bartholomé de Wrocław, fondée en 1288, devait quant à elle au départ, s'inspirant d'exemples puisés en Europe occidentale, suivre le modèle basilical. C'est un des rares exemples d'église à deux étages, l'étage inférieur étant consacré à St Bartholomé, l'étage supérieur à Ste Croix. Cette église est également marquée par l'utilisation d'une voûte ogivale tripartite dite "voûte Piast" qui pourtant n'apparaît pas dans l'intitulé de sa classification.

**G17: Hôtels de ville**

Fondé en 1383 par le roi bâtisseur Casimir le Grand, l'ancien hôtel de ville de Cracovie fut plusieurs fois remanié et agrandi avant d'être en grande partie détruit en 1817. Seul demeure aujourd'hui le beffroi original sur lequel on peut lire dans le retrait de sa partie supérieure par rapport au corps principal la trace d'un ancien chemin de ronde.

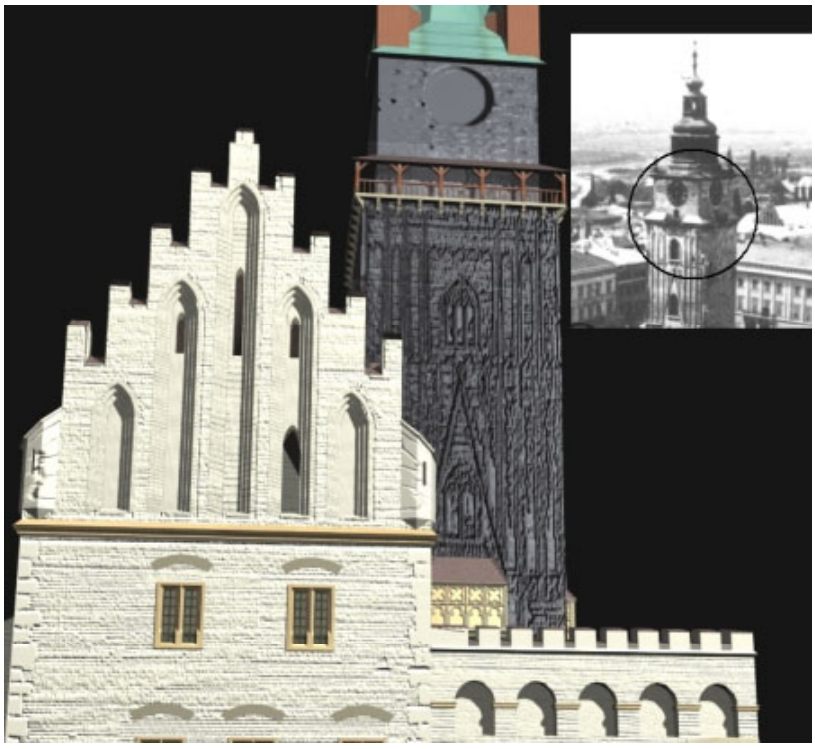


*Figure 90 : Le beffroi de l'hôtel de ville de Cracovie, restitution du chemin de ronde (im C.RAdi, MAYA), et cliché du XIXéme siècle*

**G18: Bâtiments publics**
Fondé en 1364 par Casimir le Grand, l'université qui accueillit Copernic fût reconstruite en 1494 dans sa forme actuelle. L'accès à l'étage se fait dans la cour fermée de l'édifice par une galerie extérieure. Depuis cette galerie, de petits escaliers indépendants mènent au dernier niveau. De fins potelets transmettaient la charge du débord important de la toiture au parapet de la galerie. Cette même solution technique, des potelets appuyant le débord de toiture sur la galerie, se retrouve au château royal de Wavel. Mais leur emploi est ici antérieur, et leur démontage au XIXème siècle (remplacés par des jambes de force en bois) masque une similitude évidente, et un rapport d'antériorité intéressant que la classification n'établit pas.

**Part 3 : Renaissance**
**G20-1 Le cas du château royal**
Trois galeries superposées furent construites au début de la renaissance pour relier les bâtiments existants autour d'une cour close de façon formelle par une galerie-haute au sud. La proportion dictant les hauteurs relatives de ces trois galeries suit le nombre d'or, s'inscrivant dans la logique de l'époque. La couverture très pentue, d'inspiration gothique, est supportée par une colonnade de façon indirecte : de petits supports en forme de vase sont placés entre les chapiteaux et le débord de toiture pour éviter que l'ombre portée du toit ne cache la modénature à la manière toscane des chapiteaux.
Chaque colonne est raidie par un anneau doré à mi-hauteur, l'ensemble de facture renaissance garde des affinités nettes avec la tradition locale de la construction en bois *"une touche renaissance tissée sur un fond gothique"* [Dmochowski, 1956, p178]. La décoration notamment de jambages des baies reprend des motifs d'inspiration gothique créés par des sculpteurs travaillant le bois et les réutilisant ici sur de la pierre. Ces motifs sont complétés par des figures proprement renaissance (festons, coquilles, etc..).
Deux points sont ici à noter :
- ✓ L'auteur, dérogeant à sa propre règle, érige un édifice réel en type.
- ✓ Dans son histoire comme dans sa modénature, l'édifice réel lui apparaît comme trop "entre deux tendances" pour le qualifier au travers d'un type générique.

Autrement dit, l'étude approfondie du corpus de l'édifice pousse Zbigniew Dmochowski à s'en servir comme isolant un type à part entière. Seul chaque élément de ce corpus peut ici être isolé et décrit.

**G 20-2 : La chapelle de Sigismund**
Cette chapelle est le premier édifice religieux qui témoigne en Pologne des idées humanistes en architecture. Construit en parements de grés montés par agrafes devant un remplissage de tout-venant, elle est couverte par une coupole en pierres de taille avec pendentifs en briques recouvertes de stuc.
Bien que du point de vue constructif elle soit plutôt archétype de mixité, l'auteur la présente comme un prototype du temple idéal qu'Alberti défend dans le livre VII de son "De re aedificatori" publié en 1485 :
- ✓ Composition de volumes à base de figures géométriques simples (carrés, cercles et divisions du carré).
- ✓ Emploi de la sculpture de préférence à la peinture.
- ✓ Ouvertures placées en partie haute.
- ✓ Dôme à caissons à l'exemple de celui du Panthéon de Rome.
- ✓ Emploi de matériaux nobles.

Ici aussi, l'auteur érige un édifice réel en type, ou plutôt utilise cet édifice pour expliciter ce type.

*Figure 91 : La chapelle de Sigismund*

**G 20-3 : Sukiennice**

A la suite des constructions avec galeries et des chapelles apparaît un troisième motif, marquant l'époque et le lieu, que l'auteur juge déterminant dans l'élaboration du type : les attiques.

Ceux-ci seront utilisés sur tous les bâtiments urbains en motif de couronnement, au dessus de la corniche régnant

sur l'ensemble de l'élévation. Une des raisons essentielles de l'apparition de tels attiques est fonctionnelle: un décret municipal de 1544, "De tectis aedium novarum", stipulait que toute nouvelle construction devait être entièrement enceinte de murs la séparant des constructions voisines. Il s'agissait en fait de limiter les risques de propagation d'incendies et d'accroître le potentiel défensif. Le décret ne stipulait pas la forme architecturale à donner aux murs de cloisonnement des édifices.

Une morphologie commune fût pourtant rapidement adoptée :

- ✓ Une frise, souvent inspirée de celle que composa Giovanni Maria Padovano pour la reconstruction de Sukiennice en 1555, utilisant l'arcature aveugle comme motif principal.
- ✓ Un couronnement souvent très ouvragé.

Chaque édifice devant être isolé de ses voisins, les murs de refends les séparant durent être montés à la hauteur du faîtage des toits et la pente de ceux ci inversée (faîtages sur les murs de refends, pente parallèle à l'élévation, rive axée au centre de celle-ci et perpendiculaire). L'attique permettait donc de masquer la disposition peu élégante des toits qu'imposa le décret, et une forme architecturale typique de l'époque renaissance à Cracovie trouve son origine non pas dans les principes artistiques de la renaissance mais dans la crainte du feu.

Ici aussi, l'auteur érige un édifice réel en type, ou plutôt utilise cet édifice pour expliciter ce type.

*Figure 92 : L'attique de Sukiennice (gauche), un couronnement archétypique repris dans les maisons urbaines pour masquer le dispositif de refends règlementaires.*
*L'attique de l'ancien hôtel de ville de Cracovie, restitution (im C.Radi, MAYA)*

**G 21 : Hôtels de ville**
L'auteur mentionne à nouveau ici l'ancien hôtel de ville de Cracovie et note que l'attique de son extension construite en 1562 reprend le motif de Sukiennice décrit plus haut. Un même édifice, l'ancien hôtel de ville de Cracovie, est donc classé dans deux catégories (Gothique et Renaissance) et sa forme architecturale la plus notable fait référence à une troisième, le motif de l'arcature aveugle de l'attique de Sukiennice.

**G22: Maisons urbaines**
Les galeries ouvertes sur la rue ne seront pas utilisées à Cracovie par les architectes de l'époque renaissance. Par contre, lorsque par rattachement de deux sections cadastrales l'espace le permet, des galeries intérieures sur cour seront aménagées. Il faut rappeler que le tracé de la ville (tel que prévu à sa fondation en 1257) donnait à chaque parcelle peu de largeur sur la rue (2 pièces ou 4 fenêtres) mais une longueur suffisante pour bâtir deux édifices l'un derrière l'autre en plaçant une cour entre les deux. Pour plus de détail sur le tracé urbain de Cracovie on peut se reporter à [Jamroz,1983] ou [Łukacz,1998].

**Partie 3 : Baroque**
**G 30 : Eglises à nefs uniques du baroque précoce**
L' église St Pierre et Paul, construite par l'architecte italien Bernadone, pourrait, si l'on s'attache à la composition de son élévation principale, faire penser à une église basilicale a deux collatéraux. Pourtant l'espace dévolu aux collatéraux est ici occupé par deux rangs de chapelles séparées par d'épais murs qui supportent la retombée du voûtement en berceau à lunettes. Cette église n'est donc à nef unique que par transformation d'usage.

**G 31 : Eglises à nefs uniques baroque**
Inspirée de S. Andrea della Valle à Rome, l'église Ste Anne de Cracovie introduit un ensemble de détails architectoniques qui ont vocation à créer une illusion d'espace. Il s'agit d'une réinterprétation du corpus architectural de la renaissance dans un esprit de mise en scène typique de l'époque baroque. On peut noter ici que la position de l'église, le long d'une rue assez étroite, influe sur la composition de son élévation, très peu élégante en dessin. L'architecte montre ici par l'exemple ce que Viollet Le Duc rappellera dans son septième entretien sur l'architecture : un édifice est tridimensionnel, il se lit dans l'espace et non sur une figure plane.

*Figure 93 : Eglise Ste Anne, vue de l'édifice dans l'espace de la rue, fresque intérieure peinte sur un pilier pour simuler l'effet de cannelures.*

## *ANNEXE 3 : VOCABULAIRE ARCHITECTURAL, L'ANALYSE DE J.CUISENIER*

Jean Cuisenier [Cuisenier, 1991] s'intéresse à ce que l'on appelle fréquemment l'architecture vernaculaire. Sa méthode d'investigation s'appuie sur la comparaison de modèles et de prototypes, et sur les rapports qu'un type idéal entretient avec les réalités observées. Il retrace l'historique du travail d'inventorisation mené en France sur l'architecture qu'il préfère appeler ordinaire depuis l'enquête lancée entre 1885 et 1887 à l'appel d'Alfred de Foville[Cuisenier, 1991, p16]. Dès l'introduction de son ouvrage, Jean Cuisenier situe son propos sur le champ d'une sémiotique de l'architecture. Citons-le : "Comme produit des opérations de bâtir, l'œuvre d'architecture appartient, certes, à l'univers du référent, et mène son existence indépendamment des messages échangés dont elle est l'instrument. Mais on se méprendrait sur ses caractères et sa spécificité si on ne reconnaissait pas, selon la belle formule d'Umberto Eco, que normalement nous jouissons de l'architecture comme d'un fait de communication.[…] Caractériser le signe architectural, distinguer le signifiant et le signifié, discerner le dénoté et le connoté, repérer les codes […], voilà autant de défis qu'une sémiotique de l'architecture entend relever."

En quoi l'analyse que propose Jean Cuisenier peut-elle contribuer à élucider les questions que nous souhaitons poser sur la désignation et l'utilisation d'un vocabulaire architectural stable, apte à isoler un jeu de concepts non ambigus, c'est ce dont nous allons maintenant discuter.

Jean Cuisenier s'attache d'abord à définir un modèle qu'il appelle type idéal de la maison rustique auquel il souhaite confronter un ensemble de réalités architecturales. Avant de présenter exhaustivement le modèle normatif que Charles Estienne élabore au seizième siècle, il remarque en préambule que le vocabulaire descriptif de l'édifice recouvre le plus souvent plusieurs champs sémantiques. Jean Cuisenier rappelle que chaque chapitre de l'ouvrage de Charles Estienne commence une investigation philologique tendant à fixer le champ sémantique repéré par un vocable. Cette élucidation des concepts par la philologie ne suffisant pas à déterminer de façon définitive un sens à donner à une notion, Charles Estienne entreprend d'en expliciter le contenu. Etape ultime de sa démarche, la description textuelle du modèle que fournit Charles Estienne permet à Jean Cuisenier de proposer une figuration de ce premier type idéal de la maison rustique. Ce modèle s'apparente pour l'auteur à un système de relations que Charles Estienne n'aurait qu'explicité pour ses contemporains. Jean Cuisenier met en lumière un vocabulaire servant à déterminer des lieux et leurs affectations, et un vocabulaire servant à en déterminer les parties composantes, formalisant ainsi une opposition espace / limites. Le mur est lu par sa fonction dans un agencement de l'espace, mais ne détermine pas le rôle de celui-ci. Partant de l'exemple des cours d'exploitations agricoles observées comparées au modèle normatif de Charles Estienne, l'auteur indique comment il entend rapprocher un spécimen du modèle : par rapprochements liés au programme. C'est donc bien ici le vocabulaire attaché au système de relations et plus généralement aux affectations des espaces qui va servir d'outil d'analyse.

Voulant rechercher les antériorités du modèle de Charles Estienne, de ses contemporains et de ses successeurs, Jean Cuisenier s'intéresse ensuite au type idéal de la maison primitive, ayant espéré au travers du modèle de Charles Estienne qu'une "conception architecturale arrivée à sa forme achevée rende intelligibles les moments élémentaires dont elle est issue" [Cuisenier, 1991, p67]. L'auteur rappelle que cette recherche des origines a souvent servi d'argument initial aux théories architecturales sur la génération des formes. Il cite ainsi une lignée d'auteurs qui depuis les traductions de Vitruve jusqu'à Le Corbusier tentent de légitimer un principe de composition (symétrie, modularité, etc.) par l'interprétation d'observations sur l'habitat primitif.

Jean Cuisenier explore alors la genèse du mouvement, né à la renaissance, qui va conduire les architectes ou maîtres d'œuvres à décrire l'édifice non plus par la seule utilisation du vocabulaire mais par l'adjonction de la figure au texte: articulation raisonnée d'un propos livrant, par le texte, un programme, et l'illustration anticipant, par la figure et la métrique, des œuvres [Cuisenier, 1991, p123]. On remarque ici que si le texte a toujours valeur de qualification des espaces, la forme architecturale, les parties composantes de l'espace s'expriment désormais aussi par le trait. L'auteur cite en exemple de son propos le dispositif que livre l'architecte et théoricien de l'architecture italien Sebastiano Serlio au début du seizième siècle. Celui-ci met au point un ensemble de types architecturaux qu'il décrit par le texte et projette par le dessin. Plans, coupes et façades soutiennent un texte fournissant programme, distribution des pièces et mesures du bâtiment [Cuisenier, 1991, p136]. Jean Cuisenier rappelle le contexte

d'utilisation du traité de Serlio: le texte comme "fil conducteur pour la convention liant un maître d'ouvrage et un maître d'œuvre", la figure comme "moyen pour communiquer […] par le dessin l'intention de l'architecte aux artisans […]".
Il livre ensuite deux clés permettant de mieux saisir l'importance historique du dispositif proposé par Serlio:

- La figure n'est pas ici un cadre rigide à comparer à nos plans d'exécution actuels. En effet, ce sont les maîtres artisans qui veillent au tracé des formes bâties sur le chantier même. Le plan d'architecte fournit alors une conformation à l'édifice et des grandeurs relatives que l'artisan interprète par le tracé et une mesure encore sans étalon.
- Le texte ne s'adresse qu'à ceux qui savent lire. Cette lapalissade rappelle que le dispositif séparant texte et figure correspond de fait à deux partenaires différents, et fournit à chacun une information spécifique. Le texte donne au maître d'ouvrage des indications sur la destination des locaux et leurs dimensions, la figure sert d'aide à la construction pour l'artisan.

Graphiquement, Serlio ne traite le projet qu'en coupes, élévations et plans. L'absence de la perspective dans ce dispositif graphique s'explique naturellement par le destinataire prioritaire des planches dessinées : l'artisan. A quel usage une perspective peut-elle correspondre pour lui?
Cherchant les rapports du modèle Serlien avec les réalités observées, Jean Cuisenier cite trois principes de répartition des lieux qu'un vocabulaire univoque tente de cerner : frontalité, axialité, latéralité.
Mais de quel vocabulaire s'agit il ? Ou plutôt, quel champ sémantique ce vocabulaire recouvre t'il ? Le dispositif Serlien utilise un texte qui a valeur de contrat, d'où le non dit ou l'imprécision doivent être chassés. La figure, elle, n'est utilisée que pour représenter les parties de l'édifice et leur agencement. Ainsi, paradoxalement, le vocabulaire utilisé dans la description de l'édifice ne fait plus référence à un contenu en terme d'objets physiques mais seulement de potentiel d'affectations ou de répartition des lieux. Nous avions déjà relevé l'ambiguïté du vocabulaire architectural et des classifications qu'il soutient pendant l'époque renaissance au chapitre précédent. Le dispositif Serlien, en nous permettant de mieux cerner la demande à laquelle ce vocabulaire répond, nous rappelle que la désignation de concepts architecturaux par un jeu de termes est un exercice finalisé : au maître d'ouvrage l'architecte livre un message sur l'espace bâti, à l'artisan il livre un message sur la forme architecturale, forme qu'il représente par une figure correspondant au savoir-faire, non écrit, de l'artisan. Ainsi se fait jour un problème de taille : nous devrons **nommer** des concepts **observés**, tant sur l'écrit que sur le construit. Nous verrons plus loin comment Jean-Marie Pérouse de Montclos l'a abordé.
Mais revenons d'abord à ce que Jean Cuisenier nous dit de l'utilisation du vocabulaire architectural et de ses transformations. Il rappelle que dès le seizième siècle apparaissent donc des instruments intellectuels pour décrire l'agencement de lieux dans l'espace en utilisant des vocables ou des figures relevant de la composition architecturale. En citant l'encyclopaedia of cottage, farm and villa architecture écrite par l'architecte anglais Loudon au XIXème siècle, Il décrit par un vocabulaire qui leur est attaché des catégories de modèles que Loudon, après beaucoup d'auteurs depuis Serlio, crée en fonction de la classe du maître d'ouvrage dans l'échelle sociale. Cette catégorisation est complétée par des variations stylistiques dont l'architecte anglais se sert comme de termes qualificatifs pour chaque modèle : villa-farm de style anglo-grec, farm-house de style gothique, etc.
Après l'élaboration de types déconnectant formellement la morphologie des parties du lieu et leur affectation, dont Serlio apparaît comme le précurseur, Loudon décrit ici un programme qui, sans jamais faire référence aux objets physiques à mettre en œuvre et à leur logique propre, institue deux pôles distincts de références: l'usage et le style. Les bases de l'éclectisme, forme assumée du façadisme d'aujourd'hui, sont posées. La fin du XIXème siècle voit l'avènement d'une pensée rationnelle et systématique de l'espace que Jean Cuisenier rattache dans le cadre de l'architecture ordinaire à ce qu'il nomme l'hygiénisme des notables et le positivisme des agronomes [Cuisenier, 1991, p227]. Les principes qui président à la disposition de l'édifice idéal évoluent vers une indifférenciation apparente des espaces qui sont disposés en vertu des seules lois de l'hygiène , de la sécurité et de l'efficacité. Une géométrisation encore timide marque les compositions. Jean Cuisenier insiste sur l'utilisation alors faite de la symétrie pour motif central de la composition, une symétrie comprise comme "reproduction exacte à la gauche d'un axe de ce qui est à la droite" et non comme "rapport harmonieux des parties d'un tout" telle que la définit Vitruve. Viollet Le Duc, nous dit Jean Cuisenier, la décrivait comme "une opération tellement banale et insignifiante que les Grecs n'ont pas même au l'idée de la définir"[Cuisenier, 1991, p243]. Avec Monge et la géométrie descriptive les concepteurs disposent alors des moyens de préfigurer

leur projet, de donner une définition complète de l'objet à créer. Ce nouvel outil donne à la sérialisation de la production architecturale un élan fondateur. La figure ne laisse plus de place à l'interprétation de l'artisan. Elle est un signe non ambigu que le concepteur livre au constructeur.
La trame, dont Durand dès 1802 introduit le concept, vient opportunément compléter le dispositif de rationalisation a priori du projet architectural débuté au seizième siècle. Les objets physiques ne sont plus seulement nommés mais surtout montrés, le vocabulaire de la composition, du jeu des formes, peut dès lors remplacer dans la description de l'édifice la désignation de ses parties. Jean Cuisenier détaille, en présentant les plans adoptés pour reconstruire les villages détruits au nord de la France pendant la première guerre mondiale, trois logiques opposées:

- Une logique architecturale orientée par la préoccupation de l'hygiène et du rendement
- Une logique architecturale orientée sur la rationalité des opérations de constructions et sur l'emprunt de quelques éléments à la tradition constructive (chaînes d'angle, encadrements de portes, etc.) locale pour régionaliser l'aspect de l'édifice.
- Une démarche sociale fondée sur l'étude des changements survenus chez chaque maître d'ouvrage et visant à reconstruire non pas à l'identique mais en appliquant des modèles anciens aux situations nouvelles.

On sait les succès contrastés qu'ont eu et qu'ont encore ces différentes démarches. Les citer nous permet d'introduire une nouvelle idée : le vocabulaire de l'architecture réinterprété comme outil de systématisation de la production architecturale. Jean Cuisenier s'attache à rechercher un ordre des lieux qui, " en deçà de toutes les disjonctions qu'imposent les disciplines distinctes de la géométrie et de la sociologie", fonctionne comme un système spatial et social [Cuisenier, 1991, p268]. Il identifie ainsi d'abord des divergences d'échelles d'analyse qui chacune génèrent leur vocabulaire propre, mais aussi partagent entre elles des valeurs. S'appuyant sur le formalisme de syntaxes de l'espace de Bill Hillier, il propose un modèle du bâti vu comme ordonnancement de lieux dans l'espace dont les relations sont considérées comme systèmes de contrôle habitant / étranger.
Je cite ici l'auteur; "Si tout bâtiment peut être abstraitement défini en ces termes (i.e. interfaces intérieur/extérieur par catégorisations locales), alors la variété des bâtiments effectivement construits et empiriquement observables fait transparaître un certain nombre de types comme autant de configurations définies par les paramètres qui caractérisent cette syntaxe de l'espace et par leurs interfaces. Le champ de variations est en conséquence déterminé par les valeurs que prennent ces paramètres, lesquelles dépendent à leur tour de la nature des catégories et des relations mises en ordre par l'opération de construire".
Devenu outil de systématisation de la production architecturale, comme on l'a évoqué plus haut, le vocabulaire de l'architecture est également utilisé ici par Jean Cuisenier pour déterminer l'ordre d'un graphe de composition spatiale. Il type ainsi des catégories d'habitations par recours à l'observation de régularités de syntaxes, ou règles de dépendances des espaces entre eux. Notre champ d'investigation diverge ici clairement : nous ne souhaitons faire référence qu'au vocabulaire des objets physiques, celui par lequel le bâtisseur nomme les éléments entrant en jeu dans la clôture de l'espace.
Historiquement, un vocabulaire figuré a majoritairement remplacé pour l'artisan le message écrit sur la forme architecturale de l'édifice bâti.
Précurseur d'une élaboration de types pour l'architecture, Serlio utilise un texte, qui a valeur de contrat, pour désigner des configurations et des affectations de lieux, une figure pour représenter les parties de l'édifice et leur agencement. Ce faisant, il désengage un vocabulaire écrit du champ de la construction. Il confère au texte un rôle de cadre descriptif qui fixe des critères qualitatifs et de logique spatiale pour lesquels un nouveau vocabulaire peut apparaître.
Avec l'évolution combinée des théories sur l'architecture issues de la renaissance, des techniques de représentation et d'une rationalisation de l'acte de bâtir, la figure ne laisse plus de place à l'interprétation de l'artisan. Elle est un signe non ambigu que le concepteur livre au constructeur.
Enfin, puisque chaque échelle d'analyse de l'édifice génère un vocabulaire propre, la description textuelle de l'édifice, que Serlio libère dès le seizième siècle du devoir d'en spécifier les parties physiques constitutives, peut se resserrer autour des concepts propres à un point de vue. Dès lors il devient tentant de rechercher un formalisme analytique de traitement du vocabulaire.

## *ANNEXE 4 : INVENTORISATION DU PATRIMOINE ARCHITECTURAL*

En 1974, à une époque où la pression urbaine plaçait dans une situation de péril extrême un nombre de plus en plus importants d'édifices, L'équipe de Sint-Lukasarchief a été invitée par le gouvernement local de Bruxelles à inventorier le patrimoine architectural de l'agglomération [Apers et al, 1979]. Elle l'a fait en proposant une méthode basée sur l'observation des éléments encore existants [Apers et al, 1979, page 40] dont les auteurs notent que l'étude "devra être régulièrement revue, complétée et adaptée". Les critères de sélection que l'étude prend en considération sont les suivants :

*L'immeuble comme valeur en soi*
- Aspects architecturaux formels (qualité du style, chronologie des styles, originalité d'un auteur ou d'un projet).
- Aspects architecturaux indépendants du style (modification dans le temps d'un immeuble, valeur documentaire liée à l'architecture).
- Valeur documentaire indépendante de l'architecture.

*L'immeuble dans son environnement physique*
- Relation avec son environnement physique immédiat.
- Relation avec l'environnement global.

*Critères de relativité (contexte social, historique, etc..)*

Les auteurs identifient ensuite une phase dite d'intégration qui consiste à replacer l'édifice dans un ensemble plus large, dit "ensemble de valeur", qui le fera apparaître dans son environnement [Apers et al, 1979, page 56]. Cet environnement est évalué en fonction de quatre catégories :
- Ensembles remarquables par leur valeur comme site.
- Ensembles remarquables par la présence d'éléments remarquables du point de vue architectural.
- Ensembles composés d'une succession d'éléments architecturaux.
- Ensembles remarquables par leur structure urbaine.

A travers cette étude on remarque bien sûr que l'idée de patrimoine architectural dépasse celle de l'édifice en tant qu'objet d'architecture puisqu'elle intègre par exemple les dimensions urbaines, historiques et sociales. Il apparaît donc clairement que l'édifice va fédérer un ensemble de valeurs que l'expert humain lui confère, traduites par l'expression "objet patrimonial". Si cette approche analytique est citée ici, c'est avant tout car elle définit une *perception globale* [Apers et al, 1979, page 49] de l'objet patrimonial qui interroge le travail de modélisation du corpus que nous proposons.

## ANNEXE 5 : UNE HIÉRARCHIE DE CONCEPTS ARCHITECTURAUX

Dirk Donath et Frank Petzold [Donath et al, 1997a] [Donath et al, 1997b] mènent un projet de recherche dont bien des aspects se rapprochent du nôtre puisqu'ils proposent une approche du relevé architectural intégrant d'une part une modélisation objet des édifices à relever et d'autre part une documentation historique inhérente au champ patrimonial. Ils distinguent deux grandes hiérarchies d'objets : les espaces ou vides (correspondant aux subdivisions de l'édifice, sans règle d'inclusions) et les objets bâtis (taxinomie basée sur une logique purement constructive distinguant objets noyaux, objets insérés, objets adjoints, etc..). Notre travail propose une démarche de description de l'édifice à rapprocher de celle que défendent Donath et Petzold.

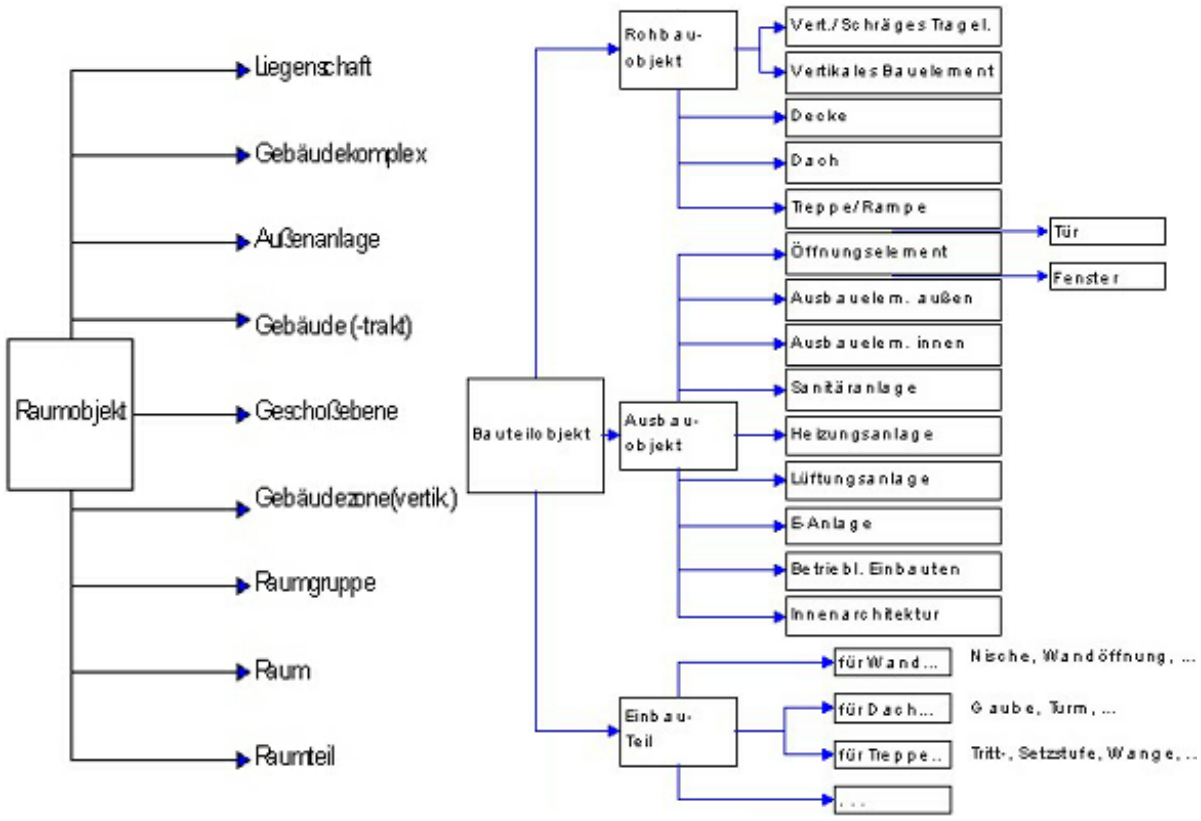



*Figure 94 : Extrait de la double hiérarchie de concepts (espaces / objets bâtis) de [Donath et al, 1997b]*

## *ANNEXE 6 : OBJETS, FRAMES*

La contribution de Xindong Wu [Wu, 1996] présente dans un premier temps ce qu'il appelle la technologie objet, puis les frames, et enfin établit quelques comparaisons entre les caractéristiques des deux. Il place au cœur de la technologie objet la classe, définie comme représentant de façon abstraite une entité du monde réel dans un package encapsulé. Il décrit comme principes bien établis en technologie objet les concepts de classe, d'encapsulation, d'héritage et de polymorphisme, sur lesquels nous reviendrons en détail plus loin. Nous ne notons ici que quelques points grâce auxquels la contribution de Xindong Wu se distingue sur le plan de la terminologie de la technologie Objet :

- Parlant du premier, l'auteur insiste sur la différence entre objet (instance) et classe en établissant que l'appel à un niveau d'abstraction qui est celui de la classe permet de distinguer "object-based programming languages" et "object-oriented programming languages"[Wu, 1996, p2].
- Parlant du second, l'auteur nomme assez singulièrement le mécanisme d'abstraction de données mis en œuvre dans la définition d'une classe par le sigle ADT (Abstract Data Types) à qui il fait recouvrir à la fois l'idée de propriétés et de méthodes encapsulées.
- Parlant d'héritage, l'auteur en définit les principes et les illustrent par l'exemple suivant : une classe "manager" peut hériter les propriétés de la classe "personne", ce qui nous fournit un exemple particulièrement clair des malentendus qui peuvent survenir dans l'utilisation de l'approche objet comme d'une technologie de programmation ou comme d'une démarche de représentation des connaissances. En effet, le terme "manager" recouvre t'il ici l'idée d'un métier, accessible à toute personne, ou de propriétés spécifiques qui ferait d'une personne une personne avec quelque chose de plus ? Voilà un problème de modélisation typique posé, nous en verrons d'autres, relatifs à notre problématique, dans la partie de ce document consacrée au projet.
- Parlant de polymorphisme, l'auteur note d'abord le manque de consensus au sein de la communauté « Objets » sur le sens que doit prendre ce terme. Il en donne comme définition préférentielle une propriété équivalente à la surcharge d'opérateurs.

Xindong Wu définit ensuite une frame comme "une structure de donnée statique utilisée pour représenter une situation" [Wu, 1996, p4]. Cette structure est définie par des relations internes reflétant la connaissance que l'on a de l'entité que représente la frame. Les relations externes, interface de l'entité, portent le nom de slots. Ceux -ci peuvent être divisés en plusieurs facettes permettant de déclencher en fonction du contexte des actions différentes. La frame est donc une "structure commune pour représenter un genre d'entités", écrit l'auteur. Il précise que ce formalisme supporte les relations d'héritage (sorte de ) et d'agrégation (partie de). Xindong Wu identifie entre Technologie Objet et frames un certain nombre de points communs :

- Entités manipulées par un identifiant.
- Entités organisées en une (des) hiérarchie (s).
- Méthodes associées aux interfaces des entités.
- Héritage simple ou multiple possible (en principe…).

Mais il les distingue en précisant les points suivants :

- L'activation de méthodes se fait en technologie objet seulement par envoi de message, contrairement au formalisme des frames qui définit une activation en fonction du contexte.
- Une frame peut être composite, ce dont l'auteur ne trouve pas d'équivalent en technologie objet.

L' encapsulation des données, et les données privées d'une entité, sont pour l'auteur une des clés de l'efficacité de la technologie Objet. Dans les frames les attributs de l'entité doivent être en libre accès pour le moteur d'inférence et par conséquent ne sont pas masquées. Enfin, l'auteur conclue en replaçant les frames comme les bases de règles dans la perspective des applications d'aide à la décision, et les oppose au formalisme objet dédié pour lui plutôt à la modélisation d'un domaine de connaissances. On voit bien entendu toute l'ambiguïté de cette opposition de fait que l'auteur constate (et qui pour lui correspond aux domaines de l'IA et du génie logiciel) : qui modélise un domaine de connaissances sans utiliser le modèle ainsi crée à fins de raisonnement et, in fine, dans un contexte de prise de décision ?

## *ANNEXE 7 : OBJETS, PATTERNS.*

Christopher Alexander a une formation initiale de mathématicien et d'architecte, et a passé une thèse de doctorat en architecture à l'université de Harvard, repris dans sa première publication de grand renom sur le processus de conception : "*notes on the synthesis of form*", publié en 1964 (Cambridge, Massachusetts, Harvard University Press). Christopher Alexander est l'auteur de nombreux ouvrages et articles qui présentent sa démarche d'analyse des lois régissant la structure de la chose bâtie. Pour Christopher Alexander, un même jeu de règles détermine la structure de la ville, de l'édifice ou de l'espace clos. Ce jeu de règles, établissant un processus de conception basé sur un raisonnement scientifique, va bien sûr à l'encontre de cette pensée sur le processus de conception qui en ferait un phénomène sous influences ( modes, politique, arbitraires des auteurs, etc…). Partant d'un constat relatif à un domaine d'application, l'architecture, Christopher Alexander développe une démarche d'analyse du processus de conception qui va au-delà du dit domaine d'application, expliquant l'importance de sa contribution.

Pour Nikos Salingaros[173], les théories de Christopher Alexander ont probablement eu plus d'impact sur le champ de l'informatique que dans le domaine de l'architecture, ce que l'on peut légitimement considérer comme un euphémisme. Nikos Salingaros indique par ailleurs que les pattern languages d'Alexander sont appliqués en programmation orientée objet, et que ce thème est aujourd'hui reconnu au sein du champ disciplinaire informatique. Nous donnons en **annexe 8** plus de détail sur les pattern languages.

Doug Lea [Lea, 1997], en introduction à son article, prend position en indiquant que, pour lui, les écrits sur l'architecture ont le mieux exploré les bases théoriques du processus de conception auquel les idées de Christopher Alexander s'appliquent avec force. Doug Lea présente d'abord les thèmes principaux émergeant des écrits d'Alexander avant d'en décrire les implications sur à la fois la démarche de modélisation Objet et le développement d'applications logicielles.

- Qualité. Alexander établit un constat d'échec: l'architecture au XXème siècle ne parvient pas à répondre à la simple exigence de produire du "mieux". Pensée anti-scientifique pour beaucoup de ses détracteurs selon Nikos Salingaros, cette prise de position s'éclaire lorsque les problèmes qu'il soulève sont détaillés:
  - Incapacité de la production architecturale à équilibrer les besoins individuels, de groupe, sociaux, et écologiques.
  - Manque d'objectif, d'ordre, d'échelle humaine.
  - Incapacité fonctionnelle et esthétique à s'adapter au contexte physique ou social.
  - Développement de produits standardisés inaptes à répondre aux exigences particulières.
  - Création d'artefacts que l'on aime pas.

  Alexander donne avec l'expression "qualité sans nom" un objectif central à tout processus de création, celui de satisfaire à une exigence de qualité concrète comme ressentie à la fois pour le concepteur et pour l'utilisateur. Doug Lea renonce à expliciter frontalement ce principe, mais on pourra se reporter à [Rising, 1996] pour une discussion sur l'interprétation de la "qualité sans nom" au développement d'applications logicielles.
- Méthode et structure. Alexander établit une opposition entre deux modes de fabrication de l'édifice. Dans la première, dite traditionnelle, l'édifice est bâti par strates historiques successives qui tendent à rapprocher l'édifice d'un état d'équilibre. Dans la seconde, l'édifice est le produit d'un processus de conception raisonné. Alexander précise cette opposition: la conception raisonnée se distingue de l'artisanat traditionnel par la séparation qu'elle marque entre le dessein et le produit, entre le génotype et le phénotype. Alexander propose une méthode dans laquelle se complètent modèles intensionnels du phénomène étudié et un jeu de contraintes extensionnelles spécifiques à chaque problème spécifique.
- Les Patterns. Doug Lea présente le formalisme des patterns comme traduisant cette approche de la conception défendue par Alexander, approche qui se veut ouverte, adaptative, plutôt qu'analytique. Alexander définit 253 patterns qui définissent un lien entre trois espaces :

[173] Voir l'introduction à la structure des *Pattern Languages* par Nikos A. Salingaros à l'adresse suivante: http://sphere.math.utsa.edu/sphere/salingar/StructurePattern.html

| Espace des problèmes (contraintes d'objectif) | Espace de construction (processus de production) | Espace des solutions (famille ou type d'artefacts) |
|---|---|---|

Chaque entrée contient cinq parties : un nom, une application prototypique, un contexte, une évaluation des contraintes (problèmes), un jeu de relations et de règles tenant lieu de solution. Au delà de cela, les patterns ont également des propriétés :

- Encapsulation (chaque pattern définit un couple problème / solution encapsulé).
- Générativité (chaque pattern définit un processus de production autonome).
- Equilibre (chaque pattern définit un espace de solutions dans lequel sont minimisés les conflits entre contraintes, la propriété d'équilibre est présentée par Doug Lea comme la recherche d'une solution optimale dont l'objectivité est difficile à établir. C'est la "qualité sans nom", qu'Alexander lui-même reconnaît évasive).
- Abstraction (les patterns sont des abstractions produites à partir d'observations et auxquelles est affecté un degré d'universalité recouvrant l'idée que les patterns peuvent n'être valides qu'au sein d'un contexte donné).
- Evolutivité (les patterns peuvent être raffinés en sous-abstractions qui différent de leur parent tout en respectant le jeu de contraintes établi en amont. Ce mécanisme de spécialisation, qui s'appuie sur l'encapsulation de la définition de chaque pattern, s'applique ici dans le sens d'un niveau de détail allant grandissant , autrement dit en échelles allant grandissant).
- Modularité (un mécanisme d'agrégation complète celui de spécialisation).

On se reportera à la bibliographie[174] de C.Alexander pour aller plus loin sur ce sujet. Il nous importe ici avant tout de voir comment peut se comprendre la contribution d'Alexander dans la perspective de la modélisation objet par classes et instances que nous utilisons. Doug Lea décrit le formalisme des patterns comme permettant d'étendre les caractéristiques de définition des classes. En POO, les classes ont en effet deux aspects proches des patterns. Le monde extérieur à la classe est pour elle l'espace des problèmes, elle est décrite par des propriétés, des responsabilités et des capacités en vu de rendre des services au monde extérieur. Le monde intérieur de la classe est son espace des solutions, qu'elle définit par un ensemble de contraintes liant ses composants, collaborateurs, etc… Chacun est connu par une vue potentiellement partielle ou incomplète sur la classe depuis le monde extérieur (espace des problèmes).
Pour Doug Lea, les patterns correspondent aux classes abstraites en terminologie objet.
Il voit en elles :

- Le moyen de relever le niveau de généricité des constructions Orientées Objet.
- Le moyen d'appliquer des notions de conception liées au formalisme des patterns aux LOO, en particulier dans l'implémentation de la notion d'héritage ou dans le principe de réutilisabilité.

Il conclue son analyse en affirmant comme Nikos Salingaros que les idées d'Alexander semblent avoir l'impact le plus important et le plus durable sur le champ des technologies Objet , et cite un ensemble d'expériences de versions Orientées Objet des patterns auxquelles je renvoie, ainsi qu'à l'encart ci-après. On se reportera également à l'article de Xavier Castellani, pour qui la notion de pattern peut être traduite par l'idée de "motifs […] qui décrivent des problèmes qui sont utilisables de nombreuses fois" [Castellani, 1998, p57]. L'auteur suggère notamment en terme d'application au contexte de la programmation objet de représenter le concept de pattern par la notation UML de collaborations

[174] Notamment C.Alexander "*A pattern language*" Oxford University Press, 1978, ISBN 0195019199 et C.Alexander "*The timeless way of building*" Oxford University Press, 1980, ISBN 0195024028

## *ANNEXE 8 : PATTERN LANGUAGES*

### *INTRODUCTION À LA STRUCTURE DES* PATTERN LANGUAGES *PAR NIKOS A. SALINGAROS*

(http://sphere.math.utsa.edu/sphere/salingar/StructurePattern.html)

Cette rapide présentation introduit le travail de Nikos A. Salingaros sur la structure des *Pattern Languages*, déjà cité.

La première phrase de son introduction explique la présence de cet encart : "*Les pattern languages nous aident à aborder la complexité d'un grand nombre de systèmes, allant d'applications logicielles à des édifices ou des villes*". Nous avons donc deux raisons de nous intéresser à ce formalisme, et d'en prendre note avant de revenir à une recherche bibliographique plus centrée sur le modèle à classes et instances.

Pour Nikos A. Salingaros, nous observons le monde autour de nous pour en comprendre la structure par recours à l'abstraction, mais aussi en documentant de solutions récurrentes obtenues dans des conditions différentes. Ces solutions prennent le nom de patterns, elles représentent des régularités de comportement. La contribution de Nikos A. Salingaros est centrée sur "*les langages qui relient les patterns*", associant aux combinaisons de nouvelles propriétés. Les patterns peuvent se voir comme une solution encapsulée, et correspondent dans cet esprit à tout module réutilisable, et bien sûr aussi en programmation Objet.

L'auteur donne trois clés pour expliciter le type de relations qui relient les patterns : les patterns peuvent contenir ou généraliser des patterns de plus petite échelle (agrégation - spécialisation), des patterns peuvent résoudre le même problème de façon différente, des patterns peuvent résoudre des problèmes différents tout en partageant la même structure, impliquant par conséquent une connection à un plus haut niveau. Il décrit les patterns comme des solutions basées sur l'observation scientifique et non sur l'invention, ce qui explique peut-être la méfiance des condisciples d'Alexander à son égard.

L'auteur identifie deux besoins auxquelles répondent les pattern languages :

- Comme moyen de comprendre et de contrôler des systèmes complexes.
- Comme outil de conception à l'aide duquel construire un tout cohérent du point de vue structurel et fonctionnel.

Les patterns sont des structures encapsulant des forces, une solution générale à un problème. Un pattern language les combinent dans un système nœuds / relations. Une combinaison cohérente de patterns génère un nouveau pattern (de plus haut niveau) contenant de nouvelles propriétés appelées émergentes car elles ne correspondent pas aux propriétés initiales détenues par les nœuds mis en relation dans la combinaison. Les pattern languages sont donc constitué d'un vocabulaire (les patterns individuels) et de règles de connection. Les connections hiérarchiques entre niveaux sont celles par lesquelles l'organisation des nœuds d'un niveau supérieur dépend de celle observée aux niveaux qui lui sont inférieurs. Une nouvelle combinaison de patterns au niveau N conduit à la définition au niveau N+1 d'un nouveau pattern, émergent, qui acquiert des propriétés indépendantes de celles de ses constituants.

Un pattern language assure cohérence et connectivité des relations entre nœuds. La cohérence interne d'un pattern pose moins de problèmes que la connectivité, comme l'illustre parfaitement l'exemple que prend l'auteur en partant du pattern qu'Alexander nomme "balcon d'un mètre quatre vingt". De nombreux patterns sociaux liés à la vie de famille peuvent exister sur un tel balcon : s'asseoir autour d'une table, faire jouer des enfants, etc.… Mais si ce balcon est pour des raisons d'économie ou de choix esthétique autoritaire ce balcon est beaucoup moins large alors il est toujours un balcon cohérent, mais un balcon dont la connectivité avec les patterns sociaux mentionnés ci-dessus est rompue. La connectivité prend ici le sens de l'inclusion de patterns appartenant à des langages différents.

Nikos A. Salingaros présente une application des pattens languages au problème des franges de ville. Il isole 12 patterns issus de la proposition d'Alexander et en évalue la validité relativement au contexte de l'étude. Je rappelle l'adresse à laquelle on pourra retrouver les publications de Nikos A. Salingaros sur l'Architecture, les patterns et l'urbanisme :

http://sphere.math.utsa.edu/sphere/salingar/contr.arch.html

## *ANNEXE 9 :VOCABULAIRE ARCHITECURAL, RELECTURE DE REFERENCES D'APRES L'APPROCHE METHODOLOGIQUE DE [PEROUSE DE MONTCLOS, 1988]*

Prenons ainsi l'ouvrage de référence écrit par Pierre Noël sur le vocabulaire attaché à la technologie de la pierre de taille, et plus généralement à la construction en maçonnerie [Noël, 1994]. On relira par exemple la définition du mur gouttereau que cet auteur donne [Noël, 1994, p191] :
*"Qui est couronné par une gouttière… s'emploie particulièrement pour désigner dans une église les murs longitudinaux de la nef qui supportent l'égout du toit"*

à la lumière de celle qu'en donne [Pérouse De Montclos, 1988, p83] :
**au chapitre mur, garde-corps et clôture :**
*"Mur : ouvrage en maçonnerie, en terre, en pan de bois, etc., dans un plan vertical"*
**au sous-chapitre Variétés de mur, de garde corps et de clôture:**
*"mur-gouttereau: Mur extérieur sous les gouttières ou les chéneaux d'un versant de toit, long-pan ou croupe…",*
définition qui renvoie vers les articles versant , long-pan et croupe.

La démarche que pose Jean-Marie Pérouse De Montclos nous pousse à ré-interroger la définition du terme donnée par Pierre Noël :
- La gouttière est-elle un signifiant du mur gouttereau ?
- Le mur gouttereau a t'il fonction de support ?
- Est-il attaché à un genre d'édifice ?

La "construction romaine" de Jean-Pierre Adam [Adam, 1989] partage avec les "principes d'analyse scientifique" de Jean-Marie Pérouse De Montclos cette particularité d'inventorier des termes qui s'appuient à la fois sur un corpus de connaissances théoriques issues d'un travail bibliographique et sur l'observation de spécimens. En effet, dans l'introduction à son ouvrage, l'auteur place au centre des sources sur lesquelles il s'appuie à la fois les dix livres d'architecture de Vitruve et la ville ensevelie de Pompéi. Il le dit en ces termes : "l'enseignement que l'on peut proposer [sur l'architecture romaine] est tout entier contenu, ou presque, dans le texte de Vitruve et dans les réalisations pompéiennes".
Jean-Pierre Adam rappelle que le texte de Vitruve nous est parvenu à travers vingt siècles d'histoire sans l'illustration des réalisations architecturales. L'auteur s'attache à rechercher dans Pompéi détruite cette illustration manquante. Sans constituer une étude du vocabulaire attaché à la construction romaine, l'ouvrage de Jean-Pierre Adam propose une logique typologique d'illustration graphique. L'auteur aborde l'ensemble des étapes de l'acte de bâtir, depuis les opérations topographiques jusqu'aux techniques du décor, en passant par ce qu'il nomme les programmes techniques (distribution d'eau, chauffage, etc..). Il propose également un lexique de modénature courante centré sur les motifs de l'ordonnancement romain. En consacrant un chapitre aux programmes et techniques de l'architecture domestique et artisanale, il rejoint brièvement les préoccupations qu'expose Jean Cuisenier sur le développement de l'architecture ordinaire. Il n'entre pas dans le cadre du présent document de détailler le travail de Jean-Pierre Adam. La "construction romaine" nous apporte néanmoins deux types d'informations qu'il était important de mettre en lumière :
- Sur les techniques de construction, l'auteur détaille l'articulation entre mise en œuvre de matériaux et objet physique élémentaire construit
- Sur la pérennité de ces techniques, l'auteur donne, en précisant les règles d'utilisation des matériaux de construction qui prévalent à l'époque romaine, un moyen de réévaluer les spécimens d'architecture du moyen âge sur lesquels aucun document écrit contemporain des œuvres ne livre de clés.

Les trois tomes du dictionnaire méthodique de l'architecture grecque et romaine [Ginouvès et al, 1985] recouvrent chacun une échelle architecturale spécifique : techniques de construction, éléments constructifs et ensembles architecturaux. Les auteurs placent d'abord leur ouvrage dans la perspective d'une utilisation dans la pratique classique de l'archéologie. Ils s'inscrivent dans une démarche de normalisation du vocabulaire de la discipline dont ils attendent une "clarification des concepts"[Ginouvès et al, 1985, tome 1, introduction, p3]. Ils placent ensuite leur travail dans la perspective d'une

informatisation des banques de données dont ils pressentent l'apport potentiel en terme de parallèles documentaires. Ils remarquent en effet que la détermination univoque d'un terme doit permettre de rassembler autour de lui un ensemble de références. S'appuyant sur le travail de Jean-Marie Pérouse De Montclos, les auteurs proposent une présentation du vocabulaire dans ce qu'ils nomment son environnement sémantique". Catégorisation plus que hiérarchisation, leur approche va plutôt ségréguer les termes qu'expliciter leurs dépendances. Ils la décrivent ainsi : " […] blocs documentaires correspondant chacun à un ensemble sémantique conventionnellement découpé, et à l'intérieur duquel l'analyse va […] du général au particulier". Les auteurs signalent que leur travail, autant sur l'architecture grecque et romaine que sur les termes plus généraux toujours en cours, s'appuie sur un ensemble de références bibliographiques desquelles ils ont tenté d'extraire des convergences. Ils posent deux questions qu'il était important de rapporter ici, puisque nous partageons avec eux le souci d'extraire un ensemble de concepts sur lesquels appuyer un travail d'analyse de l'architecture :

- Comment fixer les limites entre les concepts, autrement dit quelle variation de forme ou de fonction justifie t'elle le déplacement d'une instance d'une catégorie à une autre?
- Jusqu'où pousser la finesse de l'analyse, autrement dit quelle variation de forme ou de fonction justifie t'elle la désignation d'une nouvelle catégorie?

Ils abordent par ailleurs un point dont il nous faudra discuter dans le chapitre suivant : quels sont les liens en terme de vocabulaire (et des concepts qu'il formalise) entre les domaines de la conservation du patrimoine et de la conception architecturale aujourd'hui, dont nous avons déjà avec une référence au travail de S.Hanrot dit un mot?
Leur position sur ce point est claire, nous nous contenterai donc de les citer : " […] il ne nous parait pas souhaitable que soit utilisé, dans l'analyse des bâtiments anciens, ce vocabulaire trop particularisé, trop lié aux habitudes modernes du calibrage […] " [Ginouvès et al, 1985, tome 1, introduction, p5]. On signalera enfin qu'en proposant une traduction en trois langues de chaque terme choisi, les auteurs posent également le problème des ré-interprétations locales de vocables issus dans leur cas essentiellement du grec et du latin, problème qui sera abordé dans la partie de ce document consacrée au projet.
Les ouvrages consacrés au vocabulaire architectural sont bien sûr très nombreux, des recueils de faits dialectologiques qu'évoque Jean Cuisenier [Cuisenier, 1991, p18] aux lexiques de termes techniques orientés métiers comme celui de Pierre Noël, déjà cité.

On ne peut pas ne pas citer en particulier le dictionnaire raisonné de l'architecture de Viollet Le Duc, qui servira de référence aux acteurs de la conservation des édifices bien au-delà des frontières du pays de l'auteur. Mais il nous importait avant tout ici de faire le point sur ce que les travaux cités nous offrent comme outils de conceptualisation. On pourra, pour aller au-delà de cette présentation succincte, se reporter aux bibliographies signalées dans [Pérouse De Montclos, 1988], [Ginouvès et al, 1985] et [Cuisenier, 1991].

*ANNEXE 10 : APPROCHE OBJET, REPERES TERMINOLOGIQUES*

**Agrégation**

Une relation d'agrégation entre objets est une relation dans laquelle un objet est inclus (partie-de) dans un objet composite [Froese, 1992 p39] (voir aussi [Durnota, 1995]).

… …

… …

**Arbre d'héritage**

Il représente les hiérarchies de classes dans les systèmes à héritage simple. Cette hiérarchie de classes est constituée de nœuds (les classes) et de relations (nommées est-un). A la racine de l'arbre est généralement placée une classe particulière qui initialise la hiérarchie. Dans un système à héritage multiple l'arbre d'héritage est remplacé par un graphe d'héritage. [Blair et al, 1991]

… …

… …

**Classe**

Une classe est une abstraction décrivant des objets qui ont la même structure (les mêmes variables ou attributs) et le même comportement [Castellani, 1998]. Une classe est un patron à partir duquel peuvent être crées des objets particuliers nommés instances [Blair et al, 1991]. Une classe définit un ensemble d'opérations visibles depuis l'extérieur, un ensemble de données masquées et un ensemble de méthodes qui implémentent les opérations [Nierstrasz, 1989].

| *ADT (Abstract data type)* | [Wu, 1996] |
|---|---|
| *Templates* | [Froese, 1992 p39] |
| *Classes concrètes* | [Ducournau et al, 1998, p21] |

La classe peut dans certains Langages de programmation Orientés Objets représenter une collection par opposition aux instances qui représentent des individus. La classe reçoit alors des attributs et des méthodes qui décrivent soit la collection soit les caractéristiques des individus. [Froese, 1992 p39]

**Classe abstraite**

La distinction classe abstraite / classe concrète "*recoupe exactement la distinction aristotélicienne entre genre et espèce*" [Ducournau et al, 1998, p21]. La classe abstraite correspond à la nécessité de disposer de classes ne servant qu'à la factorisation de propriétés, mais ne pouvant engendrer d'instances.

| *Classe retardée* | [Ducournau et al, 1998, p21] |
|---|---|

Concrètement, les classes abstraites ne se distinguent pas forcément en terme de syntaxe. En C++ elles ne se distinguent des classes instanciables que par l'absence de constructeur. Le principe des *Templates* ou patrons de C++ n'est lui lié qu'au besoin de spécifier comment des classes individuelles peuvent être construites à partir d'un masque. [Stroustrup, 1996 p35, p256]

**Comportement**

Le comportement d'un objet est défini par l'exécution de ses procédures, activée par envoi de messages contenant le nom de la procédure et les arguments de l'appel [Ducournau et al, 1998]. Les méthodes représentent donc le comportement commun des objets appartenant à une classe [Masini et al, 1989].

| *Termes utilisés avec le même sens* | références |
|---|---|

Le comportement est l'ensemble des caractéristiques (propriétés) d'une classe, et se retrouve dans ses sous-classes. [Blair et al, 1991]

**Constructeur**

Le constructeur est une méthode de la classe appelée sur cette classe pour déclencher la procédure d'instanciation d'un objet : il initialise une instance de la classe [Wu, 1996, p3].

L'instance initialisée par un constructeur est détruite en C++ par une autre opération appelée destructeur. [Stroustrup, 1996, p30]. En JAVA, le mécanisme de garbage collection s'en charge [Flanagan, 1997, p60].

| | | |
|---|---|---|
| | *Procédure d'initialisation* | [Ducournau et al, 1998, p15] |
| | … | … |
| **Encapsulation** | L'encapsulation est le processus de compartimentation des éléments d'une abstraction qui constituent sa structure et son comportement ; l'encapsulation sert à séparer l'interface contractuelle d'une abstraction de son implémentation [Menzies et al, 1995][175]. L'encapsulation est un mécanisme permettant de regrouper dans une même entité des données et des opérations qui s'appliquent à ces données [Masini et al, 1989]. L'encapsulation renforce l'idée d'objet en énonçant que les données d'un objet ne peuvent pas être atteintes directement mais seulement à travers les opérations (l'interface) de l'objet [Froese, 1992, p37]. | |
| | *Abstraction de données* | [Gaillard, 1994] |
| | … | … |
| **Etat** | L'état d'un objet est l'ensemble des valeurs de ces attributs (champs, variables d'instances, etc..) à un moment donné [Blair et al, 1991]. L'ensemble des valeurs des champs évolue : l'état d'un objet est donc lié à un instant précis [Ducournau et al, 1998, p12]. | |
| | *Etat courant* | [Ducournau et al, 1998, p12] |
| | *Structure* | [Menzies et al, 1995] |
| | … | … |
| **Héritage** | L'héritage est un mécanisme permettant le partage et la réutilisation de propriétés entre les objets. La relation d'héritage est une relation de généralisation / spécialisation qui organise les objets en structure hiérarchique [Masini et al ,1989]. L'héritage est l'incorporation du comportement (données et opérations) d'une classe dans une autre [Blair et al ,1991]. Autrement dit, une classe A hérite d'une classe B si l'ensemble des caractéristiques de la classe B est inclus dans l'ensemble des caractéristiques de la classe A [Castellani, 1998].<br>Les relations d'héritage définissent une(des) hiérarchie(s) de classes qui forment un arbre d'héritage (héritage simple) ou un graphe d'héritage orienté (héritage multiple)[Froese, 1992]. | |
| | *Héritage de spécialisation* | [Castellani, 1998] |
| | *Héritage de classe* | [Nierstrasz, 1989] |
| | *Catégorisation* | [Froese, 1992] |
| | *Dérivation* | [Wu, 1996], |
| | *Extension* | [Ducournau et al, 1998] |
| | Héritage de construction (voir ci-dessous) | [Castellani, 1998] |
| **Héritage de construction** | Les héritages de construction servent à définir des liens d'héritage dont le seul but est de réutiliser les caractéristiques d'une classe dans une autre classe. Un héritage de construction est un héritage tel que les instances de la classe qui hérite ne sont pas du type défini par la classe héritée [Castellani, 1998]. | |
| | *Héritage d'implémentation* | [Castellani, 1998][176] |
| | Héritage de spécialisation (voir ci-dessus) | [Castellani, 1998] |
| **Instances** | Les instances sont les objets particuliers créés à partir d'une classe telle que définie plus haut. Chaque instance d'une classe [d'objets] a son propre jeu de variables d'instance et partage les méthodes associées aux opérations avec les autres instances de sa classe [Nierstrasz, 1989]. Les instances sont des objets pour lesquels les classes agissent comme des patrons de construction [Froese, 1992, p39]. Des objets construits à partir d'une classe sont appelés instances de la classe, chacune a son | |

[175] Définition donnée par les auteurs en référence à G.Booch "Object oriented design with applications" seconde édition, Benjamin /cummings 1994.

[176] Définition donnée par l'auteur en référence à C.Delobel, C.Lecluse, P. Richard "Bases de données : des systèmes relationnels aux systèmes à objets", InterEditions, 1991.

| | | |
|---|---|---|
| | propre état [Blair et al , 1991].<br>Une instance d'une classe est un objet constitué des champs dont la liste est donnée par la classe et dont le comportement est défini par l'interprétation des messages [Ducournau et al, 1998, p14].<br>*Variables*<br>*Objets* | <br><br>[Wu,1996], [Wu et al,1995]<br>[Blair et al, 1991] |
| | L'instance peut dans certains Langages de programmation Orientés Objets représenter un individu par opposition aux classes qui représentent des collections. | [Froese, 1992, p39] |
| **Instanciation** | L'instanciation est le mécanisme qui permet à partir d'une classe de produire des objets particuliers, les instances. L'instanciation est réalisée en appelant une méthode d'une classe (voir constructeur) qui crée et initialise un nouvel objet basé sur la définition de la classe [Blair et al, 1991].<br>*Construction* | <br>[Stroustrup, 1996, p30] |
| | … | … |
| **Liaison** | La liaison est le mécanisme permettant d'associer un sélecteur (nom de méthode) à la méthode à appliquer lors d'un envoi de message [Masini et al, 1989]. En général, la méthode la plus spécialisée (la première méthode trouvée dans les ascendants d'une classe) est choisie. Les liaisons ont lieu statiquement (à la création du programme) ou dynamiquement (à l'exécution) [Blair et al, 1991] (voir également [Ducournau, 1997] et [Ducournau et al, 1998]).<br>*lien* | <br>[Ducournau et al, 1998] |
| | … | … |
| **Message (envoi de)** | Un message est une requête adressée à un objet demandant l'exécution d'une opération sur l'objet. Il comprend généralement un objet destinataire, un sélecteur de méthode et des arguments [Masini et al, 1989]. Le choix du corps de la méthode à exécuter est disjoint du nom de la méthode tel qu'il apparaît dans le message envoyé à l'objet. Le message fournit un nom de méthode à exécuter: le corps de la méthode correspondant est recherché dans le catalogue de méthodes de la classe puis dans ceux de ses super classes (recherche ascendante ou look-up) [Ducournau et al , 1998, pages 14 à 19].<br>*Invocation de méthodes* | <br>[Blair et al, 1991] |
| | Autres utilisations de ce terme | références |
| **Métaclasse** | Une métaclasse a pour instances des classes particulières comme la classe a pour instances des objets particuliers. Elle correspond au souhait de disposer d'outils de création de classes qui permettent de manipuler les classes [Ducournau et al , 1998, p28] [Masini et al, 1989, section 2.5].<br>*Meta-model object* | <br>[Maughan et al, 1998] |
| | … | … |
| **Méthodes** | Les méthodes forment le jeu d'opérations autorisées à être exécutées dans le contexte d'un objet [Blair et al, 1991]. Les sélecteurs (noms) des méthodes forment l'interface visible depuis l'extérieur de l'objet (opérations), l'implémentation des méthodes en est disjointe et encapsulée par l'objet [Nierstrasz, 1989]. Les méthodes d'un objet sont seules aptes à manipuler les données (variables d'instances) formant l'état de l'objet [Wu et al, 1995].<br>*Procédures*<br>*Routines*<br>*Operations*<br>*Fonctions, fonctions membres*<br>*Service* | <br>[Masini et al , 1989]<br>[Ducournau et al , 1998]<br>[Durnota, 1995]<br>[Stroustrup, 1996]<br>[Castellani, 1998] |
| | … | … |

**Objet**

La définition d'un objet comme structure encapsulant des données et des procédures [Masini et al, 1989] est commune aux langages basés objets et aux langages orientés objet (modèle à classes et instances) [Tsichritzis et al, 1992]. Un système basé sur la notion d'objet est un système dans lequel un problème est représenté par un ensemble d'entités qui interagissent pour produire l'effet recherché [Blair et al, 1991].

| | |
|---|---|
| *Entités* | [Blair et al, 1991] |
| *Variables ou instances* | [Wu et al ,1995] |
| Objets atomiques ou objets composites. | [Wu et al ,1995] |
| Catégorie d'objets (Objets sémantiques / de construction /de servitude, …) | [Castellani, 1998] |

**Opérations ou interface**

L'interface d'un objet est l'ensemble des opérations applicables à un objet et connues du monde extérieur [Masini et al, 1989]. L'interface est dite publique par opposition à l'implémentation des méthodes auxquelles l'interface donne accès, dite privée [Froese, 1992].

| | |
|---|---|
| *Interface contractuel* | [Menzies et al, 1995] |
| Le mot Opérations peut être utilisé comme synonyme de méthode. | [Durnota, 1995] |

**Polymorphisme**

Le polymorphisme est la capacité donnée à deux objets différents de pouvoir associer à un même nom des procédures différentes. Une méthode polymorphe peut avoir plusieurs implémentations différentes, auquel cas le type du résultat de cette méthode dépend de la classe dans laquelle elle est implémentée [Ducournau et al , 1998, pages 13, 386].

Le polymorphisme peut concrètement correspondre à la redéfinition des méthodes d'une classe dans ses sous-classes ou à l'utilisation d'un même nom de méthode dans des parties indépendantes d'une hiérarchie de classes [Blair et al , 1991].

| | |
|---|---|
| *Surcharge* | [Ducournau et al , 1998] |
| *Polymorphisme ad hoc* | [Masini et al, 1989] |
| Exprimé différemment, le polymorphisme est la capacité donnée à un objet de réagir dynamiquement en fonction de sa forme courante (polymorphisme d'héritage). | [Snyder, 1998]<br>[Masini et al, 1989]<br>[Wu, 1996] |
| Le mot polymorphisme est aussi utilisé pour décrire la capacité de certains langages à autoriser la définition dans une classe de plusieurs méthodes portant le même nom. | |

**Propriétés**

Les propriétés structurelles (données) et comportementales (méthodes) d'un objet sont définies par sa classe [Durnota, 1995]. Les propriétés individuelles des objets (instances d'une classe) sont les données définissant ces objets et qui n'ont pas la même valeur pour tous ces objets, et les méthodes correspondantes. Les propriétés communes des instances d'une classe sont les données définissant ces instances et qui ont pas la même valeur pour toutes ces instances, et les méthodes correspondantes. Les propriétés collectives des instances d'une classe sont les données définissant des collections d'instances de la classe ou qui réfèrent une seule instance de la classe, objets, et les méthodes correspondantes [Castellani, 1998].

| | |
|---|---|
| *Caractéristiques* | [Castellani, 1998] |
| *Comportement* | [Blair et al , 1991] |
| … | … |

**Redéfinition (de méthodes)**

La redéfinition de méthodes correspond à l'idée qu'une classe peut fournir une version plus spécialisée d'un service décrit dans un de ses ascendants [Ducournau et al , 1998, p92], et plus largement à l'association de corps de procédures redéfinis par des objets receveurs d'un même message. La redéfinition de méthodes est aussi souvent appelée surcharge lorsque comprise dans le sens d'opérations de même nom appliquées à des arguments de type différents en maintenant le même comportement [Masini et al, 1989] [Nierstrasz, 1989]. Elle peut aussi être appelée surcharge simple dans le cas d'opérations dont seul le nom coïncide [Nierstrasz, 1989].

| | | |
|---|---|---|
| | *Surcharge* | [Nierstrasz, 1989] |
| | Redéfinition (d'attributs) | [Ducournau et al , 1998] |
| | Le mot surcharge de méthodes correspond à l'utilisation d'un même nom de méthode dans des parties indépendantes d'une hiérarchie de classes. | [Blair et al , 1991] |
| **Sélecteur** | Le sélecteur est le nom du message ou de la méthode qui est cherché dans une classe ou dans un ses ascendants pour déclencher l'application d'une méthode [Ducournau, 1997]. | |
| | … | … |
| | … | … |
| **Super-classe** et **Sous-classe** | Une classe héritant d'une autre classe hérite des variables et des méthodes de cette classe et ajoute les siennes. La nouvelle classe est dite sous-classe de l'ancienne et l'ancienne super- classe de la nouvelle [Blair et al , 1991]. | |
| | *Classe de base* (pour Super-classe)<br>*Classe dérivée* (pour Sous-classe) | [Stroustrup, 1996] |
| | … | … |
| **Typage** | Les variables ne sont pas strictement typées dans un langage à objets puisqu'une même variable peut désigner des instances de plusieurs classes différentes au cours de sa durée de vie [Masini et al, 1989]. On distingue typage statique (fixé, validité vérifiée à la compilation) ou dynamique (évoluant durant l'exécution du programme, validité vérifiée à l'exécution) [Nierstrasz, 1989] (voir aussi [Ducournau et al , 1998, p25]). | |
| | … | … |
| | … | … |
| **Variables (d'instance)** | L'état d'un objet est décrit par l'ensemble des valeurs prises par ses champs. Les termes variables d'instances ou attributs sont souvent préférés au mot champ [Ducournau et al, 1998, p12]. Les variables d'instances sont encapsulées par l'objet [Nierstrasz, 1989]. | |
| | *Attributs, Champs , Données membres, Primitives, Slots* | [Ducournau et al, 1998] |
| | *Données, Champs* | [Flanagan, 1997] |
| | *Item de données* | [Wu et al, 1995] |
| | *Membres* | [Stroustrup, 1996] |
| | Les Variables (de classe, statiques) sont partagées par toutes les instances d'une classe et de ses sous-classes (sens Smalltalk).<br>Les Variables (de classe) sont définies dans une métaclasse et détenues par les classes (sens ObjVLisp). | [Flanagan, 1997]<br>[Masini et al, 1989] |

*ANNEXE 11 : GRILLE D'ANALYSE DÉTAILLÉE JAVA*

| | |
|---|---|
| Encapsulation des variables | Par défaut, les variables d'instance sans qualification peuvent être accédées uniquement depuis les classes définies dans le même package, c'est le niveau d'accès dit "friendly". Les variables d'instance déclarées Public peuvent être accédées par tout client sans aucun contrôle. Déclarées protected, elles sont en fait accessibles depuis les sous-classes. Le niveau d'accès le plus restrictif est celui que définit le mot private : la variable n'est accessible que depuis la classe elle-même. Une sous-classe hérite de ces variables mais ne pourra y accéder que par le biais de méthodes protected ou public dans la super-classe. |
| Encapsulation des méthodes | Les spécificateurs d'accès des variables s'appliquent de la même façon aux méthodes. Les méthodes peuvent être qualifiées public, private ou protected. Les méthodes private ne peuvent être invoquées que par les seules méthodes de cette classe. |
| Déclaration de variables | En Java, toute variable utilisée dans un programme est nommée, typée et éventuellement initialisée. Cette dernière opération se fait par l'opérateur "=". La valeur d'une variable sera modifiée par une affectation, mais elle peut être qualifiée par exemple par le mot clé final, qui précise une valeur non modifiable pour la variable. Le qualifieur final indique au compilateur que la variable gardera tout au long de son existence une valeur constante. Le compilateur produira donc une erreur lorsqu'il y aura une tentative de modification de la valeur de ce champ.<br>Le type d'une variable est soit un type primitif soit une référence. Dans le premier cas, elle contient toujours une valeur de son type. Dans le second, elle contient soit une référence null soit une référence vers un objet dont le type est compatible avec le type de la variable. Les déclarations de variables peuvent figurer n'importe où dans le corps d'une méthode. Cependant, toute définition de variables figure forcément à l'intérieur d'une déclaration de classe puisque le concept de variables globales n'existe pas en JAVA. |
| Déclaration de méthodes | Les méthodes sont définies par un nom et des paramètres formels (ou des parenthèses vides si la méthode ne prend pas de paramètres). Le nombre de ces paramètres est fixe. Les seuls types possibles de paramètres sont les types primitifs et les références.<br>Elles sont également définies par un type du retour (void si la méthode ne retourne aucune valeur) et le corps de la méthode. En effet, contrairement au langage C++, la définition effective des méthodes de la classe doit se faire dans la déclaration de la classe.<br>Lors des appels aux méthodes, tous les paramètres sont passés par valeur. Le concept de passage par adresse n'existe pas. La valeur d'une variable de type primitif dans le code appelant ne peut donc pas être modifiée.<br>Dans le cas de références à des objets, ces références sont également passées par valeur. C'est la valeur de la référence et pas l'objet lui-même qui est passé, une méthode peut modifier une copie de la valeur de la référence sans que cela modifie la valeur de la référence du code appelant. Contrairement aux langage C , un même identificateur peut être utilisé par désigner deux méthodes à condition que leur signature (donnée de son nom, du nombre de ses paramètres formels et de leur type) soit différente. |
| Allocation - désallocation | Une définition de variable alloue la place mémoire nécessaire en fonction du type précisé. Mais contrairement au langage C++, la restitution de l'espace mémoire consommée est gérée par un système de récupération de mémoire |

| | |
|---|---|
| | automatique appelé garbage collector. Ce mécanisme restitue au système l'espace occupé par un objet quand il n'y a plus aucune référence vers cet objet.<br>Par défaut, le récupérateur de mémoire fonctionne en arrière plan pendant l'exécution d'un programme Java, mais la récupération de mémoire peut être invoquée explicitement par le programmeur.<br>La récupération d'un objet par la machine virtuelle Java s'accompagne d'un appel à la méthode finalize de l'objet qui permet au programmeur de libérer des ressources systèmes par exemple puisque le garbage collector ne s'occupe que des objets JAVA. |
| Constructeurs | Un constructeur porte en Java le même nom que la classe et n'a pas de valeur de retour. Les objets sont instanciés par le mot clé new passé à une classe. On peut spécifier plusieurs constructeurs en modifiant le nombre de ses paramètres. Pour tout objet crée, le constructeur de la superclasse est invoqué (enchaînement d'invocation de constructeurs). Le premier constructeur invoqué est celui de la classe Object (racine de la hiérarchie de classes Java) suivi des autres constructeurs dans l'ordre de la hiérarchie des classes.<br>En effet, si le constructeur d'une classe n'invoque pas explicitement le constructeur de sa superclasse (instruction super), Java fait quand même appel au constructeur par défaut super() (sans arguments) de la superclasse.<br>Si aucun constructeur n'est défini dans une classe, ce même mécanisme de constructeur sans argument est utilisé : le constructeur par défaut se contente d'invoquer le constructeur par défaut de la superclasse.<br>Un constructeur peut par ailleurs invoquer explicitement un constructeur de sa superclasse. L'invocation de super (avec des arguments relatifs à la superclasse) doit être la première instruction du constructeur de la sous-classe. L'appel au constructeur de la classe de base est suivi de l'exécution du corps du constructeur de la classe dérivée. |
| Destructeurs | Avant la récupération d'un objet, la machine virtuelle Java fait appel à la méthode finalize de l'objet. Celle-ci doit invoquer explicitement le destructeur de la superclasse. En effet, contrairement aux constructeurs, les destructeurs ne sont pas invoqués en chaîne. Il appartient au programmeur de mettre en œuvre cette chaîne à l'aide du mot clé super. |
| Valeurs de retour (méthodes) | Le type de retour d'une méthode est soit void (si la méthode ne retourne aucune valeur), soit un type primitif ou une référence vers un objet. |
| Type des variables | Le package java.lang définit un ensemble de classes et interfaces qui constitue le noyau du langage Java. JAVA est un langage fortement typé qui dispose de types primitifs et, par le biais de ce package, de classes définissant des objets pouvant contenir un type primitif et auxquels sont associées des méthodes permettant de les manipuler. Les types primitifs ne peuvent être transformés en objets mais ces classes permettent d'effectuer des opérations de conversion. |
| Agrégations | Au-delà du formalisme d'agrégation d'objets, Java autorise l'imbrication de classes sous plusieurs formes. Deux types de classes ou d'interfaces statiques (appelées classes imbriquées statiques) sont accessibles pour toutes les classes d'un package et se distinguent des classes "normales" par un nom spécifiant la classe qui la contient. Trois types de classes imbriquées (appelées classes intérieures) ne sont visibles qu'à l'intérieur de la classe qui les contient. Ce mécanisme peut être un confort dans la gestion d'évènements par exemple mais pose en terme de modélisation un problème de lisibilité et de duplication évident. |

| | |
|---|---|
| Persistance | Le mécanisme de sérialisation d'objets que propose Java depuis sa version 1.1 permet de donner une représentation binaire des instances en cours dans un projet soit pour faire communiquer des objets dans une application réseau soit pour archiver des objets sur un support non volatile. Le concepteur de la classe est responsable de l'écriture ou non des objets au travers de l'implémentation des méthodes adéquates de l'interface Serializable (WriteObject et ReadObject). Ce mécanisme ne suffit pourtant pas à rendre Java persistent (voir au chapitre précédent les exigences liées à la persistance des objets). Pour [Kappel et al, 1998], trois principes fondamentaux doivent être pris en compte : persistance orthogonale (objets, classes, méthodes doivent être persistants), persistance transitive (objets contenus et persistance implicite) et enfin persistante transparente (écriture du code identique que l'objet soit persistant ou non).<br>Ces auteurs rappellent que la recherche d'un formalisme de persistance pour Java se fait à l'heure actuelle dans deux directions : d'une part par une extension de la machine virtuelle JAVA et d'autre part au travers du pré-traitement des fichiers source Java ou du post-traitement des fichiers class. Cette dernière solution réduit la possibilité de persistance transparente, mais facilite une architecture d'application distribuée.<br>Je renvoie à leur contribution pour plus de détail sur la gestion d'objets persistants dans le cadre d'applications réseau. |
| Déclaration de classe | En Java la déclaration d'une classe spécifie ses propriétés et doit inclure la définition des méthodes. Les méthodes ont pour argument implicite la référence this, référence à l'objet courant. |
| Variables de classe | Une variable déclarée static est partagée par l'ensemble des objets d'une classe.<br>La déclaration d'une variable static est faite à l'intérieur de la classe, la syntaxe d'accès à une telle variable s'écrit en précisant le nom de la classe et non celui de l'instance. Cet appel explicite à une variable par le nom de sa classe évite les conflits de noms entre variables globales. Le mot clé final peut être employé pour définir une variable de classe de valeur constante. |
| Méthodes de classe | Comme les variables de classe, les méthodes de classe sont déclarées avec le mot clé static, et invoquées par le nom de leur classe pour éviter les conflits de nom. L'argument implicite this passé à une méthode (d'instance) n'est pas passé ici à une méthode de classe. De plus, les méthodes de classe n'ont accès ni aux variables d'instance ni aux méthodes d'instances. Elles sont par exemple les seules utilisées dans la classe Math puisque celle-ci regroupe un ensemble de méthodes relatives à des types prédéfinis (donc pas à des objets avec des variables d'instance). |
| Classes abstraites | Une classe Java peut être déclarée abstraite, qu'elle contienne ou non des méthodes abstraites. Elle ne peut dès lors instancier d'objets particuliers. Une classe contenant une méthode abstraite doit être déclarée abstraite. Une classe dont la superclassse est abstraite ne peut instancier d'objets particuliers que si elle redéfinit toutes les méthodes abstraites de la superclasse et leur fournit une implémentation. Dans le cas contraire, la sous-classe est elle aussi abstraite (Voir aussi les Interfaces au thème héritage). |
| Méthodes virtuelles | La méthode abstraite est l'équivalent Java de la méthode virtuelle pure de C++. Elle n'a pas d'implémentation mais une signature, ses règles d'usage sont explicités au point précédent. |
| Typage et généricité | JAVA ne fournit pas de mécanismes de généricité tels que les Template de C++ qui autorisent la définition d'une classe dont les instances définiront le typage. |

| | |
|---|---|
| Portée | Le texte constituant la définition complète d'une classe est regroupé dans un même fichier qui prend comme extension .java. Les classes sont par défaut accessibles depuis n'importe quelle classe du même package. Le mot clé public permet de qualifier la classe de façon à la rendre accessible partout. La notion de package permet de résoudre les conflits des noms des classes et des interfaces dans une application Java. En fait, chaque package peut contenir des classes, des interfaces et d'autres packages. Chaque fichier source précise le package auquel il appartient, et peut contenir un ensemble de directives import portant sur des fichiers sources individuels ou sur des packages. Ces directives permettent au compilateur, par le biais de la variable d'environnement CLASSPATH, et compte tenu de l'organisation hiérarchique des fichiers et des packages, de n'importer que les classes ou les packages utiles. |

| | |
|---|---|
| Héritage simple / multiple | Java a choisi de ne fournir qu'un héritage simple repéré par le mot clé extends. Autrement dit, une classe java ne peut hériter de l'implémentation de méthodes que d'une seule superclasse. Java propose en revanche un formalisme d'interfaces qui s'apparente à de l'héritage multiple puisqu'une classe va hériter d'une seule superclasse mais implémenter optionnellement plusieurs interfaces. La différence tient au fait que les interfaces contiennent des méthodes forcément abstraites pour lesquels la classe qui implémentent ces interfaces devra fournir un corps de méthodes. Les interfaces peuvent eux-mêmes être organisés en une hiérarchie d'interfaces, les variables qu'elles contiennent doivent être static et final (c'est à dire des constantes).<br>Java propose donc une alternative élégante (ou en tout cas jugée comme telle par les auteurs qui défendent ce langage) à l'héritage multiple. Il n'en reste pas moins que l'utilisation des Interfaces n'est pas équivalente à celle de l'héritage multiple: une distinction de fait est établie entre la classe qui implémente cet interface, classe pouvant produire des objets singuliers substituts aux objets du monde réel, placée dans une hiérarchie de classes potentiellement instanciables, et l'interface lui-même qui agit comme un moule de comportement qui relève essentiellement d'une démarche de génie logiciel.<br>Dernier point à citer ici, une classe ne peut pas hériter d'une superclasse déclarée final. |
| Polymorphisme d'héritage (Redéfinition de méthodes) | En Java, une méthode redéfinie dans une sous-classe doit fournir la même signature que la méthode héritée de la super-classe. A l'intérieur d'une sous-classe, on accède à une méthode d'une superclasse redéfinie dans cette sous-classe en utilisant le mot clé super. Une méthode déclarée final dans une superclasse ne peut être redéfinie dans une de ses sous-classes.<br>Java utilise le mot "method overriding" pour qualifier la redéfinition de méthodes et le mot "method overloading" pour qualifier la surcharge de méthodes (voir point *Surcharge d'opérateurs et de méthodes* ci-dessous). |
| Polymorphisme d'interface (Redéfinition de méthodes) | Chaque méthode étant repérée (voir portée) sans ambiguité, les mêmes noms de méthodes peuvent être utilisés pour désigner des méthodes de classes non reliées par une lien hiérarchique. |
| Surcharge d'opérateurs et de méthodes | Java distingue la redéfinition de méthode telle qu'explicitée ci-dessus et la surcharge de méthode qui consiste à donner un même nom à des méthodes dont les listes d'arguments diffèrent. Pour qu'une méthode en surcharge une autre il faut soit que le type des arguments diffère, soit que leur nombre diffère, soit encore qu'ils soient placés dans un ordre différent. Spécifier un type différent pour la valeur retournée par la méthode ne suffit pas.<br>Il n'y a pas de surcharges d'opérateurs en Java. |

| | |
|---|---|
| Typage | Les types en Java sont divisés en deux groupes: types primitifs (int, float, etc…) et références (classes, interfaces, tableau). Il existe en Java un grand nombre de mécanismes de conversion de types entre types primitifs ou entre objets[177]. Par contre un certain nombre de conversions sont interdites, notamment:<br>– d'un objet vers un type primitif,<br>– d'un type primitif vers un objet (sauf String),<br>– d'un type de classe vers un autre type de classe non reliée dans la hiérarchie.<br>Une conversion s'effectue par l'opérateur de conversion (cast) dans la limite de la validité de celle-ci. |
| Gestionnaire d'exceptions | Java se distingue de C++ notamment par le mécanisme de traitements des exceptions ou des erreurs qu'il fournit.<br>L'utilisation des exceptions est un appui pour :<br>– Séparer l'écriture du comportement "normal" d'un programme et la gestion des erreurs.<br>– Propager de proche en proche les exceptions d'une méthode à la méthode appelante jusqu'à atteindre une méthode capable de gérer l'exception. La gestion de l'exception n'a pas à être prise en charge dans la méthode qui la déclenche.<br>– Regrouper par catégories les exceptions pour factoriser leur gestion.<br>Les exceptions ou erreurs Java sont des objets instances de la classe java.lang.throwable. Les classes Error et Exception sont des sous classes de cette classe java.lang.throwable. La définition de nouvelles classes héritant des classes Exceptions et Erreurs permet de définir de nouveaux types d'exceptions. Les méthodes pouvant lever une ou plusieurs exceptions le déclarent dans leur entête. Les exceptions levées par une méthode sont capturées par la méthode appelante (encapsulation de l'appel de la méthode dans un bloc try - catch). La fin du bloc try contient toute la gestion des exceptions spécifiée par des clauses catch (en nombre indéfini). Les instructions du corps du bloc try sont exécutées jusqu'à la fin de ce bloc si aucune exception n'est levée. Dans le cas contraire, le contrôle est transféré à l'une des clauses catch. Si aucune clause catch n'est prévue pour traiter cette exception, le contrôle est transféré à la dernière des méthodes appelantes englobées dans un bloc try et ainsi de suite. Si une clause finally existe à la fin d'un bloc try, le contrôle est passé à cette portion du code. La clause finally est utilisée lorsque l'état d'un système doit être mise à jour quelque soit la manière dont les instructions se sont exécutées. Il n'est pas nécessaire de spécifier par le mot clé throws placé dans l'entête de la méthode les exceptions non capturées (Error et RuntimeException). |

[177] 42 conversions autorisées entre types primitifs, 16 entre objets (voir la littérature consacrée à JAVA).

*ANNEXE 12 : ANALYSE COMPARATIVE DU LANGAGE C++*

| | |
|---|---|
| Encapsulation des variables | En C++ une variable d'instance peut être déclarée private, protected ou public. Une variable qualifiée private ne peut être utilisée que par les méthodes de sa classe et par les méthodes déclarées amies de cette classe. Une variable qualifiée protected ne peut être utilisée que par les méthodes de sa classe, celles déclarées amies de cette classe, et dans ces mêmes conditions par chaque classe dérivée. Une variable qualifiée public peut être utilisée par n'importe quelle méthode. |
| Encapsulation des méthodes | Outre les règles d'accès définies ci-dessus, C++ dispose du mot clé friend pour permettre à une méthode d'accéder aux variables d'une classe extérieure. |
| Déclaration de variables | Déclaration et définition des variables peuvent être effectuées dans la même instruction qui spécifie alors le type de la variable, son nom et sa valeur. La déclaration se fait à l'intérieur de la déclaration de la classe. La règle d'accès par défaut est private. |
| Déclaration de méthodes | Une méthode (dite "fonction membre" en C++) est déclarée avec spécification de son profil et de la valeur retournée. Sa définition (son implémentation) peut être donnée à l'intérieur de la classe ou à l'extérieur de la classe en utilisant l'opérateur de résolution de portée "::". |
| Allocation - désallocation | L'opérateur new permet d'allouer une zone mémoire que l'opérateur delete libère. Il n'y a pas de mécanisme de ramasse-miettes automatisé. |
| Constructeurs | Les constructeurs sont des méthodes qui prennent le nom de la classe et une liste d'arguments pour réaliser l'initialisation de l'objet. Quand une classe a un constructeur, il est appelé à chaque création d'un objet de cette classe. Un objet peut être crée comme un objet automatique (créé chaque fois qu'une déclaration est rencontrée dans l'exécution du programme), statique (crée au lancement du programme), dynamique (par l'opérateur new), membre (crée comme membre d'une autre classe). |
| Destructeurs | Quand la classe a un destructeur, il est appelé à chaque destruction de l'objet. Les destructeurs ne sont pas hérités, ils ne peuvent renvoyer de valeur. |
| Valeurs de retour (méthodes) | Une méthode dont le type de retour n'est pas explicitement déclaré void doit renvoyer une valeur. |
| Type des variables | Les variables sont typées statiquement en C++. La classe est appelée type utilisateur par opposition aux types prédéfinis. Ceux-ci incluent les nombres, les caractères, et le type void. Des types dérivés peuvent être obtenus en se servant des opérandes pointeur, référence et tableau. |
| Agrégations | Les constructeurs d'objets membres (par agrégation) d'une classe sont appelés dans l'ordre dans lequel ils sont déclarés dans la classe. L'objet contenant est détruit avant ses objets contenus, détruits dans l'ordre inverse de leur déclaration dans la classe. |
| Persistance | L'implémentation de la persistance des objets en C++ est laissée aux soins du programmeur. On peut se reporter pour des exemples concrets à [Conway, 1997] ou [Rosenblatt, 1995]. |

| | |
|---|---|
| Déclaration de classe | En C++ la déclaration d'une classe spécifie ses membres mais peut ne pas inclure la définition des méthodes. C++ n'impose pas de contraintes sur où elle doit être écrite (fichiers isolés) pour autant que l'utilisation des noms et des types soit cohérente. La portée d'édition (interne ou externe au fichier définis) d'un objet ou d'une fonction peut être spécifiée explicitement. |
| Variables de classe | Une variable déclarée static est partagée par l'ensemble des objets d'une classe. Déclaration et définition d'une variable déclarée static sont séparées. |
| Méthodes de classe | Une méthode d'une classe peut également être déclarée static. Elle n'a pas accès aux variables non statiques de sa classe autrement que par son interface si celui-ci le permet explicitement. |
| Classes abstraites | Une classe contenant une ou plusieurs méthodes virtuelles pure est dite classe abstraite, elle ne peut instancier d'objets particuliers. Elle ne peut que servir de superclassse à d'autres classes. Une méthode virtuelle pure (c'est à dire dont la signature est suivie d'une initialisation à zéro) doit être redéfinie dans les sous-classes. |
| Méthodes virtuelles | Une méthode virtuelle est définie dans la classe de base et redéfinie dans chaque sous-classe, elle décharge le programmeur du choix d'une méthode adéquate à l'objet spécfique. |
| Typage et généricité | C++ autorise une certaine forme de généricité par le biais des patrons (templates) de classes. Un patron de classe spécifie comment des classes individuelles peuvent être construites de la même manière qu'une déclaration de classe spécifie comment des objets particuliers peuvent l'être. L'argument présent dans la spécification du patron de classe est un type qui ne sera défini qu'à la déclaration d'une classe particulière basée sur le patron de classe. |
| Portée | Le texte constituant la définition complète d'une classe n'est pas nécessairement regroupé dans un seul et même fichier. La déclaration d'une classe est habituellement placée dans un fichier d'inclusion. C++ reprend du langage C la<br>directive include qui permet d'inclure cette déclaration partout ou elle est nécessaire. Les mêmes noms pouvant être utilisés pour désigner des méthodes ou des variables différentes, l'opérateur de résolution de portée "::" permet de spécifier la classe à laquelle on fait référence. |
| Héritage simple / multiple | L'héritage est appelé en C++ dérivation. Une dérivation est qualifiée par un mot désignant les règles d'accès aux membres de la classe de base par la classe dérivée (private, protected ou public). Une classe C++ peut dériver de plusieurs superclasses: l'ordre d'énumération des superclasses détermine l'ordre d'appel des constructeurs des superclasses, appelés avant celui de la classe dérivée. Les conflits d'héritage sont réglés en faisant explicitement appel à l'opérateur de résolution de portée pour spécifier quelle origine doit être donnée à une propriété héritée dont le nom est commun dans plusieurs superclasses. Dans une structure d'héritage en diamant la superclasse peut être déclarée virtuelle dans les deux classes dérivées qui en hérite. Dans une classe héritant de ces deux classes, les propriétés de la superclasse ne seront pas dupliquées. Enfin, les règles d'accès aux propriétés ne peuvent en C++ être modifiée par des sous-classes que dans le sens d'un renforcement de la protection des accès. |

| | |
|---|---|
| Polymorphisme d'héritage (Redéfinition de méthodes) | Une méthode redéfinie dans une sous-classe doit fournir la même signature que la méthode héritée de la super-classe. Si ce n'est pas le cas, alors il y a surcharge de la méthode et non redéfinition : la nouvelle méthode est dite masquant la méthode héritée. Des méthodes de même nom peuvent avoir des signatures et des règles d'accès différentes à l'intérieur d'une classe. |
| Polymorphisme d'interface (Redéfinition de méthodes) | Les mêmes noms pouvant être utilisés pour désigner des méthodes ou des variables différentes, l'opérateur de résolution de portée "::" permet de spécifier la classe à laquelle on fait référence. |
| Surcharge d'opérateurs | La plupart des opérateurs peuvent être redéfinis dans le contexte d'une classe C++. Une classe peut également être explicitement pourvue d'opérateurs de conversion sous forme de méthodes. |
| Typage | C++ est un langage fortement typé, l'utilisateur peut définir des conversions de type appliquées explicitement ou implicitement par le compilateur. |
| Gestionnaire d'exceptions | En C++ la création de mécanisme de gestion d'exceptions (try-catch similaire à celui de JAVA) est laissée aux soins du programmeur. |

*ANNEXE 13 : ANALYSE COMPARATIVE DU LANGAGE PERL*

| | |
|---|---|
| Encapsulation des variables | L'encapsulation est le concept de base des Langages de Programmation Orientés Objets pour lequel Perl fournit la moins bonne réponse. En fait, tout le travail de protection des variables d'une classe est laissé aux soins du programmeur. Perl autorise l'accès au contenu de l'objet, mais aussi autorise de l'interdire… Dans la pratique, ce point ne pose pas de problème majeur tant que l'application reste de petite échelle. Plusieurs techniques de programmation de l'encapsulation sont proposées dans [Conway, 2000, pp 296 - 326]. |
| Encapsulation des méthodes | De la même façon, une méthode peut être cachée en dehors du contexte de l'objet seulement si le programmeur le prévoit explicitement. |
| Déclaration de variables | Les variables d'instance de l'objet sont déclarées (et initialisées éventuellement) dans le constructeur de la classe qui prend pour nom new et pour forme une méthode normale. |
| Déclaration de méthodes | Les méthodes (pour l'accession aux variables et de façon générale) sont déclarées et définies généralement à l'intérieur de la déclaration de la classe, mais peuvent l'être à l'extérieur si elle est qualifiée en conséquence. |
| Allocation -désallocation | Perl implémente un ramasse-miettes automatique (pas de mécanisme de désallocation de mémoire manuel) qui fait appel aux méthodes DESTROY des objets si celles-ci sont prévues dans l'objet pour effectuer une tâche particulière. |
| Constructeurs | Un constructeur Perl est une méthode généralement nommée new qui contient la définition des variables de l'objet et une instruction bless qui les attache à la classe.<br>Un constructeur peut prendre un nombre libre d'arguments destinés à réaliser l'initialisation de l'objet. |
| Destructeurs | La méthode DESTROY sera invoquée (si elle est définie dans la classe) à la destruction d'un objet par le ramasse-miettes automatique. |
| Valeurs de retour (méthodes) | Les méthodes d'une classe peuvent retourner une valeur par l'instruction return. Qu'elle le fasse ou non, Perl n'autorise pas la spécification du type de la valeur retournée. |
| Type des variables | Perl est un langage à typage des variables dynamique : les variables peuvent contenir n'importe quel objet (ou type), n'importe quel argument peut être passé à une méthode (avec des conséquences évidentes sur la sûreté à l'exécution). Les types standards Perl (nombres, chaînes de caractères, références) peuvent être manipulés sous forme de scalaires, de tableaux, de tableaux associatifs. |
| Agrégations | L'agrégation d'objets est réalisée par le constructeur de l'objet contenant, qui spécifie l'ordre d'appel des constructeurs des objets contenus. |
| Persistance | Perl permet d'accéder à l'identité des objets et dispose d'un ensemble de modules de réplication vers des SGBD commerciaux ou non comme vers des systèmes de fichiers. On peut se reporter à [Conway, 2000, pp 387-428] qui propose un ensemble de solutions pour gérer la persistance des objets en Perl. |

| | |
|---|---|
| Déclaration de classe | En Perl la déclaration d'une classe spécifie ses membres dans le constructeur mais peut ne pas inclure toutes ses méthodes. |
| Variables de classe | Les variables de classe sont déclarées dans la classe à côté de leurs accesseurs. Elles ne sont généralement pas accessibles directement encore que cela soit laissé aux soins du programmeur. |
| Méthodes de classe | Une méthode de classe est une méthode dont le premier argument est le nom de la classe. Les méthodes de classe gèrent l'accès aux variables de classe suivant les règles d'accès définies par le programmeur. |
| Classes abstraites | Une classe abstraite peut être considérée comme telle si le programmeur implémente une ou plusieurs méthodes abstraites définissant le comportement de l'objet en cas d'exceptions (instanciation d'objet ou méthode non redéfinie). Il n'existe donc pas de moyen natif de définir une classe abstraite en Perl, mais une pseudo classe abstraite peut être construite. |
| Méthodes virtuelles | Une méthode virtuelle (abstraite pour Perl) peut être définie mais l'est sans syntaxe particulière. Elle signifie généralement par une levée d'exception explicite son caractère de méthode abstraite. |
| Typage et généricité | La notion de classes génériques a moins d'importance dans un langage à typage dynamique comme Perl. Plusieurs techniques de programmation permettant néanmoins de décrire des classes génériques sont néanmoins rapportées dans [Conway, 2000, pp 329-336]. |
| Portée | La définition d'une classe en Perl est faite à l'intérieur d'un Package identifiant son contenu (variables et méthodes). Des méthodes d'une classe peuvent être écrites en dehors du contexte du Package, bien que cela ne favorise pas la lisibilité de la classe. Un Package est défini dans un fichier module (.pm) à appeler partout où la classe est utilisée. |
| Héritage simple / multiple | Une classe Perl déclare dans une variable nommée @ISA la ou les classes dont elle hérite. L'héritage en Perl se traduit par la recherche ascendante d'une méthode appelée sur une classe dans ses superclasses. L'héritage en Perl doit en fait être compris seulement comme une facilité pour rechercher une méthode. Les variables ne sont héritées qu'à la demande du programmeur. Les constructeurs sont hérités mais seul le premier trouvé est exécuté en cas d'absence de constructeur dans une classe instanciant un objet. Enfin, en cas d'héritage en diamant, la résolution du problème de double héritage des méthodes est laissée aux soins du programmeur. |
| Polymorphisme d'héritage (Redéfinition de méthodes) | Il n'y a pas de contraintes de syntaxe ou de signature des méthodes en Perl pour les rendre polymorphiques : elles le sont par défaut. Le polymorphisme d'héritage ajoute simplement au polymorphisme d'interface une contrainte d'appartenance à une classe donnée sur l'objet invoquant la méthode. |
| Polymorphisme d'interface (Redéfinition de méthodes) | La redéfinition de méthodes dans un ensemble de classes non reliées par une relation d'héritage n'impose qu'une contrainte: que cette méthode fasse partie de l'interface de chacune de ces classes. Perl ne prévoit pas de contrôle de type sur la signature des méthodes. |
| Surcharge d'opérateurs | Un module spécifique de la distribution standard de Perl (overload.pm) |

| | |
|---|---|
| | donne accès aux mécanismes de surcharges d'opérateurs de Perl. L'appel à ce module donne la possibilité de surcharger dans le contexte de la classe les opérateurs standards ou de spécifier des conversions de type utilisateur. |
| Typage | Perl est un langage à typage dynamique, l'utilisateur peut définir des conversions de type dans les conditions que nous venons de définir. |
| Gestionnaire d'exceptions | Ici encore, la gestion d'exception est du ressort du programmeur. En Perl cette question est rendue plus sensible par le caractère pseudo-interprété du langage. |

## *ANNEXE 14 : PRINCIPES D'UNE MÉTHODE D'ANALYSE ET DE CONCEPTION PAR OBJETS, OMT*

Les phases d'analyse et de conception ont pour objectif de produire un ensemble de modèles représentant différentes vues du système à réaliser [Ducournau et al, 1998, p106-107] :

- La vue structurelle décrit l'ensemble des entités en jeu dans le système, elle est représentée par un modèle appelé modèle à objets.
- La vue comportementale décrit les modifications que subit le système, elle est représentée par un modèle appelé modèle dynamique.
- La vue architecturale décompose le système en sous-systèmes pour en faciliter la compréhension, elle est représentée par un modèle appelé modèle fonctionnel.

Ces modèles sont élaborés en utilisant des notations, constituées de diagrammes et de formulaires, qui ont pour vocation de faciliter la lisibilité des modèles pour chaque intervenant.

### *Modèle à objets*

La notation des modèles à objets propose deux diagrammes, le diagramme de classes et le diagramme d'instances. Le premier décrit les variables et méthodes définies dans les classes ainsi que leurs relations (spécialisation, agrégation, association). Le second décrit l'utilisation des instances en montrant comment elles sont reliées.

OMT utilise le concept d'association pour enrichir le modèle objet [Ducournau et al, 1998, p107]. Une classe est représentée par un rectangle composé de trois parties : son nom, ses variables et ses méthodes. Une instance est représentée par un rectangle aux coins arrondis comportant le nom de la classe et de l'instance et la liste des attributs valués. Les relations entre les classes peuvent être de trois types: association, agrégation et spécialisation. L'agrégation est une relation partie / tout ou une relation de composition, l'association traduit un lien sémantique entre deux classes [Ducournau et al, 1998, p109]. Les relations peuvent être enrichies des notions de clé (prise par un attribut dans la relation), de contrainte (conditionnant la relation) et de cardinalité.

OMT propose le concept de module pour regrouper des classes matérialisant un point de vue sur une situation[178]. Le modèle à objets peut être découpé en plusieurs modules à l'intérieur desquels noms des classes et des associations doivent être uniques. Une même classe peut cependant être référencée dans plusieurs modules. OMT propose également le concept de contrainte servant à restreindre les valeurs que peuvent prendre les entités du modèle (objets, classes, variables, relations).

Pour [Ducournau et al, 1998, p105], le succès et la pérennité des méthodes d'analyse et de conception par objets repose en grande partie sur la présence d'outils de génération automatique de code permettant de traduire dans un Langage de Programmation Orienté Objets les modèles élaborés. Le développement des extensions d'OMT présentes dans la littérature se heurte pour les auteurs à cette difficulté.

Une discussion approfondie sur les liens entre différentes méthodes d'analyse et de conception par objets et les Langages de Programmation Orientés Objets les plus courants est proposée par Sjaak Brinkkemper, Shuguang Hong, Arjan Bulthuis et Geert van den Go dans leur article en ligne "Object-Oriented Analysis and Design : a Comparative Review"[179].

### *Vue comportementale*

Le modèle dynamique en OMT est constitué de diagrammes d'états et de diagrammes de trace d'évènements [Ducournau et al, 1998, p112] :

- Les diagrammes d'état montrent les changements d'états des objets d'une certaine classe occasionnés par des évènements. La réunion de ces diagrammes traduit l'activité globale du système. Dans ce formalisme, l'état correspond aux valeurs de variables et de relations d'un objet à un instant donné, la transition à un changement d'état occasionné par un événement qui peut être interne ou externe au système. Les diagrammes d'état s'appliquent bien à la description d'un modèle global du

[178] Comparables aux catégories de OOD et aux clusters de BON

[179] Table des matières à l'adresse http://wwwis.cs.utwente.nl:8080/dmrg/OODOC/oodoc/oo.html
Méthode OMT et Langages de Programmation Orientés Objets à l'adresse: http://wwwis.cs.utwente.nl:8080/dmrg/OODOC/oodoc/oo-19.2.3.html
Voir également " Building an OMT-Editor Using Design Patterns: An Experience Report"
Bart Wydaeghe, Kurt Verschaeve, Bart Michiels, Bruno Van Damme, Evert Arckens, and Viviane Joncker
http://info.vub.ac.be/~bwydaegh/tekst/papers/tools98/html/tools98.html

comportement des classes, ils s'appliquent en revanche moins aisément à la description des changements d'état des instances durant le déroulement d'algorithmes.

- Les diagrammes de trace d'évènements sont utilisés pour décrire comment les instances des classes interagissent, c'est à dire pour tracer l'exécution d'une séquence d'évènements qui se réalise entre des objets dans un contexte particulier du système. On peut se reporter à "Object-Oriented Analysis and Design : a Comparative Review", déjà cité, pour une description de modèles pour la vue comportementale dans d'autres méthodes d'analyse et de conception par objets.

*Modèle fonctionnel*

Le modèle fonctionnel décrit l'activité au sein du système à l'aide de diagrammes de flux de données (Data Flow Diagrams). Les diagrammes de flux de données comportent des nœuds, (opérations du système, acteurs ou réservoirs d'informations) et des arcs (flux de données ou de contrôle).

*Processus d'une méthode d'analyse et de conception par objets*

Le processus d'une méthode d'analyse et de conception par objets décrit les actions nécessaires à l'obtention des différents modèles sous forme de diagrammes, formulaires et textes, ainsi que l'enchaînement de ces activités [Ducournau et al, 1998, p118]. La mise en oeuvre de ce processus repose en OMT comme en UML sur le programmeur puisque si des outils existent pour l'obtention des modèles, il n'en existe en revanche pas pour guider le programmeur dans cette tâche. En OMT trois phases sont identifiées:

| | | | | |
|---|---|---|---|---|
| ANALYSE<br>Modèle à objets<br>Modèle dynamique<br>Modèle fonctionnel | > | CONCEPTION<br><br>Conception du système<br>Conception des objets | > | IMPLEMENTATION |

- La phase d'analyse sert à déterminer un ensemble de modèles (à objets, dynamique, fonctionnel) pour décrire le système. En OMT, une liste de tâches et de recommandations est donnée pour construire les modèles, comme par exemple éliminer les classes vagues[180].
- La phase de conception se décompose en conception du système et conception des objets. En prenant en compte les contraintes d'implémentation, les modèles issus de la phase d'analyse sont améliorés ou modifiés. La **conception du système** fait référence à des problèmes relatifs à la modularité de l'application à développer. Il s'agit d'organiser le système en sous-systèmes en abordant les questions de stockage des données ou de concurrence. La **conception des objets** fait référence à la définition des classes d'objets eu égard au modèle à objets issu de la phase d'analyse, mais aussi aux problèmes résultant du choix d'un langage de programmation pour l'implémentation. Le diagramme de classes peut nécessiter des modifications si le langage choisi ne gère pas, par exemple, l'héritage multiple. En OMT les activités liées à cette étape font l'objet de recommandations d'ordre général[181]. A partir des modèles issus de la phase d'analyse et amendés dans la phase de conception, certains outils peuvent générer le code correspondant.
- La phase d'implémentation a pour finalité de produire du code à partir des spécifications établies aux phases précédentes. Deux types de réponses sont à rapporter: les recommandations fournies pour telle ou telle méthode, dont OMT bien sûr, et vers tel ou tel langage de programmation orienté objets; et par ailleurs les outils de génération de code. Pour OMT[182], on trouvera des explications sur la définition des classes, la création d'objets, etc… dans son ouvrage de référence. De la même façon, on peut se reporter à [Ducournau et al ,1998, pp120-128] pour des références concernant l'implémentation des modèles dans le cadre des SGBD Objets ou relationnels. En ce qui concerne les

[180] exemples de tâches proposées pour construire le modèle à objets : identifier objets et classes, identifier les relations, identifier les attributs des objets et des relations, organiser et simplifier les classes en utilisant l'héritage, améliorer le modèle par itérations successives, regrouper les classes dans des modules, etc…

[181] exemples de recommandations proposées dans la phase de conception : définir de nouvelles classes et de nouvelles opérations si nécessaires, mettre les opérations dans les bonnes classes, etc…

[182] Voir " Object-Oriented Modeling and Design" , Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., , Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1991.

outils de génération de code (CASE - Computer Assisted Software Engineering - tools), la traduction du modèle à objets se limite à la génération de squelettes de définitions de classes. Le modèle dynamique n'étant pas pris en compte dans la génération automatique de code, il existe une différence importante entre la représentation définie par le biais de la méthode d'analyse et de conception par objets et sa traduction. On peut se reporter pour une discussion approfondie sur ce sujet à [Oussalah, 1997, pp182-203].

Le processus d'une méthode d'analyse et de conception par objets tel que défini ici n'intègre pas la phase de définition des besoins. Une méthode d'analyse et de conception par objets ne devient pourtant performante que si elle s'appuie sur l'établissement préalable de spécifications rigoureuses pour le système considéré [Ducournau et al ,1998, p122]. Pour [Oussalah, 1997, p59], la phase d'analyse d'un problème est souvent la moins bien traitée quelles que soient les méthodes utilisées. Cette remarque établit de toute évidence un constat de manque.

*ANNEXE 15 : TECHNIQUES DE RELEVE*

**Historique**

Dans son ouvrage, déjà cité, Jean-Pierre Adam [Adam, 1989] donne de la topographie dans l'architecture romaine une description détaillée. Nous n'en citons ici que quelques points de repères. L'auteur rappelle d'abord les trois opérations définissant la topographie: établissement des directions, mesure des distance et estimation des hauteurs. L'auteur décrit dans le détail deux instruments plus utilisé dans les travaux d'arpentage que de relevé proprement dit: la groma et le chorobate. La groma est dans le monde romain l'instrument autorisant les deux opérations de base du travail de l'arpenteur: l'alignement et la détermination de perpendiculaires. Cet instrument comporte deus axes de visées en équerre pivotant sur un bras que porte un dispositif d'ancrage au sol. Ces visés orthogonales étaient complétées de mesures de distances et de diagonales pour établir par exemple ces centuriations dont on retrouve encore les traces sur le paysage de nombreuses régions. Une opération de relevé topographique utilisant la groma consistait à tracer au sol une base (ligne) jalonnée et à déplacer le long de cette base la groma en la mettant en stations successives pour relever d'une part l'alignement à la base et d'autre part, perpendiculairement, des points dont on chaîne la distance sur l'objet à relever (opération dite de levé des coordonnées). Instrument utilisé en complément de la groma, le chorobate était une sorte d'équivalent romain de notre niveau actuel. Vitruve en donne une description sur laquelle Jean-Pierre Adam s'est appuyé pour reconstituer cet instrument afin d'expérimenter son utilisation. Le chorobate, destiné aux travaux de nivellement, comprend en partie haute un canal d'eau, et à ses extrémités deux fils à plomb qui devaient coïncider en station horizontale avec des marques placées en partie basse [Adam, 1989, p18]. Deux œilletons placés sur la partie supérieure de l'instrument permettent d'effectuer les visées horizontales nécessaires notamment lors de relevé d'objets ou de travaux d'arpentage sur des terrains en pente.

Jean-Paul Saint Aubin situe la naissance d'une théorie du relevé au XVème siècle. Il décrit deux *modes de collecte des données* complémentaires, représentés aujourd'hui par d'un côté les techniques photogrammétriques et de l'autre les techniques de relevé topographique. Ces deux modes ont, écrit l'auteur, en commun d'être fondé sur le postulat d'un rayon visuel rectiligne liant l'observateur et l'observé, et sur le principe géométrique de l'intersection des visées. Mais dans le premier mode, qui s'inscrit dans la filiation du miroir de Brunelleschi, le relevé enregistre une trace en continu de l'objet : perspectives ou photographies fournissent une représentation en plan de l'objet, utilisée comme telle ou réinterprétée en photogrammétrie par exemple. Dans le second mode, qui s'inscrit dans la filiation du plan de Rome d'Alberti, le relevé enregistre un ensemble de points sur l'objet, intersections de visées convergentes [StAubin, 1992, p18-31].

Un relevé topographique fixe deux types de données: des angles et des distances. La mesure des angles est bien sûr plus commode. Dès Alberti, la mesure des distances ne se fait que de manière indirecte, en utilisant les propriétés mathématiques des figures géométriques dessinées par l'intersection des visées. Les instruments du relevé topographiques n'ont depuis lors évolué qu'en précision: lunettes grossissantes ou règles à pinnules autorisant un meilleur pointé, verniers assurant un meilleur repérage angulaire. Seule l'apparition récente des distancemètres (déterminant une distance par l'évaluation du temps mis par un signal pour atteindre le point visé et revenir à sa source) apporte réellement une modification importante aux principes du relevé topographique. Jean-Paul Saint Aubin insiste sur un aspect important: le relevé topographique est un choix de points observés sur l'objet, choix fait par le releveur lorsqu'il considère ces points utiles au relevé. Aucune modification a posteriori du choix de ces points n'est possible sans retour sur le terrain, distinguant avec évidence cette technique de la mesure photogrammétrique où le support photographique permet une réinterrogation des données.

La photogrammétrie s'appuie sur l'intersection de rayons issus d'un point observé sur l'édifice depuis deux stations. L'utilisation de deux perspectives centrales comme source de l'intersection est, écrit Jean-Paul Saint Aubin, à rapprocher des principes de la chambre obscure de Léonard De Vinci. Laussedat théorise au XIXéme siècle un procédé qu'il nomme métrophotographie, reprenant des principes déjà utilisés à partir de perspectives dessinées, mais les appliquant à l'utilisation de la photographie. C'est sous le nom de photogrammétrie, inventé par Meydenbauer, que cette technique nous est aujourd'hui connue. Au début du XXème siècle, les premiers restituteurs permettent de tirer parti des photos stéréoscopiques connues depuis un demi siècle. Ces appareils reproduisent mécaniquement l'intersection de rayons

homologues, permettant de mesurer dans l'espace de l'objet la position d'un repère. Pourtant, la restitution analogique, historiquement utilisée en priorité dans le domaine de la cartographie, n'est traditionnellement pas utilisée pour tirer parti d'une mesure dans l'espace puisque les résultats produits sont des représentations en plan. Les restituteurs analytiques, successeurs informatiques des restituteurs analogiques (mécaniques), s'appuient sur les mêmes principes. La photogrammétrie a, dans le domaine de l'architecture, été utilisée depuis Meydenbauer essentiellement dans la production de documents plans (façades, tissus urbains). La photogrammétrie architecturale a donc une tradition assez longue et des usages clairement définis, sur lesquels l'ouvrage de Jean-Paul Saint Aubin pourra renseigner en détail. Depuis une vingtaine d'années, avec le développement des plateformes de modélisation tridimensionnelle, plusieurs propositions venant de la recherche ou de l'industrie tentent de rendre leur troisième dimension aux représentations fournies par le relevé photogrammétrique. Nous revenons sur ce point dans la section consacrée à la photogrammétrie architecturale.

**Techniques**

Nous proposons ici brièvement un point sur les principes généraux des trois techniques aujourd'hui les plus utilisées dans la mesure des édifices: relevés topographique, photogrammétrique, par balayage laser. Le relevé architectural, on l'a évoqué plus haut, n'a pas pour seule vocation de donner des dimensions à l'objet observé. Il fixe également au travers d'une analyse de l'objet mesuré, un ensemble de données qualitatives. Nous reviendrons sur ce point ultérieurement. Nous souhaitons évoquer rapidement une quatrième technique, utilisée aujourd'hui majoritairement dans le cadre d'études archéologiques du sol: les GPR (Ground Penetrating Radar). En effet, depuis une dizaine d'années, plusieurs expériences ont été menées pour rechercher sans fouilles du sol les traces d'édifices disparu. Nous renvoyons notamment à [Gao, 1997] pour qui cette technique non invasive permet de vérifier la présence sous le niveau du sol actuel d'anomalies correspondant par exemple aux fondations d'un édifice. On remarque qu'ici aussi le relevé s'appuie sur une analyse a priori de l'objet recherché. L'auteur détaille deux expériences menées sur des édifices disparus en chine, pour lesquelles l'utilisation d'un GPR a permis d'établir la position et la profondeur des vestiges. Il replace cette technique en perspective en indiquant qu'elle ne peut être utilisée comme méthode d'investigation de départ. En effet, Les GPR produisent des profils du sol comparés les uns aux autres et interprétés par lecture de leurs différences. L'archéologue doit par conséquent dresser un schéma de relevé du site définissant une grille (lignes suivant lesquelles le GPR sera déplacé) et une profondeur. Cette technique nous intéresse ici car elle bénéficie d'un avantage considérable sur les trois dont nous discuterons ci-après: elle détermine ce qui ne se voit pas. Topographie, Photogrammétrie et balayage laser se contentent d'enregistrer des données sur la surface de l'objet mesuré, laissant à l'analyste la responsabilité d'attacher une réalité matérielle, une réalité construite, à cette surface. Le travail de déduction du mode constructif d'un édifice revient alors entièrement à la charge de l'archéologue qui va si c'est possible opérer par carottages et dans le cas contraire faire appel à un savoir théorique. L'utilisation des GPR pour observer par exemple le corps d'une maçonnerie médiévale pourrait, après les premières applications au champ de l'archéologie, donner lieu à expérimentation.

**Le relevé topographique**

Les points que le topographe relève dans l'espace sont attachés à un plan de référence, et font l'objet de mesures d'angles et de distances. Les instruments topographiques repèrent un plan horizontal de référence, et une verticale de station. A grande échelle, ces instruments sont en priorité les niveaux de topographes, les théodolites, les distancemètres et les équerres optiques d'alignement (uniquement pour l'implantation d'angles droits). L'horizontalité peut être vérifiée localement (niveaux classiques) ou par visées sur des mires graduées permettant de lire les différences de niveau entre le plan horizontal de visée et les points d'appui des mires (niveaux de topographe). La mesure des angles se fait en référence à un plan (horizontal ou vertical) à l'aide de théodolites.

La mesure directe des distances a depuis longtemps posé beaucoup plus de problèmes. En effet, mesurer une distance s'est fait, jusqu'à l'apparition des distancemètres, par report le long d'une horizontale (ou d'une pente donnée) d'une mesure étalon. Ce type de mesure, appelée généralement "chaînage", doit en outre respecter l'alignement entre les deux points dont on évalue l'éloignement. Jean-Paul Saint Aubin donne pour évaluer la précision de ce type de mesure la formule suivante: (1 à 2 cm) * $\sqrt{n}$ où n est le nombre de reports de l'étalon [StAubin, 1992, p52]. La précision d'une mesure de distance par report

d'étalon est donc liée au nombre de report de cet étalon, elle ne s'utilise aujourd'hui pratiquement plus qu'en relevés sommaires d'intérieurs. Le distancemètre permet d'évaluer la distance entre l'instrument et un réflecteur posé sur un objet par comptage de fréquences de modulation d'ondes de lumière laser. Distancemètres et théodolites sont souvent couplés, permettant de relever à la fois distance et mesure d'angle. L'inconvénient majeur de ces dispositifs reste la nécessité de placer sur l'objet à mesurer des réflecteurs, point dont on imagine bien les conséquences dans le relevé architectural d'édifices de grandes dimensions.
En élévation, une lecture directe de distance peut être obtenue par lecture sur un ruban vertical ou sur une mire, avec le même inconvénient que précédemment. Une distance verticale se mesure de façon plus précise indirectement: la différence de niveau entre deux points A et B est évaluée par étude des propriétés géométriques du triangle liant ces deux points au point de station d'un théodolite S (AB= SA*tangente angleASB). Ce procédé est d'autant moins précis que l'angle de visée se rapproche de la verticale, la tangente se rapprochant alors de l'infini.
Un relevé architectural topographique s'appuie sur un canevas de plan (polygonation) défini avant la campagne de mesures. Ce canevas détermine un ensemble de polygones dont les sommets sont formés par des points choisis sur l'édifice et par les stations des instruments topographiques. Dans le relevé d'architecture, ce canevas créé un réseau de lignes qui entourent l'édifice mais aussi qui recalent intérieur et extérieur. Apparaît ici une première difficulté majeure pour le releveur: un édifice à relever est rarement assez dégagé de son environnement immédiat pour autoriser de le contourner de façon régulière. Relevés planimétriques et altimétriques des détails d'architecture appellent d'autres procédures de travail qui reprennent les principes déjà évoqués: rayonnement, trilatération, etc.. On retrouve une description détaillée des outils et méthodes du topographe dans l'ouvrage de Jean-Paul Saint Aubin, déjà cité [StAubin, 1992,].
Le relevé topographique de l'architecture construite pose trois problèmes importants. Nous l'avons vu, l'accessibilité de l'édifice joue un grand rôle dans la précision et l'exhaustivité du relevé. Le relevé topographique enregistre la position dans l'espace d'un ensemble de points dont seul le croquis de canevas rappelle le sens architectural. Il s'utilise dans une vision en plans de références de l'édifice: plans d'étage, façades, implantations. Enfin, le relevé topographique enregistre l'ensemble de points que précise le canevas de plan (polygonation), et cet ensemble seul. Toute ré-interrogation de l'objet (par exemple sur tel ou tel ordonnancement de façade) nécessite par conséquent une nouvelle campagne de mesure s'appuyant sur un nouveau canevas de plan. Le relevé topographique est dans la pratique du métier d'architecte ou de conservateur le plus répandu pour répondre à des besoins ponctuels. Rapidité de mise en œuvre, relative simplicité des dispositifs matériels utilisés et précision sont ses principaux avantages. Sa portée étant fixée a priori, le relevé topographique apparaît comme facteur d'économie. Dans le cadre d'étude patrimoniales, cela reste à démontrer puisque les deux techniques que nous allons maintenant aborder ont sur lui un avantage décisif: elles permettent d'enregistrer des données globales sur l'édifice, ré-interrogeables sans recours à de nouvelles campagnes de mesure. Les services de l'inventaire français ont par exemple pendant de longues années misées sur une collecte globale de clichés photographiquement qui seraient aujourd'hui encore susceptibles d'être réutilisés.

**Le relevé par balayage laser**

La télémétrie Laser (scanners 3D) est une technologie récente qui permet de relever un grille dense de points mesurés à la surface d'un objet. Cette technologie a été développée pour des environnements industriels où existaient un besoin de modélisation 3D a posteriori d'installations ne correspondant plus aux plans originaux. On comprend dès lors qu'EDF ait participé au développement du capteur SOISIC dont nous parlerons plus en détail. Un nuage de points observés à la surface de l'objet est obtenu par un balayage dont la densité est définie par l'utilisateur. Ce nuage est alors traité sur une plateforme logicielle spécifique pour le mailler ou pour retrouver telle ou telle primitive géométrique. Plusieurs types de capteurs existent, dont les différences tiennent essentiellement aux objectifs d'utilisation. Leur domaine de prédilection reste, lorsque utilisé dans le patrimonial, la sculpture, où les rapports de dimensions entre capteur et objet observé sont les plus favorables.

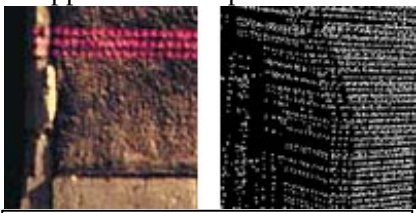


*Figure 95 : Balayage Laser, maillage 3cm in http://www.archeodunum.ch*

Certains dispositifs permettent par rotation du capteur autour d'un bras articulé de saisir l'objet sous plusieurs angles sans déplacement autour de l'objet.

Les capteurs proposés par la firme Cyberware par exemple incluent en outre une caméra couleur haute résolution. Ils permettent ainsi de fixer non seulement la position des points mais aussi leur valeur de couleur. Ces capteurs sont construits "à la demande" en fonction notamment de l'éloignement à l'objet à observer. Une expérience très intéressante de cette technologie, menée par une équipe américano-italienne sur l'œuvre sculptée de Michel-ange, est rapportée dans [Levoy, 1999].

*Figure 96: Numérisation avec le capteur SOISIC, in http://www.archeodunum.ch*

Dans ce gros projet, un capteur laser cylindrique sur trépied fabriqué par Cyberware[183] a permis d'effectuer un ensemble de relevés sur les parties accessibles des sculptures (par tranches de 3mètres de large et de 7 mètres 50 de haut), les parties moins accessibles étant mesurée par le biais de systèmes plus légers montés sur bras articulé (fabriqués par Faro Technologies[184]). Les nuages de points capturés ont été dans cette expérience réassemblés a posteriori par alignement semi-automatisé des scans et maillés sur une plateforme développée à l'université de Stanford. Ce travail démontre l'adaptation de ce type de système au relevé à petite échelle, mais qu'en est-il lorsqu'un édifice doit être appréhendé dans sa globalité? Nous proposons ici une présentation du capteur SOISIC développé par MENSI[185] et EDF, et qui a fait l'objet de plusieurs expérimentations dans le champ de l'architecture. Ce capteur est constitué d'un émetteur laser et d'un récepteur (caméra CCD). L'émetteur envoie un faisceau laser de faible puissance suivant un pas précisé par l'opérateur. Ce faisceau, réfléchi par un miroir mobile, balaye la surface de l'objet observé. Le faisceau est réfléchi par l'objet jusqu'au récepteur, une caméra CCD. C'est donc encore par triangulation (la distance émetteur/récepteur étant connue) qu'une évaluation de la position du point relevé est possible. Bien entendu, la base du triangle étant fonction de la dimension du capteur, la précision de mesure varie largement avec la distance à l'objet relevé. La précision de mesure du capteur SOISIC est de l'ordre du millimètre à cinq mètres de distance. Son utilisation dans le relevé d'installations industrielles pose ici peu de problèmes. Par contre, dans son application à l'architecture, on se heurte à la difficulté de saisir dans le champ de vision de l'appareil l'ensemble de l'édifice (rapport 3/4 entre largeur vue et distance à l'objet). Ceci implique naturellement d'effectuer plusieurs balayage à mettre en correspondance a posteriori. Cette mise en correspondance peut se faire à petite échelle en recalant deux scans par l'observation sur chacun d'eux des coordonnées de cinq sphères témoins positionnées dans la scène, ou à plus grande échelle par recalages topographiques. Nous l'avons vu, les capteurs lasers fournissent un nuage de points « anonymes », image télémétrique non interprétative de l'objet dans l'espace. Ce nuage de points doit donc au minimum être traduit aux formats des outils de CAO puis maillé pour être exploité. Dans le cas du capteur SOISIC, un logiciel nommé 3DIPSOS est fourni en complément du hardware. Ce logiciel permet bien sûr de visualiser les nuages de points et de les manipuler, mais il permet également de

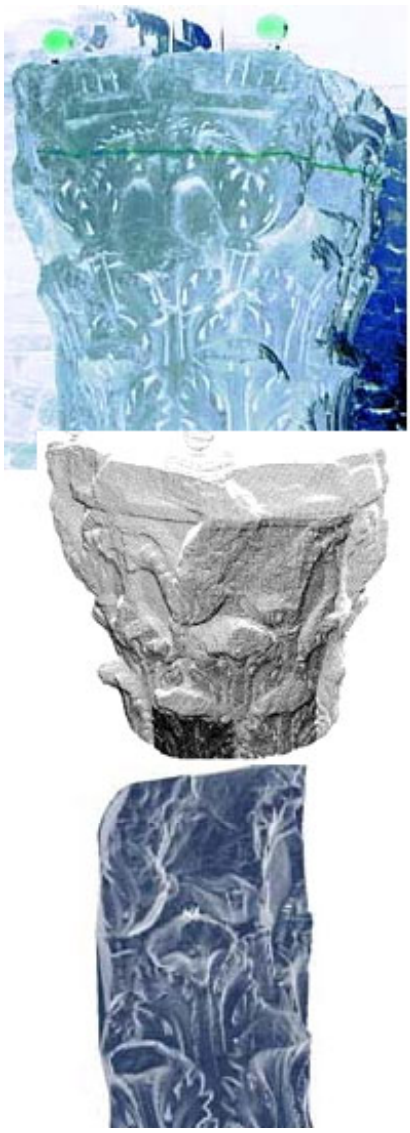


*Figure 97: Numérisation, nuage de points et surfaçage d'un chapiteau in http://www.archeodunum.ch*

[183] http://www.cyberware.com
[184] http://www.faro.com
[185] La société MENSI présente ses produits à l'adresse suivante: http://www.mensi.com

chercher sur des parties de l'image choisies interactivement (i. e manuellement) par un opérateur humain un ensemble de primitives géométriques simples (cylindres, sphères, etc…). Cette segmentation du nuage est donc basée sur la reconnaissance par l'opérateur d'une forme, le logiciel se chargeant d'en calculer les paramètres. Les applications principales du logiciel 3DIPSOS sont donc clairement à trouver dans le milieu industriel. L'étape de segmentation est pourtant même dans ce cadre entièrement laissée à la charge de l'opérateur. Un projet de recherche mené à l'ENSMP[186] en collaboration avec MENSI et EDF se propose de trouver des solutions pour réaliser de façon automatique ou semi-automatique la segmentation du nuage de points, segmentation en primitives simples telles qu'elles existent déjà dans le logiciel 3DIPSOS. Dans le champ de l'architecture patrimoniale, il est difficile de se contenter de telles entités géométriques. Le logiciel est donc au jour d'aujourd'hui assez fortement inadapté aux formes architecturales, même les plus communes. Si le balayage laser apparaît comme une technologie bien adaptée au relevé de formes dont la géométrie est simple ou dont un maillage rend compte de façon satisfaisante, il pose par contre des problèmes importants à l'échelle de l'édifice. En effet, un modèle de l'édifice n'a que peu de valeur pour l'analyste du patrimoine s'il est seulement maillé comme peut l'être une sculpture. Par ailleurs, une sculpture n'est pas significative par son dedans.
Or l'édifice, lui, n'a de sens que par ce qu'il décrit un ensemble de limites entre le dedans et le dehors. Le relevé laser fournit en fait, à l'issue de la phase de segmentation, une représentation tridimensionnelle de l'édifice sous la forme d'une sorte de feuille de papier pliée et repliée dans l'espace, bien loin de satisfaire aux besoins les plus basiques de la représentation d'un bâtiment étudié à l'échelle architecturale.
Il semble donc exclu d'utiliser un modèle maillé de l'édifice (issu du traitement du nuage de points tel que décrit plus haut) pour aller au delà de sa simple figuration. Reste alors à adapter l'étape de segmentation aux nécessités de la morphologie architecturale. Ceci constitue à l'évidence un travail de très grande ampleur dont les travaux du MAP-CRAI[187] établissent clairement la nécessité.
Le relevé par balayage laser est en définitive aujourd'hui une solution confirmée dans le cas de sculptures [Levoy, 1999] ou de restitution de grottes par exemple (pour ce dernier cas on se reportera à [StAubin, 1996]), mais reste très peu adapté à une étude de l'édifice telle que la pratique un architecte-conservateur.

**Le relevé photogrammétrique**

*La photogrammétrie : éléments de définition*

La technique de relevé par photogrammétrie traite une image globale de l'objet, un ensemble de perspectives enregistrées photographiquement. Le principe en est le suivant : depuis un objet photographié convergent vers le point de vue (centre optique de l'objectif) les rayons issus de chaque point de sa surface. Une perspective sera la section par le plan de l'image photographique de ce faisceau de rayons. Si l'on considère deux perspectives ainsi définies, l'objet photographié devient le lieu géométrique des points d'intersection de tous les couples de rayons homologues (i.e. issus du même point sur l'objet).

*Phase d'acquisition*

C'est l'enregistrement photographique des perspectives et la reconstitution du faisceau perspectif. La photogrammétrie procédant par intersection de rayons, sa précision sera d'autant plus grande que le triangle formé par les deux appareils et l'objet visé se rapprochera d'un triangle équilatéral.
Reconstituer le faisceau perspectif impose de connaître avec précision les données internes des appareils photographiques utilisés. Le matériel utilisé en photogrammétrie sera donc de haute qualité, rendant plus évidente la nécessité d'en limiter l'usage par l'établissement d'un cahier des charges précis.

*Phase de restitution*

Elle vise à définir un modèle, relatif à l'objet, formé par l'ensemble des intersections des rayons homologues. En l'absence de données externes sur la prise de vue (position des centres perspectifs, droite de visée, ...) la définition d'un système de référence propre à l'objet passe par deux étapes :

- Recherche de l'orientation relative de deux faisceaux en assurant l'intersection d'au moins cinq couples de rayons homologues.

[186] École des Mines de Paris, projet en cours présenté à l'adresse suivante: http://pantanal.ensmp.fr/Fr/Recherche
[187] UMR MAP-CRAI Nancy : http://www.map.archi.fr ou http://www.crai.archi.fr

- Mise à l'échelle par le rapport des distances séparant les points de vue lors de la prise de vues et lors de la restitution (i.e.; les bases); et orientation absolue sur des points connus.

Une fois le faisceau perspectif reconstitué puis ainsi placé et orienté dans le système de référence, tous les rayons homologues se coupent deux à deux. Une restitution aboutit donc à un modèle que l'on peut explorer, dont on peut extraire des mesures.

*Identification des points.*

L'identification de deux images homologues d'un même point de l'objet peut se faire sur chaque image si on dispose sur celles-ci de détails formels facilement reconnaissables. Plus généralement, on procède à un examen stéréoscopique fondé sur le phénomène physiologique du fusionnement binoculaire.

Aujourd'hui deux types d'appareils sont utilisés dans la restitution, les appareils de stéréophotogrammétrie analogiques (reconstitution mécanique de la convergence des rayons) et les appareils de stéréophotogrammétrie analytiques (calcul des coordonnées spatiales des points restitués).

La technique du relevé par photogrammétrie s'avère donc précise mais lourde, notamment dans la phase de restitution.

Plusieurs thèmes de recherches semblent aujourd'hui se dessiner dans la bibliographie de ce champ disciplinaire, dont nous donnons une liste non exhaustive :

- Comment relever à moindre coût (investissement matériel et humain) ?
- Comment appliquer ces techniques en dehors d'un milieu de spécialistes ?
- Comment adapter au mieux cette technique au support numérique ?
- Comment appliquer les principes de la photogrammétrie à la vidéogrammmétrie ?
- Comment tirer parti des algorithmes de traitement de la vision par ordinateur dans le cadre de la photogrammétrie numérique ?

Il n'entre bien sûr pas dans le cadre de ce travail d'aller au-delà de cette simple citation. Nous revenons par contre dans la section suivante sur quelques travaux plus centrés sur les problématiques particulières à la photogrammétrie architecturale. On pourra se reporter pour un survol plus complet des techniques photogrammétriques au journal de l'ISPRS (International Society of Photogrammetry and Remote Sensing) édité par Elsevier[188]

[188] Elsevier, éditeur du "ISPRS Journal of Photogrammetry and Remote Sensing" a son site web à l'adresse suivante : http://www.elsevier.com

## *ANNEXE 16 : LA PHOTOGRAMMÉTRIE ARCHITECTURALE*

Nous avons discuté des principes du relevé photogrammétrique dans une autre section de ces annexes, et allons ici aborder les différents champs applicatifs de cette technique dans le domaine de l'architecture. La photogrammétrie a été et reste utilisée majoritairement non pas à l'échelle architecturale mais à l'échelle urbaine: cartographie, cadastres, etc… [Gucek et al, 1997] proposent par exemple une étude du centre médiéval de la ville de Koper (Slovénie) dans laquelle leur intérêt se porte sur le revêtement des sols (relevé photogrammétrique) et sur le sous-sol (fouilles). Elle est aussi utilisée en archéologie dans le relevé de sites à partir de photos aériennes. On peut se reporter à [Doneus et al, 1997] qui détaillent une expérience combinée de photogrammétrie aérienne et d'inspection du sol par résonance magnétique sur un site archéologique en Autriche.

A l'échelle architecturale, le relevé photogrammétrique correspond à une commande qui en France émane essentiellement des services de l'état. Traditionnellement, ces relevés aboutissent à la production d'un ensemble de géométraux informant le conservateur de l'édifice. Jean Paul Saint Aubin [StAubin, 1992] donne du dossier de commande de la restitution photogrammétrique classique une description détaillée à laquelle je renvoie. Le redressement de photographies est le complément naturel de tels travaux. Il consiste à reproduire les conditions de la prise de vue sur un dispositif analogique ou analytique de façon à mettre à l'échelle dans un plan parallèle à l'objet l'image enregistrée. Les photos redressées peuvent être utilisées soit comme outil d'analyse d'une façade par exemple soit comme texture dans une maquette numérique d'édifice. On peut se reporter à [Baletti et al, 1999] pour une description détaillée de cette approche à l'occasion d'une étude préalable à restauration sur le château médiéval de Brenzone (Italie). Nous n'avons pourtant jusqu'à présent évoqué que des travaux dans lesquels l'information tridimensionnelle recueillie résulte en une représentation bidimensionnelle de l'objet observé. Le passage de ces approches en projection 2D d'un nuage de points dans l'espace aux approches réellement tridimensionnelles pose en effet un ensemble de problèmes auxquels s'intéressent de nombreuses équipes issues de la recherche ou de l'industrie[189]. On peut les résumer en deux questions :

- A quelles entités géométriques attacher les points mesurés, et comment le faire ?
- Comment attacher les observations mesurées aux informations non graphiques relatives aux édifices ? (On notera que la photogrammétrie s'appuyant sur le support photographique, elle apporte un plus par rapport aux autres techniques citées ici: elle fixe au delà d'une mesure ponctuelle une mémoire de l'édifice, ré-interrogeable qualitativement et quantitativement)

A la première question les tentatives de réponses que l'on trouvera dans la bibliographie font appel à une géométrie placée en intermédiaire entre le nuage de points et le modèle tridimensionnel du corpus architectural. Notre approche, explicitée dans la partie de ce document consacrée au projet, ne déroge pas à cette règle. Elle se distingue essentiellement par le fait qu'elle utilise cette géométrie en amont de la mesure puisque le modèle architectural dispose a priori d'un ensemble de spécificités morphologiques. Il s'agit dans notre cas donc non pas d'un processus de reconnaissance de formes dans un nuage de points mais d'étiquetage de ces points a priori, étiquetage fondé sur une analyse de l'édifice à relever, analyse qui, on l'a vu, est de toute façon traditionnellement menée dans le cadre d'études patrimoniales. Les éléments de corpus ne sont donc pour nous pas à retrouver dans un nuage de points, comme c'est le cas par exemple dans l'étude de Ayutthaya [Ogleby, 1997] où la restitution de la forme architecturale s'effectue par observation de profils décoratifs générant des surfaces de révolution à l'intérieur d'un outil de CAO (Microstation en l'occurrence). Le développement d'outils dédié à la photogrammétrie architecturale comme PhotoModeler[190], discuté dans [Hanke et al, 1997], s'inscrit également dans la démarche d'un raffinement a posteriori de la mesure en éléments de représentation du corpus architectural.

La deuxième question, attacher des informations non graphiques aux résultats du processus de mesurage des édifices, est également une question à laquelle beaucoup s'intéressent. [Ioannidis et al, 1999] proposent une approche combinée photogrammétrie architecturale / système d'informations localisées spatialement sur le cas des murs de Mécène (Grèce). S'appuyant sur un relevé photogrammétrique du

[189] Le CIPA (Comité International de Photogrammétrie Architectural) les regroupe, on peut consulter leur site (http://cipa.uibk.ac.at/) pour des références sur ce domaine.

[190] PhotoModeler est un produit de la société Eos basée au Canada ( http://www.photomodeler.com)

site, les auteurs utilisent des photos redressées pour identifier des blocs servant de polygones auxquels attacher des éléments d'informations. Ce travail situe bien la principale question que soulève la gestion d'informations attachées au relevé : la mesure doit renseigner un modèle sous-jacent. Nous citons en **annexe 17** avec plus de détails deux travaux de recherches qui illustrent les problématiques récentes en discussion aujourd'hui dans le champ de la photogrammétrie architecturale.

Dans le cadre de notre travail, il est important de noter d'une part que les problèmes que nous abordons de façon globale (renseigner un modèle architectural) sont le plus souvent dans la bibliographie que nous citons découplé en recherches sur la reconnaissance de formes et recherches sur les systèmes d'informations. Cela ne tient en rien au hasard mais au fait que dès l'origine le sujet de recherche que nous abordons a été mené à partir d'un axiome simple: mesurer un corpus modélisé a priori. Cette idée, déjà présente en filigrane chez Viollet Le Duc, continue de faire son chemin et se retrouve par exemple dans [Whiting et al, 1997] dont nous reparlerons ultérieurement. On retrouvera une présentation plus détaillée d'autres projets de recherche en photogrammétrie architecturale dans le mémoire de thèse de Pierre Drap [Drap, 1997].

*ANNEXE 17 : APPROCHES RECENTES EN PHOTOGRAMMETRIE ARCHITECTURALE*

**Une méthode d'acquisition automatique de modèles 3D à partir de photographies [Pollefeys et al, 1999]**

Je présente ici dans ses grandes lignes le travail d'extraction de modèles 3D à partir de photographies que développent les auteurs à l'université de Louvain (Belgique). Deux points permettent de mieux saisir l'importance de leur contribution : elle s'appuie sur l'analyse des images pour retrouver les paramètres des caméras (plus de mesures sur la scène ou de calibrations des caméras), et elle constitue une bonne alternative au balayage laser puisqu'elle propose un relevé global de la scène sans les restrictions de portée liées à celui-ci.

La scène est photographiée depuis différents points de vue. Les positions et orientations des caméras sont recalculées à partir des images. Une seule caméra est nécessaire à la prise de vue. Le système est constitué d'un ensemble de modules qui se passent de l'un à l'autre les informations nécessaires : module de reconstruction projective, module d'auto-calibration, module de correspondance de densités. Ces trois modules opèrent à partir d'un jeu de photographies une reconstruction 3D texturée des objets présents dans la scène. Cette reconstruction reste cependant non interprétative, ce qui constitue un avantage en terme de souplesse d'utilisation mais un handicap pour représenter un modèle architectural. Le premier module, reconstruction projective, est basée sur la recherche sur les images de motifs (coins) homologues. En fixant des contraintes additionnelles (pixels carrés par exemple) le module suivant calcule une reconstruction métrique partielle de la scène. Le module de correspondance de densités s'appuie sur la connaissance dans l'espace de la scène inférée au module 1 pour reconstituer l'ensemble de la scène. Le modèle 3D final est constitué de facettes triangulaires définies par la géométrie de la surface et texturés à l'image des photographies. Les auteurs ont appliqué leur technique sur un site archéologique romain avec succès. Bien qu'à l'évidence très porteuse, cette technique repose pourtant le problème du rôle du modèle 3D. Comment se servir d'un modèle 3D sans épaisseur (au sens architectural du terme), et dont la représentation géométrique traite indifféremment toutes les surfaces observées ? Les applications notamment en terme de réutilisation de fonds documentaires photographiés semblent potentiellement nombreuses. Basée sur la photogrammétrie, cette approche d'acquisition automatique de modèles 3D à partir de photographies n'a pas à ce jour pleinement tiré partie de ses différences avec le balayage laser. Elle lui est à l'évidence supérieure notamment en terme de coût d'usage et d'exploitation ou de restitution des textures, mais ne s'en différencie pas de façon notable dans l'absence de sémantique attachée à l'objet observé.

**Reconstruction 3D à partir d'une image en utilisant des contraintes géométriques [Van Den Heuven, 1998]**

Nous présentons avec ce deuxième travail, mené à l'université de Delft (Pays-Bas), une approche qui vise également à simplifier la phase de restitution du processus de mesurage. Son principe est toutefois très différent puiqu'il s'agit ici de retrouver un objet polyhédrique dans une scène à partir d'une seule image de cette scène et de connaissances relatives à la géométrie de l'objet observé. Cette approche vise donc à obtenir a posteriori un modèle interprétatif géométrique à partir d'une photographie.

L'auteur introduit son travail en en positionnant le champ d'application : l'acquisition de modèles 3D pour la réalité virtuelle ne nécessitant pas le degré de précision que l'on obtient en stéréophotogrammétrie classique. La méthode présentée s'appuie sur l'intervention d'un opérateur humain qui va définir en phase d'analyse de l'image un certain nombre de contraintes géométriques inhérentes à l'objet mesuré (parallélisme de ligne par exemple). Elle nécessite une calibration de l'appareil utilisé, à la différence de la démarche présentée ci-dessus.

Le processus que détaille l'auteur comprend deux phases :

- Observations photogrammétriques (position et orientation de lignes sur l'image ou points à leurs intersections).
- Description topologique de l'objet (informations sur sa géométrie).

Dans la première phase, l'opérateur relève un ensemble de lignes (souvent présentes en architecture) de façon manuelle, au moins en partie puisqu'il s'agit de définir seulement les lignes pertinentes dans la définition de l'objet. Dans la deuxième phase, un ensemble de contraintes géométriques et topologiques est affecté aux lignes relevées : coplanarité, parallélisme, perpendicularité, symétrie, distance. Ces contraintes sont séparées en deux groupes : contraintes de topologie (liées à l'image ou à l'objet observé) et contraintes

internes (liées à la géométrie de l'objet). Les premières, toutes de type coplanarité, assurent une représentation de contour valide. Les secondes représentent une connaissance a priori sur la géométrie des objets à travers trois types de contraintes : parallélisme, perpendicularité et symétrie. La redondance des contraintes géométriques est traitée par un système d'équations de conditions résolu par les moindres carrés (pour l'heure parallèlisme et perpendicularités). L'étape de reconstruction du modèle utilise un relevé partiel (partie vue de l'objet) et des contraintes (symétries et perpendicularités) qui peuvent entrer en jeu pour produire un modèle hypothètique du non-observé. La plate forme de visualisation (VRML) permet d'afficher à la fois la géométrie déduite et de plaquer sur celle ci les zones adéquates de l'image pour obtenir un modèle texturé de l'édifice. L'auteur présente un exemple d'application à l'architecture patrimoniale dans lequel l'ensemble des lignes ont été mesurées manuellement, et les principales contraintes utilisées définissent leur parallélisme et la perpendicularité des principaux plans (murs) de l'édifice. L'acquisition de l'image pose un certain nombre de problèmes que l'auteur décrit avec précision : orientation optimales de l'axe optique de l'appareil et des plans principaux de l'objet observé, conditions de prise de vue, etc..
Bien que ne se différenciant pas en terme de résultats du travail de [Pollefeys et al, 1999] (production d'une maquette tridimensionnelle non interprétative de l'objet mesuré, dans l'esprit d''un balayage laser amélioré), la contribution de [Van Den Heuven, 1998] apporte trois éléments qui nous semblent aller dans le sens d'une meilleure adéquation à l'étude du patrimoine bâti :

- La mesure est complémentée par une analyse de l'objet, aujourd'hui encore seulement géométrique, mais qui donne la possibilité de relever sur l'image des cohérences locales directement issues de la logique architecturale.
- L'ajout de contraintes géométriques permet de prendre en compte au-delà d'une morphologie une logique de composition.
- Enfin, le recours à une seule image ouvre en terme de réutilisation d'anciens clichés de nombreuses perspectives qu'il faut néanmoins étudier en fonction de la necessité dans le processus présenté de disposer des informations de calibration de l'appareil.

Ce travail positionne donc connaissances de l'objet mesuré (ici à l'échelle géométrique) et processus de mesurage photogrammétrique comme interdépendants, position que nous reprenons à notre compte comme on l'a vu dans la partie projet de ce document. Mais comme dans le cas précédent le résultat obtenu est une sorte de "photographie tridimensionnelle" de l'édifice dans laquelle la sémantique du domaine est ignorée, et de laquelle peu d'usages semblent pouvoir être faits.

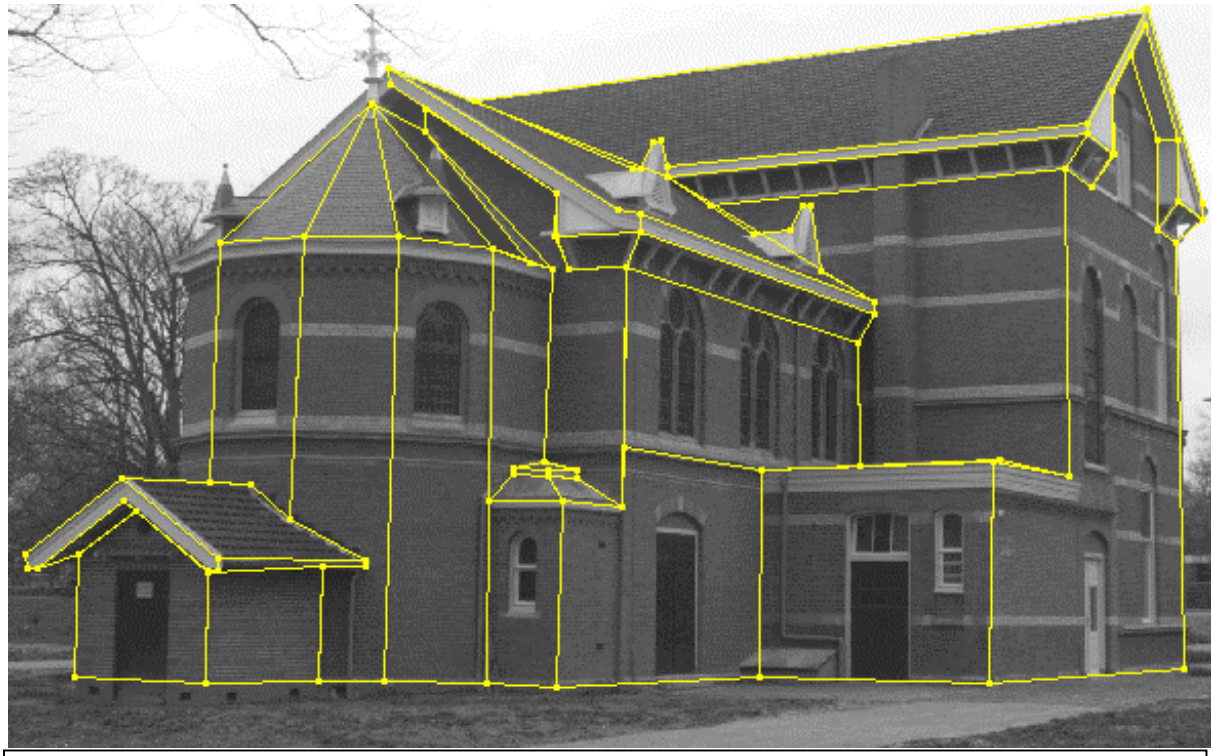


*Figure 98 : Position et orientation de lignes sur l'image, le cas d'étude présenté dans l'expérience de [Van Den Heuven, 1998]*

## ANNEXE 18 : REPRESENTATION ET ARCHITECTURE PATRIMONIALE

Ronald Stenvert, déjà cité, rappelle que notre connaissance de l'histoire de l'architecture s'appuie, outre sur des sources textuelles, sur un ensemble de données graphiques allant des géométraux ou œuvres peintes aux photographies [Stenvert, 1991, p23]. Ces représentations graphiques sont, écrit il, l'objet d'interprétations de la part de l'historien ou du conservateur qui seules leur confèrent un statut de données par opposition à un statut de simple information. Ces interprétations se font sur la base d'informations généralement incomplètes (vues partielles d'un édifice) ou subjectives (recopies d'œuvres peintes), et sont naturellement remises en question au cours du temps. L'auteur en conclue dans un premier temps qu'aucune donnée graphique ne peut servir utilement la documentation scientifique d'un édifice sans être accompagnée de données notamment textuelles renvoyant à son analyse [Stenvert, 1991, p43]. L'image, mot que nous utilisons ici en synonyme de représentation graphique d'une scène architecturale, donne lieu pour l'auteur à deux types d'analyses: globales ou internes.
Les premières traitent l'image comme un tout, ou l'illustration d'un propos, elles sont utilisées majoritairement par les historiens.
Les secondes s'intéressent au contenu de l'image, à sa forme, à son sens, elles sont utilisées majoritairement dans les domaines de l'art et de l'architecture.
L'auteur place la tripartition de Panofsky[191] comme étape fondatrice dans la formulation d'une théorie de l'image. Il en rappelle les principes en indiquant d'abord que l'objet du travail de Panofsky est d'étudier le sens potentiellement contenu dans une œuvre. Panofsky définit trois niveaux d'analyse du sens de l'image : pré-iconographique, iconographique et iconologique. Une analyse pré-iconographique va fixer par énumération ce que contient la scène représentée, combinée avec une analyse formelle des motifs présents (style par exemple). Une analyse iconographique fixe un thème conventionnel dans lequel s'inscrit l'image, un espace de référence dans la terminologie de P.Boudon. Elle s'oppose à la première en s'intéressant aux référents culturels de l'image en terme de sujets plutôt que de motifs picturaux. Le niveau iconologique est défini par le terme "sens intrinsèque" qui recouvre l'idée qu'une représentation s'inscrit plus largement dans un contexte social et historique à travers lequel elle peut être analysée. Ce principe se manifeste concrètement par exemple par une pratique spécifique de la composition graphique. L'interprétation de l'image se fait ici par une démarche synthétique plutôt qu'analytique. R.Stenvert résume les principes et la pratique de la tripartition de Panofsky dans le tableau suivant :

| *Niveau d'interprétation* | *Action concernée* | *Savoirs concernés* | *Principes correcteur de l'interprétation* |
|---|---|---|---|
| Analyse formelle divisée en deux sous-groupes: faits et expressions | Description pré-iconographique (énumération des motifs observés et analyse de formes pouvant placer l'objet étudié dans une évolution stylistique) | Expérience pratique | Histoire des styles (rapport entre objets/ évènements et forme les exprimant dans un contexte d'évolution historique ) |
| Analyse des conventions attachant un ensemble de motifs à un thème ou une allégorie | Analyse iconographique | Connaissance des sources littéraires (familiarité avec les thèmes et allégories) | Histoire des types (rapport entre thèmes/ concepts et objets/ évènements les exprimant dans un contexte d'évolution historique) |
| Contenu ou sens intrinsèque : valeur symbolique | Analyse iconologique (interprétation synthétique plutôt qu'analytique de l'image) | Intuition (familiarité avec les tendances essentielles de l'esprit humain) | Histoire des symptômes culturels ou symboles en général (rapport entre tendances essentielles de l'esprit humain et thèmes/ concepts les exprimant dans un contexte d'évolution historique) |

Tripartition de Panofsky [Stenvert, 1991, p41]

Ronald Stenvert s'appuie sur la tripartition de Panofsky pour définir une triple grille d'analyse des données graphiques, adaptée au domaine spécifique de l'architecture, dont il nomme les critères morphologie,

[191] Erwin Panofsky (1892-1968), ouvrages cités : "Studies in iconology: humanistic themes in the art of the renaissance" première édition 1939, et "Meaning in the visual arts" , première édition 1955.

typologie et iconologie, rapprochant les relations des deux derniers niveaux des liens phénotypes/génotypes. Après avoir indiqué la nécessité d'accompagner toute information graphique (existante ou nouvelle) de données relatives à son analyse, l'auteur fournit donc un instrument méthodologique pour traiter sémantiquement l'image.
Enfin, il rappelle qu'en architecture les données manipulées par l'analyste sont un substitut numérique (qu'il nomme pseudo meta source) à l'objet réel, étudiées lorsqu'il s'agit d'images en fonction de deux théories divergentes (l'image comme grille optique, l'image comme langage pictural) auxquelles, par exemple, se rattachent respectivement les travaux cités en conclusion à la section précédente.
En résumé, le travail de R.Stenvert nous éclaire sur trois points:

- L'image s'utilise dans le cadre d'études patrimoniales dans un contexte en déterminant la (les) signification(s).
- Ces significations correspondent à des échelles de lecture de l'image, elles-mêmes liées à des échelles de lecture de l'édifice.
- L'image n'est pas l'objet, elle se substitue à lui à fins d'études.

Dans la pratique quotidienne, le niveau de signification d'une image de synthèse est souvent bien en deçà d'un simple croquis à main levée. On en comprend à la lecture de ce qui précède aisément une raison majeure: elle est une production codifiée destinée à être interprétée par celui qui l'observe. On est loin de l'évidence naïve qui caractérise beaucoup de scènes dites de "réalité virtuelle". Le potentiel d'utilisation de la maquette tridimensionnelle comme outil générateur d'images se situe bien ailleurs. Nous considérons que deux points clés sont à retenir:

- La maquette 3D interdit les omissions, et par conséquent questionne le modèle exhaustivement.
- Outil privilégié de l'architecte, elle peut lui servir de mode d'interrogation des données non graphiques relatives à l'édifices (interface spatial).

Les rapports de dépendances qu'entretiennent le dessein de l'architecte, les échelles du projet et le codage de son expression graphique ne sont plus à établir. On se reportera par exemple à [Lebahar,1983] pour qui le dessin est un moyen de simulation de problèmes et de solutions, ou encore [Boudon, 1971], déjà cité. Il nous importera ici d'une part d'établir par un rapide historique le rôle de l'image dans l'activité de l'architecte et d'autre part de rechercher en quoi les techniques de représentation volumique qu'offrent les outils informatiques sont adaptées ou non à ce rôle. En effet, le dessin d'architecte est une pratique dont l'évolution passée a permis de codifier l'usage en fonction de son contexte d'utilisation.
A l'instar de l'activité de reconstruction où, par exemple, le souci de marquer les différences entre l'original et la copie est constant, la représentation graphique du bâti doit intégrer la notion d'échelles, de degré de certitude, etc… On doit reconnaître que ce souci, loin d'être pris en compte ou même simplement évoqué, est aujourd'hui à peine présent dans la plupart des travaux d'imagerie, sans doute parce que ceux-ci ont dans un passé récent été menés principalement en dehors du contexte professionnel de la conservation du patrimoine architectural.

**historique**

La représentation graphique en architecture placée entre le souci d'expliquer et celui de promouvoir n'est pas un phénomène neuf: il est apparu avec l'invention même de la perspective. Dès le début de la période renaissance deux modes de représentation de l'édifice coexistent: la représentation en géométral et la perspective (axonométrique, cavalière ou à points de fuite). Jean Cuisenier, on l'a vu plus haut, rappelle les conditions dans lesquels le géométral accompagne une description textuelle de l'édifice dans le dispositif Serlien. Le rôle de la représentation en géométral ne fera dès lors que s'affirmer[192], avec d'ailleurs des répercussions importantes dans la pratique de la conception architecturale, en particulier au XIXème siècle.
Viollet Le Duc l'explique en ces termes :"*le monument tracé sur le papier ne tient compte généralement ni du lieu, ni de l'orientation, ni des effets d'ombre et de lumière, ni de l'entourage, ni des différences de niveaux pourtant si favorables aux formes architectoniques* […] *l'architecte pense à élever des façades symétriques, pondérées, une grande boîte dans laquelle on viendra, après coup, disposer les services comme on pourra* […] *formules académiques très récentes, mais peu raisonnées* […] *qui permettent à chacun de s'ériger en juge dans les questions d'architecture* […]"[Viollet Le Duc, 1863/1977, p255-256]. Il

[192] Voir [Cuisenier, 1991], déjà cité.

ajoute plus loin, défendant l'élégance de l'église de Vézelay : "*car cette architecture est faite en vue de l'exécution, non pour satisfaire un dessin géométral*" [Viollet Le Duc, 1863/1977, p261].
Claude Perrault [1613-1688] explique, dans l'introduction de sa traduction des dix livres d'architecture de Vitruve, l'usage qu'il entend faire des figures dessinées : "*les figures sont de trois espèces: il y en a qui n'ont que le premier trait pour expliquer les mesures et les proportions qui sont prescrites dans le texte; les autres sont ombrées pour faire voir l'effet que ces proportions peuvent faire étant mises en œuvre, et pour cette même raison quelques - unes de ces figures ombrées ont été faites en perspective, lorsque l'on n'a pas eu l'intention de faire connaître ces proportions au compas, mais seulement au jugé*"[Fichet, 1979, p220].

Le dessin géométral, accompagnant une description textuelle de l'ouvrage à bâtir, s'inscrit dans la lignée de la pratique des tracés chez les confréries d'artisans du moyen âge, que l'on retrouve dans [Viollet Le Duc, 1877/1978, pp88-130]. Claude Perrault situe bien les rôles avant tout complémentaires des deux types de représentation qu'il maîtrise. En effet, si la codification en géométral permet de quantifier l'édifice, sa description en perspective permet au lecteur de le qualifier selon son *jugé*. On retrouve cette même analyse dans le traité d'architecture d'Alfred Hébrard publié en 1897: "*Les différents aspects de l'édifice à construire sont dessinées par les procédés de la géométrie descriptive et de la perspective. Les premiers sont les plus exacts, les plus commodes et les plus employés; les vues perspectives ne servent qu'à les compléter, à en rendre la compréhension plus facile*"[Hébrard, 1897, p3]. L'auteur situe ainsi le rôle de la représentation graphique dans le travail de l'architecte: "[…] *les idées et les formes ne peuvent être suffisamment définies par l'imagination et la mémoire seules, et il est indispensable d'aider nos facultés par le dessin, c'est à dire la représentation figurée de ces idées et de ces formes*".
Hérité du dispositif de Serlio, le rôle contemporain du dessin géométral (pièce du contrat maître d'œuvre / maître d'ouvrage, vecteur de communication technique entre intervenants) est à peu prés évident. Jean-Paul Saint Aubin en donne cependant une définition plus fine, en le positionnant par opposition à la photographie comme une choix raisonné d'informations [StAubin, 1992, p190]. Selon l'auteur, la représentation fonctionne dès lors comme un signifiant plus ou moins complexe à interpréter. E
n se reportant à l'aide-mémoire de l'architecte d'Eugène Barberot, publié en 1922, on note que la représentation graphique intervient alors également dans la présentation de tracés géométriques, dans la définition de la mouluration, et dans la statique graphique [Barberot, 1922, pp 3-53, 514-540, 541-573].
A la lumière des contributions que nous avons cité, le géométral, ou la représentation graphique sélective pour reprendre la terminologie de J.P Saint-Aubin, se pratique dans le champ de la conservation en architecture avec quatre rôles particuliers:

- Relevés de l'existant [StAubin, 1992], [Barberot, 1922], [Alkhoven, 1993]
- Etude de tracés [Barberot, 1922] , [Hébrard, 1897]
- Etablissement de typologies [Tajchman, 1989] , [Stenvert, 1991, [Alkhoven, 1993]
- Représentation d'hypothèses [Tajchman, 1989], [Fichet, 1979]

Cette pratique sert le travail de documentation de l'édifice, ou de recherche du conservateur, en ce qu'elle lui permet de représenter par une codification adéquate de la forme graphique le champ sémantique qu'elle recouvre[193].
A l'opposé, la perspective est comprise comme relevant d'un souci de séduction. On l'a évoqué, elle est un enregistrement sur un support plan d'une *image globale de l'objet*, non déterministe, issue du miroir de Brunelleschi ou du cadre perspectif de Dürer: elle définit par là le concept même de la photographie et de ses évolutions, notamment la photogrammétrie. E.Barberot explique ainsi à ses lecteurs comment tracer une vue perspective: "*On imagine que des rayons lumineux partant de tous les points des contours [apparents des objets] et se dirigeant vers l'œil du spectateur, sont coupés par un plan appelé tableau* […]"[Barberot, 1922, p49]. En ne consacrant à ce sujet que trois pages des 673 que comprend son ouvrage, l'auteur montre indirectement le rôle que joue alors cette forme graphique: une "*affaire de goût"* , pour reprendre ses termes.
On serait dès lors tenté de ne reconnaître comme rôle de cette forme graphique que celui d'illustrer, avec goût si possible, le propos du chercheur ou du conservateur. Le développement des technologies informatiques pendant ces vingt dernières années apporte de nombreux démentis, dont quatre nous semblent particulièrement affirmés dans le champ de l'architecture patrimoniale :

[193] On en trouvera des définitions ou exemples en particulier dans [Lebahar, 1983], [Barberot, 1922], [StAubin, 1992].

- Avec les outils de CAO, géométraux et vues perspectives sont désormais issus d'une même maquette tridimensionnelle, rompant les oppositions quantification / qualification et pièces contractuelles ou interprétatives / représentation globale.
- Les simulations, en imagerie de synthèse, d'hypothèses de restitution, rendent plus aisée la vérification, la validation et la communication des hypothèses, et posent de nouveaux problèmes de codification de l'image (voir plus haut).
- Les techniques de réalité virtuelle immersives impose à l'auteur d'une scène une validité globale du modèle et donnent à l'utilisateur les moyens de la vérifier.
- Le paradigme des SIG donne des géométraux comme maquette interprétative une définition renouvelée qui intègre la notion de champs sémantiques différents à interroger.

Le développement de ce qu'il est convenu d'appeler les technologies de l'information interroge l'accessibilité et la lisibilité de la documentation de l'édifice.

ANNEXE 19 : PHOTOREALISME ET PATRIMOINE ARCHITECTURAL

## Virtual Stonehenge [Burton et al, 1997]

L'expérience de [Burton et al, 1997], bien que s'intéressant à un site sans doute plus archéologique qu'architectural (bien que ce point soit sujet à discussions), a ceci d'exemplaire qu'elle s'est appuyée sur de gros moyens sans pour autant parvenir à faire de l'image plus qu'une image. Autrement dit, elle pose bien la question du sens (et du manque de sens) de représentations photoréalistes , et ce de l'aveu même de ses auteurs.
On se reportera à leur publication pour un intéressant aperçu des questions concrétes auxquelles ils ont du répondre.
La maquette 3D du site archéologique de StoneHenge est présentée par ses auteurs comme un outil d'aide à la gestion globale du site et de ses environs. Elle a été également, pour un fabriquant très connu de processeurs, une action de sponsoring en 1995. Ce projet a abouti à la production de deux maquettes : une en VRML disponible sur le Web, une dite photoréaliste à partir d'un relevé photogrammétrique du site. Les auteurs concluent la présentation à laquelle je renvoie par ces deux remarques que le lecteur appréciera :"*at the time of production* [Virtual Stonehenge] *was one of the biggest and best PC based heritage reconstructions to date. Much more work must be done if it is to become established as a serious tool for analysis*".

*Figure 99 Eventails de rendus photoréalistes du site de Stonehenge*

## ANNEXE 20 : MÉTHODES DE DOCUMENTATION ET D'ARCHIVAGE

L'encyclopédie du patrimoine de René Dinkel [Dinkel, 1997] est présentée comme un ouvrage de synthèse sur le patrimoine bâti rédigé à partir de documents émanant de l'ensemble des acteurs impliqués dans la conservation des édifices. Elle s'articule en deux parties principales et deux recueils de références.
La première partie, doctrines et pratiques, présente à partir de cas concrets les problèmes pratiques que pose l'intervention sur le bâti ancien, mais aussi les questions de doctrine (ou de "parti" pour reprendre une terminologie courante chez les architectes) ainsi que les différents intervenants institutionnels dans le cas français. L'auteur insiste à ce sujet sur la masse de documents techniques, scientifiques et archéologiques recueillis aujourd'hui par les services de la direction du patrimoine.
Dans la seconde partie de son ouvrage, l'auteur propose un dictionnaire très complet des termes utilisés dans l'intervention sur le bâti. S'il n'apporte, en terme de méthodologie, que peu d'enseignements comparé au travail de Jean-Marie Pérouse De Montclos, déjà cité au chapitre vocabulaire [Pérouse De Montclos, 1988], ce dictionnaire comprend par contre des définitions qui vont au-delà de la simple description de l'édifice: acteurs, doctrines, termes administratifs, etc… L'encyclopédie du patrimoine se termine par deux recueils de références: un carnet d'adresses et une bibliographie. Dans la première partie de son ouvrage, l'auteur consacre un chapitre au problème qui nous occupe: la documentation de l'édifice. Il propose d'abord une distinction entre les données recueillies sur le projet d'architecture devenu édifice bâti, et les données recueillies sur l'existence de cet édifice au travers de l'histoire, qu'il nomme "*documentation de saisie et d'analyse*"[ Dinkel, 1997, p 133].

René Dinkel s'intéresse ensuite à l'utilisation des sources anciennes et à la constitution de sources nouvelles. Dans le premier cas, l'auteur rappelle que "*la condition première d'une bonne connaissance d'un édifice est* […] *de réunir une documentation objective*"[ Dinkel, 1997, p 135]. Ce fonds documentaire comprend pour l'auteur:

- Documentation graphique et photographies anciennes.
- Analyses archéologiques et historiques.
- Traités d'architecture en relation avec la période concernée.
- Inventaires technologiques et dictionnaires de techniques architecturales.

*Figure 100: Bednarze (Atelier du tonnelier) in Codex Picturalis Baltasaris Behem [Behem, 1501]*

Les documents manipulés ici sont assez naturellement des textes (originaux ou rééditions), des illustrations (peintures, lithographies, gravures, dessins), des photographies, des plans, dont le thème n'est pas toujours, loin s'en faut, celui qui nous occupe.
L'auteur cite par exemple comme source documentaire dans l'étude de l'architecture religieuse les dessins du père bénédictin Gabriel Bucelin effectués à l'occasion de ses voyages d'abbaye en abbaye. Sur notre terrain d'expérimentation, on peut citer les illustrations du Codex Behema[194], daté de 1501, qui définissait les privilèges de la cité et les statuts des guildes. Ces illustrations ont pour thème la vie quotidienne des marchands ou artisans dans la ville, et constituent de fait le plus ancien témoignage figuré sur l'architecture de la ville. C'est en fait dans les arrières plans de ces illustrations qu'il faut rechercher des informations sur la forme de la ville.
Par ailleurs, les sources anciennes documentant un édifice ne sont centralisées dans un centre de documentation que de façon exceptionnelle, posant bien entendu le problème de leur accessibilité, et interrogeant de façon forte les *nouvelles technologies de l'information*, pour utiliser une nouvelle expression pour un problème ancien.

Trois conclusions sont à retenir :

[194] Une page web du site de la bibliothèque de la Jagiellonian University de Cracovie lui est consacrée: http://www.bj.uj.edu.pl/bjmanus/codex/titlpg_e.html

- Le fonds documentaire ancien n'est pas nécessairement attaché à l'étude de l'architecture, il nécessite par conséquent de la part de l'historien de l'architecture une analyse qui fournit les clés de son déchiffrement.
- Outre des informations relatives à l'édifice traité, le fonds documentaire ancien comprend un ensemble de données plus générales qui pourtant s'appliquent à lui et doivent donc être prises en compte.
- Les sources anciennes ne sont le plus souvent pas accessibles directement, et dans bien des cas dispersées dans plusieurs institutions.

La constitution de sources documentaires nouvelles est pour René Dinkel l'action de relever l'édifice. Ce qu'il entend par là est à la fois une mesure dimensionnelle mais aussi une évaluation archéologique destinée à fixer notamment l'état sanitaire de la construction. Le relevé apparaît donc comme le document permettant d'identifier et de situer sur un édifice les irrégularités, les dysfonctionnement de la structure, toutes les meurtrissures et anomalies qui accompagnent l'acte de construire, signant l'histoire et l'état du bâtiment.
J.P. Saint-Aubin, déjà cité [StAubin, 1992], distingue, selon le type d'information recherchée, quatre modes de relevés d'architecture :

- *relevés documents*: relevés anciens ou provenant de sources extérieures; n'offrant pas de garanties de fiabilité.
- *relevés schématiques*: mesurage à vue sans instrument, estimations de proportion, longueurs évaluées au pas; formant un groupe de méthodes de précision géométrique très faible.
- *relevés expédiés*: mesures de distances avec instruments, triangulations enchaînées; méthodes dont la précision reste encore aléatoire.
- *relevés réguliers*: mode topographique (polygonation et relevé de détails) ou photogrammétrique; on peut ici évaluer la précision de la mesure, précision qui peut atteindre le millimètre.

L'exploitation et la représentation des résultats obtenus dépendra évidemment de la méthode choisie, allant de la simple inscription à un groupe stylistique ou typologique à l'élaboration de possibles diagnostics sur l'état structurel de l'édifice.
Ainsi, un dossier à instruire sera t'il documenté en fonction du besoin d'analyse identifié : dossier de repérage préliminaire (localisation, aspects visuels, typologies,...) ou dossier d'études (exploitation *finalisée* des données primaires). La distinction que René Dinkel fait entre sources anciennes et relevé ne doit pas faire oublier qu'un relevé existant peut à brève échéance, compte tenu des transformations de l'édifice ou de l'amélioration des techniques de mesure, s'intégrer à ce qu'il nomme source ancienne. Néanmoins cette distinction reste valide notamment parce qu'elle positionne le relevé comme un travail centré sur une description de l'édifice en temps qu'objet physique par opposition à une documentation plus indirecte que serait la source ancienne. L'auteur décrit par la suite les méthodes traditionnelles de représentation graphique ou volumique de l'édifice (géométraux, maquettes, etc…) sans s'attarder sur les apports des plateformes informatiques.
René Dinkel formule cependant une conclusion qui correspond directement à notre problématique de recherche: "[…] *la constitution d'une documentation objective bien étudiée, des relevés exacts, des mesures fiables, constituent la banque de données indispensable à la confection d'images de synthèse représentatives. Ces données, avant même l'intervention de la machine, permettent d'établir les états successifs.*"[Dinkel, 1997, p 147].

*ANNEXE 21 : SYSTEMES D'INFORMATION A L'ECHELLE DE L'EDIFICE*

**Des outils d'enregistrement assistés par ordinateur automatisent la création d'un système d'information de site [Whiting et al, 1997]**

Les auteurs partent du constat que le développements des outils informatiques a bénéficié aux processus de relevé, de stockage d'informations et de représentations des édifices mais en leur proposant des outils séparés. Leur approche vise à d'une part assister à la levée des données sur site (relevé au sens large du terme) et d'autre part à générer à partir de la base de données résultante une maquette numérique dédiée à un outil de CAO. La maquette numérique est alors utilisée comme interface d'accès aux données contenues dans la base. Les auteurs partent d'une stratégie d'acquisition de données dans laquelle la base de données à renseigner contient l'ensemble des informations collectées (textes, photographies, mesures, etc..), pour échapper au partitionnement des observations qu'accompagne l'utilisation de logiciels d'imagerie, de traitements de textes, etc… Ils écrivent dans leur introduction : "*The database not the drawing should be the heritage record*".

Leur approche implique de procéder à un relevé dirigé de l'édifice dans lequel l'observateur fixe pour chaque élément observé (quelle que soit l'échelle de cet élément) trois champs basiques : position, type et identifiant. D'autres champs complètent la description de chaque enregistrement comme les dimensions de l'élément, des mots-clés, une description de l'état de l'élément, etc… Chaque enregistrement correspond néanmoins de fait non pas à un objet mais à un point relevé, seule la lecture dans le champ identifiant d'une même chaîne de caractères permet au système d'interpréter ces points comme un objet unique donnant accès aux données contenues dans les tables de la base de données. Dans la phase de création de la maquette tridimensionnelle, à la représentation de l'élément repéré par ces points est affectée un symbole tridimensionnel en fonction du type déclaré. On retrouve donc là un principe sur lequel nous nous appuyons également : un relevé dirigé par la connaisance qu'a l'observateur de ce qu'il mesure. Le développement présenté par les auteurs s'appuie sur le logiciel de CAO Autocad et permet par conséquent d'éditer la forme géométrique proposée par la librairie de modèles. Les auteurs insistent sur l'utilisation de la maquette 3D comme d'une interface pour questionner la base de données renseignée par le relevé. CART (Computer Aided Recording Tools), nom donné par les auteurs à leur développement, se présente donc comme une démarche de relevé des édifices s'appuyant, en amont, sur une connaissance de l'objet mesuré, et utilisant, en aval, une maquette numérique tridimensionnelle comme moyen d'accéder aux informations collectées à l'occasion du dit relevé. CART appelle plusieurs commentaires :

- Il pose clairement le problème d'un système d'informations à l'échelle architecturale.
- S'appuyant sur des outils commerciaux (SGBDR et logiciel de CAO standards), il ne résoud pas frontalement le problème de la portabilité des données.
- Utilisant le point comme seul concept susceptible d'enregistrement dans la base de données, concept auquel sont attachés un ensemble de descriptifs, il se place dans la perspective d'une utilisation restreinte à la gestion de sites à relever.
- Il implique un mode de relevé particulier, dont le côté positif est la souplesse (la définition du type est du ressort de l'observateur). Par contre, en traitant sans distinction ni relation le clou et la poutre, le système interdit l'utilisation de dépendances sémantiques entre objets.
- Il propose dans la maquette tridimensionnelle produite une représentation symbolique des objets relevés dont la vocation est de donner accès aux informations sous-jacentes, en utilisant l'étiquettes des points comme artifice de groupement sémantique des données.

On le voit, la démarche des auteurs, dont je n'ai cité que les principes, porte en elle largement autant de questions que de réponses. Je renvoie donc à la lecture de leur article : [Whiting et al, 1997]. Il nous faut retenir que le passage des SIG à un système d'information à l'échelle architectural pose le problème du concept spatial sur lequel appuyer la démarche d'enregistrement et de représentation des données (ici le point relevé). En corollaire, il nous faut nous interroger sur la capacité d'une représentaion 3D à interfacer l'ensemble des données hétérogènes dont nous avons établi l'existence à partir des seuls concepts spatiaux. Le problème est évoqué dans un second travail à vocation de

système d'informations dont je présente les principes ci-dessous.

**Le projet ENVIART[195] (Baroque Artificial Marble: Environmental Impacts, Degradation and Protection)**

Ce travail de recherche a pour thème central une étude de la technique du faux-marbre dans l'architecture baroque. Ce qu'on appelle faux-marbre est du gypse additionné de pigments afin, après polissage, de produire une composition de couleurs et de motifs imitant le marbre. Le terrain d'expérimentation choisi est la chapelle des ducs de Krzeszów (Pologne).
L'objectif du projet est d'abord d'étudier sur cet édifice les phénomènes de dégradations des plaquages intérieurs en faux-marbres. Ces dégradations, effets essentiellement des remontées capillaires et des infiltrations dues aux pluies (humidité des murs et cristallisation de sels minéraux en surface), ont fait l'objet d'études physico-chimiques poussées à partir d'observations et de prélèvements.
Le second objectif affiché par l'équipe du projet est de constituer une documentation hypermedia pour le Web permettant d'accéder aux résultats de leurs mesures. A cette fin, une plateforme de consultation des données[196] recueillies a été construite dont le corpus d'objets architecturaux sert d'interface, ou plus précisément dont les faux-marbres plaqués sur la structure de l'édifice servent d'interface.
Nous allons voir dans un premier temps comment cette plateforme a été conçue, puis les conclusions que tire de cette expérience Z.Wikłacz, auteur d'une thèse de doctorat de l'université polytechnique de Cracovie (soutenue en 1999) dont ce travail fait partie.
Les données auxquelles accéder étant en partie attachées à des éléments de corpus, une maquette tridimensionnelle de l'édifice a été construite manuellement. Cette maquette, au format VRML, est accessible sur le Web à l'intérieur des navigateurs standards. Toutefois, étant au départ assez détaillée, elle a du être simplifiée pour alléger son téléchargement.
Deux types de représentations ont donc finalement été choisis : une scène interactive VRML simplifiée, des images fixes plus détaillées. Les dégradations observées portent sur des surfaces planes, indépendamment d'une analyse architecturale de l'édifice. A chaque surface étudiée est attaché (lien URL dans la scène VRML) un ensemble d'observations sous la forme de fichiers HTML détaillant les procédés et résultats des mesures, de photographies de l'état actuel à petite échelle, et d'images de synthèse fixes localisant les désordres à plus grande échelle dans l'édifice. En addition à la représentation tridimensionnelle, un plan de la chapelle permet d'accéder aux informations liées aux surfaces analysées. Enfin, une interface CGI classique permet de lire les informations qui, par leur niveau de généralité, ne trouvent pas de support dans les représentations graphiques.
Ce développement entièrement hypermédia s'appuie uniquement sur des formats textuels ouverts: pages HTML, interfaces Javascript, scène VRML. Ceci correspond au souci de portabilité des développements affiché par les auteurs. De fait, le système est entièrement portable, et la quantité raisonnable d'informations à gérer rend le recours à Javascript acceptable en terme de performance. Par ailleurs, aucun mécanisme de requêtes personnalisées n'étant prévu, l'utilisation d'un Système de Gestion de Base de Données n'a pas été envisagé. L'utilisateur d'ENVIART se voir donc fixé un espace restreint de requête sur le modèle, défini par le concepteur, approche que justifie sans doute l'étroitesse du problème abordé.
Un seul concept spatial est utilisé pour interfacer les données: une face polygonale située dans l'espace de l'édifice. Trois types de données sont accessibles: mesures et commentaires (textes, format HTML), photographies (images, format JPG) et images de synthèse fixes (images JPG).

Les conclusions que formule Z.Wikłacz à l'issue de cette expérience sont à la fois simples et riches d'enseignement :

- L'utilisation d'une maquette tridimensionnelle interactive comme interface d'accès à des données

[195] Projet de recherche et de développement européen (Technologies to protect and rehabilitate European Cultural Heritage) mené par l'Institut für Anorganische und Angewandte Chemie (Universität Hamburg, Allemagne) , l'Institut für Silikatchemie und Archäometrie (Hochschule für angewandte Kunst in Wien, Autriche), l'Institute of Catalysis and Surface Chemistry (Polish Academy of Science, Cracow, Poland), le Koninklijk Instituut voor het Kunstpatrimonium (Bruxelles, Belgique) et le KZA (Konserwacja Zabytków, Cracow, Poland).
[196] Cette plateforme est consultable à l'adresse: http://www.pk.edu.pl/~wiklacz/projects/enviart/enviart.html

sur l'édifice pose le problème du niveau de définition de cette scène, proche des questions d'échelles de représentation déjà abordées.

- Un nombre important d'informations générales sur l'édifice ne sont pas reliables à tel ou tel élément de sa morphologie, que celle-ci soit décrite par de simples faces polygonales comme c'est le cas ici ou de façon plus complexe. La scène 3D ne peut donc pas être vue comme le seul moyen d'accéder aux données rassemblées sur l'édifice.
- En faisant du temps de téléchargement de fichiers VRML un problème concret incontournable, le choix d'un développement pour Internet met en lumière la complémentarité et les rôles particuliers des représentations graphiques (scènes 3D, géométraux, schémas, etc..).
- Les techniques hypermédia s'adaptent bien à la diversité des informations à prendre en compte, tant graphiques que textuelles. Pour autant, peut-on appeler système d'information une plateforme de fait centrée sur la seule présentation finalisée des informations ?

Enfin, nous ne pouvons pas conclure cette présentation du projet ENVIART sans noter que le concept spatial fédérateur utilisé ici est une "face", mot qui provient non pas du vocabulaire architectural mais de la démarche d'étude des dégradations. Nous remarquons donc que là où CART utilisait le point (lié à la problématique de la mesure), ENVIART utilise un concept spatial lui aussi lié au point de vue de l'étude.


*Figure 101 : Le projet ENVIART, types de rendus et d'interfaces.*

## *ANNEXE 22 : NOTION DE POINTS DE VUE : LES ECHELLES DE P.BOUDON*

Dans son ouvrage intitulé "sur l'espace architectural", écrit en 1971, Philippe Boudon posait le problème d'une recherche architecturale alors centrée sur la production de solutions, et qualifiait le savoir architectural *de diffus et non cumulatif*. P.Boudon s'attachait donc alors à définir des axiomes de départ, bases d'une *connaissance de l'architecture à partir d'elle-même* [Boudon, 1971, p3]. L'auteur pose en hypothèse l'architecture comme pensée de l'espace, et le concept d'échelle comme différence entre un espace géométrique et un espace architectural. Ce concept d'échelle, dont il fait remonter la première apparition à Viollet Le Duc, il le définit comme une généralisation des acceptions existantes du terme. Nous allons voir comment, partant de cette hypothèse, Philippe Boudon introduit la notion de points de vue retenant telle ou telle caractéristique de l'édifice [Boudon, 1971] [Boudon, 1977].

Voir l'architecture comme pensée de l'espace pose selon l'auteur deux problèmes [Boudon, 1971, p35] :

- *Trouver l'espace abstrait qui conditionne la pensée de l'espace*
- *Elucider les différences entre géométrie et architecture.*

Rappelant que chaque discipline appréhende la réalité construite à travers ses références propres, il positionne ce qu'il nomme *l'espace sensible* comme la projection dans les trois dimensions des espaces abstraits dans lesquels chacune évolue. Ainsi, nous dit-il, si la pensée s'éloigne de la réalité sensible, l'espace reste son support. P.Boudon s'interroge ensuite sur les liens entre géométrie et architecture. Bachelard, écrit-il, place la géométrie en intermédiaire entre le concret et l'abstrait. Cette définition ne suffit cependant pas, note l'auteur, pour qui la différence entre le géométrique et l'architectural ne tient pas à l'abstraction de l'un et au concret de l'autre: la géométrie d'une forme n'est pas la forme. Il faut, écrit encore P.Boudon, "*trouver un concept abstrait qui rende compte de cette part de l'espace architectural qui, à la différence de l'espace géométrique, est concrète* […] [Boudon, 1971, p44]. Une première notion différentiant géométrie et architecture est pour l'auteur *l'échelle*. L'architecte pense une forme (géométrisée) avec sa taille: la conception de cette forme et de sa taille sont en architecture simultanées. Le *parti architectural* , axiome de l'architecte, intention et non hypothèse, est pour P.Boudon la deuxième notion différentiant géométrie et architecture. Enfin, l'auteur décrit pour chaque étape de la démarche de conception architecturale une géométrie spécifique: l'esquisse ("*géométrie des figures en caoutchouc*"), espace perçu (angles de vues, perspectives), projet (géométrie descriptive) , maquette (objet "*réduit*"). Ce faisant, il introduit l'idée que ces géométries successives au travers desquelles l'architecte formalise son intention dans un processus par essais-erreurs correspondent à des échelles différentes du projet. Il situe donc l'échelle en amont de la géométrie.

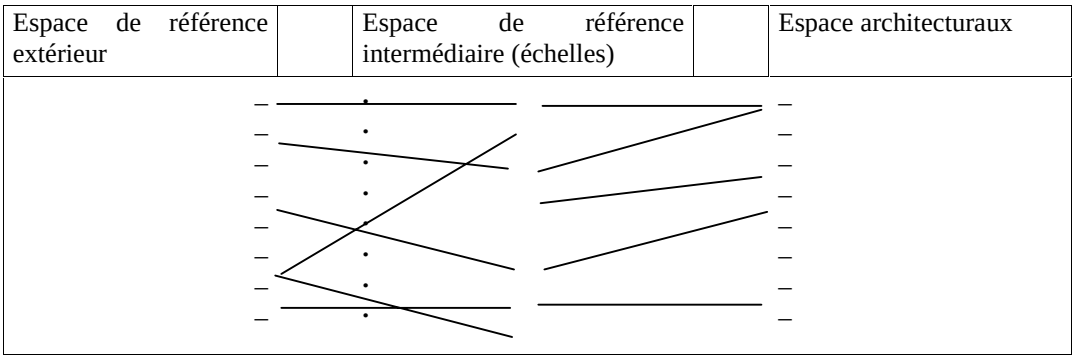
Ayant défini un *espace abstrait qui conditionne la pensée de l'espace* et différencié *géométrie et architecture*, P.Boudon tente de trouver une définition du concept d'échelle qu'il défend. L'échelle, écrit-il, est le rapport d'une partie d'un espace à une partie d'un autre espace, là où la proportion était rapport d'une partie d'un espace à une autre partie du même espace : c'est donc "*la règle de passage d'un espace à un autre*"[Boudon, 1971, p61]. Il précise plus loin son idée : "*c'est par l'échelle que sont reliés les différents niveaux de lecture et que s'établit entre eux une continuité dans l'espace architectural vrai*". L'auteur définit un concept d'échelle qui serait une mesure résultante de l'ensemble des échelles de mesure de l'édifice (esthétique, technique, etc.) et qui ferait du projet architectural un travail de *mise en congruence* de ces échelles. Il peut alors considérer le bâti comme une commune mesure des perceptions que l'on en a, autrement dit considérer la forme architecturale construite comme apte à être attachée à un ensemble de points de vues.

Dans un ouvrage postérieur à celui que nous venons de citer, Philippe Boudon applique son outil d'analyse à un cas concret : la ville nouvelle de Richelieu (Indre), fondée en 1631 [Boudon, 1977]. Cette mise à l'épreuve dans le domaine du patrimoine architectural permet à l'auteur de dresser une première liste d'échelles (ou points de vues) traduisant, écrit-il, la polysémie du concept d'échelle qu'il entend défendre. Il introduit cette liste d'échelles en fournissant une précision importante : on ne peut parler d'échelle que si celle-ci est déterminante dans le processus de conception architecturale. Autrement dit, le point de vue de l'auteur du projet, les préoccupations qui ont déterminé ses choix, sont pertinents alors que le point de vue a posteriori de l'analyste peut lui ne pas l'être. Il nous livre cet exemple : "*Toute architecture coûte une certaine somme d'argent, mais si l'architecte se propose pour faire des économies de chauffage, de réduire le linéaire de façade par lequel s'effectuent les déperditions calorifiques, alors il y a plus que la présence passive d'une référence économique, il y a utilisation de l'espace de référence économique comme échelle*

*de conception*" [Boudon, 1977, p11]. Nous citons ici les catégories d'échelles qu'isole l'auteur en appliquant ses principes au cas de la ville de Richelieu [Boudon, 1977, p26] :

- Celles qui renvoient à l'espace bâti (géographique, parcellaire, de voisinage, etc.)
- Celles qui renvoient l'édifice à un référent extérieur (symbolique, technique, dimensionnelle, etc.)
- Celles qui renvoient l'édifice à sa représentation (géométrique, cartographique, etc.)
- Celles qui relèvent de la démarche de mise à l'échelle du projet (au sens qu'en donne Viollet Le Duc)

P.Boudon pose l'échelle comme outil permettant de renseigner l'édifice, lui donnant un sens qui se rapproche des préoccupations du travail que je propose. Positionnant l'échelle en intermédiaire entre l'édifice (l'espace bâti) et des espaces de références extérieurs (voir schéma) , il nous fournit un cadre méthodologique sur lequel appuyer le travail de documentation de l'édifice patrimonial que nous souhaitons aborder.




En effet, P.Boudon conclue son étude du cas de la ville de Richelieu en écrivant que l'échelle mesure l'objet par rapport à une référence extérieure (là où la proportion mesure les parties de l'objet relativement les unes aux autres dans un système clos), "*D'où résulte une multiplicité de points de vue qui retiennent telle ou telle caractéristique référée à un espace de référence […] L'échelle est une modalité de renvoi d'une partie de l'espace architectural à une espace de référence*" [Boudon, 1977, p139].

ANNEXE 23 : SIMULATION D'HYPOTHESES

**Restitutions de l'église St Géréon**

Le sujet du travail présenté ici est l'église St Gereon (Kościół Św Gereona) bâtie sur le site de Wawel et dont ne subsistent aujourd'hui que les fondations et quelques éléments réemployés.

La reconstruction proposée s'appuie sur le relevé de ces fondations et sur les hypothèses émises par l'équipe d'archéologues et d'architectes en charge du site. L'objectif affiché dans ce travail est de proposer une restitution "grand public" de l'édifice. La plate-forme logicielle utilisée est assez courante puisqu'il s'agit du logiciel de DAO Autocad et de son module de rendu 3D Studio. L'édifice est modélisé en faces sur chacune desquelles des textures particulières, travaillées dans un logiciel de traitement d'images, sont appliquées. Les auteurs ont évalué le temps de calcul d'images nécessaire à la réalisation de leur projet à un an et demi, situant bien le poids de leur maquette. Celle-ci regroupe en effet 4000 surfaces indépendantes. Ils se placent ouvertement dans la perspective de la production d'images dites de vulgarisation. Mais dans ce cadre également, ils rappellent qu'au cours d'un tel travail se pose de façon récurrente la question suivante : faut - il lorsqu'un détail d'architecture doit être représenté sans être connu s'appuyer sur des analogies formelles pour en donner une représentation attrayante, ou plutôt refléter par une représentation incomplète l'état actuel des connaissances sur cet élément ? Dans leurs simulations, les auteurs ont choisi la première solution, et proposent d'ailleurs des reconstitutions d'une grande qualité graphique.

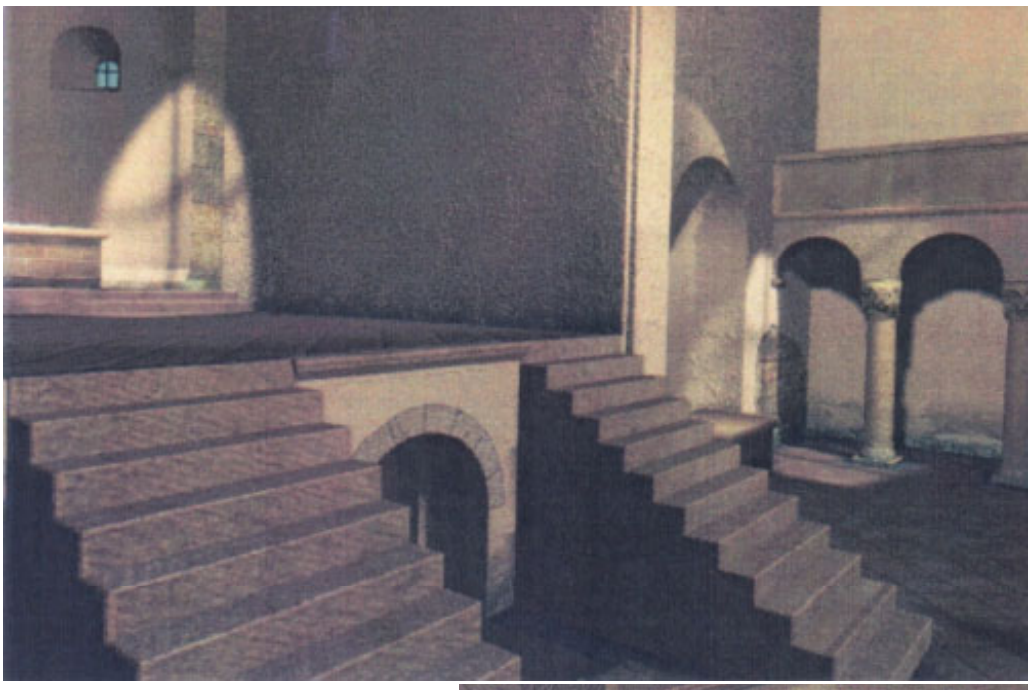
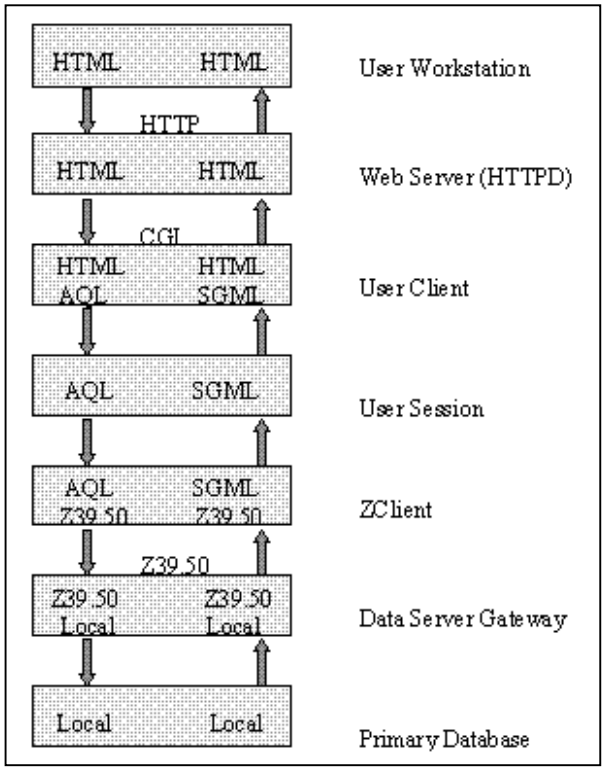


*Figure 102 : Restitution de l'église St Géréon, Autocad / 3DStudio, vues intérieures*

ANNEXE 24 : AQUARELLE, BASE DE DONNÉES CULTURELLE

## ANNEXE 24 : AQUARELLE, BASE DE DONNÉES CULTURELLE




Aquarelle, réseau d'informations sur le patrimoine culturel, est un des grands programmes de recherche et de développement initié dans le cadre de "telematics", initiative conjointe de l'union européenne et du ministère de la culture français conjointement avec l'INRIA. Projet visant à interconnecter les bases de données culturelles de divers musées et institutions publiques européennes, Aquarelle avait pour objectifs techniques notamment de :
- Développer un système de recherche d'informations permettant de localiser la documentation disponible sur divers serveurs
- Fournir des outils de navigation hypertexte
- Offrir la possibilité d'effectuer des requêtes sur un ou tous les serveurs
- Installer un serveur d'accès à Aquarelle jouant le rôle d'assistant et de gestionnaire des accès

Aquarelle mettait en relation d'une part clients Web standards et d'autres part trois logiciels de bases de données différents, Oracle, Mistral et Basis. L'interconnexion bases de données - clients Web était assurée par deux passerelles construites autour des normes SGML (Standard for General Markup Language) et Z39-50 (langage commun entre bases documentaires), dont le schéma de fonctionnement est présenté ci-contre.

L'interface utilisateur pour gérer les accès aux données était construite autour des navigateurs Web standards et permettait de formuler une requête sous forme de conditions ET / OU. Dans les conclusions de leur rapport final, les auteurs du projet insistent sur deux points qui font écho dans le cadre du travail que je présente :

- "*Pour autoriser une recherche précise d'information celle ci doit être décrite avec précision, et à la fois le modèle de données et le vocabulaire utilisés pour cette description doivent être connus du système de requête*". Autrement dit, la gestion de données pose un problème d'identification des concepts utilisés dans la définition et la classification des éléments de connaissances répertoriés.
- "*L'héritage culturel est trop divers pour que l'on puisse prétendre définir un schéma unique pour l'ensemble de ses composantes*". Autrement dit, le vocabulaire et les concepts utilisés dans chaque composante appellent un travail de définition spécifique.

An évoquant également les problèmes liés aux traductions dans les diverses langues de leur projet des concepts utilisés dans Aquarelle, les auteurs rappellent qu'en matière de patrimoine le vocabulaire n'est pas neutre. Partant de ce constat, nous devons considérer les interfaces textuelles ne permettent sans doute pas d'interroger exhaustivement et sans ambiguïté le jeu de données à présenter.
Pour plus d'informations sur ce projet, on peut se reporter à [Dalbera, 1997] ou aux sites suivant :
http://aqua.inria.fr/Aquarelle/Public/EN/project.html
http://aqua.inria.fr/Aquarelle/Public/EN/final-report.html

Nous renvoyons pour des applications plus classiques des Systèmes de Gestion de Bases de Données au patrimoine architectural par exemple à [Whiting et al, 1997], [Camara et al, 1997], ou [Ioannidis et al, 1999], déjà cités précédemment, et à la nombreuse littérature sur ce sujet.

# ANNEXE 25 : PRINCIPE DE REPRESENTATION DES DONNEES DANS LES SGBDR ET LES SGBDOO

## MODÈLE RELATIONNEL

Une base de données, collection d'informations cohérentes, permanentes et le plus souvent accessibles à plusieurs utilisateurs simultanément, est structurée par le biais d'un schéma décrit selon un modèle de données.
Les Systèmes de Gestion de Bases de Données sont des applications logicielles supportant la définition de bases de données, et assurant la cohérence, la persistance, le partage, la fiabilité et la sécurité des données
Les Systèmes de Gestion de Bases de Données relationnels supportent des bases de données décrites selon le modèle relationnel, modèle de données défini par Codd en 1970. Les concepts de base du modèle relationnel sont la relation et le domaine.
La relation (ou table) est définie par un schéma de relation ou intention. Une clé doit être associée à chaque schéma de relation. Le contenu ou extension d'une relation est un ensemble de tuples (lignes structurées comme des séquences de valeurs d'attributs). La clé est composée d'un ou plusieurs attributs d'un schéma de relation dont la valeur permet de désigner au plus un tuple de l'extension de la relation.
Le domaine est un ensemble de valeurs défini de manière intentionnelle ou extensionnelle. Le rôle que joue un domaine dans une relation est identifié par l'attribut. Dans les systèmes commerciaux, la notion de domaine est moins riche, puisque le concepteur d'un schéma relationnel ne peut définir ses propres domaines (il doit utiliser des domaines prédéfinis).
La partie manipulation du modèle relationnel est l'algèbre relationnelle, modèle formel permettant d'exprimer et de calculer les requêtes sur les relations. L'algèbre relationnelle est constituée d'un ensemble d'opérateurs algébriques dont les plus utilisés sont la sélection, la jointure, la projection et les opérateurs ensemblistes.
La construction d'une base de données s'appuie sur trois niveaux de description des données définis dans l'architecture ANSI/SPARC [Ducournau et al, 1998, p132]:

- Le schéma externe qui correspond au niveau le plus proche des utilisateurs. Dans les SGBD relationnels il s'exprime par des vues relationnelles, relations virtuelles calculées à chaque fois que l'on accéde à la base. Une vue est définie par une requête d'interrogation en SQL (Structured Query Language).
- Le schéma conceptuel qui correspond au niveau intermédiaire s'exprime avec le modèle Entité-Association dans la phase de conception puis avec un modèle informatique de haut niveau comme le modèle relationnel. Il est constitué d'un ensemble de schémas de relation. Il représente la sémantique des données indépendamment de toute implémentation. Pour élaborer ce schéma le concepteur peut s'appuyer sur une méthode d'analyse et de conception telles que déjà évoquées à la section précédente.
- Le schéma physique correspond au niveau le plus proche des disques. Il comprend entre autres la définition des index.

L'accès aux données d'une base se fait par le biais de requêtes, expressions d'un langage de définition de données ou d'un langage de manipulation de données. Le LDD (langage de description de données) permet de définir et de modifier la partie intention d'une base de données, autrement dit son schéma de données, ainsi que les utilisateurs et leurs droits d'accès. Le LMD (langage de manipulation de données) permet de manipuler la partie extension d'une base de données (interrogation, suppression, ajout et mise à jour d'informations).
Les SGBD relationnels supportent le langage SQL incluant à la fois un langage de définition de données et un langage de manipulation de données.
Les données stockées dans un SGBD relationnel sont modélisées comme si elles étaient stockées dans des tableaux nommés dont le nombre de colonnes et de lignes peut varier. Chaque colonne de ces tableaux contient des données d'un type et chaque ligne

correspond à un enregistrement. Chaque donnée est donc stockée physiquement en fonction de son type. Les types de données disponibles sont limités à ceux que le système prévoit, types dont la norme ANSI établit une liste que décrit exhaustivement [Loomis, 1995, p32]. Chaque colonne est identifiée par un nom (attribut). Les SGBD relationnels assurent un contrôle de type sur chaque colonne. D'autres contraintes peuvent être écrites (clés primaires ou clés étrangères par exemple) pour ajouter au schéma de données des éléments de sémantique.
Le modèle de données des SGBD relationnels est très simple et donc facile à comprendre pour les utilisateurs. Il repose sur une base formellement définie qui a permis à la fois de définir des méthodes de conception de schémas et des langages de manipulation standardisés. Le succès de ces systèmes s'explique donc aisément. Nous rapportons dans la partie principale du document quelles en sont les principales limitations.

*MODÈLE OBJET*
Nous allons ici décrire en quoi les Systèmes de Gestion de Bases de Données (Orientées) Objets sont des systèmes à objets, puis en quoi ils sont des Systèmes de Gestion de Bases de Données. Notons dès à présent que les bases de données orientées objets peuvent être caractérisées par cinq points essentiels :
- Leur modèle de données doit permettre de représenter des structures de données complexes.
- Les données et les traitements ne sont plus séparés puisque les méthodes fontt partie de la déclaration des objets
- Les bases de données orientées objets supportent la notion d'héritage.
- Tout objet possède une identité qui permet de le distinguer d'un autre objet.
- Il n'y a plus d'incompatibilité entre le langage de programmation et le langage de manipulation des données.

Si de de nombreux produits commerciaux existent aujourd'hui dans le domaine des Systèmes de Gestion de Bases de Données (Orientées) Objets[197], Il n'existe pas de concensus sur notamment les langages de manipulation des données utilisés.

Systèmes à objets
Le concept central d'un Système de Gestion de Bases de Données Orientées Objets est bien sûr celui d'objet, déjà évoqué plus haut. L'objet possède une identité, un état (valeurs de ces attributs) et un comportement. L'identité de l'objet , nommé object identifier ou oid [Ducournau et al, 1998, p138], permet d'assurer l'unicité de l'objet indépendamment de la valeur (ou de l'état) de l'objet. C'est une première différence par rapport aux Systèmes de Gestion de Bases de Données relationnels dans lesquels le mécanisme de clés permettant d'atteindre le même objectif est basé sur la valeur de la clé. La notion d'identifiant ou de référence de l'objet permet de définir des opérateurs de comparaisons ou de copie d'instances d'une classe :
- Opérateur d'identité entre deux objets qui évalue l'oid des objets comparés.
- Opérateur d'égalité de surface qui évalue l'égalité des valeurs de tous les attributs des objets comparés.
- Opérateur d'égalité de profondeur qui vérifie la présence de graphes de composition isomorphes pour chacun de leurs attributs et l'égalité de valeurs des attributs terminaux.

[Loomis, 1995, p95] utilise indifféremment les termes de classe et de type pour définir ce que l'auteur nomme un patron d'objets. [Ducournau et al, 1998, p140] rappellent que classes et type peuvent recouvrir des notions différentes dans le cas de systèmes dérivant de Langages de Programmation Orientés Objets tels que C++. En effet, comme on l'a vu dans le chapitre consacré aux Langages de Programmation Orientés Objets, si dans le cas de systèmes "tout objet" les classes sont des objets décrivant des types, dans les autres cas

[197] Voir notamment Objectstore (ObjectDesign, http://www.objectdesign.com/) ou O2 (O2 technology, http://www.O2tech.fr)

classes et types, objets et valeurs sont différenciés. Les auteurs proposent une définition de ces termes qu'ils considèrent admis dans la communauté base de données :

- Le mot type correspond à la notion de type utilisé en C par exemple, le type est instancié par des valeurs qui n'ont pas une identité au sens de celle des objets.
- La classe est décrite en intension (type construit + opérations spécifiques) et sert à instancier des objets. Elle a une composante extensionnelle : tous les objets instances de la classe constituent son extension.

La description d'une classe comporte un ensemble d'attributs auxquels sont associés des domaines de définition. Les Systèmes de Gestion de Bases de Données Objets fournissent des constructeurs d'ensembles ou de collections complexes (n-uplet, ensemble, multi-ensemble, liste) qui permettent la définition de classes collections polymorphes. La description d'une classe comporte également la définition d'un ensemble de méthodes gérant le comportement de l'objet. Comme dans le cas des Langages de Programmation Orientés Objets, les objets construits dans un Système de Gestion de Bases de Données Objets regroupent dans une même structure programmes et données. Ce principe d'encapsulation n'est pas toujours mis en œuvre de façon systématique dans les Systèmes de Gestion de Bases de Données Objets, [Loomis, 1995, p88] en donne les règles de mises en œuvre. [Ducournau et al , 1998, p142] l'explique en notant que l'encapsulation stricte de données peut être contraire aux nécessités des LMD déclaratifs de type SQL, et détaillent divers niveaux d'encapsulation selon les systèmes.
La notion de schéma conceptuel de données introduite plus haut est constituée dans le cadre des Systèmes de Gestion de Bases de Données Objets par l'ensemble des classes relatives au domaine modélisé, complété selon le cas par un ensemble de types. On pourra se reporter à [Lammari et al, 1999] pour une discussion approfondie sur l'optimisation de schémas conceptuels de Bases de Données Objets.
Les relations entre classes dans un Système de Gestion de Bases de Données Objets sont sans surprise celles dont nous avons déjà parlé plus haut: composition (agrégation) et spécialisation .
Dans le premier cas, l'objet composite crée par agrégation pose dans le cadre des Systèmes de Gestion de Bases de Données Objets des problèmes de contraintes d'intégrité pour lesquels les systèmes actuels apportent des solutions diverses. [Ducournau et al , 1998, p143] notent en particulier ces deux points:

- Un objet composant peut être partagé ou non entre plusieurs objets de la même classe
- Un objet composant peut être existentiellement dépendant ou non de l'objet composite (notion d'intégrité référentielle)

En ce qui concerne les relations de spécialisation (ou généralisation / spécialisation), le mécanisme d'héritage proposé par les Systèmes de Gestion de Bases de Données Objets est le même que celui proposé par les Langages de Programmation Orientés Objets .
De la même façon , la notion de surcharge présente dans les Systèmes de Gestion de Bases de Données Objets est la même que celle présente dans les Langages de Programmation Orientés Objets. On pourra se reporter au chapitre 6 de [Loomis, 1995] pour une présentation plus approfondie du modèle objet dans les Systèmes de Gestion de Bases de Données Objets.

Systèmes de Gestion de Bases de Données

Les Systèmes de Gestion de Bases de Données (Orientées) Objets doivent comme leurs équivalents relationnels assurer gestion et accès concurrents à de larges quantités de données. La persistance dans un Systèmes de Gestion de Bases de Données Objets concerne à la fois le schéma de données lui-même (ensemble des classes et type) et les objets et valeurs crées au cours de la vie du /des programme/s. [Loomis, 1995, p47] note à ce propos que la plupart des Systèmes de Gestion de Bases de Données Objets assurent la persistance du seul état des objets créés. [Ducournau et al,1998, p145] donne la définition suivante du modèle de persistance recherché :

- Transparent: données volatiles ou persistantes doivent être manipulées de la même façon.
- Orthogonalité avec type / Classe: la persistance s'applique à tout objet ou valeur.
- Intégrité: les références des objets ou valeurs complexes restent valides après toute opération sur la base.

Concrètement, les solutions offertes au concepteur diffèrent selon les systèmes utilisés. Une première solution consiste à gérer un double arbre d'héritage dans lequel la persistance est associée à une des classes sœurs au niveau du schéma. Une hiérarchie permet alors de manipuler des objets persistants, une autre des objets volatiles, démarche en contradicton avec le principe d'orthogonalité. une seconde solution consiste à réaliser de façon explicite la persistance lors de l'instanciation d'un objet. Dans ce cas c'est au programmeur de prévoir explicitement le statut de l'instance créée. Enfin, une troisième solution consiste à réaliser une persistance dynamique, c'est à dire à permettre à une instance de changer de statut au cours de la vie de l'application qui la manipule. Dans ce cas la persistance est obtenue par l'attachement de l'instance à une racine de persistance, attachement qui reste du ressort du programmeur.

## *ANNEXE 26 : LA REPRESENTATION GEOMETRIQUE*

### *CSG*

L'arbre de construction est aujourd'hui une représentation intégrée en standard dans la plupart des logiciels dits de CAO, bien que cela soit finalement assez récent. L'arbre de construction combine des solides au moyen d'opérations ensemblistes [Péroche et al,1988, p113] (intersections, unions, etc..). Ce principe étant récursif, l'ensemble des opérations successives ayant abouti à la construction d'un solide complexe est représentable sous la forme d'un arbre dont les nœuds sont des opérations booléennes et les feuilles des objets primitifs. L'arbre de construction est donc une description des opérations construisant un objet, opérations à faire effectuer par un outil adéquat. [Péroche et al,1988, p117] rappellent à ce propos que les algorithmes de visualisation acceptant en entrée directement un arbre de construction sont assez rares. Parmi ceux -ci on trouve l'algorithme de lancer de rayon sur lequel nous revenons plus loin. Partant d'un principe simple, l'arbre de construction se prête bien à une interaction au moyen d'un langage, point à partir duquel a été proposé l'outil POV-Ray. En effet, cet outil permet de décrire dans un langage simple nœuds et feuilles de l'arbre de construction d'un objet complexe. Il permet par ailleurs de paramétriser cette description, et par conséquent d'écrire des méthodes génériques de représentation pour un jeu d'objets comme par exemple un corpus architectural. Toutefois, on comprend aisément les limitations de ce modèle en rappelant que le jeu d'objets primitifs à partir duquel se constitue l'arbre est limité. Les surfaces libres ne sont pas des solides et donc ne peuvent être utilisées comme primitives dans un arbre de construction. Dans le cas qui nous occupe, on imagine bien les difficultés à représenter des éléments de statuaire sous forme d'arbre de construction. Nous aurons l'occasion de revenir sur ce point en parlant des méthodes de représentation des éléments de corpus dans la partie projet de ce document.

La plupart des outils de modélisation volumique proposent aujourd'hui des représentations hétérogènes intégrant arbres de construction et surfaces libres. Notons enfin pour conclure que l'arbre de construction décrit des opérations pour construire un objet, opérations qui peuvent être lourdes à effectuer. Ceci explique qu'un logiciel très récent d'imagerie centré sur l'animation de scènes[198] (et donc mettant la priorité sur une représentation économique des volumes) ne propose pas de CSG. Ceci explique également qu'en VRML les opérations booléennes n'existent pas, l'objectif de ce format étant avant tout de permettre à un utilisateur distant (au sens du réseau Internet) de se déplacer à l'intérieur d'une scène tridimensionnelle, scène qui par voie de conséquence se doit d'être aussi légère que possible en terme de temps de calcul d'affichage. On peut se reporter pour une application du logiciel POV-Ray à notre domaine à [Potier et al, 1998], qui l'utilisent comme outil de formulation d'hypothèses archéologiques en privilégiant quatre directions :

- Le paramétrage des dimensions.
- La hiérarchie et le liens entre variables.
- Les niveaux de modélisation.
- Les modèles-types.

### *BREP*

Dans les représentations par frontières, ou BREP, la frontière de l'objet est explicitement décrite [Péroche et al, 1988, p117]. Premières représentations utilisées en CAO, elles le sont toujours car elles sont plus légéres que les représentations par arbre de construction et bien adaptées à la plupart des algorithmes d'affichage qui supposent connues les surfaces des objets. La frontière d'un solide comprend des faces, des arêtes et des sommets. Une BREP les mémorisent ainsi que les relations topologiques d'incidence, de contiguïté et d'inclusion entre ces éléments [Péroche et al, 1998, p118]. Les auteurs rappellent que pour assurer la validité d'une Brep plusieurs contraintes doivent être prises en compte afin de définir un intérieur et un extérieur, et en cite pour exemple deux :

- Les faces ne doivent pas s'intersecter ailleurs que sur les arêtes définies par la Brep.
- Les faces doivent séparer intérieur et extérieur du solide, interdisant les faces dites pendantes.

Les représentations par frontières sont rendondantes puisqu'elles enregistrent pour un même objet ses faces, ses arêtes et ses sommets. Les informations enregistrées peuvent dès lors être contradictoires et donner lieu à erreurs lors de la visualisation. [Péroche et al, 1998, p118] notent que les imprécisions numériques sont

[198] Logiciel MAYA d' Alias|Wavefront (voir http://www.aw.sgi.com/pages/home/index.html) logiciel d'imagerie utilisé dans les travaux de restitution archéologique du MAP-GAMSAU cités précédemment.

traitées de façon diverses selon les plateformes utilisées. La phase de visualisation ou d'affichage d'une Brep consiste en fait à facettiser l'objet modélisé en utilisant diverses méthodes pour lesquelles je renvoie à [Péroche et al, 1998, p119]. L'utilisation des BREP pose avant tout un problème d'intégrité des représentations construites. Sur ce sujet on peut par exemple se reporter au site de l'International Journal of Computational Geometry and Applications (IJCGA)[199] sur lequel on trouvera un ensemble de ressources récentes.
On notera enfin qu'il existe des conversions entre modèles, et notammment l'évaluation des frontières d'un arbre de construction qui vise à convertir une représentation par arbre de construction en représentation par frontières. On trouvera sur ce sujet de nombreuses références dans [Péroche et al, 1988, p124].

[199] http://www.wspc.com/journals/ijcga/ijcga.html

# ANNEXE 27 : MODELES DE COULEUR ET D'ECLAIREMENT, LANCER DE RAYON

## MODÈLES DE COULEUR ET D'ÉCLAIREMENT

[Péroche et al, 1988, p163] proposent une rapide description des cinq modèles de couleurs les plus courants en les séparant en deux groupes: modèles basés sur la séparation trichromatique Rouge Vert Bleu (primitives additives), modèles basés sur la perception subjective. Dans le premier groupe, on trouve les modèles :

- RVB (Rouge, Vert, Bleu), qui utilise un système de coordonnées cartésiennes et définit un cube unité dont chaque axe représente une couleur primitive (additive). C'est le modèle le plus courant, adopté notamment par VRML et POV-ray.
- CMJ (Cyan, Magenta, Jaune), identique au précédent mais dont les axes sont ici Cyan, Magenta et Jaune.
- YIQ, standard adopté pour les signaux de télévision et résultant d'une transformation sur le cube RVB.

Dans le second groupe, on trouve les modèles :

- TSL (Teinte, Saturation, Luminance), dans lequel la teinte correspond à la notion permettant à l'œil de distinguer les couleurs, la saturation qui permet de mesurer la quantité de couleur par rapport au blanc, et la luminance qui caractérise l'intensité d'excitation visuelle. Ce modèle est défini par un espace en forme d'hexacône dont l'axe de symétrie est la luminance. La teinte est mesurée par l'angle nécessaire pour obtenir la teinte souhaitée autour de l'axe vertical et la saturation par la distance axe / côté.
- TSI (Teinte, Saturation, Intensité), sous-espace du précédent en forme de double cône dont les base sont le noir et le blanc.

[Péroche et al, 1988, p168] mettent en garde sur les résultats des interpolations de couleurs nécessaires dans le cas par exemple de calculs liés à la transparence. En effet, ces interpolations, notent-ils, ne donnent pas toujours le même résultat selon le modèle choisi.

En matière d'éclairement, il faut distinguer la définition de sources de lumière et celle des réactions des objets à cette lumière. Les sources de lumière peuvent être de type lumière ambiante (éclairage uniforme de la scène), lumière directionnelle (sources supposées à l'infini générant des rayons parallèles à une direction donnée), et sources ponctuelles (rayonnant soit dans toutes les directions soit selon une certaine loi).

Les objets réagissent à cette lumière à la fois en réflexion (avec production de reflets, appelée réflexion spéculaire, ou en réflexion diffuse) et en transmision (spéculaire dans le cas d'objets transparents ou diffuse dans le cas d'objets translucides).

Les modèles d'éclairement sont présentés par [Péroche et al, 1988, p171] en deux catégories: modèles empiriques et modèles analytiques. Dans le premier cas, le modèle distingue trois composantes (ambiante, diffuse et spéculaire) dont l'addition doit satisfaire le principe fondamental de conservation de l'énergie. Je renvoie à leur ouvrage pour plus de détail sur ces composantes. Dans le second cas, les auteurs présentent les principes de la radiosité, modèle issu de méthodes utilisées en ingéniérie thermique, et fondé sur l'idée d'équilibre d'énergie. le modèle de radiosité est un modèle de transfert d'énergie. Pour chaque surface de la scène la quantité d'énergie dépensée est la somme de l'énergie consommée par cette surface et de celle qu'elle réfléchit. La quantité d'énergie qu'elle réfléchit est caractérisée par le produit de la quantité d'énergie incidente à la surface et d'une constante de réflectivité. La radiosité $B_j$ d'une surface j est la quantité d'énergie dépensée, à savoir :

$$B_j = p_j H_j + E_j$$

Où $p_j$ est la réflectivité de la surface j,
$H_j$ l'énergie incidente sur la surface j,
$E_j$ L'énergie transmise par la surface j

Il est à noter qu'ici la position de l'observateur n'influe pas. Toutefois, le temps de calcul reste en radiosité assez coûteux.
Dernier point à aborder ici, le calcul de l'ombrage dans une scène peut lui aussi être effectué en fonction de plusieurs modèles s'appuyant sur la détermination des normales aux surfaces éclairées, et dont les deux principaux sont les modèles de Gouraud et de Phong [Péroche et al, 1988, pp179-181]. Reste alors à effectuer le calcul des ombres portées. Il faut cependant noter que de nombreuses plateformes de visualisation (VRML par exemple) se contentent d'implémenter un modèle d'ombrage sans traiter les ombres portées. Nous ne citerons ici que deux méthodes, celle du double tampon de profondeur et celle du lancer de rayon; et encore ne les citons-nous que pour mieux les différencier. On trouvera dans la nombreuse littérature sur le sujet toutes les précisions qui n'ont pas lieu de figurer dans le travail que je présente.
La méthode du double tampon de profondeur (voir à ce sujet le chapitre 6 de [Péroche et al ,1988]) consiste à calculer deux images de la scène, une avec la source de lumière comme point de vue et l'autre avec l'œil de l'observateur comme point de vue. On dispose de deux tampons de profondeur, un pour l'observateur et un pour l'éclairage. Pour tout point visible par l'observateur on peut alors comparer ses coordonnées avec celles du tampon de profondeur éclairage.
La méthode de lancer de rayon sur laquelle nous revenons plus loin consiste, à partir des points intersections les plus proches de l'observateur sur les rayons primaires (issus de l'œil de l'observateur), à envoyer des rayons secondaires vers les sources lumineuses.

*LANCER DE RAYON*

Comme nous venons de l'évoquer, le principe du lancer de rayon est de suivre les rayons lumineux dans le sens inverse de la propagation de la lumière, depuis l'œil de l'observateur vers les sources lumineuses, via les surfaces. Depuis l'œil de l'observateur, un rayon est envoyé pour chaque pixel de l'écran. A la première intersection avec une surface (un objet dans la scène), le rayon sera réfléchi ou transmis si la surface est transparente. Des rayons secondaires seront envoyés vers les sources lumineuses afin de déterminer si le point d'impact est à l'ombre ou non. Contrairement au tampon de profondeur (Z-buffer) par exemple, le lancer de rayon permet de modéliser la réflexion ou la transmission des surfaces, apportant un bien meilleur réalisme. On l'a vu, le lancer de rayons peut traiter directement des scènes générées en CSG. L'algorithme de lancer de rayon prend en compte de nombreux effets optiques contribuant au réalisme d'une scène: réflexions multiples, transparences, réfractions, sources lumineuses multiples par exemple. Les coûts de calcul du lancer de rayon en font cependant une méthode souvent réservée à un usage assez étroit, celui de l'imagerie réaliste. Le chapitre 7 de [Péroche et al, 1988] est consacré aux différents problèmes que soulève cette méthode et à ses perpectives d'évolution. Il est à noter que la plupart des outils d'imagerie photo-réaliste, commerciaux ou gratuits, utilisent plusieurs algorithmes, et en particulier lancer de rayon et radiosité[200]. Sur le lancer de rayon, modèle d'éclairement dit local, comme sur le modèle d'éclairement global de radiosité de nombreux travaux de recherche ou peuvent être consultés[201], notre sujet s'en éloigne par trop pour les citer dans le détail.

[200] Persistence Of Vision (POV): http://www.povray.org
RenderPark:http://www.cs.kuleuven.ac.be/cwis/research/graphics/RENDERPARK/
Rayshade : http://graphics.stanford.edu/~cek/rayshade/
Radiance: http://radsite.lbl.gov/radiance/HOME.html

[201] Voir par exemple le projet SIAMES, Synthèse d'Image, Animation, Modélisation et Simulation (http://www.irisa.fr/siames/pub/L_EQUIPE/siames.htm), l'article en ligne " Synthèse d'images et radiosité hiérarchique " de N. Holzschuch (http://www-imagis.imag.fr/Membres/Nicolas.Holzschuch/Publi/mago/mago.html) et plus généralement les publications de l'équipe iMAGIS (http://www-imagis.imag.fr/Publications/index.fr.html) équipe du laboratoire GRAVIR de l'IMAG.

## ANNEXE 28 : VISION STEREOSCOPIQUE

Nous avons déjà abordé la question de la vision stéréoscopique à travers son application dans le cadre de la phase de restitution du relevé photogrammétrique puisque celle-ci s'appuie sur la capacité du restituteur à retrouver des points homologues sur deux photographies d'un objet par le biais d'un équipement appelé restituteur. Nous revenons ici sur cette question dans un tout autre contexte: celui de la visualisation en 3D simulée d'une maquette numérique. Ce sujet est un peu anecdotique dans le cadre d'un travail de recherche comme le notre mais nous considérions qu'une technique de visualisation de ce type peut contribuer à la lisibilité d'une restitution architecturale en images de synthèse. Nous considérons par conséquent qu'il n'est pas inutile de mentionner ici quelques techniques adaptées à la simulation de la perception en relief, et en particulier celles que nous avons expérimentées. Nous ne nous interessons donc ici au thème de la vision stéréoscopique humaine pour répondre à la question : comment capturer ou calculer deux images afin de simuler la perception du relief dans un dispositif de projection ?

### *HISTORIQUE ET PRINCIPE DE BASE*

Les deux images distinctes qui se forment dans chaque œil d'un individu observant une scène sont interprétées par son cerveau et restituées sous forme d'image tridimensionnelle unique de façon naturelle. Ce phénomène physiologique est à la base de la photo en relief, autrefois observée avec un stéréoscope, aujourd'hui pouvant faire l'objet de projections en relief par le biais de dispositifs sur lesquels je reviendrai plus loin. Historiquement, les premiers couples stéréophotographiques remontent à 1841 avec les expériences de WHEATSTONE et TALBOT en Angleterre. Un premier appareil à deux objectifs, dédié à la stéréophotographie, est réalisé par David BREWSTER vers 1845. Dès la seconde moitié du dix-neuvième siècle, de nombreux appareils de prises de vues stéréoscopiques furent commercialisés avec succès, comme en témoigne les images ci-contre, datées de 1860 environ, et sur lesquelles un photographe professionnel polonais a fixé grâce à un appareil de ce type d'une part l'état d'un édifice transformé largement entre 1870 et 1872 (Sukiennice) et l'état d'un édifice attenant détruit vers 1866 (Kramy Bogate).

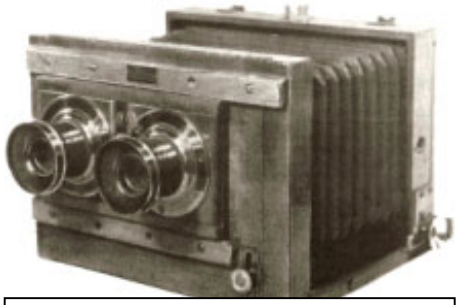


*Figure 103 : Chambre stéréo russe Kant (fin 19ème) in http://www.multimania.com/stereoscopie/appareils%20anciens.htm*

Le principe de la photo en relief est simple: deux photos sont prises dans la même direction, à partir de points de vue légèrement décalés. Concrètement, si des appareils à deux objectifs permettent de prendre des photos en relief de façon aisée, on peut également avec un seul appareil effectuer deux prises de vues à condition de respecter quelques règles simples: un même diaphragme, un même temps de pose et une même mise au point, ou encore le choix de sujets immobiles.

Dès les premières réalisations de photographies en relief, le procédé a été utilisé pour comme source de données pour des archives, et notamment en architecture avec les recueils de données de Meydenbauer. Ces données constituent un fonds documentaire remarquable sur lequel de nombreux travaux de recherche et de restitution peuvent être entrepris. Une expérience de ce type, menée dans le cadre de notre projet à partir des images ci-contre, a permis par exemple de relever grossièrement la hauteur du bâtiment dit "Kramy Bogate" aujourd'hui détruit, hauteur sur laquelle aucune information ne subsistait. Nous renvoyons également par exemple à l'expérience de Frank A. van den Heuvel (Delft University of Technology) à partir d'images originales de Meydenbauer[202] qui réutilise le cliché original de Meydenbauer pour texturer la maquette produite. .Enfin, la méthode a également été choisie par les services français de l'inventaire pour le recensement des édifices du patrimoine architectural national, ouvrant de nombreuses perspectives de réutilisation.

*Figure 104 : Stéréogramme montrant Sukiennice et Kramy Bogate, pris vers 1860 depuis le clocher de la Cathédrale.*

[202] Voir http://www.geo.tudelft.nl/frs/architec/Meydenbauer/

*VISUALISATION DES STÉRÉOGRAMMES*

Une fois la prise de vue menée à bien se pose naturellement le problème de la visualisation des stéréogrammes. Il faut ici distinguer les méthodes permettant à une personne seulement d'observer des stéréogrammes, et celles permettant soit de projeter ces images sur un écran soit de les afficher sur l'écran d'un ordinateur. Dans le premier cas on trouve naturellement les stéréoscopes simples ou complexes permettant d'observer des figures imprimées côte à côte. Suivant le même principe, visualiser un couple d'images stéréoscopiques est ainsi possible en projetant sur deux écrans, principe appliqué pour les casques de réalité virtuelle (HMD, Head Mounted Display). Ce périphérique se compose de deux petits moniteurs LCD montés en face des yeux. Chaque œil reçoit l'information nécessaire à la fusion des deux images, ce qui permet l'interprétation de la profondeur.

Dans le second cas, on trouve essentiellement trois méthodes que nous décrivons sommairement. Nous ne détaillons pas les procédés lenticulaires ou les photos Bonnet pour lesquels nous renvoyons par exemple au site web du CNRS[203], ni les procédés d'autostéréogrammes, tous inadaptés à notre problème.

- Les Anaglyphes : Ce nom a été donné en 1891 par Ducos du Hauron à ce procédé de 1853 qu'il avait perfectionné. Ducos du Hauron avait jusqu'alors travaillé sur la couleur pour la photograhie et notamment déposé un brevet de photographies en couleur dès 1868 (décrit dans "Les Couleurs en photographie: Solution du problème", publié en 1869). Le principe des anaglyphes est d'utiliser des filtres de couleurs complémentaires sur les deux yeux, par exemple rouge sur l'œil gauche, bleu ou "cyan" sur l'œil droit. L'utilisateur doit donc chausser une paire de lunettes spéciales dont les verres sont adaptés aux filtres de couleur de l'image, généralement rouge et bleu. Ce procédé permet de visualiser économiquement des images en relief, soit par projection sur un écran, soit sur un écran d'ordinateur. Le gros inconvénient de cette méthode est de mal restituer les couleurs de la scène photographiée, en particulier les couleurs proches de celles des filtres.

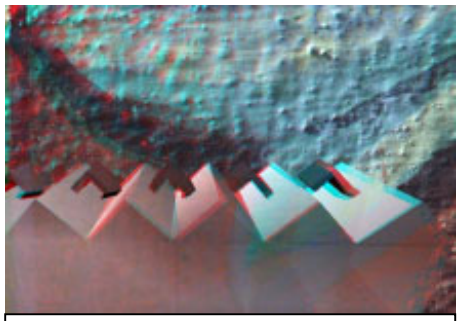


*Figure 105 : anaglyphe (modèle VRML inséré dans un cliché, im auteur)*

- La polarisation : Ce procédé est plus performant que le précédent en terme de restitution des couleurs. La méthode est cependant assez simple également. Les projecteurs sont munis de filtres polariseurs inversés pour les images gauches et droites. L'écran est choisi pour bien préserver la polarisation de la lumière, il s'agit généralement de surfaces métallisées. Chaque spectateur porte alors une paire de lunettes polarisantes dont la polarisation est inversée dans le même sens que celle des filtres des projecteurs. Ce procédé est actuellement le plus performant pour la projection d'images en relief. Par contre, il n'a pas d'applications dans le cadre de l'impression papier, et reste assez lourd à mettre en œuvre en terme de matériel requis.

*Figure 106 : lunettes polarisantes*

- Systèmes numériques: Pour afficher sur le moniteur d'un ordinateur, de façon simultanée, deux images distinctes (une à destination de l'œil gauche, l'autre à destination de l'œil droit), les deux images sont émises en alternance avec une fréquence de balayage suffisamment élévée (>60Hz) pour tromper le cerveau de l'utilisateur. Celui ci doit chausser une paire de lunettes (shutter) synchronisées avec l'affichage de manière à cacher l'image non pertinente. On distingue deux techniques de visualisation différentes, le mode actif et le mode passif. En mode actif, les lunettes se composent d'un obturateur à cristaux liquides commandé par un signal infrarouge de synchronisation avec l'affichage alterné du moniteur. Dans le cas du système passif, on applique à la surface du moniteur un modulateur à cristaux liquides. Il suffit alors à l'observateur de se munir de lunettes à verres polarisants (directions de polarisation différentes de 90°) pour recevoir correctement

[203] Voir http://www.cnrs.fr/CMA/delegation/com/expo/Bonnet/

l'information. Ces systèmes posent un certain nombre de problèmes (scintillation ambiante ou d'affichage, illumination asymétrique par exemple) sur lesquels on trouvera des informations trés détaillées dans [Rolle, 1998].

Dans le cadre de notre projet, les deux première techniques, plus légères et mieux adaptées aux conditions de projection que nécessitait notre travail, ont été expérimentées. La troisième solution reste à explorer. On peut se reporter à [Juhasz et al, 1999] pour un état de l'art des méthodes de visualisation 3D appliquées à l'architecture.

## ANNEXE 29 : POV RAYTRACER, METHODE ET FORMATS

POV-Ray est un logiciel de lancer de rayon gratuit (freeware) disponible pour les plateformes PC, Macintosh et UNIX. Les sources du logiciel sont également accessibles.
POV-Ray est probablement un des logiciels de ce type les plus utilisés aujourd'hui. Il est couplé avec un logiciel de modélisation servant en fait d'interface graphique à l'édition de scripts POV. POV-ray est un moteur de rendu qui prend en entrée un script décrivant la scène et génère l'image de cette scène.
Le script, écrit en ASCII, décrit l'ensemble de la scène de façon assez simple en intégrant notamment les opérations booléennes sur les solides. Le modeleur n'est donc qu'une interface dont beaucoup de programmeurs se passent.

Le script doit préciser :
- Quel type d'objet doit être généré.
- Quels sont les attributs de ses objets (couleur, textures,etc..).
- Où se situe l'observateur.
- Quelles sources de lumière sont présentes dans la scène.

La description des objets peut être explicite en terme de dimensions ou paramétrée, autorisant la construction de bibliothèques d'objets stockés en fichiers indépendants appelés lors du calcul de la scène. De la même façon, les attributs des objets gérant leur apparence peuvent faire l'objet de descriptions stockées indépendamment de la scène dans laquelle ces attributs sont appelés.
C'est la démarche que nous avons adopté dès les débuts de ce projet, et celle que l'on retrouve dans [Potier et al, 1998], ou dans les dernières évolutions du projet REMUS mené au laboratoire MAP-GAMSAU.
En autorisant la description d'une scène au travers d'un script ASCII, POV-ray nous permet en effet de générer des représentations ponctuelles de scènes au travers de méthode d'écriture simple depuis n'importe quel langage de programmation, en l'occurrence pour nous successivement C++, JAVA et Perl.

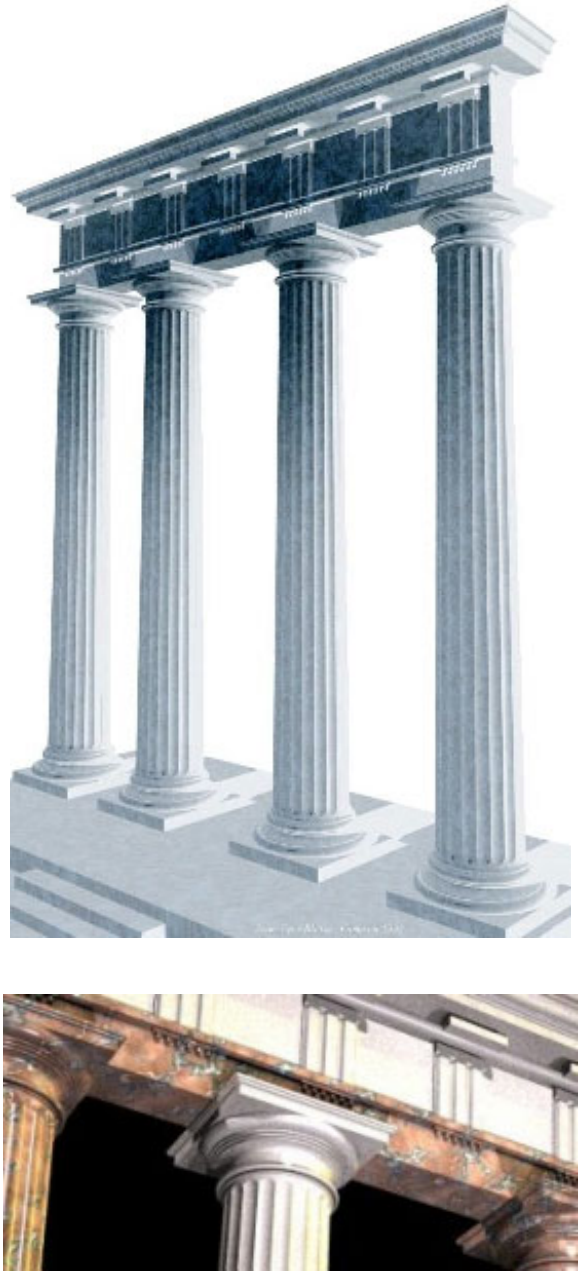


*Figure 107 : Rendus POV à partir des scripts s'appuyant sur la définition du modèle architectural*

*ANNEXE 30 : ASPECTS DU LANGAGE VRML*

**Fichiers VRML :**
Un fichier VRML est un fichier ASCII commençant par la ligne #VRML V2.0 <encoding >. Il contient une section dite *header* (contenant cette première ligne plus des paramètres optionnels), un *scene graph* (nœuds décrivant les objets et leurs propriétés), et peut contenir également des prototypes (nœuds utilisateurs) et des routages d'évènements (en réponse soit à des actions utilisateurs soit à des modifications de l'environnement comme par exemple horloges ou détections de collision

**Géométrie des objets :**
Les nœuds de type géométries ou formes (Shape) associent une définition géométrique de l'objet à une définition de son apparence. Un nœud Shape contient une des géométries possibles (Box, Cone, Cylinder,ElevationGrid, Extrusion, IndexedFaceSet, IndexedLineSet, PointSet, Sphere, Text). Les nœuds géométries peuvent contenir des propriétés comme Coordinate, Color, Normal, ou TextureCoordinate qui en complètent la définition. D'autres propriétés permettent dans le cas des nœuds Extrusion ou IndexedFaceSet de spécifier l'ordre dans lequel sont écrites les faces de l'objet (ordre d'apparence des sommets dans le champ correspondant). VRML n'autorise pas d'opérations booléennes entre objets (seules les surfaces sont modélisées). Les nœuds " Shape" sont positionnés dans l'espace par le biais de transformations ( node Transform) qui appliquent à leurs nœuds enfants (child nodes) une transformation qui s'écrit:

```
Transform {
  Center 0 0 0 (par rapport au référentiel local)
  Translation  0 0 0
  Rotation 0 0 0 0
  Scale  1 1 1
  Children [## portée sur laquelle s'applique la transformation
  ]
}
```

**Apparence, textures et éclairage :**
L'apparence d'un objet est définie à l'intérieur du nœud Shape décrit ci-dessus en spécifiant matériau et texture.
Les champs du nœud Material déterminent la façon dont la lumière se réfléchit sur l'objet :

- Le champ ambientIntensity spécifie combien de lumière ambiante reçue par les sources de lumière sera réfléchie par la surface. La lumière ambiante est omnidirectionnelle and dépend seulement du nombre de sources de lumière. La couleur ambiante est le résultat de calcul de : ambientIntensity × diffuseColor.
- Le champ diffuseColor réfléchit toutes les sources de lumière selon l'angle surface / source de lumière. Plus la surface est touchée directement, plus la réflexion est intense.
- Le champ emissiveColor modélise les effets de néon.
- Les champs specularColor et shininess déterminent les effets de brillance sur les objets. Lorsque l'angle surface / source de lumière est proche de l'angle surface / observateur, la valeur de specularColor est prise en compte dans les calculs des valeurs des couleurs ambiantes et diffuse.
- Le champ transparency spécifie le degré de transparence de l'objet, de 1.0 (complètement transparent) à 0.0 (complètement opaque).

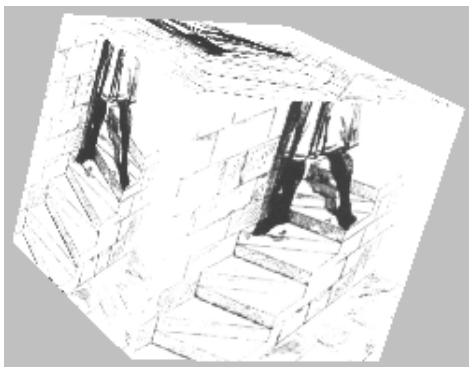
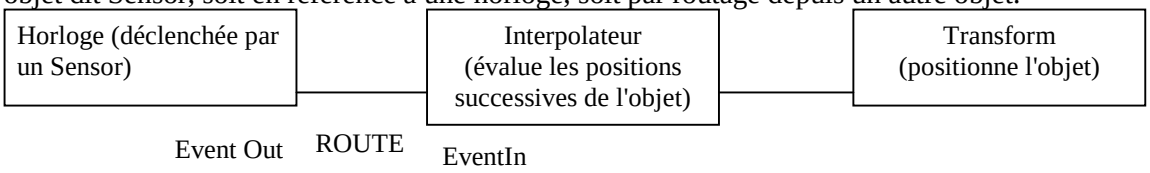


*Figure 108 : Placage d'image sur une primitive de type Box*

La texture peut être de trois types : ImageTexture, PixelTexture et MovieTexture, et fait référence à une URL indiquant où se situe la ressource à plaquer sur l'objet.

Les éclairages en VRML sont gérés classiquement par des sources directionnelles, ponctuelles ou lumière ambiante, définis par une couleur, une intensité et une intensité de réflexion.

**Fonctionnalités d'animations :**
Les animations en VRML routent par le biais d'événements de sortie ou d'entrée un déclencheur d'événement à un récepteur par l'intermédiaire d'un interpolateur. Les durées sont exprimables en absolu, en boucle ou en fractionnel (intervalles à choisir). L'animation est déclenchée soit par une action utlisateur sur un objet dit Sensor, soit en référence à une horloge, soit par routage depuis un autre objet.




La syntaxe de l'instruction ROUTE impose que chaque node soit explicitement nommé. Les interpolateurs servant à calculer les valeurs intermédiaires entre par exemple deux positions d'un objet peuvent être utilisés sur plusieurs routages. Leur type correspond au type de nœud à animer (Position, Orientation, Color, etc..).
L'attachement d'événement à un objet défini par un nœud Transform par exemple implique donc la définition de trois nœuds de gestions d'événement : nœud de détection d'événement (TouchSensor par exemple, qui réagit au click souris sur l'objet), nœud de contrôle du temps (TimeSensor, déclenché par routage depuis le TouchSensor) et enfin nœud d'interpolation (déclenché par routage depuis le TimeSensor et qui affecte les valeurs du nœud récepteur).
Enfin, il est à noter que les nœuds de détection d'événements peuvent être inclus dans un objet visible de la scène ou non.

**Affectation d'URL :**
Le nœud d'affectation d'une URL à un objet est un nœud de type "Groupes" qui prend un ensemble de nœuds enfants pour paramètres. Son champ URL peut contenir tout document gérable par un serveur Web.

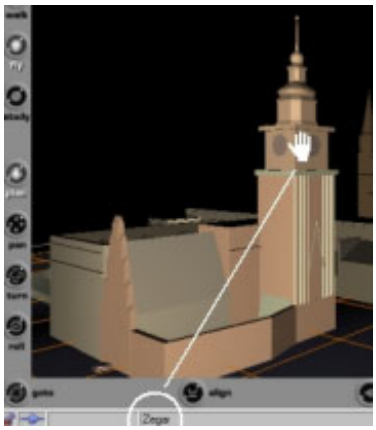


*Figure 109 :Affectation d'URL en fonction de la classe de l'entité architecturale dans une scène VRML (cas du Ratusz Krakowski)*

**Gestion des niveaux de détail (représentations alternatives)**
Le node LOD (Level Of Detail) permet de spécifier un nombre libre de représentations alternatives de l'objet en fonction de la distance observateur-objet. Sa forme est la suivante :

```
LOD {
  Level []
  Range []
}
```

où *level* contient l'ensemble des représentations alternatives de l'objet et *range* les distances objet-observateur à partir desquelles se font les basculements d'une représentation à la suivante. Si le champ *level* contient N représentations, le champ *range* doit contenir N+1 éléments. Ce mécanisme de représentations alternatives, couplé avec l'architecture nœud parent- nœud enfant de VRML, permet de simuler la notion de niveau de détail d'une façon sommaire mais efficace.

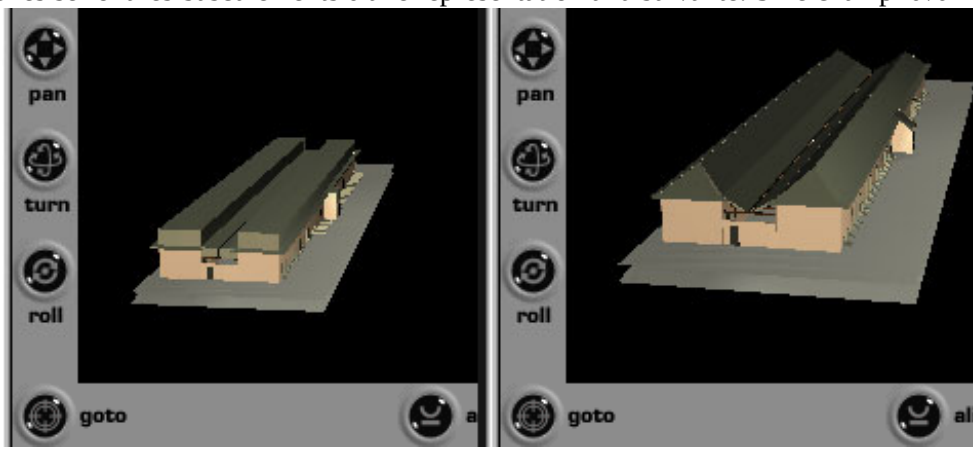



*Figure 110 : Géométries alternatives d'une visualisation en VRML (node LOD)*

Par ailleurs, les fonctionnalités d'animations de VRML peuvent être utilisées ici pour asservir le basculement d'une représentation à la suivante à tel ou tel événement.

**Insertion de scripts et communication VRML-JAVA :**
Un nœud de type script permet d'effectuer des traitements en fonction de tel ou tel événement à l'intérieur d'une scène. Il contient une liste configurable d'eventIn, d'eventOut et de fields. Le contenu du nœud script peut être une séquence d'instructions javascript :

```
Script {
  url javascript: ….
}
```

L'architecture JSAI permet de réaliser un premier type de communication VRML - Java. Le nœud script contient alors l'adresse url de la classe JAVA à utiliser. Cette classe est une extension programmeur de la classe Script du package JAVA nommé VRML. Le programmeur en redéfinit en particulier les méthodes initialize, processEvents et eventsProcessed pour assurer le traitement des données de la scène et rafraîchir les valeurs des champs du nœud Script.L'architecture EAI constitue la seconde solution pour réaliser une communication VRML - Java. L'EAI définit un mécanisme pour envoyer et pour recevoir des événements depuis un programme externe à VRML. En JAVA, l'EAI est implémenté dans un jeu de classes appelées depuis un applet pour contrôler une scène VRML. Les Classes Java concernées sont rassemblées dans un pacakge nommé vrml.external. EAI permet d'obtenir depuis l'applet une référence vers le plug-in puis la référence d'un noeud VRML ("handle") que l'on peut dès lors manipuler pour lire ou écrire des valeurs par exemple, mais aussi ajouter ou enlever des objets ou recevoir des événéments de la scène.
On trouvera plus de détail sur l'EAI sur le site de l'EAI working group (http://www.web3d.org/WorkingGroups/vrml-eai/) ou dans [Roehl et al, 1997]. Les deux méthodes, JSAI et EAI, peuvent poser des problèmes de compatibilité avec les navigateurs Web et leurs plug-ins. Ces solutions sont donc à la fois prometteuses et problématiques. En effet, on se retrouve là dans une situation où un choix technique visant à améliorer l'expressivité de la représentation de l'édifice vient en contradiction les objectifs 1 et 2 que nous nous sommes fixés en introduction à la section intitulée "Formats 3d pour le web":

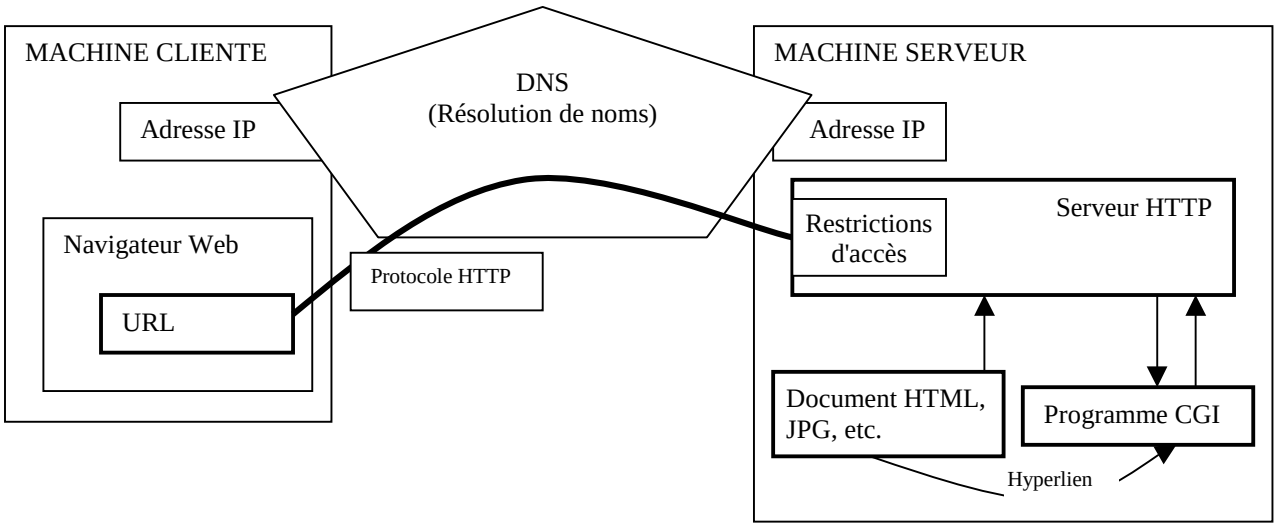
- Permettre la visualisation sur le web, sans investissement matériel, de scènes 3D.
- Permettre la visualisation de ces scènes 3D sur n'importe quelle plateforme.

## ANNEXE 31 : INTERFACES POUR LE RÉSEAU INTERNET

Dans cette annexe nous souhaitons situer les choix faits sur ce projet en terme d'architecture client-serveur sur le réseau Internet. Nous allons donc dans un premier temps évoquer sous la forme de définitions terminologiques les caractéristiques les plus significatives du réseau Internet, puis présenterons un état actuel et les perspectives de développement des langages de formatage ou de programmation les plus utilisés sur ce réseau. Nous terminerons cette section par un tour d'horizon des solutions récentes ou en devenir, avec en particulier un regard sur XML.

Aspects essentiels du réseau Internet

La littérature consacrée au réseau Internet est aujourd'hui très abondante. On peut distinguer trois catégories d'ouvrages en fonction des préoccupations des différents intervenants sur le réseau: ouvrage destinés aux administrateurs de réseaux ou aux gestionnaires de serveurs, ouvrages destinés aux programmeurs d'applications ou aux fournisseurs d'informations, ouvrages destinés aux utilisateurs du réseau. Compte tenu de l'importance des ressources disponibles, nous ne pouvons ici que proposer un ensemble de pistes bibliographiques relatives à ces catégories d'intervenants. Dans le premier cas, on peut par exemple se reporter à [Tanenbaum, 1997], [Huitema, 1994] ou [Albitz et al, 1998] qui traitent des aspects relatifs à l'administration réseau (nommage, routage, caractéristiques matérielles et logicielles). Dans le second, de nombreux ouvrages traitent des différents aspects de la production de documents pour Internet, depuis l'initiation aux langages HTML et Javascript ou à l'interface CGI et au langage Perl ([Lecomte et al, 1996], [Cohen, 1996], [Christiansen et al, 1999], [Conway, 2000], [Brenner et al, 1996]) jusqu'à la programmation réseau en Java ([Harold, 1997], [Orfali et al, 1996])en passant par l'installation et la mise en œuvre de serveurs HTTP [Laurie et al, 1997]. Le schéma qui suit récapitule les éléments essentiels d'une communication client-serveur sur Internet au travers du protocole HTTP. Ces éléments sont ensuite détaillés sous la forme d'un court glossaire.



**Services Internet** : Internet offre un ensemble de services (documents multimédias, courrier électronique , forums de discussion , consultation de bases de données , téléchargement de documents, etc..) assurés par des protocoles spécifiques (tcp/ip pour la transmission des informations , http pour le World Wide Web, smtp et pop3 pour le courrier, etc…). Ces services permettent à un client distant d'envoyer à un serveur HTTP par le biais d'une adresse URL une requête que celui-ci est chargé de traiter.

**Adresse URL (Uniform Resource Locator)** : L'adresse URL d'un document échangé entre un serveur et un client sur Internet répond à une définition standardisée qui inclut trois parties : le protocole, le nom de domaine et l'emplacement du fichier recheché sur la machine identifiée au point précédent. Le protocole peut être http, cas des pages web courantes , mais également ftp (File Transfer Protocol), gopher ou telnet. Nous ne revenons pas sur ces trois derniers dont nous ne faisons pas usage dans le cadre de ce projet.

**Serveurs HTTP** : Un serveur HTTP est un logiciel apte à répondre à des requêtes et à envoyer des pages d'information en utilisant le protocole HTTP (HyperText Transfer Protocol). Un serveur HTTP (ou WWW) attend en permanence des requêtes HTTP pour entreprendre des actions comme par exemple envoyer des documents de type HTML depuis la machine serveur jusqu'à la machine client. Ces documents sont à interpréter par la machine cliente qui doit disposer pour ce faire de logiciels adéquats comme par exemple le navigateur Netscape en ce qui concerne des pages au format HTML. Plusieurs dizaines de serveurs HTTP sont disponibles aujourd'hui sur tous les systèmes serveurs. Les pages Web sont en majorité des documents dits hypertextes contenant des hyperliens.

**Hyperliens et document hypertexte** : le document hypertexte est un fichier texte contenant des liens vers d'autres parties du document lui-même ou vers d'autres documents, sur le même ordinateur ou sur un ordinateur distant.
Un hyperlien, ou encore lien hypertexte, est formé d'une ancre et de l'adresse du document cible[204].

**Protocole HTTP (HyperText Transfer Protocol)** : Il gère la communication entre un client et un serveur HTTP. Côté client, la requête se compose d'abord d'une instruction "GET", qui indique quel fichier est demandé, et qui dit quelle version du protocole est utilisée. Ensuite sont précisés les types MIME (Multipurpose Internet Mail Extension) acceptés en retour, l'identification du client et une ligne blanche concluant la requête. En retour, le serveur répond en indiquant le statut de la requête (effectuée avec succés ou non), son identification, la version MIME utilisée, le type MIME de l'information transmise, le nombre de caractères qu'il envoie puis une ligne blanche et les données elles-mêmes.

**CGI** : abréviation de Common Gateway Interface, CGI est un standard d'interface entre les serveurs HTTP et des applications externes. Cette interface ou passerelle est dépendante du logiciel serveur HTTP, sa mise en œuvre est donc à considérer au cas par cas. Les programmes éxécutés au travers de cette passerelle sont eux a priori plateforme indépendants, ils peuvent être écrits dans un langage de script comme Perl ou dans un langage de programmation comme JAVA.

**Exécution des programmes CGI** : Une adresse URL identifie le plus souvent un fichier au format HTML, mais peut aussi référencer un programme ou script. Ce programme sera exécuté en résultat à l'activation du lien vers cette URL. Lorsque ce programme s'éxécute, il prépare une page Web (HTML) "au vol" qui est envoyée en réponse au client. Cette réponse peut bien sûr inclure un certain nombre de résultats de calculs effectués par le programme sur

[204] Une présentation globale et largement documentée de la notion d'hypertexte est proposée dans la revue en ligne Cursus vol.1 no 1 (revue de l'École de bibliothéconomie et des sciences de l'information de l'université de Montreal) par G.Teasdale à l'adresse suivante : http://www.fas.umontreal.ca/EBSI/cursus/vol1no1/teasdale.html

des données qui lui sont soumises, par le biais de formulaires par exemple. Un jeu de balises du langage HTML permet en effet au programme de récupérer les données soumises par l'utilisateur et qui lui sont transmises par le serveur HTTP. C'est par exemple le principe des interfaces Web de consultation de Systèmes de Gestion de Bases de Données.

**DNS (Domain Name System)** : C'est un système d'organisation hiérarchique des noms dans un réseau important d'ordinateurs. Chaque élément est référencé comme un domaine, que cette référence soit un ordinateur unique ou définisse des sous-domaines. Le domaine racine de cette hiérarchie est repéré par "." dans le DNS. Le point sert dans l'adressage d'un domaine ou sous-domaine à expliciter le chemin d'accès à cette adresse en spécifiant l'ensemble de ses domaines parents dans la hiérachie. Les machines contenant les informations DNS sont appelées serveurs de noms, et peuvent déléguer une partie de leurs attributions à d'autres serveurs de noms. Chaque serveur HTTP est référencé par ce mécanisme, qui adresse la requête émise par un client au serveur HTTP correspondant. Bien qu'une hiérarchie DNS puisse être utilisée en réseau privé, son application principale est le réseau Internet. Concrètement, le DNS est utilisé pour déterminer l'adresse IP d'une machine de façon à établir la communication entre client et serveur (résolution de noms).

**Résolution de noms** : Le DNS est chargé d'établir l'adresse IP d'une machine serveur sur requête d'un client mais également de l'inverse. Ce mécanisme inverse permet au serveur d'établir l'identité de la machine cliente, information essentielle pour établir des restrictions d'accès au serveur HTTP.

**Restrictions d'accès** : un serveur WWW permet de restreindre l'accès à certains documents en fonction en particulier de l'origine du client (domaine Internet, numéro IP de réseau et/ou machine) mais aussi au travers d'une identification de la personne requérante (mot de passe). Ces fonctionnalités sont implémentées de façons différentes en fonction du logiciel serveur utilisé, et peuvent être l'objet de développements externes assurant une meilleure portabilité des systèmes de sécurisation des documents.

Langages pour le Web

Nous venons de très brièvement procéder à une énumation de termes utilisés dans le cadre de l'architecture du réseau Internet. Nous allons ici procéder de même pour une question qui nous intéresse plus directement, quels langages de programmation et quels formats de structuration de documents sont aujourd'hui disponibles pour tirer profit de ce réseau ?
En effet, entre langages, formats et normes pour le web, de nombreux sigles sont apparus ces dernières années. Nous allons donc commencer par établir sous la forme d'un rapide glossaire quelques définitions terminologiques puis reviendrons plus en détail sur HTML, Javascript et XML.

### *Rappels Terminologiques*

**SGML (Standard Generalized Markup Language)** : c'est un métalangage de balisage de documents structurés, héritier du travail sur la séparation de la forme et du fonds proposé en 1967 par W.Turncliffe. Le principe établi alors visait à formaliser la distinction entre les informations contenues dans les documents et les instructions concernant leur formatage. Ces travaux furent à l'origine de la création du langage "GML" (Generalized Markup Language) par IBM en 1969. Ce langage introduisait en plus du concept de balisage, celui de types de documents formellement définis comprenant une structure d'éléments imbriqués. Un projet de norme basée sur le GML fut entrepris à partir de 1978 à l'instigation de l'ANSI (American National Standards Institute) aboutissant en 1986 à la publication de la norme SGML (ISO 8879). Cette distinction structure / contenu doit permettre notamment d'associer à chaque élément d'un document un style particulier modifiable ultèrieurement sans intervention sur le contenu. SGML permet l'échange et la réutilisation de textes électroniques tout en préservant le contenu (l'information) et la structure (les relations entre les données). SGML utilise un balisage imbriqué définissant une structure hiérarchique pour le document. SGML est un

métalangage permettant d'écrire d'autres langages, décrivant chacun un type de documents particulier: SGML permet concrètement de définir des langages de balisage sémantique des textes au travers de DTD (Définitions de Types de Documents). Ces DTD sont donc des applications SGML, applications dont la plus utilisée est le langage HTML. La définition d'une DTD est indispensable à la mise en forme d'un document SGML puisqu'elle permet de préciser pour un type de document quels sont leurs éléments communs, leurs relations et les noms des balises utilisées[205].
SGML distingue le contenu d'un document, sa description syntaxique (DTD), et sa sémantique d'affichage (style sheet). Notons enfin que SGML utilise un système de référence à des entités indépendant de la plateforme matérielle utilisée, dont une des conséquences est la possiblité de choisir des jeux de caractères différents à l'affichage. Des produits commerciaux d'édition sgml (sans conversion HTML) existent de même que des navigateurs capables de traiter directement les documents SGML[206]. Il n'en reste pas moins que ces outils sont moins diffusés que ceux centrés sur HTML. Par ailleurs, actuellement, plusieurs logiciels de traitement de texte courant peuvent convertir des fichiers SGML. Ces conversions restent cependant imparfaites.

**DSSSL (Document Style Semantics and Specification Language)** : c'est une norme connexe à SGML permettant d'échanger de façon normalisée les instructions de formatage. Ce standard, établi en 1996, fournit une syntaxe normalisée de définition de la présentation d'un document SGML. DSSSL est orienté formats de sortie, avec avant tout comme priorité les sorties papier, c'est donc une norme établie plutôt à destination du monde de l'édition. Un sous-ensemble de DSSSL baptisé DSSSL-O est développé avec pour motivation de fournir les mêmes services cette fois-ci pour l'échange de document SGML sur Internet. Des outils implémentant DSSSL sont en cours de développement, leur nombre reste cependant limité[207].

**DTD (Définition de Types de Documents, ou Document Type Definition)** : Une DTD permet de préciser la structure d'un document SGML. Elle définit un ensemble de balises utilisables pour le document, la relation entre ces balises, les attributs de qualification qui s'appliquent à chaque balise et le format des liens avec les autres documents. Des DTD spécifiques métiers ont été développées comme par exemple la DTD dite TEI (Text Encoding Initiative) formulée avec la participation de bibliothécaires pour standardiser l'encodage de textes dans le domaine des sciences humaines et de la linguistique[208]. La DTD HTML est elle adoptée par l'ensemble des navigateurs Web qui interprétent les balises du langage pour affecter tel ou tel style à une portion de document par exemple. Une page HTML doit donc se conformer à une DTD. Cependant, cette DTD a été très largement adaptée par les éditeurs de navigateurs pour inclure telle ou telle fonctionnalité que seul leur navigateur reconnaît. De fait, quatre versions du langage HTML se sont succédées rapidement ces dernières années, amendant par la tactique du fait accompli la DTD initiale.

**HTML (HyperText Markup Language)** : HTML est une instance de SGML (Standard Generalized Markup Language). C'est un langage de balisage hypertexte décrit comme une DTD de SGML. Nous reviendrons sur HTML plus loin dans ce document.

**DHTML (Dynamic HTML)** : C'est un ensemble d'outils permettant d'animer une page Web (c'est à dire ici en particulier de réagir aux évènements déclenchés par le visiteur et de gérer le rafraîchissement graphique des pages). DHTML s'appuie sur la mise en œuvre de pages

[205] On peut se reporter pour une présentation de SGML dans le contexte de son application à la mise en forme à l'ouvrage *La construction de l'anthropologie au Québec* (première publication électronique des Presses de L'Université Laval, Québec) à l'adresse : http://www.bibl.ulaval.ca/doelec/pul/intromat.html
[206] Voir http://www.oasis-open.org/cover/marcouxWhy00.html pour plus d'informations sur ces logiciels.
[207] On peut se reporter notamment pour ce qui est des limitations de DSSSL (fidélité de la sortie en particulier) ou aux outils DSSSL à l'adresse suivante: http://www.mcs.net/~dken/dslintro.htm
[208] Voir le site TEI Application Page Project Descriptions par Wendy Plotkin et C. M. Sperberg-McQueen à l'adresse: http://www.uic.edu:80/orgs/tei/app.

Web intégrant HTML (pour la mise en page), Javascript (pour la l'interaction avec les éléments du Document Object Model) et CSS (Cascading Style Sheets pour la présentation de la page). DHTML est une conséquence de l'apparition de la notion d'objet dans une page Web, notion apparue avec l'arrivée du langage de programmation interprété Javascript. Nous reviendrons sur ces notions en présentant plus en détail le langage Javascript. Notons cependant dès à présent que comme pour la DTD HTML, la compatibilité de développements DHTML peut poser problème selon les navigateurs utilisés.

**CSS (Cascading Style Sheet)** : On l'a vu, SGML distngue le contenu d'un document, sa description syntaxique (DTD), et sa "sémantique" d'affichage (style sheet). Application de ce principe, Les feuilles de style sont apparues dans la version 4 de HTML. Elles donnent la possibilité de déterminer globalement par exemple taille, couleurs et polices de caractères, fond de la page, marges, etc... . Ces feuilles de styles peuvent être incluses dans une page HTML classique ou définies dans un fichier séparé et partagées par plusieurs pages HTML. Comme toutes les évolutions antérieures de HTML, l'apparition des feuilles de styles s'accompagne d'un versionnement des navigateurs aptes à les reconnaître.

**DOM (Document Object Model)** : Le Document Object Model est un API pour des documents HTML ou XML. C'est une spécification du consortium W3C datée de 1998 destinée à fournir un interface de programmation utilisable pour divers environnements et applications, originellement développé pour permettre aux programmes Javascript et Java d'être portables sur les navigateurs. Le Document Object Model définit la structure logique d'un document et la façon de le manipuler. Chaque élément d'un document est dès lors un objet (au sens de la Programmation Orientée Objets) isolément manipulable par un langage de programmation. Le Document Object Model fournit une description logique du document qu'il modélise sous la forme de hiérarchies indépendantes représentant tel ou tel objet présent dans le document.
La spécification du Document Object Model comprend deux parties, nommément DOM Core et DOM HTML[209].

- DOM Core définit un ensemble minimal d'objets et d'interfaces permettant d'accéder aux objets contenus dans un document et de les manipuler.
- DOM HTML est une extension du précédent aux objets et méthodes specifiques aux documents HTML.

**XML (Extensible Markup Language)** : C'est, pour reprendre les termes employés par le consortium W3C dans sa présentation de cette spécification datée de 1998, le format universel de documents et de données structurés sur le Web[210]. XML est un sous-ensemble de SGML destiné à permettre l'échange de documents SGML génériques sur le Web de la même façon qu'est aujourd'hui possible l'échanges de données HTML. Nous revenons sur XML plus loin dans ce document.

**XHTML** : C'est un langage comparable à HTML mais conçu dans un souci de compatibilité avec la spécification XML présentée ci-dessus. Il introduit quelques contraintes syntaxiques assez simples (quoique lourdes en terme de réutilisation de documents existants) par rapport au langage HTML. XHTML est compatible avec les navigateurs actuels et devrait l'être avec ceux de la génération à venir[211].

**Javascript** : créé par la société Netscape, Javascript est un langage de script interprété par les navigateurs Netscape depuis leur version 2.0 (et par d'autres par la suite) permettant de programmer par exemple des réactions aux évènements utilisateur. Javascript permet en

[209] On se reportera à la spécification du DOM par le World Wide Web Consortium (W3C) pour plus de détails: http://www.w3.org/TR/REC-DOM-Level-1/cover.html
[210] Voir http://www.w3.org/XML/#9802xml10. La spécification XML est à l'adresse: http://www.w3.org/TR/1998/REC-xml-19980210
[211] Voir XHTML par Ian Graham: http://www.utoronto.ca/webdocs/HTMLdocs/NewHTML/xhtml.html

résumé de modifier et de contrôler le contenu du document HTML affiché par un navigateur côté client. Nous revenons sur ce langage plus loin dans ce document.

**Vbscript:** VBScript est le nom donné par la société Microsoft à son langage de script basé sur le langage de programmation Visual Basic. C'est un sous-ensemble de *Visual Basic for Applications* (VBA). Javascript peut faire tout ce que Vbscript peut faire et plus. Nous ne nous attarderons donc pas sur ce langage par ailleurs seulement compris par les navigateurs fournis par Microsoft (Internet Explorer).

Je ne m'attarderai que sur trois des éléments cités ci-dessus, HTML, Javascript et XML, éléments qui me semblent les plus importants à aborder dans le cadre du projet que je défends.

### *HTML*

Le langage de balisage HTML, issu de la DTD du même nom, a subi de nombreuses évolutions ces dernières années. Dans sa première version, HTML était un langage de formatage et de structuration de documents échangés sur le web. Limité en terme notamment de fonctionnalités permettant d'assurer la gestion d'évènements utilisateur, HTML a été plusieurs fois remanié avec les problèmes de compatibilité que l'on imagine. Nous allons ici d'abord rappeller quelles ont été ces évolutions du langage puis aborder deux points plus fondamentaux: les techniques d'interfaçage client-serveur et de communication avec des applications adaptées au langage HTML, puis ses limitations intrinsèques, autrement dit les raisons de l'apparition récente de XML, censé remplacer à terme l'actuel HTML. A travers ces deux points se fera naturellement le lien avec les sections suivantes consacrées à Javascript et à XML.

#### *Principe et évolutions*

On l'a indiqué précédememt, HTML est une instance de SGML (Standard Generalized Markup Language). Elle s'en éloigne cependant notablement puisque plusieurs des balises HTML sont des balises procédurales, c'est-à-dire ne s'attachant pas à décrire la structure du document mais l'apparence du texte balisé. Les principales versions du langage sont HTML 1.0, HTML 2.0, HTML 3.2 et HTML 4.0. Nous revenons rapidement sur les étapes principales de son évolution puis en présentons les aspects essentiels.
En 1989, Tim Berners-Lee et Robert Caillau ont introduit au CERN (Centre Européen pour la Recherche Nucléaire) l'idée d'un système d'informations accessible à travers les diverses plateformes matérielles utilisée dans ce centre de recherche européen[212]. HTML a donc été conçu au départ pour répondre à une exigence d'échange d'informations, en parallèle à une solution réseau HTTP. En 1991, le CERN rendait publique le résultat de ce travail et lançait ce qui est devenu le web, c'est à dire la conjonction de quatre formalismes ou développements:

- Serveurs et navigateurs.
- Protocole de transfert (HTTP).
- Langage de composition des documents échangés (HTML).
- Mécanisme d'adressage des documents (URL).

Avec le développement du navigateur Mosaic par l'organisation américaine NCSA (National Center for Supercomputer Applications), le web prenait dès 1993 un essor international. Il ne nous semble pas utile de rappeler ici combien de sociétés commerciales ont depuis lors proposé leur propre navigateur et / ou serveur. Rappellons seulement qu'en terme de serveur le plus réputé est aujourd'hui Apache, et qu'en terme de navigateur Netscape et Microsoft sont les noms les plus cités.

[212] Voir le site de cette organisation (http://cern.web.cern.ch/CERN/) et pour un historique du web l'adresse suivante: http://ei.cs.vt.edu/~wwwbtb/book/chap1/web_hist.html#1.3.1

Les caractéristiques essentielles de la première version du langage, HTML 1.0, étaient les suivantes :

- une structuration simple des documents (titre de section, liste d'éléments),
- la possibilité de contenir des images,
- des liens hypertextes pointant vers d'autres documents.

HTML 2.0 a été le premier standard largement répandu pour Internet. La plupart des navigateurs commerciaux ont implémenté cette version. HTML 2.0 intégrait les balises permettant la création de formulaires autorisant un premier niveau d'interaction système / utilisateur. Dès cette version du langage, les navigateurs commerciaux proposaient leurs propres extensions du langage. HTML 2.0 (spécification RFC 1866, 1995[213] ) reprenait en fait pour une bonne part les extensions au langage apparues de fait dès 1994.

Dans le même esprit (le fait accompli des éditeurs de navigateurs) Le consortium W3C a proposé le standard HTML 3.2 (spécification REC-html32, 1997) reprenant notamment certaines extensions au langage introduites par Netscape (tables, images de fond, etc…).

La dernière version du langage HTML, version 4.0, (spécification REC-html40, 1998), introduit un certain nombre de nouveautés ou d'évolutions, avec en premier lieu les feuilles de styles[214] mais aussi:

- L'élément OBJECT (applications externes).
- Les FRAMES (nouvel emprunt à Netscape).
- L'attribut LANG (indication relative à la langue utilisée dans tout ou partie du document).
- L'élément LINK qui permet de créer des liens exploitables par des outils de gestion ou des moteurs de recherche.

HTML est censé s'occuper de la structure d'un document plutôt que de son apparence. Censé seulement, car si ce principe était clairement affiché dans sa première version, le souci d'offrir une plus grande maîtrise de l'apparence d'une publication Internet a fait évoluer le langage dans le sens d'une description (quasi-)exhaustive
de l'apparence des éléments affichés. En effet, Le langage HTML permettait au départ de contrôler le contenu de la page mais laissait l'apparence au soin du navigateur en charge de visualiser la page. L'auteur indiquait par exemple qu'un texte devait être affiché comme titre mais le navigateur client choisissait pour l'afficher la police de caractères et sa taille. Aujourd'hui, on peut dire que seule la largeur et la hauteur d'affichage restent à la charge de la machine cliente puisque les balises spécifiques à la mise en page permettent de figer celle-ci. HTML 1.0 distinguait un contenu (connu de l'auteur et de l'utilisateur) et un contenant (connu du seul utilisateur), contenant dont la mise en page était réputée "flottante". Avec HTML 3.2 puis 4.0 contenu et contenant tendent à être maitrisés par l'auteur de la page[215]. Conséquence paradoxale de ce souci, les navigateurs tendent également à ne plus afficher de la même façon le même contenu, mais nous ne souhaitons pas détailler ce point qui reste dans le cadre de notre travail tout à fait anecdotique.
Le format d'un document HTML est celui d'un texte ASCII créé avec n'importe quel éditeur de texte. Des éditeurs de HTML sont disponibles en nombre, avec bien sûr plus ou moins de fonctionnalités et un respect plus ou moins grand des spécifications du langage. L'utilisation d'un éditeur de document HTML WYSIWYG comme netscape Gold ou aolPress revient en fait à demander à ces logiciels de traduire la mise en page effectuée dans leur interface au format HTML. Internet est aujourd'hui à la fois un terrain de compétition commerciale et un

[213] Voir http://www.w3.org/MarkUp/html-spec/ pour la version 2.0, http://www.w3.org/TR/REC-html32.html pour la version 3.2 et http://www.w3.org/TR/1998/REC-html40-19980424/ pour la version 4.0

[214] Parallèlement à l'évolution du langage, le consortium W3C a travaillé à la définition des feuilles de style (Style Sheets) destinées à autoriser le détermination de l'apparence d'élements dans un document (ou plusieurs) de façon générique (voir section précédente).

[215] On notera cependant qu'il s'agit encore en partie d'un argument commercial plus que d'une réalité puisque notamment les tables de couleurs, définition de l'affichage ou encore aspect du navigateur restent du ressort de la machine cliente.

réseau ouvert utilisant des standards de communication. Bien des paradoxes viennent de cette double appartenance.
HTML est un langage de balisage définissant la structure (et l'apparence) d'un document interprété par un navigateur. HTML d'une part définit la structure et la mise en page du document, et d'autre part autorise la création notamment des hyperliens. Ces divers éléments sont repérés dans le corps du document par des balises (tag en anglais). Ces balises sont encadrées par des signes supérieur et inférieur d'angle (< NomDeLaBalise >). Beaucoup d'entre elles vont par paire, une balise indiquant le commencement d'un élément et une balise quasi-identique (</NomDeLaBalise >) marquant la fin de sa portée. La syntaxe de base du langage HTML est la suivante :

<balise attributs> contenu auquel s'applique la balise </balise>

Les balises sont bien entendu imbricables. Les attributs d'une balise s'écrivent sous la forme suivante :

Attribut = "valeur "

Chaque balise dispose d'une liste d'attributs permettant de spécifier par exemple une cible dans le cas d'un lien, ou divers éléments de mise en page. L'emploi de minuscules ou de majuscules dans les balises était indifférent jusqu'à l'apparition de XHTML.
Un document en HTML est divisé en deux sections:

- Un en-tête (masqué par le navigateur à l'exception du titre)
- Le corps du document (affiché par le navigateur)

L'en-tête d'un document HTML contient des renseignements relatifs à son contenu (titre, mots-clés, auteur, etc..) notifiés par l'auteur en particulier pour assurer la diffusion la plus efficace de son document sur les moteurs de recherche par mots-clés courants (Altavista, voilà, etc..).Le corps du document contient des informations relatives à la mise en page du document (couleurs, typographie, etc..) et bien entendu le contenu de celui-ci.
Les tableaux ou tables sont le moyen le plus souple de créer une mise en page en HTML. La dimension d'une table peut être spécifiée en pourcentage de la fenêtre du navigateur (Une mise en page prévue lors de la
composition de la page est ainsi mieux respectée par les diverses machines client qui la consultent) ; les rangées (balise <tr>) et les cellules de la table (balise <td>) sont indépendantes en terme de présentation (fond
coloré, alignements, ..). Donnons à titre d'exemple de balises HTML les balises utilisées pour une table:

| | |
|---|---|
| Création de la table | <table> |
| Création de la première rangée | <tr> |
| Création de la première cellule | <td> |
| Fermeture de la première cellule | </td> |
| Fermeture de la première rangée | </tr> |
| Fermeture de la table | </table> |

Avant d'en terminer avec cette mise en perspective des principes du langage HTML, citons encore quelques exemples des balises les plus fréquentes.
La balise <A> identifie un texte ou une image comme lien hypertexte. Le lien est effectif sur tout élément compris entre <a> et </a>. Les documents vers lesquels renvoie le lien ainsi créé sont identifiés par une adresse placée derrière l'attribut HREF. Si l'élément compris entre <a> et </a> est un texte, alors il est reconnu car affiché par un soulignement et une couleur spécifique (voir balise <body>). S'il s'agit d'une image, celle ci est encadrée par un bord de la couleur définie dans la balise <body> à moins de préciser pour l'image un attribut BORDER valant 0.
La gestion locale des images réactives permet d'associer à chaque région identifiée à l'intérieur d'une image un lien hypertexte local ou distant. L'attribut USEMAP ajouté dans la balise <IMG> précise l'emplacement de la "carte" de référence affectant à chaque région identifiée un lien hypertexte. Si la "carte" est définie à l'intérieur du document HTML dans

lequel est placée la balise <IMG>, l'attribut USEMAP est utilisé avec le signe # (usemap="#nomDeLaCarte").
Le langage HTML permet de construire un document comme un assemblage de références externes (images par exemple). Le document se compose de différents éléments, mis en page, structurés par les balises du langage, mais restant stockés indépendamment. Le document HTML fait donc appel à des fichiers externes qui seront affichés à l'appel du document (le cas des images) ou sur requète d'un utilisateur (cas d'un hyperlien).
L'adressage de ces fichiers externes, locaux ou distants, se fait le mécanisme d'URL qui précise la méthode utilisée pour aller chercher un document (le protocole) et l'adresse du document (voir section précédente). Les fichiers "externes" seront interprétés par le navigateur pour autant qu'ils correspondent à des formats de fichiers pour lesquels le navigateur dispose d'un outil de consultation.
Par exemple, un document au format VRML ne sera reconnu par le navigateur que si celui-ci dispose d'un outil de consultation adéquat (plugin Live3D ou autre pour netscape ). Le caractère multimédia souvent vanté d'un document HTML est donc dépendant avant tout de la plateforme matérielle et logicielle côté client.
Enfin, les frames apportent une solution pour afficher dans l'espace du navigateur plusieurs documents différents simultanément. Chaque document affiché posséde sa propre adresse (URL) et un nom l'identifiant comme partie de la fenêtre principale. L'affichage des documents ainsi assemblés est indépendant en adresse comme en durée, permettant de rendre permanent l'affichage de certaines informations comme des boutons de navigation, un logo, etc…Ces propriétés peuvent, par exemple, être utilisées pour afficher de façon indépendante le contenu d'un document long et un index permettant de le parcourir.
La syntaxe à utiliser comprend deux éléments : une balise <frameset> et </frameset> qui identifie un " jeu de frames " (régle la position et la proportion de chaque frame contenu entre <frameset> et </frameset>) , et des balises <frame> qui définissent le contenu de la sous-fenêtre créée. Dans le document " parent " la balise BODY est donc remplacée par la définition de sous-fenêtres qui chacune possèderont une balise <BODY> indépendante. Aucune mise en page hormis le choix de proportions entre les sous-fenêtres n'est laissé au document parent contenant la balise <frameset>.
Si le principe du langage est simple (c'est là probablement une des raisons de son succès), il ne faut pas cependant minimiser les problèmes de compatibilité entre navigateurs-éditeurs de HTML qui se posent aujourd'hui.
En effet, le passage pour l'auteur d'un document HTML par un outil de ce genre lui masque (sauf volonté expresse) la traduction HTML que cet outil a générée. Il n'est donc pas à même d'évaleur la compatibilité de son document avec d'une part la spécification HTML et d'autre part d'autres outils.

### *Interfaçage client-serveur et communication avec des applications*

Précisons tout de suite que nous parlons ici d'éléments du langage HTML et non d'éléments qui, insérés dans un document HTML, permettent d'effectuer tel ou tel traitement. Nous reviendrons sur ces derniers en introduisant Javascript. Nous décrivons ici les balises HTML permettant soit de récupérer des données utilisateur soit de placer dans le document HTML un jeu d'instructions en langage de script soit encore de faire référence à une application externe.
Le langage HTML permet d'interroger l'utilisateur côté client par le biais de balises dites formulaires dont le contenu peut être transféré côté serveur pour d'éventuels traitements ou traités côté client par un langage de script. Ces formulaires peuvent comprendre différents types de boutons dont les principaux sont les suivants:

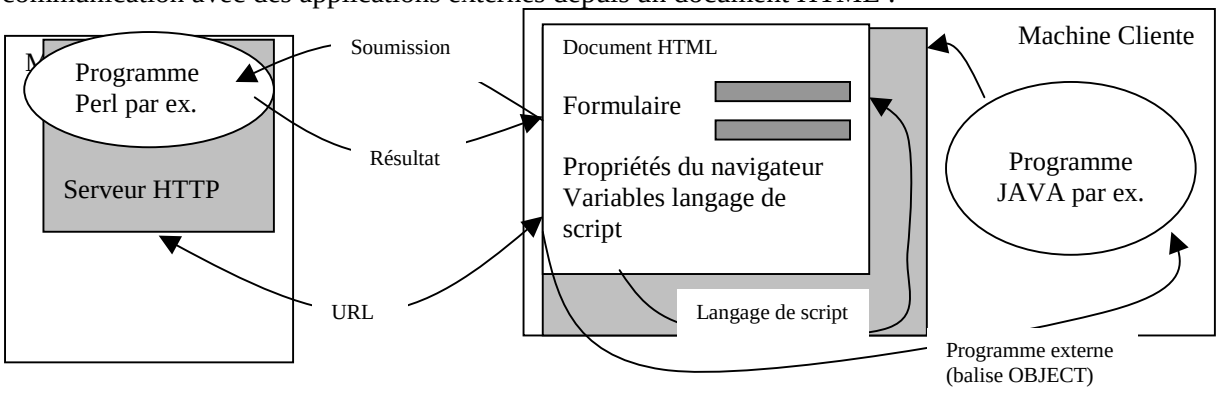
- Texte (mot de passe, ligne, caché, etc..).
- Zone de texte.
- Bouton à cocher, Bouton radio.
- Attachement de fichiers.
- Sélection simple ou multiple.

- Bouton d'effacement du formulaire.
- Bouton de soumission ou d'effacement de formulaire (texte).

Un formulaire commence par la balise <FORM METHOD="" ACTION=""> et finit par </FORM>. Ce qui est contenu entre ces balises est envoyé à la page ou au programme défini par ACTION="" sur action du Bouton de soumission en général. Les données passées à la page ou au programme chargé de les traiter sont concaténées sous la forme de couples nom du champ / valeur. La soumission d'un formulaire se traduit différemment selon le type de traitement défini par ACTION="". S'il s'agit d'un programme côté serveur (typiquement un fichier Perl par exemple), la soumission du formulaire équivaut au rafraîchissement de la page contenant le formulaire par l'affichage du résultat des traitements effectués sur le serveur et renvoyé au client. Un programme est lancé
pour chaque requête et se termine après avoir généré la réponse sous forme de document HTML. S'il s'agit d'instructions en langage de script inclus dans la page contenant le formulaire, ces instructions peuvent prévoir un rafraîchissement ou non de la page et sont de toute façon traitées côté client. On voit donc dès à présent les raisons pour lesquelles DHTML a inclus le langage de script Javascript. Les balises HTML permettant d'interroger l'utilisateur sont les mêmes quel que soit le traitement prévu, la distinction la plus sensible entre ces traitements est en fait leur localisation: machine serveur ou cliente.
La balise <Script> (close par </script>) permet de placer à l'intérieur d'un document HTML un jeu d'instructions en langage de script (Javascript ou Vbscript) destiné à effectuer tel ou tel traitement sur le contenu de la page, tant en terme de présentation graphique que d'interaction avec l'utilisateur. Nous revenons sur ce point plus loin.
Enfin, la balise <APPLET> permet de faire référence à un programme Java externe et de lui passer un certain nombre de paramètres[216]. Dans le cas d'applets en Java, la classe Applet et l'interface AppletContext permettent d'établir la communication entre l'applet et le navigateur qui l'exécute. Notons également que la balise <EMBED> permet de référencer des applications externes multimedia dites Plug-ins (qui prennent une portion de la page).
Enfin, avec la version 4.0 du HTML est apparue une balise générique d'inclusion notée <OBJECT>. Cette nouvelle balise s'inscrit en réponse aux problèmes suivants:

- L'inclusion de nouveaux ou futurs types de médias n'est possible au travers du jeu de balises existantes.
- L'élément APPLET fonctionne uniquement avec les applets en Java.

L'élément OBJECT vient en remplacement des balises IMG et APPLET, il permet également d'inclure un document HTML dans un autre.
Le schéma suivant récapitule les différents types d'nterfaçage client-serveur et de communication avec des applications externes depuis un document HTML :




[216] Avec la syntaxe suivante :

<APPLET CODE="nomDuFichierClass.class" WIDTH=LargeurAffichageEnPixels HEIGHT= HauteurAffichageEnPixels >
<PARAM name="nomDuParametre" value="Valeur">
……
</APPLET>

*Limitations*

Il n'est pas aisé de résumer les limitations du langage HTML tant elles sont relatives à des considérations diverses: en effet les limitations les plus sensibles de ce langage apparaîtront différentes selon que l'on se place du point de vue du fournisseur d'informations qui va souhaiter par exemple une plus grande expressivité du langage, ou du point de vue du graphiste pour qui la présentation prime, ou encore du point de vue du programmeur d'applications pour qui les méthodes d'interfaçage actuelle du HTML semblent étriquées.
Nous allons ici citer les principales critiques faites à l'encontre de ce langage non pour en discuter exhaustivement mais pour tenter d'expliquer pourquoi le développment de HTML est aujourd'hui interrompu au profit notamment de XML.
Commençons par rappeler quelques raisons essentielles du succès de ce langage:

- *Une grande simplicité pour le founisseur d'information comme pour l'utilisateur du réseau leur permettant de se former très rapidement, voire de s'autoformer.*
- *Un formatage des pages (présentation graphique) lui aussi simple, mais dont cette simplicité-même permet l'édition de documents rapidement avec un minimum d'apprentissage et de facteurs d'erreurs.*
- *Un mécanisme d'hyperliens facile à mettre en œuvre.*
- *Des balises de formulaire permettant d'interfacer pages HTML et programmes CGI sans grande difficulté.*

Bref, HTML a les avantages de ses inconvénients: c'est un langage simple à apprendre et à utiliser, langage qui fournit un jeu clos de balises et limite de ce fait les erreurs possibles. HTML a bien sûr également les inconvénients de ses avantages, à commencer par un nombre restreint de balises dédiées le plus souvent au formatage d'une page, et permettant une structuration somme toute triviale du document. On peut également ajouter à cette première limitation quelques autres points :

- *HTML sépare mal formatage et information.*
- *La réutilisation d'une information formatée en HTML est limitée par le coût (en temps d'intervention manuelle) de la conversion et / ou du formatage.*
- *HTML ne dispose pas de balises permettant de commmenter ou d'annoter une information: un document HTML brut ne contient de fait que des balises réglant l'apparence et la structure du document, mais pas la sémantique associée à tel ou tel élément d'information contenu dans la page.*
- *Les mécanismes d'automatisation en HTML ne reposent que sur des développements externes (CGI ou Java par exemple) par manque d'expressivité du langage.*
- *Les mécanismes de recherche d'information aujourd'hui basés sur la lecture par des moteurs de recherche du contenu des pages HTML sont notoirement peu efficaces. En effet, toujours comme conséquence du manque d'expressivité de ce langage, les moteurs de recherche se contentent de rechercher l'occurrence de mots-clés dans les documents qu'ils répertorient. Cette recherche s'effectue par comparaison de chaînes de caractères, donc sans capturer la sémantique ou le contexte d'utilisation de telle ou telle expression. Une même chaîne de caractère peut faire référence à des spécialités tout à fait distinctes, les mots "architecture" ou "restauration"en étant des exemples criants.*
- *L'évolution continue du langage (versions 2.0, 3.2, 4.0) a pour conséquence de placer les fournisseurs d'information devant la quasi-obligation de reprendre l'ensemble des documents qu'ils ont produit pour les adapter aux nouveaux navigateurs. Ce travail de mise à jour s'effectuant manuellement, il semble tout à fait déraisonnable dès que le nombre de documents produits est important. Pour éviter cet éceuil, de nombreuses organisations créent leurs sources d'informations en SGML ou en texte brut puis les traduisent au format HTML. Ceci implique nénmoins d'en passer par une étape de traduction.*

Nous le verrons, l'utilisation d'un langage de script peut aider ici ou là à prendre en compte ces limitations, mais ne suffit pas à masquer le problème de fond que pose HTML : le langage ne sait pas (ou peu) séparer la partie structuration du document (définition des titres, des paragraphes, des listes, ...) de sa présentation.
La solution recherché par le consortium W3C, un standard de transfert de données sur le web accepté et utilisé de tous, se nomme XML. Sans nécessiter de modifications au protocole de

communication, XML est une solution intermédiare entre SGML, jugé trop complexe pour une utilisation sur le web , et HTML, dont nous venons de cerner les limitations.
L'évolution du langage HTML est aujourd'hui terminée (avec la version HTML 4.01). XHTML (eXtensible HyperText Markup Language) est le langage appelé à le remplacer à court terme, XML étant un métalangage dont XHML exploite un sous ensemble en utilisant d'une une DTD (Document Type Definition) et d'autre part des feuilles de styles.

Javascript
Un langage de script utilisé dans le cadre du WEB a pour vocation d'effectuer un certain nombre de traitements sur la génération de documents HTML, soit au niveau du serveur soit lors de la visualisation d'un document sur un navigateur côté client. Le premier langage développé pour ce faire fût Javascript, proposé par la société Netscape communications dès la version 2.0 de leur navigateur en 1996. JavaScript a fait l'objet d'une spécification européenne (ECMA-262)[217].
Javascript n'a concrètement presque rien à voir avec Java si ce n'est que l'apparition de ces deux langages est liée au développement du réseau Internet. Javascript est aujourd'hui utilisé majoritairement en complément au HTML pour tout ce qui a trait à la gestion d'événements ou au traitement dynamique de la page. Dans ce cas, le code Javascript étant inclu dans une page HTML, il ne permet de programmer des tâches qu'au niveau du navigateur recevant le document. Javascript est cependant utilisable également dans l'administration des serveurs. Depuis 1996, le langage Javascript a fait l'objet de quatre versions (1.3 aujourd'hui). Comme pour HTML, la compatibilité des navigateurs avec la version courante du langage peut poser problème dès que l'on est en relation avec des institutions dont la façon de travailler exclue la remise en cause annuelle ou bisanuelle des plateformes logicielles utilisées[218].
Avant de présenter les principaux aspects de Javascript, il nous faut insister sur son avantage majeur : Javascript permet de déporter sur le client un ensemble de traitements à effectuer sur le contenu ou la présentation du document qui lui est envoyé. Javascript permet donc de limiter le nombre de connexions réseau entre serveur et client au strict minimum. En cela, Javascript est sans doute le langage de programmation le mieux adapté au réseau Internet puisqu'il agit sur le temps de chargement d'un site sans pour autant faire appel, comme JAVA, au téléchargement souvent long de fichiers éxécutables dont par ailleurs l'éxécution peut poser problème sur telle ou telle machine cliente.
Les traitements côté client qu'autorise Javascript permettent, outre de gérer des évènements utilisateur par exemple, d'adapter le contenu de la page au client rencontré en effectuant sur une page téléchargée résultant d'un programme CGI des post-traitements contextuels. Javascript est donc un langage permettant d'annihiler en partie l'aspect statique d'un document HTML. Mais son apport s'arrête là puisque, nous le verrons, ce langage est par ailleurs plutôt pauvre et surtout ne permet pas d'utiliser les entrées sorties (côté client). Une programmation Javascript doit donc être comprise comme la mise à la disposition du client d'un ensemble de procédures et de variables dédiées à la gestion locale (côté client) du contenu (graphique et informatif) d'un document ou du caractère interactif de celui-ci.
Nous allons distinguer ci-dessous quatre aspects du langage : la programmation en Javascript, l'accès aux objets du DOM (Document Object Model), la gestion d'évènements et l'inclusion de code dans un document HTML[219].

[217] ECMA - Standardizing Information and Communication Systems, adresse : http://www.ecma.ch/
[218] Cette remarque n'est pas anecdotique dans le champ d'application qui nous concerne : les professionnels et chercheurs impliqués dans l'étude de l'édifice patrimonial investisssent peu de temps sur l'équipement et la mise à jour de leur plateforme informatique, et la multiplisation des versions de tel ou tel logiciel ou langage renforce leur défiance vis à vis des services que peut rendre l'outil informatique en général.
[219] Concluons cette rapide mise en perspective de Javascript en en rappelant les caractéristiques essentielles :
-Scripts interprétés par le client.
-Débogage à l'éxécution.
-Code encapsulé dans la page HTML (ou en référence).
-Notion d'objet présente pour l'accès aux propriétés du navigateur ou aux objets natifs Javascript mais pas POO
-Gestion d'évènements natifs Javascript associés à des balises HTML.

Les ouvrages traitant du langage Javascript sont légion. Nous avons cité [Cohen, 1996] et [Lecomte et al, 1996], nous renvoyons également aux auteurs du langage[220] pour plus de détail. Rappelons en terme d'applications que des travaux de recherche font usage de ce langage pour gérer la présenation de leurs résultats sur Internet. Nous renvoyons par exemple au projet ENVIART[221] (Baroque Artificial Marble : Environmental Impacts, Degradation and Protection), présenté en annexe, et dans lequel un ensemble de scripts écrits en Javascript prend en charge la mise à disposition du client des résultats de diagnostics sur l'état des marbres d'une chapelle Baroque au travers de choix utilisateurs.

*programmation en Javascript*

Comme langage de programmation, les principales caratéristiques de Javascript sont les suivantes :

- Des types de base : nombres (pas de distinctions apparentes entre entiers et flottants), chaînes de caractères, booléens,…
- Un typage faible: une variable peut à tout moment changer de type.
- Une syntaxe proche de celle du langage C : opérateurs classiques, instructions de boucle et définitions de fonctions.
- Des objets natifs : dates, calculs mathématiques, …
- Pas d'accès aux entrées-sorties côté client.
- Un modèle objet dénaturé : pas d'héritage dans les pseudo-classes que le programmeur peut définir.

Ce dernier point mérite sans doute d'être explicité. Javascript est un langage donnant accès aux propriétés d'objets natifs Javascript ou d'objets du DOM, nous le verrons dans la section suivante. Mais Javascript n'est pas un langage de programmation que l'on peut qualifier d'orienté objet puisque la seule caractérisitique "objet" qu'il propose est un ersatz d'encapsulation. A titre d'exemple, nous donnons dans l'encadré ci-dessous le code correspondant à la définition d'une possible pseudo-classe Cercle en Javascript. Nous pensons que cet exemple illustrera clairement les limitations du langage.

| | |
|---|---|
| Définition d'une pseudo-classe **Cercle** | Function Cercle(rayon) {<br>This.rayon= rayon;<br>This.surface = surface (This.rayon);<br>This.afficher= afficher();<br>} |
| Ecriture d'une méthode de calcul de sa surface faisant appel à l'objet natif Javascript Math | Function surface(rayon) {<br>Return (Math.pow(this.rayon, 2)*Math.PI);<br>} |
| Ecriture d'une méthode d'affichage textuel du cercle avec déclaration de variable locale | Function afficher() {<br>Var chaine= 'cercle de rayon :'<br>+ this.rayon() +' et de surface :'<br>+ this.surface() +'<br>';<br>} |
| Appel au constructeur du pseudo-objet et à une de ses méthodes | MonCercle= new Cercle(20);<br>MonCercle.afficher(); |

Il faut distinguer trois types de données sur lesquelles des traitements peuvent être écrits en Javascript: les variables et / ou pseudo-objets que le programmeur déclare dans son code, les objets du DOM et les paramètres issus de la soumission de formulaires par l'utilisateur. Les

[220] Voir le site: http://developer.netscape.com/docs/manuals/

[221] Projet de recherche et de développement européen (Technologies to protect and rehabilitate European Cultural Heritage) mené par l'Institut für Anorganische und Angewandte Chemie (Universität Hamburg, Allemagne) , l'Institut für Silikatchemie und Archäometrie (Hochschule für angewandte Kunst in Wien, Autriche), l'Institute of Catalysis and Surface Chemistry (Polish Academy of Science, Cracow, Poland), le Koninklijk Instituut voor het Kunstpatrimonium (Bruxelles, Belgique) et le KZA (Konserwacja Zabytków, Cracow, Poland).

évènements Javascript associés à des balises HTML permettent de lancer des traitements conjoints sur ces divers types de données et donc de manipuler assez aisément le contenu comme la présentation d'un document.

*Javascript et les objets du DOM*

Nous l'avons vu, le Document Object Model définit la structure logique d'un document et la façon de le manipuler. Chaque élément d'un document est dès lors un objet (au sens de la Programmation Orientée Objets) isolément manipulable par un langage de programmation. En javascript le programmeur a accès aux objets dits du navigateur, c'est à dire à une hiérarchie d'objets relatifs au document en cours. La racine de cette hiérarchie est la classe Window (fenêtre) qui fait référence à la fenêtre active du navigateur. La classe Document encapsule les propriétés globales du document (couleurs, tableau de formulaires, etc..) et des méthodes comme write qui affiche le document dans l'objet Window actif. Le tableau présenté ci-dessous, emprunté au CERN[222], récapitule les objets de base de la hiérarchie d'objets manipulables en Javascript. Chacun dispose de propriétés spécifiques que l'on adresse par la syntaxe suivante, assez proche de la syntaxe Java :

Objet.propriété (exemple : monDocument.monImage.name)

Autre exmple basique, un objet est instancié par la syntaxe suivante:

Instance= new Classe (paramètres)

La hiérarchie d'objets javascript permet de manipuler des propriétés relatives au contenu de la

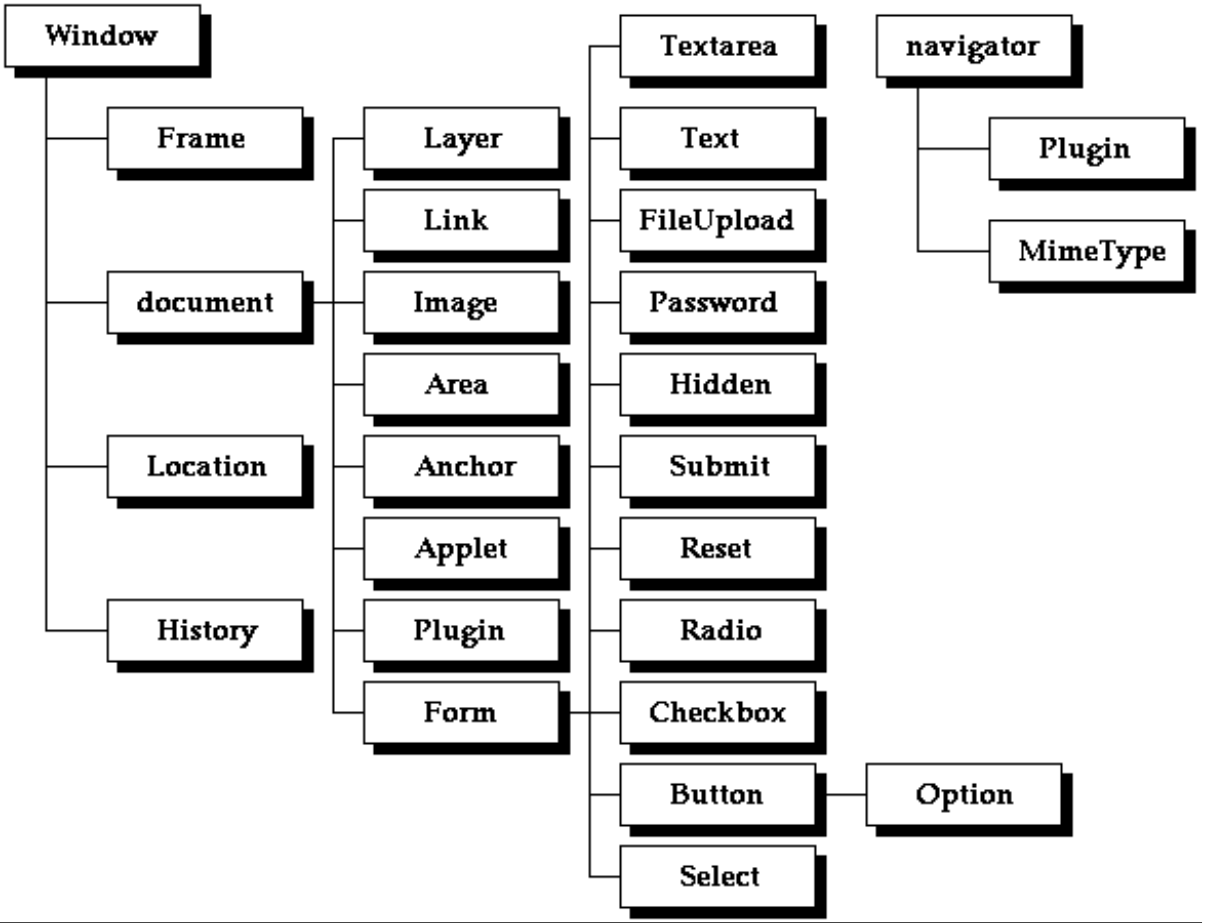
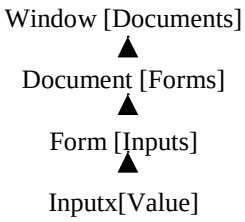



*Figure 111: Objets de base de la hiérarchie Javascript*

fenêtre courante (objet document, objet location qui donne l'url courante) mais aussi sur l'historique des pages visitées (objet history).

[222] CERN Web Office, http://dgrwww.epfl.ch/SIDGR/tutorials/JavaScript/hierarchy.html

Comme souvent dans cette hiérarchie, un objet (Form) est à la fois héritier d'un objet (Document) et propriété de celui-ci. Plus exactement, Form dérive de Document et Document a pour propriété un tableau d'instances de Form. Cette classe Form définit des ensembles de choix utilisateur traitables côté client en récupérant dans le code Javascript la propriété value d'une des propriétés Input de l'objet Form courant. On peut schématiser cette structure double héritage / composition de la façon suivante :




Chaque page dispose de ces objets de base :
- navigator: a pour propriétés la version du navigateur utilisé, les types MIME fournis et les plug-ins installés sur la machine cliente.
- window: propriétés attachées à la fenêtre dans son ensemble.
- document: propriétés relatives au contenu du document.
- location: propriétés relatives à l'URL courant.
- history: propriétés relatives aux URL que la machine cliente a visitée précédemment.

Les propriétés d'un document sont largement dépendantes de son contenu. Elles font en effet référence par exemple aux images ou aux formulaires contenues dans le document. Parmi ces propriétés, certaines jouent le rôle d'évènements, c'est ce que nous verrons dans la section suivante.

### *Gestion d'évènements*

En Javascript un événement est consécutif à une action de l'utilisateur sur un objet. Chaque objet décrit par le DOM dispose donc d'évènements associés spécifiques. Cet événement peut être associé à une fonction programmeur placée dans le document HTML ou en référence externe, ou bien encore à une méthode d'un objet (par exemple la méthode close() de l'objet Window qui ferme la fenêtre courante). La syntaxe générique d'appels d'évènements est la suivante :

```
<baliseHTML surEvenementRelatifACetteBalise= actionAssociee>
```

On l'a dit, l'action peut être décrite explicitement dans le même document mais aussi être décrite dans un document distant accessible par une adresse URL. Les évènements associés aux balises HTML peuvent être implicites côté client comme par exemple l'événement OnLoad auquel il est possible d'associer un ensemble de traitement au chargement de la page (typiquement on vérifiera la version du navigateur ou la disponibilité d'un plug-in). Ils peuvent également être masqués comme le déplacement du curseur sur une image contenue dans la page. Ils peuvent enfin être associés à des éléments de formulaires. On trouvera une liste complète de ces évènements ainsi que des balises associées dans les ouvrages cités comme sur beaucoup de sites Web dédiés à Javascript[223]. Bien que simple dans son principe, la gestion d'évènements en Javascript ne manque pas de poser des problèmes de compatibilité entre navigateurs. En effet, Javascript est implémenté de façon disons inégale selon le navigateur choisi, et la gestion d'évènements est un des aspects les plus diversement implémentés. De plus, avec quatre versions différentes en trois ans, Javascript se découvre dans l'instabilité une parenté certaine avec Java. Notons toutefois, pour qui en reste aux

[223] On peut se reporter aux pages de netscape communications consacrées au modèle d'évènements Javascript à l'adresse suivante: http://developer.netscape.com/docs/manuals/communicator/jsguide4/evnt.htm ou au JavaScript 1.1 Developer's Guide d'Arman Danesh pour la version 1.1 du langage (à l'adresse http://134.102.141.83/~leins/javascript/jsdfm.htm) publié par MacMillan , 1996, ISBN 1-57521-084-3

évènements les plus basiques, que Javascript offre des solutions pratiques légères et assez simples pour répondre aux actions utilisateurs, sans pour autant en passer par une connexion réseau et un programme CGI ou par un éxécutable Java.

*Javascript / HTML*

L'inclusion de code Javascript dans un document HTML recouvre deux réalités bien différentes: d'une part les lignes de code destinées à être interprétées par le navigateur comme du Javascript, et d'autre part les balises HTML auxquelles sont associées évènements et actions Javascript. Dans le premier cas, une balise HTML spécifique (<script>) est à placer dans l'en tête ou le corps du document selon que les instructions javascript contenues sont nécessaires ou non à l'interprétation du HTML. Plusieurs balises script peuvent être incluses dans le même document. Par ailleurs, un document ASCII indépendant de toute page HTML et contenant des instructions Javascript peut être adressé (extension.js) par un document HTML sous la forme d'une URL. Ce mécanisme permet de séparer ouvertement procédures et contenu et donc une meilleure réutilisation du code Javascript. Dans le second cas, l'appel à une méthode Javascript se traduit par l'adressage de celle-ci (par défaut dans le document courant) en affectation à l'événement associé. Javascript souffrant d'une limitation intrinsèque considérable (il ne permet pas de gérer les entrées / sorties côté client), ce langage est souvent sur de gros serveurs utilisé en complément aux documents HTML d'un côté et aux programmes CGI de l'autre.

On notera enfin que Javascript peut faire référence à un objet instance de la classe Applet, et propriété de l'objet Document. L'objet Applet donnant accès à toutes les variables et méthode publiques d'une applet, il est donné contrôle de l'applet au programme Javascript. Au delà de cela, L'objet Applet donne également accès à l'intérieur du programme Javascript à toute classe Java (du package Java identifié par le navigateur).

En conséquence, il est possible de faire référence à des variables ou méthodes publiques de chacune des classes en question, et bien entendu de faire appel à leurs constructeurs.

Notons enfin que des langages de script existent également dans le cadre d'interpréteurs de scènes VRML. C'est le cas de VRMLScript, aujourd'hui disponible uniquement sur le plug-in SGI Cosmo Player. On en trouvera une utilisation type sur le site web de l'université du Texas (Center for the Study of Digital Libraries)[224]. Le travail présenté là propose un ensemble de maquettes analytiques de terrains en 3D, rendues interactives par un ensemble de dispositifs d'interrogation et de manipulation côté client. Cette réalisation est exemplaire de ce qui est aujourd'hui autorisé sur le réseau Internet. par le fait qu'elle relie document HTML, scène VRML, langage de script et programme CGI. Nous aurons l'occasion de présenter en détail le travail que nous avons mené dans cette direction dans la partie de ce document consacrée au projet.

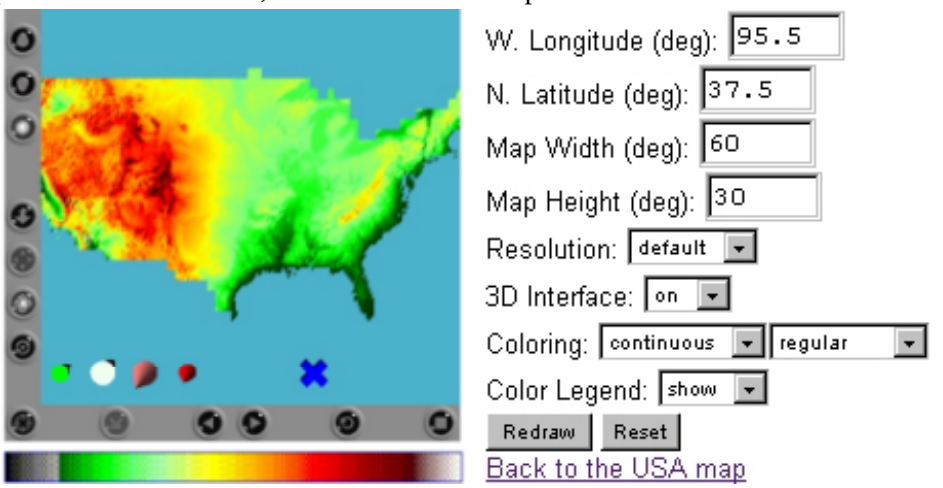



*Figure 112 :Une application HTML / VRML / VRMLScript /Javascript*

XML

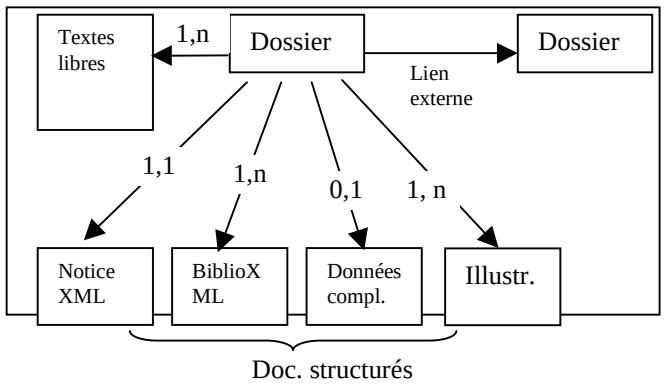
Nous allons ici dans un premier temps situer XML par rapport aux formalismes déjà présentés puis dans un deuxième temps tenterons, à partir du travail de recherche sur la réécriture, en XML, des bases de données du ministère français de la culture, d'en évaluer l'incidence sur notre travail.

[224] http://www.csdl.tamu.edu/topomaps/

XML est un langage de balisage héritier de SGML. XML n'est pas un langage de présentation de document mais plutôt un méta-langage à exploiter en spécifiant un sous-ensemble sur lequel travailler.
XML définit un jeu de règles et de conventions pour l'écriture de documents standards en format texte pouvant être échangés sur le réseau Internet. Comme HTML, XML utilise des balises et leur adjoint des qualifieurs appelés attributs (de la forme name="value"). Mais à la différence de HTML, les balises n'agissent pas sur le formatage de la page et ne sont pas attachées à une signification, mais délimitent une région du document (un sous-ensemble des données contenues dans le document). La même balise (au sens orthographique du terme) pourra donc en XML être interprétée différemment selon le contexte. Les règles syntaxiques de XML sont beaucoup plus strictes que celles du HTML, langage notoirement laxiste : un document XML comportant une erreur de balisage produira une erreur à l'interprétation.
Un document XML est constitué de deux parties, le document proprement dit et une description formelle, le Document Type Definition (DTD). Cette description contient les règles syntaxiques que doit respecter le document et la liste des éléments (ou concepts) disponibles pour le document. Chacun de ces concepts est représenté par un identifiant spécifique, l'ensemble constituant l'espace de noms disponible pour le document.
De nombreux outils de validation sont en cours de développement ou déjà proposés pour évaluer les documents XML, on se reportera au site web du consortium W3C pour suivre cette actualité.
XML a été défini par une première spécification (XML 1.0) du consortium W3C[225] en 1998, spécification suivie depuis par la définition, souvent encore non aboutie, d'un ensemble de formalismes dédiés à son exploitation :

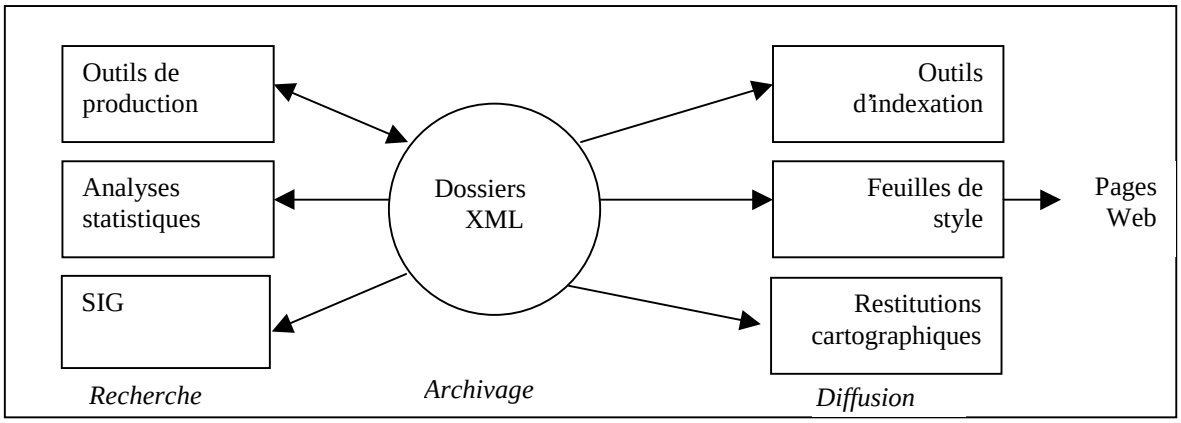
- Xlink (méthode standardisée de définition d'hyperliens dans un document XML).
- XPointer et XFragments (mécanisme de liens internes à un document XML proche des ancres de HTML).
- XSL (langage d'écriture des styles sheets, ou feuilles de style).
- DOM et CSS sont des standards dèjà présentés et repris en XML.
- XML Namespaces est une specification décrivant les associations balises - URL d'un document.
- XML Schemas sont des modules d'aide à la définition de formats basés sur XML.
- XML-QUERY est le langage d'interrogation et de manipulation de documents XML. XML-QUERY a la même vocation que SQL en bases de données[226].

XML s'appuie sur d'une part SGML et d'autre part l'expérience de HTML. Le choix de XML aujourd'hui apparaît comme une alternative intéressante au couple base de données / pages HTML. Utiliser XML requiert toutefois de définir une base de documents et des outils de manipulation, adéquats.
Les services de l'Inventaire du ministère français de la culture ont entamé une réflexion sur une refonte de leur fonds documentaire qui serait centrée sur l'utilisation de dossiers en langage XML.
La DTD en cours d'élaboration définirait un dossier d'inventaire canonique composé de divers éléments d'information textuelle ou graphique, structurée ou non (voir schéma ci-contre)[227]. Cette DTD est issue de la DTD EAD (Encoded




[225] Voir l'adresse suivante: http://www.w3.org/XML/
[226] Une bibliographie dédiée aux langages d'interrogation pour XML est disponible à l'adresse suivante: http://www.w3.org/TandS/QL/QL98/pp.html. Voir également le projet " SOLUTION XML + XQL " par S. MIRANDA (Université de Nice Sophia Antipolis) à l'adresse suivante: http://membres.tripod.fr/Locksley/ProjetXML.htm
[227] Voir aussi le site: http://www.culture.fr/BiblioML/

Archival Description) dont une description est disponible sur le site de la librairie du congrès américain (The Library of Congress )[228]. A terme (en l'occurrence plusieurs années selon l'estimation faite aujourd'hui par les développeurs engagé dans cette initiative), les dossiers XML de l'inventaire permettrait un archivage de données standardisées en XML et donnant lieu en amont à activités de recherche (formalismes de SIG, analyses statistiques, etc..) et en aval à diffusion sur le réseau Internet par le biais d'une banque de feuilles de styles.
Le schéma ci-dessous en récapitule l'organisation générale :




De nombreux développements sont aujourd'hui entamés autour du standard XML. Nous en avons présenté un ci-dessus, et fait référence à d'autres en notes de bas de page. Il n'en reste pas moins qu'XML est encore à l'heure actuelle avant tout une option pour l'avenir, et doit donc constituer pour le travail que je présente une perspective. Nous n'avons par compte tenu du caractère encore confidentiel d'XML choisi de nous lancer dès à présent dans cette direction. Nous notons cependant qu'une telle évolution semble nécessaire dans les années qui viennent.

[228] Voir le site : http://lcweb.loc.gov/ead/

## *ANNEXE 32 : ILLUSTRATION DES RÈGLES D'IDENTIFICATION DES ENTITÉS : LE CAS DES PLAFONDS EN BOIS*

La définition des entités architecturales que nous proposons s'appuie sur un ensemble de sources bibliographiques largement citées précédemment. Nous allons ici avec l'exemple d'une partie du corpus des plafonds en bois[229] illustrer notre démarche de modélisation.

### *ELEMENTS D'UN PLAFOND : LES SOURCES*

Jan Tajchman[230] distingue dans l'ouvrage qu'il consacre à ce sujet plusieurs classifications pour rendre compte des caractéristiques de ce corpus :

- Une classification par typomorphes (la poutre, la solive, l'entretoise, etc…).
- Une classification des modes d'assemblage (assemblages entre éléments de ce corpus et assemblages avec les murs).
- Une classification des profils (et un mode de description générique).
- Une classification des terminaisons de profils.
- Une classification des agencements de plafonds (et des planchers correspondant).

Il décrit par ailleurs un certain nombre d'éléments de décoration pouvant entrer dans la fabrication des typomorphes (rosettes de poutres ou de solives) ou pouvant être plaqués ultérieurement (fresques par exemple).

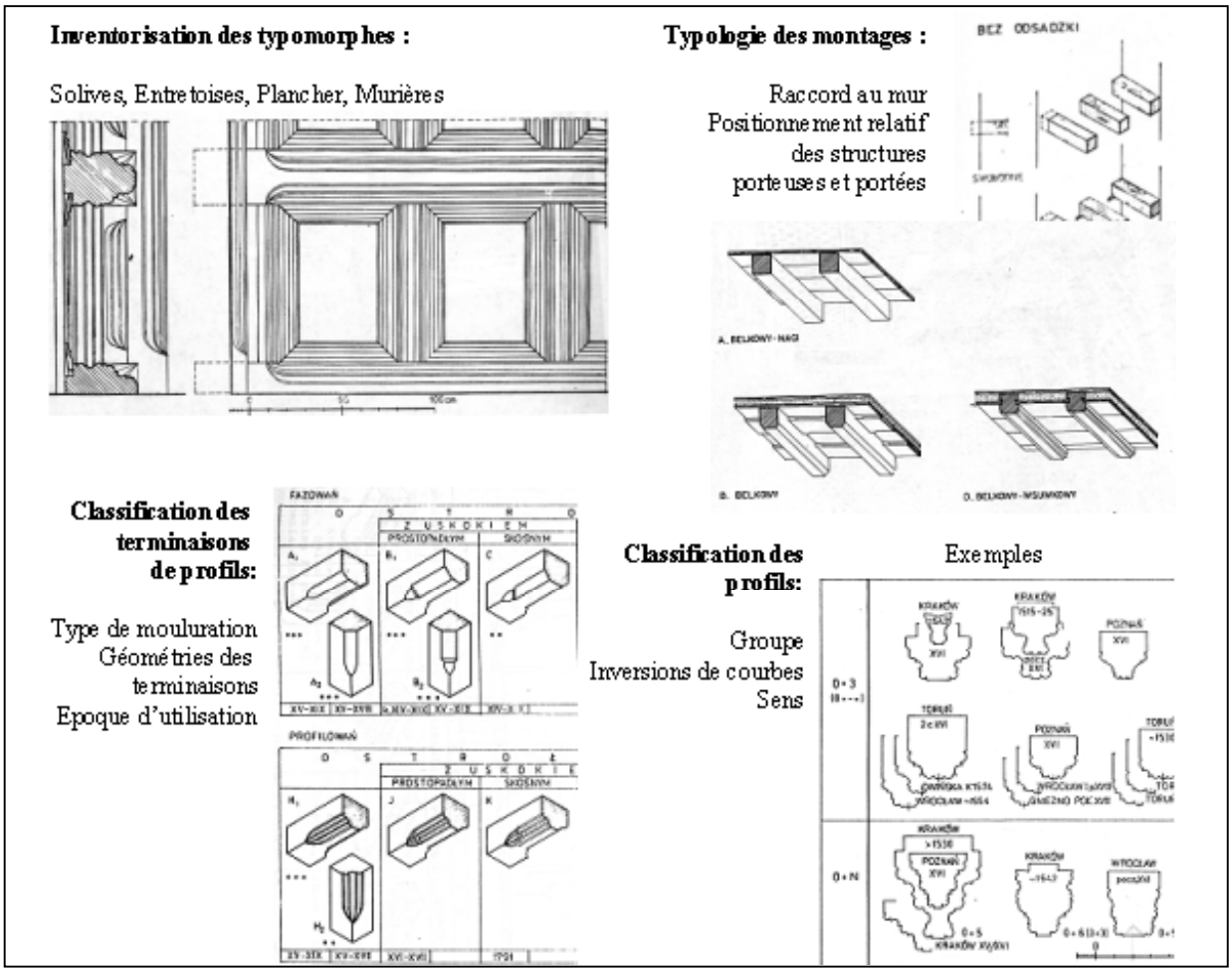



*Figure 113 : Classifications d'un corpus, le travail de Jan Tajchman sur le corpus des plafonds en bois [Tajchman, 1989]*

[229] Précisons que nous parlons du corpus des plafonds en bois bâtis en Pologne entre les treizième et dix-huitième siècles (dits de période gothique, renaissance et baroque), et utilisés en majorité dans les maisons urbaines.
[230] Voir [Tajchman, 1989], habilitation en architecture, université de Torun (Pologne)

Chaque classification correspond à un point de vue différent sur ce jeu étroit d'objets que forment les plafonds : l'auteur effectue un recensement dont il tire un ensemble de règles permettant de décrire de façon indépendante tel ou tel aspect particulier. En effet, le travail de J.Tajchman est centré d'une part sur l'analyse des techniques mises en œuvre aux différentes périodes de construction, et d'autre part sur la description des éléments de modénature pouvant aider le conservateur à repérer et à classer chronologiquement un élément de plafond même isolé ou réemployé de façon incongrue.
On doit ici ouvrir une rapide parenthèse pour indiquer qu'aujourd'hui de nombreux éléments de plafonds sont exhumés hors contexte. Comment dès lors savoir à quel plafond, et plus encore à quelle période de l'évolution de l'édifice, cet élément isolé correspond ? Les hypothèses quant à une possible origine à ces éléments isolés s'appuient dans la pratique sur deux familles d'indices : indices matériels découverts notamment sur les murs, comme par exemple les traces laissées par la présence de murières ou les ancrages des solives, indices documentaires révélant par exemple les dates de transformation d'un édifice. A la croisée des deux approches, l'ouvrage de J.Tajchman récapitule les types d'indices permettant de formuler une hypothèse raisonnable sur l'origine d'éléments isolés soit par comparaison avec des exemples que l'auteur cite soit par comparaison avec les modèles qu'il propose. Néanmoins la préoccupation affichée par l'auteur n'est pas tout à fait la nôtre : nous ne souhaitons pas étudier indépendamment la solive et son profil, nous ne considérons pas la chronologie comme une inconnue mais comme un attribut renseigné ou non. Nous ne reprenons donc pas ses classifications mais nous appuyons sur plusieurs points-clés de son travail :

- Sa classification des éléments invariants (poutres, solives, entretoises, plancher).
- Sa classification des assemblages intra-invariants dans la détermination de ceux-ci.
- Son formalisme de description des profils.
- Son formalisme de description des terminaisons de profils.

Nous reviendrons sur ces deux derniers points ci dessous, en illustration du concept d'attribut.
J.Tajchman insiste sur le lien de dépendance observable entre le profil, sa terminaison, les modes d'assemblages et la datation du plafond observé. En cela il indique de fait un point important qui vient à l'appui de notre démarche de modélisation : un élément comme la solive est un concept invariant dans ce corpus de plafonds en bois. Par contre, sa mouluration intervient pour qualifier l'instance et lui donner une morphologie particulière en fonction dans ce cas de la période de construction. Le modèle que nous proposons s'attache donc à dégager des invariants (une hiérarchie d'entités) et à les doter d'attributs permettant de les analyser (les profils par exemple). J.Tachman, en collaboration avec M.Łukacz, a proposé en parallèle aux travaux présentés dans [Tajchman, 1989], une classification plus synthétique, non publiée, des types de plafonds. Nous allons la décrire sommairement, indiquer quels points nous semblent poser problème (dans le cadre de notre travail), et enfin présenter notre interprétation.
J.Tachman et M.Łukacz définissent quatre catégories de plafonds notées A,B,C,D.

- La catégorie A correspond aux plafonds dont les solives et les planches portées sont posées parallèlement. Il s'agit donc d'une catégorie isolée sur la base du type de montage du plancher, type dont la conséquence sur la morphologie des solives est la présence de deux encoches permettant de glisser entre deux solives et de caler les planches portées.
- La catégorie B correspond aux plafonds dont les solives et les planches portées sont posées perpendiculairement, quel que soit le type de montage (planches jointives ou alternées). La catégorie B se distingue de la catégorie A par une direction de pose du plancher opposée, mais le type de montage correspondant à la catégorie B n'est pas unique, et ses conséquences sur la morphologie des solives sont variables. En résumé, si A affirme un type, B affirme seulement une différence avec A.
- La catégorie C correspond aux plafonds dans lesquels le sens de pose du plancher est le même que pour B, mais dans lesquels des entretoises intermédiaires sont posées entre les solives (cas particulier des plafonds dits à caissons). Cette catégorie n'est donc plus isolée par la seule observation du sens de pose des planches mais par la présence ou non d'éléments tiers, les entretoises intermédiaires.

- La catégorie D correspond aux plafonds dans lesquels le sens de pose du plancher est le même que pour B, et dans lesquels des entretoises intermédiaires sont posées entre les solives. Quelle différence avec C ? Les plafonds de la catégorie D ne se distinguent de ceux de la catégorie C que par une mouluration plus importante des solives, mouluration prenant naissance ici directement sous les planches portées. Cette dernière catégorie n'est donc isolée que par l'observation de particularités morphologiques tenant du décor (et non de la structure).

Avant de commenter ces choix, présentons les sous-catégories de A,B,C et D. Dans les catégories A et B figurent deux groupes de plafonds, notés I et II. Les premiers correspondent aux plafonds dont les éléments moulurés le sont sous la forme de chanfreins. Les seconds correspondent aux plafonds dont les éléments moulurés le sont sous la forme de profils a inversions de courbes plus ou moins complexes. Les plafonds de catégorie C ou D sont censés être tous du groupe II (profils à inversions de courbes). Enfin, un troisième niveau de catégorisation note la façon dont les solives sont assemblées au mur : assemblage par simple

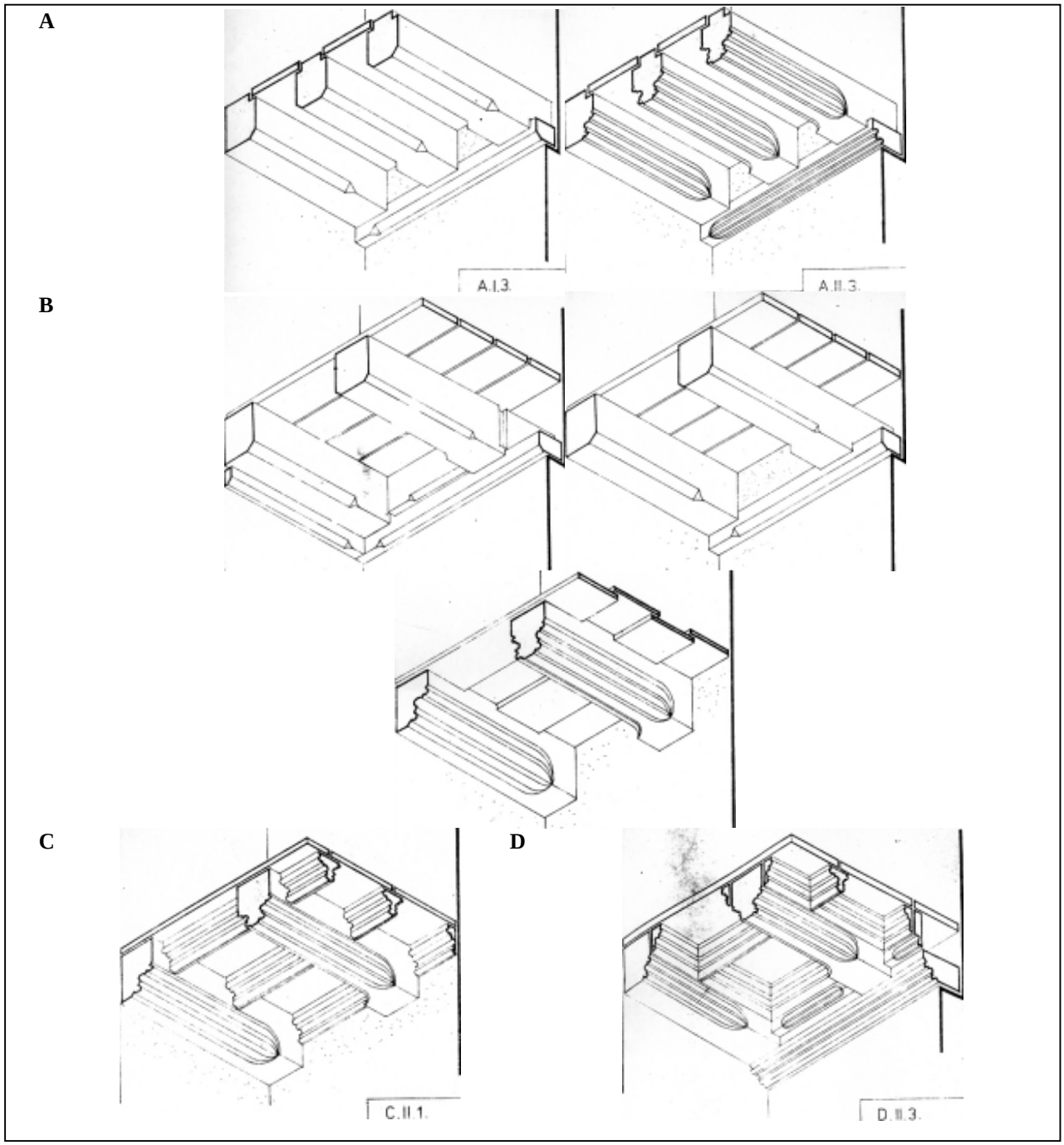



*Figure 114 : Quatre catégories de plafonds selon J.Tajchman et M.Łukacz.*

ancrage, sur murière noyée, sur murière moulurée et en avancée, avec entretoise ou hourdis, etc… Il n'est pas fait mention dans cette classification du type de terminaisons de profils ni des plafonds à gorge (planches portées glissées dans des gorges ouvertes sur les côtés d'une solive et couvrant toute sa longueur). Pour nous, même si cette contribution constitue une formidable base de travail, la logique de classification proposée ne peut être prise en compte telle quelle. En voici les quatre raisons principales :

- Le point de vue sur la collection d'objets au nom duquel une classification est élaborée change en fonction de la catégorie.
- L'échelle des éléments justifiant les lignes de division du modèle est variable : type de montage de l'ensemble entre A et B, présence ou non d'un élément entre B et C, étendue de la surface moulurée d'un même élément entre C et D.
- Non prise en compte de la totalité des individus observés (plafonds à gorge).
- Indifférenciation des éléments structurels et des éléments de décor.

A la lecture des travaux cités, il nous est apparu en fait que quatre préoccupations distinctes devaient être prises en compte :

- La logique structurelle du plafond et de son raccord aux murs.
- La morphologie des objets en jeu et les conséquences du point précédent sur celle-ci
- Les éléments de décoration et leur description générique (mouluration, terminaisons, rosettes, fresques, etc…).
- Les rapports des trois points précédents avec une classification chronologique des plafonds.

*CLASSIFICATION DES ELEMENTS D'UN PLAFOND*

A partir de ce constat, il nous fallait trouver un moyen de représenter chaque problème au sein d'un modèle unique. Cette démarche s'apparentait à la recherche d'un jeu de concepts permettant d'exprimer qu'un plafond est une structure qui offre plusieurs niveaux de lecture. Le corpus des plafonds a par conséquent été une excellente occasion d'évaluer notre démarche de modélisation (réseaux-entités-attributs), et traduit bien les difficultés que nous pouvons rencontrer à isoler l'échelle des concepts architecturaux au sein du problème global que pose un ensemble bâti. Nous avons donc choisi d'établir conformément à l'approche présentée à la section précédente une classification d'entités architecturales pertinentes à l'intérieur desquelles nous avons cherché à définir des propriétés correspondantes aux points de vue mentionnés ci-dessus.

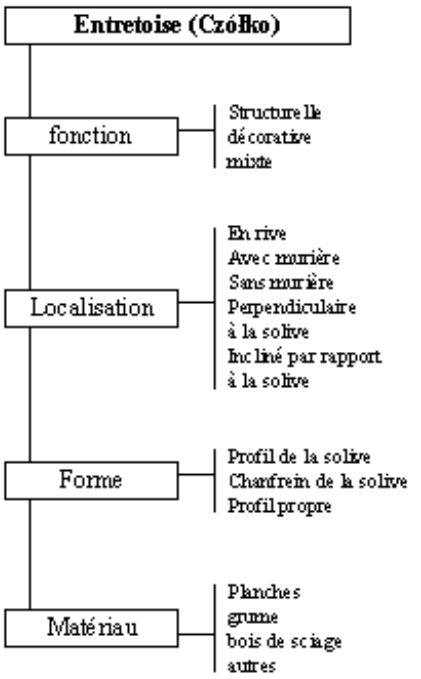



*Figure 115 : liste de déterminants dans le cas de l'entretoise, une méthode de détermination de l'entité architecturale*

Nous avons commencé par définir un jeu d'entités puis avons croisé les différentes classifications présentées ci-dessus afin d'établir pour chaque entité une liste de déterminants (voir schéma ci-contre). Ceux-ci nous ont permis de noter par exemple l'impossibilité d'affecter aux entretoises les mêmes profils qu'aux solives, ou encore de considérer ces entretoises comme un élément de décor ne relevant pas de la hiérarchie d'entités. Autrement dit, la formulation de cette liste de déterminants, établie pour tous les objets ou concepts décrits par [Tajchman, 1989], nous a permis de choisir les concepts considérés comme entités architecturales, et ceux que nous gérons dans les hiérarchies d'attributs ou (potentiellement) de réseaux.

Conformément à notre démarche de modélisation, le jeu de relations et d'entités en jeu dans un plafond doit être géré par un réseau, réseau qui porte les informations relatives à l'ensemble.

Nous présentons plus loin un schéma présentant à la fois les classes situées à la tête de la hiérarchie d'entités architecturales relatives au corpus des plafonds en bois et les principales classes des autres hiérarchies formalisées pour représenter tel ou tel aspect du problème. Les

entités architecturales représentées dans cette hiérarchie sont des objets disposant de méthodes et de propriétés permettant d'une part de gérer les questions de représentation (nous revenons sur ce point dans la description des interfaces) mais aussi de rendre compte de la classification de Tajchman et Łukacz :

- Des propriétés correspondant à leur classification peuvent être renseignées.
- Profils et Terminaisons sont décrits dans un formalisme qui reprend la typologie qu'ils établissent, et sont exprimables en tant que concept autonome par rapport aux entités.

Les objets décrits par les classes de la hiérarchie des entités architecturales héritant de la classe Solive sont donc isolés et organisés avec pour guide la définition des entités que nous avons proposée, mais intègrent sous forme d'objets imbriqués les classifications de modénature issues des sources bibliographiques. Le schéma ci-contre explicite ce principe dans le cas de la Classe BelkaZCzółkami (représentant les solives entretoisées) :

- Le concept hérite de propriétés définies dans ses classes ancêtres.
- Trois objets représentent les notions liées à la gestion de la modénature de l'objet.

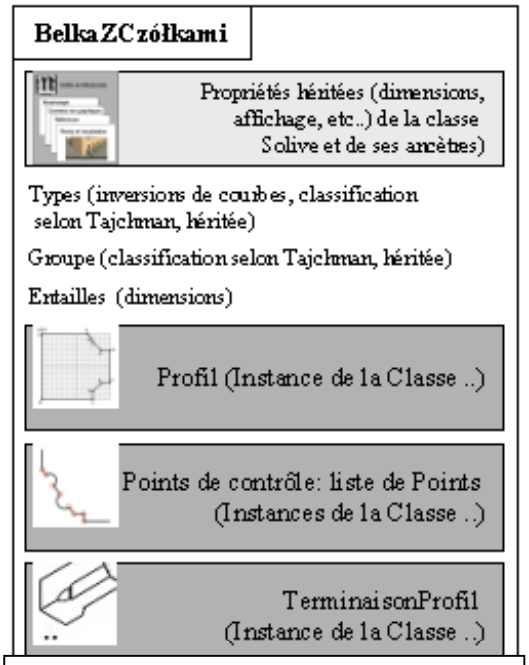



*Figure 116 : La classe BelkaZCzółkami, une spécialisation par ajout de propriétés, une agrégation de propriétés liées à des points de vue sur l'objet*

Pourtant, un point de vue introduit par J.Tajchman reste aujourd'hui à exprimer dans le travail que nous proposons : la notion d'assemblage. En effet, elle recouvre ici à la fois l'idée de positionnement relatif des objets, idée dont nous reparlerons ci dessous (voir "relations"), mais aussi celle de typologie d'assemblages. A l'heure actuelle, nous n'avons pas élaboré de formalisme adéquat puisque seules les entités (intégrant les particularités morphologiques liées à ces types d'assemblages) font partie de notre modèle. Nous avons préféré dans un premier temps développer un formalisme de description générique des profils, mais plaçons en perspective la détermination d'un formalisme d'assemblages qui rende mieux compte de la dépendance structurelle des éléments d'un plafond.

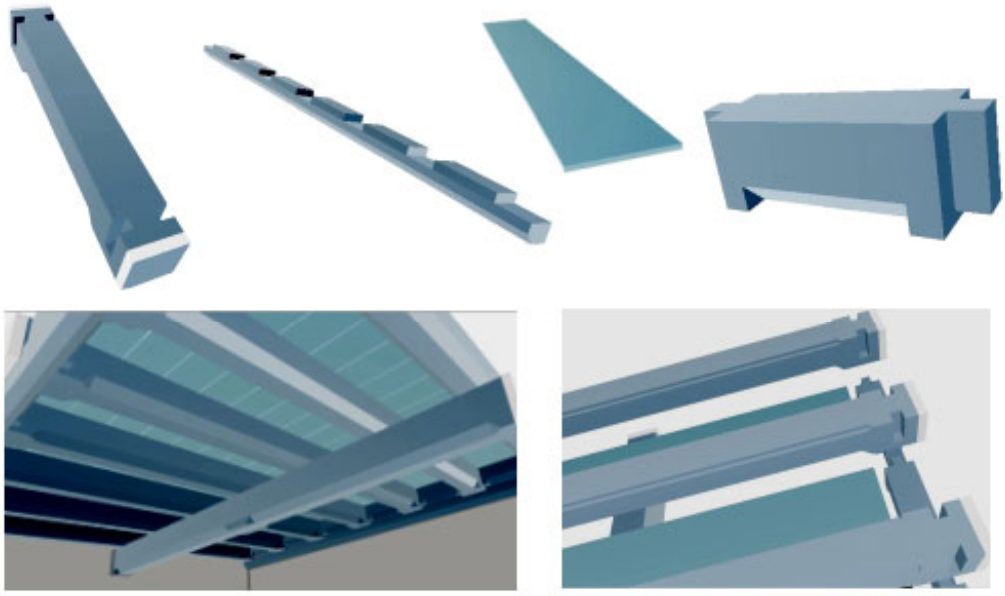


*Figure 117 : Visualisation d'éléments du modèle, entités et réseaux, sur le corpus des plafonds en bois (VRML, im auteur)*

## ANNEXE 33 : LE CORPUS DES PLAFONDS EN BOIS, HIÉRARCHIE DE CLASSES PARTIELLE

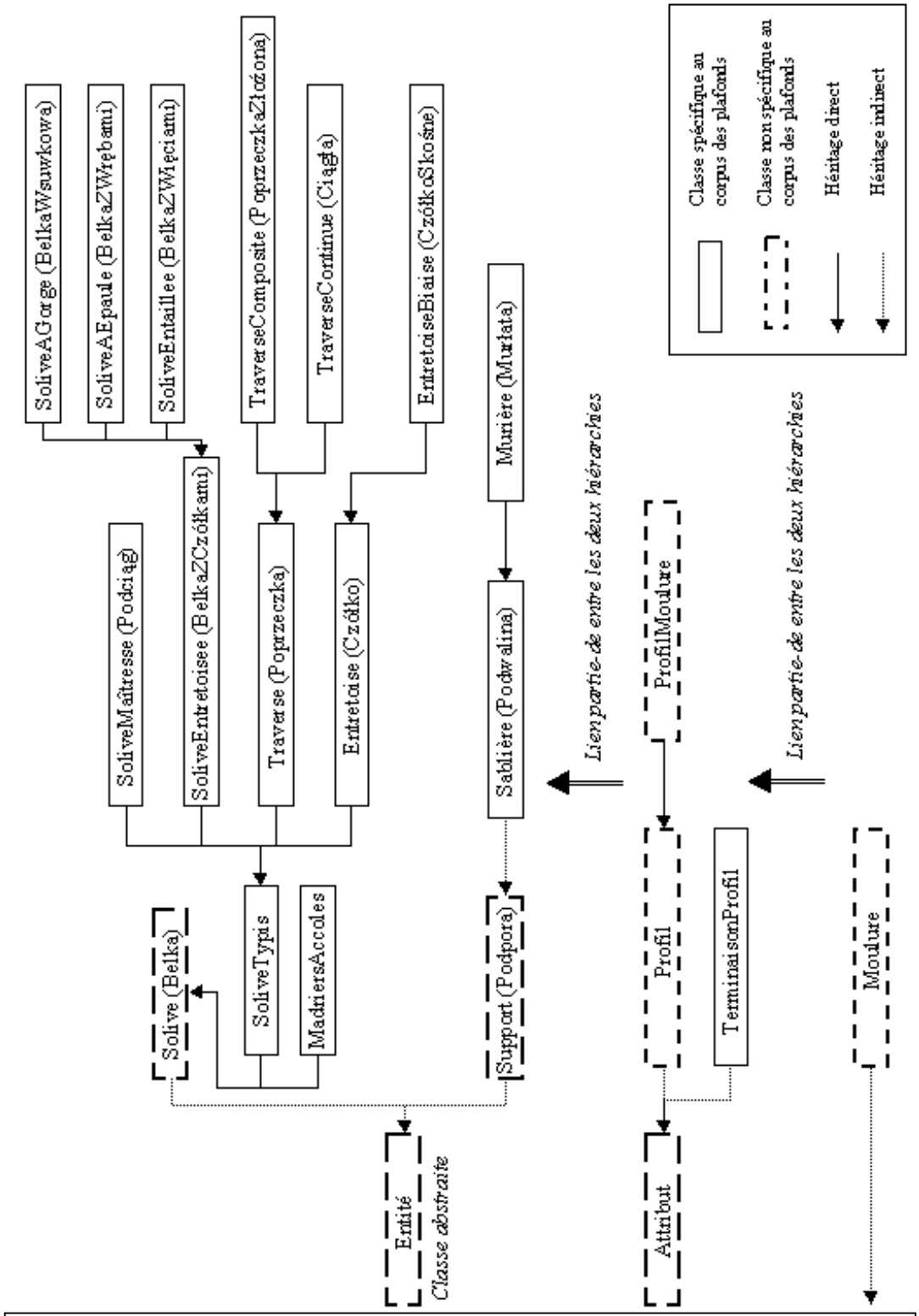



*Figure 118 : Hiérarchie des principaux concepts architecturaux en jeu pour représenter le corpus des plafonds en bois.*

## ANNEXE 34 : TÊTE DE LA HIÉRARCHIE D'ENTITÉS

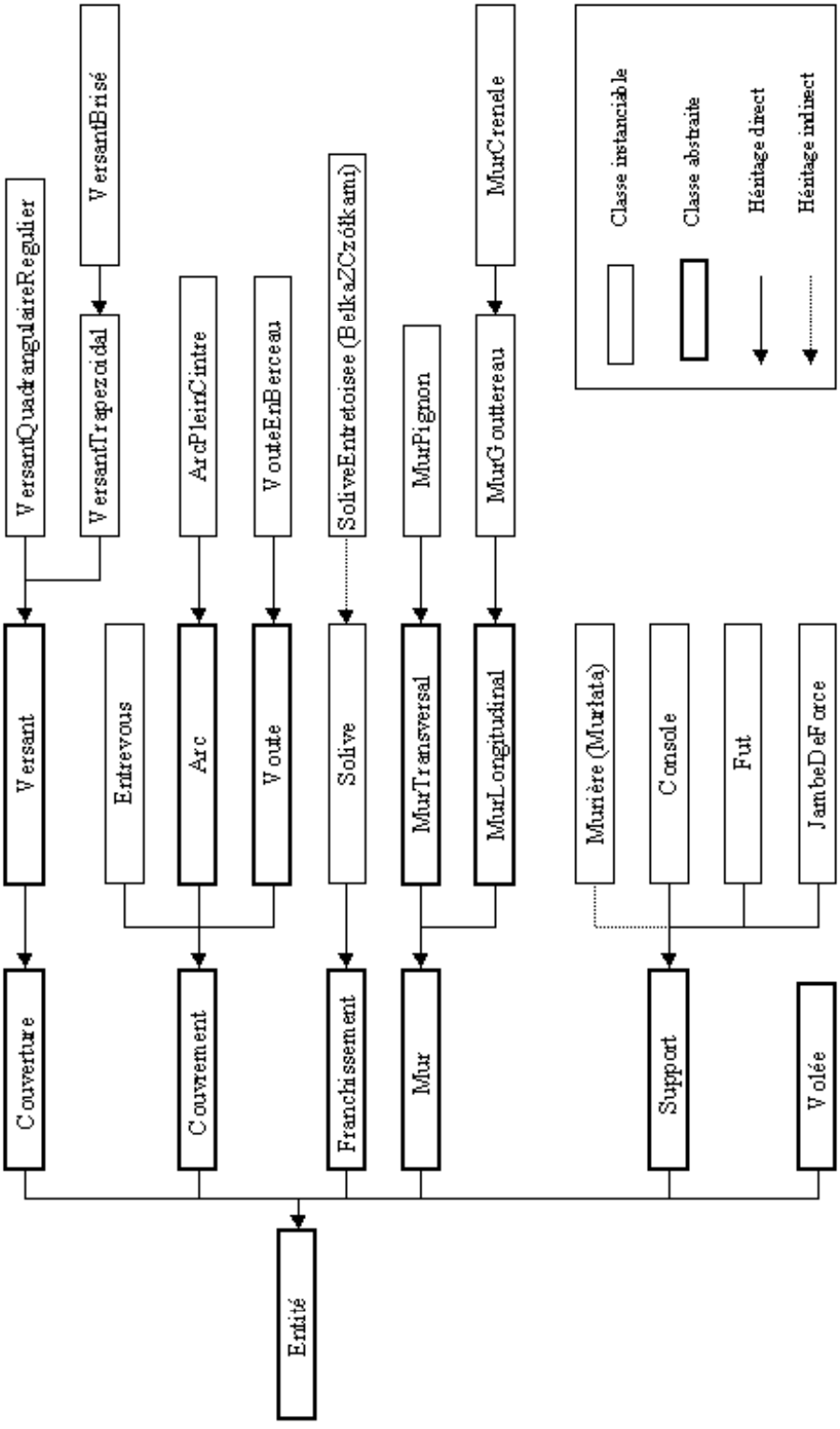



*Figure 119 : Hiérarchie des Entités architecturales, niveau de dérivation fonctionnelle de la classification et premier niveau de dérivation morphologique*

Remarque : Cette figure ne présente ni la totalité des concepts dérivables d'entités, ni la totalité des classes dérivées effectivement de la classe Entité dans la partie implémentée du modèle.

## ANNEXE 35 : ILLUSTRATION DES RÈGLES D'IDENTIFICATION DES DÉPENDANCES DE POSITION

Une relation architecturale est repérée par un vocable établissant une correspondance entre un ou plusieurs objets architecturaux, ou groupes d'objets architecturaux. La sémantique de cette relation ne relève pas de la seule discipline architecture mais plutôt du concept d'échelles que définit P.Boudon [Boudon, 1971] [Boudon, 1977]. La relation architecturale est une relation d'ordre rapportant à un concept architectural un ou plusieurs concepts architecturaux eu égard à un objectif d'analyse. Elle établit qu'un ou plusieurs concepts architecturaux sont à relier à une autre notion pour renseigner un élément de sémantique particulier du modèle. La relation architecturale est une analyse du modèle permettant de considérer un groupe de concepts comme porteur d'un sens que ses membres isolés n'ont pas. La relation détermine la logique d'intégration d'un groupe. Est elle un concept ? Concrètement, du point de vue de l'implémentation, la réponse est oui dans le cadre des expériences que nous menons aujourd'hui puisqu'un ensemble de classes au sens de la Programmation Orientée Objets lui confère une existence comparable à celle des autres concepts du modèle. Pourtant, elle renvoie à la question de l'existence réelle des concepts abstraits, ou idées, telle que la rapportent [Ducournau et al, 1998, p7]. Nous partirons de l'hypothèse que la relation est bien un concept indépendant des individus auxquels elle s'applique.
Une fois cette hypothèse émise, il nous faut l'illustrer par des exemples:

- Mur isolé flanqué de colonnes sur les deux faces [Viollet Le Duc, 1863/1977, p297].
- Ni l'Acropole d'Athènes, ni le Forum romain (…) ne nous donnent des dispositions symétriques d'ensemble [Viollet Le Duc, 1863/1977, p254].
- En bourgogne, l'architecture romane du XIIème siècles est toute clunisienne, en Champagne, elle est plutôt cistercienne, …[Viollet Le Duc, 1863/1977, p277].
- Ils (*i.e les Grecs*) ont incliné les colonnes d'angles vers l'intérieur (…) et ils ont donné à ces colonnes d'angle un diamètre plus fort [Viollet Le Duc, 1863/1977, p290].

Ces exemples montrent en premier lieu qu'une relation s'établit soit entre deux concepts dont l'un sert de référent (le mur, référent de deux colonnes) soit entre un groupe et un tiers, sans rapport avec lui (une disposition symétrique, référence à un axe de symétrie). Ils montrent en second lieu qu'une relation s'établit en fonction d'un angle d'analyse, s'adressant aux propriétés quantitatives ou qualitatives des objets liés[231] :
Enfin, ces exemples montrent qu'une relation architecturale intervient dans la détermination de telle ou telle propriété des concepts en jeu (positionnement par exemple) ou seulement dans l'évaluation-comparaison de ses propriétés déjà renseignées (comparaisons dimensionnelles par exemple). En admettant l'hypothèse selon laquelle la relation peut être réifiée, nous nous heurtons à la difficulté de prendre en compte les trois lignes de division que nous venons d'illustrer :

- Relations unaires, relations n-aires.
- Relations quantitatives / qualitatives.
- Relations déterminantes des objets observés / comparaison d'états.

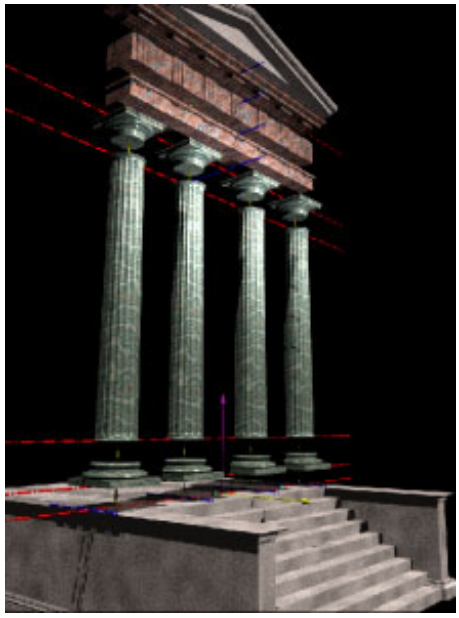


*Figure 120 : Relations d'axialités réglant l'ordonnance d'un portique dorique théorique (rendu POV, im auteur)*

[231] -Mur isolé flanqué de colonnes sur les deux faces: la position des éléments cités.
-Ni l'Acropole d'Athènes, ni le Forum romain (…) ne nous donnent des dispositions symétriques d'ensemble: l'étude de la composition d'ensemble.
-En bourgogne, l'architecture romane du XIIème siècles est toute clunisienne, en Champagne, elle est plutôt cistercienne: une analyse stylistico-historique.
-Ils (i.e les Grecs) ont incliné les colonnes d'angles vers l'intérieur (…) et ils ont donné à ces colonnes d'angle un diamètre plus fort : la comparaison d'entités architecturales en fonction de leur position.

Nous nous sommes restreints dans le travail présenté ici à la mise en œuvre de mécanismes de positionnement relatif des objets architecturaux. Nous proposons un formalisme de relation dans lequel une correspondance générique est établie entre des objets observés et un concept tiers, objet physique ou être architectonique. Nous l'expérimentons sur le cas des dépendances topologiques présentées ci-après.

*DEPENDANCES TOPOLOGIQUES*

Les dépendances topologiques, ou dépendances de position, correspondront pour nous aux vocables aplomb, au droit, flanquer, etc… Ces relations ont trait aux dispositions relatives d'entités architecturales eu égard soit à des axialités soit à des alignements, concepts tiers de la relation. Dans un cas comme dans l'autre le vocable désigne une contrainte fixant les positions relatives de deux entités le long d'un axe principal, mais reste imprécis sur les positions relatives de deux entités le long des autres axes. Nous allons ci-dessous recenser les relations d'axialité ou d'alignement que nous prenons en compte en indiquant graphiquement ce que le vocable contraint, et en précisant textuellement ce qu'il ne contraint pas.

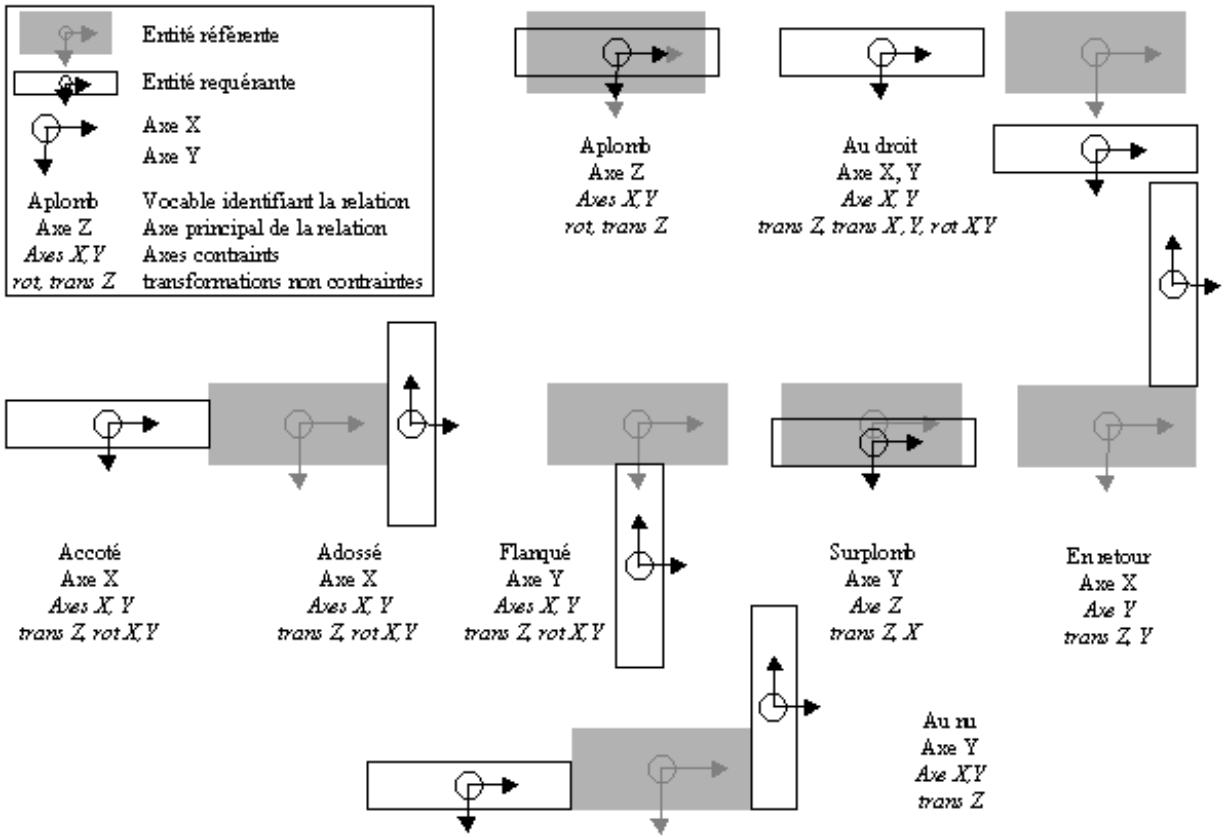



*Figure 121 : Dépendances topologiques telles que les contraignent les vocables concernés*

Nous proposons dans le schéma suivant notre interprétation visant à lever les ambiguïtés soulignées précédemment.
Nous reprenons les mêmes graphiques et plaçons cette fois-ci en commentaire textuel les paramètres joints aux méthodes reprenant les vocables.

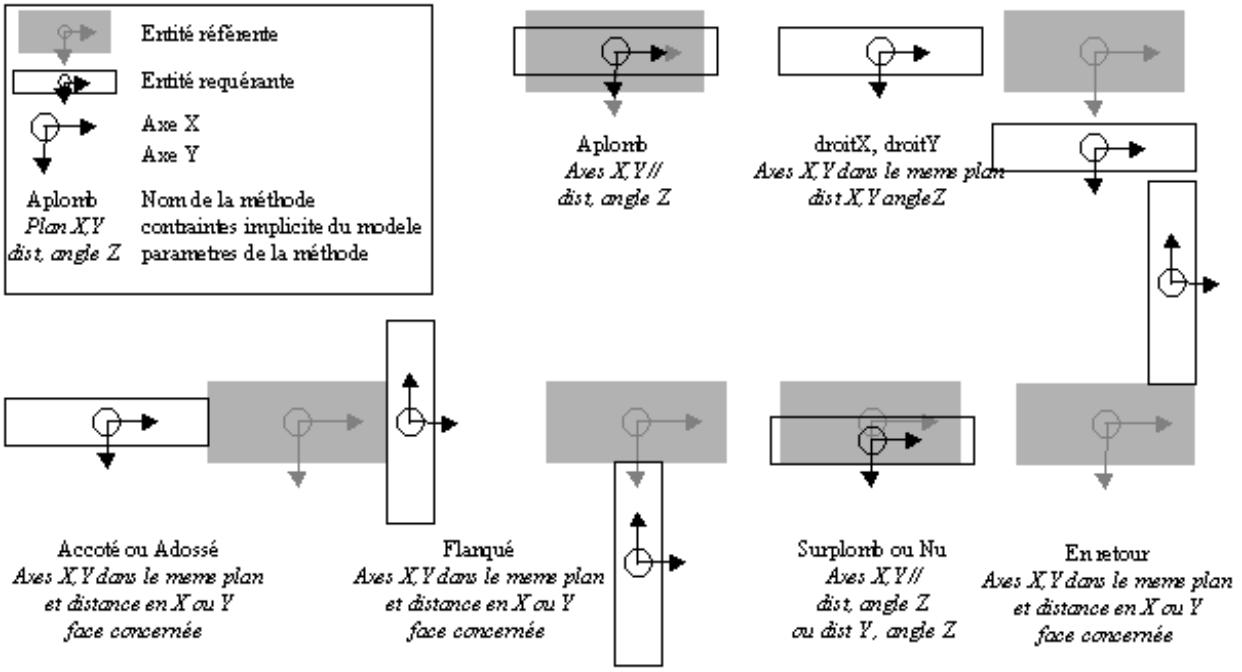



*Figure 122 : Interprétation des vocables de positionnement, paramètres additionnels nécessaires à la définition de relations non ambiguës.*

## ANNEXE 36 : ILLUSTRATION DES RÈGLES D'IDENTIFICATION DES ATTRIBUTS

Nous présentons ici trois classes dérivant de la classe Attribut, classes qui nous semblent représentatives des différents problèmes traités par les attributs. Le premier attribut présenté, la scotie, est une moulure utilisée largement aux périodes antique, renaissance et néo-classique. Sa particularité tient à un tracé dont il nous semblait important de rappeler le mode de définition dans la pratique: une succession d'arcs dont les centres se décalent.
Le second attribut présenté, la moulure, propose un formalisme générique de moulure permettant de gérer le troisième attribut présenté, le profil, comme une succession d'objets de type "moulure" dont les tracés deviennent dès lors indépendants les uns des autres. Il faut rappeler qu'ici notre objectif est double :

- D'une part, intégrer ces notions au modèle.
- D'autre part, faciliter leur représentation tridimensionnelle en définissant un formalisme de points de contrôle / liste de segments intermédiaire. Ce formalisme permet de prendre en compte les contraintes liées à la visualisation (niveau de détail en particulier) mais il permet également de représenter le mode de description théorique des moulures tel que la bibliographie nous le transmet.

Il nous semble en effet pertinent dans le cadre d'un modèle architectural dédié au patrimoine de prendre en compte non seulement la connaissance que nous avons des objets observés mais la façon dont leurs auteurs les concevaient. Il nous semble donc pertinent d'exprimer par exemple une mouluration en deux dimensions (celles de leurs représentations traditionnelles, voir l'image de la doucine ci-contre [Barberot, 1922, p152]) quand elles sont à l'observation tridimensionnelles. Les méthodes de tracé que de telles figures font en effet pleinement partie de cette connaissance sur l'édifice patrimonial que nous souhaitons traiter.
Par ailleurs, ces méthodes, le plus souvent basées sur le tracé d'une succession de cercles facile à mettre en œuvre "avec les moyens du bord" sur un chantier sont souvent également utilisées par l'artisan dans la construction de l'objet concerné.

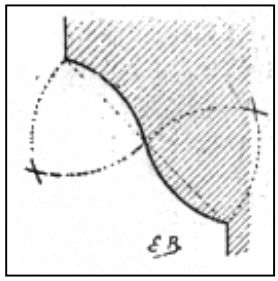


*Figure 123 :Tracé d'une doucine [Barberot, 1922, p152]*

### LA SCOTIE

Jean-Marie Pérouse de Montclos donne de cette moulure la définition suivante : *gorge à profil demi-ovale formé de plusieurs segments dont les rayons sont de longueur croissante*…[Pérouse De Montclos, 1988, p184]. Toutes les définitions n'ont pas la même précision, Pierre Noël ne voit dans cet objet qu'une *Moulure dont la partie inférieure est plus saillante que la partie supèrieure* …[Noël, 1994, p318]. Nous avons souhaité rendre compte de la méthode de construction de cette moulure telle que la décrit [Barberot, 1922, p153], méthode illustrée par le schéma ci-contre et qui reprend la logique de tracé décrite par Jean-Marie Pérouse de Montclos.
L'importance des méthodes de tracé graphique des moulures est attestée par toute la littérature sur le sujet que nous avons cité, nous ne revenons donc pas sur ce point.
La scotie définit une sorte de moulure qui a pour propriétés spécifiques:

- Une méthode de tracé (graphique).
- Des points de contrôle (de tangence entre arcs).
- Des points utilisés comme centre pour ce tracé.

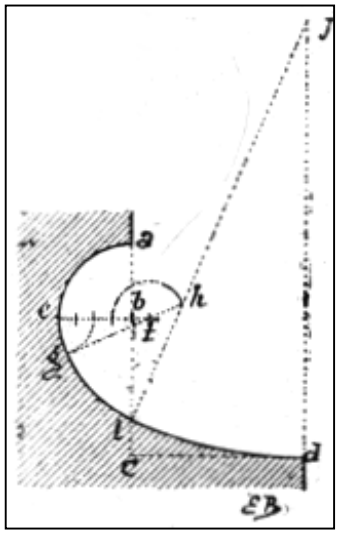


*Figure 124 : Tracé de la scotie selon[Barberot, 1922, p153]*

Un objet instance de la classe Scotie est totalement renseigné en terme de dimensions par :

- Le renseignement de ses propriétés dimensionnelles héritées, à savoir les points extrémités de son tracé (points a et d sur le schéma), son référentiel et sa boîte englobante.

– L'exécution de la méthode extractionProfil qui affecte une valeur à l'ensemble des points de contrôle et des points centres propriétés de l'objet.

La méthode extractionProfil renseigne également la liste de points permettant de générer la représentation de la scotie. Cette liste de points, nommée ligne, est héritée de la classe moulure. Elle gére une liste de points de contrôle (ou points de modification de courbure) entre lesquels un segment ou une liste de segments seront tracés en fonction du type de maquette recherché. Autrement dit, cette méthode applique à l'intérieur de la scotie la méthode retenue pour la représentation des moulures, à savoir deux points de contrôle que joignent un nombre indéterminé de segments (en fait, un nombre considéré comme paramètre du modèle théorique).

*LA MOULURE*

Jean-Marie Pérouse de Montclos explicite avec une parfaite clarté la pensée présidant au dessin d'une moulure : *ornement … créé par la translation d'un profil élémentaire selon une directrice*…[Pérouse De Montclos, 1988, p179]. L'auteur définit un profil comme *la ligne délimitant la section orthogonale d'une moulure…* [Pérouse De Montclos, 1988, p179], sa définition indique bien que la moulure est pensée en deux dimensions. Toutefois, elle ne fait pas référence aux ornements obtenus par révolution d'un profil autour d'un axe. Par ailleurs, le rapport entre moulure et profil est entaché d'une ambiguïté qui nous posait problème : à une moulure correspond un profil élémentaire, d'un profil on peut exprimer un corps de moulures (c'est à dire un ensemble de moulure, ou *composition de profils élémentaires ou de moulures* [Pérouse De Montclos, 1988, p179] ).

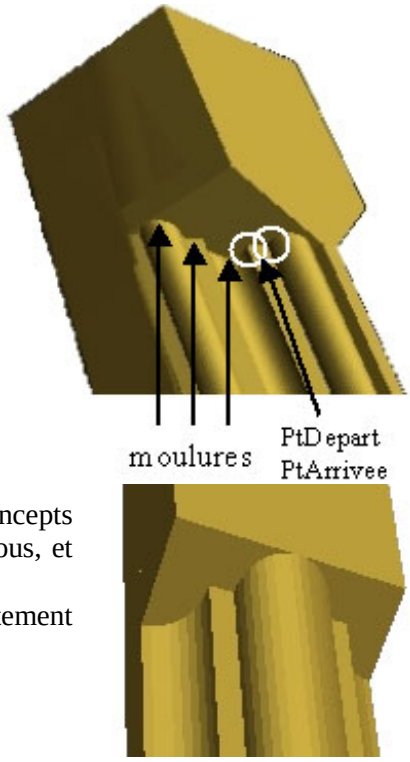



*Figure 125 : Principe de définition des moulures illustré sur le cas de profils observés sur le corpus des plafonds en bois (VRML, im auteur)*

Nous avons souhaité lever cette ambiguité en distinguant deux concepts complémentaires : celui de moulure, dont je donne le contenu ci-dessous, et celui de profil, plus générique, détaillé par la suite.

La moulure est dans la définition que nous lui donnons complètement déterminée par:

– Un objet Point nomme ptDepart (contraint dans le plan xy).
– Un objet Point nomme ptArrivee (contraint dans le plan xy).
– Un type fixant la concavité ou la convexité de la moulure et son tracé (plat, demi-rond, quart de rond, etc..).
– Un objet LigneBrisee contenant la liste des points intermédiaires servant au dessin de la moulure, calculés en fonction de son type (points dont le nombre est un paramètre du modèle).

Une méthode de la classe Moulure nommée renseigneLigne effectue les opérations nécessaires à l'affectation de la propriété LigneBrisee à partir d'une part des points extrémités de la moulure et d'autre part de son type. Une propriété générique (héritée de la classe Entité) détient l'information relative au niveau de détail souhaité, c'est à dire au nombre de segments à afficher entre les points extrémités de la moulure. Il y a donc distinction entre l'information relative à la définition théorique de la moulure (un quart de rond par exemple) et l'information relative à sa visualisation.

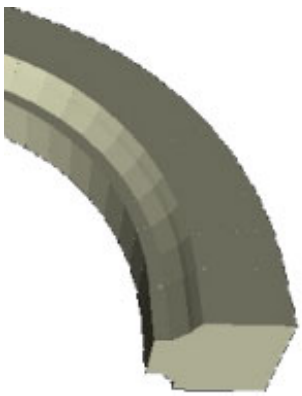


*Figure 126 : Profil mouluré d'un arc Tiers Point (VRML, im auteur)*

*LES PROFILS*

En nous appuyant sur le travail de J.Tachman et M.Łukacz, nous avons choisi de distinguer le concept de profil plats et un concept en dérivant, le profil mouluré. La classe ProfilMouluré correspond à des objets dont les points de contrôle des moulures suffisent à déterminer complètement la morphologie. La classe Profil correspond à des objets dont la morphologie

n'est connue qu'au travers des types de chacune de ses moulures. Cette distinction, outre qu'elle reprend un fait établi par nos sources bibliographiques, se justifie également du point de vue du formalisme de représentation des connaissances choisi. En effet, le profil Mouluré doit disposer de méthodes spécifiques permettant d'évaluer chacune de ses moulures. Autrement dit, le profil mouluré est une spécialisation du profil puisque les propriétés nécessaires et suffisantes à la détermination du second ne détermine pas complètement le premier.

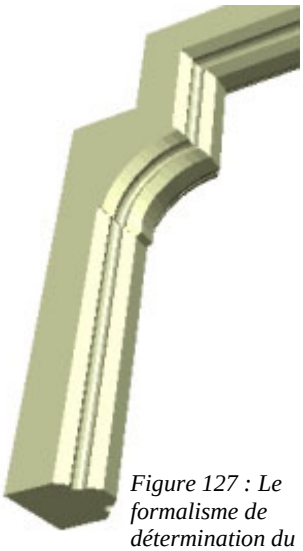


*Figure 127 : Le formalisme de détermination du Profil appliqué au cas d'une baie (VRML, im auteur)*

Dans le profil plat, ou profil à fasces, les points de contrôle des moulures correspondent aux sommets nécessaires et suffisants dans la détermination de la morphologie de l'objet. La propriété trace (instance de la classe ligneBrisee) de l'objet Profil est une collection de points directement affectée par la boîte englobante de l'objet et par les points de contrôle des moulures. L'objet Profil est doté de méthodes d'affichage qui génèrent, à partir de la propriété trace, des formes tridimensionnelles pertinentes au regard du contexte d'utilisation (solives, claveaux d'arcs, piédroits, etc…).

Un même formalisme de détermination de l'attribut Profil est donc mis en œuvre dans diverses entités architecturales, conformément à la définition que nous avons donnée au concept d'attribut. La détermination d'une méthode de description générique des profils nous permet également de résoudre des problèmes de représentation non liés à la modénature d'un objet, comme par exemple les ébrasements de baie.

Cette approche revient à confier à une hiérarchie de concepts architecturaux essentiellement morphologiques, les attributs, le soin de gérer la représentation tridimensionnelle de concepts architecturaux essentiellement sémantiques, les entités architecturales.

Le Profil Mouluré possède par héritage la propriété trace, ligne brisée contenant tous les points utilisés dans la représentation, et donc fonction de la propriété niveau de détail de l'entité appelante.

La propriété trace est, je l'ai indiqué, caractéristique de la seule représentation de l'objet: si dans le cas des profils plats elle contient l'ensemble des points de contrôle, caractéristique de l'objet, elle contient ici également l'ensemble des sommets des segments représentant chaque moulure, sommets dont le nombre varie en fonction d'un choix de représentation.

*Figure 128 : Profil plat: les points de contrôle suffisent à visualiser l'objet (VRML, im auteur)*

La propriété trace ici n'est renseignée ici qu'après lecture de l'ensemble des propriétés ligne de moulures contenues dans le profil. Pour qualifier le profil mouluré on dispose d'une liste de moulures (nommée moulures, voir section précédente) , d'une liste de points de contrôle (listePts, arêtes du profil ou points de jonction entre deux moulures) et de types correspondants (convexe demi-rond, concave quart de rond, etc..). La méthode renseigneTrace de l'objet profilMoulure lit les tableaux listePts et types et instancie pour chaque binôme (2 points, 1 type) une moulure affectée à moulures.

La méthode renseigneLigne de chaque moulure calcule les points nécessaires à la représentation de cette moulure et les affecte à trace. Ainsi, la définition d'une liste de points de contrôle et de types suffit à renseigner la morphologie d'entités difficiles à relever ou à représenter, en particulier les solives dont nous avons largement traité précédemment. Comme pour les profils simples, un même formalisme de détermination de l'attribut Profil est donc mis en œuvre dans diverses entités architecturales. Les méthodes d'affichage de l'objet Profil mouluré se chargent de produire une représentation tridimensionnelle propre à l'entité appelante.

*Figure 129 : Points de contrôle d'un profil mouluré*

*Figure 130 : Trois échelles du modèle, trois types de rendu*

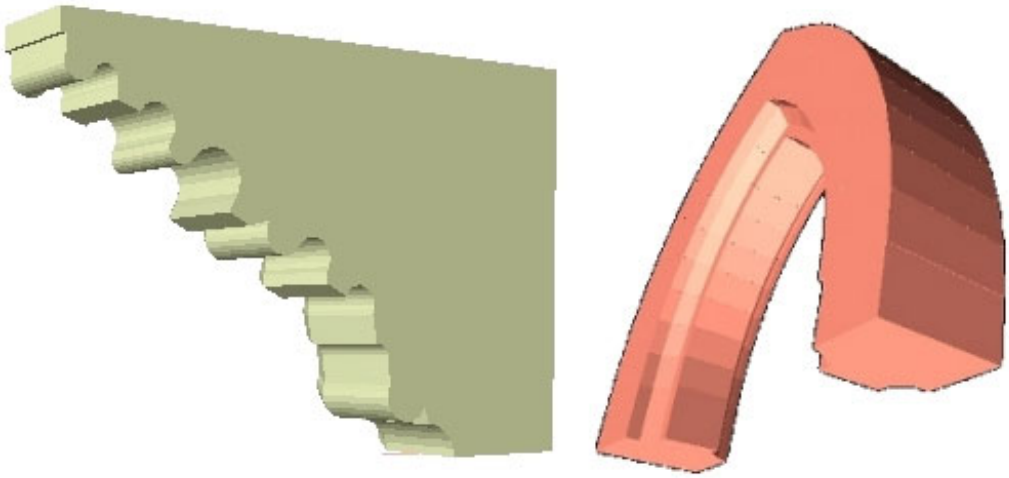
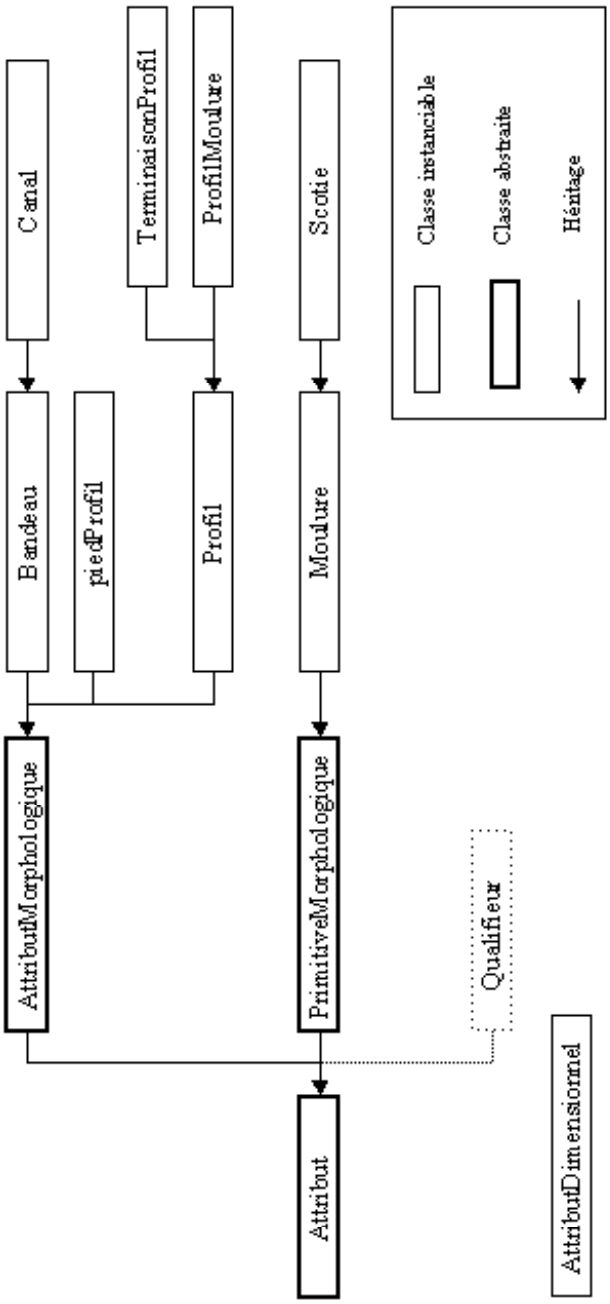


*Figure 131 : Représentations tridimensionnelles de l'attribut Profil des entités Console et ArcTiersPoint (VRML, im auteur)*

## *ANNEXE 37 : TÊTE DE LA HIÉRARCHIE D'ATTRIBUTS*



## ANNEXE 38 : ILLUSTRATION DE LA NOTION DE PROPORTION

### UNE ARCHITECTURE MODULAIRE : L'EDIFICE ANTIQUE

Le processus Paros repose sur l'hypothèse de l'existence d'un modèle théorique de l'architecture construite. Nous avons choisi l'architecture antique comme terrain d'expérimentation du procédé. Ce type d'architecture se prête bien à une modélisation : le rôle central des règles de proportions modulaires nous permet une description simple de la morphologie des entités. L'hypothèse de base du projet nous

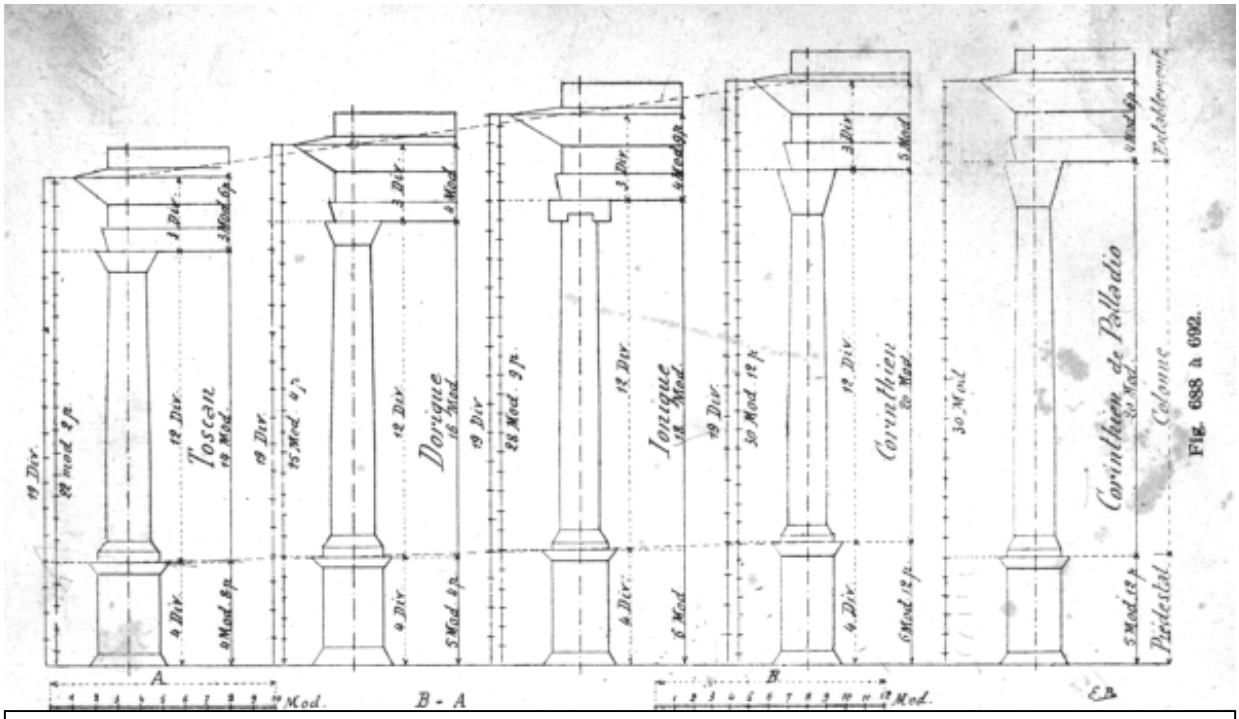



*Figure 132 : Le module et les cinq ordres d'architecture selon [Barberot, 1922, p549]*

c modèle théorique repose sur l'existence de régularités proportionnelles dans le système constructif et dans la modénature de l'architecture antique. La notion de norme, pour définir cette approche constructive, est tentante. Un édifice antique sacré était construit autour de cette notion de module, unité de mesure qui structure sa composition. L'ensemble des dimensions inter et intra entités sont réductibles à un rapport simple au module. Par ailleurs, la détermination d'un modèle de composition, modèle connu sous le nom d'ordres, et repris largement à la renaissance, peut laisser penser qu'il est possible à partir d'une dimension et d'une affiliation à un ordre de déduire l'essentiel des caractères de l'édifice. La plupart des traités d'architecture écrits à partir de la Renaissance contiennent ainsi une définition canonique des ordres d'architecture, et ce jusqu'aux ouvrages pratiques du début de ce siècle. Cependant l'existence d'une norme architecturale dans l'antiquité pose quelques problèmes, rappelés dans [Drap, 1997] :

- Stabilité de la norme dans le temps : la notion d'Antiquité couvre près de dix siècles comme l'illustrent deux de nos expérimentations : le temple Jupiter Capitolin à Rome au VIe siècle avant J.-C. et le portique nord du forum de la ville d'Arles en Gaule au IVe siècle après J.-C. De plus, les architectes de la Renaissance s'appuient sur cette norme en utilisant à la fois une définition théorique, celle que livrent les traductions de Vitruve, et des définitions par l'exemple, celles que livre l'observation des monuments antiques encore en élévation. On pourra se reporter par exemple à [Palladio, 1738/1965] ou [Fichet, 1979].
- Stabilité de la norme dans l'espace : l'architecture antique s'est développée dans tout le bassin méditerranéen, d'abord sous l'influence hellénistique puis sous celle des conquêtes romaines. L'étendue du domaine en regard des moyens de communication de l'époque rend peu probable une norme unique d'Arles à Palmyre.

Choisy, après bien d'autres, constate cette variabilité du modèle lorsqu'il figure l'évolution de l'entablement Ionique (voir schéma ci-dessous). Après avoir identifié deux éléments qu'il qualifie de

facultatifs, la frise et les denticules, A. Choisy commente l'évolution de chaque entité en jeu (architrave, frise et corniche) en distinguant :

- Des observations (passage d'une architrave à deux bandes à une architrave à trois bandes au cinquième siècle).
- Des hypothèses techniques (corniche avancée pour garantir l'édifice contre les eaux pluviales).

Il apparaît donc difficile de considérer la définition des ordres et les proportions modulaires comme une

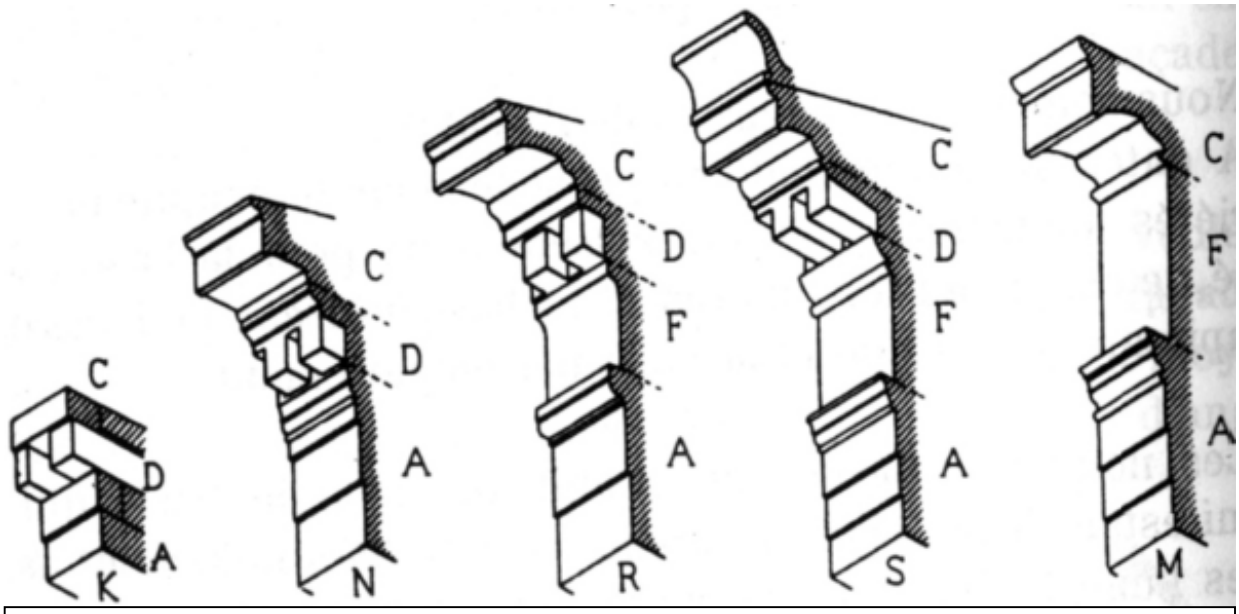



*Figure 133 : Evolution de l'entablement Ionique selon [Choisy, 1899/1991, p362]*

Par ailleurs, outre le problème de la stabilité de la norme il reste celui de sa transmission jusqu'à nous. Elle s'est effectuée selon deux voies : les édifices eux-mêmes, qui sont souvent ruinés voire arasés, dont les vestiges sont à déchiffrer, et la connaissance livresque. Il ne nous reste qu'un seul traité d'architecture décrivant les édifices grecs et romains de la période antique ; c'est le De architectura de Vitruve ([Vitruve, 1684/1988]). Un seul livre, un seul auteur pour couvrir dix siècles d'architecture du bassin méditerranéen. Le De architectura est écrit à Rome, à la fin de la République. L'auteur, pour les deux raisons évoquées plus haut, ne pouvait avoir eu connaissance de l'ensemble de la production architecturale des cinq derniers siècles dans le bassin méditerranéen, même s'il avait voyagé et possédait une connaissance livresque de l'architecture hellénistique. Par ailleurs, il ne possédait sûrement pas l'autorité pour imposer après lui une norme architecturale dans tout l'empire. Vitruve ne pouvait être que le témoin d'un état de la production architecturale dans l'antiquité en un temps et un lieu donnés.

L'étude de l'édifice doit donc prendre en compte l'utilisation possible d'éléments de réemploi (issus d'autres constructions) et doit, par la mesure, déterminer les proportions modulaires de la modénature des entités architecturales. Il faut citer à ce propos, l'étude du temple Portunus par Jean-Pierre Adam [Adam, 1994] chapitre V, où l'analyse de la campagne de mesures faite sur le temple permet de retrouver les tracés régulateurs et les proportions modulaires. L'analyse doit prendre en compte :

- les hypothèses archéologiques sur les rapports modulaires,
- l'incertitude de mesure,
- l'érosion et la dégradation de la pierre,
- les restaurations passées du bâtiment au cours desquelles des éléments ont été remplacés,
- les proportions modulaires sont alors confirmées et réévaluées à l'aide la mesure.

Dans les expèriences d'Arles [Drap et al, 1995] et de Rome [Florenzano et al, 1997b] nous avons utilisé des tables de références permettant d'établir des grandeurs non mesurées, c'est à dire de renseigner des propriétés des entités ou de leurs attributs morphologiques. Ces tables établissent une série de rapports entre la grandeur à évaluer et le module. La valeur de celui-ci est obtenue soit en le mesurant (le module correspond au rayon du pied du fût, quel que soit l'ordre) soit en le déduisant de l'observation de

propriétés morphologiques des entités ou des attributs morphologiques (le module vaut par exemple deux fois la hauteur de la plinthe de la base dans l'ordre toscan).

Je renvoie aux publications concernées pour un compte-rendu plus détaillé de ces expèriences. Le module nous a donc permis de tirer parti d'une notion de proportion propre à l'architecture antique. Peu d'exemple ultèrieurs permettent d'établir de façon aussi rudimentaire (table de rapports au module) les possibles dimensions relatives des éléments de corpus constituant l'édifice. Toutefois on notera que la plupart des architectures raisonnées font peu ou prou appel à une logique de dimensionnement, depuis l'incontournable modulor de Le Corbusier jusqu'aux trames économiques des grands ensembles de logements de la deuxième moitié du vingtième siècle. Notre débat n'est pas là, nous allons maintenant tenter d'évaluer le rôle de l'idée de norme dans une architecture non modulaire, celle de l'édifice moyennageux.

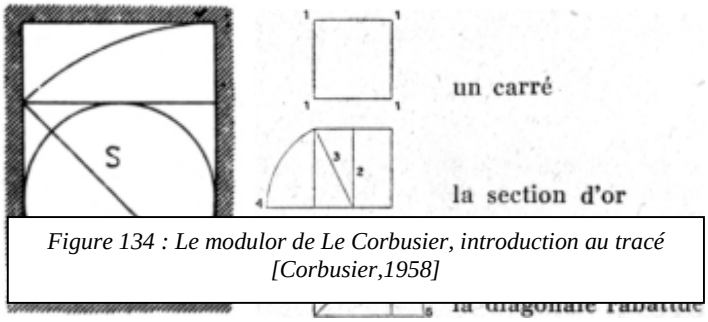



*Figure 134 : Le modulor de Le Corbusier, introduction au tracé [Corbusier,1958]*

*L'EDIFICE MOYENAGEUX: CONVENANCES CONSTRUCTIVES ET TRACES REGULATEURS*

Selon les termes même d'A.Choisy [Choisy, 1899/1991, p403], "*il est difficile de déterminer d'après les monuments du moyen-âge la loi qui présidait à leur mise en proportion*". Il propose une explication sous la forme suivante: "*l'exécution souvent grossière, les sujétions de l'appareil, les épaisseurs inégales des lits de mortier compliquent et voilent la loi des tracés*" [Choisy, 1899/1991, p403]. On peut ajouter que les nombreuses transformations subies par la plupart des édifices de cette époque ne favorisent pas leur lecture. Il est par conséquent assez naturel que les propositions de tracés que l'auteur présente soient centrées sur de grands édifices religieux comme la cathédrale d'Amiens ou Notre Dame de Paris.

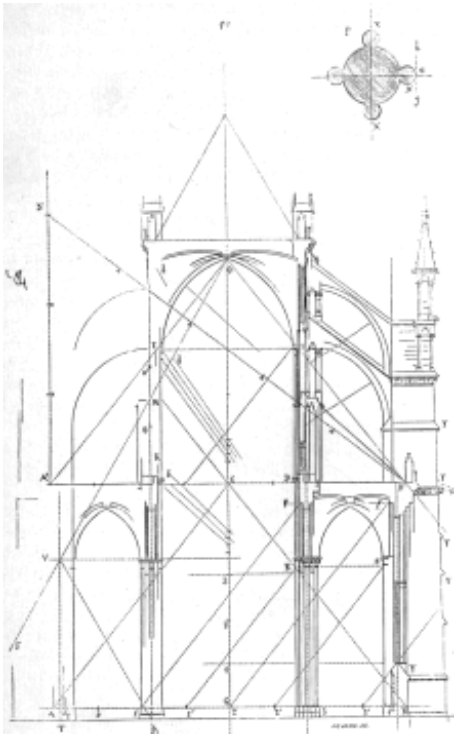


*Figure 135 : Etude du tracé en coupe de la cathédrale d'Amiens [Viollet Le Duc, 1863/1977, p406]*

Pourtant, lui comme Eugène Viollet Le Duc se sont efforcés dans leurs ouvrages d'apporter de nombreux indices sur ce que l'on doit comprendre par norme au moyen âge. S'il n'y a pas ici de système de grandeurs relatives comparable au module de l'antiquité, Eugène Viollet Le Duc voit "*dans leur système de proportions appliquées à l'architecture des lois dérivées évidemment de certains principes antiques, bien que ces artistes ne songeassent guère à imiter la forme de l'architecture antique et que le point de départ de leur système de construction fût, …, absolument étranger au système admis soit par les grecs, soit par les romains*" [Viollet Le Duc, 1863/1977, p405]. Dans son neuvième entretien, intitulé "sur des principes et des connaissances nécessaires aux architectes" [Viollet Le Duc,1977, pp385-448], l'auteur dresse une liste de recommandations sur la proportion à partir d'exemples pris à diverses époques, en bâtissant des reconstructions interprétatives pour retrouver les tracés régulateurs d'édifices conservés ou non. Je renvoie à cet ouvrage pour plus de détails, et me bornerai ici à citer quelques règles identifiées par A. Choisy [Choisy, 1899/1991, pp404-406]: "*il est pourtant des règles qui se dégagent, entre autres celle des cotes entières et celle des rapports simples :*

- *Les cotes entières - en ce qui concerne l'adoption de cotes entières, les vérifications peuvent se multiplier à l'infini. Parmi les cotes dont la détermination est la plus sûre, celles des épaisseurs de piles, de contreforts ou de colonnes s'expriment toujours en pieds et pouces par des chiffres d'une simplicité remarquable; il est peu d'édifices où l'on ne rencontre par exemple des colonnes dont le diamètre ne mesure exactement un pied.*

– *Les rapports simples - quand aux rapports simples, dans bien des cas ils se manifestent à première vue, l'intention se lit :*

*A. A la nef d'Amiens, la pensée de l'architecte fut d'établir dans la composition deux grandes divisions égales, séparées par un bandeau R qui se développe à mi-hauteur,*

*B. A Notre Dame, l'intention évidente a été de donner au corps de la façade un contour en forme de carré, et aux tours une hauteur égale à la moitié du coté du carré de base,*

*… Non seulement les architectes du moyen-âge s'astreignent, comme ceux de l'antiquité, à l'observation des rapports simples: les rapports qu'ils préfèrent sont précisément ceux que recherchait l'antiquité classique… on peut traduire en rapports simples les proportions données par certains triangles remarquables, le triangle équilatéral, le triangle égyptien et ses dérivés.*
*Ces triangles restèrent en faveur pendant tout le moyen-âge …."*

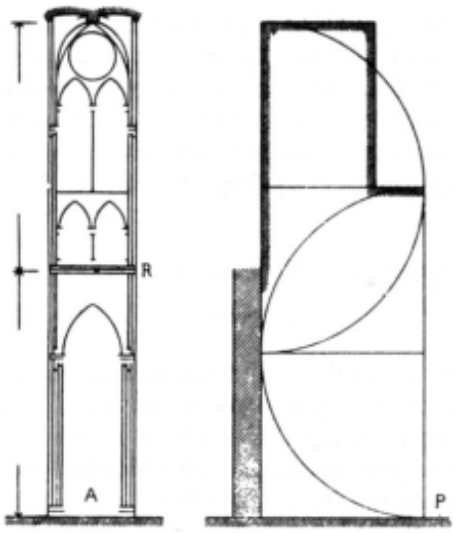



*Figure 136 : Analyse des tracés de la cathédrale d'Amiens (coupe) et de Notre Dame de Paris (façade)[Choisy, 1899/1991, p404]*

Notons enfin que le même auteur recense sous le titre "illusions optiques, corrections perspectives, dissymétries" un ensemble de dispositifs qui règlent la composition d'édifices dont le maître d'œuvre juge que tel ou tel aspect doit être rectifié.
Nous citons dans le schéma ci-dessous quelques-uns uns de ces procédés, accompagnés de procédés mis en œuvre dans le même esprit par les grecs. Je renvoie pour les premiers à [Choisy, 1899/1991, pp403-414] et pour les seconds à [Choisy, 1899/1991, pp384-409].

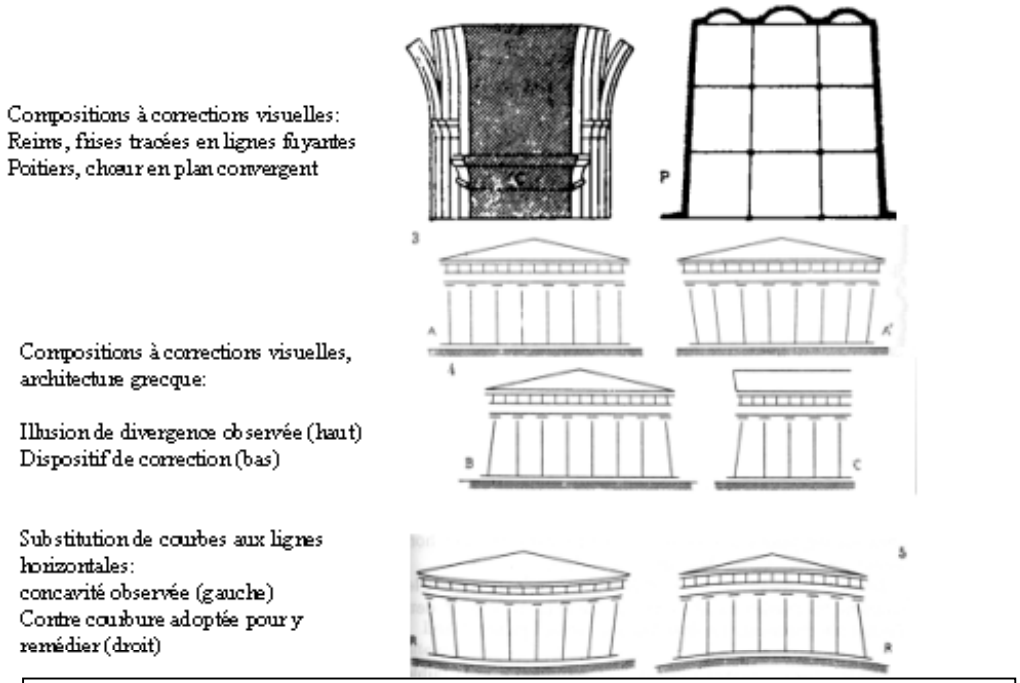



*Figure 137 : Dispositifs de rectification des compositions à différentes pèriodes [Choisy, 1899/1991]*

Je renvoie également aux sixième, septième et neuvième entretiens sur l'architecture de Viollet Le duc [Viollet Le Duc,1977] dans lesquels l'auteur propose son analyse des notions de style et de composition et discute de façon détaillée des moyens mis en œuvre dans l'architecture antique puis dans l'architecture du moyen âge pour mettre l'édifice en scène [Viollet Le Duc,1977, p254] .

*ANNEXE 39 : MÉTHODES DE REPRÉSENTATION : CHOIX TECHNIQUES ET OBJECTIFS*

**Imagerie de synthèse :** une description de la scène pour le logiciel de rendu Pov -Ray (Persistence Of Vision ray tracer) était générée en résultat au processus de mesurage. Cette description textuelle contenait pour chaque entité en jeu trois parties : une déclaration de variables (correspondantes aux valeurs des attributs morphologiques des entités), une description textuelle paramétrée des opérations à effectuer pour représenter l'objet en question (Opérations booléennes exclusivement) et enfin un appel explicite à l'objet.

*Figure 138 Rendu POV du chapiteau toscan selon Vitruve [Vitruve, 1988] (im auteur)*

L'entité disposait à la fois d'une méthode d'écriture de ses variables pour le script global à exécuter et d'une méthode d'écriture des opérations nécessaires à sa construction. Cette distinction, encore inutilisée à l'heure où nous changions l'architecture logicielle de ce projet, devait permettre d'extraire explicitement la partie descriptive des opérations nécessaires à la construction d'une entité de la définition de sa classe. De la même façon, nous devions paramétriser les sections textures définissant l'apparence des entités. Le script global, au format ASCII, était lancé en ligne de commande sous DOS. Nous n'avons pas à proprement parler abandonné POV-Ray mais l'avons mis en sommeil jusqu'à cette année.

Le principal inconvénient de cette solution était l'incapacité de la scène produite à s'interfacer avec des données non graphiques, objectif devenu alors essentiel. Pourtant, pour autant que Pov -Ray soit cantonné à son rôle de module de rendu, il peut s'avérer comme une alternative aux modules de rendu des grands éditeurs de logiciels de CAO/DAO. En effet, POV bénéficie d'avantages importants : simplicité et stabilité de la syntaxe, gratuité du moteur de rendu comme de l'interface d'édition de scènes, et bien sûr qualité de rendu.

De plus, POV autorise aujourd'hui la définition d'objets par une suite d'opérations booléennes mais aussi par facettes, permettant une traduction plus aisée de méthodes d'affichage existantes pour VRML. Le schéma ci-dessous récapitule les trois sections d'un script POV, déclaration des variables, définition générique d'une fonction assurant les opérations nécessaires à la construction de la maquette, appel de cette fonction.

```
CHAPITEAU  ////////////////////

#declare modim = 0.297396
#declare filetLz = 0.038
#declare rayonFilet = 0.365837
#declare echineLz = 0.0595
#declare gorgerinLz = 0.110001
#declare abaqueLx = 0.78069
#declare abaqueLy = 0.78069
#declare abaqueLz = 0.09
#declare Orix =  1.984107
#declare Oriy =  5.014945
#declare Oriz=  5.316096
```

```
#declare chatos = union {
box { <abaqueLx/2, abaqueLz, -
abaqueLy/2>, <-abaqueLx/2, 0,
abaqueLy/2>
translate <0, gorgerinLz+echineLz, 0>
  texture{}
}
torus {abaqueLx - echineLz, echineLz
 translate <0, gorgerinLz + echineLz, 0>
  texture{}
}
cylinder {<0,0,0>, <0, filetLz, 0>, rayonFilet
translate <0, gorgerinLz + echineLz,0>
  texture{}
}
etc….
}
```

```
object  {chatos translate
<0,1,0>}

object  {chatos translate <
Oriy, Oriz, -Orix>}
```

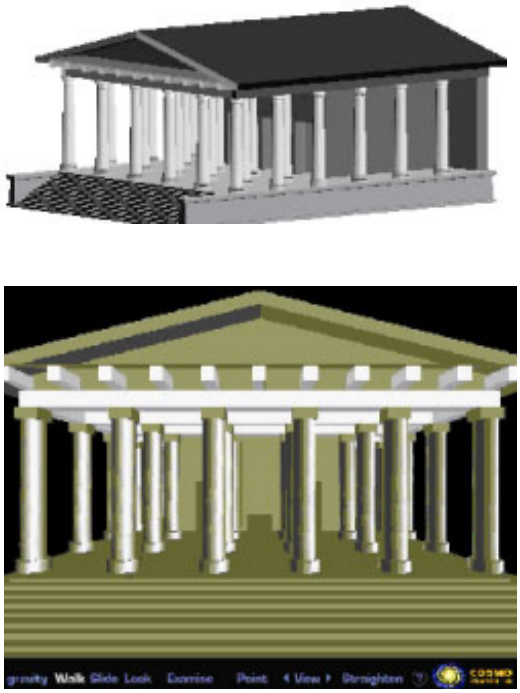


*Figure 139 : Mise en scène de l'hypothèse de Filippo Coarelli sur le temple Jupiter Capitolin; modeleur Autocad et VRML 1.0 (Expérience de Rome, voir [Florenzano et al, 1998]).*

**Visualisation dans un outil de CAO :** Un script pour le modeleur Autocad était également généré, autorisant la visualisation et l'évaluation de la scène à l'intérieur du modeleur. Un script se présentait sous la forme d'une suite de commandes Autocad prenant en paramètre telle ou telle information numérique des entités, script à exécuter en ligne de commande à l'intérieur du modeleur. Chaque entité disposait d'une méthode effectuant les tâches requises pour compléter le script de son réseau : renseignement des grandeurs et écriture sur le fichier script. Bien qu'autorisant effectivement la constitution d'une maquette numérique fidèle au modèle, cette procédure posait un certain nombre de problèmes qui nous ont poussé à l'abandonner (problème de maintenance en fonction des versionnements du modeleur, instabilité à l'éxécution des scripts). Par ailleurs, cette dépendance vis à vis de l'éditeur de logiciels (icic Autodesk) devenait génante dans l'élaboration du programme de coopération ARKIW.

**Indépendance de plateforme et réseau Internet :** La visualisation du résultat d'un processus de mesurage était également prévue par le biais de scènes VRML 1.0, sous toutefois une forme simplifiée. Si l'objectif ici était clairement la visualisation de la scène à l'intérieur d'un navigateur, le format VRML 1.0 était cependant d'une expressivité limitée, et nous n'avions alors pas pris position clairement en sa faveur, tablant encore sur la notion de format d'échanges dont nous avons déjà évoqué les contrecoups. Trois événements nous ont finalement conduit à reconsidérer notre position sur VRML et à, en définitive, en faire le format de visualisation privilégié du modèle : l'interfaçage naturel graphique / non graphique qu'autorise VRML était devenu un objectif majeur; la version 2.0 du langage apportait un nombre considérable d'avancées (liens avec JAVA, routage d'évènements et animations, etc..), enfin la gratuité et la disponibilité sur toutes les plate-formes des outils de visualisation VRML nous permettait de travailler en partenariat dans de meilleures conditions. Il sera donc dans la suite de cette section essentiellement question de la visualisation du modèle architectural en VRML.

# ANNEXE 40 : EXPERIMENTATIONS DES OUTILS DE VISUALISATION

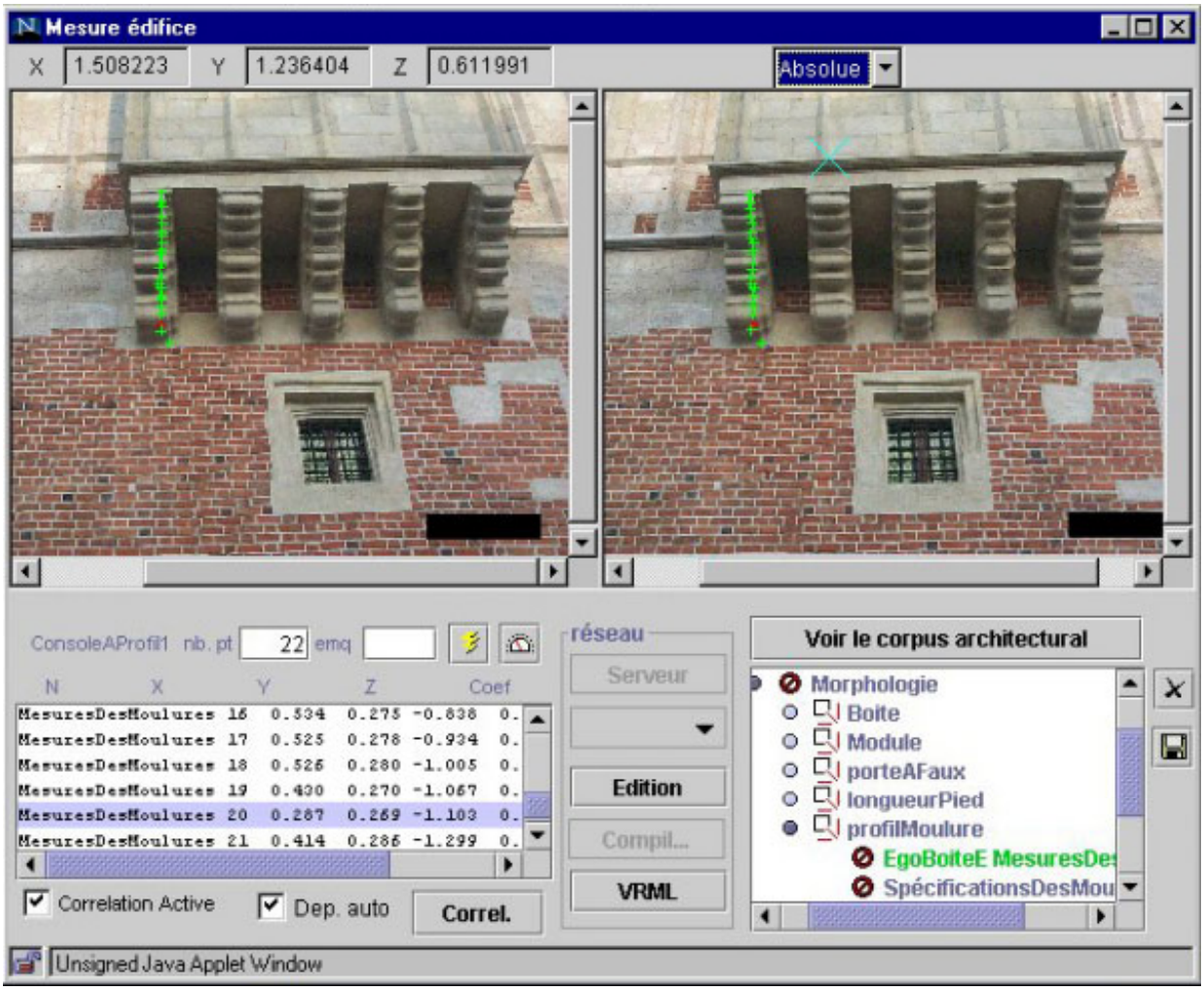



*Figure 140 : L'interface de saisie de l'ARPENTEUR; points mesurés à la surface des consoles moulurées de l'oriel Sud (Expèrience de l'ancien hôtel de ville de Cracovie, voir [Drap et al, 1999a] )*

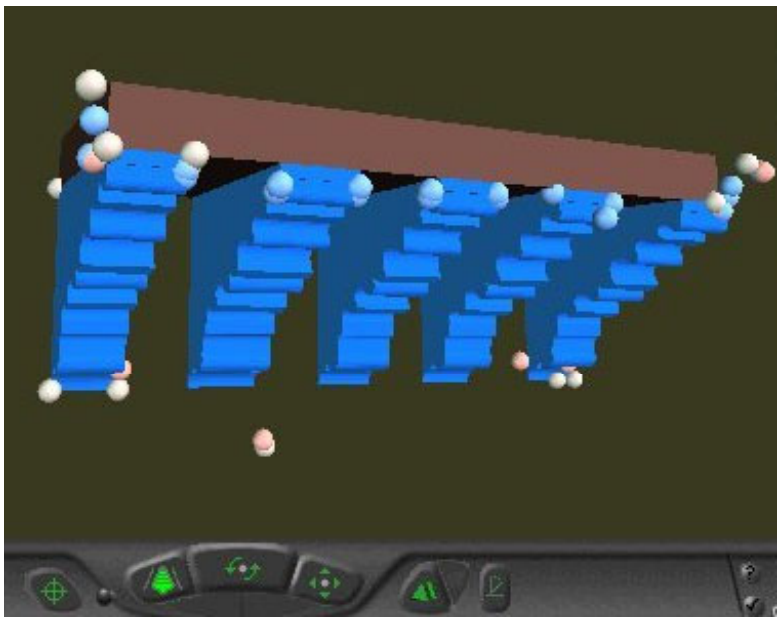


*Figure 141: sortie VRML en résultat au processus de mesurage (ARPENTEUR, Expèrience de l'ancien hôtel de ville de Cracovie, voir [Drap et al, 1999a] )*

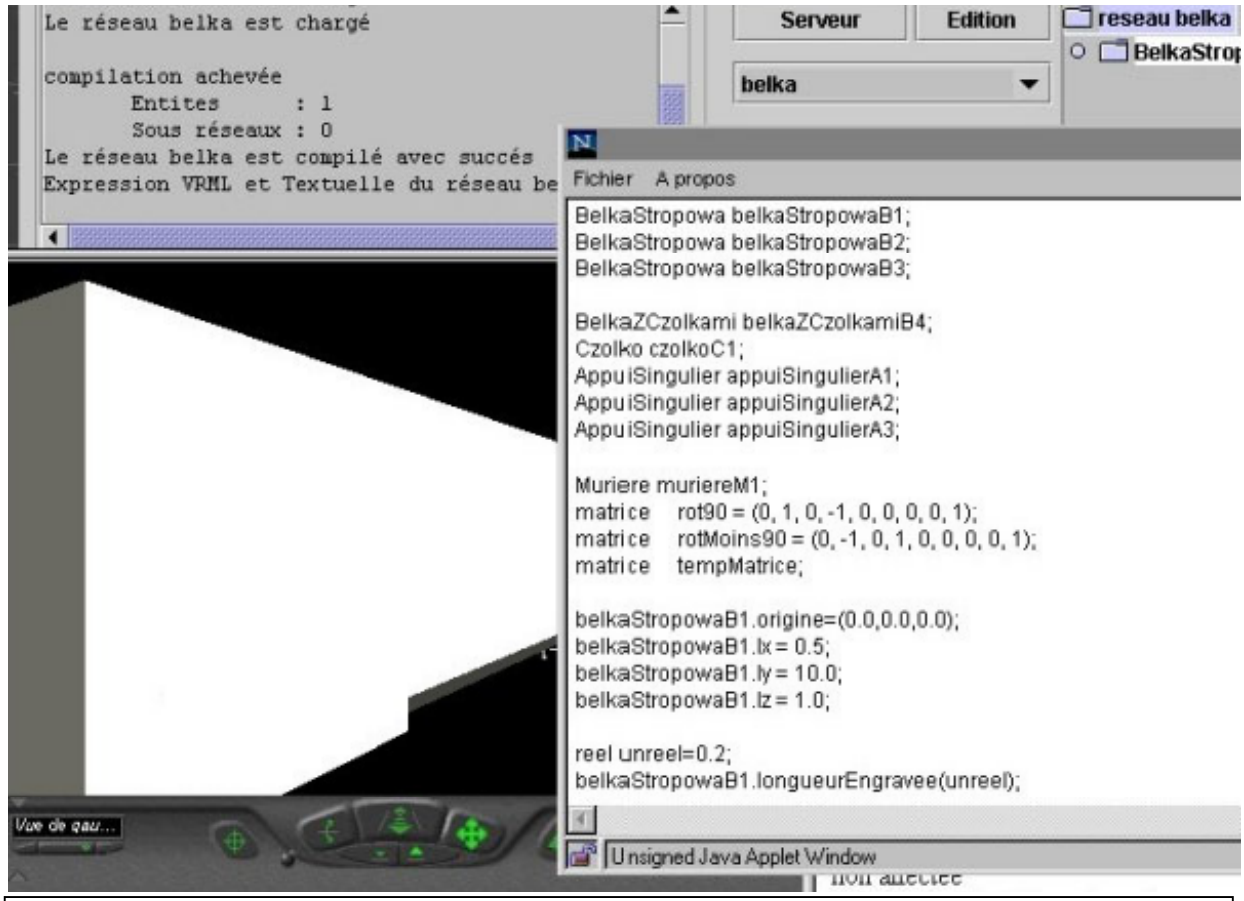



*Figure 142 : Interface de l'applet HUBLOT; affichage du contenu du script LaDemarche exprimant le jeu d'entités en cours sur le cas d'un plafond*

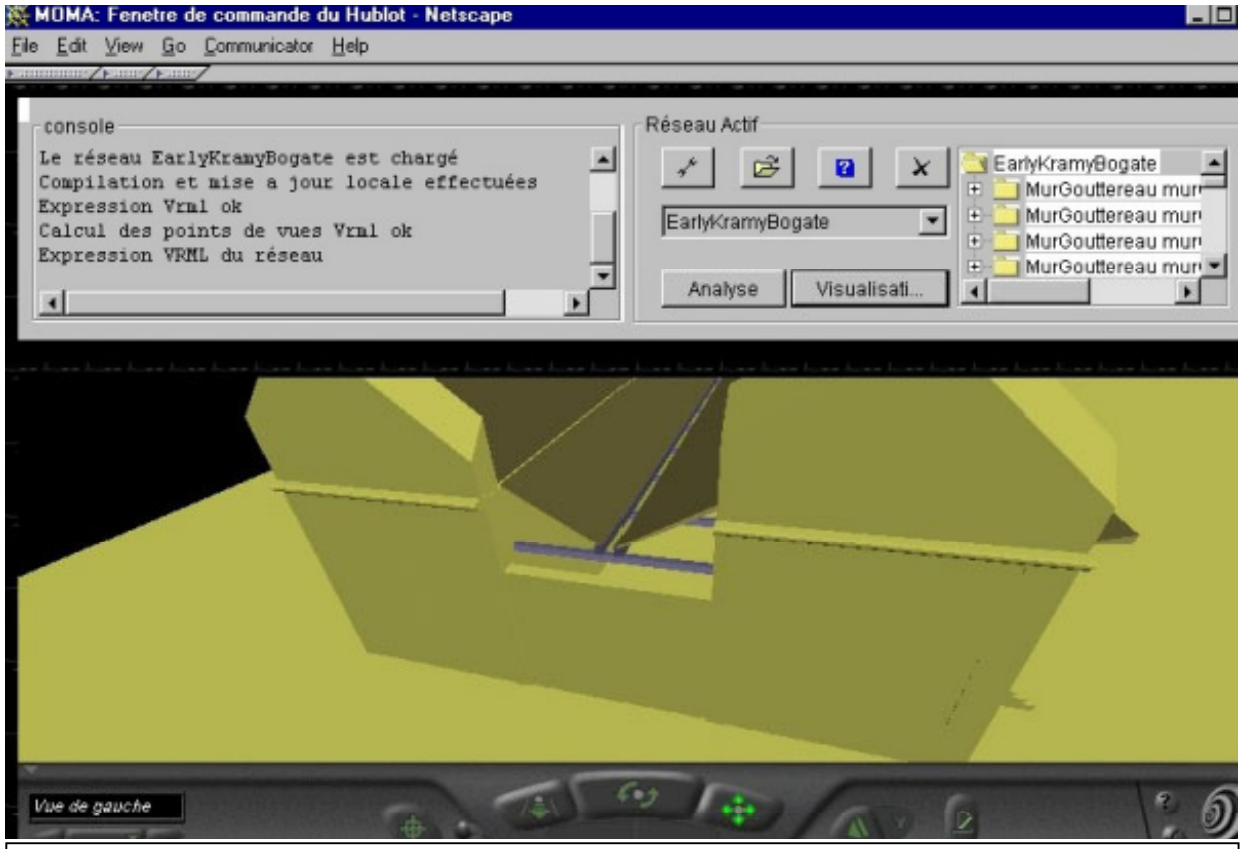
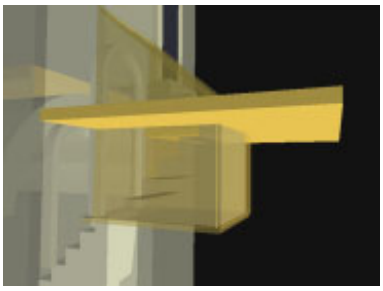
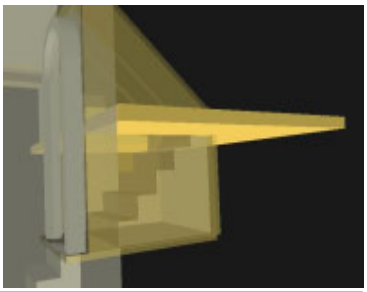



*Figure 143 : Interface de l'applet HUBLOT; statut du calcul (gauche), choix du réseau et actions (centre), choix des entités du réseau sélectionné (droite). Reconstitution de l'ancien marché au drap (Kramy Bogate) en cours d'élaboration.*

*Figure 144: Reconstitution de l'ancien marché au drap (Kramy Bogate) en VRML à partir du jeu d'entités architecturales disponibles dans l'interface de l'applet HUBLOT. Détail du système d'évacuation des eaux pluviales et de couverture de la galerie centrale, hypothèse figurant l'édifice dans sa dernière pèriode de présence sur la place (XIXème siècle)*
*(voir [Drap et al, 1999b]]*

*Figure 146: Reconstitution de l'ancien marché au drap (Kramy Bogate) en VRML à partir du jeu d'entités architecturales disponibles dans l'interface de l'application VALIDEUR. Détail de l'entrée latérale, hypothèse figurant l'édifice dans sa pèriode intermédiaire de présence sur la place (XVI-XVIIème siècle)*
*(voir [Dudek et al, 1999b])*





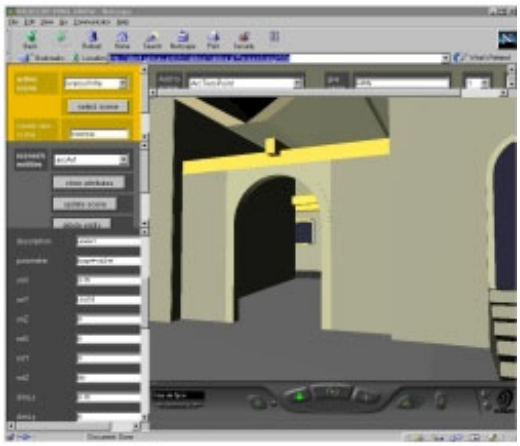


*Figure 145 : Reconstitution de l'ancien marché au drap (Kramy Bogate) en VRML à partir du jeu d'entités architecturales disponibles dans l'interface de l'application VALIDEUR.. Schémas de circulation et d'accès aux réserves en sous-sol, hypothèse figurant l'édifice dans sa pèriode intermédiaire de présence sur la place (XVI-XVIIème siècle)*
*(voir [Dudek et al, 1999b])*

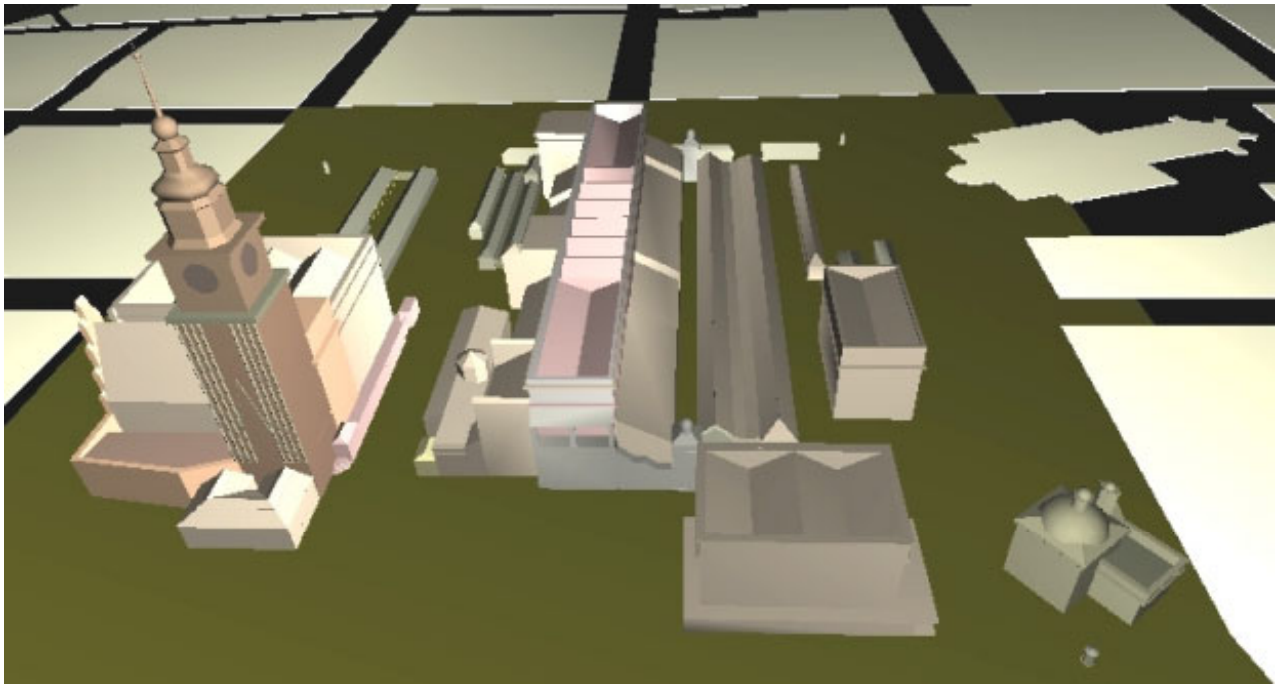


*Figure 147 : Interface VRML de l'outil de consultation et de mise à jour de la base documentaire SOL. Bâtiments représentés sommairement, hypothèse de restitution de la place centrale (Rynek Główny) à la fin du XVIIème siècle.*
*(voir [Dudek et al, 1999a])*

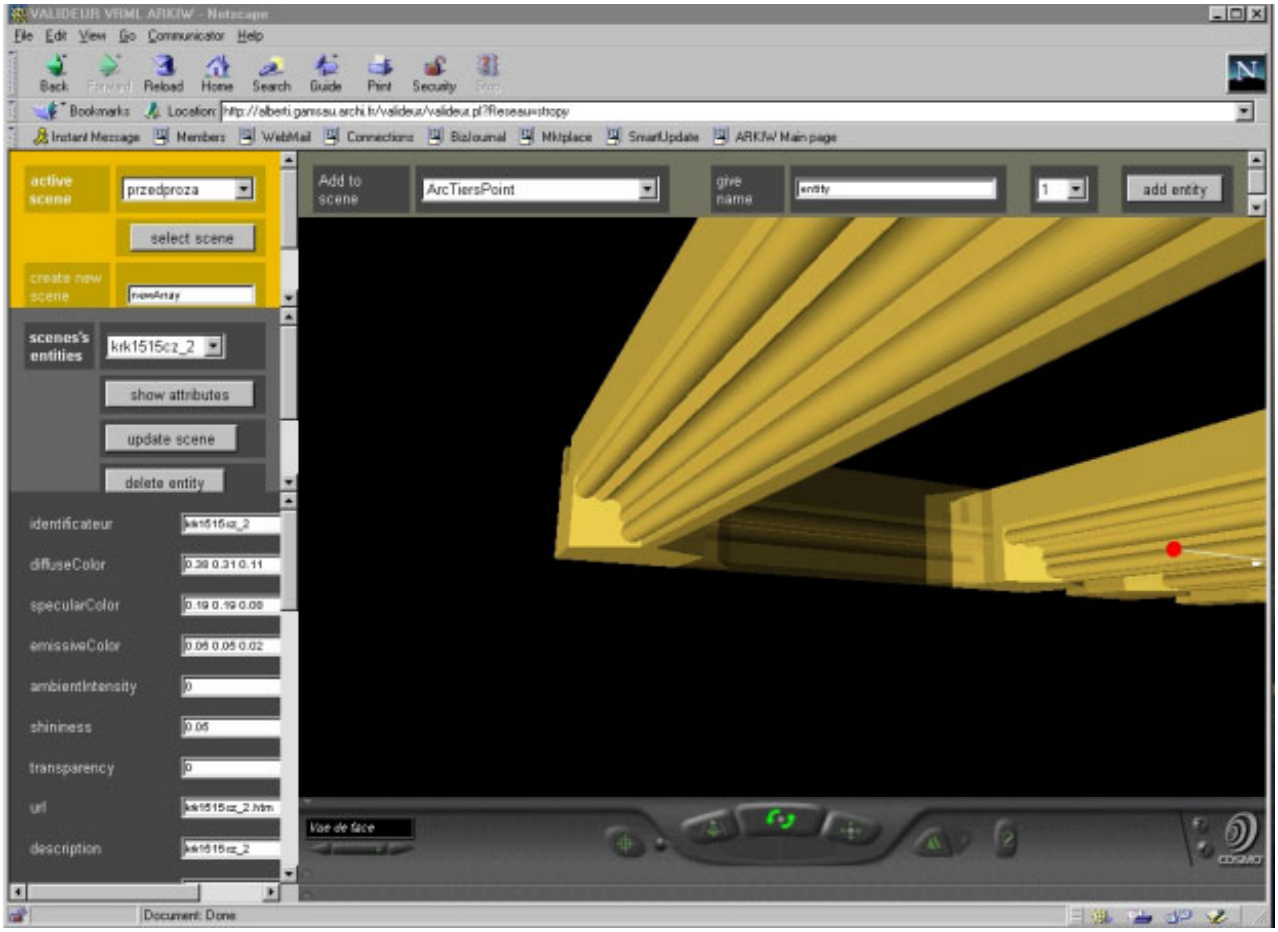



*Figure 148: Interface de l'application VALIDEUR, cas d'un plafond en bois.*

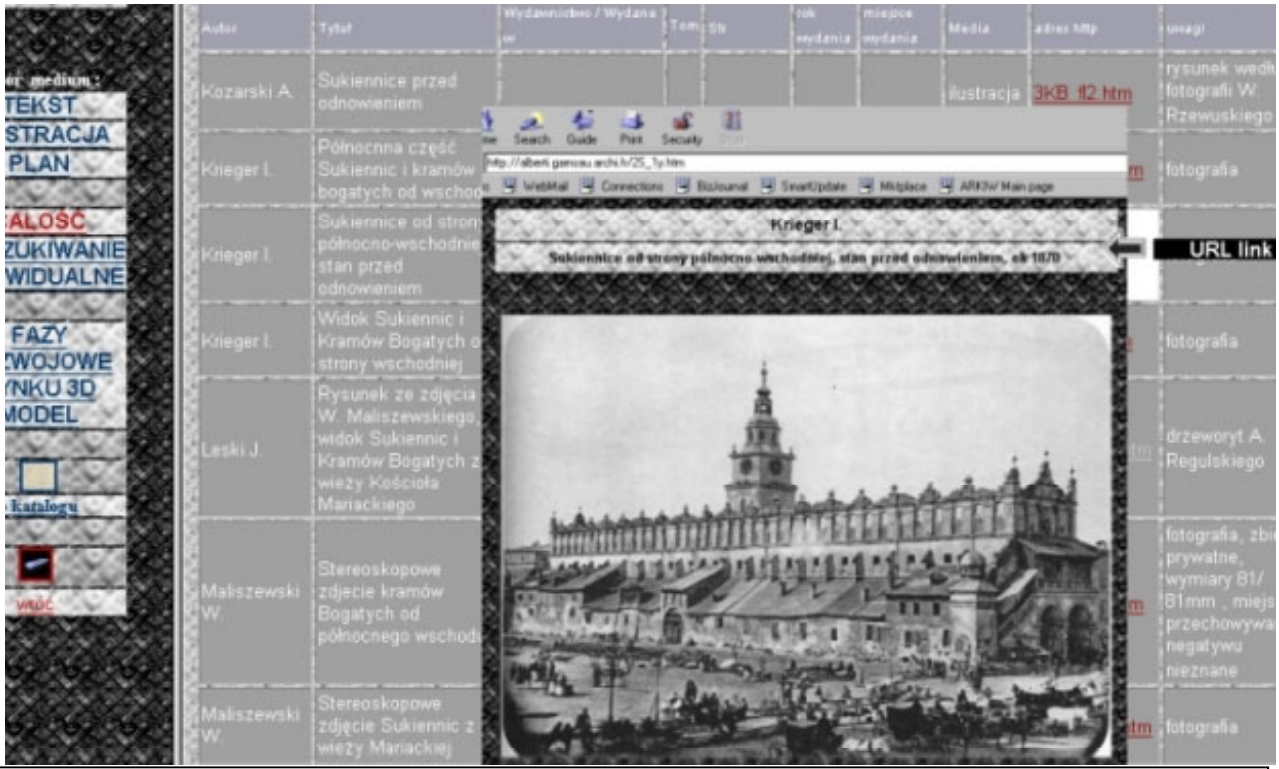



*Figure 149: Interface textuelle de l'outil de consultation et de mise à jour de la base documentaire SOL. Format d'affichage des réponses aux requêtes adressées au SGBD et page référencée par le lien URL.*
*(voir [Dudek et al, 1999a])*

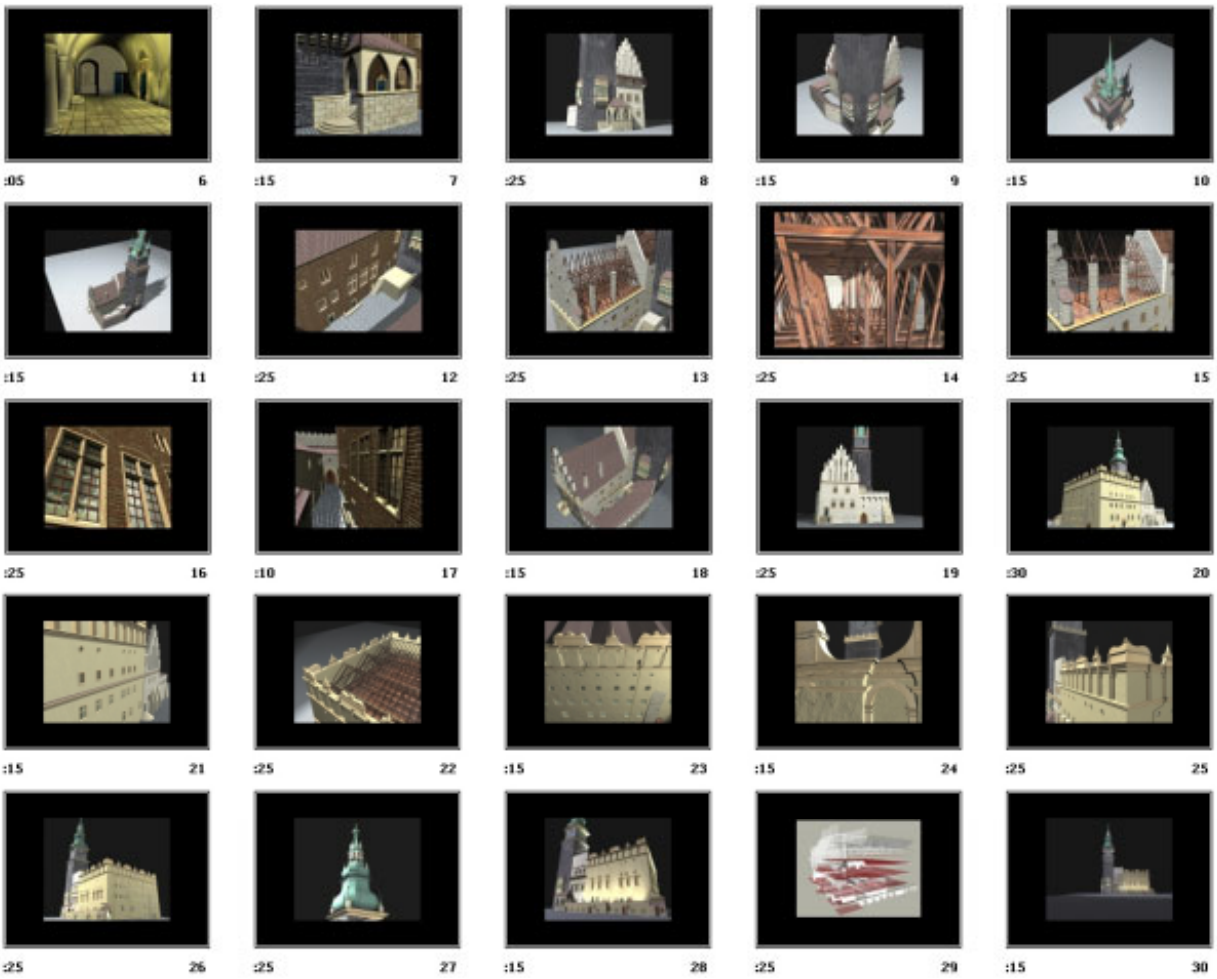



*Figure 150 : Restitutions de l'ancien hôtel de ville de Cracovie (Maya, im C.Radi)*

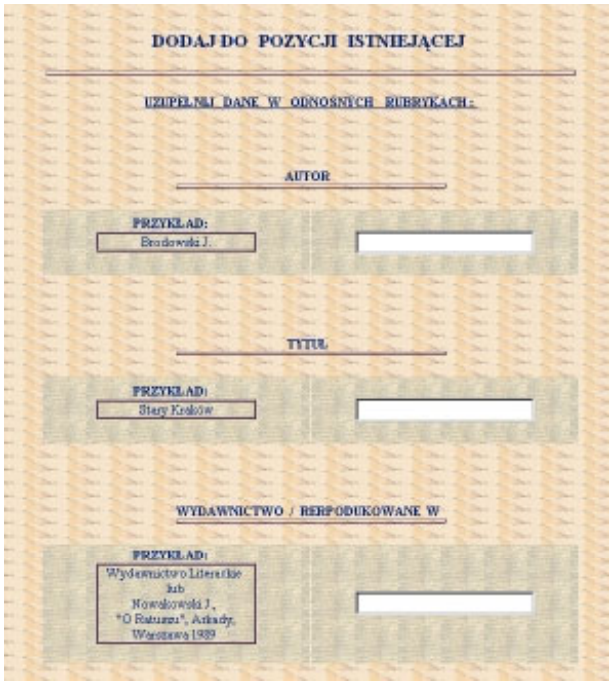



*Figure 151: : Interface textuelle de l'outil de consultation et de mise à jour de la base documentaire SOL. Formulaire de mise à jour. (voir [Dudek et al, 1999a])*

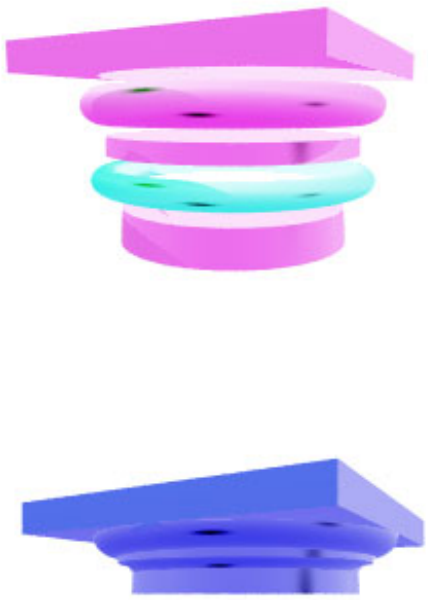


*Figure 152: Représentation d'un chapiteau toscan en rendu POV-Ray. Décomposition des primitives utilisées dans la méthode paramétrique de la classe ChapiteauToscan (haut) et rendu effectif (bas).*

ANNEXE 41 : DOCUMENTATION DES CONCEPTS: SCHÉMA ET TABLES DE RAPPORTS MODULAIRES

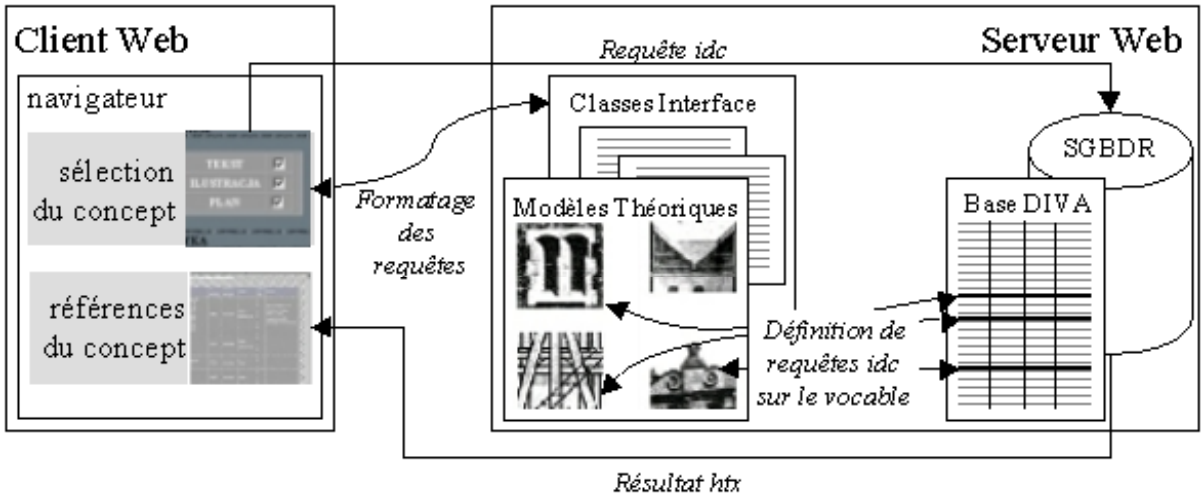



*Figure 153 : Architecture Client-Serveur de l'application DIVA*

Dans le cas des tables de rapports modulaires fixant les proportions relatives des attributs morphologiques d'entités architecturales du corpus, il s'agit de décrire, au niveau de la classe, des valeurs par défaut permettant d'établir à partir du module les valeurs canoniques (théoriques) des attributs morphologiques de chaque entité. Ce type d'approche permet par exemple de renseigner les dimensions d'entités non mesurées sur un édifice, comme on peut le voir dans les expériences d'Arles ou de Rome (voir [Florenzano et al, 1996a], [Florenzano et al, 1997a]). La solution adoptée pour attacher ces connaissances au modèle dans ces deux expériences était rustique : un fichier en ASCII avec séparateurs jouait le rôle de table de valeurs que la méthode d'évaluation par défaut des entités architecturales lisait afin de renseigner les valeurs de ses attributs morphologiques. Les rapports étaient précisés sous la forme de fractions, en référence à la bibliographie relative à cette question. L'adresse du fichier ASCII en question était propriété de la classe de base Entité. Pour rustique qu'elle fût, cette solution permettait néanmoins de sauvegarder l'essentiel : séparer données et traitements. Pourtant, la lecture effective du fichier (et donc le décodage de ses séparateurs) se faisait dans une méthode de l'entité, rendant le mécanisme d'auto-évaluation des entités dépendant du format du fichier ASCII. Cette solution a été remise en cause, la lecture de données externes étant confiée aujourd'hui à des objets dédiés à cette tâche. Le court extrait des tables de rapports modulaires pour les bases Attiques cité ci-dessous permettra de se faire une idée du type de donnée concerné.

| **Base Attique (proportions selon Vitruve) [Vitruve, 1684/1988]** | | **Base Attique (proportions selon Vignole) [Gromort, 1960]** | |
|---|---|---|---|
| Nom de l'attribut | *Rapport modulaire et fraction correspondante* | Nom de l'attribut | *Rapport modulaire et fraction correspondante* |
| baseLz | 1 module (1/1) | baseLz | 1 module (1/1) |
| plintheLx | 3 modules (3/1) | plintheLx | 2 modules 10/12 (34/12) |
| plintheLy | 3 modules (3/1) | plintheLy | 2 modules 10/12 (34/12) |
| plintheLz | 1/3 module (1/3) | plintheLz | 1/3 module (1/3) |
| toreInfLz | 1/4 module (1/4) | toreInfLz | 1/4 module (1/4) |
| rayonToreInf | 1 module 1/2 (3/2) | rayonToreInf | 1 module 5/12 (17/12) |
| scotieLz | 1/4 module (1/4) | scotieLz | 1/9 module (1/9) |
| toreSupLz | 1/6 module (1/6) | toreSupLz | 7/36 module (7/36) |
| RayonToreSup | 1 module 1/6 (7/6) | rayonToreSup | 1 module2/9 (11/9) |

*ANNEXE 42 : IMPLÉMENTATION JAVA*

Héritage multiple et polymorphisme

Les liens d'héritage en JAVA sont signifiés par le mot clé extends placé entre le nom de la classe à définir et celui de la classe dont elle hérite :

```
public class Arc extends Couvrement {}
```

JAVA n'autorise pas l'héritage multiple mais permet de le simuler par le biais des interfaces. Le mécanisme d'interface de JAVA s'appuie sur le concept de classe abstraite. Mais dans une classe interface, toutes les méthodes sont abstraites : les méthodes abstraites ne sont plus identifiées par le mot clé abstract. La définition d'une interface est proche de celle d'une classe, mais la syntaxe impose l'utilisation du mot Interface à la place du mot classe:

```
public interface ObjetVrml {}
```

L'interface est une classe dont on n'hérite pas mais que l'on implémente. L'interface simule l'héritage multiple en autorisant une classe à implémenter plusieurs Interfaces et à hériter d'une seule classe. La différence tient au fait que le programmeur doit impérativement fournir le code de toutes les méthodes de l'interface. Autrement dit, une classe B ne peut hériter que d'une classe abstraite A (et donc d'un seul jeu de méthodes) mais elle peut implémenter plusieurs classes Interfaces (et donc redéfinir plusieurs jeux de méthodes). Nous avons utilisé ce mécanisme pour autonomiser un objet dédié à la gestion des représentations en VRML, l'interface ObjetVrml, qu'implémente la classe ItemTopos. La syntaxe de la définition de la classe devient dès lors:

```
public class ItemTopos extends ItemMoma implements ObjetVrml {}
```

L'interface ObjetVrml est ainsi implémenté au niveau de généricité maximal des concepts architecturaux, mais il l'est également dans la spécialisation des Points pour assurer la visualisation en VRML des seuls points mesurés au cours du processus de mesurage.

JAVA est un langage polymorphique si l'on s'en tient à la définition suivante du mot polymorphisme: caractéristique d'une méthode à rendre des services différents en fonction de l'objet sur lequel elle est appelée. Mais on notera que JAVA interdit la modification du profil d'une méthode redéfinie par rapport à sa définition initiale. La méthode d'écriture du rapport html définie dans la classe Entité ainsi :

```
public void ecritureRapportHtml (StringBuffer html, Entier niveau) { …
   }
```

est redéfinie dans ses sous-classes, comme par exemple ArcPleinCintre avec le même profil :

```
public void ecritureRapportHtml (StringBuffer html, Entier niveau) {
String name = new String(this.getClass().getName());
super.ecritureRapportHtml(html, niveau);
StringBuffer dec = new StringBuffer("");
niveau.inc(2);
for (int i = 0; i < niveau.entier(); i++) {
dec.append(" ");
html.append(dec.toString() + "<B>" +
name.substring(name.lastIndexOf(".") + 1) + "<BR></B>");
}
html.append("  Dans ecriturerapportHTML RayonIntrados vaut:"
+(quelAttribut("rayonIntrados").dimLx().valeur()));
}
```

et ne peut l'être sous la forme suivante:

```
public String ecritureRapportHtml (StringBuffer html) {}
```

On notera au passage également qu'une méthode redéfinie ou non doit définir un type de retour. Deux méthodes peuvent porter un même nom dans une classe si et seulement si les types de paramètres et de retour sont différents. Il n'y a donc pas de difficulté à définir par exemple plusieurs constructeurs.

Un objet instancié sur une classe donnée peut être utilisé en temps qu'instance de ses classes parentes par utilisation du casting, ce qui nous permet par exemple de définir une méthode générique de transfert des caratéristiques morphologiques des attributs d'un objet source héritant de ItemArchitectural dans la classe ItemArchitectural (liaison dynamique):

```
public void transfertMorphologie (ItemArchitectural source) {
```

```
    super.transfertMorphologie(source);
    int nb = nbAttribut();
    int i = 0;
    for (i = nb-1; i>=0; i--) {
       effaceAttribut(i);
    }
    for (i = 0; i < source.nbAttribut(); i++) {
      attribut((Attribut)source.attribut(i).clone());
    }
  }
```

où l'attribut casté par l'expression (Attribut)source réalise une duplication affectée à l'itemArchitectural source.

Aspects généraux de la définition des classes

Nous abordons ici quelques points concrets de l'implémentation JAVA relatifs à la définition des classes et aux propriétés (attributs et méthodes) des objets.

*CONSTRUCTEURS / DESTRUCTEURS D'OBJETS*

L'instanciation d'objets à partir des classes les définissant est réalisée en JAVA par appel de l'opérateur new sur un constructeur de classe. Le constructeur est une méthode sans type de retour dont le nom correspond à celui de la classe. De même qu'un nom de méthode peut être utilisé plusieurs fois dans une même classe pour autant que leurs profils soient différents, des constructeurs admettant différents paramètres peuvent être appelés pour instancier des objets à partir d'une même classe. C'est le cas par exemple de la classe Attribut dont nous donnons ici trois constructeurs, servant à définir différents états initiaux des objets instanciés :

```
public Attribut () {
init();
identificateur ("Attribut") ;
}

// constructeur avec parametres par defaut pour ident
public Attribut (ItemArchitectural ent, String s) {
init();
identificateur (s) ;
setOwner(ent);
}

public Attribut (ItemArchitectural ent, String s, int[] d) {
setOwner(ent);
typeEgo = d;
init();
identificateur (s) ;
dimLx.valeur(0.0);
dimLy.valeur(0.0);
dimLz.valeur(0.0);
}
```

Parallèlement aux constructeurs, JAVA permet de définir un déconstructeur (un seul par classe) qui sera utilisé lors de la destruction de l'objet. Le déconstructeur doit forcément se nommer finalize, il ne prend aucun paramètre et ne renvoie aucun type (void).

La définition d'un déconstructeur est optionnelle, elle n'est donc utilisée que lorsque la destruction de l'objet doit s'accompagner d'actions comme dans cet exemple :

```
protected void finalize() {
    try {
                file.close();
    } catch (Exception e) {
    }
}
```

où la méthode finalize ferme un fichier d'entrée-sortie afin de s'assurer que le descripteur de fichier est fermé (cas de la classe réseaux). La destruction effective de l'objet est réalisée par le ramasse miettes

JAVA (GC, garbage collector). La méthode finalize est invoquée lors de la récuperation de l'objet par le garbage collector.
La gestion de la mémoire permettant à un programme JAVA de s'exécuter n'est pas du ressort du programmeur.
D'un côté, l'opérateur new se charge d'allouer de la mémoire à la demande. De l'autre, le ramasse miettes se charge de repérer les objets inutiles, de libérer la mémoire qu'ils utilisent inutilement, et d'exécuter les actions décrites dans les méthodes finalize. Le ramasse miettes JAVA opère de façon totalement automatisée, il fonctionne en permanence dans un thread de faible priorité.
Si ce mécanisme apporte un confort évident (le programmeur est déchargé du souci de gérer la mémoire nécessaire a l'exécution de son programme), son automatisation peut poser problème dans le cadre du développement d'outils pour lesquels la gestion de la mémoire est critique.
Dans l'implémentation des classes architecturales il n'est pas fait usage de la notion de classe abstraite. On l'a vu plus haut, le concept d'Interface implémenté par telle ou telle classe concrète peut apporter une alternative en terme de mécanisme d'abstraction.

*COLLECTIONS D'ELEMENTS*

Dans les classes Réseau, Entité et Attribut il nous fallait gérer des listes d'objets : liste d'entités dans un réseau, liste d'attributs dans une entité, liste d'EGO dans un attribut. Une première piste à étudier est le formalisme de tableaux proposé par JAVA. Dans ce langage, les tableaux sont des objets aptes à représenter des collections ordonnées d'éléments, éléments qui peuvent être des variables des types de base (int, boolean, double, char...) ou des références sur des objets. La taille d'un tableau n'est pas spécifiée à la déclaration mais déterminée quand l'objet tableau manipulé, jusque là simple référence, est effectivement initialisé en utilisant l'opérateur new. Mais cette taille à la création est fixe, elle ne peut plus être modifiée par la suite.
Les tableaux JAVA peuvent être unidimensionnels ou multidimensionnels. Dans le premier cas, la déclaration d'un tableau unidimensionnel se fait en indiquant le type des éléments du tableau (type de base ou Objet) et le nom l'identifiant:

```
Pt[] pts
```

L'accès aux éléments du tableau se fait par indices dans les bornes du tableau de 0 à length-1, length étant une variable d'instance déclarée public des objets tableaux.
Un tableau multidimensionnel est considéré comme un tableau dont les éléments sont eux mêmes des tableaux.

```
float[][] result
```

La première dimension du tableau doit être déclarée à la création du tableau, l'accès aux éléments du tableau se fait par indices dans les bornes du tableau de 0 à length-1 pour la première dimension et dans les bornes de 0 à [expression1].length-1 pour la seconde dimension.

```
result[2][1] = (float)pos.z();
```

Le formalisme de tableaux pose cependant dans notre cas un problème puisqu'ils ne sont pas extensibles. Nous gérons des collections d'objets qui dans le cas des réseaux par exemple ne sont pas de taille fixe. JAVA propose pour ce faire des outils dédiés à la gestion de collections de données sur lesquels nous nous appuyons.
Le package java.util rassemble en effet des classes d'utilitaires parmi lesquelles certaines permettent de gérer les collections de données: classes Vector, Dictionary, Hashtable, BitSet et l'interface Enumeration.
La classe Dictionary est une classe abstraite qui permet de représenter les tableaux associatifs. Elle est dérivée dans une classe représentant les tables de hachage, la classe Hashtable. La classe BitSet permet de manipuler des vecteurs de bits; vecteurs redimensionnables. Enfin, l'interface Enumeration est utilisée itérer sur les éléments d'une suite.
Nous nous servons essentiellement de la première classe citée, java.util.Vector, qui définit des structures semblables aux listes, de dimension variable, et dont les éléments peuvent être des objets quelconques (de types différents).
Les instances de la classe Vector peuvent être vus comme des tableaux automatiquement agrandis en fonction des besoins. Dans cette structure, plus souple que les tableaux, chaque élément doit contenir uniquement des références à des objets (et par conséquent pas de types de base). Ceci convient bien à notre cas puisque nous devons gérer des listes de concepts architecturaux ou géométriques.

La classe Vector est définie par les propriétés suivantes :
- elementData (éléments de l'objet Vector)
- elementCount (nombre d'éléments)
- capacityIncrement (fixant de combien le vecteur doit être étendu quand c'est nécessaire).

La classe Vector définit des objets qui contiennent toutes les méthodes qui permettent de gérer ces listes (ajouter, supprimer, énumérer).
L'instruction :

```
Vector  morphologie = new Vector();
```

construit un vecteur initialement vide. Comme pour les tableaux, l'accès aux éléments d'une instance de la classe Vector se fait par un indice. La classe Vector comporte de nombreuses méthodes qui permettent d'ajouter, d'insérer, de supprimer ou de rechercher des éléments. Les instructions suivantes en introduisent un certain nombre:

```
public Attribut attribut(int i) {
    if ((i >= 0) && (i < nbAttribut()))
      return (Attribut) morphologie.elementAt(i);
    else return null;
}
public void attribut(Attribut c) {
    morphologie.addElement(c);
}
public void effaceAttribut(int i) {
    if ((i >= 0) && (i < nbAttribut()))
       morphologie.removeElementAt(i);
}
  public int nbAttribut() {
    return morphologie.size();
}
  public void attribut(Attribut p, int n) {
    if ((n >= 0) && (n < nbAttribut()))
      morphologie.setElementAt(p, n);
}
```

qui s'appliquent au cas de la liste d'attributs morphologie définie dans la classe ItemArchitectural. En systématisant le recours à la classe Vector pour gérer nos listes d'objets, nous visons avant tout une meilleure lisibilité des développements puisque les méthodes qui permettent de gérer ces listes sont déportées dans une classe à laquelle chaque développeur peut faire appel.

*DUPLICATION D'OBJETS*

Autre choix d'implémentation systematisé dans la hiérarchie de concepts architecturaux, les méthodes d'affectation et de duplication sont à replacer dans le cadre des spécificités du langage. En Java toute variable d'un type tableau ou classe a la sémantique des accès par référence: elle est un pointeur géré automatiquement par le compilateur. Un objet JAVA est toujours accédé par référence, un type primitif JAVA est toujours accédé par valeur. Enfin, une fois initialisée, une référence peut encore être changée. La désignation de la référence elle-même est differenciée de l'accès à l'objet référencé par l'opérateur "." accompagnant l'occurrence de l'objet, comme le montre cet exemple pris dans la méthode `ecrireLaCamera` de la classe Reseau:

```
Pt v1;
Pt v3;
Pt axe;
// affectation des quatre instances de la classe Pt
v1.x(0.0);
v1.y(1.0);
v1.z(0.0);
// accès à l'objet référencé par v1
axe = v3;
// axe est modifié et référence désormais le même objet que v3
```

La conséquence de l'accès par référence est le caractère superficiel de l'affectation par l'expression **a=b**. Celle-ci créé une nouvelle référence sur l'objet **a**, mais n'en fait pas une duplication. Chaque modification ultérieure de **a** se répercute dans **b** : l'objet n'est pas dupliqué mais partagé. Dans le cas par

exemple de la duplication souhaitée d'entités mesurées à fins d'analyses comparatives, cela pose un évident problème. La redéfinition de la méthode de clonage de la super classe JAVA Object permet d'y remédier. L'opération de duplication doit être adaptée à chaque classe pour que la méthode clone retourne un nouvel objet dont la valeur initiale est une copie de l'état de l'objet auquel la méthode a été appliquée. Dans notre cas la copie profonde est effectuée en deux étapes:

```
public Object clone() {
    ItemTopos ia = new ItemTopos();
    ia.affecte(this);
    return ia;
}
public Object clone() {
        ArcPleinCintre e = new ArcPleinCintre();
        e.transfertMorphologie(this);
        return e;
}
```

Une méthode clone est redéfinie dans chaque classe et retourne l'objet dupliqué.

```
public void affecte (ItemTopos a) {
   super.affecte(a);
   transfertMorphologie(a);
   lumiereVrml.affecte(a.lumiereVrml());
   boiteEnglobante (a.boiteEnglobante());
}
```

Une méthode nommée affecte est chargée de transférer les valeurs des propriétés des concepts à dupliquer.

*EXCEPTIONS*

En JAVA les exceptions et leur traitement font partie du noyau du langage : leur gestion est obligatoire pour toute méthode pouvant potentiellement générer une exception, exception qui doit alors être récupérée (instruction try) ou renvoyée (mot clé throws dans la déclaration de la méthode) sous peine de déclencher une erreur à la compilation[232]. Les exceptions JAVA interrompent l'exécution d'une méthode et sont notifiées à la méthode en cause ou à celle(s) qui l'a appelée. Ce sont des instances des classes de la hiérarchie des Exceptions, hiérarchies que le programmeur peut étendre en fonction de ses impératifs particuliers. En effet, la génération d'une exception se fait en utilisant le mot clé throws suivi du nom d'un objet dont la classe est dérivée de la classe JAVA Throwable.

La gestion d'erreur par exceptions permet d'isoler le traitement d'erreur des instructions à exécuter en cas d'exécution normale, c'est donc un mecanisme assurant une meilleure lisibilité du code produit.

L'instruction throws monException permet de déclencher l'exception monException, référence sur une instance de la classe MonException. MonException comme toutes les classes d'exception est dérivée de la classe Throwable ou d'une de ses classes dérivées. L'exception JAVA peut être déclenchée par le système mais aussi n'importe où dans une méthode :

```
public void nouvelleEntite(Entite e) throws
java.rmi.RemoteException;
```

Le bloc try-catch permet de décrire les séries d'instructions susceptibles de déclencher une exception et les instructions à exécuter si une exception est déclenchée. Le bloc suivant le mot clé try correspond au traitement normal du programme, traitement susceptible de déclencher des exceptions. Il peut inclure à son tour d'autres instructions déclenchant des exceptions. Le bloc suivant le mot-clé catch (suivi entre parenthèses du type de l'exception interceptée) spécifie les instructions à exécuter en cas de déclenchement d'exception, et doit être même si aucune instruction n'est prévue suivi de {} :

```
public void initUstn (String repertoire, String ressource, String
cellule) {
    try {
      fichUstn = new StreamDGN(repertoire + identificateur() + ".dgn",
ressource + seed3d.dgn");
      fichCelUstn = new StreamCEL(cellule, fichUstn.UO());
    } catch (Exception e) {}
```

[232] [Flanagan, 1997, p43] explique ainsi comment le programmeur peut savoir si une méthode est susceptible de déclencher une exception : "lire la documentation ou ne rien faire et attendre que le compilateur vous dise ou placer vos déclaration *throws* (…)".

```
}
```

Si le bloc try est susceptible de déclencher des exceptions de divers types, le bloc try doit être suivi d'un bloc catch par type d'exception à intercepter. Le dernier bloc catch peut être suivi de l'instruction finally spécifiant les instructions qu'il faut toujours exécuter à la suite du bloc try si aucune exception n'a été déclenchée ou à la suite du traitement d'un catch. En fait, les exemples ci-dessus correspondent aux alternatives possibles pour gérer les exceptions déclenchées par une méthode :

- Interception de l'exception par un bloc catch écrit dans la méthode.
- Renvoi de l'exception par la déclaration dans l'en-tête de la méthode du type d'exception potentiellement renvoyée. La méthode appelante devra alors à son tour intercepter ou renvoyer l'exception.

Aspects spécifiques à certaines classes

Nous abordons ici quelques points concrets de l'implémentation JAVA relatifs à la définition de certaines classes et de leurs méthodes spécifiques.

*METHODES FINALES*

Dans le cas d'un héritage simple, la machine virtuelle JAVA se charge à l'appel d'une méthode de lancer la méthode la plus spécifique dans une hiérarchie de classes en fonction de l'objet requérant. Si la méthode n'est pas trouvée dans la classe de l'objet, elle est recherchée de façon ascendante dans l'arbre. Cette recherche, coûteuse en temps, peut être évitée en déclarant la méthode finale. La méthode est dès lors reconnue par le compilateur quel que soit l'objet requérant comme celle qui doit être exécutée. La localisation de la méthode est écrite directement dans le code machine (byte code) produit par la JVM. Ce formalisme d'optimisation n'est pas utilisé dans les classes architecturales, où son application n'a pas encore été possible, mais dans plusieurs classes outils.

*GESTION DES ENTREES-SORTIES*

Le package java.io contient la hiérarchie des (nombreuses) classes JAVA dévolues à la gestion des entrées-sorties, et particulièrement a la de gestion des flux de données. Ces classes permettent d'accéder aux données soit en lecture (classes dérivant de la classe abstract InputStream), soit en écriture (classes dérivant de la classe abstract OutputStream). La plupart des méthodes d'entrée/sortie déclenchent une exception de classe IOException ou de ses dérivées EOFException, FileNotFoundException, InterruptedIOException ou UTFDataFormatException à intercepter.

Les données entrantes ou sortantes sont manipulables avec les mêmes classes, quelque soit le type de source de données:

- un fichier (classes FileInputStream et FileOutputStream),
- une chaîne de caractères (classe StringBufferInputStream),
- un tableau d'octets (classes ByteArrayInputStream et ByteArrayOutputStream),
- un pipeline (classes PipedInputStream et PipedOutputStream),
- l'entrée et les sorties standards (champs System.in, System.out et System.err ou FileDescriptor.in, FileDescriptor.out et FileDescriptor.err),
- une URL,
- une socket.

Toutefois, dans le cas de notre implémentation, la lecture de données sur un fichier peut se faire sur le serveur via une URL ou localement. Or les méthodes de lecture sur un fichier de la classe File (fichiers locaux) diffèrent de celles à mettre en œuvre dans le cas d'applets. Une classe LectureFichier a donc été introduite pour rendre les accès à un fichier transparents. Cette classe définit des objets capables de lire sur un fichier local ou non nommé `nom` et de renvoyer son contenu sous la forme d'un objet de type `String`. Ces objets sont instanciés et utilisés à l'intérieur de méthodes des classes architecturales, comme par exemple dans le cadre de l'évaluation des valeurs par défaut des entités :

```
public void lectureTableReference(String nom, boolean local) {
    LectureFichier lecteur = new lectureFichier(local);
    tableReference = lecteur.lecture(nom);
  }
```

*VARIABLES DE CLASSES*

En JAVA il est possible de déclarer des méthodes et des variables de classe. Les variables de classe, copies d'une variable partagées par tout objet de la classe, sont utilisées dans notre implémentation pour assurer la cohérence d'état par rapport au mesurage de l'ensemble d'entités d'un réseau mesuré:

```
public static final int BRUTE = 0;
```

Les méthodes de classe opèrent uniquement sur des variables de classe. Il n'y a pas de méthodes de classe dans les classes architecturales proposées.

*VARIABLES FINALES (CONSTANTES)*

Dans l'exemple précédent, le nom de la variable est précédée du mot clé `static` mais aussi du mot clé `final`. Une variable d'instance ou de classe dont la déclaration est précédée du qualifieur `final` est une constante. Initialisée, sa valeur ne peut plus être changée. Dans notre implémentation, les variables finales correspondent aux mêmes usages que les variables de classe.

*SURCHARGE*

Le langage JAVA autorise la surcharge de méthodes mais pas la surcharge d'opérateurs. Pourtant, dans le cas des points par exemple, la surcharge d'opérateurs peut apporter une meilleure lisibilité des méthodes des classes architecturales. Une façon de contourner la difficulté consiste à définir une méthode dans la classe Point dont le nom traduit l'effet:

```
v2 = (entite(i).ref ().origine ().moins (entite(i).ref ().vx
().multiplie (entite(i).nux ())))).x () ;
```

*LA CLASSE ATTRIBUTDIMENSIONNEL*

Cette classe introduite dans l'implémentation JAVA des classes architecturales ne correspond à aucun élément du modèle évoqué aux chapitres précédents. Elle définit en fait des couples valeur / statut représentant les propriétés dimensionnelles des entités et leur statut vis à vis du calcul (dimension calculée, affectée par défaut, non affectée). Les instances de cette classe sont des objets capables d'une part de renvoyer une valeur (méthodes de représentation par exemple) et d'autre part d'informer l'entité et son réseau de son statut vis à vis du calcul. Chaque propriété dimensionnelle des entités dans l'implémentation JAVA est en fait soit un objet de cette classe AttributDimensionnel :

```
private AttributDimensionnel longueurEngravee;
```

soit un objet de la classe Attribut :

```
private Attribut isolation;
```

lui même héritant de trois objets de cette classe AttributDimensionnel

```
public  AttributDimensionnel dimLx;
public  AttributDimensionnel dimLy;
public  AttributDimensionnel dimLz;
```

*METHODE DE FACETTISATION DES PROFILS*

Comme indiqué précédemment, un profil mouluré est décrit par une liste de moulures, une liste de points de contrôle et de types correspondants (convexe demi-rond, concave quart de rond, etc..). La méthode permettant de générer la visualisation du profil se charge le cas échéant de retrouver l'ensemble des points intermédiaires nécessaires à la représentation par facettes du profil. Les paramètres utilisés sont la hauteur de l'Entité appelante et les éléments de définition du profil. Les éléments à évaluer sont tous les points intermédiaires de la représentation, dans le plan de définition de la moulure. Le nombre de points à trouver entre chaque couple de points de contrôle est le niveau de détail de l'entité.

# ANNEXE 43 : IMPLÉMENTATION PERL

## INSTANCIATION / PERSISTANCE

Toute classe PERL définit un constructeur habituellement nommé new, et laisse le soin au programmeur de décider s'il entend procéder à l'initialisation de l'objet en même temps qu'à sa création ou s'il préfère l'en séparer. La différence majeure entre JAVA et PERL ici est qu'en PERL, le constructeur est non seulement chargé de la création de l'objet mais aussi de la définition des attributs de l'objet à créer. En effet, les variables d'instance ou attributs d'une classe PERL ne sont pas définis directement dans la classe mais seulement dans la méthode de création de l'objet, le constructeur new:

```
sub new {
        my $proto = shift;
        my $class = ref($proto) || $proto;
        my $self  = $class->SUPER::new();
        $self->{angleRampantYPos}  = undef;
        $self->{angleRampantYNeg}  = undef;
        bless ($self, $class);
        return $self;
    }
```

Chaque classe PERL est définie dans un fichier texte d'extension ".pm" qui contient, en en-tête, trois informations:

```
package MurPignon;
use classes::MurTransversal;
@ISA = ("MurTransversal");
```

Le nom de la classe derrière le mot-clé package, un éventuel chemin permettant de localiser la classe parente (mot-clé use) et la déclaration d'héritage @ISA. Une classe PERL est en effet un package qui se contente de déclarer dans la variable (tableau) @ISA la ou les classes dont il hérite. Je ne rentre pas ici dans une discussion sur la gestion de l'héritage multiple en PERL puisque nous ne nous en servons pas, mais renvoie à [Conway, 2000, pp168-203]. Deux points méritent d'être notés dès à présent :

- Les variables d'instance ne sont pas typées, une valeur indéfinie leur est assignée (mot clé undef, undefined value).
- Le tableau @ISA ne définit qu'un contenant non typé, de taille non spécifiée, pour une liste de valeurs.

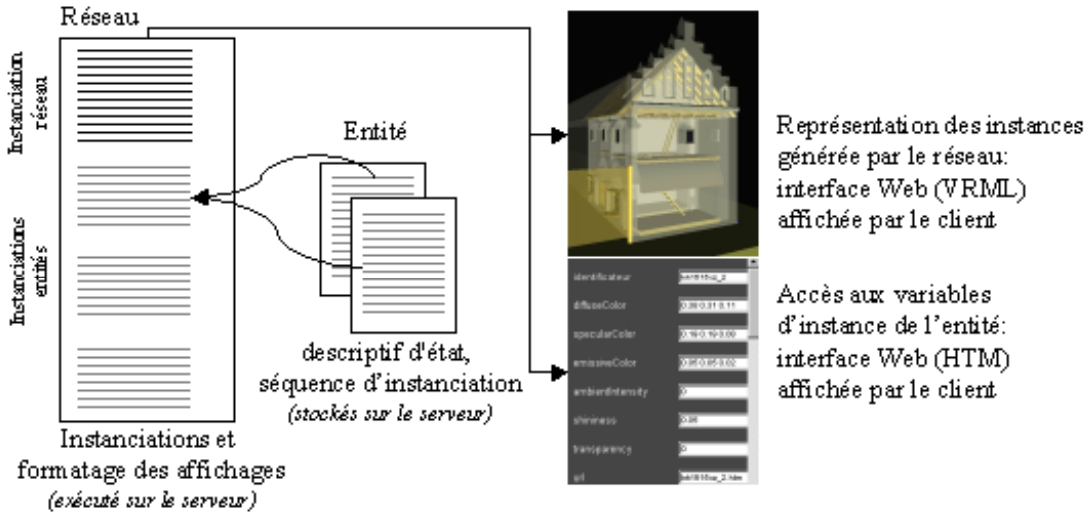



*Figure 154 : Mécanisme d'instanciation et de ré-instanciation des objets dans l'implémentation PERL*

Comme en JAVA, les objets PERL sont détruits effectivement par un ramasse-miettes automatique. La méthode DESTROY de chaque instance est appelée systématiquement lors de la destruction de l'objet, qui ici a lieu dès qu'aucune référence à l'objet ne subsiste. Nous n'avons pas dans notre implémentation eu besoin de recourir à cette méthode DESTROY.
PERL ne dispose pas de mécanisme natif permettant de déclarer des variables de classe. Il est toutefois possible de parvenir au même résultat en utilisant une variable lexicale dont le compteur de référence est maintenu non nul.
Dans l'implémentation expérimentée, la persistance des objets architecturaux n'est pas assurée. Ce qui l'est, en revanche, ce sont les valeurs des variables d'instances nécessaires à la ré-initialisation d'objets identiques (dans un même état). Ce mécanisme s'appuie sur les formalismes très simples d'entrée-sortie vers des fichiers qu'autorise PERL. Toutes les entités sont manipulables individuellement au travers du réseau, qui se charge de les instancier sur le serveur Web et de les afficher sur le client Web en concaténant leurs descriptifs d'état individuels. Le schéma ci-dessous illustre la méthode employée :
Si dans l'état actuel du développement les descriptifs d'état[233] des instances et leurs séquences d'instanciation[234] sont stockées sur le serveur, rien n'empêche a priori de les déporter sur le client. Il n'en reste pas moins que cela multiplierait les appels réseau et n'a pas pour nous d'intérêt majeur.

*LISTES D'OBJETS*

Nous avons vu qu'en JAVA la gestion de collections d'éléments dans les classes Réseau, Entité et Attribut était confiée à une hiérarchie d'objets (Vector). En PERL le problème ne se pose pas puisque les tableaux de PERL sont non typés et modifiables dynamiquement. Rien ne s'oppose donc à leur utilisation pour gérer les listes d'objets que contiennent nos classes. Le tableau PERL est un conteneur pour une liste. Une liste est une séquence de valeurs séparées par des virgules et identifiée par un indice. Des accès globaux ou partiels aux tableaux sont possibles. Le tableau @_ contient par défaut les paramètres d'appel d'une méthode:

```
sub setAttributes {
   my $self = shift;
   my $reseau=@_[0];
   my $param=@_[1];
```

Le tableau déclaré par : $self->{entites} = [] dans le constructueur du réseau représente le liste d'identificateurs des entités du réseau, liste accessible par une méthode d'entrée sortie qui, avec arguments, fixe une(des) valeurs, et sans argument la(les) récupère:

```
sub entites{
        my $self = shift;
        if (@_) { @{ $self->{entites} } = @_ }
        return @{ $self->{entites} };
    }
```

Une liste contenant des valeurs parmi lesquelles figurent une liste est par défaut aplatie (flattened), permettant d'accéder aux valeurs des listes contenues sans référence à elles. Il est toutefois possible d'avoir recours à une syntaxe spécifique (\@) pour écrire des tableaux multidimensionnels.

*ENTREES-SORTIES*

Les tâches communes d'accès aux fichiers (ouverture, lecture ou écriture) en PERL utilisent des fonctions et des opérateurs d'entrée-sortie simples. La fonction d'entrée-sortie basique est la fonction open qui prend en argument un descripteur de fichier (symbole utilisé pour représenter le fichier quand on le lit ou quand on écrit) et une chaîne correspondant au nom du fichier. Les notations des trois modes d'accès les plus répandus sont < pour la lecture, > pour l'écriture et >> pour la concaténation. La fonction close avec comme argument le descripteur de fichier permet de fermer ce dernier quelque soit le type du flux considéré:

```
my $htmout= $self->interfaceVrml->htmFile();
open (OUT, ">>$htmout");
print OUT "<EMBED SRC=\"".$ficVrml."\" WIDTH=490 HEIGHT=365 ALIGN=center>\n";
close (OUT);
```

[233] Nous appelons "descriptif d'état" les valeurs des variables d'instance d'un objet réévaluées après intervention sur un jeu d'objets.
[234] Nous appelons "séquence d'instanciation" l'activation à partir du descriptif d'état d'un objet de ses méthodes pour par exemple produire les représentations en VRML ou HTML.

PERL laisse à la charge du programmeur la vérification des valeurs de retour des fonctions open et close (et des fonctions d'entrée-sorties en général). La fonction flock (descripteur de fichier, statut du verrou) permet de verrouiller un fichier devant être accédé par plusieurs processus. Cette fonction du langage se substitue aux techniques de verrouillage des systèmes d'exploitation. Je ne présenterai pas ici de façon plus étendue les fonctions d'entrées-sorties de PERL et leurs applications, je renvoie pour cela à [Christiansen et al, 199, pp261-361] qui les décrivent avec détail. Je préciserai seulement qu'avec les versions 5 et ultérieures du langage des modules (nommé IO) de manipulation des fichiers sont apparus, déchargeant le programmeur de tâches comme la copie de fichiers par exemple. Dans notre implémentation, l'accès aux fichiers est un point central puisque nous nous appuyons sur une architecture statique : le client fait effectuer sur le serveur une série d'opérations visant à mettre à jour le jeu de représentations du modèle que le serveur lui a transmis. Le client ne dispose que de ce jeu de représentations. Le serveur, lui, ne dispose que de descriptifs d'état et reste à l'écoute du client pour d'éventuelles ré-évaluations du jeu d'instances prenant en compte d'une part les descriptifs d'état et d'autre part les choix utilisateur transmis depuis le client. Autrement dit, seule la présence de fichiers fixant côté serveur un état du jeu d'instances permet au client de faire effectuer sur ce jeu d'instances une série d'opérations. Les performances relatives à ce jeu d'opérations sont ici celles du serveur, point non négligeable dans un environnement Web. Nous sommes donc dans une solution où seul le serveur effectue des traitements, solution à l'opposé des choix possibles en JAVA. Le coût du transfert (en particulier dans le cas des scènes VRML réévaluées côté serveur et transmises aux clients) peut être excessif : on verra que nous distinguons dans l'interface l'action de réévaluation des instances de l'action de réaffichage pour partiellement résoudre ce problème. Ce qui, en Perl, est gagné d'un côté (opérations à la seule charge du serveur) peut apparaître comme perdu de l'autre (temps de transfert des résultats). Nous pensons que seule la remise en perspective du choix technique par rapport aux objectifs poursuivis permet de justifier l'une ou l'autre des solutions. Dans le cadre du projet pour lequel ce travail a été élaboré, le choix PERL fut délibéré.
Par ailleurs, l'écriture de rapports ASCII (fichiers HTML, VRML ou autres) côté serveur permet de réemployer les résultats d'une évaluation de réseau dans le cadre d'autres applications autour du modèle, en particulier dans le cadre de l'outil SOL présenté plus loin.
Plus encore, dans l'application VALIDEUR (instanciation de réseaux côté serveur, visualisation et interaction côté client), c'est à la fois l'application (classes PERL interprétées dynamiquement) et les données servant à l'instanciation des réseaux et de leurs entités qui sont traités sous forme de fichiers ASCII. En procédant ainsi, nous allons dans le sens d'un outil entièrement portable, objectif majeur du projet concerné.

*METHODES D'AFFICHAGE DES OBJETS*

On vient de le voir, le portage PERL des classes architecturales fait un usage immodéré des entrées-sorties sur des fichiers ASCII. Cette logique, ancrée dans la tradition SGML-HTML, est cependant assez éloignée du développement d'applets JAVA. Nous allons donc ici rapidement détailler quels sont ces fichiers et quelles méthodes des objets sont chargées de les lire ou de les écrire.
Nous distinguons deux grandes familles de fichiers:

- Les fichiers formattés pour interprétation par le navigateur. Ces fichiers sont transmis au client et affichés dans l'espace du navigateur. Ils forment l'interface entre l'utilisateur et le système et sont à dessein écrits dans les formats standards HTML et VRML.
- Les fichiers de données utilisateur stockés sur le serveur. Ces fichiers contiennent les informations relatives à l'instanciation d'un réseau organisé par l'utilisateur côté client, et stockés côté serveur entre deux opérations de ré-évaluation.

Une hiérarchie de classes PERL est chargée de gérer la création des interfaces affichées côté client. Ces classes se chargent d'abord de l'écriture des fichiers HTML transmis au client. Ces fichiers présentent à l'utilisateur des formulaires classiques lui permettant de modifier les variables d'instances des objets en cours ou d'en instancier de nouveaux. Ces classes sont également chargées du décodage des sélections utilisateur correspondantes, c'est à dire en particulier de l'écriture de nouvelles valeurs dans les descriptifs d'état des objets existants ou de l'écriture de nouveaux descriptifs d'état pour les entités instanciées. Nous reviendrons plus en détail sur cette hiérarchie de classes dans la section suivante. L'écriture du fichier VRML correspondant à la scène en cours est du ressort du réseau qui fait appel aux méthodes

correspondantes des entités qu'il contient. Chaque ré-évaluation d'un réseau fait intervenir une série de méthodes d'écriture puis d'affichage des objets :Lecture de la configuration du système (objets disponibles au moment de la ré-évaluation).
Lecture du contenu du réseau et calcul des points de vue standards.
Ecriture de l'en-tête du fichier VRML.
Pour chaque entité (modifiée),

- lecture / écriture (éventuelle) du descriptif d'état,
- écriture d'une section du fichier VRML,
- écriture de la séquence d'instanciation,
- écriture du formulaire HTML individuel de l'entité.

Ecriture de la clôture du fichier VRML.
Ecriture de la séquence d'instanciation du réseau.
Ecriture des pages / formulaires HTML relatifs au réseau.
Les méthodes d'affichage des entités sont donc indissociables des méthodes d'écriture des descriptifs d'état ou des séquences d'instanciation comme la suivante :

```
sub writeDatFile{
my $self = shift;
my $updateFile= "./".$self->identificateur().".dat";
my $nomVariableDansScript= $self->identificateur();
  open (OUT, ">$updateFile");
       print OUT "use classes::ArcPleinCintre;\n";
       …
       print OUT "my \$".$nomVariableDansScript."= ArcPleinCintre->new();\n";
   close (OUT);
   $self  ->SUPER::writeDatFile($updateFile);
}
```

La plupart de ces méthodes sont surchargées par chaque objet, ou dans le cas de la méthode d'écriture du fichier VRML font appel à des sections génériques définies dans la classe abstraite Entité :

```
sub ecritureWrl {
 my $self = shift;
$fic=$self->ficVrml();
$self->initWrl();
$self->writeSensorsWrl();
open (OUT, ">>$fic");
close (OUT);
$self->closeSensorsWrl();
$self->endWrl();
}
```

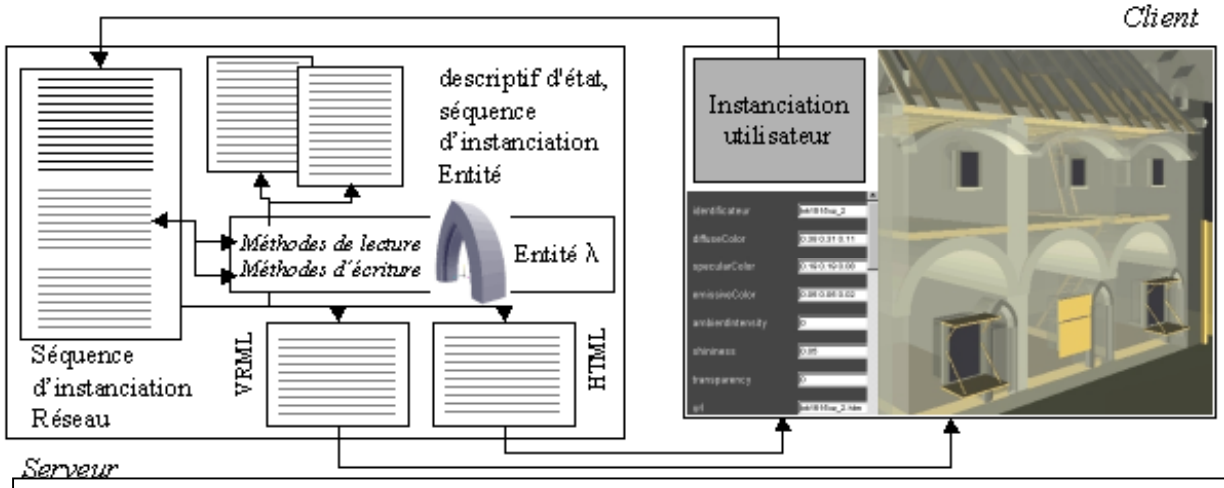



*Figure 155 : Appels aux méthodes des entités dans la séquence d'instanciation des réseaux*

La figure ci-dessus situe l'activation des méthodes d'affichages des objets dans le processus d'instanciation d'un réseau.

## *ANNEXE 44 : LANGAGE DE DESCRIPTION DES SCENES*

Principe

Dans le cadre des réseaux libres le modèle architectural théorique est décrit dans un script ASCII qui peut être lu et interprété par un analyseur syntaxique membre de la classe Réseau. Chaque réseau possède un analyseur syntaxique, ce réseau est décrit par un script qui permet d'accéder aux entités qui le composent et aux grandeurs qui le caractérisent. Le lien entre le script et le réseau est assuré par un identificateur commun au réseau et au script. Les règles générales des langages de programmation structurés ont été retenues : les variables sont typées et doivent être déclarées avant leur utilisation. Le choix technique qu'a opéré P.Drap dans le cadre de l'implémentation initiale du modèle en C++ s'est porté sur une grammaire régulière mise en œuvre par des outils standards du monde UNIX destinés à la production de langages réguliers. Flex++ et Bison++ sont développés par Alain Coëtmeur[235], [Coëtmeur, 1993]. Il s'agit d'adaptations des célèbres FLEX et Bison du GNU, eux-mêmes adaptations de leurs ancêtres LEX et YACC, générateur de compilateurs sur UNIX.
Le vocabulaire utilisé correspondait aux entités architecturales et à leurs positions relatives, comme l'illustre l'exemple ci-dessous où:

```
Régner (Chapiteau1, Chapiteau2) ;
Au Droit (Fut1, Fut2, d = 1.230) ;
Aplomb (Chapiteau1, Fut1) ;
Aplomb (Chapiteau2, Fut2) ;
```

sont quatre phrases valides dans lesquelles :

- *Régner* correspond à une altitude commune des entités considérées.
- *Au Droit* correspond au positionnement des entités considérées sur un axe commun en plan avec un paramètre fixant une distance entre elles.
- *Aplomb* fixe un axe vertical commun aux entités considérées.

Le principe du passage par un script ASCII pour organiser les réseaux a été repris dans l'implémentation JAVA avec plusieurs avancées importantes.

Évolutivité : une implémentation du raffinement de classes

Cette première version de LaDemarche avait pour inconvénient majeur son aspect statique. Grammaire de type LALR, elle utilisait un analyseur LEX et était donc dotée après compilation d'un ensemble fini d'items de vocabulaire. Le passage à JAVA a permis de transformer ce langage statique en un langage dans lequel il est possible d'intégrer de nouveaux concepts sans recompilation. Le formalisme des Métaclasses JAVA permet de lier le nom d'une classe avec sa représentation en machine. Il est donc possible d'avoir accès aux méthodes d'objets sans fixer un jeu de tokens dans un dictionnaire.
Dans cette seconde version de LaDémarche, P.Drap a également introduit la notion d'objet mutant, formalisée en UML mais pas implémentée en JAVA. Toutes les instances d'une classe donnée disposent d'un ensemble de propriétés dont le contenu peut varier pendant l'existence de l'instance. Par contre, ces propriétés ne peuvent pas disparaître, pas plus que de nouvelles propriétés peuvent être ajoutées à l'instance. L'ajout de propriétés à une instance implique en fait un changement de classe.
Par ailleurs, la mutation d'une instance d'une classe vers une classe "sœur" ne peut être effectuée sans prendre le risque de perdre une partie des informations (propriétés) de la classe initiale. L'implémentation proposée est donc une mutation restreinte au raffinement d'instances, c'est à dire au strict ajout de propriétés. Ce raffinement de classe permet à une instance d'une classe A d'être raffinée en instance d'une sous-classe de A. Nous sommes ainsi certains qu'à la suite de cette opération tous les attributs de l'instance mutante sont renseignés. Les nouveaux attributs définis dans la sous-classe sont affectés par des valeurs par défaut.
Cette approche a été développée pour manipuler au sein d'une même structure de contrôle un groupe hétérogène d'objets architecturaux. Ainsi, par exemple, nous pouvons grouper dix solives dont deux sont des murières (dérivées de Solives) de la façon suivante :

[235] Alain Coëtmeur, *"Bison++ et Flex++ "*, Annexe à *« Bison, the YACC-compatible Parser Generator »* par Charles Donnely, Richard Stallman, Edité par la Free Software Foundation, 675 Massachusetts Av. Cambridge, MA 02139 USA,Décembre 1992.

| | | |
|---|---|---|
| Solive ensSolive[10];<br>Un tableau de dix solives | ensSolive[0].rafinement(Muriere);<br>l'instance 0 est raffinée en instance de la classe Muriere | ensSolive[9].rafinement(Muriere); l'instance 9 est raffinée en instance de la classe Muriere |

Ce script LaDemarche définit un tableau polymorphe de dix Solives dont deux sont des Murières. Nous reparlerons du langage de description de scènes dans la section "Expérimentations" et renvoyons pour plus de détails sur son implémentation à la bibliographie de son auteur.

## ANNEXE 45 : SQLSERVER, CONFIGURATION, INDEX, INTÉGRITÉ ET TRAITEMENTS DES CONTENUS

SQL Server : configuration, index et intégrité.

L'installation de SQL Server est recommandée sur un système d'exploitation NTserver 4.x avec système de fichiers NTFS. Le principal écueil que nous avons rencontré ici est la définition du jeu de caractères par défaut.

En effet, une fois un jeu de caractère choisi, il devient impossible d'une part d'en changer sans réinstaller le serveur et d'autre part de mélanger des bases codées avec différents jeux de caractères, ce qui était notre besoin. Nous avons donc dû pré et post-traité les caractères Latin 2 (Europe Centrale) au travers de modules PERL (pré-traitement) et de scripts Javascript (post-traitement).

Le souci de performance n'est pas majeur dans notre cas (bases de poids réduit). Disons néanmoins que deux types d'index sont gérés dans SQL Server, et ce en arbres binaires équilibrés ( B-Tree ) :

- Index Cluster où les enregistrements sont ordonnés physiquement dans la base en fonction de l'index.
- Index Non Cluster où les enregistrements ne sont pas triés.

Marc Israel [Israel, 1997] rapporte les critères de choix d'une solution ou de l'autre dans le chapitre 7 de son ouvrage, intitulé "*Optimisation*". Nous avons choisi en conséquence d'utiliser des Index non cluster.

Les règles d'intégrité d'une base sont implémentées sur SQL Server de la façon suivante :

*INTEGRITE DE DOMAINE*

Elle est assurée par quatre mécanismes, un type de données, une règle de validation, une valeur par défaut et la clé étrangère sur lesquels nous revenons en indiquant notre position.

SQL Server propose dix-neuf types de données systèmes et permet à l'utilisateur de définir ses propres types de données. Nous n'avons pas eu à tirer parti de cette dernière possibilité, et n'avons fait qu'un choix susceptible d'être rapporté ici, celui de ne pas faire appel aux types de données système gérant les dates. La raison en est qu'il nous a semblé trop aléatoire d'imposer un format de date unique pour des champs pouvant comprendre à la fois une date précise, une date réduite à l'année, ou bien encore une plage de dates hétérogènes. Nous traitons les dates sous forme de chaînes de caractères pour lesquelles des modules d'interfaçage appropriés effectuent les opérations de tri.

*INTEGRITE D'ENTITE*

Elle assure l'unicité de chaque enregistrement, généralement par l'existence d'une clé primaire (elle peut également être assurée par la contrainte UNIQUE acceptant les valeurs NULL). Dans notre cas il n'était pas possible d'assurer l'unicité des enregistrement sans faire appel à un champ dédié, champ de type Entier nommé "id" qui contient la clé primaire des tables.

*INTEGRITE REFERENTIELLE*

Elle assure la cohérence entre plusieurs tables en rendant impossible :

- L'insertion d'enregistrements dans une table enfant s'il n'existe pas d'enregistrement associé dans la table parent.
- La suppression d'enregistrements dans la table parent s'il existe des enregistrements associés dans la table enfant.
- La mise à jour de colonnes d'enregistrements dans la table parent s'il existe des enregistrements associés dans la table enfant.

Dans notre cas, les bases utilisées ne contiennent pas de tables associées.

*INTEGRITE D'ENTREPRISE*

Elle représente les règles utilisateur relatives à la cohérence de chaque table d'une base. De façon générale, ce mécanisme aurait pu nous intéresser. Mais il faut rappeler que nous nous plaçons dans un environnement Web dans lequel la détection d'erreurs par le SGBD n'est pas aisément traduisible en termes clairs côté client en cas de manipulation erronée. Nous avons donc privilégié un pré-traitement des enregistrements à ajouter à la base lorsque ce type de contrainte était rencontré.

Nos choix se traduisent par la seule mise en œuvre des mécanismes d'intégrité déclarative de SQL Server, mise en place à la création des tables, et non des mécanismes d'intégrité procédurale que l'outil autorise.

Traitements des contenus

Nous venons de le voir, de nombreuses tâches sont laissées à la charge de l'interface pour toutes les fonctions relatives à la mise à jour de la base. Nous parlons ici non pas des interfaces de requêtes mais des interfaces dédiées à la mise à jour en ligne de la base documentaire, c'est à dire de l'ajout comme de la modification d'enregistrements. Ces tâches peuvent être résumées de la façon suivante :

- Coder (et décoder) les caractères accentués Latin 2.
- Vérifier la validité des contenus (contraintes intégrité de domaine et intégrité d'entreprise).
- Eventuellement affecter des valeurs par défaut aux champs non renseignés par l'utilisateur.
- Générer une clé primaire valide (contrainte intégrité d'entité).
- Coder les informations relatives aux données (par exemple, identifiant d'un édifice ou d'un élément de corpus).
- Générer de nouveaux codes dans le cas d'ajouts de critères.
- Formuler les requêtes SQL en modification adressées au SGBD.

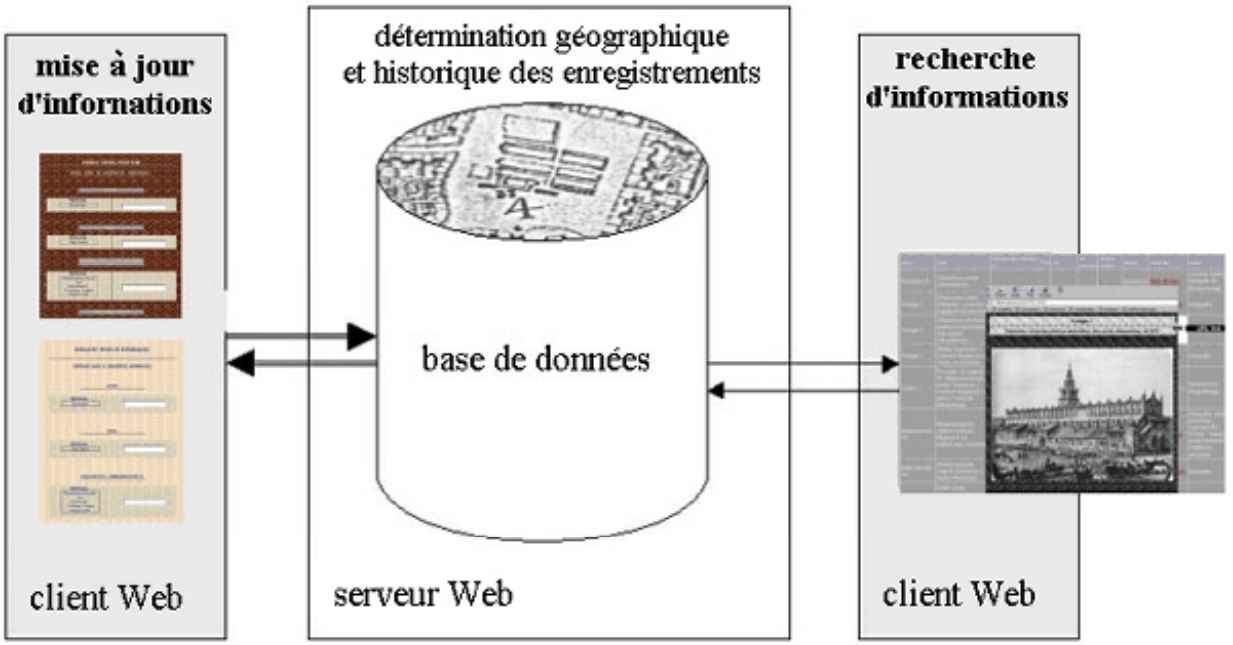



*Figure 156 : l'interfaçage Web, recherche et indexation d'informations.*

Ces tâches, on l'a vu pour la création des interfaces de requêtes, sont prises en charge par un ensemble de modules PERL (voire de simples scripts dans les premiers développements). Ils vont en définitive formater un document idc (voir section suivante) contenant une instruction SQL valide transmise au SGBD.

Ce formatage doit comprendre la création d'interfaces HTML permettant à l'utilisateur côté client de rentrer les contenus par le biais de formulaires standards et la vérification de ces contenus.

Il doit également comprendre la génération d'informations propres à la gestion du système, à savoir la clé primaire et les codes relatifs à la création de nouveaux critères descriptifs (par exemple, un nouveau thème de la liste *problématiques* associée à chaque enregistrement dans la base documentaire SOL). Ils assurent le suivi des connections et l'authentification des utilisateurs puisque nous sommes en mode de sécurité intégrée.

Le document idc formalise la requête adressée au SGBD. Sa création finalise les étapes de pré-traitement.

Elle finalise aussi le formatage des post-traitements. Les modules PERL génèrent à la fois la requête idc mais aussi le formatage des réponses (format htx) qui inclut sous la forme de script Javascript les post-traitements. Ceux -ci sont donc déportés côté client, le serveur ayant fini son travail une fois la requête au SGBD traitée.

## ANNEXE 46 : SQLSERVER, INTERFACES WEB

SQL Server s'appuie sur IDC (Internet Database Connector), une DLL interfaçant le serveur Web Windows IIS et les pilotes ODBC de SQLServer. Le schéma ci-dessous récapitule le parcours d'une requête depuis le navigateur jusqu'au SGBD.

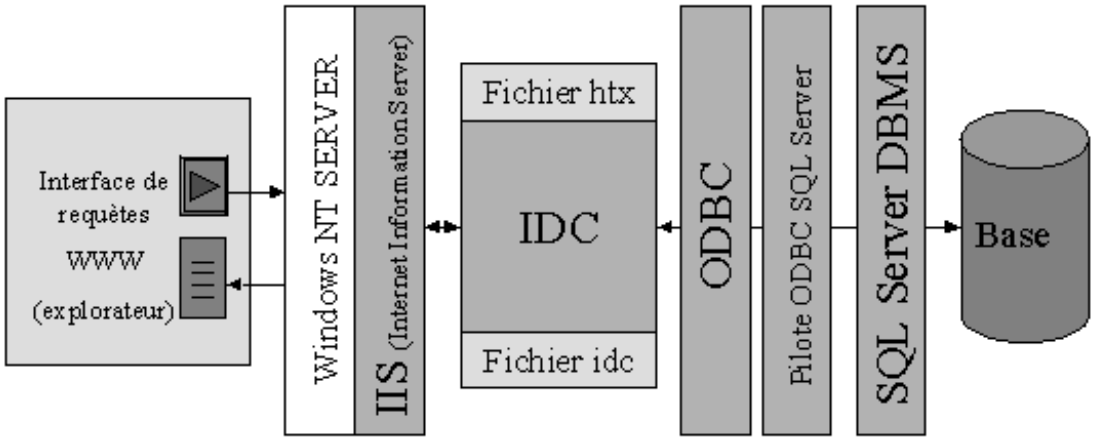



*Figure 157 : Interface IIS / pilotes ODBC SQL Server*

L'interfaçage d'une base SQLServer pour le b commence par la création sur le serveur IIS d'une source de données ODBC, service de Windows NT, qui accèdera au SGBD. Une fois cette source de données établie, deux types de fichiers ASCII sont à créer, les fichiers IDC de formulation des requêtes et les fichiers HTX de formatage des réponses.

Les fichiers IDC contiennent trois informations, le nom de la source de données, le nom du fichier HTX à utiliser et la requête (statique ou dynamique) à exécuter. Les fichiers HTX sont des fichiers en HTML étendu par six mots clés pour répondre aux besoins de traitement des résultats (données et conditions).

Les éléments qui nous intéressent ici sont ceux qui permettent de lier une requête formulée sur une page HTML à la base. Les fichiers IDC et HTX sont dans notre cas écrits dynamiquement. Le tableau ci-après décrit sommairement la syntaxe générique de ces fichiers :

| En tête du fichier IDC |
|---|
| Datasource: diva ***(nom de la base)***<br>Template: ***nomDuFichierHtx***.htx<br>SQLStatement:<br>+ ***Instructions SQL***<br>requête statique:<br>*nomDeChamp= valeur*<br>requête dynamique<br>*nomDeChamp= %nomDuParamètre%* |

| Balises spécifiques du fichier HTX (boucle de lecture des enregistrements) |
|---|
| <%begindetail%><br><%***nomDuChampALire***%><br>…..<br><%enddetail%> |

| Balises spécifiques du fichier HTX (structure de contrôle) |
|---|
| <%if idc.***nomDUChampAControler*** EQ "..."%><br>instructions HTML<br><%else%><br>instructions HTML<br><%endif%> |

Les structures de contrôle des fichiers HTX sont utilisées pour :

- Valider l'affichage des champs en fonction de leur contenu.
- Réagir aux requêtes sans enregistrements valides trouvés.
- Assurer des post-traitements éventuels.

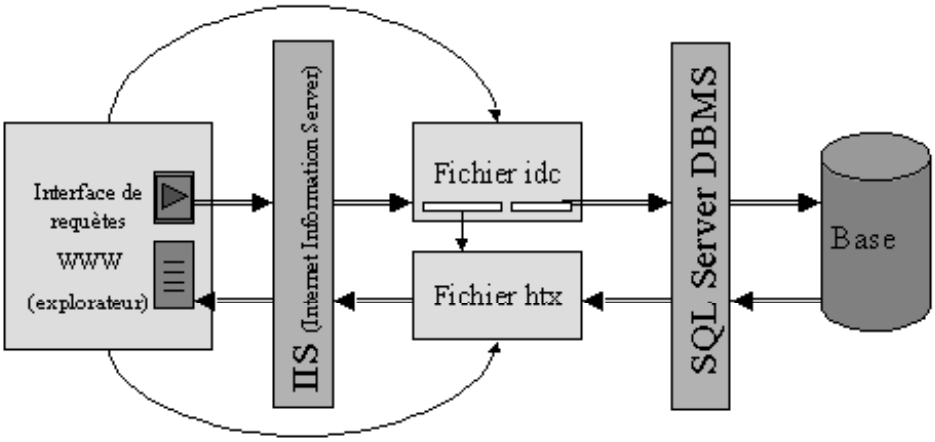



*Figure 158 : Rôle des fichiers IDC et HTX dans l'interfaçage Web du SGBD SQL Server*

Ce principe s'applique en requête comme en indexation dans les interfaces textuels puisque dans les deux cas les modules d'interfaçage vont écrire à la fois les fichiers IDC et les fichiers HTX correspondant aux choix utilisateur.
Dans le cas des interfaces graphiques (Image et VRML) nous sommes toujours dans le contexte de requêtes dynamiques (PULL) mais cette fois-ci les fichiers idc et htx sont écrits indépendamment, l'interface se contentant de les appeler. Ils sont écrits par de petits scripts PERL s'appuyant sur les listes de critères affichées par l'interface textuelle, et sont donc mis à jour en cas de modification de ces listes.

# ANNEXE 47 : APPLICATION HUBLOT, SCHÉMAS DE FONCTIONNEMENT

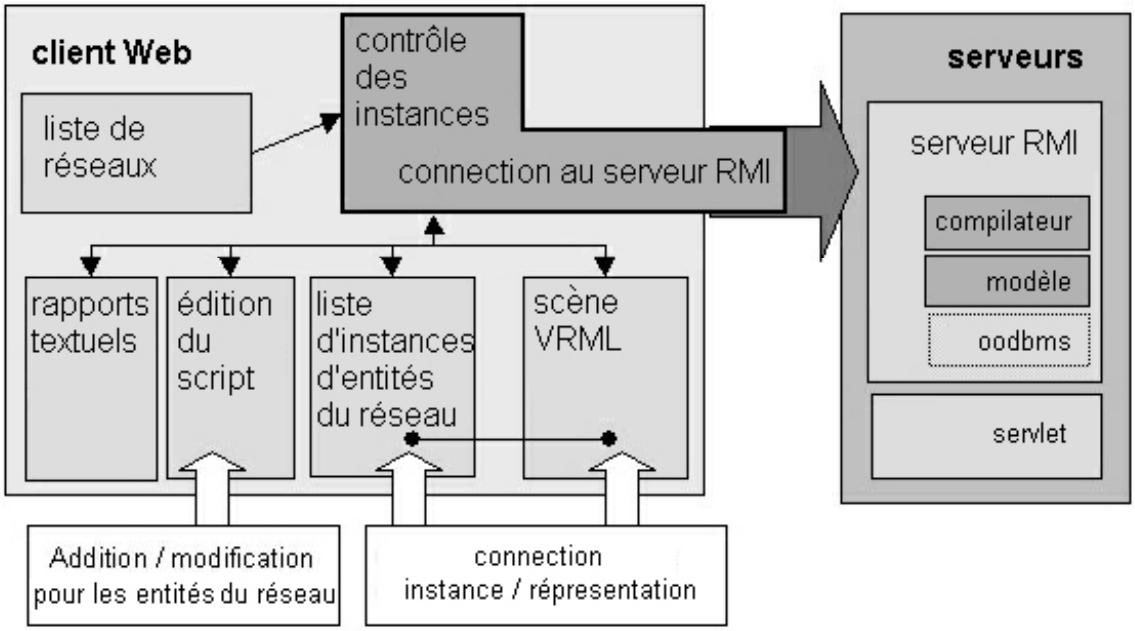



*Figure 159 Architecture Client / Serveur de l'application HUBLOT*

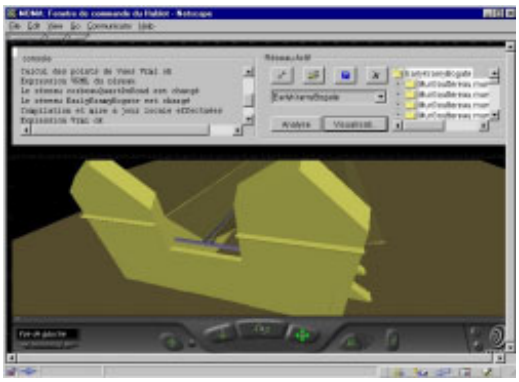


*Figure 160: Scène VRML générée dans l'application HUBLOT*

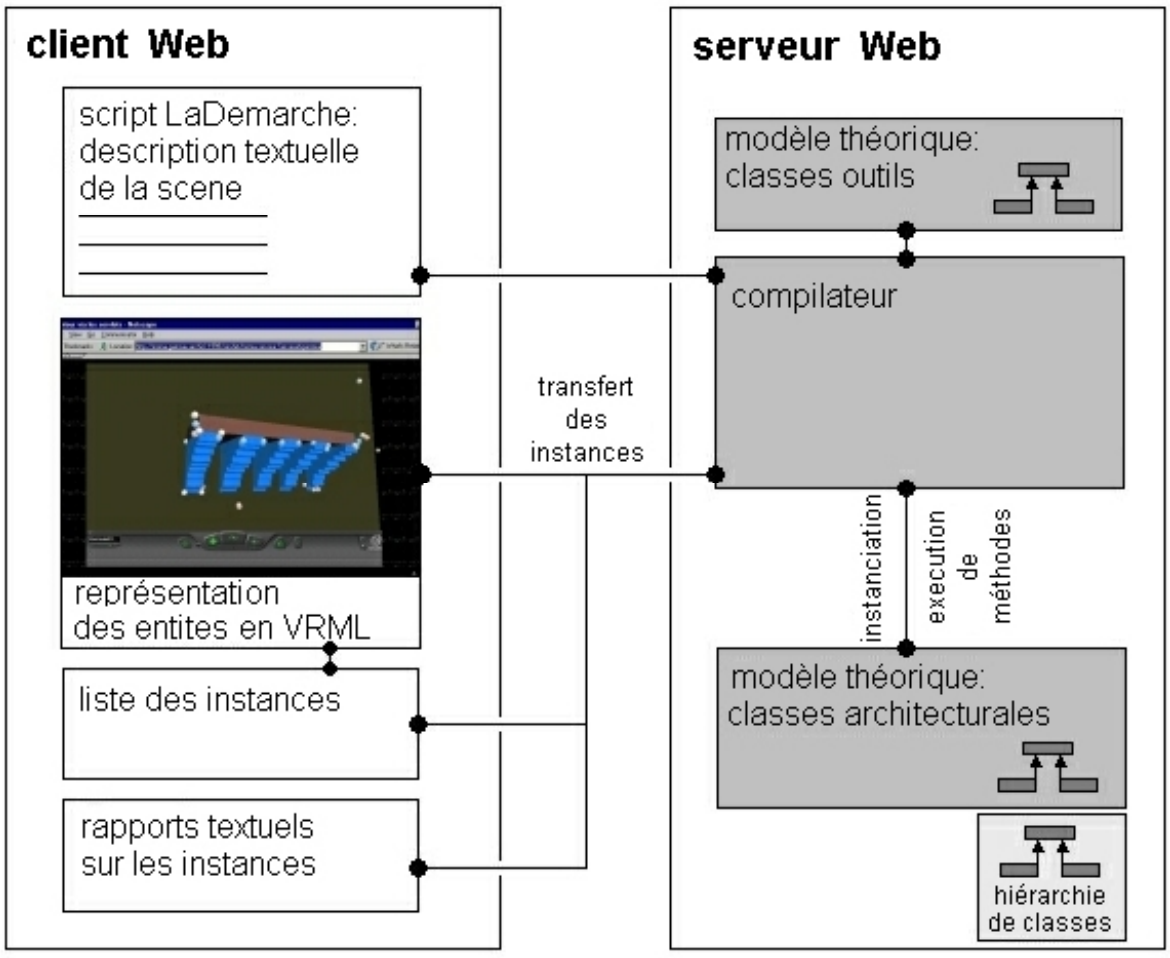



*Figure 161 : Interaction avec le modèle dans l'application HUBLOT*

## ANNEXE 48 : SOL, SCHÉMAS DE PRINCIPE DES INTERFACES

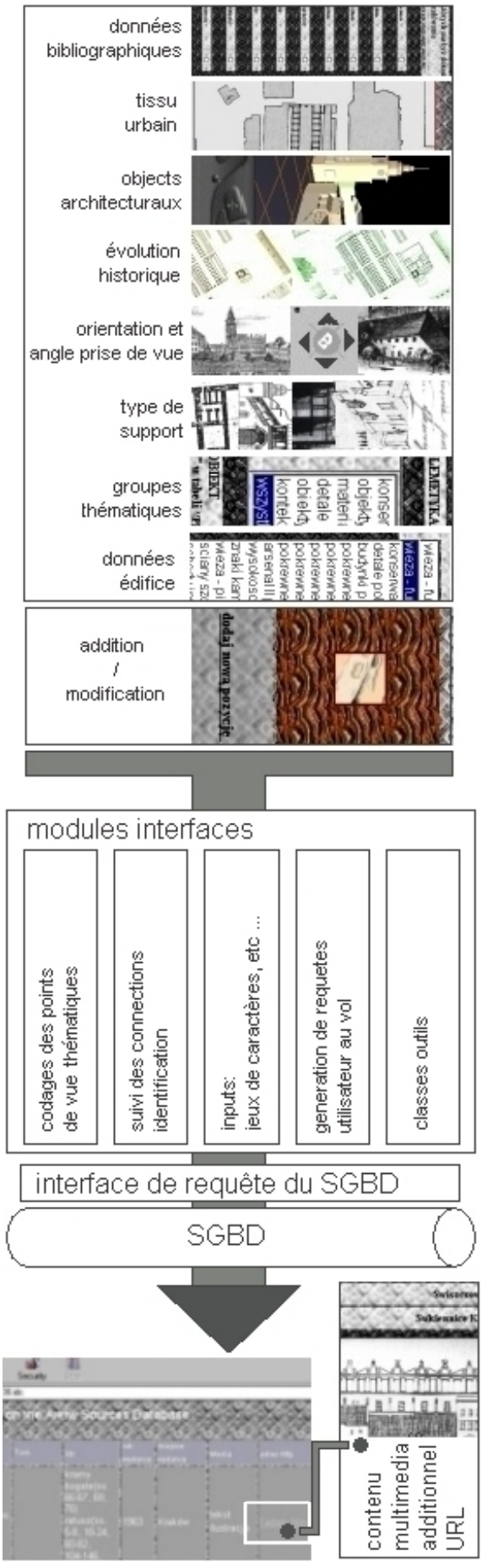



*Figure 162 : Type d'interface et traitements des requêtes dans l'outil SOL*

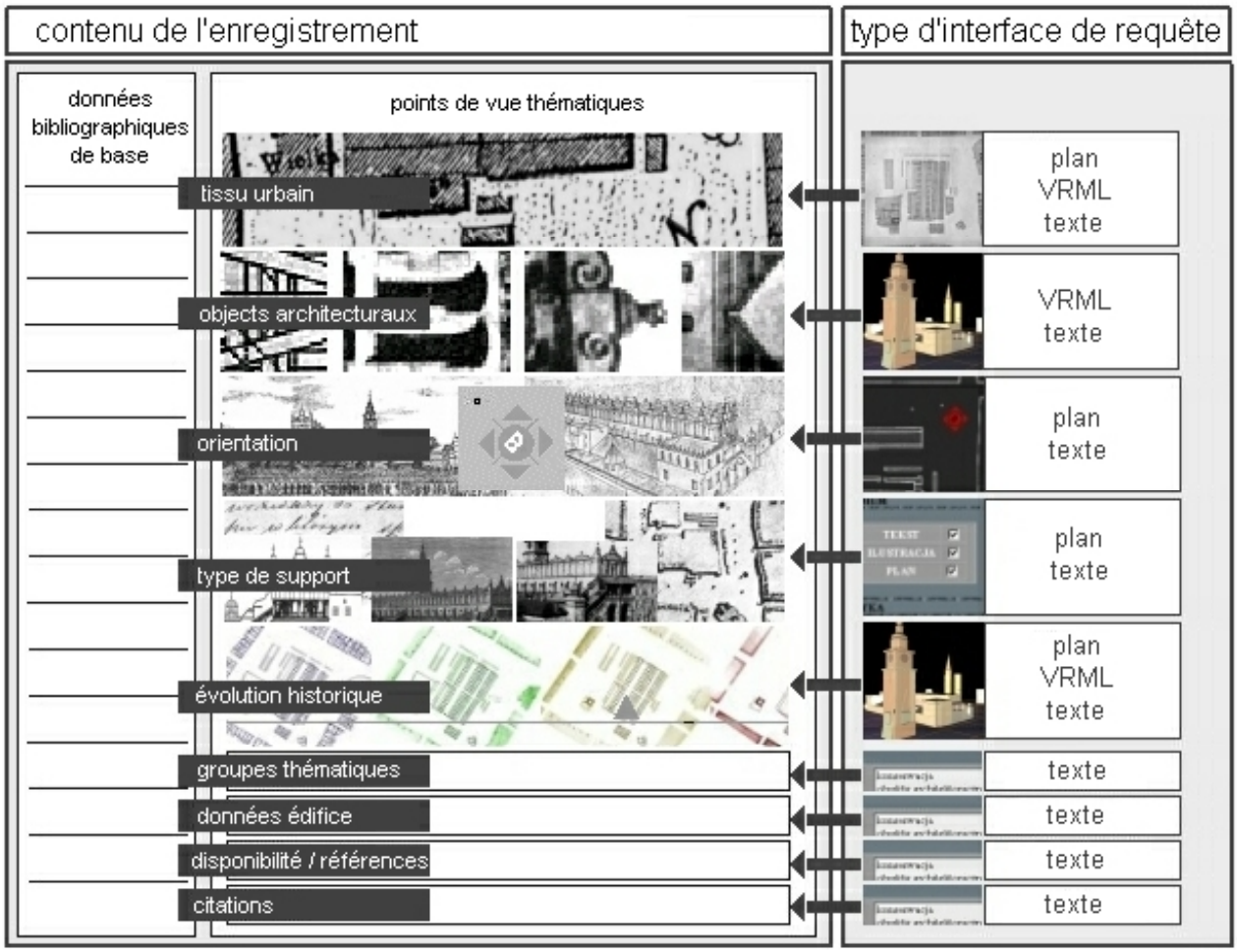



*Figure 163 : Type d'interface de requête proposée en fonction du champ concerné*

## ANNEXE 49 : HIÉRARCHIES DE CLASSES ET MODULES DÉVELOPPÉS

*TROIS CATÉGORIES REPRÉSENTANT LES CLASSES CHARGÉES DE MANIPULER LES CONCEPTS ARCHITECTURAUX, LES INTERFACES ET ET LA GÉOMÉTRIE.*

| |
|---|
| Quatre hiérarchies de concepts architecturaux :<br>ItemArchitectural (64 classes dérivées) |
| Une hiérachie de concept gérant l'interfaçage des classes architecturales<br>Interface (6 classes dérivées) |
| Une hiérarchie d'objets représentant des outils géométriques<br>ItemGéométrique (5 classes dérivées) |

*CINQ CATÉGORIES REPRÉSENTANT DES MODULES À FORTE RÉUTILISABILITÉ EN CHARGE D'EXPLOITER LES CGI.*

| |
|---|
| Une hiérarchie d'objets chargés du contrôle d'accès:<br>Connexion (3 classes dérivées) |
| Une hiérarchie d'objets chargés du codage / décodage pour le contrôle d'accès<br>Codage (3 classes dérivées, 2 classes outil) |
| Une hiérarchie d'objets chargés du formatage HTML<br>HtmElements (4 classes dérivées) |
| Une hiérarchie d'objets chargés du formatage de la gestion des entrées utilisateur<br>Input (3 classes dérivées) |
| Un ensemble d'objets chargés de l'écriture du VRML<br>VrmlOut (4 classes outil) |

..

*ANNEXE 50 : EVALUATION DES DÉVELOPPEMENTS ET PERSPECTIVES*

## Définition et évolution du modèle

Nous avons isolé quatre grandes catégories de concepts censés représenter l'édifice à étudier et les connaissances dont il relève : entités, attributs, relations et réseaux. Si la logique de classification des deux premières est assez clairement définie et implémentée, il n'en est pas de même dans le cas des deux dernières. Je m'attacherai donc ici à rappeler les directions que nous comptons prendre pour développer les concepts de relation et de réseau. Pour autant, la définition des entités et des attributs n'est pas à l'heure actuelle exempte d'ambiguïtés et pose un certain nombre de problèmes que je dois aussi évoquer. Enfin, j'aborderai la question de l'évolution de ce modèle en distinguant l'idée d'extension des hiérarchies par raffinement de concepts existants et l'idée de modification de ces concepts

### La notion de relation

Cette notion n'est implémentée aujourd'hui que sous la forme de dépendances topologiques entre une entité référente et une entité requérante. Il nous reste d'abord à expérimenter des dépendances de groupe (une entité référente, des entités requérantes). Un tel mécanisme a en fait comme objectif premier de faciliter la constitution de maquettes fixant la localisation de groupes d'instances. Mais au-delà de cet objectif pratique, la question posée ici est celle de la relation comme concept autonome à l'intérieur du modèle, disposant de propriétés permettant de qualifier les instances. En effet, une relation, même uniquement topologique, a vocation à fixer des éléments de sémantique propres. Ce n'est pas le cas aujourd'hui où la relation se limite à servir d'outil de mise en place topologique de l'entité architecturale. Deux questions distinctes sont posées :

- Représentation et manipulation de la relation dans la maquette figurant l'édifice : sous quelle forme une relation doit-elle être représentée pour autoriser sa manipulation ? Nous comptons ici expérimenter une représentation alternative des relations et des objets physiques.
- Persistance des relations : de même qu'elle doit être accessible de façon autonome, la relation doit pouvoir être documentée tant au niveau du concept que des instances. Cette documentation pourrait prendre une forme proche de celle expérimentée pour les entités.

Par ailleurs, la notion de relation est à étendre à des assemblages qui ne soient pas fondés sur des critères fixant la localisation des entités. Cette notion de relation étendue renvoie à la notion de réseau puisqu'elle devient déterminante de la logique de constitution du réseau. Prenons un exemple, et disons que nous cherchions à constituer sur un édifice un réseau basé sur une relation stylistique. Est-ce la relation qui va fixer cette logique de regroupement ou le réseau ? Nous considérons que la relation ne doit ici intervenir que comme un mécanisme de gestion de liste générique. Autant la localisation d'entités et de réseaux peut tirer bénéfice d'une classification de relations topologiques, autant, nous semble t'il, l'assemblage de groupes d'objets sur des critères qualitatifs doit-il être géré par un mécanisme de relations génériques qui ne laisse de place à une ambiguïté entre relations et réseaux.

Enfin, un point de vue introduit par J.Tajchman reste aujourd'hui à exprimer dans le travail que nous proposons : la notion d'assemblage. En effet, elle recouvre à la fois l'idée de positionnement relatif des objets, idée dont nous avons parlé ci-dessus, mais aussi celle de typologie d'assemblages. Nous comptons mener à bien une expérimentation sur le corpus des plafonds en bois dans laquelle les accessoires d'assemblage (ancrages, tenons, etc…) seront des attributs et une relation générique d'assemblage portera la qualification de l'assemblage.

Nous avons soigneusement évité d'aborder l'épineuse question de possibles relations topologiques redondantes voire incohérentes entre entités. A cela deux raisons : d'une part nous n'avons pas travaillé sur ce point et ne pouvons qu'en mentionner l'existence, et d'autre part nous ne pensons pas traiter ce point dans l'immédiat si ce n'est en terme d'interface aux outils élaborés. En effet, les points sur lesquels nous souhaitons travailler sont nombreux, nous laissons celui-là à une perspective plus lointaine. Dans l'attente, nous considérons qu'il doit être de la responsabilité du modélisateur informé par le système de choisir ses relations, et rappelons à ce sujet que c'est là exactement la façon dont l'édifice réel est bâti.

Les réseaux

Le formalisme de réseau ne nous sert aujourd'hui qu'à regrouper des éléments du modèle au sein d'un concept réduit à servir de "sac à objets" (de type Entité et/ou Relation). Il est clair que cette limitation est très lourde à bien des égards.

Commençons par évoquer les limites de ce modèle dans la représentation des objets physiques. Je l'ai dit, la manipulation de groupes d'entités peut faire référence à deux interprétations différentes du terme "divisions" : des parties d'édifices (l'aile, l'avant-corps) ou des modèles d'assemblage canoniques (l'arcade, le portique, etc..). Dans les deux cas, la question posée est simple : comment tirer parti d'une connaissance à priori sur un groupe d'entités et de relations lors de l'instanciation d'un nouveau réseau ? Comment instancier une arcade sans devoir préciser le type de concepts qu'une arcade contient ? Dans ce cas, le mécanisme actuel n'est après tout pas si mauvais : pour peu qu'une séquence d'instanciation adéquate ait été décrite, une arcade peut être instanciée à partir d'une classe réseau générique capable d'instancier n'importe quelle liste d'entités et de relations qualifiée.

La première étape que nous souhaitons franchir sera donc d'élaborer un mécanisme d'instanciation des réseaux à partir de séquences d'instanciation canoniques. Une seule classe Réseau dite générique sera dès lors concernée, évitant de raffiner inutilement ce concept en une hiérarchie de classes dont la seule justification aurait été un état initial différent.

Cette idée peut s'appliquer de la même manière dans le cas de réseaux basés sur par exemple une concordance stylistique ou sur un modèle d'agencement d'objets canonique comme une colonnade. De la même façon, l'idée de norme ou de rythme peut être gérée de façon générique, le réseau se contentant de donner accès aux objets positionnés (des entités) et à la règle fixant leurs positions (une relation).

Une distinction peut être établie entre réseaux d'agencements, ou assemblages prédéfinies d'entités et de relations, et réseaux de concordance visant à établir des comparaisons entre propriétés qualitatives. Pourtant cette distinction nous semble a priori nécessaire essentiellement pour faciliter la lisibilité du modèle.

Par contre, le mécanisme de réseau actuel est bien en peine de représenter la notion d'espace clos. Autrement dit, la qualification d'usage d'un réseau (ce qu'est l'espace délimité) plutôt que sa qualification de bornage (comment cet espace est délimité) n'est pas du tout à notre portée. En effet, seuls les objets physiques sont manipulables dans notre modèle. Nous n'avons pas l'intention de remettre en cause ce choix de modélisation dans l'immédiat.

Entités et attributs : problèmes de modélisation

Les hiérarchies des entités et des attributs sont celle sur lesquelles nous avons porté l'essentiel de notre effort. Plusieurs points restent pourtant à prendre en compte. Nous les présentons un par un sous la forme d'une liste commentée :

- L'incomplétude d'une entité n'est pas gérée en terme de représentation. Si les dimensions (attributs morphologiques) ou d'autres propriétés d'une entité peuvent ne pas être renseignées, nous n'avons pas encore étudié le moyen de rendre compte de cette incomplétude dans la visualisation de l'entité. La solution envisagée passe par un affichage autonome de chaque attribut constituant l'entité.
- Le mécanisme de valeurs par défaut mis en place dans l'implémentation JAVA n'a pas été porté à l'identique dans l'implémentation PERL. Dans l'entité les attributs observés et les attributs renseignées par un mécanisme de valeurs par défaut ne diffèrent pas. La définition de l'attribut devra ici être complétée.
- L'attribut n'est manipulable que par l'intermédiaire de l'entité qui le contient. Nous n'entendons pas remettre en cause ce choix de modélisation mais pensons expérimenter un outil de documentation des attributs qui ne donne pas accès à l'instanciation autonome d'attributs mais au renseignement d'attributs existants.
- Un attribut devrait en théorie porter des informations qualitatives, une inscription stylistique par exemple, et des informations quantitatives. Dans l'état actuel de notre travail, seul le deuxième groupe d'informations a été traité. Les attributs dits qualifieurs restent donc à implémenter. Il n'y a pas là de difficultés particulières en terme de modélisation mais l'étude de ces concepts réclame du temps et de l'attention.
- Les éléments de statuaire ou du décor peint ne sont pas traités. La même remarque qu'au point précédent s'impose. De plus, le rapport entre un décor peint et son support est loin d'être simple à gérer dans notre modèle puisque ce décor peut être partagé par plusieurs entités ou attributs.

- Les modes de construction sont représentés comme propriété des entités et ne figurent pas comme concept à part entière dans notre modèle. Ce point mérite d'être abordé là encore en étudiant de façon approfondie ce domaine particulier.

Mise à jour et évolution du modèle

Nous avons avec le CLASSEUR expérimenté un outil dédié au raffinement du modèle par des non-spécialistes. Je ne reviens pas sur les conclusions de cette expérience, déjà présentées précédemment. Il me semble pourtant important de noter que notre travail peut s'orienter dans deux directions :

- La mise à jour du modèle par la modification des concepts existants.
- L'évolution de la hiérarchie de classes par ajout de nouveaux concepts.

Il doit être clair que rien d'abouti n'a été élaboré jusqu'à présent. Nous souhaitons placer comme l'une des perspectives de travail qu'ouvre notre projet la détermination d'un formalisme d'évolution du modèle s'appuyant sur les travaux que nous avons cités dans la partie Etat de l'art, notamment [Castellani, 1998] et [Sunyé et al, 1997].

Exploitation du modèle : la représentation

La visualisation de scènes figurant un ensemble d'instances du modèle correspond à deux grandes familles d'objectifs, la simulation d'hypothèses de restitution et la navigation par la représentation tridimensionnelle de l'édifice dans une base documentaire. Nous avons insisté sur l'importance de ce type de résultats dans le domaine d'application qui est le nôtre, celui de l'architecture patrimoniale, nous ne revenons donc pas sur ce point.

Nous entamons ou entamerons un ensemble de développements visant à diversifier le type de maquettes produites et à améliorer leur pertinence au regard des attentes des conservateurs du patrimoine bâti. Je vais donc résumer les directions que prennent ou prendront les travaux sur la représentation du modèle en distinguant ces deux aspects. Dans un premier temps, je rendrai compte des progrès que nous devons faire pour assurer une plus grande variété de formats et d'usages des maquettes produites et pour tirer un meilleur parti de la plateforme VRML utilisée aujourd'hui.

Dans un second temps, j'évoquerai de façon plus prospective les grandes questions que pose l'utilisation de ces maquettes dans le cadre particulier d'études sur l'édifice patrimonial.

Diversification des formats et usages des maquettes

Nous avons indiqué les raisons qui nous ont poussé à choisir VRML comme format permettant la visualisation de scènes bâties à partir de notre modèle. Nous aovns également présenté l'outil nous permettant de créer ces scènes au travers du réseau Internet, VALIDEUR. J'ai insisté sur la contrainte que nous nous fixons de délivrer des scènes autonomes pouvant être réutilisées en dehors du contexte de l'application ayant servi à les créer. Enfin, nous avons décrit la méthode utilisée pour écrire chaque entité à représenter au format VRML, méthode permettant d'intégrer des déclencheurs d'évènements à router en fonction du besoin utilisateur. Plusieurs points restent cependant à traiter, points que je livre sous la forme d'une courte liste :

- Si chaque entité possède déjà les mécanismes de déclenchement d'événements, l'outil de création de scènes ne donne pas accès au routage de ces évènements.
- Le basculement (par node Switch) entre représentations alternatives (notion de niveau de détail) ou entre évènements alternatifs n'est pas implémenté dans ce même outil.
- La visualisation de l'entité utilise un seul descriptif de texture, ne permettant pas (sauf dans le cas des plafonds) de visualiser ses attributs de façon différente.
- L'insertion de nodes gérant des caméras stéréos n'est ni systématique ni liée à l'ensemble des points de vue par défaut gérés par le système.
- Ces points de vue par défaut sont d'une grande pauvreté : seules les vues géométrales et quatre axonométries sont calculées. Une définition interactive de points du vue utilisateurs devrait être autorisée rapidement.
- La classe vrmlOut ne rend pas compte d'éléments de la syntaxe VRML intéressants, comme par exemple la définition de textures plaquées (images fixes ou animées), le suivi des déplacements de l'observateur dans le cas des réseaux, etc...

- L'affichage de l'entité utilise un mécanisme de valeurs par défaut qui ne fait pas référence à la connaissance que l'utilisateur peut avoir des propriétés qualitatives de l'entité (matériau, opus, etc..).
- Enfin, l'éclairement de la scène par choix utilisateur dans l'outil de création de scènes n'est pas complètement développé.

Il nous reste donc beaucoup à faire pour améliorer la création de scènes VRML, et ce à la fois dans la définition des méthodes adéquates des concepts architecturaux et dans l'interface de l'outil de création de scènes.

Par ailleurs, nous avons entamé le (re)développement de méthodes de représentation basées sur le format POV pour lequel beaucoup de questions identiques se posent (points de vue et caméras stéréos, textures, visualisation des attributs, etc...).
Nous pensons qu'avec l'expérience aujourd'hui acquise sur la plateforme VRML nous pourrons grandement améliorer l'expressivité de nos visualisations. En effet, la demande que notre domaine d'application exprime n'est pas un plus grand réalisme des images mais clairement une meilleure prise en compte de la sémantique du domaine, et en particulier la définition d'un formalisme de représentation des édifices permettant de les visualiser aux différentes échelles dans une scène.

Codification de l'image et outil de génération de scènes
En parlant de prise en compte de la sémantique du domaine, nous avons déjà résumé ce qui va être dit ici. En effet, la plus grande difficulté à laquelle nous sommes confrontés en terme de représentation réside dans la nécessité de prendre en compte des notions telles que l'incomplétude d'un objet, l'incertitude dans sa définition géométrique, notions qui vont à l'encontre de ce que la maquette numérique sait bien faire : représenter de façon exhaustive. Je dois donc ici revenir sur l'idée de codification de la représentation déjà évoquée pus haut. L'image doit s'adapter au problème représenté et non l'inverse : si la connaissance que l'on a d'un élément est incomplète, alors il faut pouvoir le signaler dans l'image elle-même. Au-delà de l'outil de visualisation qu'elle est par essence, la maquette peut être conçue comme un outil de compréhension de l'édifice.
Cette idée toute simple, présente dans la représentation dessinée traditionnelle, fait l'objet de peu de travaux dans le domaine de la simulation d'hypothèses de restitution en imagerie de synthèse. Elle est pourtant au cœur des préoccupations de nos interlocuteurs. Nous avons donc entamé une réflexion sur ce sujet afin de recenser les moyens dont nous disposons concrètement. Nous pensons qu'il faut distinguer deux cas :

- L'élément est complètement défini géométriquement mais sa présence sur l'édifice reste incertaine. Il s'agit par exemple typiquement de restitutions d'édifices disparus utilisant un élément dont rien ne permet d'affirmer ou d'infirmer la présence. Nous pensons que dans ce cas la texturation de l'élément peut être utilisée pour autant qu'à l'image de la carte géographique une légende vienne l'expliciter.
- L'élément est observable mais partiellement détruit. Il s'agit par exemple de la baie illustrée à la section 3.2.1.i.1 dont ne subsiste qu'une partie du linteau et un piédroit. Nous pensons que dans ce cas il sera nécessaire d'une part d'autoriser une texturation autonome des attributs des entités et d'autre part de renvoyer l'attribut à une URL spécifique. Bien sûr, il est possible de jouer sur une texture plaquée reprenant une photographie de l'objet incomplet. Cela nous semble, compte tenu du travail demandé en rapport au résultat attendu, un vœu pieu.

Ces pistes de réflexion sur une possible codification de la représentation renvoient à plusieurs points qu'il nous faudra aborder rapidement, et dont je donne en conclusion une courte liste :

- La gestion des niveaux de détail intra -entités (représentations alternatives de chaque entité) et intra-réseaux (représentation alternative du réseau ou de ses entités).
- L'expression autonome des attributs.
- Le lien de l'image aux sources documentaires.
- L'accès utilisateur aux évènements alternatifs.

Disons enfin que dans le domaine d'application qui est le nôtre, nous croyons que la maquette numérique, utilisée pour générer des représentations d'hypothèses de restitution d'édifices ou de tissus urbains, a

vocation à être interprétative. Nous chercherons en définitive à apporter à la représentation tridimensionnelle en imagerie de synthèse cette évidence du dessin, expression interprétative de l'édifice, qui en fait l'outil privilégié de raisonnement de l'architecte et du conservateur.

Documentation : modèle et maquettes

Dans notre domaine d'application, un modèle et des maquettes en figurant des instances n'a de sens qu'encadré et justifié par un ensemble de sources documentaires. Ceci n'est à vrai dire pas particulier à ce domaine d'application mais revêt pour le conservateur du patrimoine une importance particulière.
Nous devrions en fait dire que le manque de justification documentaire est pour une grande part à l'origine du désintérêt à peine voilé qu'expriment beaucoup de spécialistes de l'architecture patrimoniale envers les maquettes numériques en général. Il nous importait donc tout particulièrement d'insister sur ce point.
Je l'ai dit, nous distinguons ici les données relatives à la définition du modèle et celles relatives au renseignement des instances puisque les sources documentaires interviennent à ces deux niveaux. Nous aovns indiqué les choix que nous avons fait pour instrumenter un mécanisme de renseignement documentaire des concepts et des instances : un formalisme de lien URL basique ne remettant pas en cause le principe de portabilité maximale que nous voulons respecter.
Plusieurs manques apparaissent dans l'état actuel de notre travail :

- Seules les entités architecturales sont concernées par le mécanisme de documentation expérimenté. Les attributs, manipulables uniquement au travers de leurs entités propriétaires, ne sont présents dans les interfaces développés que sous la forme d'instances. Les réseaux ne bénéficient d'aucun mécanisme de ce type, pas plus que les relations.
- Si données et traitements sont effectivement séparés dans le mécanisme de valeurs par défaut expérimenté, cette séparation n'est pas totale puisque l'entité se charge elle-même de décoder le fichier ASCII la concernant. Nous confions désormais cette tâche à des objets autonomes.
- Les interfaces tridimensionnelles n'intègrent pas la notion d'échelle variable du modèle sur laquelle peu de travaux de recherche ont été menés dans le champ patrimonial. Il s'agit ici d'attacher au même jeu d'objets des données différentes selon qu'il est interrogé à l'échelle urbaine ou à l'échelle architecturale par exemple. Ce point est donc à rapprocher des questions de représentation évoquées à la section précédente. Il nous faudra approfondir notre démarche de modélisation du corpus architectural pour prendre en compte des niveaux de définition alternatifs (degrés d'incertitude ou de complétude, échelles urbaines ou architecturales, etc..).

L'outil DIVA permettant de renseigner les concepts doit être mis à niveau dans un proche avenir. Il s'agira d'abord de faciliter l'usage de l'outil en autorisant des requêtes à partir de l'image. En effet, il est courant, comme le montre clairement l'ouvrage de Jean-Marie Pérouse de Montclos [Pérouse De Montclos, 1988], que l'utilisateur connaisse la morphologie de l'objet sans pouvoir lui affecter un vocable non ambigu.
Nous comptons donc autoriser la formulation de requêtes sur la base DIVA à partir soit d'images photographiques soit de maquettes numériques. Par ailleurs, nous pensons qu'il sera possible de référencer des instances de ces concepts sous la forme d'un catalogue illustré répertoriant diverses interprétations régionales du terme.
Une structuration XML de ces références comme des autres URL additionnelles sera également à expérimenter.
Des points plus anecdotiques peuvent également être rapportés : suivi des connections au niveau des applications, séquences d'instanciations, autonomie des rapports des attributs, etc...

Si la documentation du modèle et de ses instances ne justifient pas l'identification et l'organisation du modèle, elle justifie en revanche en grande partie son usage. Il nous semble par conséquent que l'utilisation de ce modèle dans le cadre d'outils comme DIVA et SOL permet d'en mieux évaluer la pertinence. Nous pensons avoir dans ce cadre démontré qu'un outil de référencement bibliographique après tout très standard pouvait grandement bénéficier d'un effort de modélisation des connaissances relatives au domaine d'application. C'est parce qu'il y a modélisation du corpus architectural que la référence bibliographique peut être attachée à un édifice de façon évidente.

Nous pensons avoir là apporté une réponse adéquate aux préoccupations de nos interlocuteurs, spécialistes de l'architecture patrimoniale, réponse s'appuyant sur l'approche objet à la fois comme moyen d'organiser des éléments de connaissance et comme paradigme de programmation.

## *LISTE DES FIGURES*

## Figures en annexe: