Approche Orientée Services pour la Réutilisation de Processus et d'Outils de Modélisation

Jorge Luis Perez Medina

UNIVERSITE DE GRENOBLE

N° attribué par la bibliothèque
|__|__|__|__|__|__|__|__|__|__|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE GRENOBLE

Spécialité : « Informatique »

préparée au laboratoire d'Informatique de Grenoble

dans le cadre de **l'Ecole Doctorale « Mathématiques, Sciences et technologies de l'information, Informatique »**

présentée et soutenue publiquement

par

Jorge Luis PEREZ MEDINA

le 15 juin 2010

APPROCHE ORIENTÉE SERVICES POUR LA RÉUTILISATION DE PROCESSUS ET D'OUTILS DE MODELISATION

Sous la direction de Dominique RIEU
et Sophie DUPUY-CHESSA

JURY

PR. Corine CAUVET, Professeur Rapporteur
DR. Sophie DUPUY-CHESSA, Co-encadrant de thèse
PR. Christophe KOLSKI, Examinateur
PR. Yves LEDRU, Président
MDC HDR. Isabelle MIRBEL, Rapporteur
PR. Dominique RIEU, Directeur de thèse

À mes parents : Ninoska et Pablo
À mes frères : Luis Alberto et Pablo Rafael
À ma sœur : Beatriz Azorena
À ma nièce : Ana Beatriz
À mon adorable épouse : Kristel
À ma petite : Paola Valentina
À ma belle famille : André et Ginette

C'est avec beaucoup d'émotions que je vous remercie pour le soutien et la confiance que vous m'avez toujours accordés. Il m'est difficile de traduire par les mots l'affection et la gratitude que je vous réserve. Aussi, est-ce à vous que je dédie ces heures de travail, de joies et de difficultés que vous avez partagées. J'espère que vous vous réjouissez de la réussite qui couronne mes efforts. Merci !

Remerciements

Je tiens à exprimer mon immense reconnaissance à mes directrices de thèse, Mesdames Dominique Rieu et Sophie Dupuy-Chessa. Elles m'ont guidé et soutenu durant ces quatre années vécues en France. L'intérêt constant qu'elles ont pris pour ce travail, m'a permis de finaliser cette expérience dans le monde de la recherche.

Je tiens à remercier les membres du jury :

- Monsieur Yves Ledru, Professeur à l'université Joseph Fourier, qui m'a fait l'honneur de présider ce jury.
- Madame Corine Cauvet, Professeur à l'université Paul Cézanne - Aix-Marseille III, de m'avoir fait l'honneur de rapporter ma thèse.
- Madame Isabelle Mirbel, Maître de Conférences (HDR) à l'université de Nice Sophia Antipolis, de m'avoir fait l'honneur de rapporter ma thèse.
- Monsieur Christophe Kolski, Professeur à l'université de Valenciennes et du Hainaut-Cambrésis, d'avoir accepté d'être examinateur de ma thèse, et pour son intérêt sur mon travail.

Je tiens à remercier les membres de l'équipe SIGMA : Christine, Jean-Pierre, Agnès, Claudia, Monique, Amsem, Charlotte, Ali, Loic, Saidi, Marco, Alexandra, c'est vraiment agréable de travailler en équipe.

Je tiens également à remercier Luz-Maria qui m'a consacré du temps pour réaliser mon dossier de soutenance et les démarches requises. Luz-Maria c'est un plaisir de travailler dans le même bureau que toi.

Je remercie aussi les membres de l'équipe IIHM et les anciens membres avec ce qui j'ai partagé de nombreux moments : Joëlle, Gaëlle, François, Laurence, Renaud, Michael, David, Alex, Lionel, Céline, Emeric, Eric, Yoann, Adriano.

Un remerciement particulier est adressé à mes amis : David H., Lina, Elodie, Amanda, Charly, David J., Jean-Sébastien, Nicolas, Guillaume, Mathieu, Bill, Marco, Rami et Ali.

Enfin, je désire remercier tout particulièrement des amis, grâce auxquels j'ai pu puiser toute la force dont j'avais besoin pour arriver au terme de ce travail.

Approche Orientée Services pour la Réutilisation de Processus et d'Outils de Modélisation

Table des matières

TABLE DE FIGURES

LISTE DE TABLEAUX

LISTE DES ABRÉVIATIONS

BPEL : Business Process Modeling Language
CORBA : Common Object Request Broker Architecture
CREWS : Cooperative Requirements Engineering With Scenarios
CWM : Common Warehoure Metamodel
DTD : Document Type Definition
EBJ : Entreprise JavaBeans
IHM : Interaction Homme-Machine
IM : Ingénierie de méthodes
J2EE : Java 2 Entreprise Edition
MOF : Meta-Object Facilities
OCL : Object Constraint Language
OMG : Object Management Group
OMT : Object Modeling Technique
RUP : Rational Unified Process
SI : Système d'information
SOA : Service-Oriented Architecture
SOAP : Simple Object Access Protocol
SO2M : Service-Oriented Meta-Method
SO2MX: Service-Oriented Meta-Method eXecution
2TUP : Two Track Unified Process
UML : Unified Modelling Language
XMI : XML Metadata Interchange
XML : Unified Modeling Language

CHAPITRE 1

Introduction générale

Ce chapitre présente en premier lieu le contexte général de cette thèse (§1) ainsi que la problématique abordée (§2). Les principes de la solution sont abordés (§3). Les principales contributions réalisées sont ensuite introduites (§4) avant de préciser le plan de la thèse (§5).

CHAPITRE 1

1. LE CONTEXTE GENERAL

L'ingénierie des modèles est une branche du génie logiciel où tout artefact logiciel est considéré comme un modèle. Dans ce contexte, un modèle est une spécification souvent partielle des fonctionnalités, de la structure et/ou du comportement d'un système ou d'une application [OMG-MDA, 2003]. Ainsi différents modèles sont utilisés au cours d'un processus de développement logiciel. Dans ces processus, les concepteurs de modèles jouent différents rôles et s'appuient sur des outils tels que les Ateliers de Génie Logiciel (AGL), et les outils d'Ingénierie Dirigée par les Modèles (IDM), pour résoudre leurs besoins en termes de gestion de modèles.

La gestion de modèles regroupe l'ensemble des fonctionnalités permettant de représenter, créer, stocker et manipuler les modèles, avec pour but d'expliquer et de représenter les systèmes sous différents aspects (statique, dynamique, fonctionnel, interactionnel) et à différents niveaux d'abstraction (expression de besoins, analyse, conception,…). Dans le domaine du Génie Logiciel, l'Object Management Group a proposé l'approche dirigée par les modèles, MDA (Model Driven Architecture), dont les concepts sont orientés-modèles plutôt que orientés-objets.

L'approche MDA [OMG-MDA, 2003] offre le pouvoir d'abstraction, de raffinement et des vues multiples sur les modèles. Surtout, elle donne la possibilité de concevoir des modèles indépendants des plateformes et de l'environnement d'implantation. Le MDA a été étendu à l'IDM [Action IDM, 2007] qui ne se limite pas aux modèles UML et met les modèles, et non pas les programmes, au centre de la démarche en Génie logiciel. L'IDM est donc utilisé pour décrire l'ensemble des concepts et technologies autour de la gestion de modèles. La plupart des outils IDM sont utilisés pour automatiser les transformations de modèles qui permettent la transformation d'un modèle vers un autre modèle, la génération de code, la validation, le test, la traçabilité.

Un AGL est un ensemble cohérent d'outils informatiques formant des environnements d'aide à la conception, au développement et à la mise au point des logiciels. Ils sont dotés de fonctionnalités plus ou moins avancées (par exemple : l'édition de modèles, la génération de code, la méta-modélisation, …) adaptées aux différentes phases de la production d'un logiciel. Actuellement les besoins des concepteurs en termes de gestion de processus et produits sont divers et les outils de modélisation ne sont pas complets car les besoins et les usages autour des modèles ne sont pas consensuels.

2. LA PROBLEMATIQUE

Dans le processus de développement du logiciel, nous partons des constats suivants au sujet des concepteurs et des outils. Les concepteurs de modèles ont des besoins en terme de gestion de processus et de produits divers. Ils jouent différents rôles et travaillent en collaboration avec des équipes dans lesquelles participent des personnes de différents domaines avec des compétences diverses.

Les outils actuels (donc les outils IDM et les AGLs) ne permettent pas la gestion coordonnée et coopérative de modèles requise par plusieurs spécialistes qui travaillent ensemble dans un contexte de conception collaborative proposé dans [Juras et al. 2006]. Ces concepteurs s'appuient sur des outils de modélisation hétérogènes incomplets, limités en fonctionnalités et souvent réduits à supporter quelques méta-modèles et modèles.

De nombreux travaux sont intéressants pour résoudre le problème d'intégration d'outils [Wick, 2006]. Si les outils proposés sont intéressants pour résoudre le problème technique d'intégration, ils ne répondent pas aux questions du choix des outils de modélisation en fonction des acteurs et des processus de conception. La question que nous cherchons donc à résoudre est comment construire un environnement de gestion de modèles qui répondent aux besoins organisationnels et techniques des entreprises.

3. LES PRINCIPES DE LA SOLUTION

Nous pensons que l'approche orientée services peut répondre à notre problème. En effet, la conception d'une plateforme orientée services pour la réutilisation des processus et des outils de modélisation peut conduire à faciliter la construction d'environnements de modélisation adaptés aux besoins spécifiques des concepteurs de modèles. Nous proposons donc une plateforme permettant de trouver, de coupler et d'utiliser des briques fonctionnelles répondant aux besoins spécifiques de chaque concepteur.

Au niveau conceptuel, nous considérons un environnement de modélisation comme un ensemble d' outils décrits sous la forme de services. Afin de faciliter la construction des environnements de modélisation, notre approche définit trois niveaux de services. Le niveau intentionnel de services correspond à la modélisation des buts stratégiques des concepteurs de modèles, qui peuvent être mis en œuvre par des démarches de conception décrites au niveau organisationnel. Le niveau organisationnel garantit la réutilisation des services opérationnels d'une manière coordonnée, mais également, il garantit la création et la gestion des fragments de méthodes. Notre but ici est uniquement de fournir à chaque concepteur de modèle l'environnement logiciel lui permettant de

mener à bien ses activités. Le niveau opérationnel de service fournit un support pour la création, et la manipulation des modèles.

Chaque niveau de service est structuré selon un méta-modèle. La description sémantique des services dans les trois niveaux d'abstraction est réalisée au moyen d'ontologies.

L'approche proposée est basée sur trois principes : la réutilisation, l'utilisation de services, l'utilisation d'ontologies.

3.1. La réutilisation

Notre approche est basée sur la réutilisation de processus existants, de modèles et d'outils pour trouver une solution aux besoins de concepteurs de modèles et des chefs de projets. Nous utilisons la notion de services [Bieber *et al.*, 2001] pour la gestion de modèles dans le but de créer des environnements de modélisation adaptés aux besoins des différents acteurs de projets. L'environnement de gestion de modèles proposé repose sur trois niveaux d'abstraction : opérationnel, organisationnel et intentionnel. Les trois niveaux offrent des services de gestion de modèles de natures différentes. 1) les services opérationnels réalisent des opérations automatisées (par exemple : l'édition d'un modèle ou la transformation de modèle); 2) les services organisationnels proposent les fragments de méthode de conception, dédiées à l'ingénierie des systèmes à base de modèles. Le choix d'un processus de conception détermine les modèles qui vont être utilisés, et donc, leurs environnements de modélisation; 3) les services intentionnels correspondent à la modélisation des buts visés par toute personne ou toute organisation manipulant des modèles [Rolland *et al.*, 2008]. Le niveau de service intentionnel permet de justifier l'existence des buts organisationnels qui répondent aux besoins des concepteurs de modèles.

3.2. L'utilisation de services

Dans le cadre de cette thèse, l'approche orientée services nous paraît être une solution à nos préoccupations de réutilisation, mais elle offre de plus la possibilité de sélectionner, voire de découvrir, les services répondant à certaines caractéristiques, puis de les composer pour offrir un service plus complet. Nous adoptons la définition du service proposée par Papazoglou [Papazoglou *et al.*, 2005]. Un service est défini comme « un ensemble d'applications modulaires auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées. Un service peut effectuer des actions allant de simples requêtes à des processus métiers complexes. Les services permettent d'intégrer des systèmes d'information hétérogènes en utilisant des protocoles et des formats de

données standardisés, autorisant ainsi un faible couplage et une grande souplesse vis-à-vis des choix technologiques effectués ».

L'approche à base de services repose sur l'idée qu'une application peut être réalisée par composition des services logiciels mis à disposition par des fournisseurs divers. Cette composition peut intégrer des services hétérogènes implémentés avec des technologies différentes, voire même des logiciels propriétaires.

Dans cette proposition, la notion de service est utilisée pour, (1) désigner un but stratégique de modélisation requis par des concepteurs de modèles, (2) faire référence à des fragments de méthode disponibles permettant de répondre à un objectif de modélisation donné et (3) désigner un environnement intégrant des outils de support aux activités de modélisation.

3.3. L'utilisation d'ontologies

Les ontologies, qui définissent une terminologie pour partager un vocabulaire commun au sein d'un domaine, sont aujourd'hui largement utilisées pour répondre à ce besoin. Il est essentiel de disposer d'une sémantique riche des services afin de faciliter la communication et la compréhension des intervenants qui partagent ses activités dans un domaine particulier.

Dans nos propositions, nous avons besoin de représenter les connaissances relatives aux domaines des systèmes d'information pour établir un vocabulaire commun entre les concepteurs de modèles. Nous avons intégré à chaque niveau de services les ontologies du domaine des systèmes d'information issues de Guzélian [Guzélian, 2007].

Nous avons choisi d'intégrer les ontologies dans nos méta-modèles de manière à offrir une vision intégrée de chaque niveau d'abstraction. La modélisation de ces ontologies s'appuie sur le concept de catégorisations. Les ontologies sont intégrées sous la forme d'imitation du patron « Concept – Term », une adaptation du patron Item-Description.

En nous basant sur ces trois principes, ce travail de recherche a abouti à une approche orientée services pour la réutilisation de processus et d'outils de modélisation basée sur trois niveaux d'abstraction offrant des réponses à la création, la recherche et le stockage des buts, processus et outils de modélisation sous la forme de services.

4. LE PLAN DE LA THESE

Ce rapport de thèse est organisé en cinq chapitres (cf. Figure 1).

Le second chapitre est dédié à un état de l'art sur l'ingénierie de modèles, l'ingénierie de besoins, l'ingénierie de méthodes et les processus dans les approches à base de services. Nous définissons des critères de classification des approches, qui permettent par la suite de décrire chaque approche étudiée selon ces critères. Une comparaison nous permet de positionner notre approche par rapport aux travaux existants.

Nous proposons dans le troisième chapitre une approche orientée services pour la réutilisation de processus et d'outils de modélisation basée sur trois niveaux d'abstraction. Nous commençons pour détailler l'utilisation des ontologies dans le domaine des systèmes d'information, puis nous présentons deux cas d'étude qu'illustrent nos trois niveaux de services.

Le quatrième chapitre propose une plateforme de support pour notre approche orientée service. Une vision globale d'utilisation de la plateforme est aussi présentée.

Le cinquième chapitre clôture ce document en rappelant les propositions de cette thèse, et en donnant des perspectives possibles sur un plan d'approfondissement des travaux réalisés ainsi que sur un plan d'élargissement du domaine de la recherche.

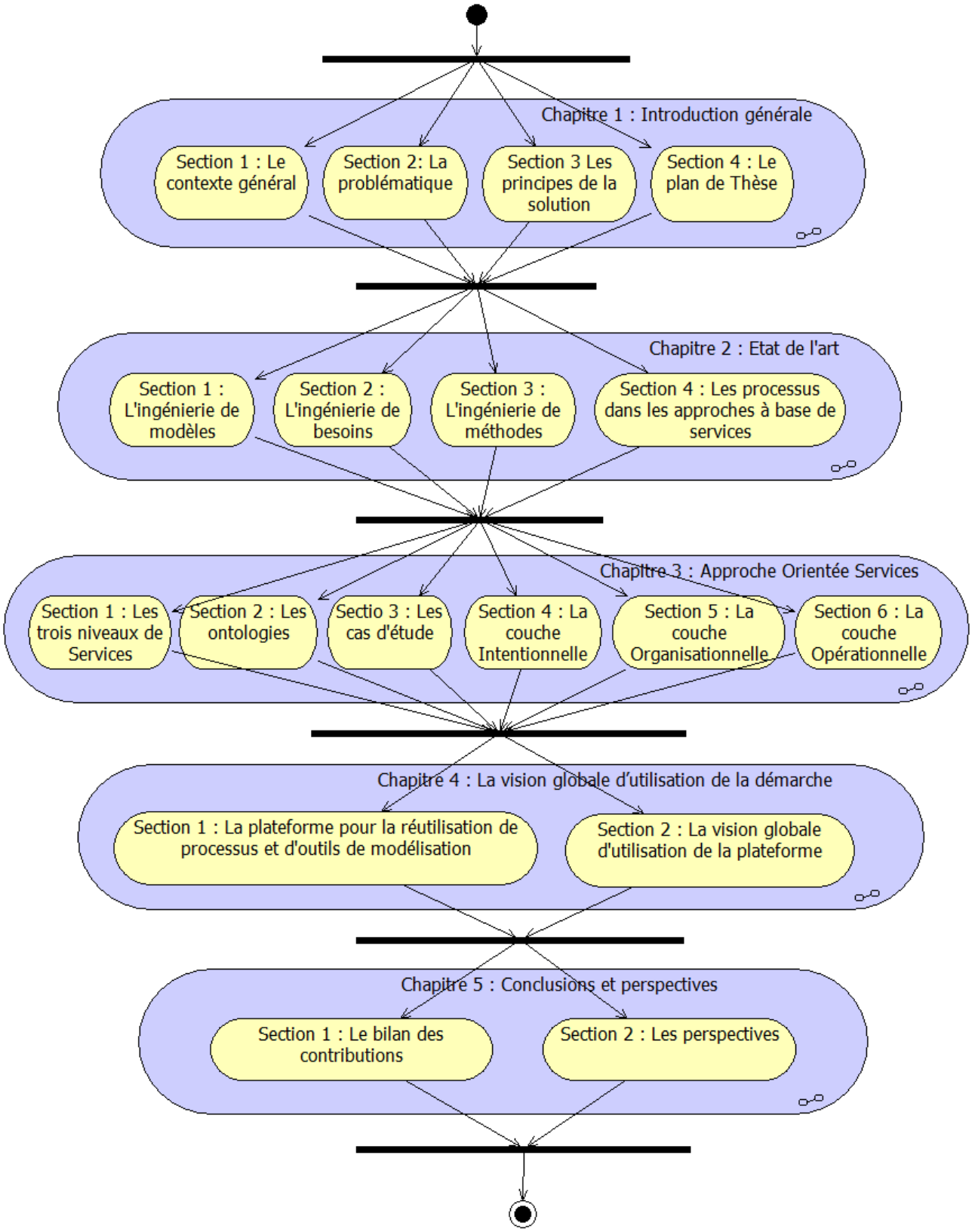


FIGURE 1. PLAN DE LA THESE

CHAPITRE 2

Etat de l'art

Ce chapitre expose un état de l'art sur l'ingénierie des modèles (§1), l'ingénierie des besoins (§2), l'ingénierie des méthodes (§3) et les processus dans les approches à base des services (§4). Cet état de l'art constitue une vue d'ensemble des travaux existants permettant ainsi de positionner les propositions de cette thèse.

CHAPITRE 2

INTRODUCTION

L'ingénierie des modèles (§1) porte sur la description des éléments fondamentaux de modélisation. L'étude de l'ingénierie de modèles décrit le contexte technologique dans lequel se sont déroulés les travaux présentés dans cette thèse. Nous abordons tout d'abord les concepts et les caractéristiques des éléments fondamentaux de modélisation. Nous présenterons aussi les différents types d'outils de modélisation et les comparons suivant des critères technologiques et fonctionnels utiles pour la sélection des environnements de modélisation. L'étude des outils de gestion de modèles aide à définir les caractéristiques opérationnelles requises pour l'approche que nous proposons afin d'assister les méthodes de conception des systèmes.

L'ingénierie des besoins (§2) permet de comprendre et mettre en œuvre les objectifs de l'entreprise dans lequel le système sera utilisé. Le contexte le plus abstrait de fonctionnement du système est centré autour des concepts de scénarios, stratégies, buts et intentions. Cette approche nous aidera à définir un moyen de comprendre les buts des concepteurs de modèles et d'organiser les besoins des concepteurs de modèles.

L'ingénierie des méthodes (§3) permet de construire une méthode à partir de fragments existants, éprouvés et réutilisables. Dans notre étude, nous utiliserons cette approche pour réutiliser des processus (méthodes) afin de faciliter la construction des environnements adaptés aux besoins des concepteurs. Nous nous intéressons à l'ingénierie de méthodes situationnelles, plus particulièrement aux différents aspects concernant la réutilisation et la gestion de modèles pour des méthodes situationnelles, ce qui nous permettra de positionner notre approche orientée services pour la réutilisation de processus mais également d'outils de modélisation.

Finalement, nous présentons au (§4) les travaux proches des nôtres en terme de solution. Cette section décrit la façon dont les processus sont pris en compte dans les approches à base de services. Nous proposons un ensemble de critères permettant de comparer ces approches : le domaine applicatif de chaque approche, la portée de la mise en œuvre de chaque approche, la technologie utilisée, le type d'approche, les clients de l'approche, la communication client-fournisseur en utilisant les ontologies, les niveaux d'abstraction considérés, la modélisation de l'approche et le formalisme d'expression, et finalement les outils supportant l'approche.

La caractérisation des approches à base de services pour la capitalisation et la réutilisation des processus nous permet de positionner notre approche par rapport aux travaux existants.

1. L'INGENIERIE DE MODELES

1.1. Concepts de base dans le domaine de l'ingénierie de Modèles

Suite à l'approche objet des années 80 et de son principe du « **tout est objet** », l'ingénierie du logiciel s'oriente aujourd'hui vers l'ingénierie dirigée par les modèles (IDM) et le principe du « **tout est modèle** ». Cette nouvelle approche peut être considérée comme une continuité de la technologie de l'objet [Bézivin, 2004a], [Bézivin, 2005]. Tout comme les approches patterns [Gamma *et al.*, 1995] et aspects [Kiczales *et al.*, 1997], l'IDM vise à gérer un grand nombre de modèles pour exprimer séparément les préoccupations des utilisateurs, des concepteurs, des architectes, etc.

Alors que l'approche objet est fondée sur deux relations primordiales « *Instance De* » et « *Hérite De* », l'IDM est basée sur un autre jeu de concepts et de relations. Le concept central de l'IDM est la notion de modèle pour laquelle il n'existe pas à ce jour de définition unifiée.

1.1.1. Les Modèles

Minsky [Minsky M., 1965] exprime que « pour un observateur **B**, **M** est un modèle de l'objet **O**, si **M** aide **B** à répondre aux questions qu'il se pose sur **O** ». Cette définition trace les grandes lignes d'un raisonnement basé sur l'intention de représenter un comportement particulier, dépendant d'un contexte d'usage. De nombreux travaux tels que : l'OMG[1], [Bézivin *et al.*, 2001], [Seidewitz E., 2003] s'accordent sur un relatif consensus autour de la relation entre un modèle, le système étudié et de leur complémentarité. Ce consensus est réalisé autour de deux aspects : les relations entre les éléments du modèle (par exemple : les relations d'héritage, cardinalité, …) et les relations inter-modèle (par exemple : un modèle est une représentation de, un modèle est conforme à…). La prise en compte de ces deux aspects peut nous aider à comprendre comment utiliser des modèles (pour raisonner et analyser des systèmes) et comment utiliser les méta-modèles pour spécifier les langages de modélisation.

[1]. The Object Management Group, http://www.omg.org/.

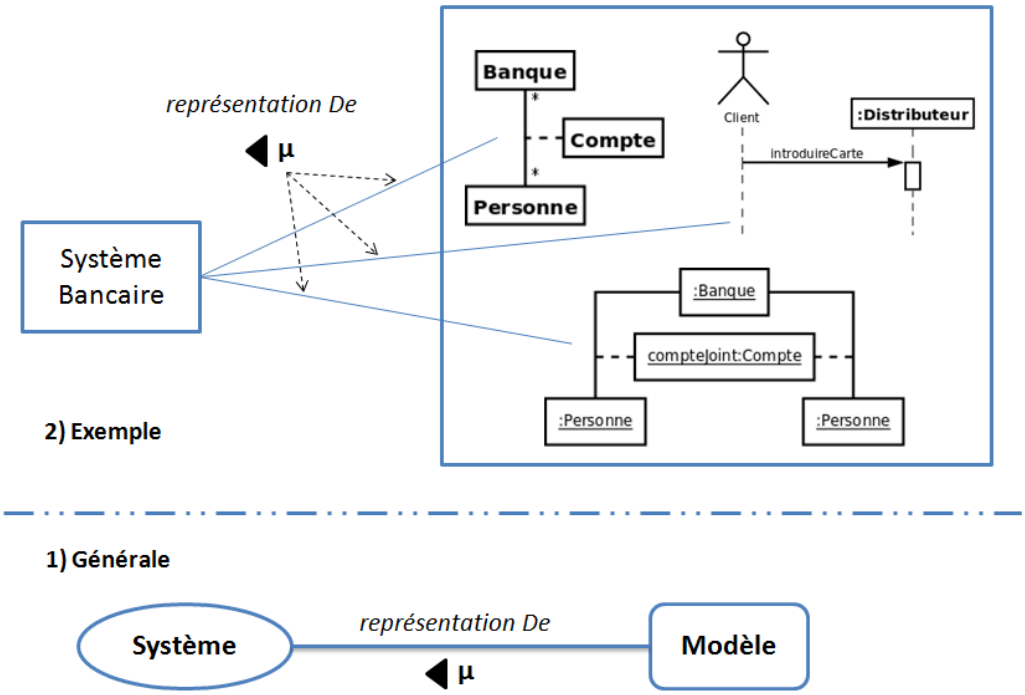


FIGURE 2. RELATIONS ENTRE SYSTEME ET MODELE

Pour J.M. Favre **un modèle est une représentation simplifiée d'un système** [Favre J-M., 2004a], [Favre J-M., 2004b], **modélisée sous la forme d'un ensemble de faits construits dans une intention particulière.** On déduit de cette définition la première relation entre le modèle et le système qu'il représente, appelée : « *représentation De* » dans [Alkinson *et al.*, 2003] [Bézivin, 2004b] [Seidewitz E., 2003], et notée avec le symbole **μ** sur la figure 2 (partie : représentation générale). Cette représentation (modèle) est utilisée pour répondre à des questions sur le système modélisé. Par exemple, la figure 2 (partie : exemple), illustre qu'un modèle est une représentation d'un système (cf. : un système bancaire) selon un point de vue particulier (diagramme de classes, diagramme d'objets, diagramme de séquence).

Notons dans la figure précédente qu'un modèle est exprimé dans un langage particulier. Dans l'IDM, ce langage est décrit par un méta-modèle.

1.1.2. Les Méta-modèles

Un méta-modèle est un modèle de langage de modélisation. Il définit le langage d'expression d'un modèle [OMG, 2006]. La relation qui unit un méta-modèle à un modèle est appelée « *est conforme A* » et noté χ sur la figure 3.

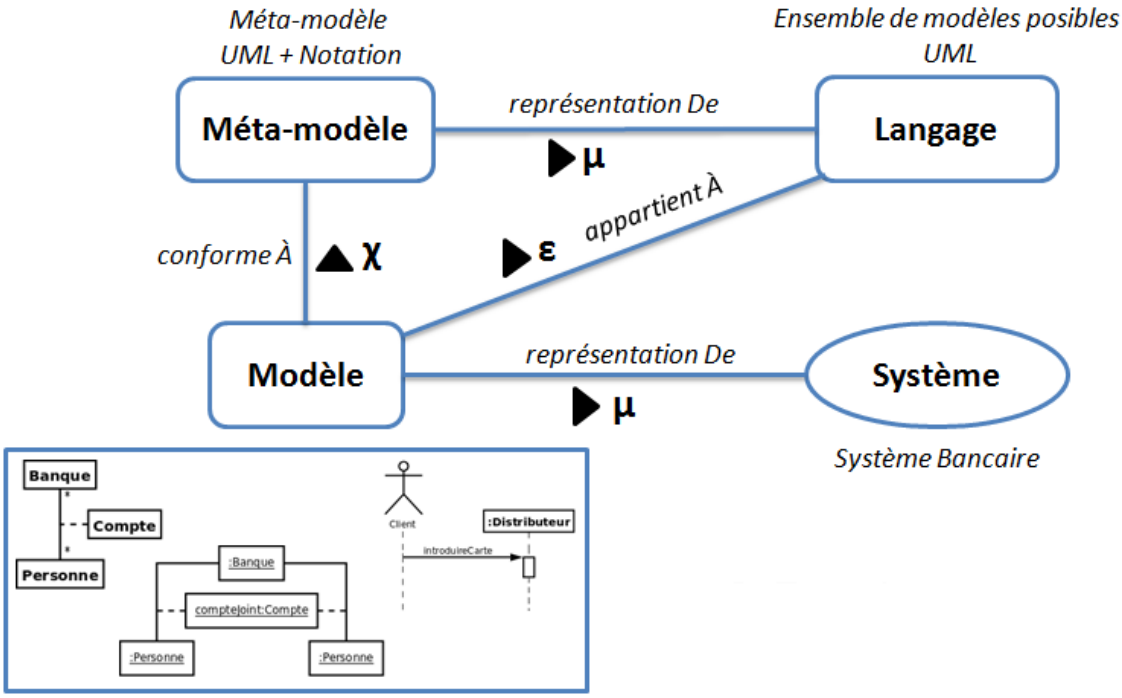


FIGURE 3. RELATIONS ENTRE SYSTEME, MODELE, METAMODELE ET LANGAGE

Dans la figure précédente, les modèles de classes, d'objets et de séquences sont conformes au méta-modèle UML. Ceci se fait notamment sous la forme d'une notation explicite. Par conséquence, les modèles de classes, d'objets et de séquences doivent, pour être utilisables, être conformes à cette notation. La notation est alors considérée comme un modèle représentant l'ensemble de modèles possibles (μ) et à laquelle chacun d'entre eux doit se conformer (χ). Ces deux relations permettent ainsi de bien distinguer le langage qui joue le rôle de système, du (ou des) méta-modèle(s) qui jouent le rôle de modèle(s) de ce langage.

La notion de méta-modèle nous donne en intention un moyen de manipuler le langage de modélisation et ainsi de comprendre et de capitaliser les connaissances en modélisation. L'IDM, utilise la relation « *appartient A* » (noté **ε** sur la figure 3) pour exprimer la relation entre le modèle et le langage dans lequel il est écrit. Le modèle auquel est conforme un méta-modèle est appelé **méta-méta-modèle** [OMG, 2007a] [OMG, 2007b].

Contrairement à la définition du langage de l'OMG, nous utiliserons le terme langage pour décrire la grammaire avec lesquelles sont construits les modèles.

1.1.3. Les Méta-méta-modèles

Un méta-méta-modèle est un modèle qui décrit un langage de méta-modélisation, c'est-à-dire les éléments nécessaires à la définition des langages de modélisation. En tant que modèle, il doit également être défini à partir d'un langage de modélisation. Pour limiter le nombre de niveaux

d'abstraction, on considère alors les langages capables de réflexivité, c'est-à-dire capables de se décrire eux-mêmes, comme le méta-méta-modèle MOF (Meta-Object Facility) [OMG, 2006] proposé par l'OMG ou la grammaire EBNF2 ou EBNF3.

C'est sur ces principes que se fonde la pyramide de l'OMG pour établir une hiérarchie de modélisation (figure 4). Pour un niveau de modélisation Mx, si on parle de modèle Mi, le modèle Mi+1 est son méta-modèle.

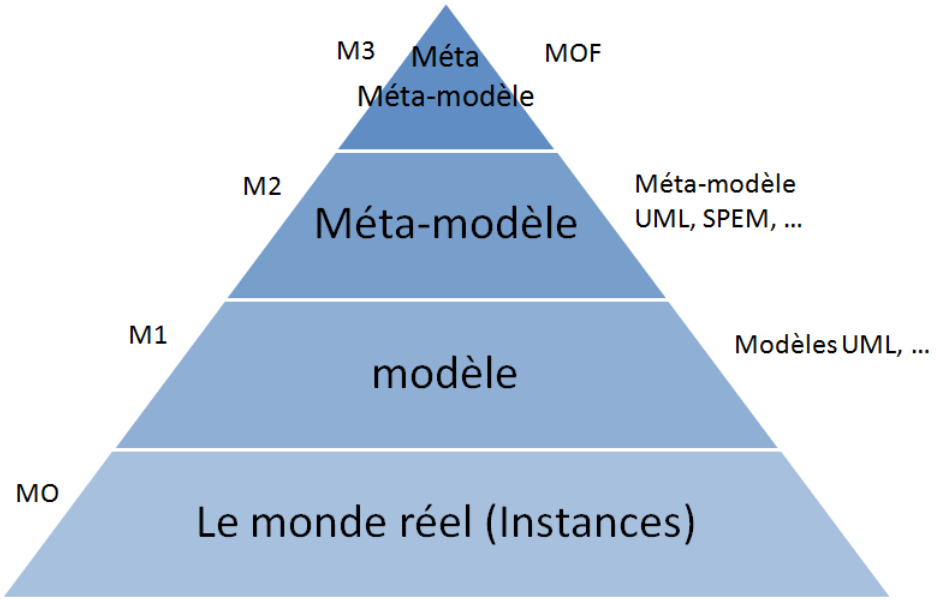


FIGURE 4. PYRAMIDE DE MODELISATION DE L'OMG [BEZIVIN, 2003]

L'approche consistant à considérer une hiérarchie de méta-modèles est également appliquée aux technologies comme les langages de programmation, les documents XML, etc. Chaque hiérarchie définit un espace technique [Kurtev, 2002], [Bézivin *et al.*, 2005a], [Bézivin *et al.*, 2005b]. Donc, un espace technique est l'ensemble des outils et techniques issus d'une pyramide de méta-modèles dont le sommet est occupé par un méta-méta-modèle commun. Le tableau 1 montre les niveaux de modélisation et les relations « *conforme A* » entre les modèles de différents espaces techniques.

2. BNF: Backus-Naur form, un langage pour la définition de grammaires.
3. Extended BNF

M3	Grammarware	Modelware	XLMware
	EBNF	MOF	XML
M2	Grammaire C, Grammaire Java	metamodèle UML	Schéma XML
M1	Un programme C	Un diagramme de classes	Un document XML

Légende

x : Relation « conforme A » **M1** : modèle **M2** : méta-modèle **M3** : méta-méta-modèle

TABLEAU 1. ESPACES TECHNIQUES [FAVRE *ET AL.*, 2006]

1.1.4. Les Langages Spécifiques au Domaine

La définition du concept de méta-modèle comme langage de description de modèles, permet la création de nombreux méta-modèles dédiés à un domaine particulier. Ces méta-modèles correspondent aux Langages Spécifiques au Domaine (LSD) ou en anglais DSL (Domain Specific Language) offrant ainsi aux utilisateurs des concepts propres à leur métier. Nous prendrons comme exemple, le langage CTT [Paternò, 1997] (voir figure 5), qui est un DSL issu du domaine de l'IHM. Il permet d'utiliser les arbres des tâches pour exprimer l'interaction de l'utilisateur.

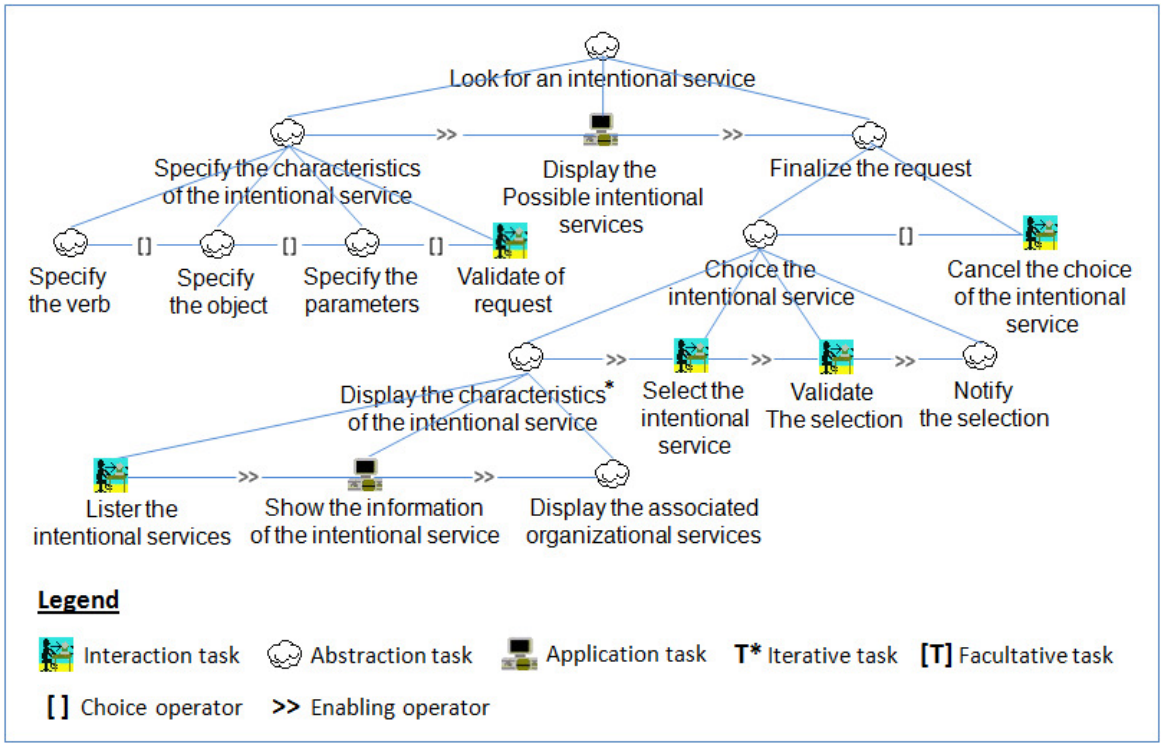


FIGURE 5. EXTRAIT D'UN ARBRE DE TACHE AVEC LE LANGAGE CTT [PATERNO, 1997]

Dans la figure précédente, une tâche désigne un travail déterminé qu'on doit exécuter. Une tâche abstraite (représentée par exemple, par l'élément :  avec la description : « *Look for and intentional service* ») est une tâche qui peut être décomposée. Les feuilles de l'arbre constituent les tâches élémentaires. Elles peuvent être des tâches utilisateur, des tâches d'interaction ou des tâches application. La tâche utilisateur est entièrement exécutée par l'utilisateur, elle requiert des activités cognitives ou physiques sans interaction avec le système. La tâche d'interaction est exécutée par les interactions de l'utilisateur avec le système. Par exemple, dans la figure 4 une tâche d'interaction est représentée par l'élément :  avec la description : « *Validate the request* ».

Une tâche application est entièrement exécutée par le système. Elle reçoit les informations du système et peut fournir des informations à l'utilisateur (représentée par exemple par l'élément :  avec la description : « *Show the information of the intentional service* »).

Ces DSL sont importants pour nous car nous ne souhaitons pas nous limiter à UML et nous nous intéressons particulièrement au domaine de l'IHM.

1.1.5. Les langages de méta-modélisation

Comme nous l'avons indiqué dans les sections précédentes, l'OMG a défini le standard MOF comme langage standard de méta-modélisation [OMG, 2006], c'est-à-dire, un langage commun à tous les langages de modélisation. Cependant, nous disposons à ce jour de nombreux langages de méta-modélisation, tels que :

- **Ecore** [Budisnsky *et al.*, 2003] est un langage de méta-modélisation qui fait partie d'Eclipse Modelling Framework (EMF)4. Ecore permet de modéliser des classes, leurs attributs et les relations qu'elles entretiennent entre elles. Ces relations entre les classes ont des caractéristiques telles que l'unicité, la cardinalité, ou l'existence d'une relation opposée.
- **Domain Specific language (DSL)** [Cook *et al.*, 2007] est un langage de méta-modélisation qui fait référence au concept de langage spécifique au domaine qui a été étudié précédemment. Ce langage de méta-modélisation permet de définir de nombreux méta-modèles dédiés à des domaines particuliers.
- **MetaGME** [Ledeczi *et al.*, 2001] est un langage de méta-modélisation permettant de créer des méta-modèles spécifiques à un domaine. L'élaboration de méta-modèles est basée sur un diagramme de classes UML et de contraintes OCL.

4. EMF : est un Framework de modélisation et génération de code pour supporter la création d'outils et d'applications dirigées par les modèles [Budisnky et al., 2003]. Dans cet environnement, la persistance des modèles est assurée par un fichier XMI (XML metadata interchange) spécifique à EMF.

- **Kernel Metametamodel (KM3)** [Jouault *et al.*, 2006] est un langage de méta-modélisation permettant de définir des méta-modèles sur la base de déclaration de classes java. Il sert entre autre de modèle pivot entre EMF et DSL [Kurtev *et al.*, 2005].
- **Java Metadata Interface (JMI)** [Sun, 2002] est un langage de méta-modélisation qui permet d'implémenter une infrastructure dynamique et portable pour gérer la création, l'enregistrement, l'accès, la recherche, et l'échange des métadonnées. JMI est basé sur le standard MOF. JMI définit des règles permettant de construire des interfaces Java à partir de méta-modèles.
- **Kermeta** [Muller *et al.*, 2005] est un langage de méta-modélisation défini pour écrire des transformations entre modèles. Il dispose d'un environnement de développement de méta-modèles basé sur Essential MOF (EMOF) qui est un sous-ensemble d'Ecore.

Le tableau 2 résume des relations entre les langages de méta-modélisation.

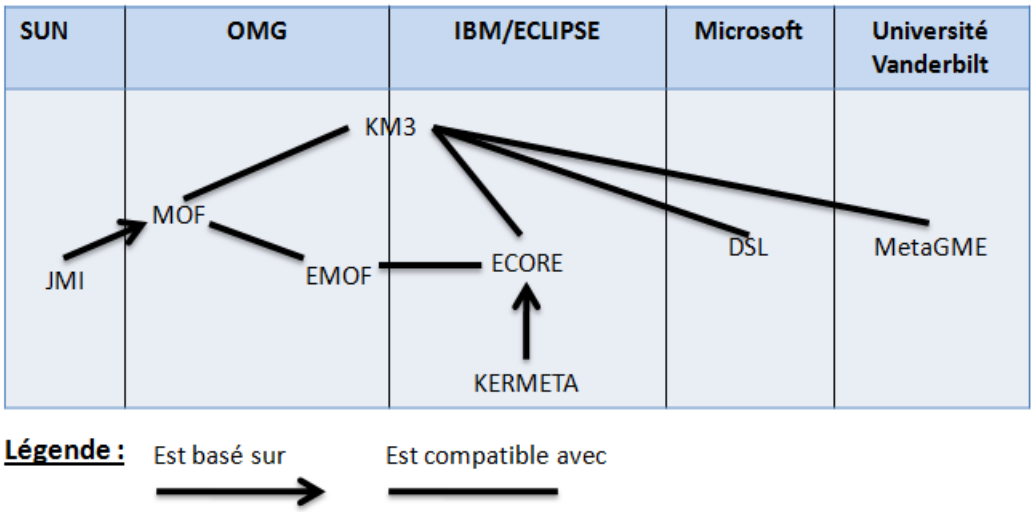


TABLEAU 2. LES LANGAGES DE METAMODELISATION

1.1.6. Le format d'échange normalisé : XMI

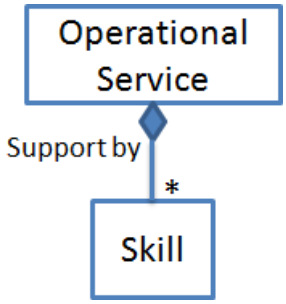
Le **XML Metadata Interchange (XMI)** est le langage standard de l'OMG pour échanger les méta-données via le **eXtensible Markup Language (XML)**. XMI se focalise sur les échanges de méta-données conformes au MOF [OMG-XMI, 2003]. L'usage le plus commun de XMI est l'échange de modèles UML, bien qu'il puisse être utilisé pour la sérialisation de modèles d'autres langages (méta-modèles).

XMI décrit comment utiliser les balises XML pour représenter un modèle UML en XML. Donc, cette représentation facilite les échanges de données (ou méta-données) entre les différents outils ou plateformes de modélisation. En effet, XMI définit des règles permettant de construire des DTD (Document Type Definition) et des schémas XML à partir de méta-modèle, et inversement. Ainsi, il est

possible de sérialiser un méta-modèle dans un document XML, mais aussi, à partir de document XML il est possible de reconstruire des méta-modèles.

XMI a l'avantage de regrouper les méta-données et les instances dans un même document ce qui permet à une application de comprendre les instances grâce à leurs métadonnées. Ceci facilite les échanges entre applications, certains outils peuvent automatiser ces échanges en utilisant un moteur de transformation du type XSLT [Bézivin *et al.*, 2002]. Donc, XMI est utilisé pour sérialiser et échanger des méta-modèles MOF et des modèles basés sur ces méta-modèles sous forme de fichiers en utilisant des dialectes XML. Ceci est possible parce que XMI ne définit pas un dialecte XML unique, mais un ensemble de règles qui permettent de créer une DTD ou schéma XML pour différents méta-modèles. Ainsi, les méta-modèles conformes à MOF ou à Ecore et leurs modèles respectifs peuvent être portables en utilisant XMI.

La Figure 6 présente un modèle UML élémentaire, et sa représentation correspondante en XMI.




```
<?xml version = '1.0' encoding = 'UTF-8' ?>
 <XMI xmi.version = '1.2' xmlns:UML = 'org.omg.xmi.namespace.UML'>
   <XMI.content>

   <UML:Model xmi.id = 'a1' name = 'model 1' isSpecification = 'false' isRoot = 'false' isLeaf =
    'false' isAbstract = 'false'>

   <UML:Class xmi.id = 'a4' name = 'OperationalService' visibility = 'public' isSpecification =
    'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' isActive = 'false'>
   </UML:Class>

    <UML:Class xmi.id = 'a6' name = 'Skill' visibility = 'public' isSpecification = 'false' isRoot =
     'false' isLeaf = 'false'  isAbstract = 'false' isActive = 'false'>
    </UML:Class>

    <UML:Association xmi.id = 'a15' name = 'support by' isSpecification = 'false' isRoot = 'false'
     isLeaf = 'false' isAbstract = 'false'>
     </UML:Association>

   </XMI.content>
 </XMI>
```

FIGURE 6. UN MODELE UML ET SA REPRESENTATION EN XMI

1.2. Le savoir faire en IDM

De la même manière qu'il existe une collection de modèles ayant diverses propriétés, diverses fonctionnalités sur ces modèles peuvent être réalisées. La gestion de modèles englobe tous les types de fonctionnalités qui permettent la représentation et/ou la manipulation de modèles.

La représentation de modèles correspond aux fonctionnalités de base sur les modèles. Elle regroupe toutes les fonctionnalités d'édition de modèles telles que : l'affichage-présentation, la capture-récupération, le stockage, etc.

La manipulation de modèles comprend aussi les fonctionnalités qui permettent de comparer les modèles telles que : la confrontation de modèles, la mise en correspondance, l'alignement, la fusion, la transformation, etc. Etant donné que le but de l'IDM est d'encapsuler le savoir faire sur les

modèles autour d'un processus de transformation et de vérification de la cohérence, nous avons choisi de leur consacrer une partie pour mieux les expliciter.

1.2.1. La transformation de modèles

Selon [Mens *et al.*, 2006] une transformation peut être vue comme un ensemble de règles qui décrivent comment un ou des modèles sources sont transformés en modèles cibles. Les règles d'une transformation se basent sur les méta-modèles dans lesquels sont exprimés les modèles.
Le modèle produit par une transformation peut être du code, des cas de test, des modèles graphiques, etc.

Il y a plusieurs types de transformation : modèles vers modèles et modèles vers code (ou modèles vers texte) [Czarnecki et *al.*, 2003]. L'IDM, considère toutes ces opérations sur des modèles comme des transformations. C'est une des idées clef dans l'IDM « considérer toutes les opérations génératives de la même manière ».

Actuellement, différents travaux de transformation de modèles [Bézivin *et al.*, 2006], [Jouault, 2006], [Czarnecki *et al.*, 2003] existent. Ces approches visent à considérer une transformation comme un autre modèle (cf. figure 7) conforme à son propre méta-modèle (lui-même défini à l'aide d'un langage de méta-modélisation, par exemple MOF). La figure 6 illustre le principe de transformation d'un modèle. La transformation d'un **modèle source Ma** : par exemple, le diagramme de cas d'utilisation (conforme à son **méta-modèle MMa** : par exemple, UML) vers un **modèle cible Mb** : par exemple, l'arbre de tâches (conforme à son **méta-modèle MMb** : par exemple, DSL) est réalisé par le **modèle de transformation Mt** (conforme à son **méta-modèle MMt** : par exemple, ATL). Touts les méta-modèles utilisés dans le processus de transformation doivent être conformes à une spécification commune, par exemple la MOF, proposée par l'OMG. Finalement ces méta-modèles peuvent être outillés en utilisant la plate-forme Eclipse.

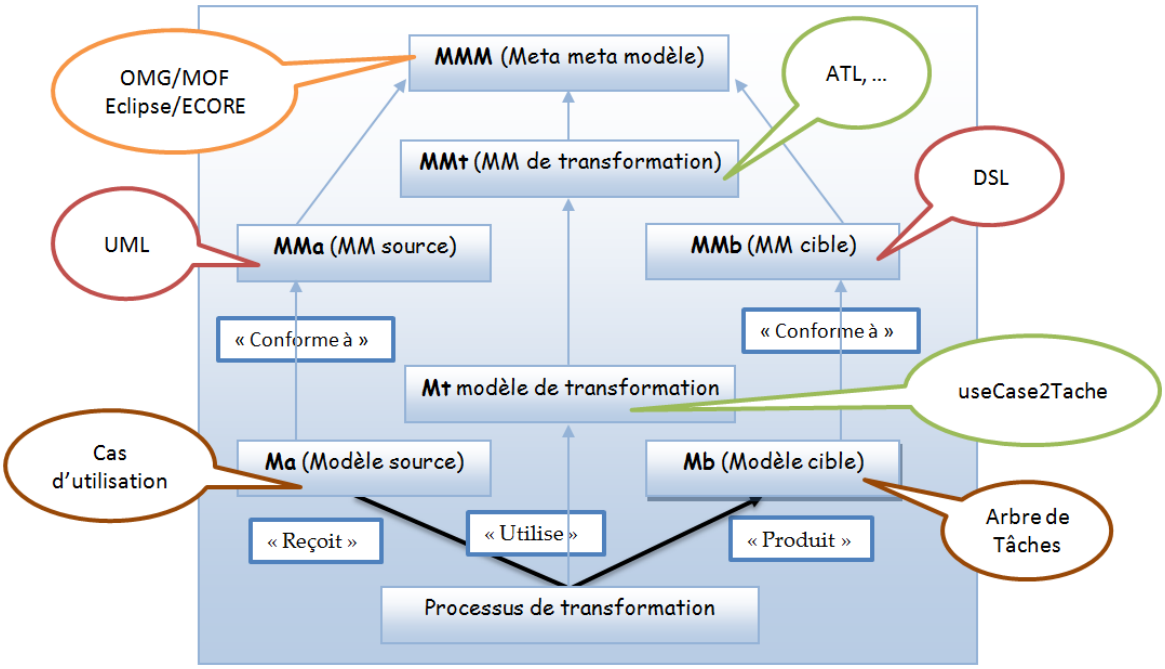


FIGURE 7. LE PRINCIPE DE TRANSFORMATION D'UN MODELE

On peut définir un méta-modèle de transformation pour définir un langage abstrait de transformations. Les modèles instances de ce méta-modèle sont des spécifications de transformations entre deux méta-modèles spécifiques. De cette manière, la spécification de transformation devient lisible (comprendre un modèle est plus facile que de comprendre du code), et réutilisable (le méta-modèle représente les règles de transformation de manière abstraite indépendante du langage de sa mise en œuvre).

Les règles de transformation sont établies entre le méta-modèle source et le méta-modèle cible, c'est-à-dire entre l'ensemble des concepts du modèle source et celui du modèle cible. Le processus de transformation prend en entrée un modèle conforme au méta-modèle source et produit en sortie un ou plusieurs autre(s) modèle(s) conforme(s) au méta-modèle cible, en utilisant les règles préalablement établies. Un exemple de transformation qui explicite le mécanisme de la transformation d'un diagramme de cas d'utilisation vers des arbres de tâches, est présenté dans la section suivant. La transformation permet de générer une tâche pour chaque cas d'utilisation et de structurer partiellement ces tâches.

1.2.2. Les standards et langages pour la transformation de modèles

De nombreux langages de transformation sont à ce jour disponibles :

- les langages graphiques comme TrML [Etien *et al.*, 2007],
- les langages basés sur XML-XSLT [W3C, 2007],
- les langages basés sur le langage de programmantion Java JMI [JMI, 2002],

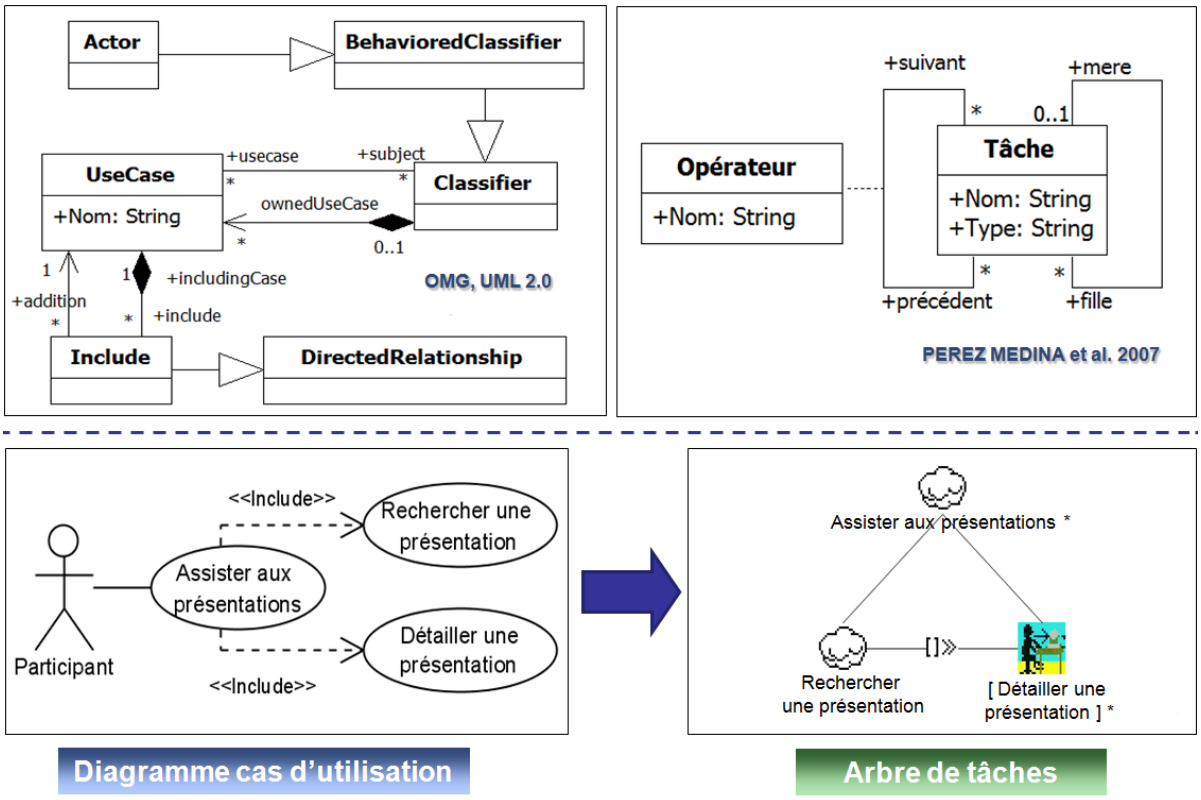
- les langages ad-hoc comme MOLA [Kalmins *et al.*, 2004] et MTL [Vojtisek *et al.*, 2004],
- et enfin, les langages basés sur le standard (Query, View, Transformation) QVT [OMG-QVT, 2005].

QVT est une spécification de l'OMG. Il définit plusieurs types de syntaxes : déclarative, impérative, hybride (QVT-relation, etc.) ainsi qu'une sémantique (QVT-Core). Les principes de QVT ont été implémentés dans de nombreux langages comme ATL [Allilaire *et al.*, 2004], Medini QVT [MediniQVT, 2007], Viatra2 [Horvath *et al.*, 2006] et TGG [Greenyer, 2006].

La figure 8 montre un exemple d'une règle de transformation en ATL pour la transformation du diagramme de cas d'utilisation en arbre de tâches [Pérez-Medina et al., 2007].

La première partie de la règle de transformation spécifie qu'à un cas d'utilisation correspond une tâche. Dans la transformation ATL, nous pouvons voir que nous partons d'un cas d'utilisation (partie 1) pour obtenir une tâche (partie 2). Pour créer la tâche (partie 2), le nom du cas d'utilisation (u.nom) est récupéré et utilisé pour donner un nom (nom) à la tâche créée.

La deuxième partie de la règle de transformation exprime le fait qu'un « include » d'un diagramme de cas d'utilisation donne lieu à une sous-tâche dans un arbre de tâches (fille<-u.include). Ainsi un « include » entre deux cas d'utilisation donne lieu à une sous tâche. Les sous-tâches seront juste classées en tant que tâches filles d'une tâche donnée. Ainsi, une inclusion de cas d'utilisation donne lieu à un lien mère/fille (partie 3). Par contre, rien n'est spécifié pour le séquencement (partie 4). Nous obtenons un arbre de tâches conforme à son méta-modèle. Ce modèle reste néanmoins incomplet. Le concepteur en IHM devra donc le compléter, avec en particulier, les opérations de séquencement entre tâches.




```
module Include2Soustache; -- Module Template

create OUT : MetamodelTache from
       IN : MetamodelUseCase;

rule useCase2Tache {

from    u: MetamodelUseCase!UseCase } 1
to      t: MetamodelTache!Tache (
                nom <-u.nom,
           3 { mere <-u.includingCase,
               fille <- u.include,
           4 { suivant <- u.includingCase.include,
               precedent <- u.includingCase.include ) } 2

                }
```

FIGURE 8. EXEMPLE DE REGLE DE TRANSFORMATION EN ATL [PEREZ-MEDINA ET AL., 2007]

1.2.3. La mise en correspondance par tissage

L'IDM n'est pas limitée aux transformations de modèles. [Didonet del Faro *et al.*, 2005] propose une autre opération sur les modèles appelée « le tissage de modèles ou méta-modèles ». Le tissage est une opération sur les modèles qui précise les différents types de relations (i.e. des liens) entre les éléments de modèles (ou méta-modèles). Ce lien propose alors la relation entre le modèle source et

le modèle cible de la transformation. Elle permet également de voir quelles règles sont utilisées pour réaliser cette transformation.

Afin d'expliquer le tissage de modèles, nous considérons un système d'information simple pour une bibliothèque décrit dans [Didonet *et al.*, 2007]. Dans ce contexte, la figure 9 montre un exemple d'un modèle de tissage qui capture les relations entre un méta-modèle avec des structures fixes et des relations référentielles : « Foreign Keys » (représentant une clé étrangère en base de données relationnelle, de référence : « R1 ») et un méta-modèle qui contient des structures imbriquées (« Nested ») (représentant une base hiérarchique de données en XML, de référence : « X1 »). Ainsi, les deux schémas représentent la même information, mais des structures de données différentes sont utilisées. Une opération de tissage est spécifiée pour saisir les liens entre les deux schémas avec toutes les informations sémantiquement pertinentes.

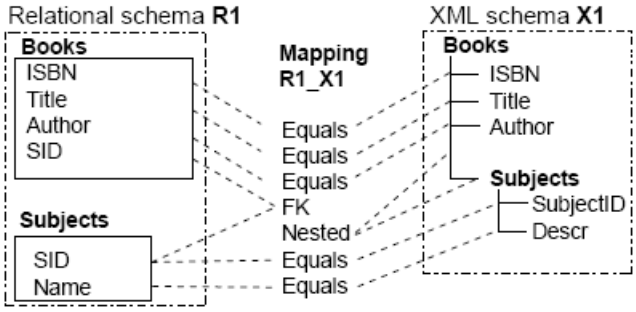


FIGURE 9. LIENS ENTRE UN MODELE RELATIONNEL ET UN SCHEMA XML

Dans la figure précédente, la base de données relationnelle (« RI ») possède deux tables : **Livres** (avec les attributs : ISBN, Titre, Auteur, SID) et **Sujets** (SID, Name). L'attribut SID référencie les sujets d'un livre. Le schéma X1a la même structure de base, sauf pour la clé référentielle dans livres, cette correspondance est représentée par la structure imbriquée (« nested ») entre les livres et les sujets.

Les liens entre R1 et X1 sont représentés par le modèle de tissage R1_X1 comme l'illustre la figure 8. Par exemple : alors que les sujets ont un « ***nom*** » dans R1, ils sont appelés « ***Descr*** » dans X1. Le modèle de tissage dispose de trois structures de tissage : 1) « ***Equals*** » une correspondance entre inter-schéma qui indique les égalités telles que R1.Books.ISBN = X1.Books.ISBN, R1.Books.Title = X1.Books.Title ; 2) « ***ForeignKey (FK)*** » une correspondance intra-schéma qui indique la contrainte d'intégrité référentielle entre R1.Books.SID et R1.Subjects.SID ; 3) « ***Nested*** » une autre intra-correspondance qui représente la relation entre X1.Books et X1.Books.Subjects.

Deux langages à ce jour sont disponibles pour effectuer la mise en correspondance de modèles :

- **Atlas Model Weaver language**, est un langage de tissage (weaving language) qui a une partie « core » fournissant les concepts génériques de base permettant de créer les liens structurels entre les modèles [Didonet Del Faro *et al.*, 2005]. Les liens inter-modèles sont

stockés dans un modèle appelé « weaving model » qui est défini sur la base du méta-modèle « weaving metamodel ».

- **Embedded Constraint Language (ECL)** [Gray *et al.*, 2001] est un langage de contraintes qui est une extension du langage de contraintes OCL. ECL définit un ensemble d'opérateurs permettent de naviguer dans la structure hiérarchique d'un modèle.

1.3. Les outils de gestion de modèles

De nombreux outils, tant commerciaux que dans le monde de la recherche, sont aujourd'hui disponibles ou en cours de développement. Un grand nombre d'entre eux sont publiés en logiciels libres et sont utilisables gratuitement. Ces outils sont conçus comme des « frameworks » [Amelunxen, 2006], ou des plugins [Alliaire *et al.*, 2004]. Plusieurs travaux de classification [Eclipse, 2007], [Planet, 2007] et de comparaison [García *et al.*, 2004], [Tariq *et al.*, 2005], [Meylan, 2006] d'outils ont été proposés.

Dans ces études, nous pouvons distinguer trois catégories d'outils de gestion de modèles : les Ateliers de Génie Logiciel (AGLs), les outils de méta-modélisation et les outils de transformation de modèles.

Il existe une grande diversité d'outils qu'il est parfois nécessaire de combiner les uns aux autres pour obtenir un ensemble de fonctionnalités permettant d'aider à la réalisation de la plupart des activités du cycle de développement d'un système d'information.

1.3.1. Les Ateliers de Génie Logiciel (AGL)

Les Ateliers de Génie Logiciel (AGL) ou ateliers CASE (Computer Aided Software Engineering) sont un ensemble d'outils logiciels, formant des environnements d'aide à la modélisation du système d'information, à sa validation, à la génération de ses applications et de ses interfaces, à son administration et à la mise au point des logiciels. Les AGL sont dotés de fonctionnalités d'édition de modèles adaptées aux différentes phases de la production d'un logiciel et facilitent la communication et la coordination entre ces différentes phases. Un AGL est basé sur des méthodologies qui formalisent le processus logiciel, et à l'intérieur de ce processus, chacune des phases qui le composent.

Généralement, les outils sont limités à la création de modèles conformes à des modèles propriétaires. Par exemple : l'AGL CTTE [Mori *et al.*, 2002] permet la modélisation des arbres de

tâches conformes au modèle CTT [Paternò, 1997]). Ils peuvent également s'appuyer sur le langage de modélisation UML (par exemple : starUML[5], Rational Rose[6], ArgoUML[7], Objecteering[8], …).

Actuellement, les AGLs offrent les fonctionnalités classiques de modélisation, (par exemple : la création, l'édition, la documentation, la suppression, l'impression, la visualisation de modèles).

Nous notons qu'un apport essentiel des AGL est de permettre de documenter automatiquement un programme ou un modèle, et de maintenir en permanence à jour cette documentation, et ce tout au long de sa conception. Certains AGL peuvent aller jusqu'à la génération de code ou à l'inverse peuvent inclure des fonctionnalités de rétro-ingénierie. Enfin, un AGL facilite la collaboration des différents spécialistes, ainsi que la maintenance ultérieure des programmes en incitant les acteurs des équipes projets à partager les mêmes méthodes. Le tableau 3 montre une liste des AGL.

Outil	Version	Plateforme	Description
AndroMDA	3	Java VM	AndroMDA est un outil open source capable de générer du code dans n'importe quel langage à partir de modèles UML. La génération du code s'effectue grâce à des composants que l'on appelle « cartridge ». Il existe une multitude de cartridges déjà existants que l'on peut utiliser en parallèle pour générer le code que l'on souhaite. Les cartridges déjà existant génèrent tous du code Java pour des librairies open source bien connues telles que Spring, Hibernate ou Struts. Il est possible d'écrire ses propres cartridges afin de générer du code dans un autre langage ou améliorer les cartridges déjà existants. http://www.andromda.org/index.php
Magic Draw	10	Java VM	MagicDraw est un outil graphique de modélisation UML disposant de fonctions de travail collaboratif. Conçu pour les Consultants Métier, les analystes développeurs, les développeurs, les ingénieurs qualité, et les rédacteurs de documentation, cet outil de développement facilite l'analyse et la conception de systèmes orientés objets(OO) et de bases de données. Il fournit un mécanisme d'engineering de code de l'industrie (avec un reverse complet pour les environnements J2EE, C#, C++, CORBA IDL, .NET, XML Schéma, WSDL) ainsi que le modèle de base de données, la génération de DDL. http://www.magicdraw.com/
Umbrello UML Modeller	2.0	Linux	Umbrello UML Modeller est un logiciel libre de modélisation UML disponible en natif sous Unix et faisant partie de l'environnement de bureau KDE. Il peut générer du code à partir d'un modèle UML, vers le langage Java, ou C++, PHP, Python, Ruby, SQL, etc. Il est distribué selon les termes de la licence GNU GPL. http://uml.sourceforge.net/index.php
StarUML	5.02	Windows	StarUML est un logiciel de modélisation UML open source sous une licence modifiée de GNU GPL. StarUML est écrit en Delphi, il est modulaire et propose plusieurs générateurs de code. StarUML permet d'installer des modules pour la génération de code source C++, C#, Java, ... ou le module patterns. http://staruml.sourceforge .net/en/
UMLet	10.3	Java VM	UMLet est un outil léger pour dessiner rapidement des diagrammes UML, avec une interface utilisateur simple. UMLet permet l'exportation de diagrammes UML au format SVG, JPG, PDF, LaTeX et conviviale EPS. http://www.umlet.com/
Rational Rose Modeler	1.0	Windows	Rational Rose Modeler est un outil de modélisation UML. Il permet la génération de code Java. http://www-01.ibm.com/software/awdtools/developer/rose/modeler/
MetaEdit+	3.0	Windows, Unix	Est un outil qui permet la génération de code à partir de modèles de haut niveau. http://www.metacase.com/index.html
PowerDesigner	9.5	Windows	PowerDesigner est un outil de modélisation d'applications informatiques. La version 9.5 permet de générer des scripts SQL compatibles avec PostgreSQL. http://www.sybase.com/products/modelingdevelopment/powerdesigner
Open ModelSphere	3.1	Windows	Open- ModelSphere est un outil de modélisation de données qui prend en charge la modélisation conceptuelle, logique et physique d'applications. Il supporte plusieurs formalismes, entre autres : entité-association, Datarun, et information engineering. Les modèles conceptuels peuvent être convertis en modèles relationnels et vice-versa. Il permet de visualiser graphiquement l'architecture de la base de données

[5].The Open Source UML/MDA Platfrom : http://staruml.sourceforge.net/en/
[6].IBM Rational software : http://www-01.ibm.com/software/fr/rational/
[7].ArgoUML : http://argouml.tigris.org/
[8].The Open Source UML/MDA Platfrom : http://www.objecteering.com/

			relationnelle. Open ModelSphere supporte l'ensemble des systèmes de gestion de bases de données de façon générique pour certains systèmes de gestion de bases de données, comme : Oracle, Informix et DB2. http://www.modelsphere.org/index.html
Visual UML	3.1	Windows	Visual UML est un outil de modélisation UML. Il permet d'analyser, de dessiner, de coder, de tester et de déployer. Visual UML permet de dessiner tous les types de diagrammes UML, d'inverser le code source JAVA vers UML et la génération de code source à partir de diagrammes et d'élaborer la documentation. http://www.visualuml.com/
Poseidon	8.0	Java VM	Poseidon est un outil de modélisation des diagrammes UML, qui supporte la génération de code Java et l'échange des données sous la forme XMI et SVG. http://www.gentleware.com/
Fujaba	4.0	Java VM	Fujaba est un outil de modélisation open source qui permet la ré-ingénierie de code JAVA vers les modèles UML. http://www.fujaba.de/
Together Designer	2008	Windows, Linux	Together est un outil de modélisation qui supporte les modèles UML et BPMN. http://www.borland.com/us/products/together/index.html
Objecteering	5.2	Windows, Unix	Objecteering est un outil de modélisation UML qui supporte la gestion de modèles basées sur le standard BPMN. Cet outil permet la création des profils UML qui peuvent être ajoutés pour définir des modèles personnalisées. http://www.objecteering.com/
Visual paradigm for UML	4.2	Java VM	Visual Paradigm pour UML est un outil de modélisation qui supporte UML 2.0 et la ré-ingénierie de code Java. Cet outil permet l'exportation des modèles sous format HTML et/ou PDF. La version professionnelle supporte l'intégration des IDEs telles que : Eclipse, JBuilder, Netbeans. http://www.visual-paradigm.com/

TABLEAU 3. LISTE DES AGL

1.3.2. Les Outils IDM

Dans le contexte de l'IDM, les concepts de méta-modélisation et de transformation de modèles jouent un rôle fondamental. De nombreux outils sont aujourd'hui disponibles.

Le tableau 4 présente une liste des outils que nous avons examinés. Nous utiliserons le qualificatif (OM) pour faire référence aux outils de méta-modélisation et (OT) pour signaler les outils de transformation de modèles. Ces outils couvrent les fonctionnalités suivantes de gestion de modèles :

- support à la méta-modélisation et à la modélisation,
- support à la transformation de modèles, d'un espace de modélisation ou d'un niveau d'abstraction vers un autre,
- expression (affichage/présentation) textuelle et graphique de modèles et méta-modèles,
- expression de manière textuelle et graphique de règles de transformation,
- support à la vérification de la cohérence inter-modèles,
- interopérabilité des différents éditeurs de modèles.

Outil	Version	Catégorie	Description
ACCELEO GPL – Open source	2.0	OT	Acceleo est un **générateur de code** basé sur les standards MDA, permettant le passage de modèles de haut niveau fonctionnel vers des cibles technologiques hétérogènes. http://www.acceleo.org/pages/accueil/fr
AndroMDA Open source	3.2	OT	Outil basé sur les Template pour **générer du code J2EE** à partir d'UML/XML. http://galaxy.andromda.org/index.php?option=com_frontpage&Itemid=48

ADT Open source	2.0	OT	ATL est un langage permettant de **transformer des modèles**. ATL fait partie du projet M2M (Model to Model) de la fondation Eclipse. ADT propose un environnement de développement de transformation intégré et des librairies de transformations open source. http://www.eclipse.org/m2m/atl/
AToM3 Open source	2.2	OM, OT	Un Outil pour **méta-modélisation** et **méta-transformation** soutenant la modélisation de systèmes complexes. http://atom3.cs.mcgill.ca/
DSL Tools (Visual Studio 2005 SDK)	4.0	OM, OT	DSL Tools permet la **construction d'éditeurs graphiques personnalisés** et la **génération de code source** à partir d'une modélisation des concepts métiers exprimée dans un langage spécifique. http://msdn2.microsoft.com/en-us/vstudio/aa718368.aspx
Kermeta GPL – Open source	0.4.1	OM, OT	Kermeta est un **langage de metamodélisation exécutable** disponible en Open-source. Il dispose d'un environnement de développement de metamodèles basé sur EMOF dans un environnement Eclipse. Il permet non seulement de décrire la structure des méta-modèles, mais aussi leur comportement. Il permet ainsi de définir et d'outiller de nouveaux langages en améliorant la manière de spécifier, simuler et tester la sémantique opérationnelle des metamodèles. http://www.kermeta.org/
ModFact GPL – Open source	1.0.1	OT, OM	ModFact fournit un framework pour la construction d'applications réparties selon une approche orientée modèle, conformément aux standards de l'OMG relatifs au Model Driven Architecture (MDA). http://modfact.lip6.fr/
Merlin Open source	0.5.0	OT, OM	Un outil de modélisation basé sur la **transformation de modèle vers modèle** et **la génération de code**. http://merlingenerator.sourceforge.net/merlin/index.php
MDA Workbench Open source	3.0	OT	Est un plugin d'Eclipse basé sur **la modélisation** et la **transformation de code**. http://sourceforge.net/projects/mda-workbench/
MOFLON Open source	1.1.0	OT, OM	Un **outil de metamodélisation**, basé sur un environnement intégré pour le développement d'applications JAVA. http://www.moflon.org/
OptimalJ Professional Edition	3.0	OT	Un outil pour la **génération de code**. Il est basé sur l'utilisation de patrons/templates. http://www.compuware.com/products/optimalj/
QVT Partners BSD like licence	0.1	OT	Il est basé sur le standard QVT pour la **transformation de modèles** et **la génération de code**. http://qvtp.org/downloads/qvtp-eclipse/
SmartQVT Open source	0.1.4	OT	Un outil de **transformation de modèle vers modèle** basé sur le standard QVT-Opérationnel. http://smartqvt.elibel.tm.fr/
UMLX Open source	0.0.2	OT	Un outil de modélisation pour la **transformation de modèle vers modèle**. http://dev.eclipse.org/viewcvs/indextech.cgi/gmt-home/subprojects/UMLX/

TABLEAU 4. OUTILS DE META-MODELISATION

1.3.2.1. Les Outils de Méta-modélisation (OM)

Les outils de méta-modélisation (OM) permettent de modifier le méta-modèle sur lequel ils sont basés, ou tout au moins, ils doivent permettre d'étendre ce méta-modèle.

Les outils énoncés dans le tableau 4 permettent l'expression de méta-modèles et de modèles sous une forme textuelle ou graphique (Tableau 5). Nous notons également que des contraintes peuvent être ajoutées pour compléter les modèles et méta-modèles. Les contraintes sont écrites en OCL, un langage de contraintes pour le langage UML.

Outil	Expression (Metamodèles)		Expression (Modèle)	
	Graphique (G) ou Textuelle (T)	Contraintes	Graphique (G) ou Textuelle (T)	Contraintes
ACCELEO	G, T	OCL	G, T	OCL
AndroMDA	T	OCL	G, T	OCL
ADT	T	OCL	T	OCL
AToM3	G	-	G	-
DSL tools	G, T	-	G, T	-
Kermeta	T	OCL	G, T	OCL
ModFact	G	-	G	-
Merlin	G, T	OCL	G, T	OCL
MDA Workbench	G, T	OCL	G, T	OCL

MOFLON	G, T	OCL	G, T	OCL
OptimalJ	G	OCL	G	OCL
QVT Partners	G, T	OCL	T	OCL
SmartQVT	T	OCL	T	OCL
UMLX	G, T	OCL	G, T	OCL

TABLEAU 5. OUTILS EN TERME D'EXPRESSION DE METAMODELES ET MODELES

1.3.2.2. Les Outils de Transformation de Modèles (OT)

Comme nous l'avons mentionné dans les sections précédentes, l'IDM propose des opérations de transformation de modèles et de mise en correspondance.

Les outils de transformation permettent la transformation d'un modèle vers un ou plusieurs modèles cibles ou la modélisation de règles de cohérence inter modèles. Ces outils sont nombreux et divers, la plupart d'entre eux sont plus ou moins intégrables dans les environnements de développement standard ou les AGL. Le tableau 6 synthétise les opérations disponibles pour quelques outils qui supportent les transformations de modèles.

La majorité des outils supportent le standard QVT. Cela garantit que le résultat d'une transformation est compatible avec un autre outil qui utilise QVT.

Le tableau 6 illustre la forme résultante (texte ou modèle) de la génération de modèles. Le mot « texte » est utilisé lorsque le résultat d'une transformation est textuelle. Généralement, le résultat est un code écrit dans un langage de programmation (Java, C, C++, Cobol, etc.) qui peut être compilé ou interprété. Le terme XMI est utilisé lorsque le résultat de la transformation est un modèle dans la forme XMI (XML Metadata Interchange), qui peut être chargé dans de nombreux outils de modélisation.

Outil	Transformation				Mise en correspondance
	Langage	Expression Graphique (G) ou Textuel (T)	Modèle généré		
			XMI	Texte	
ACCELEO	QVT, JMI	T	-	Oui	-
AndroMDA	ATL, MofScript	T	Oui	Oui	-
ADT	ATL	T	Oui	Oui	Oui
AToM3	Multi formalism (python)	G		Oui	-
DSL tools	Notation XML	T	Oui	Oui	-
Kermeta	QVT	T	-	Oui	-
ModFact	QVT	T	-	Oui	-
Merlin	QVT, JET	T	Oui	Oui	-
MDA Workbench	QVT	T	-	Oui	-
MOFLON	JMI	G	-	Oui	-
OptimalJ	QVT	T	-	Oui	-
QVT Partners	QVT	T	Oui	Oui	-
SmartQVT	QVT	T	Oui	Oui	-
UMLX	XSLT, QVT	G	Oui	Oui	-

TABLEAU 6. OUTILS EN TERME DE TRANSFORMATION ET MISE EN CORRESPONDANCE

1.3.2.3. Les Outils en terme d'interopérabilité

Les outils étudiés offrent de bonnes solutions pour la méta-modélisation et les transformations. En complément, il peut être nécessaire de réutiliser des modèles, des méta-modèles ou de transformations dans un autre outil. Donc, il est très important de connaître la capacité d'un outil à interagir avec d'autres outils.

Le tableau 7 illustre qu'une grande partie des outils sont centrés sur la spécification MOF. Plusieurs formats de mise en œuvre sont proposés pour le MOF : eCore, MDR, KM3, DSL et CWM.

En termes d'interopérabilité, Eclipse propose de facto un mécanisme pour le stockage et la récupération des modèles basés sur XML. Ainsi, la grande majorité des outils sont basés sur Eclipse et peuvent interagir avec d'autres outils Eclipse.

Outil	Repository			Interopérabilité avec autres outils
	Méta-modélisation	Transformation	Contraints	
ACCELEO	DSL, MDR, ECORE	-	XMI	Eclipse, Netbeans
AndroMDA	MOF, DSL	-	XMI	Eclipse
ADT	DSL, KM3, MDR, ECORE	Text (ATL)	XMI	Eclipse, Netbeans
AToM3	Proprietary graphical multi - formalism			-
DSL tools	DSL - Proprietary notation	XML / XMI	-	Eclipse, Netbeans
Kermeta	ECORE	Text (QVT)	XMI	Eclipse
ModFact	ECORE	XMI	XMI	Eclipse
Merlin	ECORE	Text (QVT)	XMI	Eclipse
MDA Workbench	ECORE	XMI	XMI	Eclipse
MOFLON	ECORE	-	XMI	Eclipse
OptimalJ	CWM, ECORE	XMI	XMI	Eclipse
QVT Partners	ECORE	Text (QVT)	XMI	Eclipse
SmartQVT	ECORE	Text (QVT)	XMI	Eclipse
UMLX	ECORE	XMI, XSLT	XMI, XSLT	Eclipse

TABLEAU 7. OUTILS EN TERME D'INTEROPERABILITE

1.4. Synthèse

Nous avons étudié les éléments fondamentaux de la modélisation, l'Ingénierie Dirigée par les Modèles (L'IDM) et les outils de gestion de modèles. Nous nous intéressons plus particulièrement aux différents aspects concernant la réutilisation et la gestion de modèles qui nous aideront par la suite à positionner notre approche orientée services pour la réutilisation de processus et d'outils de modélisation.

L'IDM vise à fournir un cadre pour le développement de logiciels, qui est vu comme une série de raffinements entre modèles du même système à divers niveaux d'abstraction. Cette approche part de l'idée que les systèmes logiciels peuvent être décrits par un ou plusieurs modèles, chacun présentant un point de vue du système considéré. Cela augmente le niveau d'abstraction et sert de base pour la communication entre les différents acteurs qui s'approprient ces différents vues du

système. Le développement est automatisé par cette série de raffinements, appelés transformations de modèles. Au cœur de l'IDM se trouve le concept de modèle, mais aussi les concepts de langage, de méta-modèle, de transformation de modèles, de vérification de la cohérence, etc.

Il existe un grand nombre d'outils qui aident à la résolution de plusieurs problèmes comme l'édition de modèles.

Nous avons détaillé dans cette section les différents types d'outils de modélisation. Des solutions étudiées offrent des environnements de modélisation qui dépassent le cadre de la modélisation. Les produits offrent des fonctionnalités de méta-modélisation, de réalisation de transformations et de vérification de cohérence.

L'étude comparative de ces outils nous a permis de comprendre des aspects technologiques et fonctionnels (tels que : plateforme, opérations sur les modèles, stockage et langage de modélisation, etc.) qui doivent être pris en considération pour la construction d'environnements de modélisation adaptés aux besoins des concepteurs.

2. L'INGENIERIE DE BESOINS

2.1. Introduction

Le terme d'ingénierie des besoins fut introduit par J. Hagelstein [Hagelstein, 1988] et E. Dubois [Dubois *et al.*, 1989] pour désigner la partie du développement des systèmes d'information qui concerne l'investigation des problèmes et des besoins des utilisateurs et le développement des spécifications du futur système.

L'ingénierie des Besoins est une phase importante dans un projet de développement de système. L'ingénierie des besoins aide à exprimer ce que le système doit faire et non comment il doit le faire. De plus, afin d'avoir une spécification des besoins complète, il faut aussi raisonner sur le pourquoi : « Pourquoi est-ce que le système doit être construit ? ».

La grande majorité des approches d'ingénierie des besoins se basent sur un des ces deux concepts : scénario ou but. Un **scénario** décrit « un ensemble fini d'interactions entre des agents dont le comportement est dirigé par des objectifs distincts » [Ben Achour, 1999]. Chaque interaction implique deux agents et permet d'atteindre un but donné. La collection d'actions décrite dans un scénario correspond à un chemin unique parmi l'ensemble des comportements que les acteurs impliqués dans le scénario peuvent avoir. Par ailleurs, tout scénario peut être caractérisé par une situation initiale, un résultat, et un ensemble de conditions susceptibles d'infléchir le comportement des agents.

Un **but** peut être défini comme « quelque chose que quelqu'un espère réaliser dans le futur » [Ben Achour, 1999]. Un but reflète une intention, un objectif que l'on cherche à atteindre sans pour autant dire comment l'atteindre. Il est associé à un résultat souhaité matérialisé par un ensemble d'états d'objets. Tout but peut être exprimé sous la forme d'une clause composée d'un verbe principal et de paramètres jouant chacun un rôle différent par rapport au but.

Il est possible de combiner les deux concepts. Tout but peut être attaché à un et un seul scénario (qui l'illustre), et inversement, tout scénario illustre un et un seul but. Une paire but-scénario est appelée fragment de besoin, et spécifie une partie de la spécification du système qui est envisagée/désirée. Les fragments de besoins peuvent être classés à divers niveaux d'abstraction : le niveau contextuel auquel sont identifiés les services rendus par le système dans le contexte de l'organisation dans lequel il est sensé fonctionner, le niveau d'interaction dans lequel sont décrits le comportement du système et les interactions qu'il doit réaliser avec ses utilisateurs et autres agents, et le niveau physique auquel sont décrits les comportements des objets internes du système dans le cadre de la réalisation de ses diverses fonctions [Ben Achour, 1999].

Il y a plusieurs approches dans le domaine de l'ingénierie des besoins. Ces approches sont catégorisées en trois types :

- Les approches à base de scénarios : en particulier, approche de Jacobson fondée sur les cas d'utilisation [Jackson, 1995].
- Les approches dirigées par les buts : KAOS [Lamsweerde *et al.*, 1998], I* Framework [Yu, 1997], NATURE [Jarke *et al.*, 1995], MAP [Rolland *et al.*, 2000].
- Les approches couplant l'ingénierie des besoins par buts et scénarios : CREWS l'Ecritoire [Maiden, 1998], [Ben Achour, 1999], [Tawbi, 2001], Arbres de tâches [Normand, 1992].

Les approches dirigées par les buts offrent, généralement, une modélisation des buts par décomposition sous la forme d'arbres et/ou [Bubenko *et al.*, 1994]. Les approches dirigées par les scénarios dérivent les modèles conceptuels à partir des scénarios [Dano *et al.*, 1997] et sont utilisés pour raisonner sur des choix de conception [Caroll, 1995]. Finalement, dans les approches basées sur le couplage buts-scénarios [Maiden, 1998], [Tawbi, 2001] les scénarios sont utilisés pour décrire différentes manières possibles d'atteindre le même but, donc, les buts sont opérationnalisés par les scénarios [Ben Achour, 1999].

Cette section présente la structure d'un scénario et d'un but. Ensuite, nous présenterons l'approche MAP, une démarche de modélisation dirigée par les buts. Finalement, nous présenterons la notion d'arbre de tâches, une démarche de modélisation basée sur le couplage buts/scénarios.

2.2. Les scénarios

La notion de scénario a été introduite pour fournir un moyen de décrire les perspectives des utilisateurs, et leur façon d'utiliser l'application. Un scénario doit être décrit d'une manière compréhensible par tous les participants à la conception, pour faciliter la communication entre les différents types d'utilisateurs et les concepteurs de l'application.

Généralement, un scénario est écrit en langage naturel. Il est constitué d'une ou plusieurs actions. Dans un scénario, une combinaison des actions décrit un chemin unique menant d'un état initial à un état final des agents du système. La structure linguistique d'un scénario a été définie par [Ben Achour, 1999] et étendue dans [Tawbi, 2001]. La figure 10 présente la structure linguistique d'un scénario selon [Tawbi, 2001]. Dans les paragraphes ci-dessous, nous détaillons les concepts présentés à la figure 10.

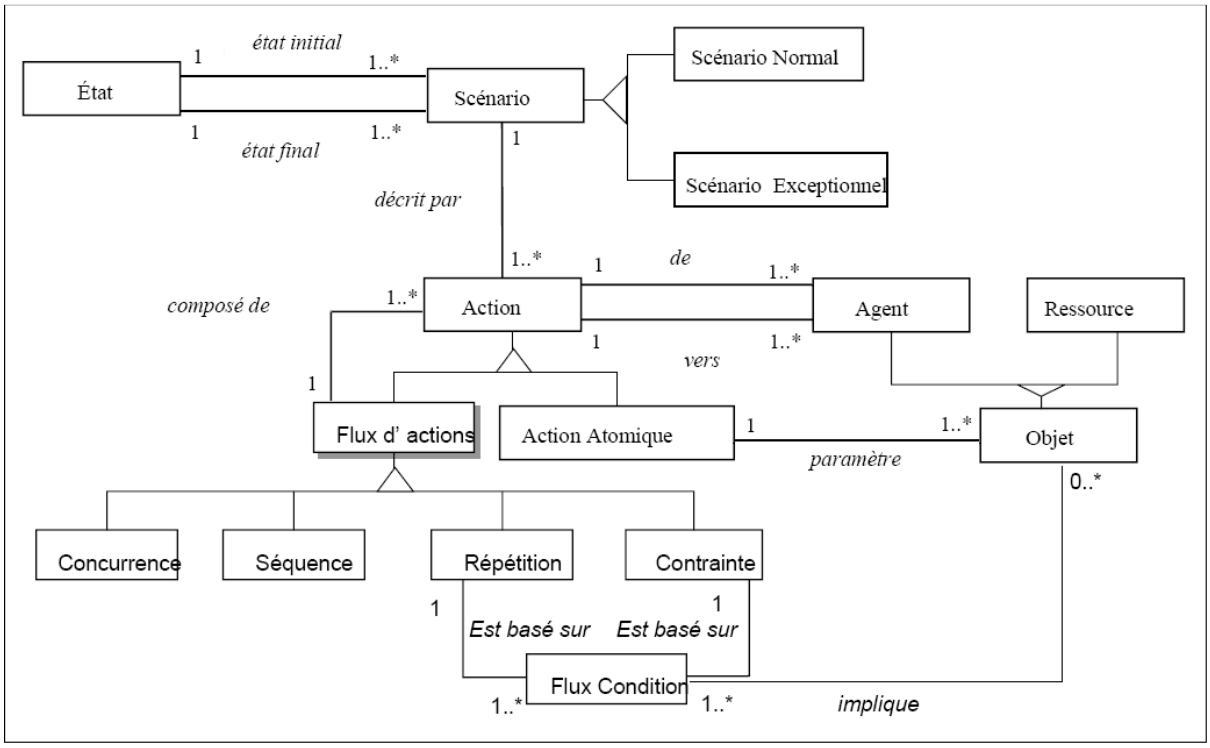


FIGURE 10. LA STRUCTURE D'UN SCENARIO [TAWBI, 2001]

Un scénario est caractérisé par deux états : Initial et final. **Un état initial** attaché à un scénario définit les pré-conditions nécessaires au déclenchement de celui-ci alors qu'un **état final** définit les états atteints à la fin du scénario. Par exemple, le scénario : « retirer de l'argent à partir du guichet automatique bancaire (GAB) dans le cas normal » ne peut se déclencher que si les deux états « l'utilisateur a une carte » et « le GAB est prêt » sont vrais. Le déclenchement de ce scénario se termine donc avec les trois états finaux : « l'utilisateur a une carte », « l'utilisateur a de l'argent » et « le GAB est prêt ».

La figure 9 spécifie deux types des scénarios : les scénarios normaux et les scénarios exceptionnels. Un **scénario normal** permet d'atteindre le but, alors qu'un **scénario exceptionnel** se termine par la non satisfaction du but. Par exemple, le scénario associé au but « retirer de l'argent à partir du GAB en saisissant trois fois un code erroné » est du type exceptionnel avec les états finaux : « l'utilisateur n'a pas une carte », « l'utilisateur n'a pas d'argent » et « le GAB est prêt ».

Les actions sont de deux types : atomique et flux. Une **action atomique** est une interaction entre deux agents affectant un objet. Un **agent** et un **objet** peuvent figurer dans plusieurs interactions différentes. La clause « l'utilisateur insère une carte dans le GAB » est un exemple d'action atomique, le paramètre de cette action étant « une carte », et c'est une action de communication impliquant les deux agents : « l'utilisateur » et « le GAB ».

Un **flux d'actions** permet de définir l'ordonnancement entre les interactions dans un scénario. Il est composé de plusieurs actions. Les flux d'actions sont classés en quatre types : séquence, concurrence, répétition ou contrainte.

Une **séquence** est composée de deux actions, la deuxième action se déroule suite à la première. Par exemple : « si la carte est valide, le guichet automatique bancaire affiche un message demandant le code de la carte à l'utilisateur ».

Contrairement à une séquence, il n'y a aucun ordre spécifique entre deux actions concurrentes. Donc, deux actions en concurrence ne commencent pas et ne se terminent pas nécessairement au même moment.

Finalement, les deux types de flux (**contrainte** et **répétition**) sont associés à des conditions du flux caractérisant la progression des actions dans un scénario. Par exemple : « si le code est valide, un message demandant le montant est affiché par le guichet automatique bancaire à l'utilisateur », donc, la condition « si le code est valide » identifie un cas unique d'utilisation du guichet automatique bancaire décrit par un scénario.

2.3. Les buts

Un but est défini comme un objectif à atteindre dans le futur système [Plih *et al.*, 1998]. Dans la plupart des approches dirigées par les buts, un but est une proposition en langage naturel formalisée selon une structure inspirée de [Prat, 1999]. Prat définit un but comme une structure composée d'un verbe suivi d'un ensemble de paramètres. Chaque paramètre joue un rôle différent par rapport au verbe. La figure 11 montre la structure d'un but selon la proposition de Prat.

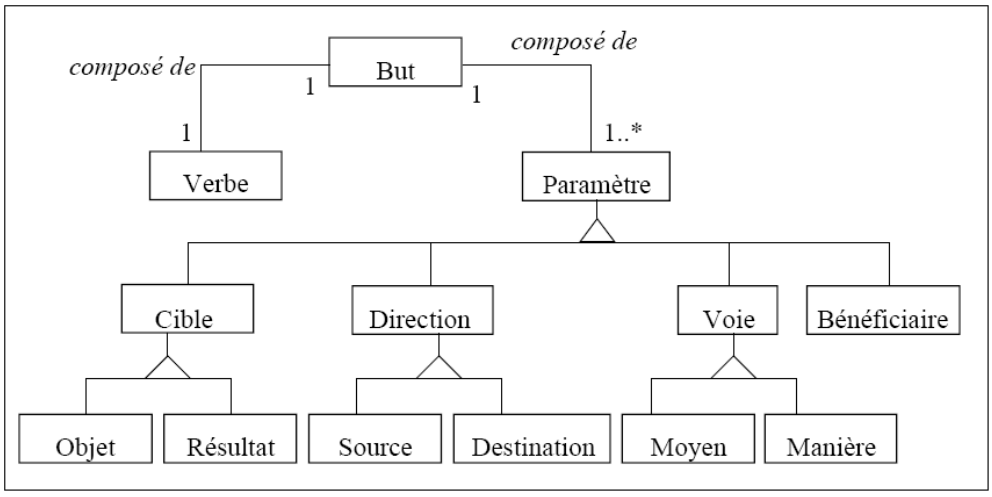


FIGURE 11. LA STRUCTURE DU BUT [PRAT, 1999]

Le paramètre cible concerne la ou les entités affectées par le but. Il y a deux types de cibles : l'objet et le résultat. **L'objet** est un élément qui existe avant la réalisation du but et peut éventuellement être modifié ou supprimé par celui-ci, alors que **le résultat** est l'entité qui résulte de la réalisation du but désigné par le verbe.

La direction identifie respectivement l'endroit initial et final de l'objet. **La source** est le point du départ du but (source d'information ou lieu physique) et la destination est son point d'arrivée.

La voie est spécialisée par les deux paramètres : manière et moyen. **La manière** spécifie la façon d'atteindre le but et **le moyen** est l'entité ou l'outil, par lequel le but est atteint.

Finalement, **le bénéficiaire** est la personne ou le groupe en faveur de qui le but doit être atteint. Le tableau 8 montre deux buts formalisés selon cette structure. Le deuxième exemple montre que cette structure peut être appliquée d'une manière récursive sur le paramètre manière.

But	Formalisation
Fournir des services de retrait bancaires à nos clients au moyen d'un guichet automatique bancaire	(Fournir)*Verbe* (des services de retrait bancaires)*Résultat* à (nos clients)*Bénéficiaire* au (moyen d'un guichet automatique bancaire)*Moyen*
Fidéliser nos clients en fournissant des services de retrait d'argent au moyen d'un guichet automatique bancaire	(Fidéliser)*Verbe* (nos clients)*Bénéficiaire* en ((fournissant)*Verbe* (des services de retrait d'argent)*Résultat* au (moyen d'un guichet automatique bancaire)*Moyen*)*Manière*

TABLEAU 8. BUTS FORMALISES SELON LA STRUCTURE DE PRAT [PRAT, 1999]

2.4. Le MAP

Les besoins utilisateurs peuvent être décrits par un modèle orienté buts appelé carte (MAP) [Rolland *et al.*, 2000]. Le modèle MAP est un système de représentation qui a été, à l'origine, développé pour la modélisation intentionnelle des processus de développement. La MAP exprime les besoins des utilisateurs par des buts et des stratégies pour les atteindre.

Le système de représentation de MAP utilise le concept d'intention et se différencie des autres modèles par l'introduction du concept de stratégie pour atteindre une intention. La MAP est graphiquement représentée comme un graphe orienté et étiqueté avec un « Démarrer » et un « Arrêter » (Figure 13). Les intentions sont les nœuds et les stratégies sont les arcs. Une section est ainsi représentée par deux nœuds reliés par une flèche.

Dans ce système de représentation, une intention est ce qu'on cherche à atteindre. Une stratégie est une manière de réaliser une intention. La figure 12 présente le méta-modèle du MAP, ses concepts clés et leurs relations en utilisant la notation UML. Le méta-modèle du MAP montre qu'une carte est composée de deux ou plusieurs sections. **Une section** est une agrégation d'une **intention source**, d'une **intention cible** et d'une **stratégie**. Chaque section peut être utilisée pour réaliser une intention cible, une fois que l'intention source a été atteinte.

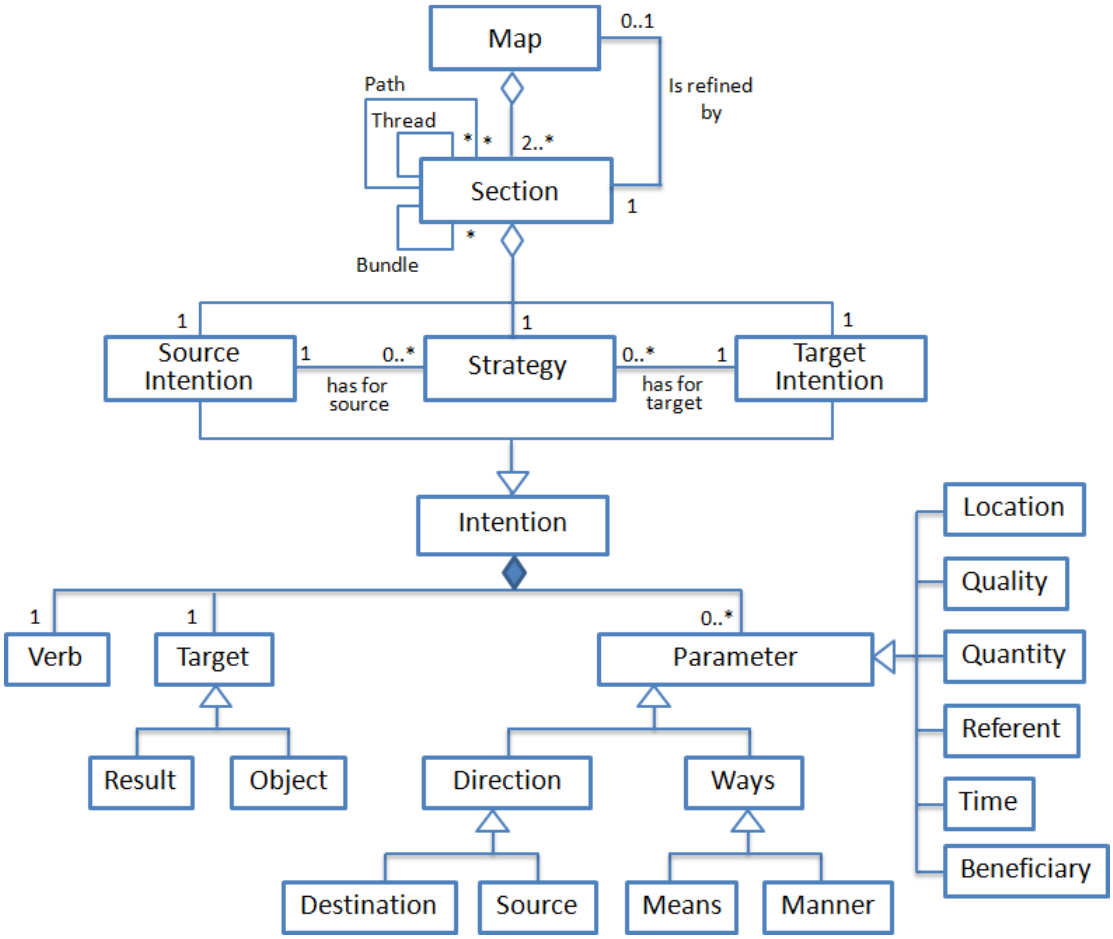


FIGURE 12. META-MODELE DE LA CARTE (MAP) [ROLLAND, 2007]

Une intention est un but qui peut être réalisé par l'exécution d'un processus. Selon Jackson [Jackson, 1995] une intention est une déclaration « optative » qui exprime ce que l'on veut, un état ou un résultat que l'on cherche à atteindre. Par exemple, comme le montre la figure 13, « Découvrir un but » est une intention dans le processus de conception des besoins fonctionnels d'un système d'information. Chaque carte possède deux intentions particulières : Démarrer et Arrêter pour respectivement commencer et terminer l'exécution de la carte. De plus, une intention ne peut apparaître qu'une seule fois dans la même carte.

Une intention (représentée dans la MAP de la figure 13 par un nœud) capture la notion de tâche que l'utilisateur projette d'accomplir à un moment donné du processus. Une intention est une expression en langage naturel initialement développé par [Prat *et al.*, 1997], [Prat *et al.*, 1999]. Cette approche inspirée de la grammaire des cas de Fillmore [Fillmore, 1968] et des extensions de Dik [Dik, 1989] se fonde sur le fait que la sémantique d'une intention est capturée par un verbe auquel on associe plusieurs paramètres qui jouent des rôles spécifiques par rapport au verbe [Rolland, 2007]. The structure linguistique d'une intention est la suivante :

Intention : Verb <Target> [<Parameter>]* -- * éléments répétitifs, [] éléments facultatifs

Le tableau 9 résume les paramètres considérés par le méta-modèle MAP (figure 12). En complément à la structure linguistique, Prat [Prat, 1997] propose une classification des verbes et définit un cadre indicatif pour cataloguer les paramètres obligatoires et facultatifs. Par exemple, le cadre indicatif du verbe « demeurer » (rester dans tel état) est « demeurer [Qual, (Ref), (Loc), (Time)] ». Ce cadre signifie que le verbe « demeurer » est toujours suivi d'une qualité et éventuellement suivi d'un référent, un emplacement et un point temporel.

Paramètre	Description
Target	Désigne une entité concernée par l'objectif. Deux types de target sont considérés : l'objet et le résultat.
Object	Un objet (Obj) existe avant que l'objectif soit atteint.
Result	Le résultat (Res) peuvent être de deux types : a) une entité qui n'existe pas avant l'objectif est atteint ; b) une entité abstraite, laquelle existe mais elle se concrétise suite à la réalisation des objectifs.
Source	Les deux types de direction (Dir), à savoir la source (So) et la destination (Dest) correspondent à l'emplacement initial et final des objets qui doivent être communiqués.
Destination	
Means	Le moyen (Mea) désigne comment un objectif doit être effectué.
Manner	La manière (Man) définit la façon de réaliser le but.
Beneficiary	Le bénéficiaire (Ben) est la personne (ou le groupe de personnes) en faveur de laquelle l'objectif est atteint.
Referent	Le Référent (Ref) correspond à l'élément pour lequel une action est effectuée, ou un état est obtenu ou maintenu.
Quality	La qualité (Qual) définit une propriété qui doit être atteinte ou préservée.
Location	La localisation (Loc) situe le but dans l'espace. Par exemple : Faire (une réservation)*Res* (dans une agence de voyage)*Loc*
Time	Le temps (Time) situe l'objectif à terme. Par exemple : Supprimer (l'option de réserver)*Obj* (après 8 jours)*Time*
Quantity	La quantité (Quan) évalue quantitativement une évolution qui devrait se produire ; Par exemple : Réduire(prix)*Obj* (dans un 3%)*Quan*

TABLEAU 9. PARAMETRES D'UNE INTENTION [ROLLAND, 2007]

Une stratégie (représentée dans la MAP de la figure 13 par une flèche) est une manière ou un moyen de réaliser une intention cible à partir d'une intention source. Dans une MAP, les stratégies correspondent aux différentes façons de réaliser les intentions. Une stratégie s'associe à l'intention auquel elle s'applique. Elle a pour objectif principal d'extérioriser la façon d'atteindre cette intention puisqu'elle permet de distinguer l'intention et la façon de la réaliser. En outre, fournir plusieurs stratégies pour atteindre la même intention permet de suggérer des façons alternatives de réaliser cette intention. Dans l'exemple de la figure 13, deux stratégies différentes sont proposées pour satisfaire l'intention « Ecrire un scénario » en partant de l'intention « Découvrir un but » : « stratégie de prose libre » et « stratégie de structure prédéfinie ».

Une même stratégie peut apparaître dans une ou plusieurs sections d'une même MAP. Deux sections reposant sur une même stratégie peuvent donc être similaires ou différentes selon les processus qu'elles référencent. Finalement, les stratégies dans le système de représentation MAP fournissent les moyens pour capturer la variabilité dans l'accomplissement de l'intention.

Une section est un triplet <intention source, intention cible, stratégie>. Une section exprime la réalisation d'une intention cible en utilisant la stratégie une fois que l'intention source a été réalisée. Par exemple, dans la figure 13 les intentions : « Ecrire un scénario » et « Conceptualiser un scénario » et la stratégie : « avec un support automatique » définissent la section : < Ecrire un scénario, Conceptualiser un scénario, avec un support automatique >.

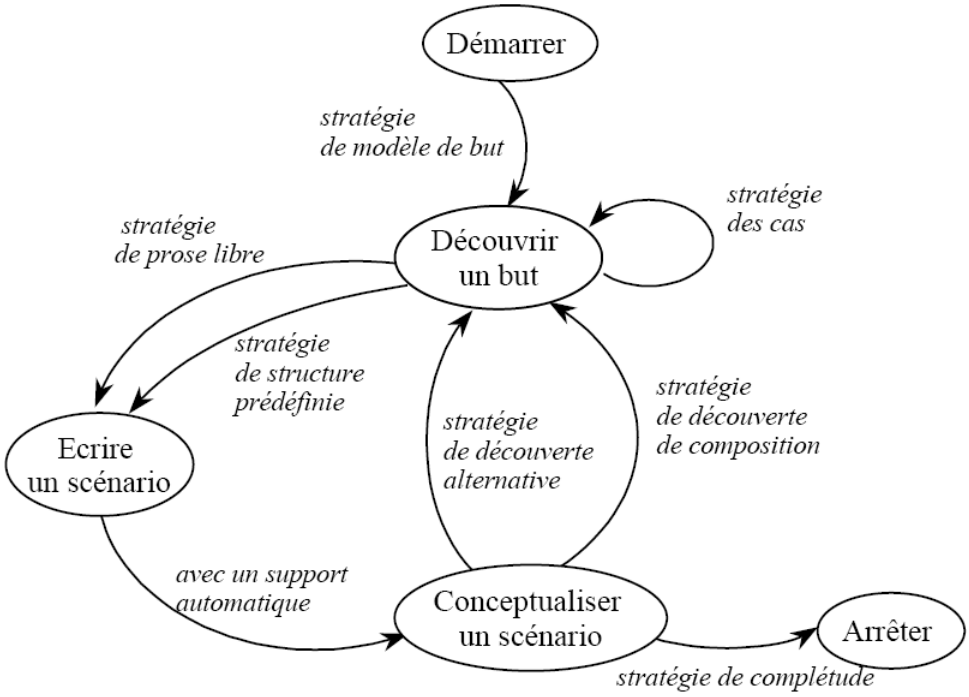


FIGURE 13. EXEMPLE D'UNE CARTE [TAWBI, 2001]

2.5. Les arbres de tâches

Nous rappelons que les arbres de tâches correspondent une démarche de modélisation basée sur le couplage buts/scénarios.

D'une manière générale, une tâche désigne un « travail déterminé qu'on doit exécuter » [Le petit Robert, 2010]. Pourtant en IHM, la notion de tâche contient aussi le but à atteindre. Normand voit dans la tâche « un objectif à atteindre par l'utilisateur à l'aide d'un système interactif » [Normand, 1992]. De plus, il affirme qu'une tâche est un but que l'utilisateur vise à atteindre assorti d'une procédure (ou) plan qui décrit les moyens pour atteindre ce but [Normand, 1992]. Dans ce contexte, un but est un état du système que l'utilisateur souhaite obtenir et une procédure est le composant

exécutif d'une tâche : un ensemble d'opérations organisé par des relations temporelles et structurelles. Les relations temporelles entre les opérateurs d'une procédure traduisent la séquentialité, l'interruptibilité, voire le parallélisme. Les relations structurelles servent à exprimer la composition logique des opérations ou la possibilité de choix.

Une tâche est une activité (considérée comme) nécessaire, ou utilisée pour atteindre un objectif en utilisant un moyen donné [Diaper et *al.*, 2003]. Une tâche est en général décomposable en sous-tâches, jusqu'au niveau des actions.

Une action désigne une opération terminale ; elle intervient dans l'accomplissement d'un but terminal. Un but terminal et les actions qui permettent de l'atteindre définissent une tâche élémentaire. Une tâche élémentaire est une tâche n'impliquant pas de résolution de problème ou de structure de contrôle (alternatives, répétition,…). Elle est généralement focalisée sur un ou des objets.

Une procédure élémentaire est la procédure associée à une tâche élémentaire.

Une tâche qui n'est pas élémentaire est dite composée ou encore : une tâche composée est une tâche dont la description inclut des sous-tâches.

La figure 14 montre une représentation hiérarchique d'un arbre de tâches.

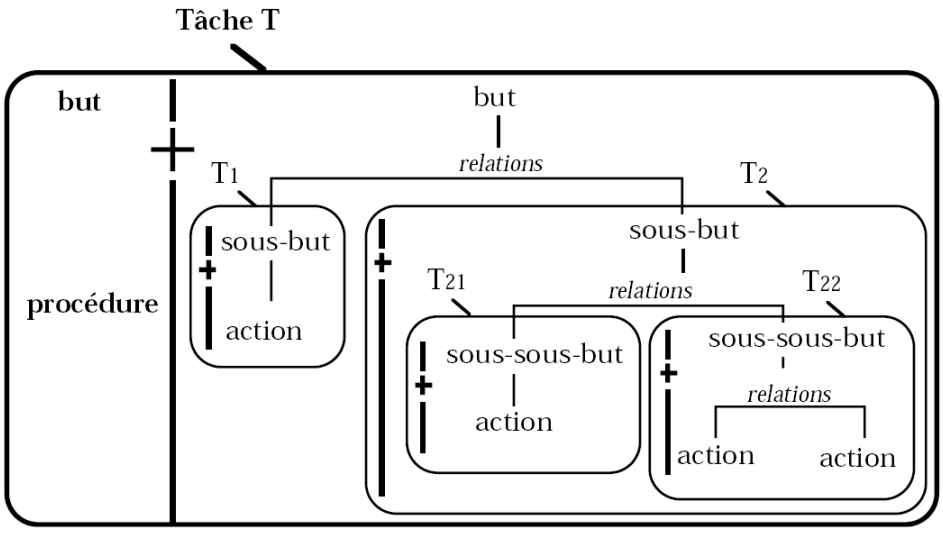


FIGURE 14. REPRESENTATION D'UN ARBRE DE TACHE [NORMAND, 1992]

Les feuilles de l'arbre constituent les tâches élémentaires. Un rectangle encapsule une tâche. Les relations sont représentées par une structure arborescente. Dans l'exemple, le but est atteint par l'accomplissement de deux sous-tâches : T1 et T2. La tâche T1correspond un sous-but qui peut être atteint par une action. La tâche T2 correspond à un sous-but qui doit être décomposé plus avant. Les tâches : T1, T21 et T22 sont des tâches élémentaires, et finalement les tâches T et T2 sont des tâches composées.

Un bref historique des modèles de tâches est proposé par Van Wellie [Wellie, 2001]. On y apprend que les modèles de tâches ont été introduits par Annett [Annett *et al.*, 1967].

Les modèles de tâches peuvent être utilisées dans l'ingénierie des besoins mais également dans les phases d'analyse et de conception. Parmi les modèles de tâches les plus utilisés selon [Limbourg *et al.*, 2001], nous distinguons :

1. Le modèle GTA (Groupware Task Analysis) permet de modéliser la complexité des tâches dans un environnement coopératif ;
2. Le modèle TKS (Task Knowledge Structure) [Johnson *et al.*, 1991] est une représentation conceptuelle de la connaissance qu'a un utilisateur d'une tâche particulière. Une tâche est ici déterminée selon le rôle de la personne ;
3. Le modèle CTT (Concurrent Task Trees) [Paternò, 1997] et le modèle MAD (Méthode Analytique de Description de tâches) [Pierret-Golbreich *et al.*, 1989] permettent de décrire différents types de tâches et les relations entre tâches de même niveau en utilisant des opérateurs.

Le but de l'analyse de tâches est de mettre en évidence et de décrire les tâches à effectuer pour accomplir un travail. L'analyse des tâches consiste à relier objectifs, tâches et actions. Il s'agit de comprendre les objectifs des utilisateurs et de comprendre comment ils passent des objectifs aux tâches puis aux actions. L'enchainement de tâches nous permet identifier, comprendre et étudier les scenarios. Une analyse très fine des tâches peut également servir à prédire ou à expliquer les performances d'un utilisateur dans un environnement donné.

2.6. Synthèse

Dans cette section, nous avons présenté quelques concepts issus de l'ingénierie de besoins en particulier les scénarios, stratégies, tâches, buts et intentions.

Nous avons proposé une catégorisation des approches : les approches à base de scénarios, les approches dirigées par les buts et les approches couplant les objectifs et scénarios. Nous avons également étudié la structure d'un scénario, d'un but, la formalisation d'une intention et les arbres de tâches.

L'étude des arbres de tâches nous permettra de clarifier dans notre étude les relations entre les objectifs, les tâches et les actions des concepteurs de modèles.

Nous avons enfin étudié le modèle de la carte MAP qui sera utile pour notre étude des intentions des concepteurs de modèles. Cette approche permet d'exprimer les besoins des utilisateurs par des buts et des stratégies pour les atteindre.

Les concepts étudiés dans cette section constituent le point de départ pour définir et organiser les besoins des concepteurs de modèles, ce qui nous permettra d'établir un niveau d'abstraction intentionnel pour faciliter la réutilisation de processus et des outils de modélisation.

3. L'INGENIERIE DE METHODES

3.1. Introduction

Le terme méthode vient du grec « ***methodos*** » qui signifie « moyen d'investigation ». Harmsen [Harmsen, 1997] détermine ce que sont ces moyens d'investigation : « une collection de procédures, de techniques, de descriptions de produits et d'outils pour le support effectif, efficace et consistant du processus d'ingénierie d'un SI ».

D'autres définitions de la notion de méthode ont été proposées dans : [Lyytinen *et al.*, 1989], [Seligmann *et al.*, 1989], [Brinkkemper, 1990], [Smolander *et al.*, 1991], [Kronlof, 1993] [Wynekoop *et al.*, 1993], [Harmsen *et al.*, 1994], [Prakash, 1994], [Brinkkemper,1996]. La plupart d'entre elles convergent vers l'idée qu'une méthode est basée sur des modèles (systèmes de concepts) pour décrire le produit et les règles de conduite méthodologique pour façonner un produit de qualité avec une efficacité raisonnable. Cette acceptation est synthétisée par G. Booch [Booch, 1991] qui définit une méthode comme « un processus rigoureux permettant de générer un ensemble de modèles qui décrit divers aspects d'un logiciel en cours de construction en utilisant une certaine notation bien définie ». En d'autres termes, une méthode traite les deux aspects de l'ingénierie, le produit et le processus, et comporte deux éléments : un ou plusieurs modèles de produit et un ou plusieurs modèles de processus [Prakash, 1999], [Seligmann *et al.*, 1989].

La figure 15 montre la vue générale d'une méthode. Chaque modèle de produit représente la structure d'une classe de produits obtenus en appliquant la méthode dans différentes applications tandis que le modèle de processus représente la démarche à suivre pour obtenir le produit cible.

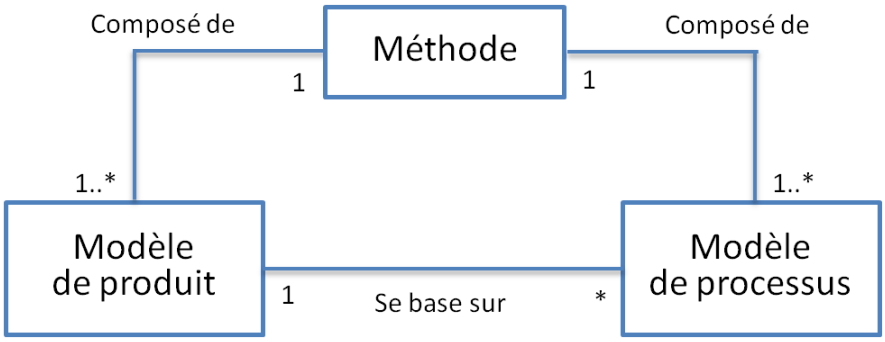


FIGURE 15. STRUCTURE D'UNE METHODE [RALYTE, 2001]

Cette section commence par la présentation des concepts de base de l'ingénierie de méthodes, puis nous détaillerons l'ingénierie de méthodes. Nous présentons également l'ingénierie de méthodes situationnelles, plus particulièrement nous aborderons les aspects concernant la réutilisation et la

gestion de modèles pour les méthodes situationnelles. Finalement, nous présenterons les approches actuellement prédominantes en ingénierie de méthodes situationnelles.

3.1.1. Les quatre niveaux de modélisation de produits et de processus

Une méthode est composée d'un modèle de processus et d'un ou plusieurs méta-modèles produit. Selon Rieu [Rieu, 1999], l'ensemble des modèles de produits ou de processus d'une méthode permettent de représenter le système sous différents aspects (statique, dynamique, fonctionnel) et à différents niveaux d'abstraction (expression des besoins, analyse, conception). La figure 16 présente les quatre niveaux de modélisation pour les produits et les processus tels que définis par l'OMG [OMG, 2006]. Les produits représentent le résultat à atteindre et les processus le chemin à parcourir pour atteindre le résultat.

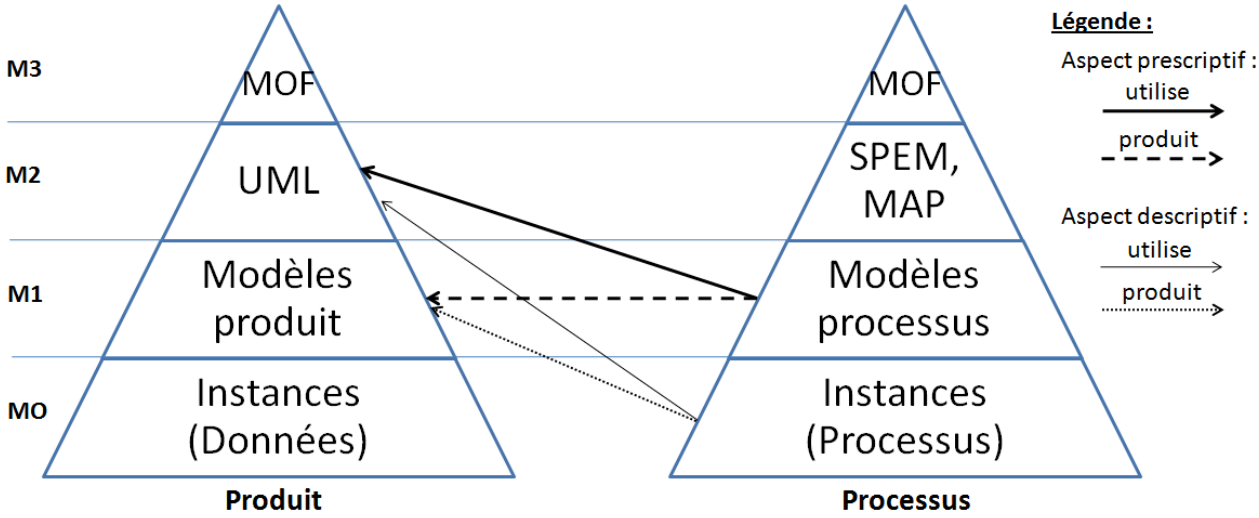


FIGURE 16. LES 4 NIVEAUX DE MODELISATION PRODUIT ET PROCESSUS [HUG, 2009]

Du côté des produits, le premier niveau M0 représente le niveau des instances, c'est-à-dire les objets du monde réel représentés dans le système. Le niveau M1 est le niveau des modèles produit, ce sont des diagrammes de classes par exemple. Le niveau M2 est le niveau du méta-modèle. Le terme méta-modèle de produits est généralement utilisé pour désigner un formalisme permettant de décrire un produit, le plus courant méta-modèle de produit est UML [OMG, 2007a], [OMG, 2007b]. UML définit les concepts qui pourront être instanciés pour créer des modèles composés de classes, d'associations et d'attributs, entre autres. Le niveau M3 est le niveau de la méta-méta-modélisation. Par exemple, l'OMG a défini le MOF [OMG, 2006] mais il en existe d'autres comme Ecore [Ecore, 2009] [Budisnsky *et al.*, 2003].

Du côté des processus, le niveau M0 représente l'exécution des processus réels. Par exemple, un processus d'ingénierie de SI comme RUP. Le niveau M1 représente les modèles de processus à

utiliser, comme le modèle de processus du RUP par exemple. Le niveau M2 est celui des méta-modèles de processus qui représentent les modèles qui permettront de définir des modèles de processus, comme le méta-modèle de l'OMG, SPEM [OMG, 2008] ou le méta-modèle de la MAP [Rolland *et al.*, 1999]. Enfin, le niveau M3 bénéficie du méta-méta-modèle du MOF par exemple.

Deux aspects sont à considérer. D'une part, les modèles de processus prescrivent l'utilisation de méta-modèles produit comme UML pour réaliser des modèles produit (en gras sur la figure). Le modèle de processus représente donc la démarche à suivre a priori. D'autre part, un processus d'ingénierie de systèmes d'information réel (instance), utilise les méta-modèles produit pour produire des modèles produit : cet aspect du processus est descriptif, car on ne fait que modéliser la trace de ce qui se passe réellement. Ainsi, RUP est composé du modèle de processus du RUP et le méta-modèle produit utilisé est UML.

3.1.2. Le modèle de produit

Le produit est le résultat à atteindre lors de l'application d'une méthode [Olle *et al.*, 1992]. Il est exprimé dans les termes d'un modèle de produit.

Le méta-modèle de produit d'une méthode formalise les concepts pour décrire les produits qui résultent de l'application de la méthode. Une méthode peut comporter plusieurs modèles de produit permettant de modéliser différentes facettes d'un SI (statique, dynamique, fonctionnel) [Olle *et al.*, 1992], à différents niveaux de détails (par exemple : classe, paquetage, sous-système) [Muller, 1997], [Fowler, 1997] [UML, 2000] et à différents niveaux d'abstraction (objet, classe, méta-classe) [Shlaer *et al.*, 1988], [Graham, 1994], [Coad *et al.*, 1991].

La figure 17 montre un exemple d'un méta-modèle et d'un modèle de produit. Le méta-modèle contient les classes permettant de décrire les modèles de produits. Le méta-modèle de la figure 17 représente le méta-modèle des arbres de tâches. Dans ce méta-modèle une tâche a deux associations. La première, l'association mère/fille, permet d'exprimer la décomposition. Une tâche est reliée par le lien mère-fille (s'il existe) sans passer par l'intermédiaire d'un opérateur binaire. La deuxième, l'association suivant/précédent, est utilisée pour représenter l'ordre chronologique avec l'opérateur de séquencèrent adéquat. Ce méta-modèle définit la syntaxe des arbres de tâches dont un modèle est donné en figure 17.

Les modèles de produits définissent des produits d'ingénierie de systèmes d'information. Ce modèle de produits représente les activités d'interaction entre un utilisateur et le système dans le but de choisir un service intentionnel.

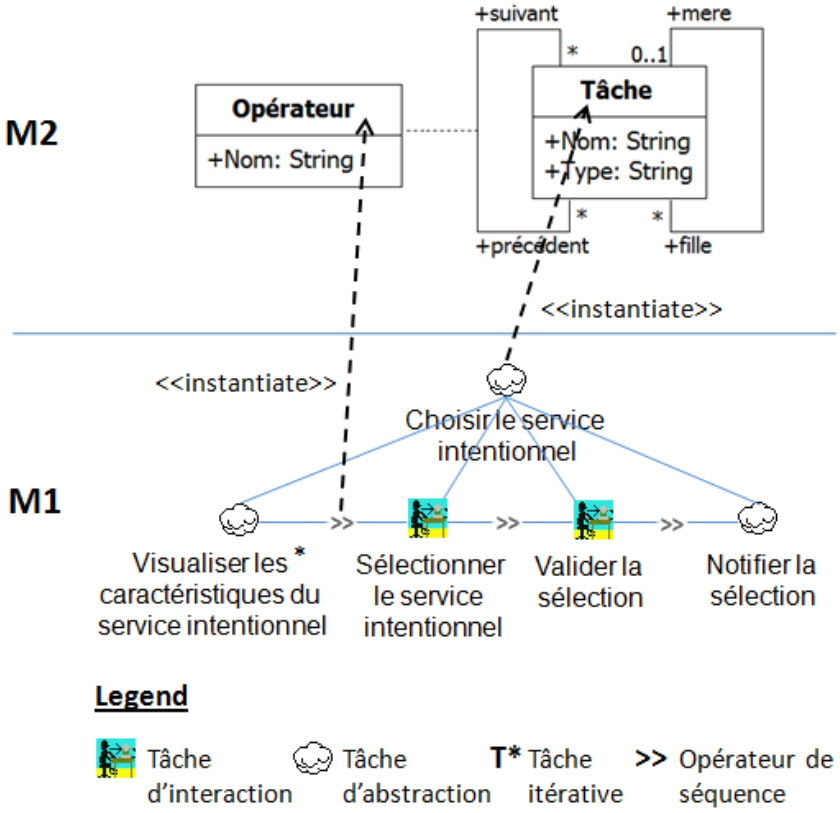


FIGURE 17. EXEMPLE D'UN META-MODELE ET D'UN MODELE DE PRODUIT

3.1.3. Le modèle de processus

Le processus est la route à suivre pour construire un modèle de produit [Olle *et al.*, 1992]. Un modèle de processus est un ensemble d'activités inter-reliées et menées dans le but de définir un modèle de produit [Franckson *et al.*, 1991]. Il est exprimé dans les termes d'un modèle de processus.

Un modèle de processus prescrit une manière de faire, une démarche méthodologique pour atteindre la cible souhaitée. Il décrit à un niveau abstrait et idéal, la façon d'organiser la production du modèle de produit : les étapes, les activités qu'elles comprennent, leur ordonnancement, et parfois les critères pour passer d'une étape à une autre. Le modèle de processus est à la fois guidé et contraint par un méta-modèle en amont qui définit les concepts dont les différents acteurs d'un processus d'ingénierie de SI ont besoin pour définir leurs processus. La figure 18 montre un exemple d'un méta-modèle de processus et d'un modèle de processus de la méthode CREWS L'Écritoire.

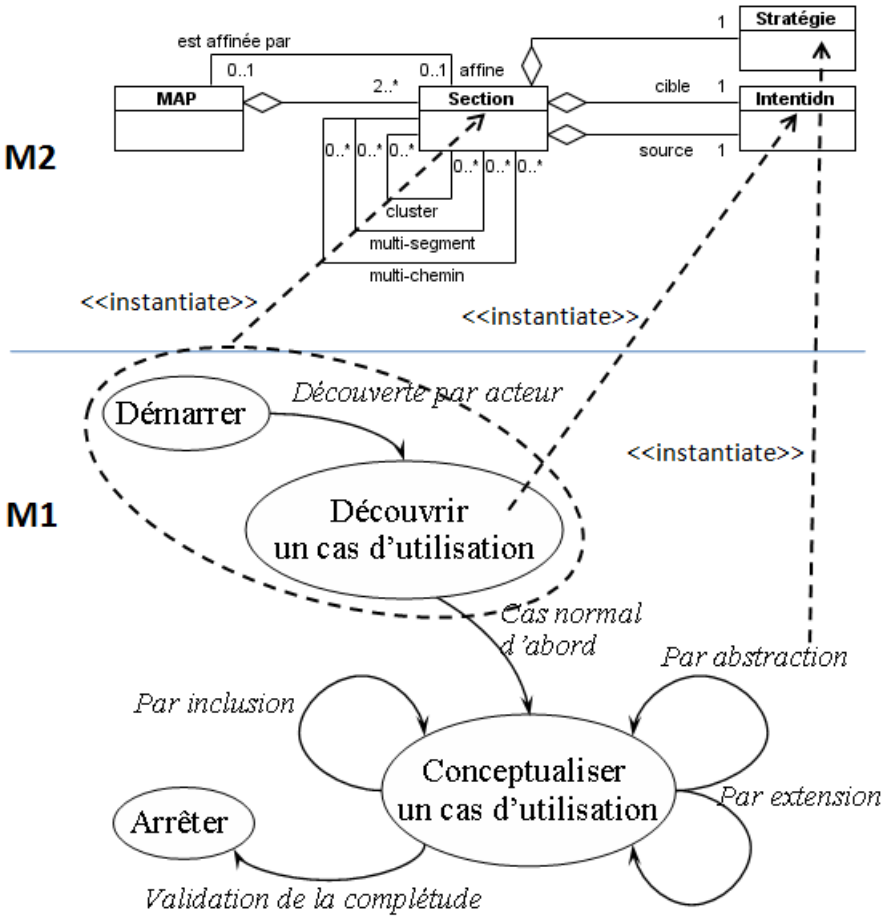


FIGURE 18. EXEMPLE D'UN META-MODELE ET D'UN MODELE DE PROCESSUS [RALYTE, 2001]

Le formalisme utilisé pour ce modèle de processus est celui de MAP [Rolland *et al.*, 1999]. Les nœuds représentent les intentions de l'ingénieur des besoins à un moment donné du processus et les arcs représentent les stratégies pour atteindre ces intentions. Une section est composée d'une Intention source, d'une Intention cible et d'une stratégie liant l'intention source à l'intention cible.

La Figure 19 montre une catégorisation pour la modélisation des processus d'ingénierie initialement proposée par Dowson [Dowson, 1988], augmentée par Rolland [Rolland, 2005] et organisée de façon chronologique par Hug [Hug, 2009]. Cette classification est basée sur les modèles orienté-activité, orienté-produit, orienté-décision, orienté-contexte et orienté-stratégie.

Période	1970	1980	1990	1995	
Type de modèle	**Activité**	**Produit**	**Décision**	**Contexte**	**Stratégie**
Description	Les modèles orientés activité se focalisent sur les activités exécutées pour élaborer un produit et sur leur ordonnancement	Les modèles orientés produit couplent l'état du produit à l'activité qui génère cet état. Ils visualisent le processus comme un diagramme de transitions d'état.	Les modèles orientés décision perçoivent les transformations successives du produit comme résultats de décisions. Ces modèles se focalisent sur le concept d'intention comme l'extension du concept d'activité.	Les modèles contextuels définissent les processus à travers la combinaison de situations observables avec un ensemble d'intentions (aboutissant à des décisions) spécifiques. Il s'agit de décrire le processus comme étant dépendant à la fois de la situation et de l'intention; en d'autres termes, il dépend du contexte de réalisation.	Le modèle orienté stratégie est une extension du modèle contextuel qui vise à représenter plusieurs démanches dans le même modèle de processus. Il est donc, multi-démarches et prévoit plusieurs chemins possibles pour élaborer le produit. Il est basé sur les notions d'intention d'ingénierie et de stratégies à suivre pour réaliser ces intentions.
Exemples	- **Cascade** [Royce, 1970] - **Modèle en V** [McDermid *et al.*, 1984] - **Spirale** [Boehm, 1988] - **RUP** [Jacobson *et al.*, 1999]	- **Statecharts** [Harel, 1987] - **EPM** [Humphrey *et al.*, 1989] - **ViewPoints** [Finkelstein *et al.*, 1990] - **ESF** [Franckson et al., 1991]	- **IBIS** [Kunz *et al.*, 1970] - **GLBIS** [Potts, 1989] - **DAIDA** [Rose et al., 1991][Jarke *et al.*, 1992]	- **NATURE** [Rolland *et al.*, 1995] - **F3** [Rolland et al., 1994c]	-**MAP** [Rolland et al., 1999]

FIGURE 19. CLASSIFICATION DES MODELES DE PROCESSUS [HUG, 2009]

Après cette explication de la notion de méthode et de ses deux éléments fondamentaux, nous présentons le domaine de l'ingénierie de méthodes dans la section suivante.

3.2. L'ingénierie de méthodes

L'ingénierie de méthodes (**IM**) est une branche du génie logiciel qui a pour but d'aider à la construction de nouvelles méthodes d'ingénierie des systèmes d'information. Un des principes fondamentaux de l'IM est l'optimisation, la réutilisation, la flexibilité et l'adaptabilité de ces méthodes et leur décomposition en parties modulaires [Rolland, 2005].

Brinkkemper [Brinkkemper, 1996] définit l'ingénierie des méthodes comme « **une discipline de conceptualisation, de construction et d'adaptation de méthodes, de techniques et d'outils pour le développement des systèmes d'information** ». Elle traite la définition de nouvelles méthodes d'ingénierie des systèmes d'information.

Plusieurs autres définitions restreignent la notion d'ingénierie des méthodes à la construction de nouvelles méthodes à partir de celles déjà existantes. Par exemple, Punter [Punter *et al.*, 1996] définit l'ingénierie des méthodes comme « **une approche de construction des méthodes combinant différentes (parties de) méthodes pour développer une solution optimale du problème donné** ».

Kumar [Kumar *et al.*, 1992], au contraire, propose une définition plus générale qui n'impose pas l'utilisation des méthodes existantes comme point de départ de l'ingénierie des méthodes. Il définit cette dernière comme « **une proposition pour la conception et le développement d'une méta-méthodologie destinée à la conception des méthodes de développement des systèmes d'information** ».

Dans cette section, nous définissons l'ingénierie de méthodes comme la discipline visant à apporter des solutions efficaces à la construction, l'amélioration et l'évolution des méthodes de développement de SI et des systèmes logiciels qui répondent aux exigences particulières d'une situation d'entreprise. Notre définition présente l'ingénierie de méthodes comme une activité répondant à un besoin, celui de construire une méthode adaptée au contexte particulier d'un projet d'entreprise. Elle s'apparente en cela à ce qui a été qualifié par Kumar et Welker [Kumar *et al.*, 1992] d'ingénierie de méthodes situationnelles qui vise à « **adapter une méthode de développement de SI et les outils associés à chacun des projets spécifiques auxquels elle est appliquée** ».

3.3. L'ingénierie de méthodes situationnelles

L'ingénierie de méthodes situationnelles promeut la notion de composants de méthodes réutilisables, de processus de sélection et d'assemblage de ces composants selon la situation particulière de chaque projet [Brinkkemper *et al.*, 1998], [Ralyté *et al.*, 2001b].
L'approche situationnelle trouve sa justification dans l'analyse de la pratique des méthodes qui montre qu'une méthode n'est jamais suivie à la lettre mais au contraire adaptée à la situation de chaque projet. Donc, cette discipline vise à construire des nouvelles méthodes d'ingénierie des systèmes d'information en réutilisant et assemblant différents fragments de méthodes qui ont déjà fait leurs preuves « à la volée » pour satisfaire aux spécificités des projets [Saeki *et al.*, 1994], [Harmsen *et al.*, 1994], [Harmsen, 1997], [Brinkkemper *et al.*, 1998], [Song, 1995], [Rolland *et al.*, 1996], [Rolland *et al.*, 1998d], [Rolland *et al.*, 1999], [Ralyté, 1999], [Ralyté *et al.*, 2001a], [Ralyté *et al.*, 2001b], [Deneckere *et al.*, 1998], [Deneckere, 2001], [Ralyté, 2002].

Plusieurs types de fragments de méthodes ont émergé dans la littérature :

1. ***Method fragment*** (figure 20.A) : un fragment de méthode est un composant standardisé basée sur une partie cohérente d'une méthode. Il correspond soit à un produit soit à un fragment de processus [Harmsen *et al.*, 1994], [Brinkkemper, 1996] ;
2. ***Method chunks*** (figure 20.B) : les morceaux de méthodes visent à associer des composantes réutilisables à leur description pour faciliter la recherche et l'extraction de composants en fonction des besoins de l'utilisateur [Ralyté *et al.*, 2001a], [Mirbel *et al.*, 2006], [Agerfalk *et al.*, 2007] ;
3. ***Components*** (figure 20.C) : les composants de méthode consistent en des descriptions de processus (des règles et des recommandations), des notations (des règles sémantiques, syntaxiques et symboliques), et des concepts réutilisables [Agerfalk, 2003], [Wistrand *et al.*, 2004], [Karlsson, 2005] ;

4. ***OPEN Process Framework*** (OPF) (figure 20.D) : consiste en des fragments de processus générés à partir des éléments contenus dans un méta-modèle prescrit. Ce méta-modèle est basé sur la norme internationale ISO/IEC 24744 [Henderson-Sellers, 2002] ;
5. ***Method services*** (figure 20.E) : les services méthode proposent un nouveau type de fragments de méthodes réutilisables. Ils fournissent des connaissances métier et ils permettent la construction de méthodes par assemblage de services [Guzélian *et al.*, 2007], [Deneckère *et al.*, 2008], [Rolland, 2008].

Historiquement, le terme fragment a été le premier à apparaître, bien avant les termes composant, "*method chunk*", etc. Nous utiliserons dans le reste de cette thèse le terme : « fragment de méthode ».

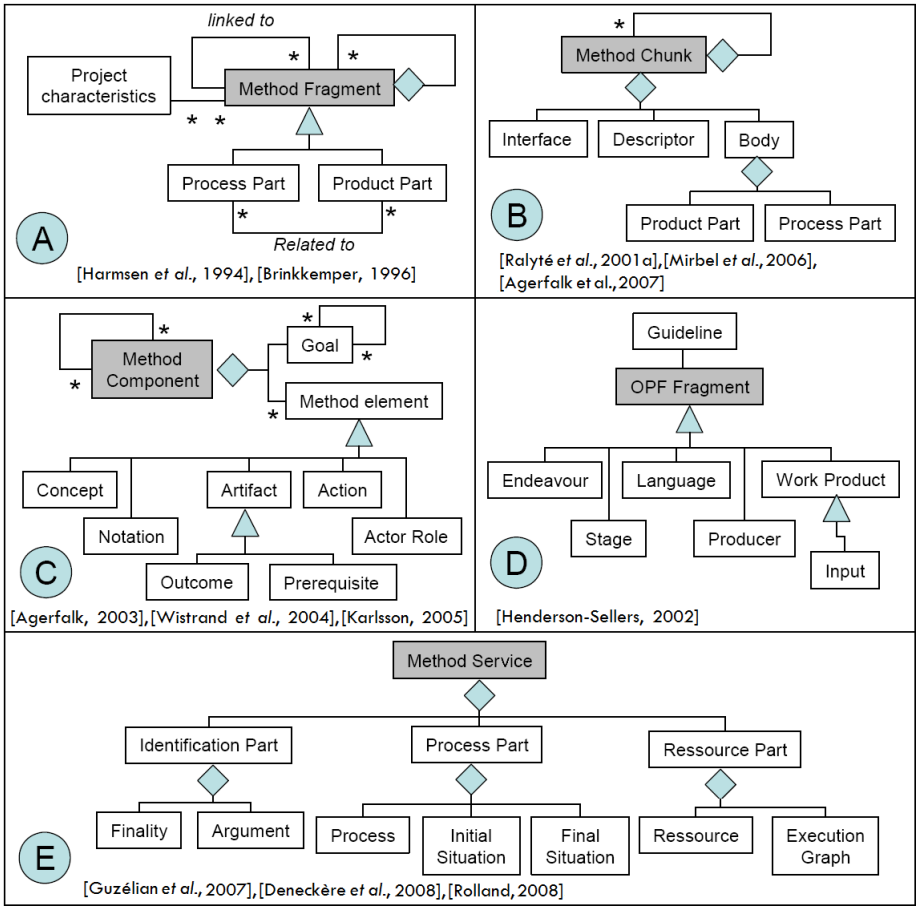


FIGURE 20. META-MODELES DE PLUSIEURS TYPES DE FRAGMENTS [DENECKERE *ET AL.*, 2008]

Le consensus de toutes ces approches se trouve dans le rôle principal d'un fragment de méthode. Il consiste à fournir des directives aux concepteurs de systèmes pour comprendre certaines activités de développement (par exemple : la modélisation métier, la spécification des besoins, la conception, etc.) et pour fournir les définitions de concepts à utiliser dans cette activité [Ralyté *et al.*, 2006]. Les fragments de méthode sont alors, des blocs de constructions réutilisables qui permettent de définir

des méthodes de manière modulaire. Les méthodes ainsi obtenues sont elles-mêmes modulaires et peuvent être modifiées et étendues facilement.

La figure 21 montre un exemple d'un fragment de méthode conforme au méta-modèle du fragment « chunk » (figure 20.B). Ce fragment de méthode peut aider à réaliser l'activité de la découverte des besoins fonctionnels d'un système avec la stratégie de cas d'utilisation. Les parties de produit utilisées pour la découverte des cas d'utilisation sont l'acteur et le cas d'utilisation.

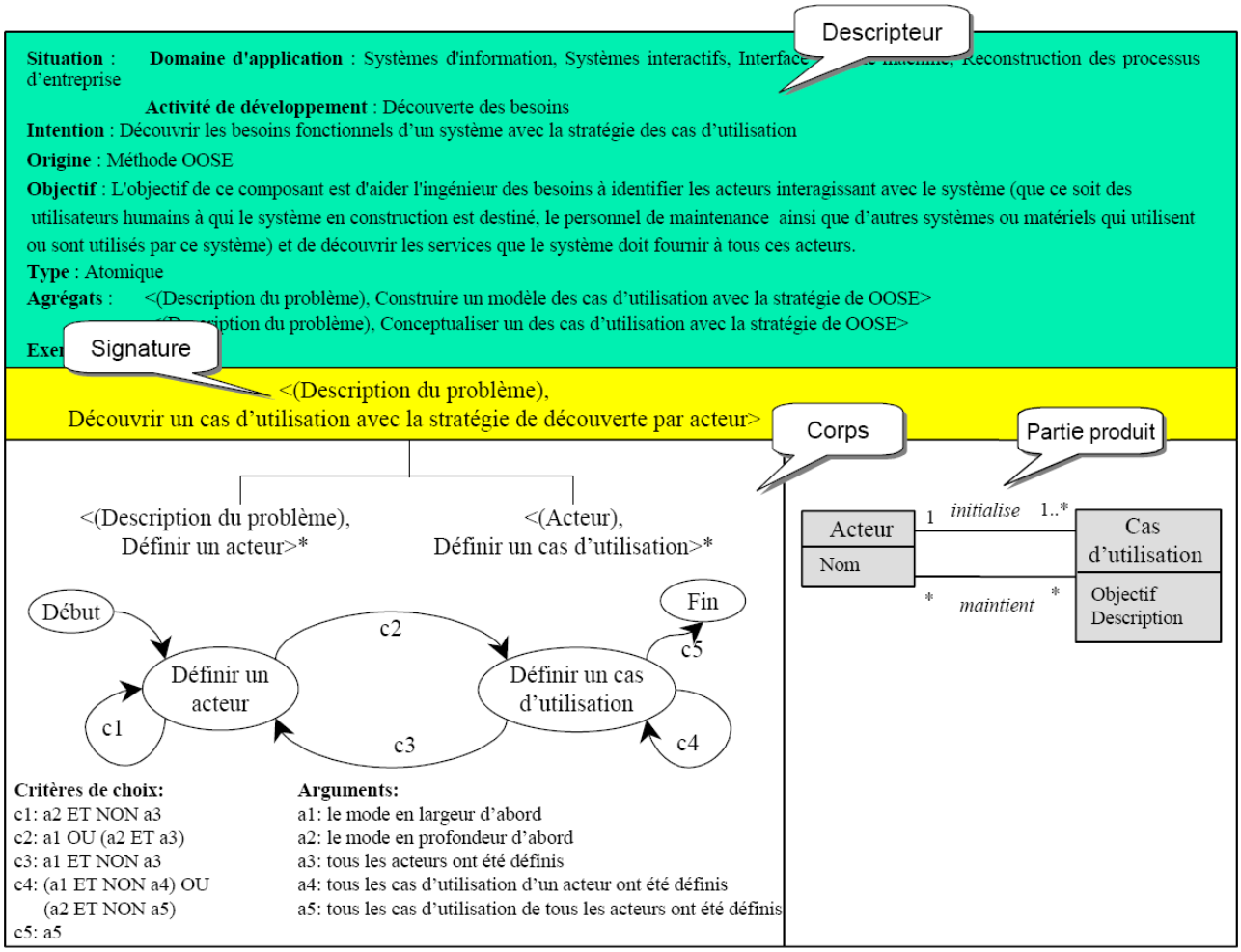


FIGURE 21. EXEMPLE DE FRAGMENT DE METHODE ISSU DE LA METHODE OOSE [RALYTE ET AL., 2001A]

Le domaine de l'ingénierie des méthodes situationnelles que nous venons d'introduire s'intéresse à la réutilisation de fragments de méthodes pour la construction de solutions de conception adaptées aux spécificités du projet.

La réutilisation dans l'ingénierie des méthodes situationnelles est inspirée de la réutilisation dans le monde du logiciel où elle est définie comme une approche de développement selon laquelle il est possible de construire un système à partir de composants existants, produits à l'occasion de développements antérieurs [Ralyté, 2001]. Le principe de réutilisation logicielle est appliqué

aujourd'hui à toutes les étapes du cycle de développement d'un logiciel. Initialement introduite pour améliorer la productivité de la programmation, la réutilisation intervient dans les activités d'expression des besoins, d'analyse et de conception ainsi qu'en ingénierie des méthodes.

Trois approches prédominent actuellement en ingénierie des méthodes situationnelles :

1. L'approche par extension de méthodes appelée : la ré-ingénierie des méthodes ;
2. L'approche par assemblage des composants réutilisables ;
3. L'approche par méta-modélisation de processus d'ingénierie de SI.

Les modèles de processus de ré-ingénierie et d'assemblage sont modélisés au moyen du formalisme MAP [Rolland *et al.*, 1999], [Benjamen, 1999].

3.3.1. La ré-ingénierie des méthodes

Dans la section précédente nous avons montré comment l'ingénierie des méthodes applique le principe de réutilisation pour construire de nouvelles méthodes d'ingénierie des SI.

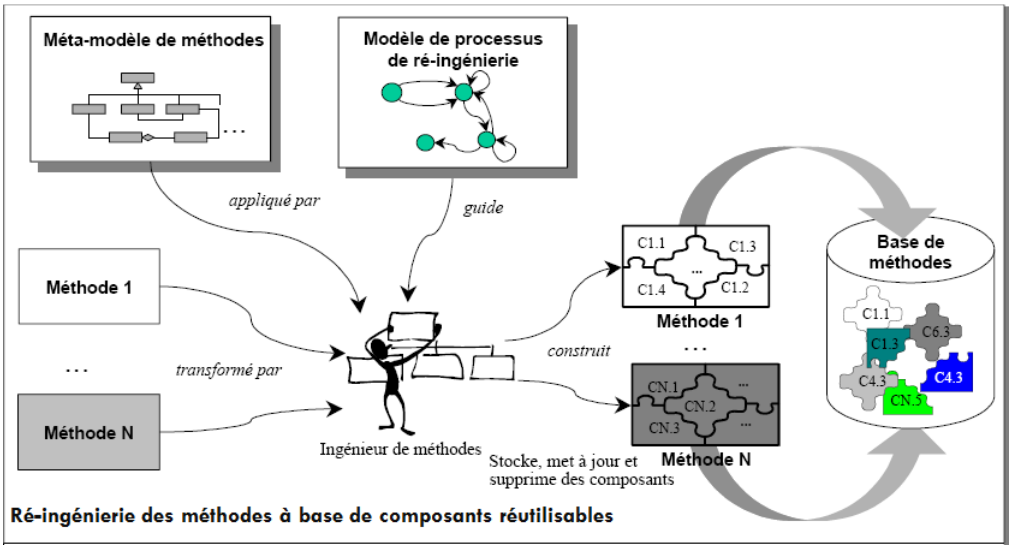


FIGURE 22. APERÇU DE L'APPROCHE DE RE-INGENIERIE DES METHODES ET L'ASSEMBLAGE DES COMPOSANTS REUTILISABLES [RALYTE, 2001]

L'alimentation d'une base de méthodes requiert la **réingénierie des méthodes** existantes dont tout ou partie a fait ses preuves. Elle se fonde sur le découpage modulaire des méthodes en composants réutilisables (voir figure 22).

Il y a peu de travaux sur la ré-ingénierie des méthodes sous forme de composants réutilisables. La plupart des approches [Brinkkemper *et al.*, 1999], [Punter et *et al.*, 1996], [Song, 1997] considèrent comme réutilisables les différents modèles ou diagrammes de méthodes existantes.

Parfois, le niveau de granularité est celui d'un concept [Brinkkemper *et al.*, 1999], d'une propriété, d'un critère, d'une directive, d'une notation ou d'une action [Song, 1997]. Toutefois, ces approches n'expriment pas clairement quand, une étape, un modèle, un diagramme ou un concept peut être considéré comme un module réutilisable dans la construction de nouvelles méthodes.

D'après Ralyté et Rolland [Ralyté et al., 2001b], la ré-ingénierie des méthodes sous forme de composants réutilisables est centrée sur la décomposition du modèle de processus de la méthode en directives autonomes et réutilisables en dehors de la méthode d'origine.

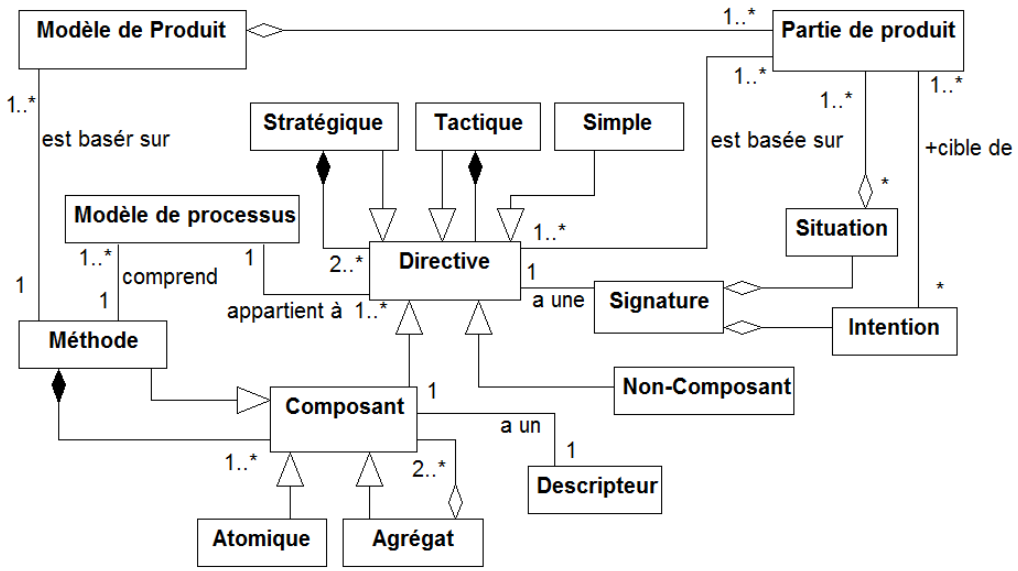


FIGURE 23. META-MODELE D'UNE METHODE MODULAIRE [RALYTE ET AL, 2001A]

La figure 23 présente un méta-modèle de représentation modulaire des méthodes [Ralyté *et al.*, 2001a]. Selon ce méta-modèle, une méthode est vue comme un ensemble de composants de différents niveaux de granularité, la méthode elle-même pouvant être un composant du plus haut niveau de granularité. La définition d'un composant de méthode est dirigée par la décomposition d'une démarche de méthode en sous-démarches que l'on appelle des directives.

Suivant ce raisonnement, la décomposition d'une méthode en composants est basée sur la décomposition de son modèle de processus en directives. Un composant de méthode est vu comme une directive couplée aux parties de produit nécessaires à l'exécution de la démarche capturée dans cette directive.

La partie produit du composant est en quelque sorte un sous-modèle de l'un des modèles de produit de la méthode correspondante. Plus précisément, cette partie contient les différentes parties de produit appartenant au modèle de produit de la méthode, nécessaires à l'application du processus capturé dans la partie processus du composant.

De la même manière, la partie processus du composant est un sous-modèle d'un des modèles de processus de la méthode correspondante. Ce sous-modèle de processus est représenté sous forme

d'une directive. C'est la partie la plus importante d'un composant car elle définit une démarche que l'ingénieur de développement doit suivre pour atteindre l'objectif du composant.

Un autre concept important dans le méta-modèle de méthodes modulaires (Figure 23) est **la signature de la directive**. Toute directive a une signature qui représente les conditions d'utilisation de celle-ci. Un composant étant une spécialisation d'une directive, la signature de cette dernière est aussi la signature du composant.

Afin de s'adapter aux différents modèles de processus des méthodes existantes, le méta modèle prévoit trois types de directives : **simple, tactique** ou **stratégique**. Une **directive simple** est atomique, elle ne se décompose pas et propose une ou plusieurs actions à exécuter, manuellement ou avec l'aide d'un outil.

Une **directive tactique** a une structure d'arbre permettant de décomposer le processus méthodique en sous-processus. Finalement, une **directive stratégique** s'appuie sur le formalisme MAP qui permet de représenter une démarche méthodologique comme un graphe dirigé et étiqueté dont les nœuds sont des intentions (ou buts) à atteindre et les arcs sont les stratégies pour les atteindre. Le formalisme MAP permet ainsi de représenter plusieurs processus dans une même directive, il est donc **multi-démarches**.

Finalement, le concept de **descripteur** (Figure 23) est associé à chaque composant et permet de définir le contexte de réutilisation de celui-ci.

Une autre tendance qui porte sur la modification du Produit et de la démarche d'une méthode spécifique selon la situation en cours et les besoins de l'ingénieur de méthodes a été proposée par Deneckere [Deneckere 2001]. L'approche guide l'ingénieur de méthodes avec des patrons d'extension l'aidant à identifier des situations types et en lui fournissant des conseils pour exécuter l'extension selon ces situations. L'intégration de concepts ou de démarches selon les besoins de l'application est supportée par l'utilisation de la technique des patrons. Donc, les intégrations sont regroupées dans une bibliothèque spécifique sous l'appellation de Patrons définis pour guider l'ingénieur de méthodes lors de l'extension de la méthode d'origine. Les contributions de cette approche sont :

1. Une technique de représentation de la connaissance sous forme de patrons (voir figure 24) décrits par différents langages de description et de manipulation [Deneckere, 2001]. Un patron est composé de deux parties : la connaissance réutilisable (le corps) et les aspects applicatifs (la signature). Le corps encapsule la description du processus à appliquer au produit en modification (par exemple, à la partie de la méthode que l'on veut pouvoir construire ou modifier pour l'adapter à la situation en cours). La signature représente la situation avant modification, l'intention à réaliser et la cible de cette modification. On appelle également ce concept une interface [Rolland *et al.*, 1998]. Elle est vue comme un triplet

<situation, intention, cible> associé au corps. L'interface est la partie visible d'un patron. **La situation** représente le projet en cours. Elle représente toute partie du produit en cours de développement ou toute partie du processus en cours d'exécution pouvant faire l'objet d'une prise de décision de la part de l'ingénieur d'application. **L'intention** reflète le choix que peut faire l'ingénieur d'application à un moment donné du processus. L'intention conduit le processus d'utilisation des patrons. **La cible** représente la partie de la méthode qu'il faut obtenir après application du patron. **Le corps** explique comment procéder pour atteindre l'intention dans cette situation particulière. La figure 24, montre un exemple de l'interface d'un patron.

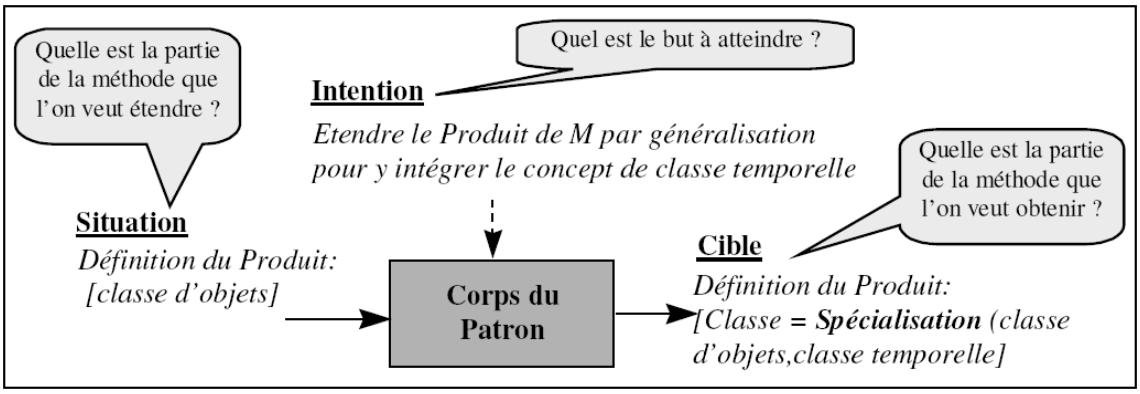


FIGURE 24. EXEMPLE DE L'INTERFACE D'UN PATRON [DENECKERE, 2001]

2. Une technique d'organisation des patrons à l'aide de cartes d'extension de processus facilitant le guidage lors de la réutilisation de patrons. Ces cartes sont des modèles de processus exécutables [Rolland99], [Benjamen, 1999] permettant d'étendre une méthode orientée objet à un domaine d'application spécifique. Elles permettent de gérer la cohérence de la méthode en prenant en compte les contraintes de précédence associées à chaque patron d'extension [Deneckere, 2001]. La figure 25 illustre par un exemple la relation existante entre une carte d'extension de processus et les patrons.

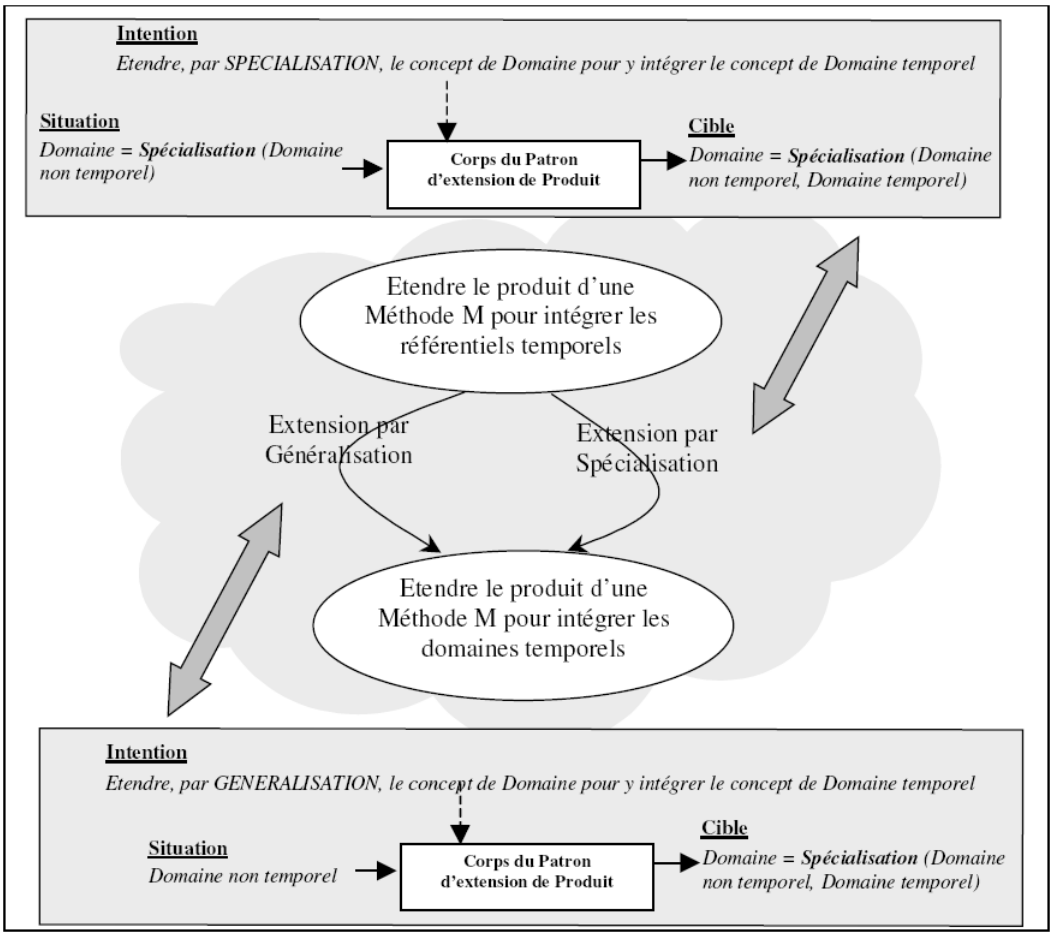


FIGURE 25. EXEMPLE DE PATRONS ORGANISES PAR UNE CARTE DE PROCESSUS [DENECKERE, 2001]

3. Une technique de conception des patrons génériques par l'utilisation de méta-patrons permettant de faciliter le travail de l'ingénieur de méthodes lors de la conception d'une carte d'extension. Ces patrons génériques ont été générés suite à l'inventaire de toutes les stratégies possibles pouvant se trouver sur une carte d'extension et toute la connaissance relative à chacune d'entre elles est encapsulée dans un patron générique spécifique [Deneckere, 2002]. La figure 26 illustre par un exemple de la démarche, il s'agit d'instanciations successives permettant d'obtenir, à partir d'un méta-patron, un patron pour l'intégration d'un élément spécifique, puis un patron instancié permettant d'étendre une méthode particulière. Ici, le concept à intégrer est celui de *Classe Calendrier* et la méthode à modifier est O*.

   La première étape représente l'instanciation du méta-patron Spécification pour le concept de Classe Calendrier. La seconde étape représente l'instanciation du patron (obtenu à l'étape précédente) à la méthode O*. Finalement, le concept de *Classe Calendrier* est inséré en tant que sous-type du concept de *Classe* et le concept déjà existant de *Granule* est supprimé.

Cette approche permet donc d'offrir à l'ingénieur de méthodes un processus de guidage, tant au niveau de l'exécution des patrons d'extension (carte) qu'au niveau de leur conception (Patron générique).

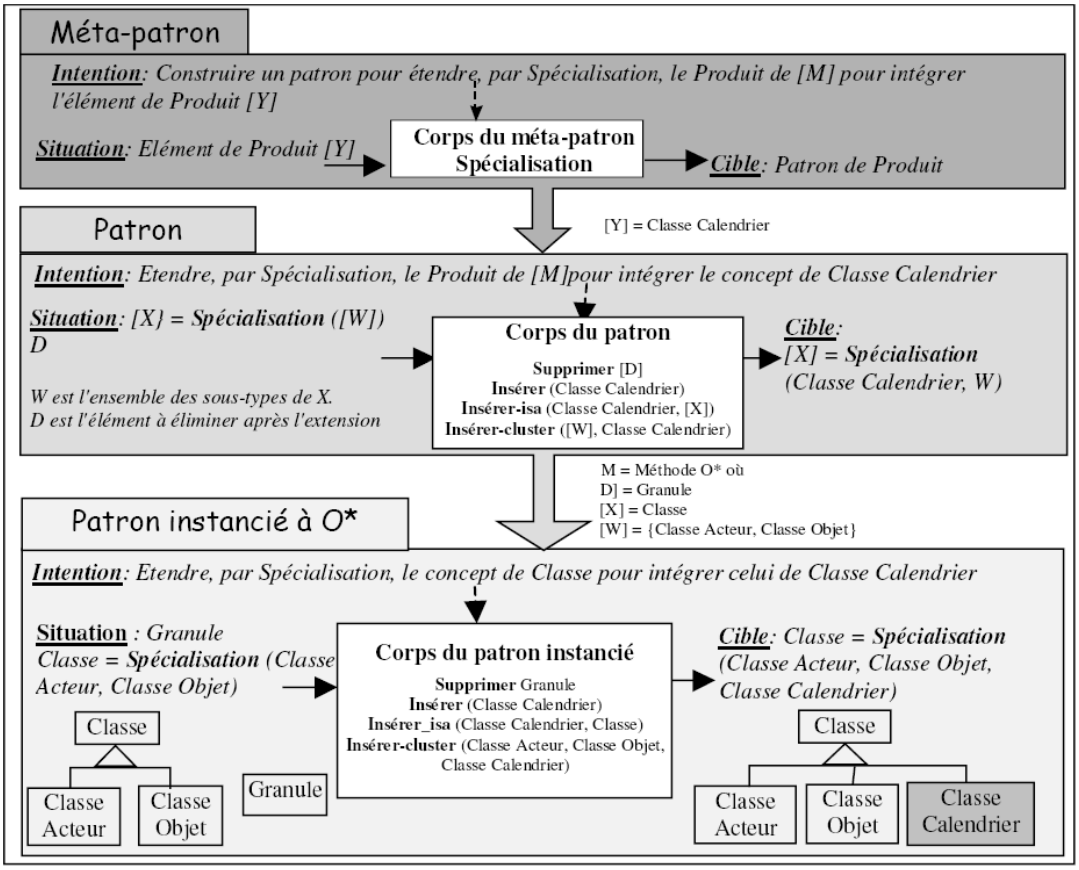


FIGURE 26. EXEMPLE D'UTILISATION D'UN META-PATRON [DENECKERE, 2001]

3.3.2. L'approche par assemblage des composants

La deuxième approche situationnelle concerne **l'assemblage des composants de méthodes** issus de différentes méthodes avec l'objectif d'en construire de nouvelles ou d'en enrichir des existantes. L'élément principal de cette approche est le modèle de processus d'assemblage que l'ingénieur d'applications applique dans le but de construire une nouvelle méthode à partir d'un ensemble de composants stockés dans une base. Ce modèle de processus guide l'ingénieur d'applications dans la sélection des composants et dans leur assemblage qui est fondé sur l'application des opérateurs d'assemblage, des mesures de similarité et des règles de validation de la qualité d'assemblage. La nouvelle méthode, le résultat du processus d'assemblage, peut à son tour être stockée dans la base de méthodes. Les opérateurs d'assemblage permettent d'assembler les parties de produit des différents composants ainsi que leurs directives. Finalement, lorsqu'on applique le processus

d'assemblage on obtient une nouvelle méthode qui est instance du méta-modèle proposé dans la première partie de l'approche (voir figure 27).

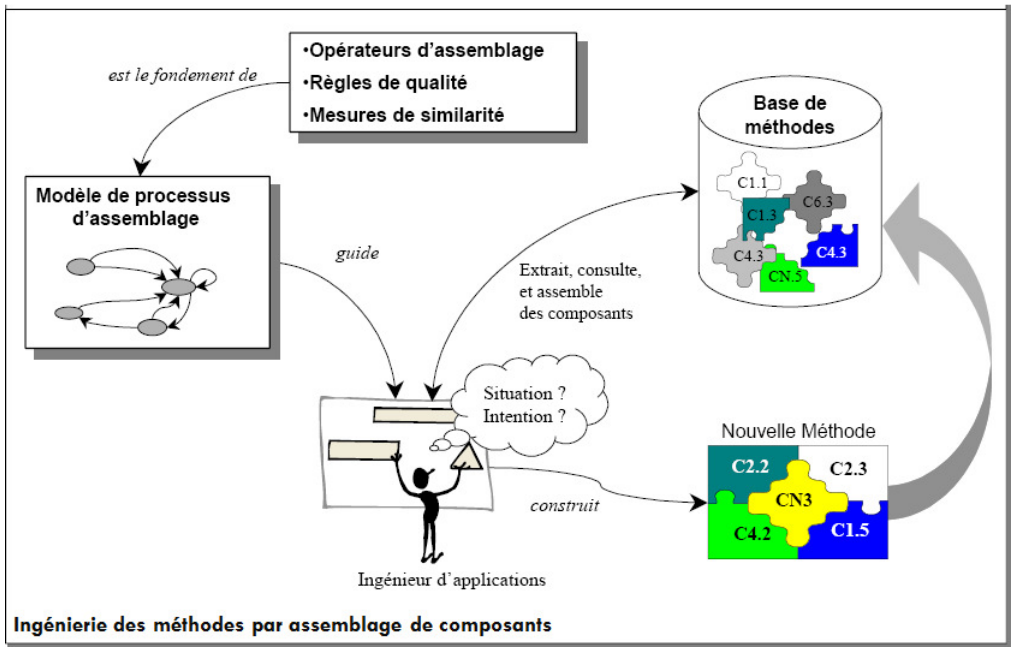


FIGURE 27. APERÇU DE L'APPROCHE D'ASSEMBLAGE DES COMPOSANTS REUTILISABLES [RALYTE, 2001]

Les travaux réalisés dans le domaine de l'assemblage de composants introduisent la notion de composant de méthode qui est vu comme une solution testée et acceptée pour résoudre un problème fréquemment rencontré lors du développement de logiciels. En plus de la notion de composant de méthode, la notion de patron de conception est également utilisée dans l'ingénierie de méthodes. Les patrons de conception génériques proposées par Rolland [Rolland *et al.*, 1996a], [Rolland *et al.*, 1996b] définissent des règles génériques régissant la construction de méthodes différentes mais similaires. Les patrons génériques aident à construire des méthodes situationnelles. Ils permettent de savoir quels sont les meilleurs processus dans telle ou telle situation et guident l'ingénieur de méthodes lors de la construction de sa méthode. Contrairement aux patrons génériques les patrons de conception spécifiques au domaine [Deneckère *et al.*, 1998] permettent de sélectionner, en premier lieu, le méta-patron correspondant au domaine d'extension puis de guider la modification de la méthode en appliquant les patrons suggérés par celui-ci. Le résultat est l'extension des méthodes avec de nouveaux concepts spécifiques.

Par rapport à la démarche méthodologique pour l'analyse et la réalisation de l'assemblage de méthodes, les travaux proposés par Ralyté [Ralyté *et al.*, 200a] prennent en compte l'assemblage des composants qui ont des objectifs similaires dans le processus de l'ingénierie des systèmes et proposent des manières différentes pour les atteindre. De plus, ses solutions permettent d'assembler des composants qui se recouvrent partiellement, c'est-à-dire qui ont des concepts similaires dans leurs

parties de produit et des intentions similaires dans leurs modèles de processus. Ces deux cas d'assemblage de composants sont intégrés sous forme de deux stratégies : **la stratégie d'association** et **la stratégie d'intégration.**

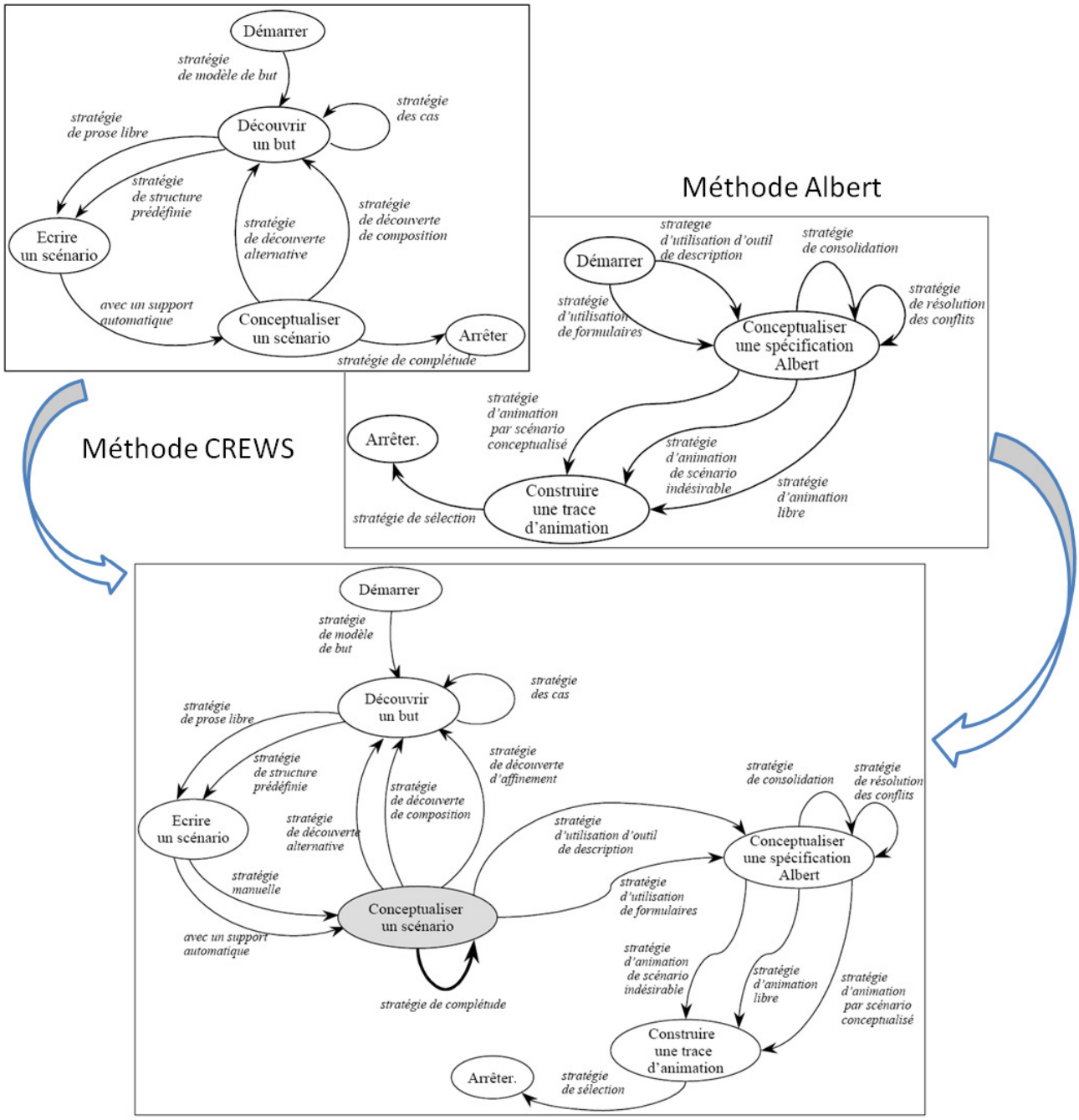


FIGURE 28. L'ASSEMBLAGE SUIVANT LA STRATEGIE D'ASSOCIATION [RALYTE, 2001]

Le processus d'assemblage suivant la stratégie d'association est basée sur l'identification des liens ou des concepts permettant de faire la connexion entre les modèles de produits de deux composants et de déterminer l'ordre d'exécution de leurs modèles de processus. En général dans ce cas, le produit résultat de l'application du premier composant est utilisé ensuite en tant que produit source dans le deuxième composant. Par exemple, la figure 28 illustre le processus d'assemblage

des composants des méthodes CREWS l'Écritoire et Albert [Heymans, 1998] qui représentent deux processus complémentaires dans la conception d'un système d'information. Le premier celui de CREWS sert à découvrir des besoins du système et à les conceptualiser sous forme de buts et de scénarios tandis que le deuxième celui d'Albert permet de valider ces besoins par les animations des scénarios écrits au préalable. Ceci veut dire que le premier composant produit des scénarios qui sont utilisés par le deuxième. Par conséquent, l'assemblage de ces composants est de type association.

Le processus d'assemblage suivant la stratégie d'intégration consiste à fusionner les éléments communs des modèles de produits des composants ainsi que ceux de leurs modèles de processus. Par exemple, le composant qui produit le modèle des cas d'utilisation de la méthode OOSE de Jacobson [Jacobson *et al.*, 1992] pourrait être enrichi par des directives d'écriture des scénarios et d'identification des buts défini dans les composants de la méthode CREWS l'Écritoire [Rolland *et al.*, 1998a], [Rolland *et al.*, 1998b]. La figure 29 illustre le processus d'assemblage du méta-modèle de produit de la méthode OOSE avec le méta-modèle de produit de l'approche CREWS l'Écritoire.

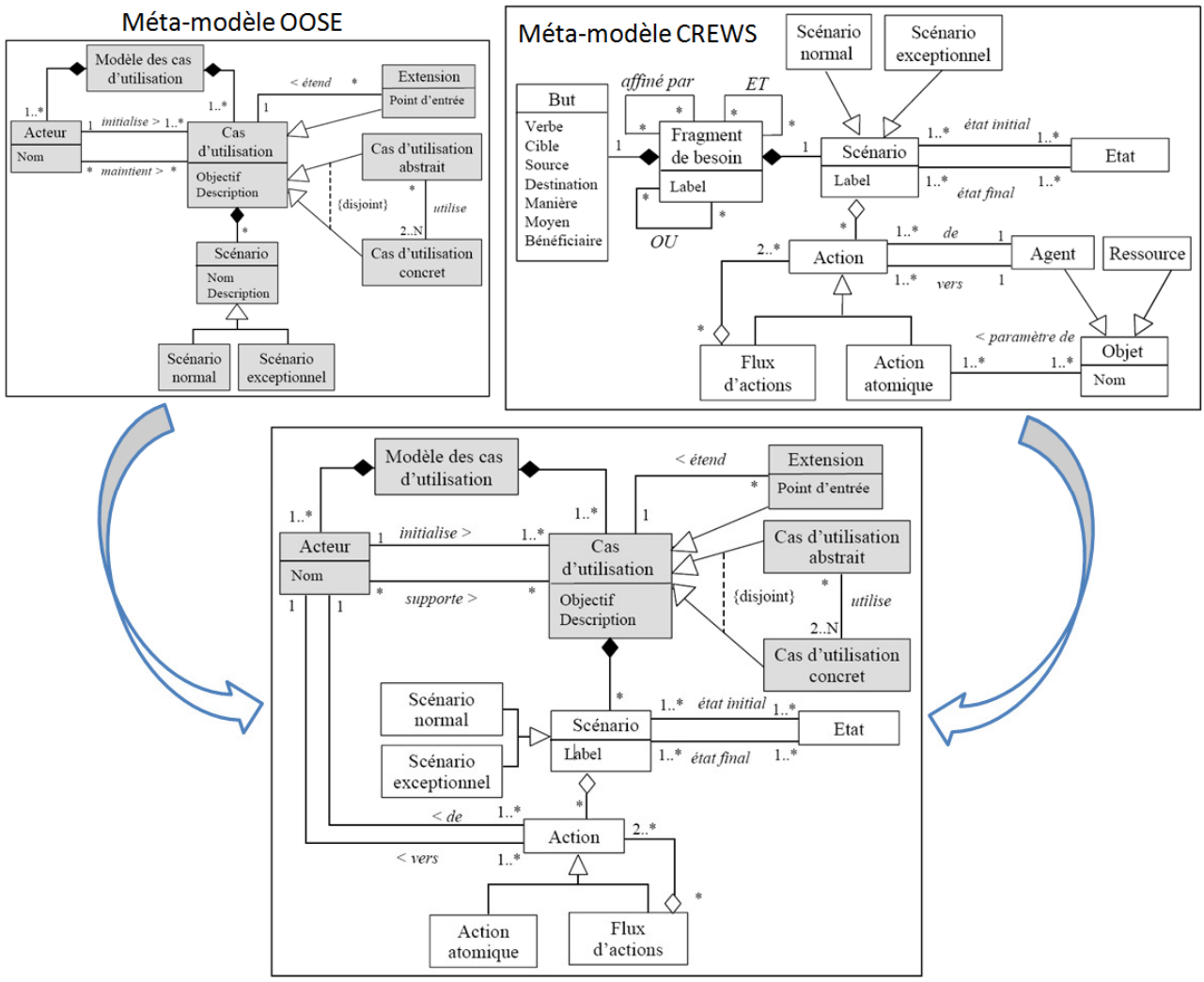


FIGURE 29. L'ASSEMBLAGE SUIVANT LA STRATEGIE D'INTEGRATION [RALYTE, 2001]

Le méta-modèle des cas d'utilisation de la méthode OOSE (la partie en haut et à gauche de la figure 29) est destiné à la description du comportement d'un système du point de vue de ses utilisateurs. Le concept principal de ce méta-modèle est un cas d'utilisation. Celui-ci est composé d'un ensemble de scénarios. Il a toujours un scénario normal et plusieurs scénarios d'exception. Contrairement au scénario du méta-modèle CREWS l'Écritoire (la partie en haut et à droite de la figure 29), le scénario n'est pas décomposé en sous-éléments; il s'agit d'une description informelle.

Le composant permet la réutilisation des descriptions communes à plusieurs cas d'utilisations par le biais des cas d'utilisation abstraits. Un cas d'utilisation peut être concret ou abstrait. Les cas abstraits sont des extractions des descriptions communes à plusieurs cas concrets. Ils ne peuvent pas être instanciés en tant que tels car ils ne décrivent pas des scénarios complets mais seulement des fragments des scénarios qui sont partagés par plusieurs cas. Ils permettent de réutiliser ces descriptions communes dans celles de nouveaux cas concrets. Le composant permet aussi d'étendre les cas d'utilisation par des extensions optionnelles définies par des cas d'extension qui sont aussi considérés comme des cas d'utilisation.

Chaque cas d'utilisation a un acteur qui l'initialise en interagissant avec le système. Il peut aussi avoir un ou plusieurs acteurs qui interviennent au cours de l'exécution d'un cas d'utilisation. Finalement, le modèle des cas d'utilisation est défini comme une collection des cas d'utilisation initiés par des acteurs.

Le méta-modèle de méthode de l'approche CREWS L'Ecritoire est destiné à la découverte des besoins à partir de scénarios textuels. Le concept principal de l'approche est appelé un Fragment de Besoin (FB). Un Fragment de Besoin est un couple <But, Scénario>, où le but est défini comme "*quelque chose que l'utilisateur du système espère obtenir dans le futur*" et le scénario est défini comme "*le comportement possible du système limité à un ensemble d'interactions significatives entre plusieurs agents pour atteindre le but*".

L'assemblage des méta-modèles de produits consiste à (1) remplacer le concept de scénario d'OOSE par celui de CREWS l'Écritoire, car la structure de ce dernier est beaucoup plus complexe et à (2) fusionner le concept acteur d'OOSE et le concept agent de CREWS l'Écritoire car ces deux concepts ont la même signification dans la réalisation d'un cas d'utilisation.

3.3.3. L'approche par méta-modélisation de processus d'ingénierie de SI

L'approche par méta-modélisation a pour objectif d'offrir aux organisations des méthodes d'ingénierie de SI utiles et utilisables. En particulier, le processus de développement doit être adapté aux contraintes et spécificités de leurs projets et/ou permet de définir de nouveaux processus « from scratch ».

Récemment, Hug [Hug, 2009] focalise ses efforts sur le processus d'ingénierie de SI et leur méta-modélisation, c'est-à-dire à un niveau d'abstraction plus élevé que l'approche de l'ingénierie des méthodes situationnelles. Il s'agit de proposer une méthode pour construire des méta-modèles de processus pour l'ingénierie des systèmes d'information.

Hug [Hug, 2009] a proposé un méta-modèle de domaine des processus d'ingénierie de SI qui contient les concepts principaux du domaine (Figure 30). Ce méta-modèle a été obtenu en réalisant une analyse des classes équivalentes des différents méta-modèles de processus existants et en les pondérant afin de ne sélectionner que les classes les plus importantes. Les relations définies entre ces classes proviennent également d'une analyse des différents méta-modèles de processus existants.

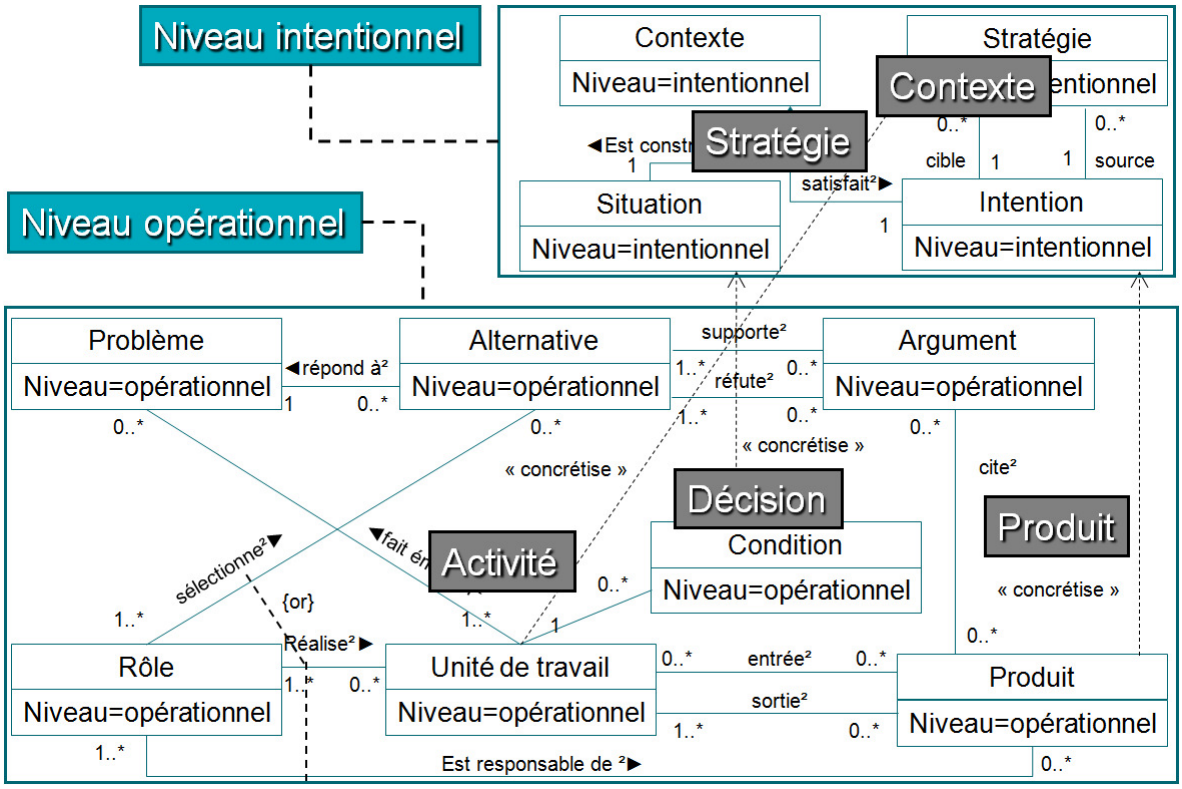


FIGURE 30. META-MODELE DE DOMAINE [HUG, 2009]

Ce méta-modèle de domaine prend en compte les cinq points de vue de la modélisation des processus d'ingénierie de SI : activité, produit, décision, contexte et stratégie. Il prend également en compte deux niveaux d'abstraction des processus : le niveau intentionnel et le niveau opérationnel.

La figure 31 illustre ces cinq points de vue et ces deux niveaux d'abstraction. Seuls les principaux concepts sont représentés sur cette figure. Chaque concept du méta-modèle de domaine est associé à un point de vue et à un niveau d'abstraction qu'il soit opérationnel ou intentionnel. Par exemple, le concept « Rôle » se situe au niveau opérationnel et est issu des méta-modèles de processus orientés « activité ».

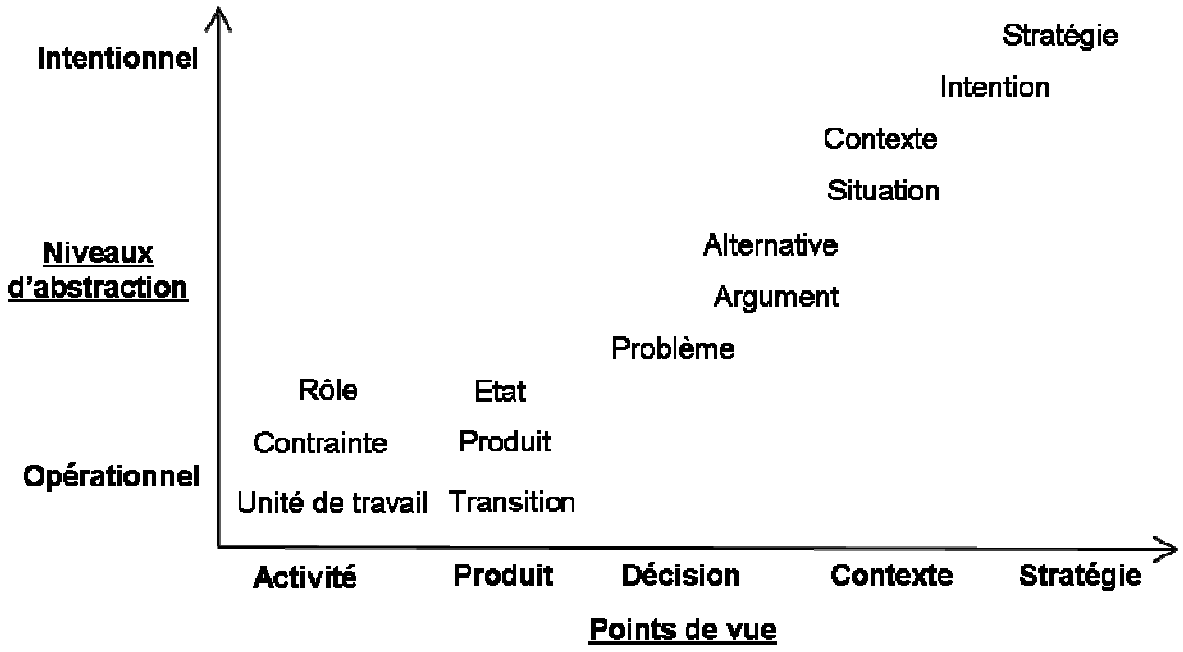


FIGURE 31. POINTS DE VUE ET NIVEAUX D'ABSTRACTION [HUG, 2009]

Parallèlement à la proposition d'un méta-modèle de domaine des processus, Hug a proposé une méthode pour la construction de méta-modèles de processus (Figure 32). La construction de méta-modèles de processus est basée sur l'imitation des patrons génériques et de domaine spécifiques à la méta-modélisation des processus d'ingénierie de SI. Les patrons génériques sont des patrons produit qui peuvent être imités sur des classes du méta-modèle de domaine. Les patrons de domaine sont des fragments issus de méta-modèles de processus existants qui permettent d'enrichir les classes du méta-modèle de domaine.

La construction de méta-modèles de processus est, également, basée sur l'utilisation d'un graphe conceptuel qui permet la navigation et le choix de concepts à intégrer aux méta-modèles de processus de manière cohérente. Les ingénieurs des méthodes sont également guidés dans la sélection des attributs des classes de leurs méta-modèles. La méthode propose ensuite l'instanciation des méta-modèles de processus, en passant par la sélection de formalismes adaptés au contexte de l'organisation.

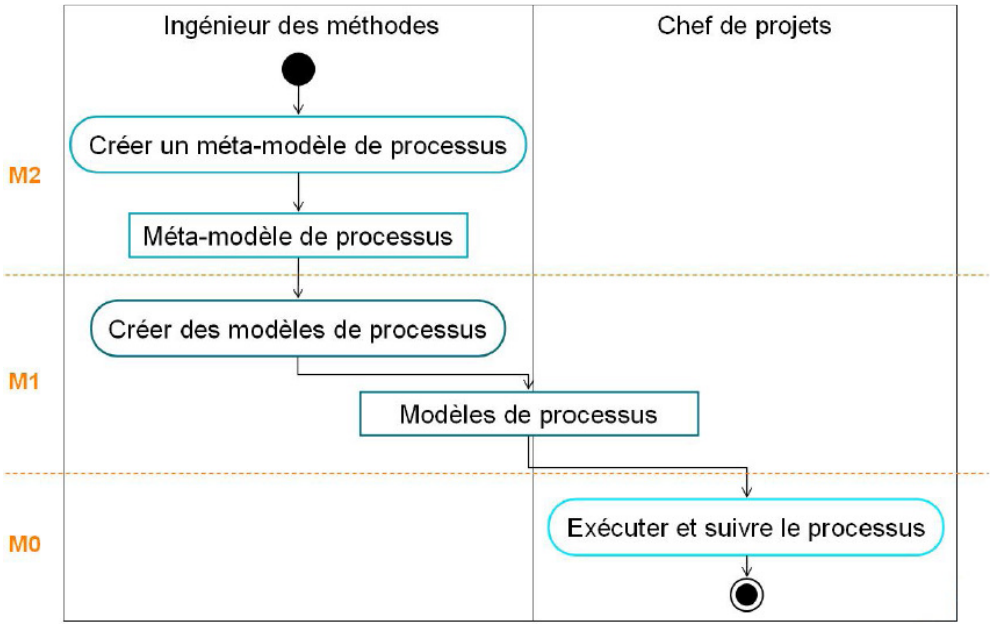


FIGURE 32. METHODE POUR LA CONSTRUCTION DU META-MODELE DE PROCESSUS [HUG, 2009]

Le graphe conceptuel regroupe les concepts du méta-modèle de domaine et les concepts issus de l'imitation des patrons génériques et de domaine, telles que : unité du travail, catégorie d'unité de travail, etc. Ce graphe représente l'ensemble des concepts pouvant être intégrés aux méta-modèles de processus. Il permet à la fois de cadrer la création des méta-modèles de processus et de guider les ingénieurs des méthodes dans le choix des concepts, via différentes relations. Les relations définies entre les concepts sont de trois types : complétude, abstraction et précision. La complétude permet d'étendre le méta-modèle en cours de construction à un autre point de vue. L'abstraction permet d'atteindre des concepts du niveau intentionnel et la précision permet de raffiner un concept, via l'imitation de patrons. Les relations entre les concepts ont été définies de manière cohérente, afin que les méta-modèles sous-jacents soient également cohérents.

Finalement, l'enjeu primordial de ces travaux est de guider les ingénieurs des méthodes dans la création de leurs méta-modèles et modèles de processus d'ingénierie de SI mais également dans l'exécution de ces processus et dans leur suivi, pour pouvoir guider, contrôler, mesurer et optimiser la production des SI.

3.4. Synthèse

Dans cette section nous avons présenté un état de l'art sur la réutilisation et la gestion de modèles dans le contexte de l'ingénierie de méthodes situationnelles. Cette étude bibliographique a permis de constater que l'ingénierie de méthodes permet en général de construire des processus spécifiques aux projets par l'assemblage/composition ou extension de fragments de méthodes. Nous

remarquons que le choix et l'adaptation d'une méthode permettent à chaque projet de sélectionner des méthodes parmi différentes approches et de les accorder à leurs besoins. Pour ce faire, nous nous basons sur la notion des fragments méthodes comme source d'inspiration de notre approche, ce qui nous permettra de concevoir un niveau d'abstraction organisationnel à base de services. Dans la section suivante, nous détaillerons les processus existants dans les approches à base de services.

4. LES PROCESSUS DANS LES APPROCHES A BASE DE SERVICES

4.1. La notion de service

L'ingénierie dirigée par les services [Papazoglou, 2003] est un paradigme qui utilise des services comme éléments fondamentaux dans le développement d'applications. Cette approche accélère le développement d'applications en permettant la composition de services distribués [Papazoglou *et al.*, 2005]. Un service est un ensemble de modules logiciels autonomes qui sont décrits, publiés, découverts, composés et négociés à la demande d'un client.

Les services exécutent des fonctions qui peuvent être aussi bien de simples requêtes dans un formulaire, que des processus métiers complexes [Papazoglou *et al.*, 2005]. Pour soutenir l'intégration des applications basées sur différents processus métier, la modélisation de services est supportée par l'architecture orientée service (SOA) [Papazoglou *et al.*, 2005]. Une architecture orientée services est l'ensemble des politiques, patrons de conception et environnements qui permettent l'intégration des différentes fonctionnalités exposées comme des services. Dans une telle architecture, l'utilisation des services suit un protocole bien établi : publication, découverte et invocation.

L'objectif de SOA est de fournir des services aux utilisateurs finaux ou aux applications. Cette approche définit une interaction entre les agents logiciels comme un échange de messages entre clients et fournisseurs. Les agents peuvent être simultanément des clients et des fournisseurs.

La figure 33 présente les principaux acteurs qui interviennent dans une architecture à services.

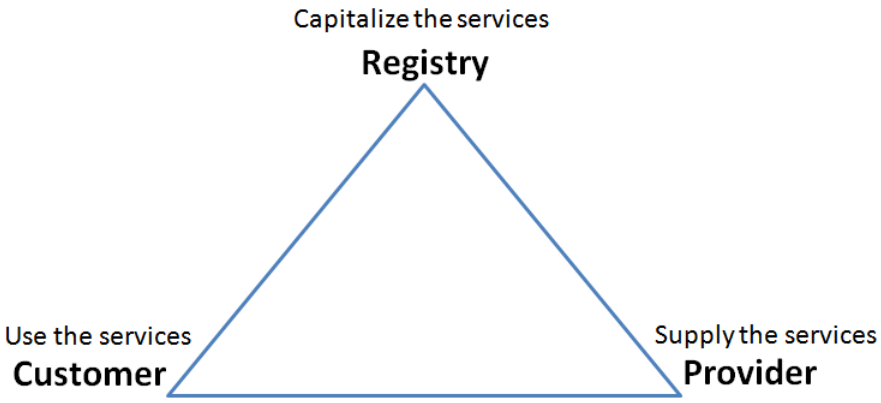


FIGURE 33. L'APPOCHE A BASE DE SERVICES

Le premier acteur est le **fournisseur du service** (« provider »). Il représente une personne/une organisation (ou des agents) qui offrent des fonctionnalités sous forme de service. Après le développement d'un service, un fournisseur doit mettre à disposition des éventuels utilisateurs les informations nécessaires pour pouvoir utiliser le service, c'est-à-dire la **description du service**.

La description du service est ensuite publiée dans un registre de services (« registry ») (appelé également annuaire de services ou courtier de services) qui sert d'acteur intermédiaire entre fournisseurs et consommateurs de services.

Un utilisateur de service, nommé le **client du service** (« customer »), interroge le registre de services pour obtenir les services disponibles qui correspondent à ses besoins. La découverte d'un service est réalisée grâce à la description du service disponible dans l'annuaire. Après avoir sélectionné le service qu'il veut utiliser, le client du service peut, dans certains cas, négocier auprès du fournisseur les termes suivant lesquels il peut utiliser ce service. A la fin de la négociation, un accord de service est réalisé entre le client et le fournisseur. La plupart du temps, cet accord de service contractualise les termes de l'utilisation du service par le client sans garantie totale du résultat. Grâce aux informations disponibles dans la description du service, le client de service peut, dès lors, réaliser la liaison et appeler les fonctionnalités du service.

Ces interactions entre clients et fournisseurs de services sont réalisées dans une architecture à services à travers un environnement d'intégration et d'exécution de services où les clients sont les agents logiciels qui demandent l'exécution d'un service.

L'approche à base de services repose sur l'idée qu'une application peut être réalisée par composition des services logiciels mis à disposition par des fournisseurs divers. Cette composition peut intégrer des services hétérogènes implémentés avec des technologies différentes, voire même des logiciels propriétaires.

4.1.1. La composition de services

La composition de services est le mécanisme qui permet l'intégration des services dans une application [Benatallah *et al.*, 2005]. Le résultat de la composition de services peut être un nouveau service, appelé service composite. Dans ce cas, la composition est dite composition récursive ou hiérarchique.

La composition de services est aujourd'hui un sujet de grand intérêt autant pour le monde de la recherche que pour le monde industriel. De nombreuses recherches visent à développer des modèles et des outils de composition de services.

Une composition de services assemble les fonctionnalités des services pour réaliser une fonctionnalité plus complexe. La spécification de la composition consiste à identifier les composants qui fournissent les services nécessaires et à décrire leurs interactions [Ferguson *et al.*, 2005].

La composition de services est largement utilisée pour les services Web. Parmi les modèles de composition de services nous citons : la composition de services exprimée par procédés et la composition sémantique de services.

4.1.1.1. Les modèles de composition de services par procédés

Le modèle de composition de services exprimé par procédés décrit à travers un procédé le comportement de la composition des services. Un procédé est composé d'un ensemble d'activités et d'un flot de contrôle. Deux approches existent actuellement pour ce type de composition de services : l'orchestration de services et la chorégraphie de services.

L'orchestration de services [Ferguson *et al.*, 2005] offre une vision centralisée de la logique de coordination d'une composition de services Web. Elle présente la vision (interne) du fournisseur du service composite résultant de cette composition. L'exécution de l'orchestration de services est contrôlée par une entité centrale, appelée moteur d'exécution, qui gère l'invocation des différents services intervenant dans la composition selon la logique définie par le procédé. Un procédé spécifie l'ordre et les conditions des invocations des services participant à l'orchestration de services. Parmi les langages existants nous citons le langage BPEL.

La chorégraphie de services [Barros *et al.*, 2005] décrit, d'un point de vue global, la manière dont un ensemble de services collaborent pour atteindre un but commun. Elle spécifie les interactions des services participants et les dépendances entre ces interactions, qui peuvent être des dépendances de flot de contrôle (une interaction doit être précédée par une autre), des dépendances de données (une interaction nécessite la réception d'un message avant de débuter), etc. Une chorégraphie ne décrit aucune action interne d'un service participant qui n'a pas d'effet visible à l'extérieur (telles que des traitements internes ou des transformations de données, par exemple) [Barros *et al.*, 2005]. L'intérêt principal d'une chorégraphie est de vérifier qu'à l'exécution tous les échanges de messages entre partenaires sont réalisés conformément à cette spécification.

4.1.1.2. Les modèles de composition sémantiques de services

La description WSDL d'un service WEB permet de décrire la fonctionnalité remplie par le service. Elle spécifie d'un point de vue syntaxique les opérations du service, les types de données d'entrée et de sortie, les différents points de sortie, les différents points d'entrées du service et le moyen de réaliser la liaison avec le service. Le Web sémantique ajoute à cette description syntaxique la description sémantique des capacités du service. En général, le modèle sémantique d'un service est réalisé en ajoutant un autre niveau à la pile de protocoles standard des services Web qui permet d'exprimer la sémantique des aspects fonctionnels, non-fonctionnels et comportementaux d'un service. Actuellement, plusieurs propositions de descriptions sémantiques ont été proposées : Web Service Modeling Ontology (WSMO) [De Brujin *et al.*, 2005], OWL-S [Martin *et al.*, 2005] et WSDLS [Patil *et al.*, 2004].

Les éléments principaux du Web sémantique sont les ontologies. Une ontologie permet de standardiser le modèle d'information d'un métier. Elle définit et classifie la terminologie d'un domaine. Cela permet de réaliser la liaison entre les concepts concrets, familiers aux différents acteurs humains d'un domaine, et leurs correspondants abstraits du système d'information du domaine [Fensel *et al.*, 2002].

L'approche sémantique permet de décrire de façon abstraite une application en termes des buts (Goals) à atteindre. Cette description est bien sûr réalisée à travers des données sémantiques. Le Web sémantique propose une nouvelle approche pour la réalisation des solutions à services en utilisant un modèle plus riche d'informations - les ontologies. Toutefois, pour l'instant, la réalisation des objectifs de l'adaptation sémantique n'est pas encore tout à fait acquise, le Web sémantique restant un point ouvert dans le domaine de la recherche.

4.2. Les services méthode

L'approche à base de services [Guzélian *et al.*, 2004], [Deneckere *et al.*, 2008], [Rolland, 2008] a été appliquée à des services particuliers : les services méthode. Les services méthode sont des fragments de méthodes réutilisables. Ils fournissent des connaissances méthode et ils permettent la construction de méthodes par assemblage. Les services méthodes sont des services métier proposés aux concepteurs/développeurs de systèmes d'information pour résoudre leurs problèmes de développement. Par exemple : un service méthode peut offrir une démarche pour construire un diagramme de cas d'utilisation.

La structure globale d'un service méthode est composée par un **niveau méta** qui fournit des connaissances sur l'utilisation (définition du problème auquel il répond, définition du contexte dans lequel il s'applique...) et un **niveau de base** qui offre une solution (voir figure 34)

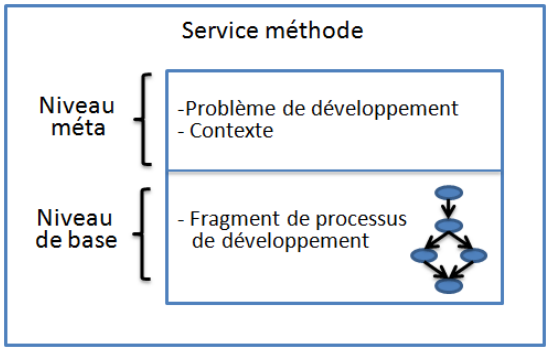


FIGURE 34. VISION GLOBALE D'UN SERVICE METHODE [GUZELIAN, 2007]

Le niveau méta correspond à la vision métier des services méthode. Cette vision suggère de définir un service méthode comme une solution à un problème de développement de SI. Un problème

de développement est exprimé sous la forme d'un but à atteindre dans un certain contexte. Un service méthode contient une partie contextuelle explicitant la situation de développement dans laquelle l'usage de ce service méthode est approprié. Il fournit aussi les avantages et les inconvénients liés à son usage. La conception des services méthode met l'accent sur leur usage.

Concernant le niveau de base, un service méthode fournit un processus. Il définit une démarche pour répondre à un problème de développement et pour atteindre un certain objectif. Ce processus peut nécessiter la réutilisation d'autres services. En considérant le processus permettant de résoudre un problème plutôt que la solution, il est possible d'exprimer de la " variabilité " et donc d'adapter le service au moment de son exploitation. Par ailleurs, en fournissant des fragments de processus, les services méthode peuvent être assemblés pour élaborer des processus de développement complexes.

Au cours des dix dernières années, les approches d'ingénierie de méthodes ont largement adopté une vision modulaire d'une méthode afin d'appliquer des approches orientées composants.

Plus récemment, des approches à base de services ont été proposées. Dans cette section, nous présenterons quatre approches à base de services :

1. **L'approche iSOA (Intentional Service Oriented Architecture)** [Kaabi, 2007], qui propose de découvrir et de décrire les services intentionnels nécessaires à une organisation en développant une perspective de modélisation intentionnelle.
2. **Service-Oriented Meta-Method** (SO2M) proposé par Guzélien [Guzélien, 2007], apparaît comme la première tentative pour appliquer une approche orientée services dans le contexte de l'ingénierie de méthodes. **SO2M** considère les "***method chunks***" comme des services « méthode ». **SO2M** est définie comme un ensemble de services « méthode » auquel est associé un processus de composition pour assurer la recherche, la sélection et la composition de services.
3. **Method as Service** (MaaS) proposé par Rolland [Rolland, 2008], qui consiste en une révision de la notation de composant de méthode dans une perspective « service ». L'objectif central de cette approche consiste à mieux définir la granularité, l'interopérabilité et la disponibilité d'un service méthode.
4. **L'approche Semantically AnnotaTed Intentions for Services** (SATIS) propose par Mirbel [Mirbel *et al.*, 2009], est une approche qui s'inscrit dans la famille des approches de recherche de service web par les buts. Cette approche permet aux utilisateurs non-informaticiens d'exprimer leurs besoins et de les aider à trouver les services web disponibles qui correspondent à leurs besoins.

Dans cette section nous avons présenté une introduction aux approches à base de services. Quatre approches prédominantes ont été citées. Nous les décrivons plus en détails dans les sections suivantes avec l'objectif de les comparer. Cette comparaison sera basée sur des critères de classification que nous présentons dans la section suivante.

4.3. Les critères de classification des approches

Cette section est consacrée à la classification et à la comparaison des approches du domaine de l'ingénierie des systèmes à base de services. Nous nous intéressons plus particulièrement aux processus d'ingénierie des besoins et des méthodes dans les approches à base de services.

Nous commençons par définir un cadre de référence permettant la caractérisation et l'analyse de ces approches. Ce cadre nous permettra de positionner notre approche par rapport aux travaux existants.

4.3.1. La définition des critères de classification des approches

Cette section présente la définition des critères de classification et comparaison des approches étudiées. Nous nous intéressons aux questions présentées dans le tableau ci-dessous.

Vue	Question	Critère
Pourquoi ?	Dans quel cadre utiliser l'approche ?	Domaine applicatif
Qui ?	Qui sont les clients de l'approche ?	Clients de l'approche, communication client-fournisseur
Quand ?	Quand utiliser l'approche ?	Portée de la mise en œuvre de l'approche, la technologie utilisée
Comment ?	A quels niveaux d'abstraction les services sont-ils définis ?	Les niveaux d'abstraction considérés
	Comment les approches sont-elles développées ?	Le type d'approche, le langage utilisé et sa notation, les outils supportant l'approche

TABLEAU 10. DEFINITION DES CRITERES DE CLASSIFICATION

Ces questions nous amènent à définir les critères suivants : le domaine applicatif de l'approche, la portée de la mise en œuvre de l'approche, la technologie utilisée, le type d'approche, les clients de l'approche, la communication client-fournisseur en utilisant les ontologies, les niveaux d'abstraction

considérés, la modélisation de l'approche et le formalisme d'expression, et finalement les outils supportant l'approche. Dans les paragraphes suivants, chacun des critères est présenté.

a) Le domaine applicatif de l'approche

Le domaine applicatif de l'approche à base de service doit répondre à la question : « Dans quel cadre utiliser l'approche ? ».

Un travail de développement a un ou plusieurs domaines d'application. L'utilisation des approches à base de services fait l'objet de méthodes essentiellement centrées sur : l'ingénierie des besoins, l'ingénierie des méthodes, la gestion des processus métier ou la construction d'environnements de modélisation.

b) Les clients de l'approche

Il s'agit de définir les clients de l'approche à base de services. Une approche à base de services peut être utilisée par un ou plusieurs utilisateurs stratégiques, métiers ou opérationnels.

Les clients stratégiques correspondent aux spécialistes de haut niveau. Ils ont une vision transversale des métiers, c'est-à-dire ils sont liés à la direction de l'organisation. Ils traitent des problèmes liés à la gestion du projet et à la coordination des différentes équipes. Ils expriment aussi des exigences liées aux orientations stratégiques de l'organisation.

Les clients métiers comprennent les décideurs liés à un métier spécifique. Il exprime principalement des exigences analytiques, fonctionnelles (règles de gestion, règles de calcul) ainsi que non fonctionnelles (ergonomie d'outil). Ces utilisateurs ont une vocation plus organisationnelle que les stratèges.

Les clients opérationnels sont liés aux sources de données existantes et aux services qui sont ou qui seront déployés. Les concepteurs de services exécutables font partie de ce groupe.

c) Communication Client- Fournisseur

Un autre critère que nous étudions est la communication entre clients et fournisseurs. Il s'agit d'identifier s'il existe un vocabulaire commun entre eux. Dans les approches étudiées, ce sont les ontologies qui fournissent un langage de description commun à tous les services pour faciliter la recherche et la spécification de services.

d) Portée de l'approche

La portée de la mise en œuvre de l'approche doit répondre à la question : « Quand utiliser l'approche ? ». Nous considérons les deux aspects suivants : lors de la conception, ou à l'exécution.

De nombreux chercheurs s'intéressent aux différentes approches utilisées pour la modélisation des systèmes d'information à base de services [Quartel *et al.*, 2004], [Papazoglou *et al.*, 2007], [Mecella *et al.*, 2002], [Henkel *et al.*, 2004]. Ce sont des approches à utiliser lors de la conception qui permettent de passer graduellement des processus métiers aux services exécutables à l'aide d'un ensemble de principes et de directives. [Penserini *et al.*, 2006] et [van Lamsweerde, 2000] s'intéressent aux approches guidées par les buts pour la modélisation des systèmes d'information. Les buts des utilisateurs sont identifiés et affinés jusqu'à l'obtention des services exécutables qui réalisent ces buts.

Pour les approches utilisées lors de l'exécution, il est nécessaire d'orchestrer l'enchaînement des services selon un canevas prédéfini, et de les exécuter à travers des "scripts d'orchestration". Ces scripts peuvent représenter des processus métier ou des workflows inter/intra-entreprise. Ils décrivent les interactions entre applications en identifiant les messages, et en déterminant la logique et les séquences d'invocation. L'orchestration décrit la manière selon laquelle les services peuvent interagir au moyen de messages, d'une logique métier et selon un ordre d'exécution des interactions. Ces interactions peuvent couvrir des applications et/ou des organisations.

e) La technologie utilisée

Dans le cas où la portée de la mise en œuvre correspond à l'exécution, nous sommes intéressés à connaître la technologie utilisée. Plusieurs implémentations de l'approche à services existent actuellement. Parmi eux, nous établions une distinction entre les approches basées sur des services génériques (par exemple : OSGi [OSGi, 2010], Jini [Jini, 2010], UPnP [UPnP, 2010]) et/ou celles basées sur des services web.

Les services web représentent actuellement l'ensemble de standards les plus connus pour la réalisation des applications Internet. Cette technologie repose sur des standards très populaires, tels que le XML ou le protocole HTTP. L'important intérêt porté aux services Web a conduit actuellement à une petite confusion qui est de rendre synonyme la notion de services Web avec la notion de services et d'architecture à services. Même si les standards développés autour des services Web sont très avancés et nombreux aujourd'hui, il s'agit seulement d'une implémentation particulière des principes de l'approche à services.

Les services génériques, la deuxième technologie que nous considérons, sont de plus en plus utilisés pour la réalisation des applications à services. A la différence de la technologie web, les services génériques prennent en compte la disponibilité dynamique des fournisseurs de services.

f) Niveaux d'abstraction considérés

La notion des niveaux d'abstraction répond à la question : « A quels niveaux d'abstraction les services sont-ils définis ? ». L'intérêt de cet aspect est de définir les niveaux de modélisation des approches. [Baida *et al.*, 2004] suggère trois perspectives sur les services : le niveau technologique, le niveau métier et le niveau applicatif (ou opérationnel). De plus, Kaabi [Kaabi, 2007] définit la perspective intentionnelle qui se traduit par le fait qu'un service exhibe une intentionnalité formulée par le but qu'il permet à ses clients d'atteindre [Rolland *et al.*, 2005].

- Le niveau intentionnel permet d'atteindre l'intention ou le but d'un acteur dans un contexte donné. Selon la granularité de l'intention, la réalisation de celle-ci peut se faire par l'exécution d'un service opérationnel ou par la composition de services métier dirigée par les intentions. L'avantage de ce niveau est qu'il offre une définition orientée but, compréhensible par les utilisateurs.
- Le niveau organisationnel (ou métier) [Pallos, 2001] considère un service comme un groupement logique de composants requis pour satisfaire une demande métier particulière. Le niveau d'abstraction métier pour un service est utilisé dans le domaine de la gestion pour faire référence à une activité ou un processus de gestion qui produit une valeur ajoutée ; les services sont offerts par un fournisseur à son environnement. Deux aspects doivent être considérés : 1) la capitalisation et la réutilisation des connaissances relatives au métier que doit servir le système d'information (par exemple, il doit être possible de réutiliser la description d'un processus de gestion de modèles) ; 2) les connaissances relatives au métier de concepteur/développeur de systèmes d'information (par exemple, la description du processus de construction d'une spécification de besoins doit pouvoir être réutilisée dans différents projets). La réutilisation de ces deux types de connaissances permet d'améliorer la qualité des produits mais aussi d'apporter un support méthodologique dans le processus de développement de systèmes d'information. Ici nous l'appellerons le niveau organisationnel.
- Le niveau applicatif permet d'établir un pont entre le niveau métier et le niveau technologique. Par exemple, [Quartel *et al.*, 2004] définit le service applicatif comme le service d'une application qui permet de supporter un processus métier. Cette définition oscille entre le niveau métier et le niveau technologique. Ici nous l'appellerons le niveau opérationnel.

- Le niveau d'abstraction technologique représente l'aspect exécutable d'un service. Il est influencé par les domaines proches de l'ingénierie du web et le développement de standards liés aux services web [Papazoglou, 2002]. Dans ce contexte, les services s'appuient sur le triplet de standards suivant : « Web Service Description Language » (WSDL), « Simple Object Access Protocol » (SOAP) et « Universal Description, Discovery and Interation » (UDDI).

g) Type d'approche

Les approches à base de services peuvent être de trois types : ascendantes, descendantes, mixtes.

- Les approches ascendantes [Hüsemann *et al.*, 2000] se concentrent sur la façon dont les données peuvent être extraites et transformées en modèles plus abstraits tout en étant guidé par les services exécutables.
- Les approches descendantes [Prat et al, 2002], [Prakash et al, 2003] considèrent à différents niveaux de granularité les exigences du système, les exigences utilisateurs comme étant un point de départ et laissent l'identification des processus organisationnels à une phase postérieure. Dans ce type d'approche, les besoins sont exprimés sous forme de requêtes. Les résultats des requêtes sont liés aux stratégies et processus pour résoudre les objectifs.
- Les approches mixtes sont caractérisées par l'analyse des exigences des utilisateurs (démarche descendante) et l'analyse des services exécutables (démarche ascendante).

h) Le langage utilisé et sa notation

La description d'un processus à base de services fait appel à des notations. Nous avons identifié deux types de notations : semi-formelle et formelle.

La notation semi-formelle, comme les diagrammes, introduit une approche basée sur l'utilisation de méta-modèles [Orriëns *et al.*, 2003], mais dont la sémantique n'est pas formellement définie.

La notation formelle se base sur l'utilisation d'une sémantique formelle bien définie qui permet d'une part, de vérifier les propriétés que la démarche doit remplir et, d'autre part, de valider le processus en utilisant des techniques comme par exemple l'instanciation, la simulation ou l'exécution [Berardi *et al.*, 2003].

Plus généralement, nous nous intéressons à caractériser le langage utilisé ainsi que son formalisme, qui peut être semi-formel ou formel.

i) Les outils de support

La notion d'outils de support répond à la question : « Comment les approches sont-elles développées ? ». Il s'agit de définir les outils de support et l'architecture d'exécution pour la mise en œuvre des approches à base de services.

En synthèse, le cadre de référence que nous avons défini permet de caractériser les approches à base de services. La table suivante synthétise les critères de comparaison définis.

Vue	Question	Critère	Valeurs
Pourquoi ?	Dans quel cadre utiliser l'approche ?	Domaine applicatif	L'ingénierie des besoins, l'ingénierie des méthodes, la gestion des processus métier, la construction d'environnements de modélisation
Qui ?	Qui sont les clients de l'approche ?	Clients de l'approche,	Stratégiques, métiers, opérationnels
		Communication client-fournisseur	Oui/Non (si oui, utilisation d'ontologies)
Quand ?	Quand utiliser l'approche ?	Portée de la mise en œuvre de l'approche	Conception, exécution
		La technologie utilisée	Générique, services web
Comment ?	A quels niveaux d'abstraction les services sont-ils définis ?	Les niveaux d'abstraction considérés	Intentionnel, organisationnel, opérationnel, technologique
	Comment les approches sont elles développées ?	Le type d'approche	Ascendante, descendante, mixte
		Type de la notation	Semi-formelle, formelle
		Le langage	Set of (langage(valeur), notation(valeur))
		Les outils supportant l'approche	Set of (outils)

TABLEAU 11. RESUME DU CADRE DE REFERENCE

4.4. L'analyse des approches à base de services

Cette section présente une évaluation de quatre approches à base de services :

1. **L'approche iSOA (Intentional Service Oriented Architecture)** [Kaabi, 2007].
2. **Service-Oriented Meta-Method** (SO2M) proposée par Guzélien [Guzélien, 2007].
3. **Method as Service** (MaaS) proposée par Rolland [Rolland, 2008].
4. **L'approche Semantically AnnotaTed Intentions for Services** (SATIS) proposée par Mirbel [Mirbel *et al.*, 2009].

4.4.1. L'approche iSOA

L'approche iSOA (intentional Service Oriented Architecture) [Kaabi, 2007] permet de découvrir et de décrire les services intentionnels nécessaires à une organisation. Dans cette approche, les services intentionnels correspondent aux besoins qui permettent de satisfaire les buts organisationnels. Par homogénéité entre la notion de service intentionnel et le processus d'élicitation des services du business, ISOA propose de modéliser les processus métier de l'entreprise dans une perspective elle-même intentionnelle. Donc, ISOA propose un déplacement de toute la perspective SOA « centrée fonction » vers une position équivalente mais « centrée intention » (Figure 33).

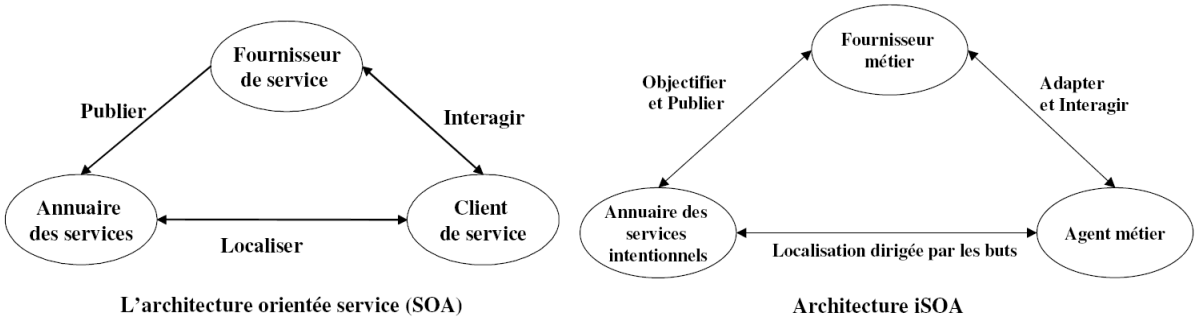


FIGURE 35. LES ARCHITECTURES SOA [PAPAZOGLOU ET AL., 2005] ET ISOA [KAABI, 2007]

Dans cette perspective, les services sont décrits en termes d'intentions et de stratégies pour les atteindre, les publier, les rechercher les composer.

L'architecture iSOA hérite des mêmes rôles et opérations que SOA mais s'en différencie par les deux point suivants :

- les agents métiers remplacent les agents logiciels au niveau des interactions, et
- la description intentionnelle du service remplace la description technique.

Dans la vision iSOA, les organisations qui offrent des e-services centrés business, (donc, les fournisseurs métier) les décrivent dans un mode intentionnel (basé d'intentions et de stratégies). La

publication de ces services est réalisée en utilisant un annuaire de services intentionnels. L'agent métier (le client) qui recherche les services coïncidant avec ses exigences effectue une découverte dirigée par les intentions. Une fois le service localisé, l'agent métier peut interagir directement avec le fournisseur métier et exécuter les services par adaptation aux conditions spécifiques du moment.

Parmi les contributions de cette approche, nous nous intéressons ici à deux résultats, qui sont :

- Un modèle intentionnel de représentation des services : le modèle MIS,
- Un modèle opérationnel de services : le modèle MOS.

Les sections suivantes décrivent ces deux modèles.

4.4.1.1. Le modèle MIS

Le **Modèle Intentionnel de Services (MIS)** permet de représenter, à un niveau intentionnel, les services du système à développer et d'aider un client à exprimer son intention et se concentrer sur les buts qu'un service permet de réaliser.

Le modèle MIS définit chaque service intentionnel en tant que brique de construction d'applications visibles au travers de son interface qui apporte la connaissance situationnelle et intentionnelle. Chaque service intentionnel s'applique dans une situation particulière pour réaliser une intention particulière.

La figure 36 présente le méta-modèle MIS. Dans cette figure nous ne détaillons que le concept central du modèle, celui de service intentionnel (appelé service dans la figure). Le point de vue intentionnel se traduit par le fait qu'un service exhibe une intentionnalité formulée par le but qu'il permet à ses clients d'atteindre [Rolland, 2005].

Le modèle MIS classe les services intentionnels en atomiques et agrégats. Un **service atomique** n'est pas décomposable en d'autres services intentionnels alors qu'un service agrégat l'est.

Un service atomique est associé à un but que l'on qualifie « d'opérationnalisable », c'est-à-dire pour lequel on est capable de définir une séquence d'actions qui permet de réaliser le but. Donc, ce sont les actions qui composent le service atomique qui permettent d'assurer l'opérationnalisation du but du service atomique. Cela permet de dire qu'un service atomique est exécutable. Par exemple, le service permettant de Payer une réservation par carte de crédit est un service atomique car le but qui lui associé est opérationnalisable dans le sens où l'on sait définir un processus de paiement d'une réservation par carte de crédit.

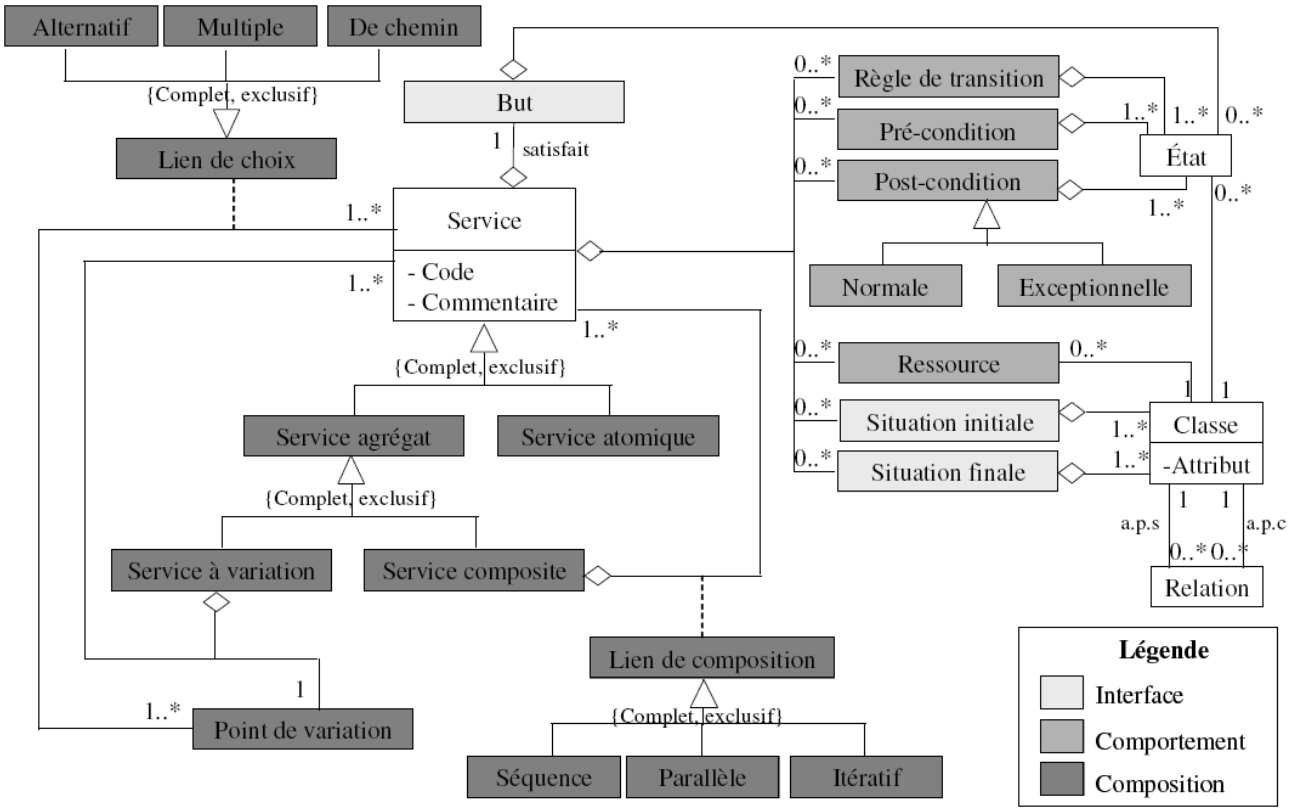


FIGURE 36. META-MODELE MIS [KAABI, 2007]

Les buts de haut niveau tels que les buts stratégiques ne sont pas d'emblée « opérationnalisables ». Ils nécessitent d'être décomposés en sous buts qui eux-mêmes peuvent nécessiter une décomposition jusqu'à atteindre des buts opérationnalisables. **Les services agrégats** sont associés à des buts de haut niveau. La composition d'un service agrégat en services suit la décomposition du but en sous buts. Cette composition est calquée sur l'affinement ET/OU du but, représenté par des graphes ET/OU. La composition d'un service intentionnel introduit deux types de services pour cette raison : ceux qui sont justifiés par une décomposition OU du but « les variantes » et ceux qui correspondent à une décomposition ET « les composites ».

Le modèle MIS introduit la variabilité dans la manière de réaliser le but d'un service. Les variantes correspondent aux différentes manières d'atteindre le but. Etant donné qu'un service intentionnel peut être composé d'autres services intentionnels, ayant chacun son propre but et par conséquent des variantes associées, il en résulte que le service intentionnel est défini comme un réseau de variantes attachées à des points de variation.

Enfin, le modèle MIS introduit la composition de services dirigée par les buts et elle se situe à l'échelle stratégique et intentionnelle des clients. On entend par composition de services dirigée par les buts le fait que le service composite est associé à un but de haut niveau et sa composition suit la décomposition du but père en sous buts.

L'approche iSOA définit un processus pour construire un modèle MIS à l'aide d'une carte MAP. Le modèle de la carte MAP est adapté à la découverte des services. En effet, il met en évidence les

différentes stratégies pour satisfaire le même but et aussi les différentes combinaisons de stratégies et de buts pour atteindre un but. MAP aide aussi à la découverte de la variabilité des services. Chaque combinaison possible de buts et de stratégies identifie une variante de services possible pour atteindre le but du service global. L'approche iSOA considère que les services sont en relation avec les objectifs du business et, manifestement, aident à les atteindre. Il semble donc naturel de donner au fournisseur métier les moyens de construire un modèle du business à partir duquel il soit aisé d'identifier les services, de les décrire selon les termes de MIS et de les publier. L'approche iSOA utilise le modèle de la carte MAP pour modéliser le business dans des termes intentionnels et de développer un ensemble de règles méthodologiques pour dériver les services d'une carte intentionnelle. Les différents concepts du modèle de services sont identifiés à partir d'une carte MAP.

De plus, l'approche iSOA propose une architecture agents pour guider les développeurs dans l'orchestration et l'implémentation des services du modèle MIS. Les services sont implémentés sous la forme de composants architecturaux appelés agents et sont structurés hiérarchiquement. Chaque agent est responsable de la satisfaction du but du service associé.

4.4.1.2. Le modèle MOS

Le modèle MIS, présenté dans la section précédente, classe les services intentionnels en deux catégories : agrégat et atomique. Le service intentionnel atomique est directement opérationnalisable par l'exécution d'un ou plusieurs éléments logiciels.

Un service intentionnel atomique du modèle MIS est opérationnalisé par l'exécution d'un service logiciel. Il y a donc une relation bidirectionnelle entre le modèle MIS et le modèle MOS : (i) Relation causale de MIS vers MOS qui représente les services logiciels à utiliser : le service intentionnel atomique du modèle MIS est opérationnalisé par un service logiciel du modèle MOS. (ii) Relation causale de MOS vers MIS : le service logiciel délivre un résultat considéré comme l'information nécessaire pour que le but du service atomique soit réalisé.

Cette relation bidirectionnelle des modèles MIS et MOS permet de faire la transition entre la notion de service à un niveau intentionnel et celle à un niveau opérationnel. Ceci met en avant le fait que les deux notions de service coexistent et coopèrent à la fois au niveau intentionnel et au niveau opérationnel.

Dans le cadre de l'ingénierie des systèmes à base de services, les applications métier peuvent être réalisées par la réutilisation et la coordination de plusieurs services métier. Par conséquent, le modèle MOS propose, à travers le concept de service logiciel, le concept de **service métier** pour représenter l'unité logicielle qui est fournie par une organisation et le concept de **service de coordination** pour représenter l'application métier construite par composition de services métier et qui est invoquée par le **service d'interface utilisateur**. La figure 37 montre que la description d'un

service logiciel assemble trois types d'éléments logiciels correspondant au service d'interface utilisateur, au service de coordination métier et au service métier.

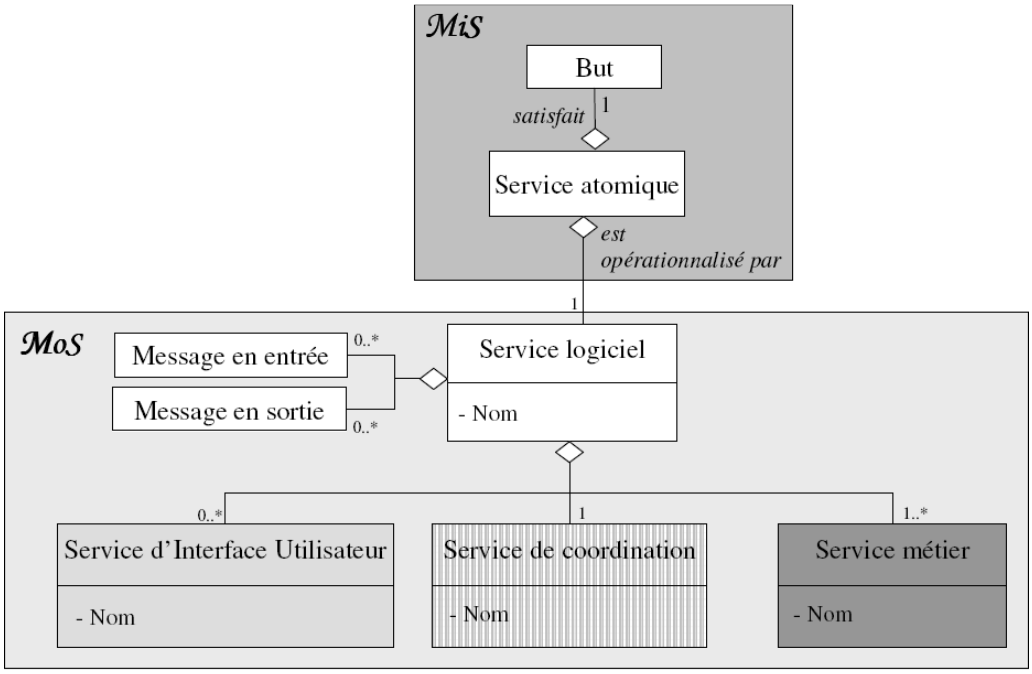


FIGURE 37. LE META-MODELE MOS [KAABI, 2007]

Le service logiciel est le moyen d'opérationnaliser le service intentionnel atomique. L'exécution du service logiciel permet d'interagir avec le service d'interface utilisateur, qui délègue au service de coordination la réalisation d'une transaction métier complexe et distribuée, qui à son tour, délègue aux services métier la réalisation de transaction métier. A la fin de cette chaîne de délégation, la réalisation du but se traduit par le fait que le service d'interface délivre le résultat du service logiciel.

Le service d'interface utilisateur représente la spécification nécessaire au développement de l'interface de l'application. Le service d'interface utilisateur a comme objectif la gestion du dialogue avec l'utilisateur pour que le service logiciel l'aide à atteindre son but.

Le service métier décrit les transactions métier. La spécification ne décrit pas la manière dont les transactions métier doivent être implémentées mais elle s'attache à décrire comment elles sont utilisées indépendamment de leur implémentation. Un service métier est fourni par un agent et est décrit à l'aide d'une interface. C'est la partie visible qui permet de comprendre la capacité du service sans entrer dans le corps de celui-ci.

Le service de coordination permet, d'une part, de coordonner l'ensemble des services métier à travers un ensemble d'activités ordonnées et, d'autre part, de rendre le service d'interface utilisateur indépendant des services métier mis en œuvre.

La table suivante synthétise l'évaluation de l'approche iSOA.

Vue	Question	Critère	iSOA [Kaabi, 2007] Valeurs
Pourquoi ?	Dans quel cadre utiliser l'approche ?	Domaine applicatif	L'ingénierie des besoins
Qui ?	Qui sont les clients de l'approche ?	Clients de l'approche,	Métiers, opérationnels (Spécialistes métier, concepteurs/développeurs)
		Communication client-fournisseur	Non
Quand ?	Quand utiliser l'approche ?	Portée de la mise en œuvre de l'approche	Conception
		La technologie utilisée	Générique
Comment ?	A quels niveaux d'abstraction les services sont-ils définis ?	Les niveaux d'abstraction considérés	Intentionnel, opérationnel
	Comment les approches sont elles développées ?	Le type d'approche	Descendante
		Type de la notation	Semi-formelle
		Le langage	MAP, arbres de décomposition ET/OU
		Les outils supportant l'approche	Aucune

TABLEAU 12. RESUME DU CADRE DE REFERENCE POUR L'APPROCHE ISOA

L'approche iSOA concerne l'ingénierie des besoins. L'approche est descendante et utilise un formalisme semi-formel basé sur des arbres de décomposition ET/ OU. Donc on peut en conclure que l'approche iSOA de Kaabi [Kaabi, 2007] est utilisée lors de la conception et s'appuie sur une technologie générique pour spécifier les services.

4.4.2. L'approche SO2M

SO2M [Guzélien, 2007] est une approche qui permet de construire des méthodes adaptées aux besoins des concepteurs/développeurs. SO2M propose un méta-modèle de type modulaire qui permet de définir et de rendre réutilisable un ensemble de services, appelés « services méthode » - qu'il est possible de composer de manière dynamique pour construire des méthodes particulières adaptées à un contexte donné. Donc, les services méthodes sont des unités réutilisables offrant des fragments de méthode, ils sont disponibles et ils peuvent être composés pour répondre aux besoins des concepteurs/développeurs de systèmes d'information. Cette solution facilite la flexibilité dans

l'usage des méthodes, une adaptation aux besoins des concepteurs/développeurs et la capitalisation et la réutilisation des connaissances mises en œuvre dans le développement de systèmes d'information. Deux types de connaissances sont réutilisés :

1. Les connaissances de domaine, il s'agit des connaissances relatives au métier pour lequel le système d'information est développé. Par exemple, il est possible d'identifier, de modéliser et de rendre accessible des connaissances sur le domaine bancaire. Ces connaissances pourront être utilisées au moment de la construction d'un SI pour une banque particulière.
2. Les connaissances relatives aux pratiques des développeurs, il s'agit des connaissances relatives au métier du développeur. Ces connaissances peuvent aussi être identifiées, formalisées et réutilisées. Par exemple, la tâche d'identification des cas d'utilisation peut être décrite par une démarche générale et réutilisable dans de nombreuses situations.

Dans SO2M, les composants, appelés services, permettent la réutilisation de ces deux formes de connaissances. Ainsi, SO2M met à la disposition des concepteurs/développeurs des fragments de démarche, des formalismes de représentation avec des règles d'utilisation, des modèles de conception mais aussi des fragments de modèle de domaine ou encore des représentations de processus métier. Par ailleurs, une méthode ou un fragment de méthode, construit de toute pièce avec SO2M, peut être à son tour considéré comme un composant en vue d'être réutilisé. La figure 38 résume les finalités de SO2M.

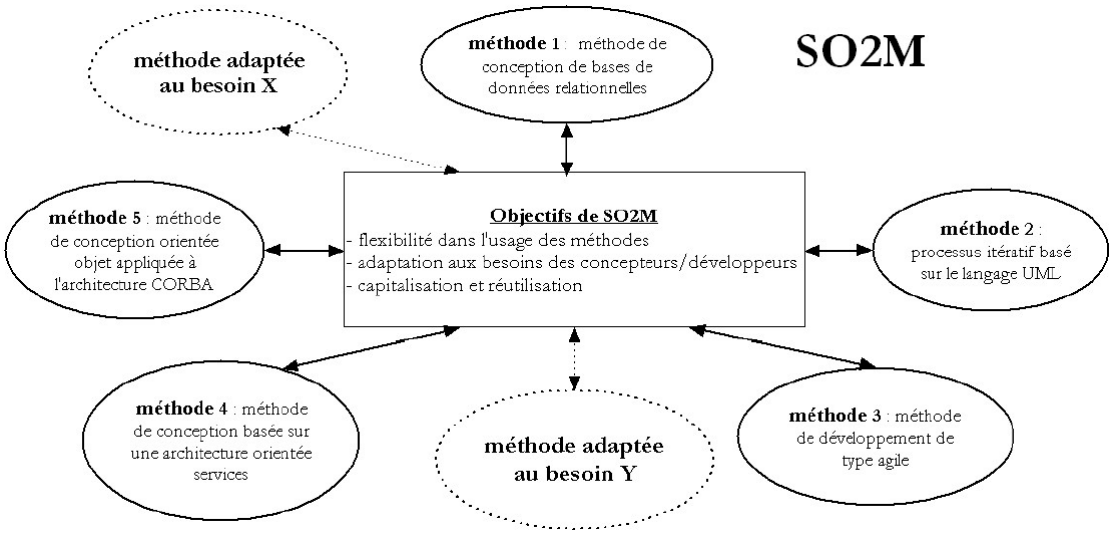


FIGURE 38. LES FINALITES DE SO2M [GUZELIAN, 2007]

SO2M permet de définir l'ensemble des connaissances nécessaires à la description et à l'utilisation des services méthode. Les services méthode sont des unités réutilisables qui proposent un ou plusieurs fragments de démarches pour résoudre des problèmes de développement de systèmes

d'information. Ils permettent ainsi de répondre aux besoins de capitalisation et de réutilisation des connaissances en ingénierie des SI.

Dans SO2M les services méthodes sont définis comme des composants de type processus, ces composants définissent des fragments de méthode, que l'on peut assembler pour définir des nouvelles méthodes. L'orientation service permet d'associer à ces composantes deux propriétés essentielles :

1. Leur description est centrée sur l'usage et donc sur le problème auquel ils répondent. Les services méthode sont centrés sur des problèmes d'ingénierie des SI.
2. Leur composition est dynamique et mise en œuvre par l'orientation but et par des mécanismes d'abstraction utilisés dans la spécification des services, c'est-à-dire que l'assemblage est effectué en tenant compte de chaque besoin. Chaque nouveau besoin est résolu en recherchant et en sélectionnant les services méthode les plus adaptés à ce besoin et à son contexte. La construction d'une méthode ou d'un fragment de méthode répondant aux besoins d'un projet consiste à rechercher et à composer les services méthodes qui contribuent à la satisfaction des besoins. Les liens entre les services sont identifiés et établis en fonction des besoins, c'est-à-dire de manière dynamique au moment où l'on cherche à résoudre un problème de développement. Les liens entre les services sont construits en fonction des besoins, une telle approche autorise de nombreuses possibilités d'assemblage de services. Chaque assemblage de services est construit en fonction du contexte et correspond à un processus personnalisé.

La figure 39 schématise les services méthode. Chaque méthode est élaborée en fonction d'un besoin dans un contexte particulier. Chaque service « méthode » peut être considéré comme un fragment de méthode permettant de réaliser un certain but. Les services méthodes sont considérés comme des composants auxquels sont associés, d'une part, un niveau de description exprimant toutes les connaissances liées à leur usage avec une orientation but exprimée sous la forme de modèles de la carte MAP et d'autre part, un mécanisme de composition permettant de les assembler en fonction des besoins.

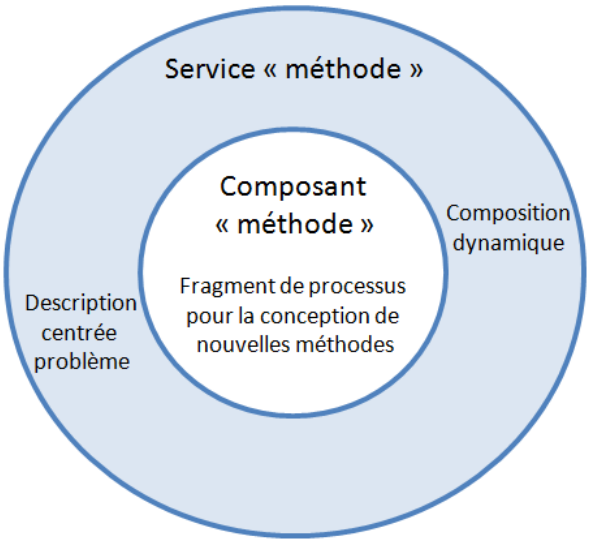


FIGURE 39. SERVICE METHODE ET COMPOSANT METHODE [GUZELIAN, 2007]

L'orientation but permet, d'une part, de mettre l'accent sur l'usage des services et d'autre part, d'envisager et de spécifier plusieurs manières (et donc plusieurs services) pour réaliser un même but. Le concept de but peut être vu comme une forme d'abstraction regroupant tous les processus permettant de le satisfaire. Les différents processus représentent les différentes manières possibles de réaliser le même but. Les différentes manières de réaliser un but permettent d'adapter les solutions pour satisfaire le but. De plus, l'orientation but du modèle de services favorise l'intégration de méthodes hétérogènes, facilite leur recherche et leur composition et permet réduire la distance entre les requêtes des concepteurs/développeurs et les services disponibles. Les utilisateurs de cette approche sont tous les acteurs qui ont des problèmes de développement à résoudre ; il peut s'agir de problèmes de spécification de besoins, de conception d'architecture, d'implémentation… L'usage de SO2M est présenté par un diagramme de cas d'utilisation (figure 40). Chaque cas d'utilisation représente une fonctionnalité de SO2M.

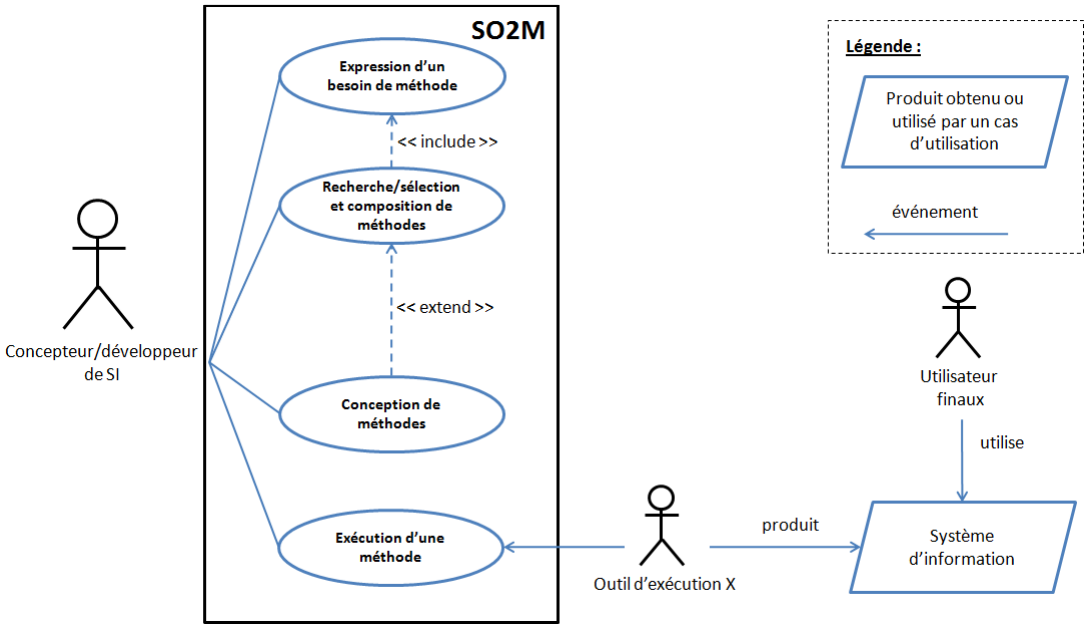


FIGURE 40. LES USAGES DE SO2M [GUZELIAN, 2007]

Le cas d'utilisation « **Expression d'un besoin de méthode** » permet à un concepteur/développeur de SI d'exprimer une requête sous la forme d'un problème de développement de SI à résoudre. Par exemple, « construire un diagramme d'activités », « exprimer les exigences du SI avec un modèle des cas d'utilisation »...

Le cas d'utilisation « **Recherche/sélection et composition de méthodes** » a pour enjeu de satisfaire le besoin exprimé en recherchant des solutions parmi les méthodes disponibles. La satisfaction d'un besoin peut nécessiter plusieurs méthodes qu'il convient alors d'assembler. A ce niveau, le concepteur/développeur obtient une méthode ou un fragment de méthode qu'il peut exécuter.

Le cas d'utilisation « **Conception de méthodes** » consiste à enrichir SO2M de nouvelles méthodes. La conception d'une nouvelle méthode peut être effectuée à partir de la méthode ou du fragment de méthode obtenu par le cas d'utilisation précédent.

Le cas d'utilisation « **Exécution d'une méthode** » consiste à exécuter la méthode obtenue. Ce cas d'utilisation est réalisé par un outil d'exécution de processus. Cet outil est externe à SO2M ; il peut par exemple être un outil de type « workflow » qui gère et exécute des processus dont l'ordre des activités est prédéfini.

Le méta-modèle SO2M

Le méta-modèle SO2M (figure 41) est basé sur l'utilisation de deux types de services : des services externes et des services « méthode ».

Les services externes correspondent à toute forme d'information disponible sur le Web et ayant un intérêt dans le développement des systèmes d'information. Ces services peuvent être des services Web [Maesano *et al.*, 2003], des composants EJB [Heineman *et al.*, 2001], des composants CORBA [Heineman *et al.*, 2001], des composants de méthode [Ralyté, 2001], des composants de type « paquet de configuration » [Karlsson, 2002], des patrons de conception [Fowler, 1996]... mais aussi des documents types disponibles sur le Web.

Les services « méthode » sont des services internes, c'est-à-dire des services existants dans la base de services du méta-modèle SO2M. Ils sont spécifiés par trois parties : identifiant, processus et ressource. **La partie identifiant** spécifie l'objectif du service méthode et le contexte d'utilisation. **La partie processus** fournit un processus pour réaliser l'objectif du service méthode. **La partie ressource** décrit le processus de manière opérationnelle. Par la suite, nous détaillerons la partie identifiant.

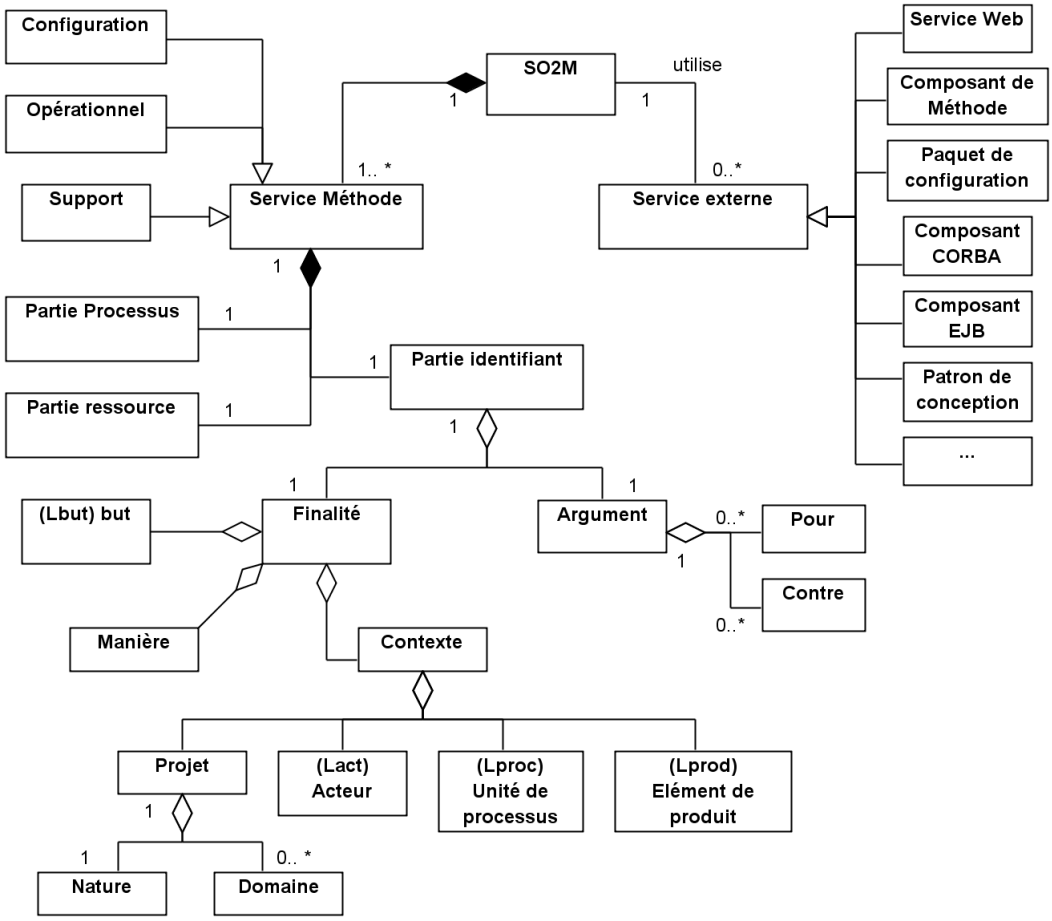


FIGURE 41. META-MODELE SO2M [GUZELIAN, 2007]

La partie identifiant précise ce que fait le service en indiquant le but que celui-ci permet de résoudre. Cette partie est essentielle au moment de la recherche d'un service. L'identifiant d'un service « méthode » comprend deux types d'informations (figure 41) : la finalité et les arguments.

La finalité permet de définir le problème auquel le service répond. Elle est décrite par trois éléments : le but, la manière et le contexte.

- **Le but** spécifie l'objectif et les fonctionnalités d'un service méthode. La description du but repose sur une ontologie des buts (voir annexe B). Par exemple, « Spécifier les besoins ».
- **La manière** précise comment le but est réalisé. Par exemple, Le but « Spécifier les besoins » peut être réalisé de plusieurs façons. Chaque but associé à une manière correspond à un service. Donc, la spécification des besoins peut être réalisée, par les fonctions, avec les cas d'utilisation, avec une approche orientée but, Business Modeling, ou d'une autre manière. Les manières de réaliser ce but donnent des solutions de natures différentes. Ceci engendre des services « méthode » différents : « Spécifier les besoins avec les cas d'utilisation », « Spécifier les besoins par une approche orientée but »...
- **Le contexte** définit la situation dans laquelle le service méthode peut être utilisé. Il précise dans quels cas l'utilisation du service est appropriée. La définition du contexte fait référence aux « **quatre P** » du processus unifié UP [Jacobson *et al.*, 1999] : le Projet, les Personnes, le Processus et les Produits. Ces quatre dimensions permettent de caractériser toute situation de projet. Elles correspondent respectivement au type de projet pour lequel le service est approprié, aux personnes qui utilisent le service pour satisfaire un but, à l'étape du processus où le service peut être utilisé et au résultat obtenu à l'exécution du processus fourni par le service. Les acteurs, l'unité de processus et les éléments de produit sont définis par les ontologies d'acteurs (voir annexe C), processus (voir annexe D) et produits (voir annexe E). Ces ontologies contribuent à l'automatisation de la recherche et de la composition de services en facilitant la mise en correspondance entre les services « méthode » et les besoins des concepteurs/développeurs.

Les arguments (figure 41) fournissent les avantages (par exemple : les arguments « pour ») et les inconvénients (par exemple : les arguments « contre ») de l'utilisation d'un service méthode. Ainsi, la notion d'argument facilite le choix du service le mieux adapté à la réalisation d'un but ou d'un besoin d'un concepteur/développeur donné. Par exemple, les arguments « pour » peuvent préciser que le service est configurable et facile à utiliser. Les arguments « contre » peuvent indiquer que le temps de réalisation est important et que le service utilise des techniques complexes.

Finalement, le méta-modèle SO2M a été mis en œuvre par le prototype SO2MX. Ce prototype permet la conception et la manipulation des services « méthode ». Il est accessible à l'adresse http://gwladysg.free.fr/ Cette plate-forme logicielle offre un ensemble de fonctionnalités permettant de mettre en œuvre le processus de composition de services. Le développement de ce prototype est basé sur une architecture à trois niveaux et utilise le langage XML pour les services ainsi que le langage OWL pour les ontologies.

La table suivante synthétise l'évaluation de l'approche SO2M.

Vue	Question	Critère	SO2M [Guzélian, 2007] Valeurs
Pourquoi ?	Dans quel cadre utiliser l'approche ?	Domaine applicatif	L'ingénierie des méthodes
Qui ?	Qui sont les clients de l'approche ?	Clients de l'approche,	Stratégiques, métiers
		Communication client-fournisseur	Ontologies de buts, d'acteurs, de processus, de produits
Quand ?	Quand utiliser l'approche ?	Portée de la mise en œuvre de l'approche	Conception, exécution
		La technologie utilisée	Générique
Comment ?	A quels niveaux d'abstraction les services sont-ils définis ?	Les niveaux d'abstraction considérés	Intentionnel, organisationnel
	Comment les approches sont elles développées ?	Le type d'approche	Descendante
		Type de la notation	Semi-formelle
		Le langage	Modèle de Services (graphe de composition, XML en utilisant une DTD)
		Les outils supportant l'approche	Plateforme SO2MX

TABLEAU 13. RESUME DU CADRE DE REFERENCE POUR L'APPROCHE SO2M

SO2M est une approche à base de services pour l'ingénierie des méthodes. L'approche SO2M de Guzélian [Guzélian, 2007] fournit une plateforme pour la recherche et la composition de services. Cette approche est descendante et utilisée lors de la conception et l'exécution. Les clients de l'approche sont des concepteurs stratégies et métiers qui communiquent en utilisant des ontologies du domaine de l'ingénierie des systèmes d'information.

4.4.3. L'approche MaaS

L'approche « Method as a Service » (**MaaS**) proposée par Rolland [Rolland, 2008] consiste à concevoir une méthode comme un service. L'approche vise à adapter les technologies des services web aux besoins de l'ingénierie de méthodes, afin de faciliter la construction rapide de méthodes adaptées aux préférences des concepteurs de modèles et des situations de projet. Dans cette perspective, une méthode devient des services méthode qui sont mis en œuvre sous la forme des

services web. Il s'agit d'offrir une solution pour réorganiser les méthodes dans un portefeuille de services méthodes qui sont distribués via le réseau Internet.

Malgré une acceptation commune d'une vision modulaire de méthodes, MaaS propose d'adopter un paradigme à base de services par analogie à l'approche « Software as a Service » (SaaS) [Papazoglou, 2003]. Il s'agit d'utiliser l'Architecture Orientée Service (dont le sigle en anglais s'appelle : SOA) et la définition du fragment de méthode appelée : « method chunk ». Cela mène à la notion de MOA (Method Oriented Architecture).

La figure 42 montre les trois acteurs et leurs interactions dans le MOA : le fournisseur de méthode, l'annuaire de méthode et le client de méthode.

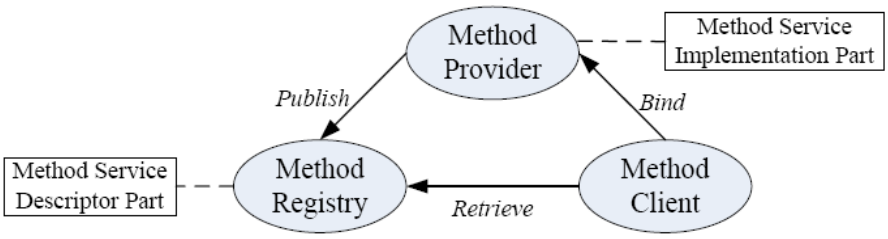


FIGURE 42. METHOD ORIENTED ARCHITECTURE [ROLLAND, 2008]

Le fournisseur de méthodes crée des services de méthode et publie leurs descripteurs sur l'annuaire de méthodes. **Le client de méthodes** récupère des services de méthode de l'annuaire, en utilisant des mécanismes de récupération construits avec des descripteurs sémantiques. Des descripteurs opérationnels (WDSL) sont utilisés par les clients de méthode pour invoquer la partie de l'implémentation de services de méthode de leur fournisseur. Cette implémentation d'un fragment de processus est soit un service web atomique soit une composition de services web réalisée par un processus BPEL.

Dans MOA, chaque méthode de services doit être considérée comme un composant autonome, qui doit être découvert et sélectionné de façon dynamique. Une méthode de service peut alors être vue et exécutée comme une méthode. Pour être conforme à cette exigence, l'approche utilise la notion de "***method chunks***" [Ralyté *et al.*, 2001a] pour deux raisons :

1. l'intentionnalité (le processus de décomposition et de récupération), le descripteur et l'interface d'une « method chunk » sont utilisés pour décrire la partie intentionnelle du service de méthode. Cette partie sera utilisée pour récupérer et sélectionner les services de méthode dans l'annuaire de services.
2. la récursivité, une « method chunck » utilise une décomposition intentionnelle. Pour décomposer un objectif principal en des objectifs plus simples, une method chunk permet une décomposition de fragments de méthode logiquement reliés les uns aux autres et toujours

décrits sur la même manière. Cette description récursive d'une « method chunk » permet l'implémentation des services selon le principe du processus de composition de MOA.

La figure 43 montre la structure d'un service de méthode. Cette figure montre que la structure d'un service de méthode combine une partie de description avec une partie d'implémentation.

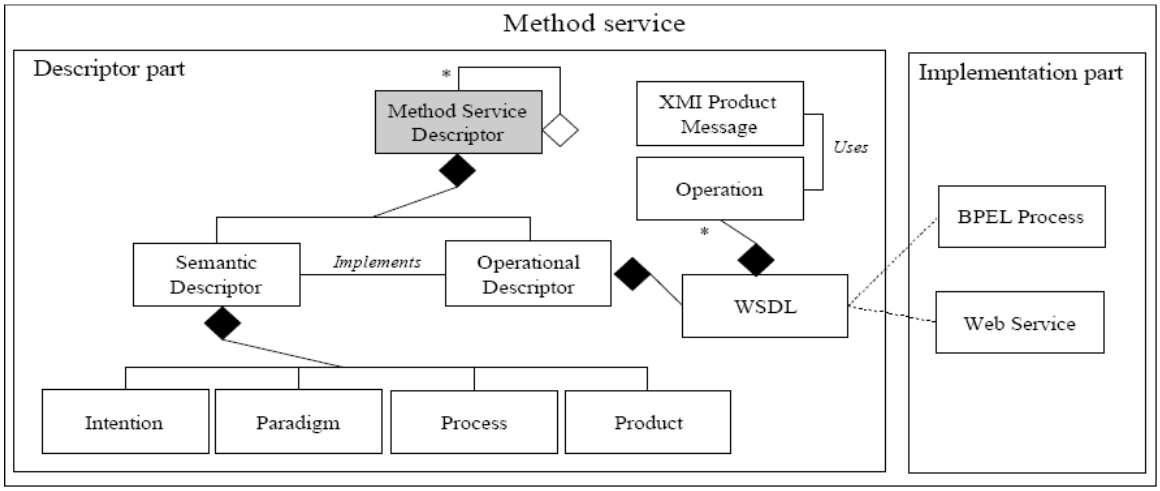


FIGURE 43. METHOD ORIENTED ARCHITECTURE [ROLLAND, 2008]

Le descripteur sémantique décrit le fragment de processus mis en œuvre par le service. Le but principal de ce descripteur est de documenter les services de méthode par quatre sous-parties : Intention, Paradigme, Processus et Produit. La récupération et la composition de fragments sont faites par des intentions.

L'intention définit les intentions de l'utilisation de services méthode et le contexte dans lequel ils peuvent être réutilisés. **Le paradigme** décrit l'avis du fragment. **Le processus** est la description d'activités exécutées sur les produits d'entrée. **Le produit** est la description du méta-modèle d'entrée et de modèles de produits de sortie du service de méthode.

L'opérationnalisation de services est exécutée par un descripteur opérationnel et une partie de l'implémentation.

L'opérationnalisation met en œuvre le processus décrit dans la partie sémantique par un service Web ou une composition de services Web (sous la forme d'un processus BPEL) exposé par un descripteur WSDL. Le service Web mis en œuvre est un outil fournissant la façon de supporter des services de méthode. La dimension produit est mise en œuvre par des méta-modèles conformes à MOF et le schéma XMI.

Le standard WSDL est le descripteur opérationnel d'un service de méthode. Il contient les définitions de chaque opération exécutée incluant leurs entrées et des messages de sortie (*XMI Product Message*).

La table suivante synthétise l'évaluation de l'approche MaaS.

Vue	Question	Critère	MaaS [Rolland, 2008] Valeurs
Pourquoi ?	Dans quel cadre utiliser l'approche ?	Domaine applicatif	L'ingénierie des méthodes
Qui ?	Qui sont les clients de l'approche ?	Clients de l'approche,	Métiers, opérationnels
		Communication client-fournisseur	Non
Quand ?	Quand utiliser l'approche ?	Portée de la mise en œuvre de l'approche	Conception, exécution
		La technologie utilisée	Services web
Comment ?	A quels niveaux d'abstraction les services sont-ils définis ?	Les niveaux d'abstraction considérés	Organisationnel, opérationnel
	Comment les approches sont elles développées ?	Le type d'approche	Descendante
		Type de la notation	Semi-formelle
		Le langage	XMI-XML, WSDL, UDDI, BPEL
		Les outils supportant l'approche	Aucune

TABLEAU 14. RESUME DU CADRE DE REFERENCE POUR L'APPROCHE MAAS

4.4.3. L'approche SATIS

L'approche SATIS (Semantically AnnotaTed Intentions for Services) [Mirbel *et al.*, 2009] s'inscrit dans le courant des approches de recherche de services web par les buts. SATIS a pour objectif principal d'offrir à des utilisateurs finaux non-informaticiens de décrire leurs démarches de recherche de services web pour opérationnaliser un service métier. Cette approche permet aux utilisateurs finaux d'exprimer leurs besoins (sous forme d'intentions ou buts et de stratégies mises en œuvre pour atteindre ces intentions) et de les aider à trouver les services web disponibles qui correspondent à leurs besoins. Cette approche se focalise sur un domaine métier particulier pour lequel des connaissances du domaine et des descriptions de services web sont disponibles : par exemple un domaine où des services web sémantiques (décrits à l'aide d'annotations sémantiques) et une ou plusieurs ontologies sont disponibles.

SATIS propose un environnement permettant de mettre à la disposition des membres d'une communauté des fragments réutilisables de démarche pour implémenter leur(s) but(s) métier(s). Pour

cela, cette approche s'appuie sur la spécification de haut niveau d'activités métiers à l'aide d'un modèle intentionnel dans le but d'en dériver des spécifications de services web. Cette approche s'intéresse également au partage de ces fragments de démarche dans une communauté d'utilisateurs qui partagent un même intérêt pour un sujet ou un problème et collaborent sur une période prolongée pour partager des idées, trouver des solutions et construire des innovations dans le contexte d'un domaine d'application particulier.

SATIS s'appuie sur les modèles et langages du Web sémantique comme un cadre unifié pour représenter (i) les besoins intentionnels de haut niveau des utilisateurs finaux, (ii) les patrons de spécifications de services web et (iii) les spécifications des services web. En ce qui concerne les besoins intentionnels de haut niveau des utilisateurs finaux, l'approche a adapté le modèle de la carte MAP [Rolland, 2007] en rassemblant ses éléments dans une ontologie dédiée à la représentation des processus intentionnels [Corby *et al.*, 2009] ; Cette ontologie est spécifiée en RDFS [W3C, 2004]. Les spécifications de services web sont représentées à l'aide de l'ontologie OWL-S [OWL-S, 2005]. Enfin, les patrons de spécifications de services web sont spécifiés à l'aide du langage de requête pour RDF SPARQL [W3C-SPARQL, 2009]. Ainsi les patrons de spécification de services web sont modélisés par des patrons de graphes qui sont projetés sur les graphes des annotations de services web.

La figure 44 montre les cinq phases de l'approche SATIS. Ces phases seront discutées plus en détail dans la suite de ce document. L'approche s'appuie sur un vocabulaire commun et sémantique. Elle s'appuie sur :

- Une ontologie dédiée aux processus intentionnels (la MAP étendue pour les services Web),
- Une ontologie dédiée au domaine métier (ou « *Business Domain* »),
- Une ontologie pour la description des services Web, comme OWL-S [OWL-S, 2004] par exemple.

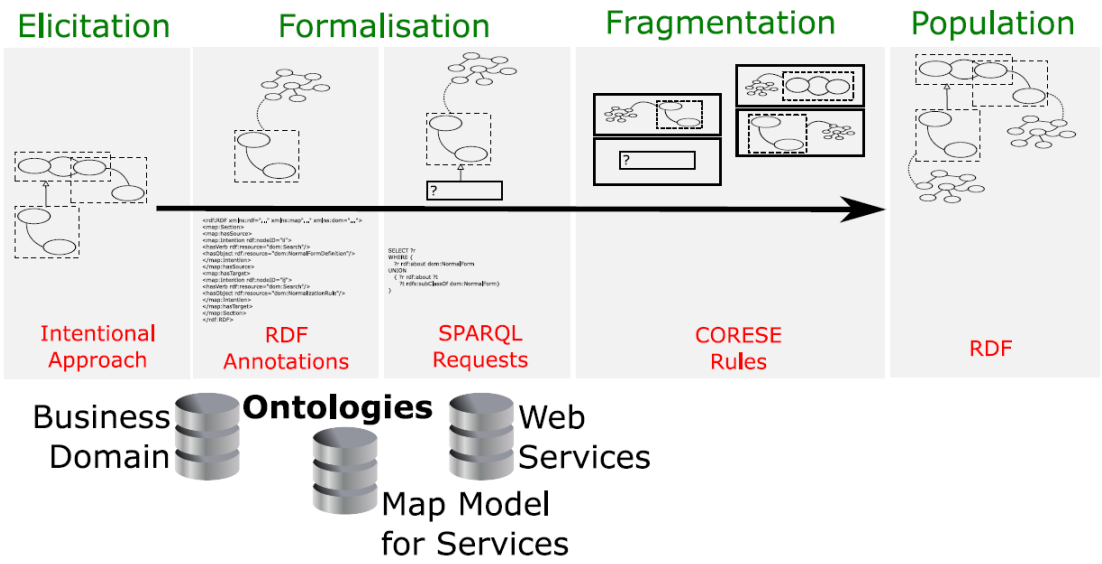


FIGURE 44. L'APPROCHE SATIS [MIRBEL ET AL., 2009]

Au cours de **la phase d'élicitation,** les utilisateurs finaux définissent leurs fragments de démarches en décrivant les intentions et les stratégies. La figure 45 montre un exemple de besoin intentionnel de haut niveau des utilisateurs finaux. Cet exemple porte sur une démarche pour rechercher des informations sur les bases de données relationnelles, par exemple dans le cadre de la conception d'un cours sur ce sujet dans une communauté d'enseignants. Nous pouvons remarquer que l'intention de recherche de ressources sur l'historique des bases de données relationnelles est facultative (il existe deux chemins de l'intention du départ « start » à l'intention finale « stop », l'un incluant cette intention, l'autre non) et que deux stratégies sont proposées pour réaliser l'intention de chercher des ressources sur le pilotage d'une base de données relationnelle depuis un langage de programmation.

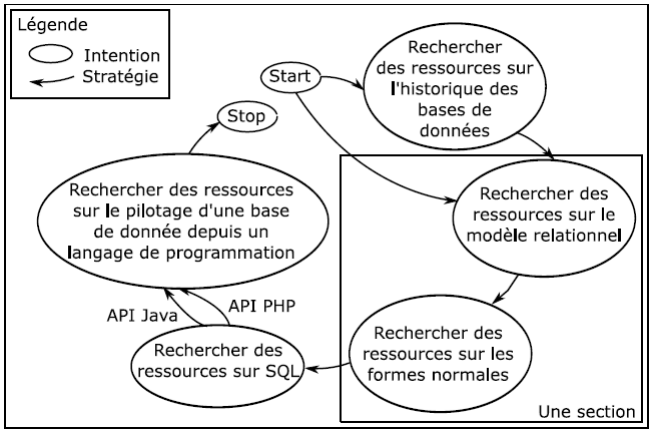


FIGURE 45. EXEMPLE DE DEMARCHE [MIRBEL ET AL., 2009]

La phase de formalisation, est divisée en deux activités. D'abord, le modèle de la carte MAP est raffiné. Le raffinement peut être réalisé au niveau d'une section en donnant une nouvelle carte qui décrit comment atteindre une intention cible d'une façon plus détaillée.

La deuxième sous-étape consiste à générer et/ou écrire (d'une façon semi-automatique) une requête RDF SPARQL [W3C-SPARQL, 2009] afin de concrétiser chaque section par un service Web approprié ou un ensemble de spécifications de services Web. En effet, la requête SPARQL vise à récupérer la description d'un service Web ou un ensemble de services Web supportant l'accomplissement de l'intention cible de la section à laquelle la requête est associée. La figure 46 montre le résultat de la réalisation de la phase de formalisation.

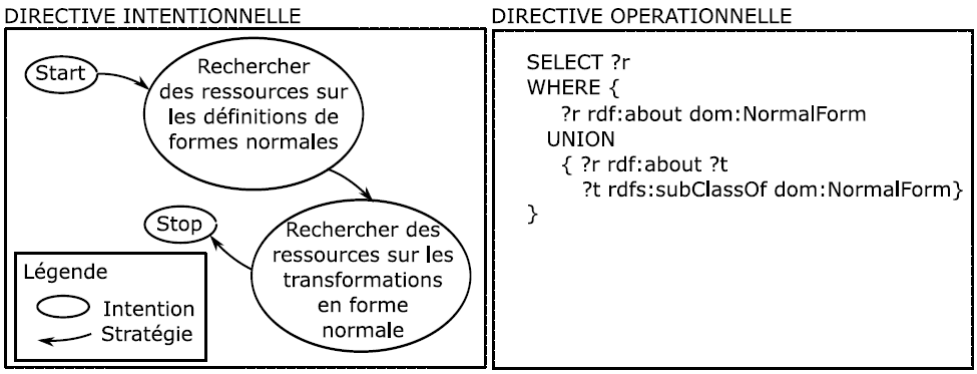


FIGURE 46. MODELE DE LA CARTE ET SA REQUETE RDF SPARQL [MIRBEL ET AL., 2009]

La phase de fragmentation consiste en la transformation de toutes les spécifications capturées pendant la phase de formalisation en un ensemble de règles CORESE [Corby *et al.*, 2008]. CORESE est un moteur de recherche sémantique basé sur le modèle des graphes conceptuels. CORESE intègre un moteur de chaînage arrière exploitant des règles implémentées par des requêtes SPARQL de la forme CONSTRUCT-WHERE. La représentation d'un fragment de démarche est réalisée par la requête SPARQL, dont la clause CONSTRUCT est un patron de graphe permettant de construire la représentation RDF de la section de la carte considérée et dont la clause WHERE est un patron de graphe qui représente soit une carte (règle abstraite) soit un critère de recherche de ressources pertinentes (règle concrète). Dans le cas d'une règle abstraite, la clause WHERE est un patron de graphe représentant une sous-carte permettant la réalisation de l'intention cible du fragment. Dans le cas d'une règle concrète, la clause WHERE est un graphe qui permet de retrouver les ressources pertinentes, c'est-à-dire celles avec les annotations RDF pour lesquelles il existe des appariements avec ce graphe requête. Les règles correspondant aux fragments présentés dans la figure 46 sont représentées dans la figure 47.

REGLE ABSTRAITE

```
CONSTRUCT {
  _:s map:hasTarget _:i
  _:i map:hasObject d:NormalForm
  _:s map:operationalizedBy ?g
}
WHERE {
  GRAPH ?g {
    ?s1 map:hasSource ?i0
    ?i0 rdf:type map:Start
    ?s1 map:hasTarget ?i1
    ?i1 map:hasObject d:NormalFormDef
    ?s2 map:hasSource ?i1
    ?s2 map:hasTarget ?i2
    ?i2 map:hasObject d:NormalizationRule
    ?s3 map:hasSource ?i2
    ?s3 map:hasTarget ?i4
    ?i4 rdf:type map:Stop }
}
```

REGLE CONCRETE

```
CONSTRUCT {
  _:s map:hasTarget _:i
  _:i map:hasObject d:NormalFormDefinition
  _:s map:hasResource ?r
}
WHERE {
    ?r rdf:about dom:NormalForm
 UNION
  { ?r rdf:about ?t
    ?t rdfs:subClassOf dom:NormalForm }
}
```

FIGURE 47. EXEMPLE DE FRAGMENTS DES REGLES CORESE [MIRBEL ET AL., 2009]

La dernière étape de SATIS « **population** » consiste à dériver les spécifications sémantiques des services web pour opérationnaliser l'ensemble des intentions et des stratégies associées à la requête en cours de réalisation. Dans cette dérivation, les spécifications CORESE correspondent à la base de connaissances (stockées sur la forme de requêtes) d'annotations RDF décrivant les services Web disponibles. L'instanciation du processus de recherche et de découverte des services web est réalisée en appliquant les règles qui implémentent les fragments de démarches en chaînage arrière. Ce processus d'instanciation est fait à l'exécution et dépend des services Web disponibles.

La table suivante synthétise l'évaluation de l'approche SATIS.

Vue	Question	Critère	SATIS [Mirbel *et al.*, 2009] Valeurs
Pourquoi ?	Dans quel cadre utiliser l'approche ?	Domaine applicatif	La gestion des processus métier
Qui ?	Qui sont les clients de l'approche ?	Clients de l'approche,	Métiers, opérationnels (Expert du domaine métier, utilisateurs finaux « non informaticiens »)
		Communication client-fournisseur	Ontologie de processus intentionnels basée sur RDFS, Spécification de services web basée sur OWL-S
Quand ?	Quand utiliser l'approche ?	Portée de la mise en œuvre de l'approche	Conception, exécution
		La technologie utilisée	Services web
Comment ?	A quels niveaux d'abstraction les	Les niveaux d'abstraction considérés	Intentionnel, opérationnel

	services sont-ils définis ?		
	Comment les approches sont elles développées ?	Le type d'approche	Descendante
		Type de la notation	Semi-formelle
		Le langage	MAP, RDF SPARQL, OWL-S
		Les outils supportant l'approche	Moteur de recherche CORESE

TABLEAU 15. RESUME DU CADRE DE REFERENCE POUR L'APPROCHE MAAS

4.5. L'analyse comparative

Dans cette section nous avons présenté un état de l'art sur quatre approches à base de services. Cette revue bibliographique a permis de constater que l'ingénierie à base de services permet la spécification et le raffinement des services pour concevoir ou exécuter des services. La table 16 synthétise les processus que nous avons évalués au cours de cette section.

Les approches étudiées permettent en général de construire des processus spécifiques par l'assemblage / composition ou extension de fragments de méthodes sous la forme de services. Ces approches sont généralement de type descendantes. Les approches concernent des domaines d'application diverses, la plupart d'entre elles couvrent les spécifications de besoins en utilisant le modèle de la carte MAP.

L'approche iSOA propose un processus incrémental de raffinement des besoins des utilisateurs dans le but de spécifier les caractéristiques des services, comme c'est également le cas de l'approche SATIS. L'approche SATIS se distingue de l'approche iSOA par le fait que SATIS s'appuie sur les modèles et langages du web sémantique pour enrichir la description des besoins des utilisateurs et ainsi proposer des moyens de raisonnement et d'explications des services web trouvés pour implémenter un besoin métier.

L'approche SO2M proposée par [Guzélien, 2007], utilise la notion de services pour construire des méthodes adaptées aux besoins des concepteurs/développeurs. Cette approche ne traite pas de l'implémentation d'un service « méthode » considérée par l'approche MaaS.

Nous avons constaté que les approches ne font pas une forte distinction entre les aspects intentionnels, organisationnels et opérationnels. Par exemple, iSOA définit les services intentionnels qui permettent de satisfaire les buts organisationnels. SATIS propose des services intentionnels pour opérationnaliser un processus métier, cependant, la spécification métier est définie au niveau intentionnel.

Vue	Question	Critère	iSOA [Kaabi, 2007]	SO2M [Guzélian, 2007]	MaaS [Rolland, 2008]	SATIS [Mirbel *et al.*, 2009]
Pourquoi ?	Dans quel cadre utiliser l'approche ?	Domaine applicatif	L'ingénierie des besoins	L'ingénierie des méthodes	L'ingénierie des méthodes	La gestion des processus métier
Qui ?	Qui sont les clients de l'approche ?	Clients de l'approche,	Métiers, opérationnels (Spécialistes métier, concepteurs/développeurs)	Stratégiques, métiers	Métiers, opérationnels	Métiers, opérationnels (Expert du domaine métier, utilisateurs finaux « non informaticiens »)
		Communication client-fournisseur	Non	Ontologies de buts, d'acteurs, de processus, de produits	Non	Ontologie de processus intentionnels basée sur RDFS, Spécification de services web basée sur OWL-S
Quand ?	Quand utiliser l'approche ?	Portée de la mise en œuvre de l'approche	Conception	Conception, exécution	Conception, exécution	Conception, exécution
		La technologie utilisée	Générique	Générique	Services web	Services web
Comment ?	A quels niveaux d'abstraction les services sont-ils définis ?	Les niveaux d'abstraction considérés	Intentionnel, opérationnel	Intentionnel, organisationnel	Organisationnel, opérationnel	Intentionnel, opérationnel
	Comment les approches sont elles développées ?	Le type d'approche	Descendante	Descendante	Descendante	Descendante
		Type de la notation	Semi-formelle	Semi-formelle	Semi-formelle	Semi-formelle
		Le langage	MAP, arbres de décomposition ET/OU	Modèle de Services (graphe de composition, XML en utilisant une DTD)	XMI-XML, WSDL, UDDI, BPEL	MAP, RDF SPARQL, OWL-S
		Les outils supportant l'approche	Aucune	Plateforme SO2MX	Aucune	Moteur de recherche CORESE

TABLEAU 16. RESUME DU CADRE DE REFERENCE

Nous avons constaté, également, parmi les approches qu'aucune n'offre de mécanismes pour construire des environnements de modélisation de support aux concepteurs/développeurs qui utilisent les services méthodes.

Concernant la communication entre concepteurs, nous avons identifié que les approches SO2M et SATIS utilisent des ontologies pour spécifier un vocabulaire commun entre clients et fournisseurs. Parmi ces deux approches SO2M est la plus représentative de nos objectifs. Cette approche utilise les ontologies de buts, d'acteurs, de processus, de produits. Cependant, ces ontologies ne sont que

faiblement couplées aux méta-modèles de service. Une plus grande intégration permettrait d'avoir un modèle de service plus uniforme et réutilisable.

Pour ce faire, nous nous basons sur la notion des services méthodes comme source d'inspiration de notre approche. Notre position est de faciliter le travail des concepteurs de modèles en les aidant dans le choix de processus, des modèles et des environnements de modélisation adaptés à leurs besoins spécifiques. Nous proposons d'établir une séparation entre les aspects intentionnels, organisationnels et opérationnels afin de clairement distinguer les buts de modélisation, les méthodes / stratégies qui permettant réaliser les buts et les outils de modélisation qui serviront de support aux méthodes choisies. Pour cela, la partie intentionnelle permet de séparer les buts des processus métier. Le niveau organisationnel facilite la sélection de processus. Le niveau opérationnel permet de choisir l'ensemble des outils appropriés au processus métier sélectionné au niveau organisationnel.

Dans cette section nous avons présenté quatre approches à base de services. Nous avons établi des critères de comparaison et de classification. A partir de ces critères nous avons réalisé le positionnement de notre approche. Le chapitre suivant présente notre approche.

CHAPITRE 3

La démarche

Ce chapitre présente une approche orientée services pour la réutilisation de processus et des outils de modélisation. Nous commençons par présenter de façon générale les trois niveaux de services (§1). Ensuite, nous détaillons l'utilisation des ontologies dans le domaine des systèmes d'information (§2), puis nous présentons deux méthodes que seront utilisées dans les études de cas, présentées dans ce chapitre (§3). Dans (§4), (§5) et (§6) nous présentons respectivement les niveaux et les méta-modèles des services intentionnel, organisationnel et opérationnel. Chaque niveau est présenté avec un exemple. Finalement, nous concluons le chapitre par la section 7.

CHAPITRE 3

INTRODUCTION

Ce chapitre est consacré à la définition de trois niveaux de services dénommés : intentionnel, organisationnel et opérationnel. Ces trois niveaux correspondent à trois niveaux d'abstraction pour les services de gestion de modèles qui permettent d'aider les concepteurs à exprimer leurs intentions, et les liens vers les méthodes et les outils de modélisation.

Ce chapitre est composé de sept sections. Nous présentons l'approche générale pour la réutilisation des processus et des outils de modélisation à la section 1. La section 2 est consacrée à la présentation des ontologies utilisées dans les trois niveaux de services. La section 3 présente deux méthodes que seront utilisées dans les études de cas, présentées dans ce chapitre. Les sections 4, 5 et 6 présentent respectivement les niveaux et les méta-modèles des services intentionnel, organisationnel et opérationnel. Chaque niveau est présenté avec un exemple. Finalement, nous concluons le chapitre par la section 7.

1. LES TROIS NIVEAUX DE SERVICES

Notre approche s'appuie sur trois niveaux de modélisation (voir figure 48) dont les fournisseurs, les clients et les services sont différents.

Le premier niveau correspond à **la couche opérationnelle**. Cette couche permet de définir l'infrastructure de modélisation qu'un concepteur de modèles veut utiliser. Il s'agit d'offrir des services aux concepteurs de modèles (« ***model designers*** »), pour faciliter la création de leur environnement de modélisation. **Le client** est donc un concepteur de modèles qui désire gérer des modèles de manière individuelle ou collaborative (avec d'autres concepteurs). Donc, il a besoin de définir et d'ajuster un environnement de modélisation offrant des outils adaptés à son contexte d'utilisation, ses compétences et ses préférences en termes de gestion de modèles. Par exemple : un « utilisateur lambda » peut avoir besoin d'un environnement de modélisation qui lui offre l'édition de modèles UML et la transformation de ces modèles dans un autre langage de programmation ou de modélisation. **Un fournisseur** correspond aux sociétés ou à des institutions proposant des outils (AGLs, outils de modélisation, outils de transformation,…) de gestion de modèles (« ***enterprises which have tools*** »). Leur but est de diffuser leurs outils en les insérant dans la plate-forme en tant que

services opérationnels. Par exemple, l'outil de modélisation d'arbres de tâches, ConcurTask Trees, est proposé par l'université de Pise.

La couche organisationnelle est inspirée des travaux de Ralyté [Ralyté, 1999] qui a adopté les idées de l'Ingénierie des Méthodes Situationnelles [Kumar *et al.*, 1992]. Dans notre travail, nous utilisons la notion de service pour soutenir la construction de processus de conception de systèmes en assemblant des fragments de méthodes. Il s'agit d'offrir un support à base de services à des groupes de projet. Dans cette couche, les rôles et les activités sont exprimés sous la forme de processus de développement simplifiés. La simplification est liée au fait que l'objectif n'est pas de capitaliser l'intégralité des fragments de méthode mais uniquement les connaissances (liées à ces fragments) permettant à chaque concepteur, donc à chaque rôle du groupe projet, de disposer de l'environnement de modélisation qui lui convient. Un fragment de méthode est modélisé par un ensemble d'activités manipulant des modèles et exécutées par des acteurs jouant des rôles. Les activités sont donc décrites en termes d'actions sur des modèles, sans prendre en considération l'ordonnancement des actions.

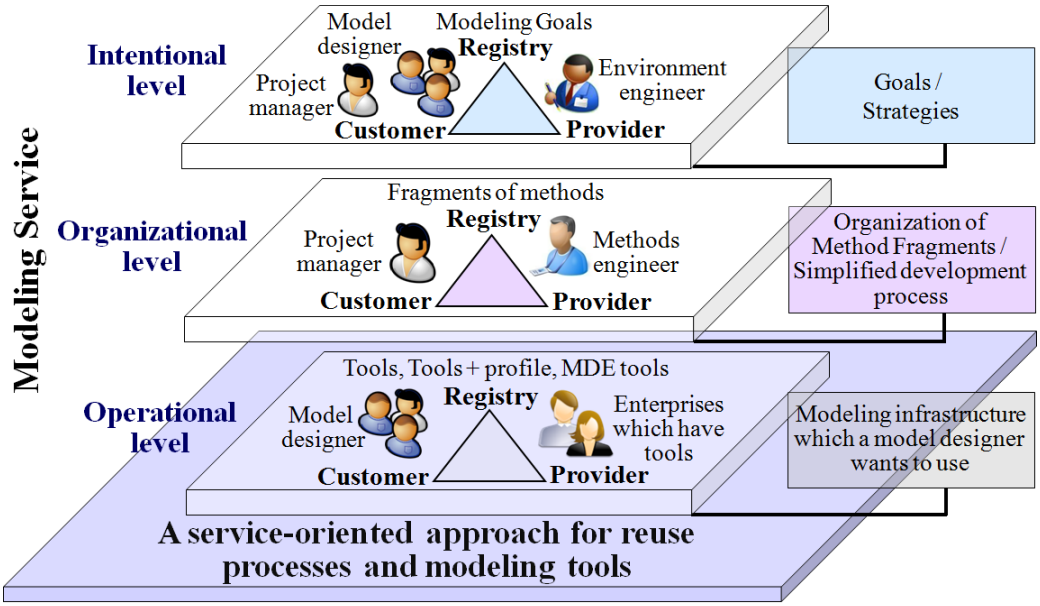


FIGURE 48. TROIS NIVEAUX DE SERVICE

La couche organisationnelle permet de réutiliser de façon coordonnée les services opérationnels. **Les clients** sont, dans ce cas, les chefs de projet (« ***project manager*** ») cherchant à définir et à administrer des rôles et des activités sous la forme de processus de développement. Les chefs de projet vont choisir des services organisationnels (parties de processus de conception) qui nécessitent la mise en place de services opérationnels de gestion de modèles. Ainsi ils définissent la création des environnements de gestion de modèles pour les concepteurs impliqués dans leurs processus de développement. **Un fournisseur** est alors un ingénieur de méthodes (« **method engineer** »). Un ingénieur de méthodes a pour but de rendre réutilisable et d'outiller tout ou partie de sa méthode

sous la forme de services organisationnels. Par exemple, un fragment du processus pour spécifier l'interaction homme-machine peut être défini pour faire partie d'une méthode de conception de systèmes interactifs. Un tel fragment peut par exemple être extrait de [Juras *et al.*, 2006].

Finalement, les clients peuvent utiliser les services opérationnels offrant un environnement de modélisation support du processus choisi, par exemple : « pour spécifier l'interaction homme-machine », les clients ont besoin d'un environnement de modélisation qui supporte des modèles d'arbres de tâches et de description des dispositifs d'interaction.

La couche intentionnelle permet la modélisation des buts. Il s'agit de conceptualiser des besoins stratégiques de modélisation requis par un sujet individuel, un groupe d'individus, une unité de travail ou toute organisation qui participe au processus de développement de systèmes. Cette couche permet de réutiliser les services organisationnels. **Le fournisseur de la couche intentionnelle** correspond à l'ingénieur de l'environnement (« ***environment engineer*** »). Il s'agit d'un nouveau métier qui s'occupe de l'administration et la gestion de la plateforme. **Les clients de la couche intentionnelle** sont ceux des couches organisationnelle et opérationnelle, c'est-à-dire les concepteurs de modèles (« ***model designer*** ») et les chefs de projets (« ***project manager*** »). Pour ces clients, les services sont les buts proposés par l'ingénieur de l'environnement qui peuvent être vus comme les buts à atteindre pour la gestion de modèles, comme par exemple : «**spécifier un Système Interactif**». Ce but est réalisé au niveau organisationnel par des services organisationnels. Les clients font un choix parmi les services organisationnels associés au service intentionnel. Par exemple, le but «spécifier l'interaction de l'utilisateur» est atteint par plusieurs fragments de processus, parmi lesquels le client devra choisir.

Dans cette section nous avons présenté notre approche de gestion de modèles avec la notion de services. Nous avons considéré trois niveaux d'abstraction : **opérationnel**, **organisationnel** et **intentionnel**. Nous les détaillons en sections 4, 5 et 6. La section suivante présente une série d'ontologies que seront utilisées dans notre approche.

2. LES ONTOLOGIES

L'utilisation des ontologies nous semble tout à fait pertinente pour disposer, dans les environnements à base de services, d'un vocabulaire commun entre fournisseurs et clients. Cette section introduit un ensemble d'ontologies utilisées dans le contexte de nos services. Nous présentons dans un premier temps le principe d'utilisation des ontologies (buts, produits, processus et rôles) proposées par Guzélian [Guzélian *et al.*, 2004][Guzélian, 2007]. Pour disposer d'une modélisation uniforme et complète, ces ontologies sont intégrées dans les méta-modèles correspondant aux trois niveaux d'abstraction. La modélisation de ces ontologies s'appuie sur le concept de catégorisations. Nous

présentons des patrons pour la catégorisation, puis une adaptation du patron Item-Description que nous avons appelé « **Concept-Term** ».

2.1. L'intégration des ontologies

2.1.1. Le travail de référence

Les ontologies, qui définissent une terminologie pour partager un vocabulaire commun au sein d'un domaine, sont aujourd'hui largement utilisées pour répondre à ce besoin. Il est essentiel de disposer d'une sémantique riche des services afin de faciliter la communication et la compréhension des intervenants qui partagent ses activités dans un domaine particulier.

Dans le contexte des services, les ontologies (buts, produits, processus et rôles) proposées par Guzélian [Guzélian *et al.*, 2004][Guzélian, 2007] définissent une terminologie réutilisable et partageable par les fournisseurs des services « méthode » (ingénieurs de méthodes) et par les clients qui les utilisent (concepteurs/développeurs de SI). Les ontologies permettent aussi d'enrichir la sémantique de la description des services. Ce niveau sémantique joue un rôle essentiel dans l'automatisation de la recherche/sélection, dans l'adaptation et dans la composition de services.

Les ontologies définissent un ensemble de termes qui seront utilisés pour définir les valeurs d'un concept au moment de la spécification de services particuliers. Les termes d'une même ontologie peuvent être utilisés dans les différentes parties de la description d'un service « méthode ». Par exemple, l'ontologie des produits fournit une typologie des artéfacts de SI qui peuvent intervenir dans la description du contexte et dans la définition des ressources d'un même service.

L'utilisation des ontologies de Guzélian [Guzélian *et al.*, 2004][Guzélian, 2007] dans la spécification des services repose sur un mécanisme qui permet de faire le lien entre le modèle de services, les services « méthode » et les ontologies de l'ingénierie des systèmes d'information. Ce mécanisme est illustré par la figure 49. Le modèle de services est utilisé par l'ingénieur de services « méthode » pour spécifier des services. Au contraire, les ontologies sont partagées à la fois par l'ingénieur de services « méthode » (fournisseur de services) et le concepteur de systèmes d'information (consommateur de services). Par exemple, l'ontologie des buts peut à la fois être utilisée au moment de la conception des services pour définir le but d'un service et au moment de la recherche de services pour formuler les requêtes.

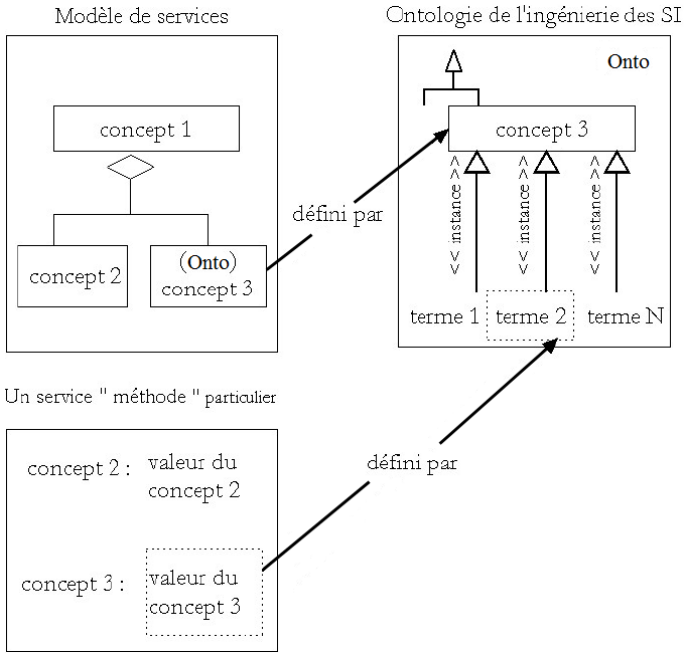


FIGURE 49. MECANISME D'UTILISATION DES ONTOLOGIES [GUZELIAN, 2007]

Le modèle de services est constitué de concepts décrits par des diagrammes de classes UML. Il propose un ensemble structuré de concepts pour spécifier les services « méthode ». Certains concepts du modèle de services peuvent être décrits dans une ontologie de l'ingénierie des SI. **Les ontologies** (dénotés comme « Onto » dans la figure 49) sont composées de concepts, de termes et de relations entre les concepts et les termes. Dans le modèle de services, un concept défini par une ontologie de l'ingénierie des SI est référencé par : (nom de l'ontologie) nom du concept. Par exemple, le concept de but décrit dans l'ontologie des verbes qui décrivent les intentions spécifiques pour la gestion de modèles est noté comme « Lbut » But dans le modèle de services.

La spécification d'un service « méthode » particulier est une instanciation des concepts du modèle de services. Au moment de la définition du service « méthode », la valeur d'un concept du modèle de services est définie par un terme appartenant à une ontologie de l'ingénierie des SI. Dans un service « méthode », un concept instancié par un terme de l'ontologie est référencé par le nom du terme. Par exemple, le concept de but peut avoir pour valeur le verbe « construire » (figure 50). Cependant, ce concept peut avoir d'autres termes comme développer, créer...

La figure 50 montre un exemple d'utilisation de l'ontologie de verbes. Notons que le concept « but » de la définition de la partie identifiant du modèle de service SO2M est décrit pour le concept « verbe » dans l'ontologie de buts. Cette définition permet de standardiser et classifier l'utilisation des concepts pour spécifier les buts et permet d'établir un vocabulaire commun entre les fournisseurs et les concepteurs.

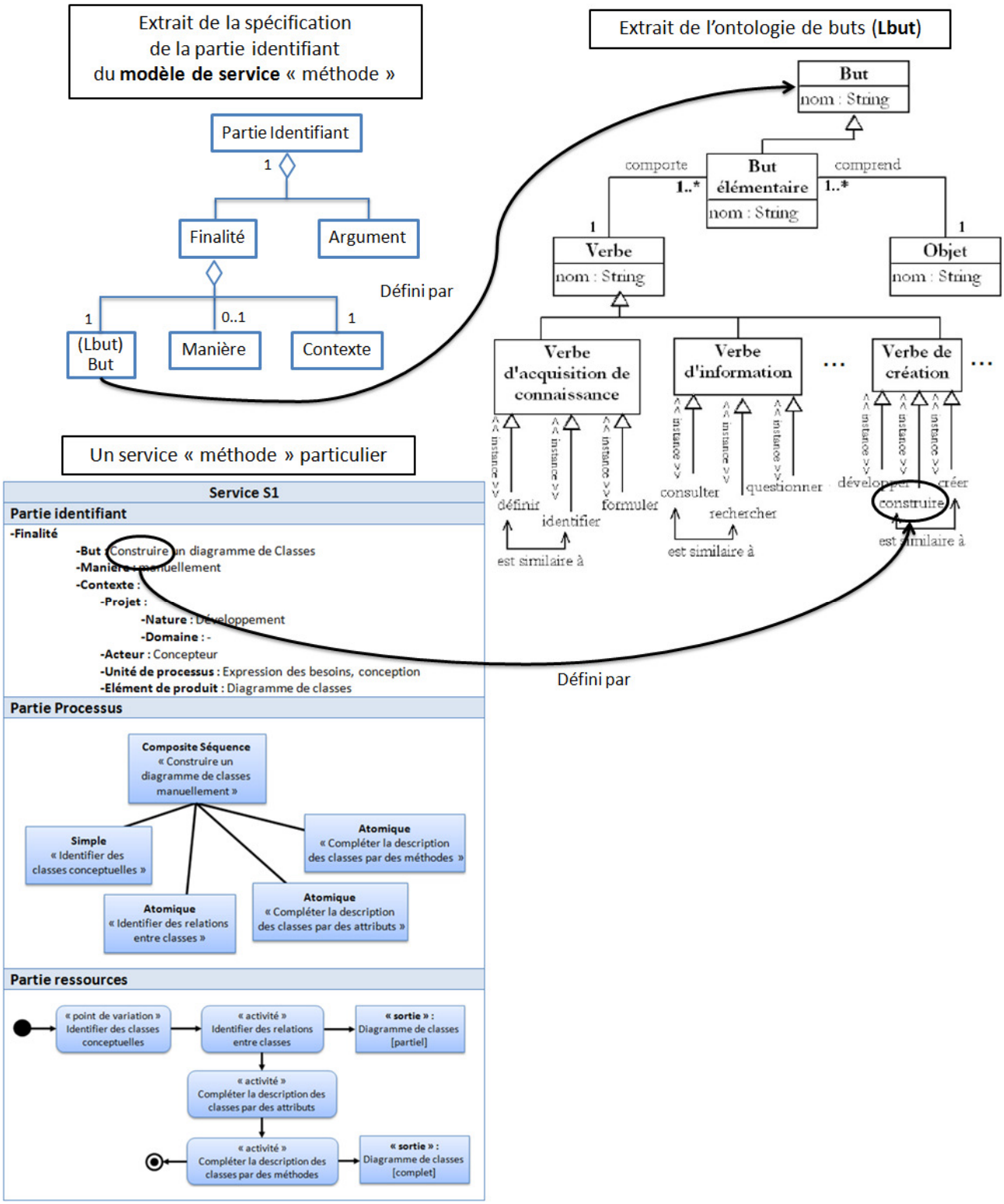


FIGURE 50. EXEMPLE D'UTILISATION DES ONTOLOGIES [GUZELIAN, 2007]

L'utilisation des ontologies de Guzélian dans la spécification des services est un mécanisme très intéressant qui permet de faire le lien entre le modèle de services et les concepts du domaine de l'ingénierie des systèmes d'information. Cependant, nous considérons très important le fait d'avoir un modèle de services lisible (complèt et uniforme) où on intègre la spécification des services et les termes du domaine de systèmes d'information. La section suivante présente le patron Concept –

Term, un mécanisme utilisé pour réaliser l'intégration des ontologies dans les spécifications de nos modèles de services.

2.1.2. Le patron Concept – Term

Nous avons énoncé au début de cette section que l'utilisation des ontologies nous semble tout à fait pertinente pour disposer, dans les environnements à base de services, d'un vocabulaire commun entre fournisseurs et clients. Notre approche consiste à les intégrer dans les différents méta-modèles correspondant aux trois niveaux d'abstraction de manière à disposer de méta-modèles complets et uniformes.

Pour intégrer une ontologie, dans nos modèles de service, nous nous appuyons sur le principe de catégorisation. La catégorisation permet de classer des concepts selon des propriétés communes. Dans ce contexte, le patron générique Item-Description proposé par [Coad, 1992] (voir Annexe F) ainsi que le patron Type-Object de [Johnson *et al.*, 1997] (voir Annexe G) permettent tous les deux de répondre à ce besoin. Nous nous sommes inspirés du patron Item-Description par le fait que la classe ItemDescription détient les valeurs des attributs qui peuvent s'appliquer à plus d'un objet de type Item. Par rapport au patron Type-Object, nous sommes intéressés par le fait que les instances d'une classe doivent être groupées selon des attributs et/ou comportements communs.

La figure 51 présente une adaptation des patrons Item-Description, Type-Object auquel nous avons rajouté deux classes énumérées pour ajouter les termes et les concepts utilisées dans le patron. Nous appellerons ce patron : Patron « **Concept – Term** ».

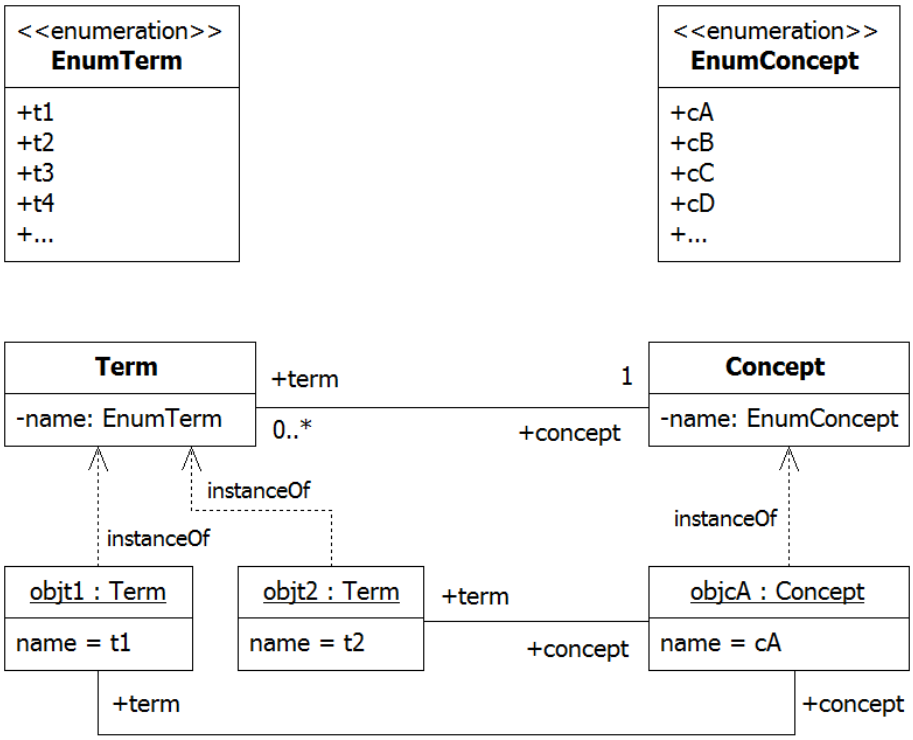


FIGURE 51. PATRON CONCEPT-TERM

Chaque concept de l'ontologie est représenté par :

1. Une valeur du type énuméré « **EnumConcept** », par exemple : cA.
2. Une instance de la classe « **Concept** » dont l'attribut « **name** » est valué par une valeur du type énuméré « **EnumConcept** », par exemple : objcA.

Chaque terme de l'ontologie est représenté par :

1. Une valeur du type énuméré « **EnumTerm** », par exemple : t1.
2. Une instance de la classe « **Term** » dont l'attribut « **name** » est valué par une valeur du type énuméré « **EnumTerm** », par exemple : objt1.

Chaque instance de la classe Concept conserve une référence vers ses instances de la classe Term via le rôle term. Inversement, chaque instance de la classe Term pointe vers son Concept.

2.1.3. L'utilisation du Patron Concept – Term dans les méta-modèles de services

La figure 52 montre la vision globale d'utilisation du patron Concept – Term dans les méta-modèles de services de notre approche. Au moins une classe du méta-modèle de services est associée à la classe Term de l'ontologie pour utiliser ses éléments énumérés.

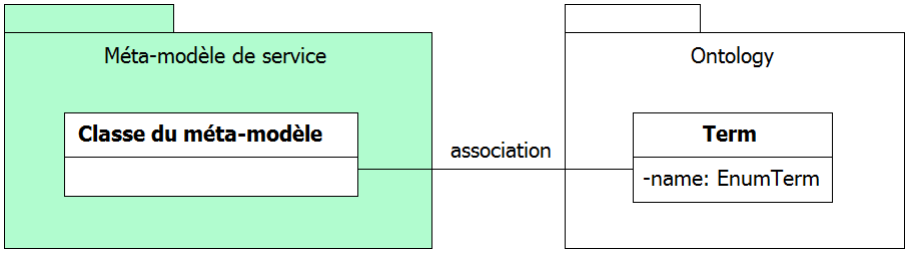


FIGURE 52. UTILISATION DU PATRON CONCEPT – TERM

2.2. Les ontologies utilisées dans les méta-modèles.

Cette section décrit des exemples d'imitation de patrons des ontologies que seront utilisées dans les méta-modèles des différents niveaux d'abstraction. Les ontologies que nous avons adoptées sont : l'ontologie des verbes « **OntologyVerb** », l'ontologie des produits « **OntologyProduct** », l'ontologie des acteurs « **OntologyActor** », l'ontologie des processus « **OntologyProcess** » et l'ontologie des facteurs de qualité du logiciel « **OntologyQualityFactor** »,

2.2.1. L'ontologie des verbes « OntologyVerb »

Les ontologies de buts [Guzélian, 2007] décrivent les intentions spécifiques pour la gestion de modèles (par exemple : **améliorer** un processus métier, **personnaliser** le processus de spécification de besoins, **élaborer** un scénario, **automatiser** une procédure administrative, etc.). Nous les avons utilisées pour modéliser les verbes avec le patron Concept – Term (Figure 53). Chaque verbe, modélisé par une instance de TermVerb, est lié à son concept, modélisé par une instance de la classe ConceptVerb. Un ConceptVerb référence donc les verbes de même « nature ». Les verbes et leur catégorisation sont utilisés, dans le méta-modèle de la couche intentionnelle, pour exprimer les intentions des concepteurs de modèles. Un verbe possède plusieurs **verbes synonymes** (par exemple : développer, construire, concevoir, etc.). Donc, les verbes synonymes appartiennent à la même catégorie de verbes.

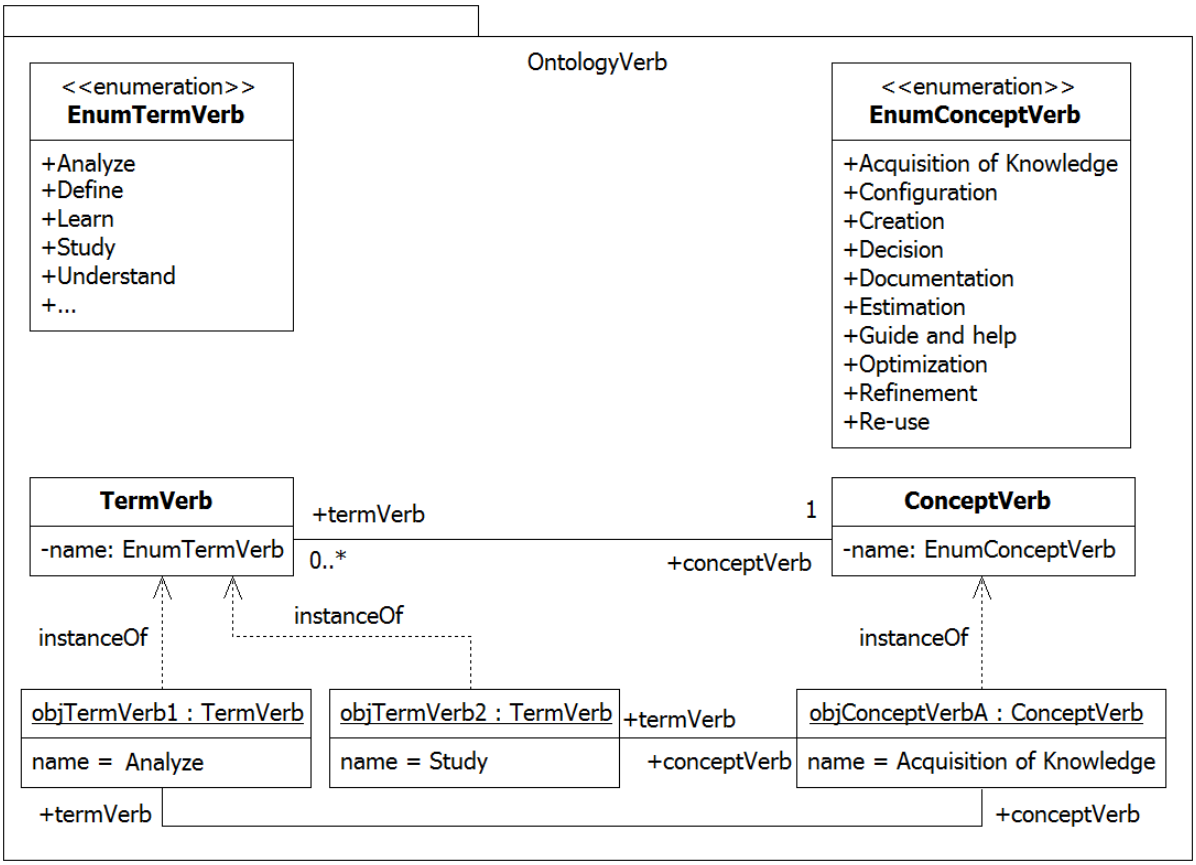


FIGURE 53. EXEMPLE D'IMITATION DU PATRON CONCEPT-TERM POUR LA REPRESENTATION DE L'ONTOLOGIE DES VERBES

Les valeurs de EnumConceptVerb et EnumTermVerb sont issues de l'ontologie des buts de Guzélian [Guzélian, 2007]. L'ontologie des buts représente l'ensemble des problèmes que l'on peut rencontrer dans le développement de SI. Cette catégorisation montre des exemples de but utilisant les différentes catégories de verbes. Dans notre contexte de travail, les verbes expriment des actions plus ou moins complexes de modélisation. Donc, l'ontologie des buts propose une classification issue des types d'activités (ou d'actions) que les concepteurs de modèles mettent en œuvre durant le développement. Les activités (ou actions) usuelles sont l'acquisition de connaissances, la prise de décision, la construction de modèles, la documentation, etc. Nous avons classé ces activités (ou actions) suivant notre patron Concept – Term appliqué à l'ontologie des verbes (Tableau 17).

Concept de Verbe (ConceptVerb)	**Verbes (TermVerb)**	**Exemple d'utilisation**
Acquisition de connaissance Ils concernent la recherche, la collecte, et l'analyse d'informations.	Analyser, apprendre, comprendre, définir, déterminer, expliquer, exprimer, étudier, formuler, identifier, spécifier	-**Spécifier** un système interactif -**Identifier** les cas d'utilisation
Guidage et aide Ils concernent le guidage et	Conduire, consulter, orienter, rechercher, questionner	-**Consulter** la liste de besoins non fonctionnels -**Rechercher** les différentes façons de décrire

l'aide à la réalisation de tâches de développement.		un scénario
Documentation Ils concernent l'explication, la description, et la représentation des produits du développement.	Caractériser, décrire, documenter, expliciter, informer, représenter, structurer, visualiser	-**Documenter** des classes Java avec un diagramme de classes -**Décrire** un scénario de façon textuelle
Création de produits Ils concernent la construction de produit de conception ou développement.	Accomplir, automatiser, concrétiser, construire, créer, concevoir, dessiner, développer, élaborer, établir, exécuter, faire, formaliser, fabriquer, générer, implémenter, matérialisera, produire, réaliser	-**Elaborer** un scenario -**Construire** un diagramme de séquence
Configuration Ils concernent la définition et l'adaptation de processus de développement.	Adapter, changer, configurer, évoluer, personnaliser, piloter	-**Personnaliser** le processus de spécification de besoins -**Adapter** le processus de construction d'un diagramme de cas d'utilisation
Optimisation ils concernent l'amélioration, la normalisation des produits et des processus de développement.	Améliorer, augmenter, contrôler, faciliter, minimiser, normaliser, synchroniser, systématiser	-**Normaliser** un schéma relationnel -**Améliorer** un processus métier
Raffinement ils concernent la décomposition et le niveau de détail de définition des éléments de produit et des processus.	Détailler, décomposer, préciser, spécialiser, raffiner, réduire	-**Décomposer** une activité en sous-activités -**Raffiner** un scénario
Estimation ils concernent la validation, le test et la vérification.	Calculer, comparer, estimer, examiner, évaluer, prédire, tester, vérifier, valider, simuler	-**Valider** un MCD -**Vérifier** la cohérence entre différents diagrammes UML
Décision ils concernent les choix de stratégie et les choix de solutions parmi un ensemble de possibilités.	Choisir, décider, juger, sélectionner	-**Choisir** une méthode de conception de systèmes d'information -**Sélectionner** une architecture parmi une architecture orientée services et une architecture trois tiers
Réutilisation ils concernent l'utilisation, la modification et la transformation de produits existants.	Ajuster, appliquer, compléter, imiter, modifier, reformer, reproduire, réutiliser, utiliser	-Réutiliser un composant métier -Reproduire le comportement d'un objet

TABLEAU 17. TYPOLOGIE DES VERBES POUR LA GESTION DE MODELES

2.2.2. L'ontologie de produits « OntologyProduct »

L'ontologie des produits permet de structurer les différents objets qui peuvent être utilisés ou produits au cours de la gestion de modèles (par exemple : améliorer un **processus métier,** élaborer **un scenario,** etc.). Donc, elle définit une catégorisation des éléments de produit. La figure 54 présente

un exemple d'imitation du patron **Concept-Term** pour la modélisation des produits en utilisant l'ontologie des produits proposée par Guzélian [Guzélian, 2007].

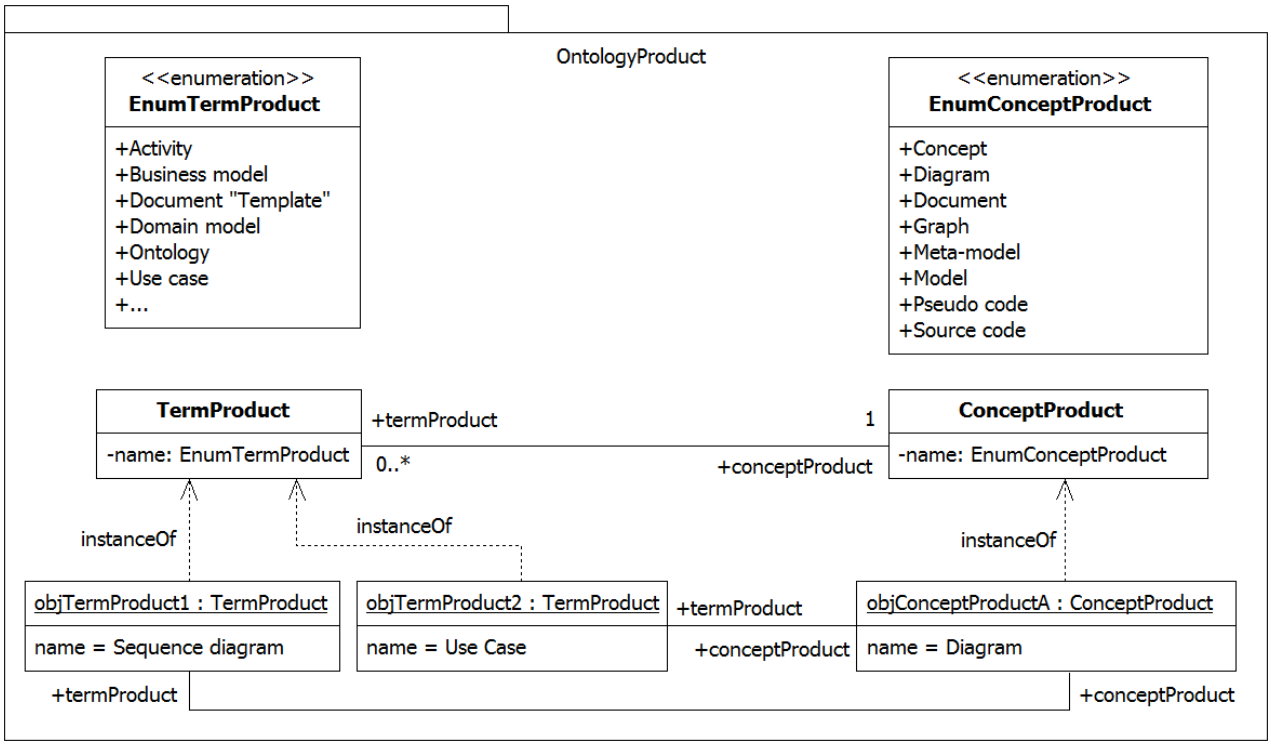


FIGURE 54. EXEMPLE D'IMITATION DU PATRON CONCEPT-TERM POUR LA REPRESENTATION DE L'ONTOLOGIE DES PRODUITS

Les valeurs de EnumConceptProduct et EnumTermProduct sont issues de l'ontologie des produits de Guzélian [Guzélian, 2007] (voir le tableau 18). Cette catégorisation montre des exemples d'objets. Les catégories de produits retenues sont : le concept, le diagramme, le document le modèle, le méta-modèle, le pseudo-code et le code source.

Concept de produit (ConceptProduct)	**Produits (TermProduct)**	**Exemple d'utilisation**
Concept Représentation générale et abstraite d'une réalité. Le concept se distingue aussi bien du produit représenté par ce concept, que du mot ou de l'énoncé verbal, qui est le signe de ce concept mental.	Activité, classe, composant métier, processus métier, stéréotype, tache	-Améliorer un **processus métier** -Réutiliser un **composant métier**
Diagramme Présentation graphique d'un ensemble d'éléments, le plus souvent sous la forme d'un graphique associant éléments et relations.	Diagramme : des cas d'utilisation, de classes, d'objets, de séquence, de collaboration, d'états, d'activités, de composants, de déploiement, diagramme de scénario	-Identifier le **cas d'utilisation** -Construire un **diagramme de séquence**
Document Un document est défini comme le support physique d'une	Glossaire, document « template », liste de besoins non fonctionnels,	-Définir une **liste des concepts**

information. Plus précisément on peut le définir comme un ensemble de données informatives présentes sur un support, sous une forme permanente et lisible pour l'utilisateur.	besoins de distribution, liste de concepts	-Consulter le **document d'expression des besoins**
Graphe Les graphes son des structures définies par deux ensembles : l'ensemble des sommets et l'ensemble des arêtes, chaque arête étant un couple de sommets adjacents.	Graphe RDF, arbre décisionnel, arbre des tâches	-Créer un **graphe RDF** -Construire un **arbre décisionnel**
Méta-modèle Est un modèle de langage de modélisation. Il définit le langage d'expression d'un modèle [OMG, 2006]	UML, modèle entité-association de la méthode Merise, DSL, un profil pour UML	-Créer **le DSL** pour un domaine spécifique -Définir **un profil pour UML**
Modèle Représentation simplifiée d'un système modélisé sous la forme d'un ensemble de faits construits dans une intention particulière	Modèle de domaine, ontologie, modèle de test, modèle d'analyse, modèle de conception, modèle conceptuel des données (MCD)	-Créer **un modèle d'analyse** -Valider **un MCD**
Pseudo code Le pseudo-code est une façon de décrire un algorithme sans référence à un langage de programmation en particulier. L'écriture en pseudo-code permet souvent de bien prendre toute la mesure de la difficulté de la mise en œuvre de l'algorithme, et de développer une démarche structurée dans la construction de celui-ci, en oubliant temporairement la syntaxe rigide d'un langage de programmation.	Algorithme, règle fonctionnelle	-Concevoir **un algorithme** indépendamment du langage qui sera utilisé ensuite pour le programmer -Ecrire **les règles fonctionnelles** d'un système
Code Source Le code source est un ensemble d'instructions écrites dans un langage de programmation informatique de haut niveau, compréhensible par un être humain entraîné, permettant d'obtenir un programme pour un ordinateur.	Classe java, composant JavaBeans, service WSDL	-Editer **une classe en java** -Elaborer **un composant JavaBeans**

TABLEAU 18. TYPOLOGIE DES PRODUITS POUR LA GESTION DE MODELES [GUZELIAN, 2007]

2.2.3. L'ontologie des acteurs « OntologyActor »

L'ontologie des acteurs représente les rôles joués par les personnes qui ont en charge les problèmes d'ingénierie. Nous avons utilisé l'ontologie des acteurs proposée par Guzélian [Guzélian, 2007] qui regroupe des types d'acteurs définis dans le processus unifié associé à UML [Kruchten, 2000]. Dans notre contexte ces acteurs correspondent aux rôles responsables de la réalisation d'un fragment de méthode spécifié sous la forme d'un service organisationnel.

La figure 55 présente un exemple d'imitation du patron **Concept-Term** pour la modélisation des acteurs. Cette ontologie est catégorisée par 6 types d'acteurs. Ces acteurs peuvent être des

architectes, des administrateurs, des concepteurs, des développeurs, des superviseurs, ou des testeurs.

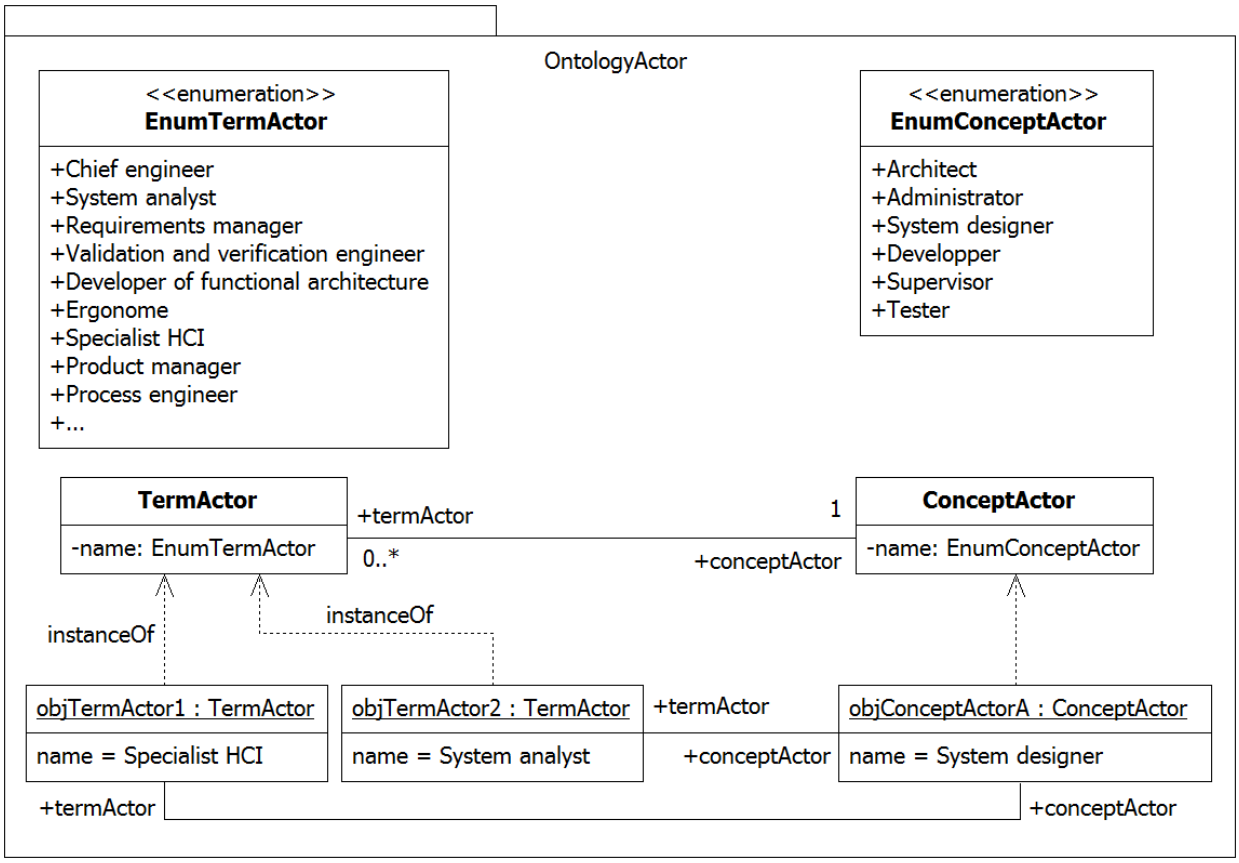


FIGURE 55. EXEMPLE D'IMITATION DU PATRON CONCEPT-TERM POUR LA REPRESENTATION DE L'ONTOLOGIE DES ACTEURS

Le tableau 19 montre le classement des acteurs et ses descriptions.

Concept d'acteur (ConceptActor)	Acteurs (TermActor)
Concepteur Un concepteur est une personne qui est chargé d'identifier les besoins des utilisateurs et de les spécifier	Concepteur de cas d'utilisation, concepteur métier, ingénieur de méthode, Concepteur d'interface utilisateur, concepteur de services, spécialiste IHM, ergonome
Développeur Un développeur est un informaticien qui réalise des logiciels en créant des algorithmes et en les mettant en œuvre dans un langage de programmation	Ingénieurs d'application, ingénieur de composants, Programmeur
Testeur Un testeur est une personne chargée de tester le système (ou logiciel) produit, et de livrer à intervalles réguliers un rapport décrivant les erreurs trouvées dans le produit évalué	Testeur de système, testeur de performance, intégrateur de système, testeur d'intégration
Superviseur Le superviseur est la personne responsable qui a pour	Auditeur du modèle métier, chef de projet, responsable de validation de besoins

fonction de contrôler, auditer, gérer le déroulement des activités de conception d'un système	
Architecte Un architecte est une personne dont l'activité consiste en l'analyse technique nécessaire à la conception du diagramme d'architecture, c'est-à-dire le plan de construction d'un logiciel	Architecte logiciel, urbaniste
Administrateur L'administrateur est la personne responsable des serveurs d'une organisation	Administrateur de système, administrateur de base de données

TABLEAU 19. TYPOLOGIE DES ACTEURS POUR LA GESTION DE MODELES [GUZELIAN, 2007]

2.2.4. L'ontologie de Processus « OntologyProcess »

L'ontologie des processus définit une terminologie pour préciser dans quels cas l'utilisation du fragment de processus (spécifié sous la forme d'un service organisationnel) est appropriée. Nous avons utilisé la spécification de SPEM 2.0 [OMG, 2008], en particulier, le concept WorkDefinition pour créer l'ontologie des processus. Dans SPEM la notion de WorkDefinition est un concept central qui généralise toutes les activités de travail de SPEM. Ces processus peuvent être des activités, des disciplines, des itérations, des cycles de vie de développement, des phases, des processus, des tâches, ou des techniques.

La figure 56 présente un exemple d'imitation du patron **Concept-Term** pour la modélisation des processus. Cette ontologie est catégorisée par 8 types de processus qui correspondent aux WorkDefinition de SPEM.

Dans notre ontologie de processus, chaque terme de l'ontologie (représenté par l'élément : **TermProcess**) correspond à un fragment de la méthode. Un fragment de méthode « **TermProcess** » est caractérisé par le nom de la méthode (par exemple : RUP), le nom du fragment (par exemple : Elaboration), et une référence vers des sources bibliographique (par exemple : www-01.ibm.com/software/awdtools/rup/). Une instance de la clase « **TermProcess** » utilisé l'attribut « **nameMethod** » pour référencer une valeur du type énuméré « **EnumTermMethod** », par exemple : RUP.

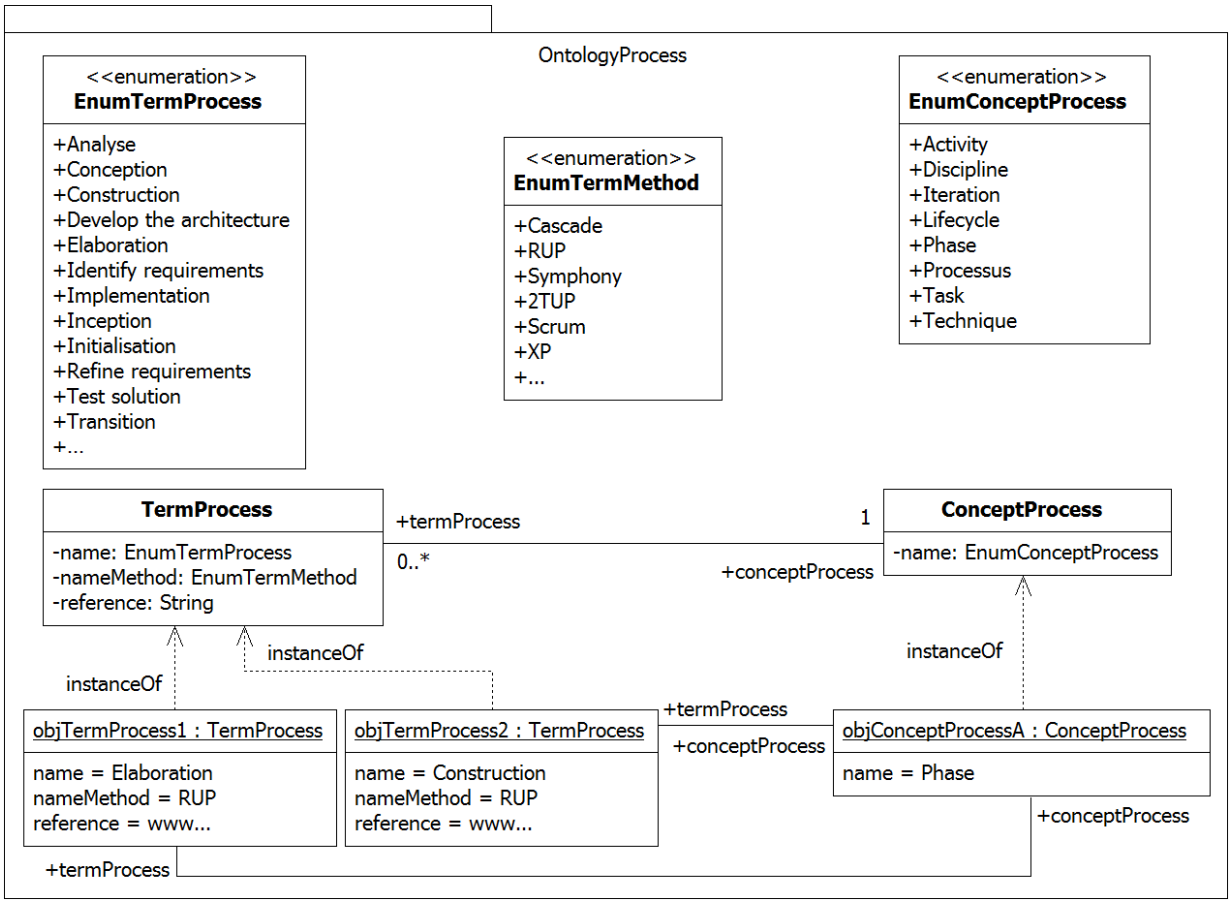


FIGURE 56. EXEMPLE D'IMITATION DU PATRON CONCEPT-TERM POUR LA REPRESENTATION DE L'ONTOLOGIE DE PROCESSUS

Le tableau 20 montre le classement des processus et ses descriptions.

Concept de processus (ConceptProcess)	**Processus (TermProcess)**
Activité Une activité est une unité de travail général assignable qui définit les unités de base de travail au sein d'un processus, ainsi que d'un processus lui-même.	Décomposition organisationnelle de la méthode Symphony étendue, Spécifications externes de l'interaction de la méthode Symphony étendue, …
Discipline Les disciplines correspondent à un regroupement d'unités qui portent sur un même élément de produit.	Expression des besoins de la méthode RUP, Analyse et conception de la méthode RUP, mise en œuvre de la méthode RUP, …
Phase Elles correspondent à un regroupement temporel des unités de processus. Elles représentent une période importante dans un projet.	Spécification conceptuelle des besoins de la méthode Symphony, Spécification organisationnelle et interactionnelle des besoins de la méthode Symphony, Inception de la méthode RUP, Elaboration de la méthode RUP, …
Processus Désigne un ensemble d'activités qui utilisent des ressources effectuées par des concepteurs pour	-

effectuer une tâche donnée dont le résultat final attendu est un produit.	
Cycle de vie Elles correspondent aux approches de développement usuelles utilisées en ingénierie des Systèmes d'Information.	Le cycle de vie en V, le cycle de vie en spiral, …
Tâche Une tâche définit le travail étant effectué par les instances de définition de rôles.	Spécifier l'interaction de la méthode Symphony, …
Technique Une technique est une ou un ensemble de méthode(s), elle est souvent associée à un savoir-faire professionnel. La technique couvre l'ensemble des procédés ou simplement des méthodes dictées par la pratique de certains métiers.	-

TABLEAU 20. TYPOLOGIE DES PROCESSUS POUR LA GESTION DE MODELES

2.2.5. L'ontologie de facteurs de qualité logicielle « OntologyQualityFactor »

L'ontologie de facteurs de qualité logicielle définit une terminologie pour préciser l'appréciation globale d'un logiciel. Certains acteurs décrivent la qualité logicielle en termes de « **conformité aux besoins** ». La qualité logicielle est définie par des organisations internationales telles que l'IEEE Standard [IEEE std 729, 1983] comme : La totalité des propriétés et caractéristiques d'un produit logiciel qui lui confèrent leur capacité pour répondre à certaines exigences. L'ensemble des caractéristiques du logiciel détermine le degré d'adéquation du logiciel aux besoins « qualité » du client.

Cette norme définit la qualité logicielle en termes de propriétés et caractéristiques que nous avons intégré sous la forme d'une ontologie.

La figure 57 montre un exemple d'imitation du patron **Concept-Term** pour la modélisation des facteurs de qualité logicielle. Cette ontologie est catégorisée par 13 types de concepts de qualité. Donc, plusieurs termes de qualité correspondent à un même concept de qualité.

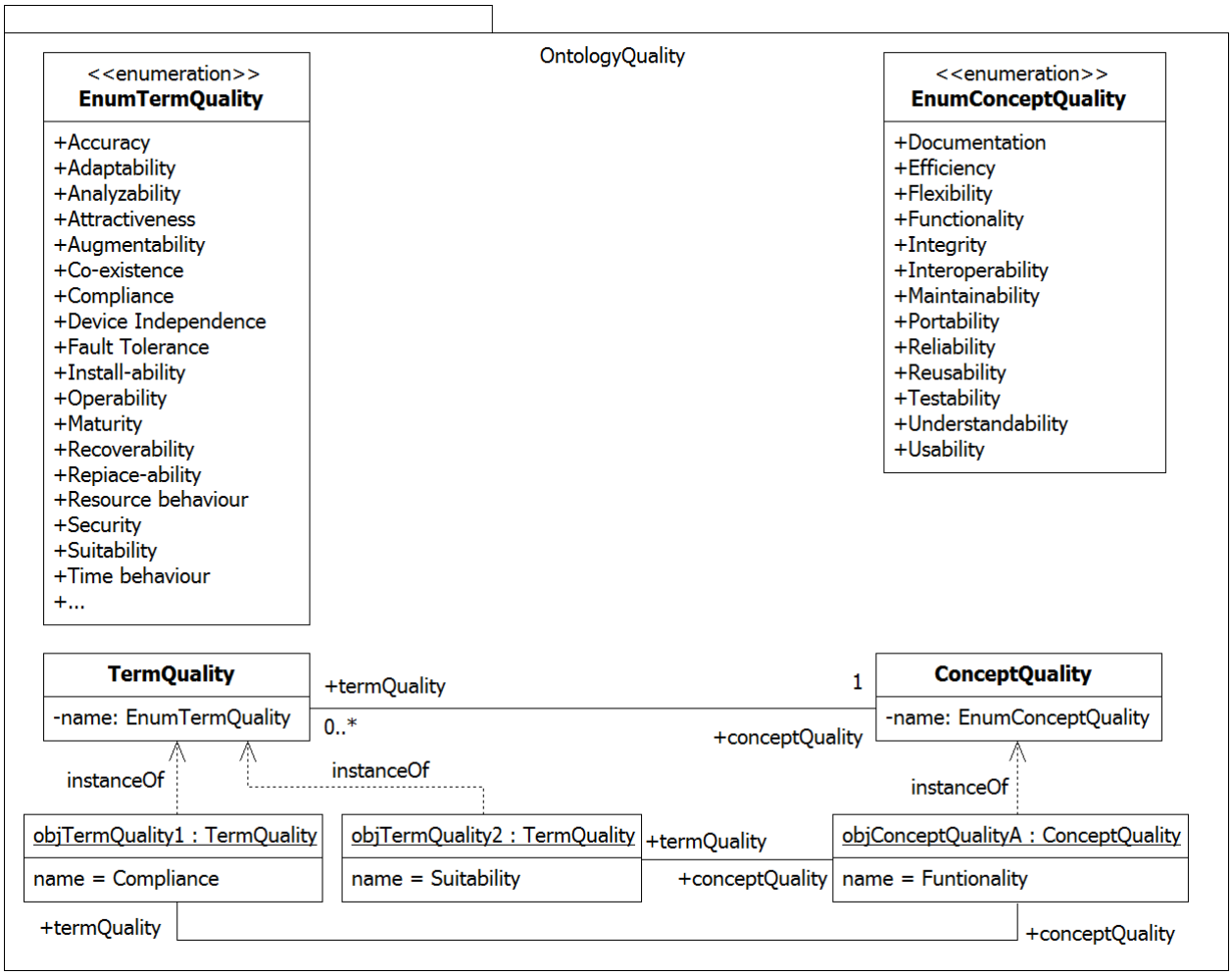


FIGURE 57. EXEMPLE D'IMITATION DU PATRON CONCEPT-TERM POUR LA REPRESENTATION DE L'ONTOLOGIE DE FACTEURS DE QUALITE LOGICIELLE

Le tableau 21 résume l'ontologie des facteurs de qualité inspirée des modèle de qualité telles que : FURPS [Grady, 1992], McCall [McCall et *al.*, 1977], Boehm [Boehm et *al.*, 1978], IEEE 1061[IEEE std 1061, 1992], Dromey [Dromey, 1995], ISO 9126 [ISO std 9126, 1991]. Ces modèles sont présentés sous la forme de facteurs de qualité et critères (McCall [McCall et *al.*, 1977]), facteurs et sous facteurs (IEEE 1061 [IEEE std 1061, 1992]), caractéristiques et sous caractéristiques (ISO 9126 [ISO std 9126, 1991]), etc. Dans le contexte de nos travaux, nous utilisons les termes catégorie de concepts et termes de qualité. Les catégories de concepts de qualité retenues sont : Documentation, Efficiency, Flexibility, Functionality, Integrity, Interoperability, Maintainability, Portability, Reliability, Reusability, Testability, Understandability, Usability.

Concept de qualité (ConceptQuality)	**Terme de Qualité (TermQuality)**
Documentation Le degré auquel le logiciel fournit une documentation significative.	Adéquation de la documentation, compréhension de la documentation.
Efficiency (Efficacité)	Comportement vis-à-vis des ressources.

Exprime la capacité du logiciel à exploiter au mieux les ressources offertes par la ou les machines où le logiciel sera implanté. Le logiciel doit se limiter à l'utilisation des ressources strictement nécessaires à l'accomplissement de ses fonctions.	
Flexibility (flexibilité) L'effort exigé pour modifier un logiciel. Ce concept exprime l'effort pour changer ou modifier un produit logiciel pour l'adapter à d'autre environnement ou à d'autres applications différentes ceux pour lesquels il a été conçu.	Extensibilité, interchangeabilité.
Functionality (Fonctionalité) Capacité d'un produit logiciel à fournir les fonctions qui respectent des besoins exposés et impliqués quand le logiciel est utilisé dans des conditions indiquées.	Conformité, convenance, exactitude, pertinence, sécurité.
Integrity (Intégrité) Exprime la faculté du logiciel à protéger ses fonctions et ses données d'accès non autorisés.	Contrôle d'accès, audit, vérification de l'intégrité du logiciel.
Interoperability (interopérabilité) Exprime la capacité d'un logiciel à interagir avec différents logiciels. C'est-à-dire, la capacité du logiciel à communiquer et à utiliser les ressources d'autres logiciels comme, par exemple, les documents créés par une certaine application.	Communication de données, intégration de divers composants, interopérabilité.
Maintainability (Maintenabilité) Capacité d'un produit logiciel à être modifié. Exprime la simplicité de correction et de modification du logiciel, et même, parfois, la possibilité de modification du logiciel en cours d'exécution. Les modifications peuvent inclure des corrections, des améliorations ou l'adaptation du logiciel pour changer d'environnement et/ou d'exigences fonctionnels.	Stabilité, analysabilité, évolutivité, modificabilité, simplicité, vieillissement (comportement dans le temps).
Portability(Portabilité) Exprime la possibilité de compiler le code source et/ou d'exécuter le logiciel sur des platesformes (machines, systèmes d'exploitation, environnements) différents.	Installabilité, remplaçabilité, adaptabilité, compatibilité, standardisation, conformité.
Reliability (Fiabilité) Capacité du logiciel à maintenir son niveau de service dans des conditions précises et pendant une période déterminée. Exprime la faculté du logiciel à gérer correctement ses propres erreurs de fonctionnement en cours d'exécution.	Tolérance aux erreurs, homogénéité, précision, simplicité, maturité, réalisabilité, complétude, cohérence, possibilité de récupération.
Reusability (Réutilisabilité) Exprime la capacité de concevoir le logiciel avec des composants déjà conçus tout en permettant la réutilisation simple de ses propres composants pour le développement d'autres logiciels.	Indépendance de la plateforme, généralité, modularité.
Testability (Testabilité) Exprime la facilité pour réaliser des procédures de test permettant de s'assurer de l'adéquation des fonctionnalités.	Simplicité, instrumentation, Auto-description.
Understandability (Compréhensibilité) Décrit le degré de compréhension et d'apprentissage du logiciel	Lisibilité, cohérence, concision, simplicité d'utilisation.
Usability (utilisabilité) Décrit la facilité d'utilisation du logiciel. Facilité avec laquelle l'utilisateur peut utiliser un système (généralement une application logicielle).	Guidage, aide, efficacité, efficience, satisfaction, accessibilité.

L'utilisabilité d'une interface est habituellement déterminée selon certains critères liés au comportement de l'utilisateur tels que le temps d'apprentissage, la vitesse d'exécution de la tâche et le nombre d'erreurs commises. L'adéquation de la tâche ainsi que la satisfaction de l'utilisateur sont également pris en compte.	

TABLEAU 21. TYPOLOGIE DES FACTEURS DE QUALITE POUR LA GESTION DE MODELES

2.2.6. Synthèse

L'approche générale :

Nos méta-modèles s'appuient sur des ontologies ou des normes existantes. Dans de nombreux travaux modèles ou méta-modèles et ontologies sont dissociés, l'ontologie servant à « typer » des éléments du modèle ou méta-modèle. Nous avons choisi d'intégrer les ontologies dans nos méta-modèles de manière à offrir une vision intégrée de chaque niveau d'abstraction. Les ontologies sont intégrées sous la forme d'imitation du patron « **Concept – Term** ». Dans cette section, nous avons présenté ce patron ainsi que les ontologies utilisées dans nos méta-modèles.

3. LES CAS D'ETUDE

L'essence de notre étude vise le choix de processus et d'outils de modélisation appropriés. Pour illustrer notre approche, nous nous appuierons sur deux méthodes de conception de systèmes interactifs Symphonie étendue [Juras *et al.*, 2006] et RUP [Kruchten, 2000].

Les sections suivantes présentent ces deux méthodes qui seront utilisées dans les études de cas, présentées dans le reste de ce document.

3.1. La méthode Symphony étendue

Cette section présente une vision générale de la méthode Symphony étendue, utilisée pour la conception des systèmes interactifs (voir la figure 58). Cette méthode est une extension de la démarche de conception Symphony (voir annexe A) [Hassine *et al.*, 2002]. Symphony étendue intègre les pratiques de conception de l'Interaction Homme-Machine (IHM) et un processus de développement du génie logiciel. Symphony est une démarche inspirée des pratiques de l'Unified Process : la démarche est itérative, incrémentale, orientée composants métier et pilotée par les cas

d'utilisation. Elle a été initialement proposée par la société UMANIS. Cependant, Symphony a été étendue récemment pour intégrer la conception de Systèmes Interactifs [Juras *et al.*, 2006].

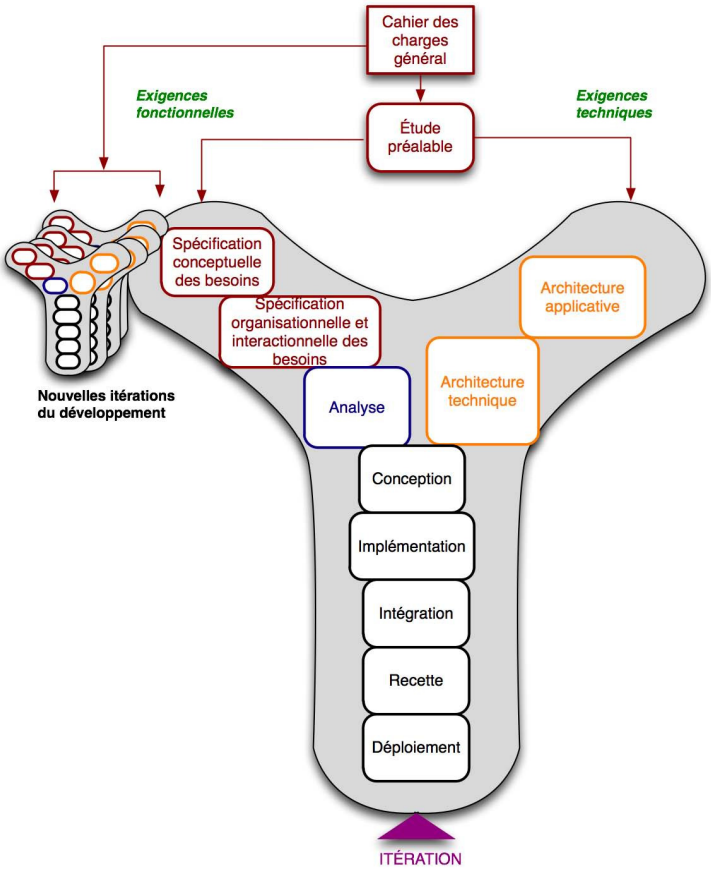


FIGURE 58. « SYMPHONY » UNE METHODE DE CONCEPTION DES SYSTEMES INTERACTIFS [JURAS ET AL., 2006]

Cette nouvelle méthode prend en compte les aspects fonctionnels (méthodes de GL pour le développement du noyau fonctionnel) et l'étude de l'interaction de l'utilisateur. La conception des fonctionnalités est effectuée par les « Spécialistes GL » et les « Spécialistes Métier » qui utilisent des modèles UML.

Concernant les aspects méthodologiques en IHM, Symphony étendue utilise une approche basée sur la modélisation des tâches et des diagrammes spécifiques à la description des dispositifs d'interaction. Les concepteurs sont les « Spécialistes IHM » et les « Ergonomes logiciels ».

La méthode Symphony étendue est constituée de plusieurs phases : spécification conceptuelle des besoins, spécification organisationnelle et interactionnelle des besoins, analyse, architecture applicative, etc. Les phases sont structurées en activités. Ces activités concourent à la production d'un ensemble d'entités appelées ProduitSymphony ou « artefact » qui décrivent les différentes facettes du système final en fonction de celles fournies en entrée. Un Produit Symphony est d'un type donné, il peut s'agir d'un document texte, d'un diagramme UML, d'un exécutable, etc. Une activité

Symphony est réalisée par un ou plusieurs ActeursSymphony, chacun pouvant être le responsable de la réalisation d'un certain nombre de produits du système.

Nous nous concentrons sur la phase de "**Spécification Organisationnelle et Interactionnelle des Besoins**" de la méthode Symphony étendue pour les systèmes mixtes proposée par Juras [Juras *et al.*, 2006] (représentée par la figure 59). Cette phase a pour objectif d'analyser un processus métier à un niveau organisationnel (le « qui fait quoi et quand » du futur système), et d'envisager une interaction homme-machine satisfaisante pour réaliser les activités des différents acteurs. Elle débute par une activité de coordination entre le spécialiste métier et le spécialiste GL qui a pour but de décomposer le processus métier étudié en sous-processus (« Décomposition organisationnelle »). La collaboration entre le Spécialiste GL et le Spécialiste métier pour réaliser une décomposition organisationnelle des Processus Métier (PM) et Processus Métier Composant (PMC) donne lieu à un diagramme d'activités décrivant l'organisation, les flux d'information et de responsabilité entre acteurs. Il va aussi permettre de distinguer les Processus Métier manuels des Processus Métier Informatisés. Le diagramme d'activités pour le PMC permet d'extraire les Processus Métier Informatisés et de les organiser en tant que cas d'utilisation initiaux dans une première version de la cartographie des cas d'utilisation. Cette activité va aussi permettre au Spécialiste métier et au Spécialiste GL de mettre en évidence des cas d'utilisation complémentaires qui n'avaient pas encore été identifiés par le découpage organisationnel.

La collaboration [Blanco *et al.*, 2007] prend en compte la cohérence et la synchronisation des activités entre les différentes spécialités intervenant dans un processus de conception. La figure 59 montre un exemple de la collaboration entre concepteurs. Le formalisme utilisé dans cette figure est une adaptation des diagrammes d'activités d'UML. Les collaborations sont des conteneurs stéréotypés englobant les activités et les spécialistes concernés, l'existence du planning associé, l'affectation d'acteurs à ces activités avec un objectif propre fixé par acteur ou activité.

Une **coordination** (« COORD ») suppose une décomposition de travail en activités de buts similaires. Il s'agit d'une synchronisation sur des modèles distincts. Mintzberg [Mintzberg, 1982] définit la coordination comme un ajustement itératif de résultats exhaustifs. Par résultats exhaustifs, l'auteur laisse suggérer que des acteurs effectuent leurs activités de manière individuelle et que les résultats obtenus sont améliorés par des « va et vient » entre les activités. Cette définition montre un aspect organisationnel de la coordination. En effet, la coordination est vue comme un processus prescriptif qui définit les interdépendances des activités des concepteurs. Se coordonner consiste à décomposer et répartir les activités entre les acteurs de la conception, mais aussi à les recomposer par le biais de liens de dépendances et ce, afin de permettre une organisation et une gestion du processus. La figure 59 montre, par exemple, l'activité de coordination : **Décomposition organisationnelle**.

Une **coopération** (« COOP ») suppose une action commune des différents acteurs partageant un objectif commun et coordonnant leurs actions de manière autonome, sans processus prédéfini. La coopération est centrée sur les modèles. Il s'agit de réaliser un modèle commun à plusieurs acteurs du développement. Dans la figure 59, une activité de coopération est, par exemple, la **Consolidation des Cas d'utilisation**. Dans ces activités, chaque spécialiste apporte ses modèles et la coopération permet de produire des modèles communs.

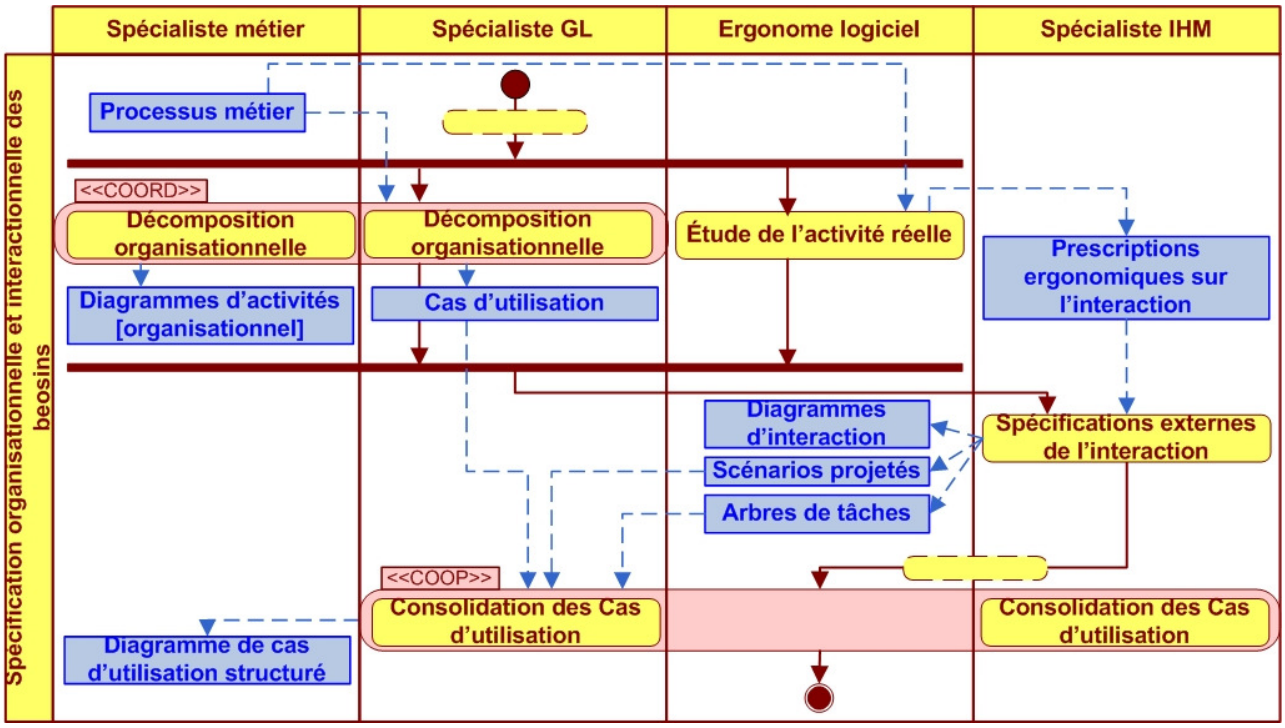


FIGURE 59. ACTIVITES DE SPECIFICATION ORGANISATIONNNELLE ET INTERACTIONNELLE DES BESOINS DE LA METHODE SYMPHONY ETENDUE [JURAS ET AL., 2006]

L'activité suivante correspond aux spécifications externes de l'interaction (voir la figure 60). Dans un premier temps, le Spécialiste IHM réalise une projection de la tâche de l'utilisateur, se basant sur les préconisations ergonomiques de l'interaction, c'est-à-dire envisager la future interaction à mettre en place. Pour cela, dans le cadre particulier de la conception d'un Système Mixte, il débute par la rédaction d'un scénario appelé « projeté abstrait », qui décrit le déroulement d'une situation sans décrire l'interaction à travers les objets physiques et numériques manipulés par l'utilisateur. Un scénario projeté abstrait s'appuie sur les tâches utilisateur afin de décrire le niveau intentionnel du futur système, c'est-à-dire, indépendamment des technologies.

La projection abstraite est suivie d'une projection concrète qui permet d'identifier les modes (sens) utilisés par l'utilisateur lors de l'interaction et de proposer des types de dispositifs génériques permettant de réaliser cette interaction mixte.

Puis les scénarios de projection concrète sont enrichis par une formalisation sous forme d'un arbre de tâches représentant la décomposition fonctionnelle des activités de l'utilisateur et du système.

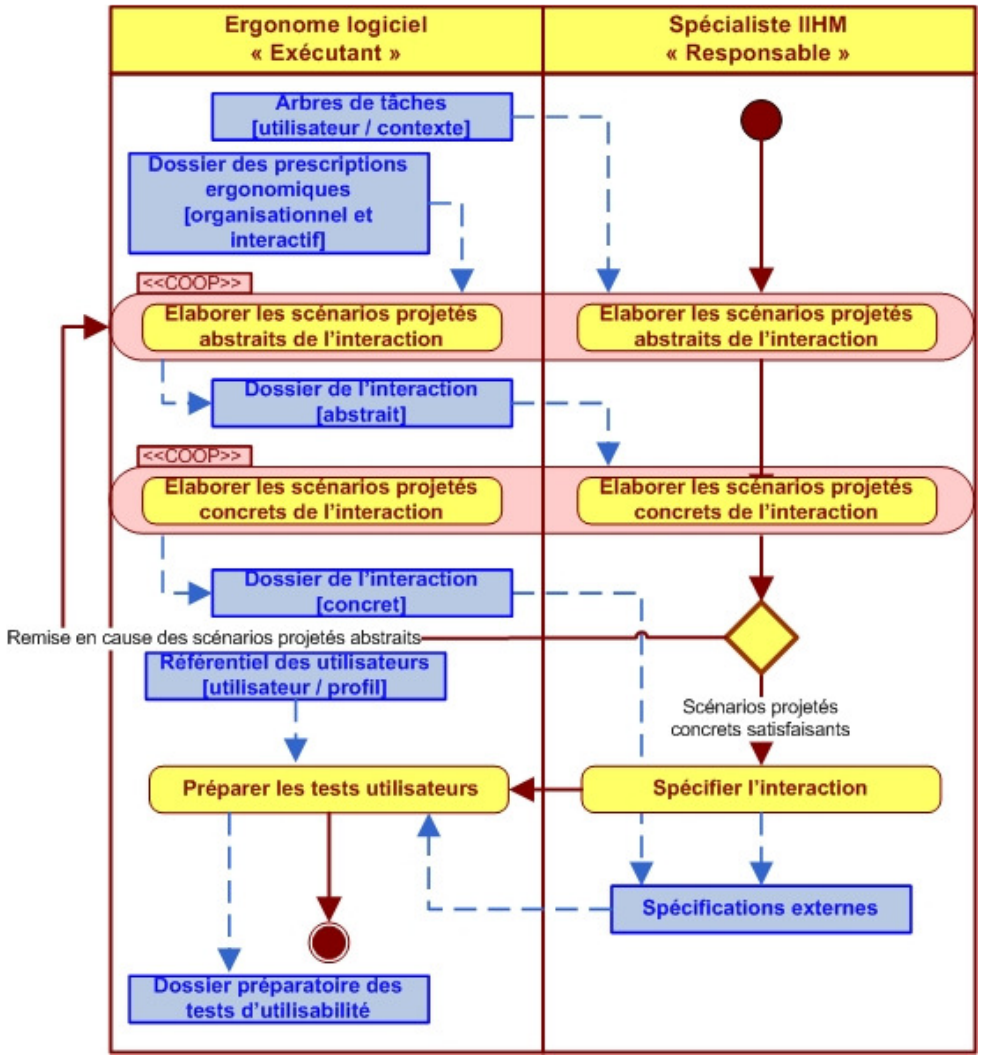


FIGURE 60. ACTIVITE DE SPECIFICATION EXTERNE DE L'INTERACTION DE LA METHODE SYMPHONY ETENDUE [JURAS ET AL., 2006]

Pour terminer cette spécification interactionnelle, chaque tâche interactive de l'arbre de tâches va donner lieu à un diagramme d'interaction mixte qui synthétise l'assemblage des dispositifs génériques choisis pour réaliser la tâche.

La phase se termine par une activité de coopération entre spécialiste GL et IHM qui consiste à faire le lien entre la spécification métier et la spécification interactionnelle.

3.2. La méthode RUP

Rational Unified Process (RUP) [Jacobson et al., 1999] est un processus de développement de logiciel itératif initialement proposé par [Jacobson et al., 1999] et étendu par [Kruchten, 2000].

RUP est basé sur un ensemble de éléments, décrivant : 1) ce qui doit être produit (les artefacts) ; 2) les compétences nécessaires exigées (les rôles) ; 3) des descriptions détaillées de comment des buts spécifiques de développement sont réalisés (les activités et les étapes d'activité) et quand ils doivent être réalisés (les enchaînements d'activités).

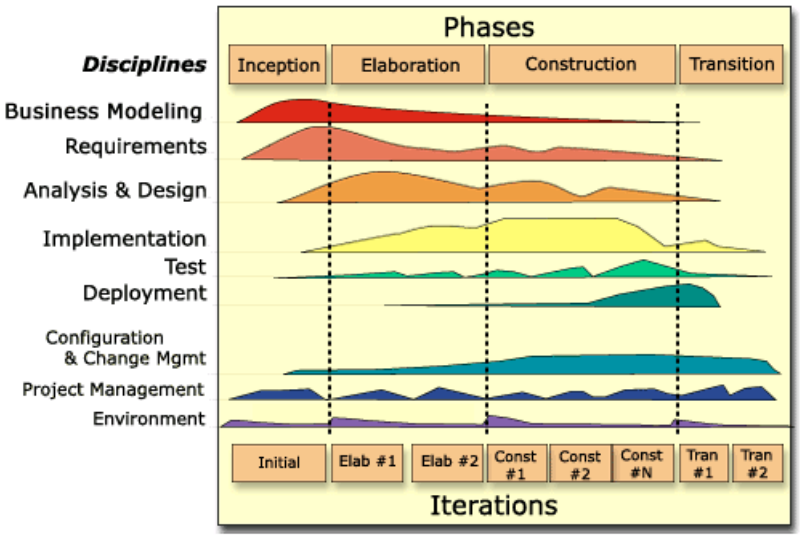


FIGURE 61. VUE GENERALE DE LA METHODE RUP [KRUCHTEN, 2000]

RUP établit quatre phases de développement : inception, élaboration, construction, transition (voir figure 61). Chaque phase est organisée dans un certain nombre d'itérations séparées qui doivent satisfaire des critères définis avant que la phase suivante soit effectuée. A l'intérieur de chaque phase, les tâches sont catégorisées dans neuf disciplines : six disciplines d'ingénierie (par exemple, modélisation métier, gestion des exigences, analyse et conception, implémentation, tests, déploiement) et trois disciplines de soutien (par exemple, gestion de la configuration et des changements, direction de projet, environnement).

Nous décrirons ci-dessous **la phase d'élaboration** en nous focalisant sur la modélisation de l'interface de l'utilisateur de Kruchten. La figure 62 montre l'enchaînement d'activités de gestion des exigences correspondant à la phase d'élaboration.

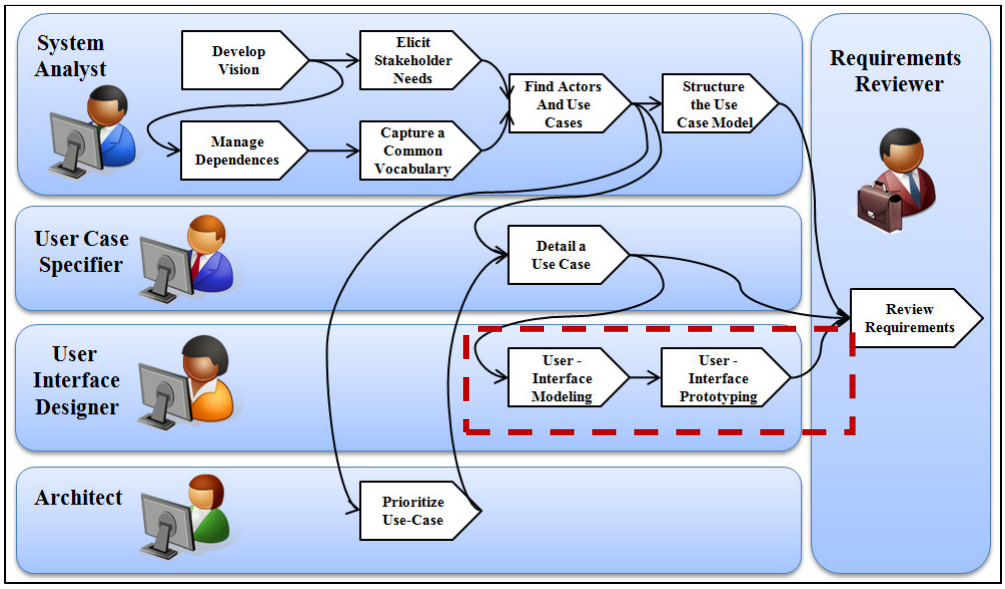


FIGURE 62. ENCHAINEMENT DES ACTIVITES DE GESTION DES EXIGENCES DE LA PHASE D'ELABORATION DE LA METHODE RUP [KRUCHTEN, 2000]

Dans cette phase les concepteurs de modèles spécifient les besoins et définissent une architecture de base. Plus précisément les objectifs de la phase d'élaboration sont les suivants :

- créer et valider une architecture de référence ;
- décrire une vision de référence du produit ;
- définir un plan de référence très détaillé pour la phase de construction ;
- démontrer que l'architecture de référence est compatible avec la vision du produit, et ce, pour un coût acceptable et dans des délais raisonnables.

Concernant les interfaces utilisateurs, les spécialistes IHM dirigent et coordonnent la conception et le prototypage de l'interface de l'utilisateur. Pour modéliser l'interface utilisateur, les spécialistes IHM se basent sur **le modèle des cas d'utilisation,** mais également sur des **spécifications supplémentaires** pour analyser les exigences non fonctionnelles (telles que : esthétique, facilité d'apprentissage et d'utilisation) et **les attributs des exigences.** Ces derniers correspondent aux caractéristiques des différents types d'exigences et leurs dépendances. Le résultat de la modélisation de l'interface utilisateur produit les modèles suivants : 1) Les « storyboards » des cas d'utilisation ; 2) Un modèle de description d'acteurs ; 3) Les « Boundary Classes » qui montrent comment l'interface utilisateur est réalisée dans le modèle d'analyse.

Dans cette section nous avons présenté deux méthodes de conception de systèmes interactifs. Plus précisément, nous avons détaillé les processus qui permettent d'envisager une interaction homme-machine satisfaisante. La section suivante présente la couche intentionnelle où nous abordons les aspects concernant à la modélisation d'un service intentionnel.

4. LA COUCHE INTENTIONNELLE

4.1. La modélisation d'un service intentionnel

Cette section présente le méta-modèle de service intentionnel. Un **service intentionnel** est un service orienté métier décrit d'un point de vue des objectifs (par exemple : spécifier un système interactif, identifier les cas d'utilisation, élaborer un scenario, personnaliser le processus de spécification de besoins…). Un service intentionnel correspond à la modélisation des buts de gestion de modèles. Un service intentionnel peut être composé de services plus élémentaires et est réalisé par des services organisationnels correspondant à des fragments de méthodes. La figure 63 montre une vision globale du méta-modèle intentionnel.

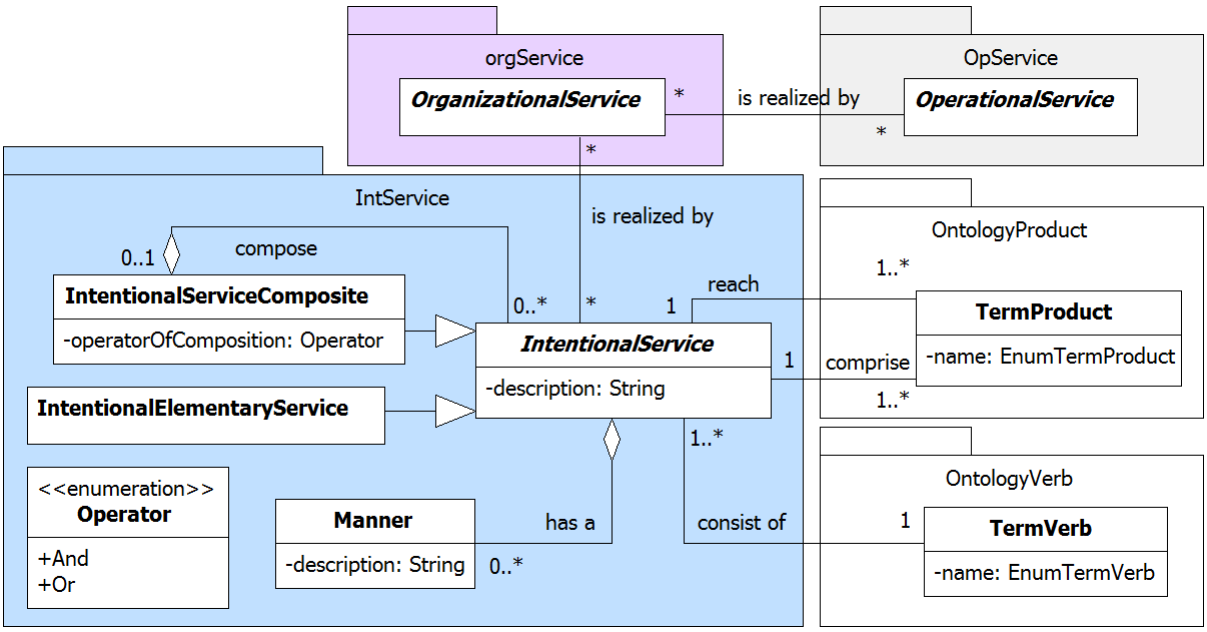


FIGURE 63. META-MODELE DE SERVICES INTENTIONNELS

Un service intentionnel est défini par **un verbe** représenté par l'ontologie de verbes « TermVerb » décrit dans la section précédente. Un service intentionnel est également associé à des **produits** représentés par l'ontologie de produits « TermProduct » décrite dans la section précédente. Il s'agit des produits utiles à la réalisation du service (association « comprise ») et aux produits générés par le service (association « reach »). De plus, un service intentionnel est lié à une façon de faire, une manière, pour réaliser l'intention.

Par la suite, nous détaillerons ces éléments (verbe, objet produit et manière) puis nous expliquerons comment nous avons défini le formalisme d'expression des services intentionnels.

L'élément central d'un service intentionnel est le **verbe**, qui est représenté par la classe « TermVerb » de l'ontologie de verbes. Les verbes décrivant les intentions spécifiques pour la gestion de modèles, par exemples : **améliorer** un processus métier, **personnaliser** le processus de spécification de besoins, **élaborer** un scénario, **automatiser** une procédure administrative, sont donc des intances de « TermVerb » ; Ils sont rattachés à leur catégorie (instance de « ConceptVerb »).

Le produit est l'élément qui complète le verbe pour décrire l'intention, cet élément est représenté par la classe « TermProduct » (figure 63) de l'ontologie de produits « OntologyProduct » qui regroupe les différents objets qui peuvent être élaborés, modifiés ou utilisés au cours de la gestion de modèles comme scénario ou processus métier. Ce sont les objets sur lesquels portent les actions. Ils sont instance de « TermProduct » et sont rattachés à leur catégorie (instance de « ConceptProduct ») : par exemple, modèle, méta-modèle, documentation. Dans la figure 61 seule la classe TermProduct apparait.

La manière (voir figure 64) est une façon complémentaire d'exprimer l'intention du verbe, c'est un élément qui permet de décrire une approche possible pour réaliser le service intentionnel. Par exemple, dans le but « spécifier les besoins avec l'utilisation des scénarios », la phrase « avec l'utilisation des scénarios » correspond à la manière de procéder pour résoudre le but à atteindre.

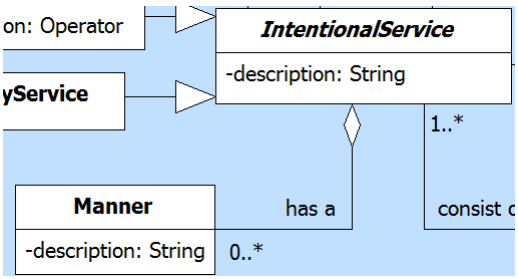


FIGURE 64. EXTRAIT DU META-MODELE DE SERVICE INTENTIONNEL

4.2. Le formalisme d'expression des services intentionnels

4.2.1. La construction lexicale des intentions

Nous utilisons une approche linguistique pour formuler une intention. Cette approche s'appuie sur la déclaration structurelle d'une intention proposée par Rolland [Rolland, 2007]. Ces travaux sur les intentions ont déjà été utilisés pour les services web [Lee *et al.*, 2005]. Dans ce travail, un but est constitué de contenus (action, contraintes et paramètres tels que l'objet, le résultat, la source, la

destination), de propriétés (compétences, vues, pré et post-condition) et d'un plan. De façon similaire, nous avons retenu dans notre travail les éléments constitutifs du contenu. La déclaration structurelle reprend les éléments du méta-modèle de service intentionnel (figure 63). En conséquence, nous avons ici structuré les intentions de la manière suivante :

- **Intention<**
  <TermVerb(name)>
  {<**ObjectComprise**(name)>} -- {} élément répétitif
  {<**ObjectReach**(name)>}
  [***Manner***] -- {} élément facultatif
  >

L'élément "***Manner***" qui se trouve dans les crochets « [] » corresponde à un élément facultatif. Nous présentons dans le tableau ci-dessous deux exemples.

Exemple 1:	
Intention	Spécifier les fonctionnalités d'un système interactif
Forme générale	**Intention<** **<TermVerb**(Spécifier)> **<ObjectComprise**(les fonctionnalités)> **<ObjectReach**(Cartographie des processus métier d'un système interactif)> **>**
Exemple 2:	
Intention	Spécifier les besoins d'un système avec l'utilisation des scénarios
Forme générale	**Intention<** **<TermVerb**(Spécifier)> **<ObjectComprise**(les besoins d'un système)> **<ObjectReach**(Diagramme des scénarios)> [***Manner***(avec l'utilisation des scénarios)] **>**

TABLEAU 22. EXEMPLES DE LA STRUCTURE D'UNE INTENTION

Par la suite, nous présentons un formalisme de représentation graphique des services intentionnels.

4.2.2. Le formalisme de représentation graphique des services intentionnels

Nous proposons un formalisme d'expression graphique des services intentionnels afin de représenter les instances du méta-modèle de services intentionnels. Le tableau 23 propose le formalisme pour exprimer les services intentionnels. Notons que dans la représentation d'un service intentionnel, sa description correspond à la structure d'une intention présentée précédemment.

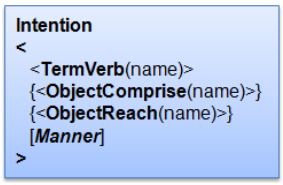
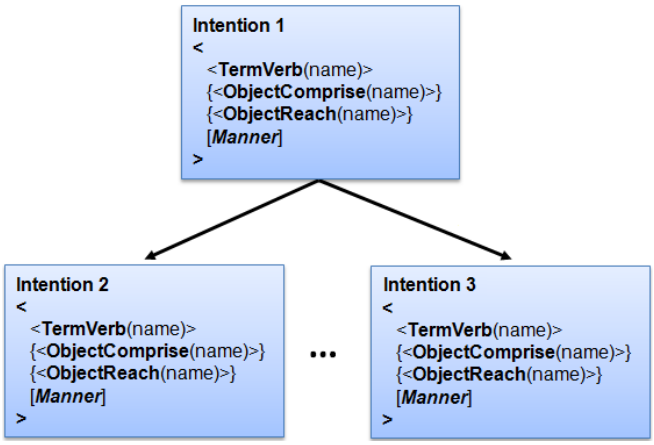
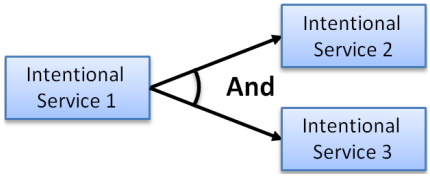
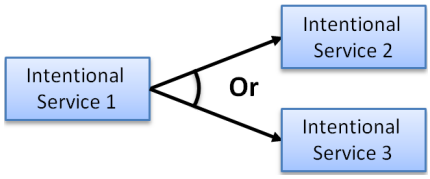
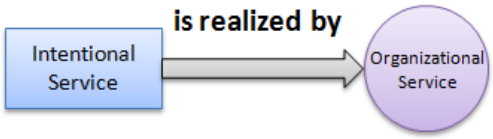
Element	Formalism
Service intentionnel	
Composition des services intentionnels	
Association of composition	Operation of composition – AND 
	Operation of composition – Or 
Association between an intentional service and an organizational service	

TABLEAU 23. FORMALISME D'EXPRESSION DES SERVICES INTENTIONNELS

La formalisation de la décomposition des services intentionnels est inspiré des arbres ET/OU [Yue, 1987]. Un graphe ET /OU modélisant les buts d'un système est d'utilité primordiale en ingénierie des exigences : il fournit un critère de complétude et de pertinence des exigences ; il permet de structurer les informations par couches d'abstraction (raffinements – ET) ; il met en évidence les alternatives à évaluer et à négocier (raffinements – OU). La figure 65 montre l'arbre de décomposition des services que nous proposons.

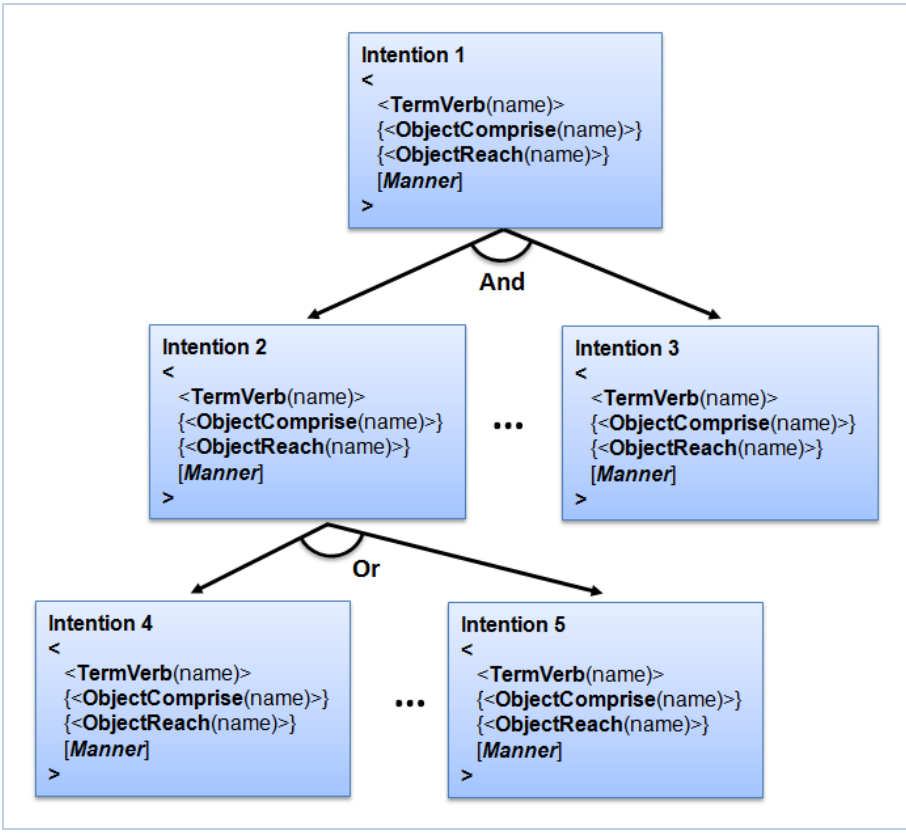


FIGURE 65. REPRESENTATION GRAPHIQUE UTILISEE POUR VISUALISER LA DECOMPOSITION DES SERVICES INTENTIONNELS

4.3. Exemple

Cette section présente un exemple pour illustrer le méta-modèle de service intentionnel. Il est basé sur la méthode de conception Symphony étendue proposée par Juras [Juras *et al.*, 2006]. Au niveau intentionnel, l'approche que nous proposons doit garantir un environnement de modélisation des buts stratégiques requis par les spécialistes GL et les spécialistes de l'IHM, qui participent au processus de développement des systèmes interactifs (le contexte de notre exemple). Les services intentionnels sont représentés sous la forme de graphes ET/OU dont les nœuds sont structurés suivant la syntaxe définie dans la section 4.2.1.

La représentation de la figure 66 montre une composition des buts à atteindre pour le but principal de spécification d'un système interactif. Les rectangles correspondent aux services et les liens décrivent la composition des services intentionnels. Par exemple : le service ***"Specify an Interactive System"*** est décomposé par les services : "***Specify Functional aspects***" qui prend en compte les besoins des concepteurs du GL, "***Specify Interactional aspects***" qui correspond aux besoins des spécialistes de l'IHM et "***Specify Software architecture***" qui peut être un besoin commun aux spécialistes des deux communautés. Ici, nous avons défini une décomposition basée sur la relation de raffinement « And ».

De la même façon, le service "***Specify Interactional aspects***" peut être décomposé par les services : "***Design the User Interfaces***" pour résoudre les besoins de spécification des IHM du système interactif, "***Specify the Interaction Process***" pour étudier l'interaction et le comportement de l'utilisateur et "***Concretize the User Interface***" pour développer l'interface concrète du système interactif. Nous remarquons que tous ces services peuvent être décomposés, cependant nous nous limiterons à la décomposition du service de spécification des aspects interactionnels. Finalement, tous les services peuvent être liés à des services organisationnels pour proposer une solution (en termes de processus) aux buts correspondants.

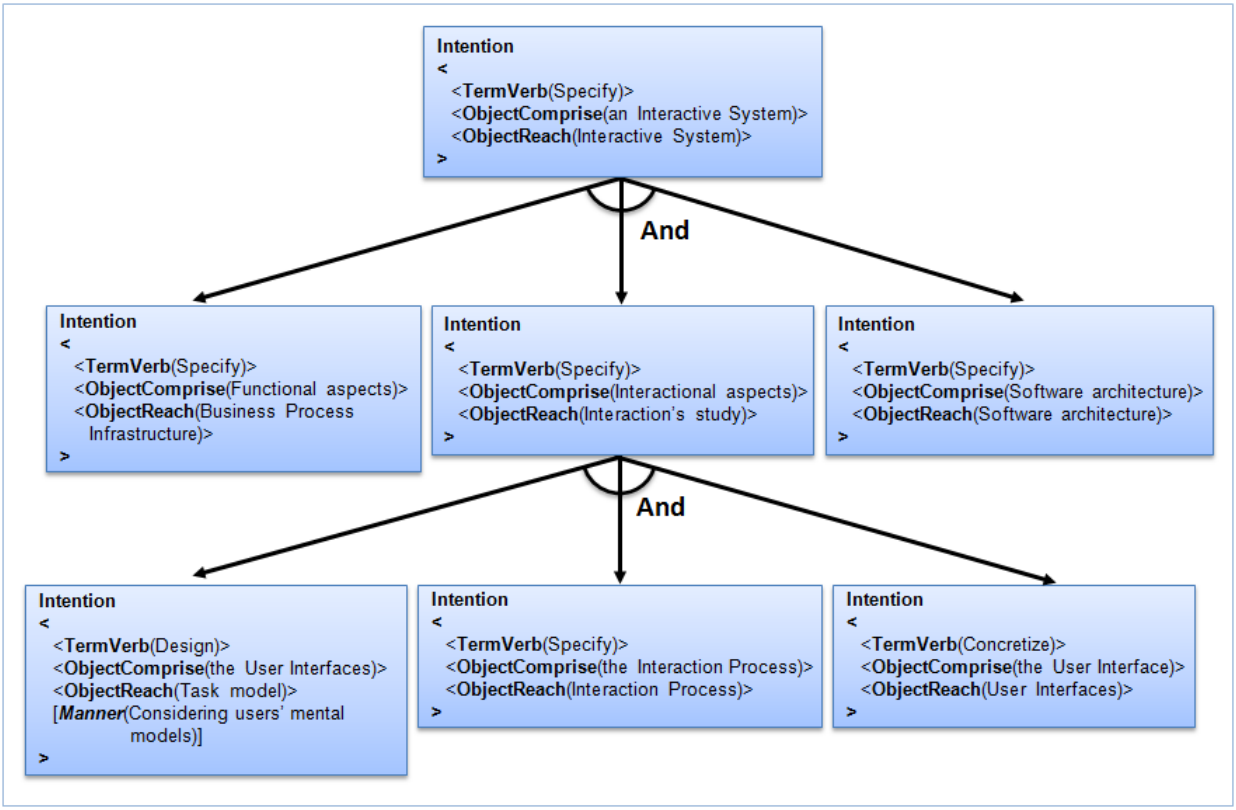


FIGURE 66. L'INSTANCE D'UN META-MODELE DE SERVICE INTENTIONNEL

5. LA COUCHE ORGANISATIONNELLE

5.1. La modélisation d'un service organisationnel

Cette section présente le méta-modèle de la couche organisationnelle. Les services organisationnels (figure 67) sont inspirés des travaux de Ralyté [Ralyté *et al.*, 2001a] qui propose un modèle de processus pour l'ingénierie de méthodes et du modèle de coordination de Ellis [Ellis *et al.*, 1994] qui considère qu'une activité est un ensemble d'opérations qu'un acteur interprétant un rôle particulier

peut effectuer pour résoudre un but défini. L'ingénierie de méthodes permet de représenter un processus de conception comme un ensemble de fragments de méthodes réutilisables. Dans le contexte de notre travail, nous utilisons la notion de services pour capitaliser et réutiliser des fragments de méthodes.

Rappel :
Nous rappelons que notre objectif est uniquement de fournir à chaque concepteur de modèle l'environnement logiciel lui permettant de mener à bien ses activités. Le méta-modèle de service organisationnel que nous proposons est donc très simple, en particulier les enchainements de tâches ne sont pas représentés.

Un service organisationnel correspond à un ou plusieurs fragments de méthodes existantes (**TermProcess** dans l'ontologie **OntologyProcess**) : par exemple, les fragments illustrés en figure 56 (cycle de vie de Symphony étendue), figure 59 (spécification organisationnel et interactionnel des besoins de Symphony étendue), figure 61 (cycle de vie de la méthode RUP), etc. De plus un service organisationnel peut être composé par d'autres services organisationnels.

La composition d'un service organisationnel ainsi que la correspondance entre un service organisationnel et les fragments existants sont représentés dans le paquetage « **orgService** ». La pertinence d'une composition de services est laissée à l'appréciation de l'ingénieur de méthodes. Néanmoins la composition est facilitée par l'utilisation d'un méta-modèle commun et par un vocabulaire commun défini dans nos ontologies. Au niveau le plus fin de la décomposition, un service élémentaire organisationnel est défini en termes de manipulation de modèles (figure 67).

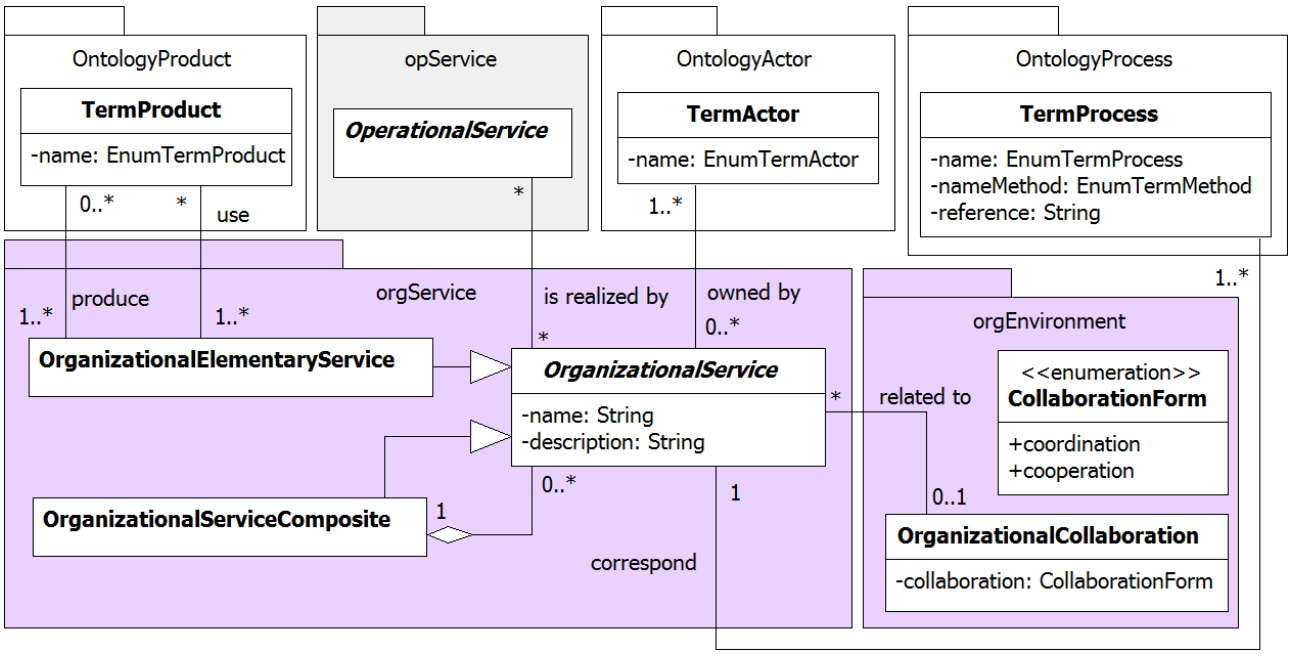


FIGURE 67. META-MODELE DE SERVICES ORGANISATIONNELS

Dans notre méta-modèle de service organisationnel (figure 67), un fragment de méthode est représenté par un service élémentaire organisationnel défini en termes de manipulation de modèles. **La manipulation des produits** consiste en la description de chaque service élémentaire sous la forme d'actions sur les produits utilisés ou produits représentés par la classe « TermProduct » de la Ontologie de Produits « OntologyProduct » : par exemple, un processus de spécification interactionnelle des besoins, produit un modèle de tâche de l'utilisateur qui peut être utilisé par d'autres fragments de méthode. Nous utilisons ici l'ontologie des produits (« OntologyProduct »).

Un service organisationnel est effectué par un ou plusieurs rôles joués par des acteurs. Les acteurs correspondent aux rôles responsables de la réalisation d'un service organisationnel. Pour ce faire, nous utilisons l'ontologie des acteurs « OntologyActor » qui définit un ensemble de rôles (concepteur, développeur, architecte, etc.) pour les acteurs qui ont en charge les problèmes d'ingénierie [Guzélian, 2007].

Un concepteur de modèles peut définir plusieurs services organisationnels. Ils peuvent avoir des visions différentes et complémentaires sur le système à modéliser. Chacun peut avoir besoin de conserver ses habitudes de modélisation et prévoir des activités collaboratives afin de synchroniser ses travaux sur des objectifs communs et/ou des modèles consensuels. Par conséquent, il est nécessaire de garantir la traçabilité et la cohérence entre les modèles utilisés par les différents spécialistes. Au niveau organisationnel nous ne prenons en compte que la forme (coordination ou coopération) de la collaboration.

A ce niveau, le **terme de collaboration** [Blanco *et al.*, 2007] est utilisé pour représenter les tâches de coordination et de coopération entre les concepteurs. La collaboration prend en compte la cohérence et la synchronisation des activités entre les différents spécialistes intervenant dans un processus de conception des systèmes interactifs. Les activités de **coordination** reposent sur une décomposition du travail en activités de buts similaires. Il s'agit d'une synchronisation sur des modèles distincts, alors que les activités de **coopération** sont basées sur un objectif de modélisation commun. La coopération est centrée sur les modèles. Il s'agit de réaliser un modèle commun à plusieurs acteurs du développement. Chaque spécialiste apporte ses modèles et la coopération permet de produire des modèles consensuels ou communs.

La figure 68 montre l'extrait du méta-modèle des services organisationnels représentant la collaboration.

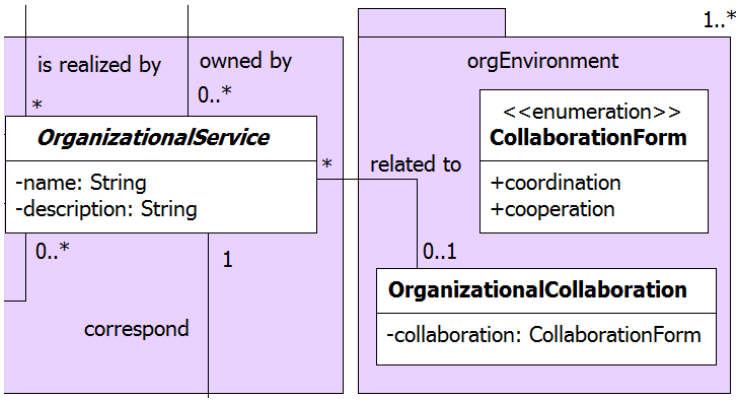


FIGURE 68. EXTRAIT DU META-MODELE DE SERVICE ORGANISATIONNEL REPRESENTANT LA COLLABORATION

Un fragment de méthode « **TermProcess** » correspondant à un service organisationnel est caractérisé par le nom de la méthode (par exemple : Symphony étendue), le nom du fragment (par exemple : spécification organisationnel et interactionnel des besoins de Symphony étendue), et une référence vers des sources bibliographique. Chaque fragment correspond à un « **TermProcess** » défini dans l'ontologie de processus « **OntologyProcess** ». Par exemple : le fragment Spécification organisationnel et interactionnel des besoins de Symphony étendue.

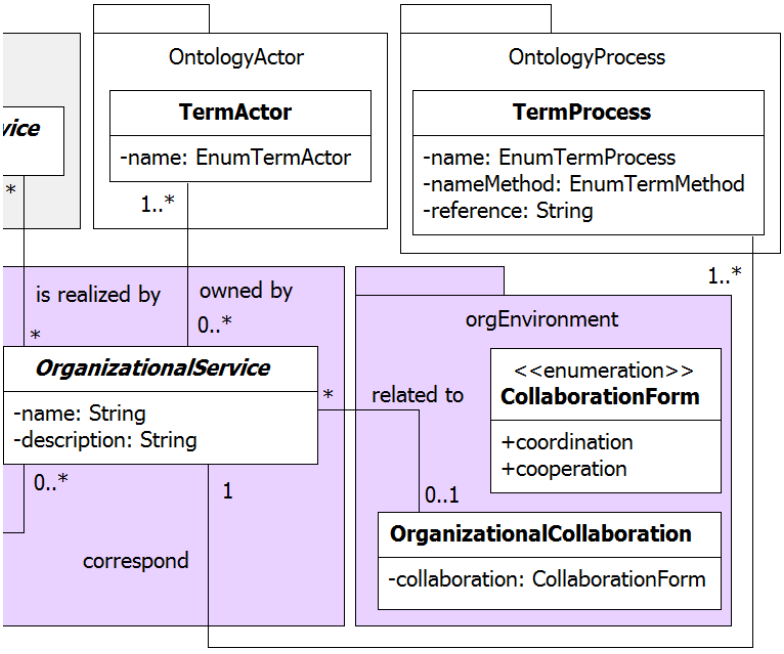


FIGURE 69. EXTRAIT DU META-MODELE DE SERVICE ORGANISATIONNEL REPRESENTANT L'ASPECT PROCESSUS

Un autre aspect considéré par notre modèle organisationnel est le fait que les services opérationnels sont supportés par les services organisationnels (voir figure 70). Cela signifie que les services organisationnels élémentaires utilisent les services opérationnels pour supporter la gestion

des activités de modélisation. Par exemple, un service organisationnel qui comprend la description du modèle de tâches peut être lié aux services opérationnels qui offrent toutes les fonctionnalités pour faciliter l'édition des arbres de tâches. Nous verrons dans la section 6 qu'un service organisationnel peut être supporté pour plusieurs outils définis en termes de services opérationnels.

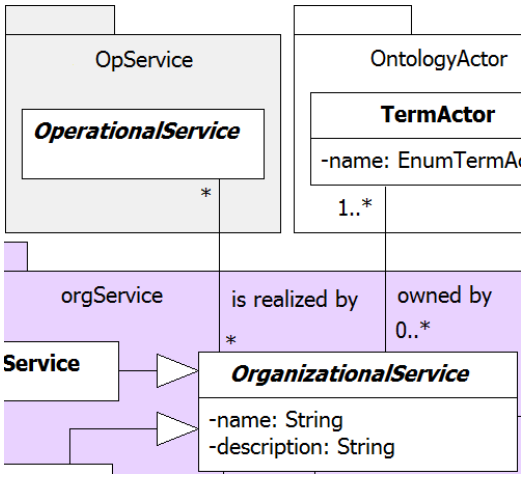


FIGURE 70. EXTRAIT DU META-MODELE DE SERVICE ORGANISATIONNEL REPRESENTANT LA RELATION PARMI UN SERVICE ORGANISATIONNEL ET UN SERVICE OPERATIONNEL

Par la suite, nous présenterons un formalisme de représentation graphique pour les services organisationnels.

5.2. Le formalisme d'expression des services organisationnels

Nous proposons dans le tableau 24 un formalisme d'expression des services organisationnels afin de représenter les objets instances du méta-modèle de services organisationnels.

Element	Formalism
Organisationnel Service	**Organizational Service [collaboration]**

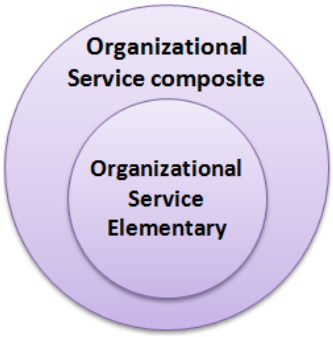
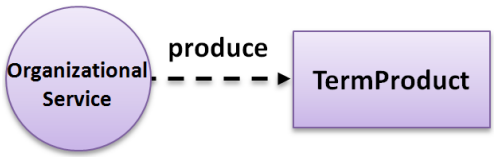
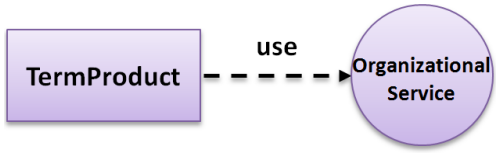
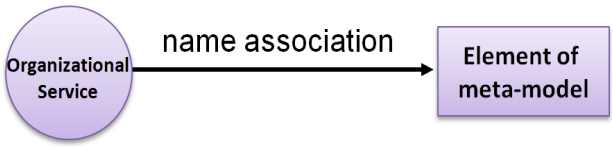
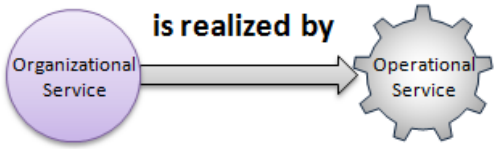
Composition of organizational services		
TermProduct (Model)		
	produced model	
	used model	
TermActor (Role)		
TermProcess		
Association Organizational Service – elements of meta-model (except TermProduct)		
CollaborationForm	Coordination	**[COORD]**
	Cooperation	**[COOP]**
Association between an organizational service and an operational service		

TABLEAU 24. FORMALISME D'EXPRESSION DES SERVICES ORGANISATIONNELS

5.3. Exemples

Cette section présente trois exemples pour illustrer le méta-modèle du service organisationnel. Nos exemples reprennent les fragments de méthodes : Symphonie étendue [Juras *et al.*, 2006] et RUP [Kruchten, 2000].

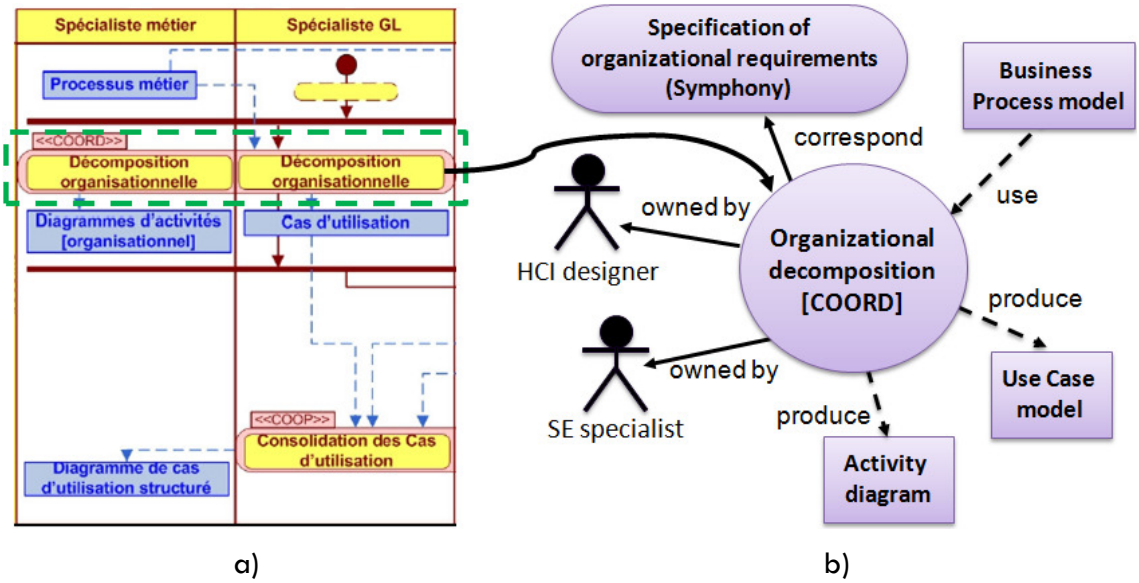
5.3.1. Les fragments de la méthode Symphony étendue [Juras *et al.*, 2006]

Cette section présente deux exemples pour illustrer le méta-modèle du service organisationnel. Nos exemples reprennent les fragments de méthodes présentés dans le tableau suivant qui correspond à la phase de spécification organisationnelle et interactionnelle des besoins de la méthode Symphony étendue [Juras *et al.*, 2006] pour les systèmes mixtes proposée par Juras et décrite dans la figure 59. Le premier exemple correspond à l'activité de coordination : **« Décomposition organisationnelle »**. Le deuxième exemple correspond à l'activité : **« Spécification externe de l'interaction »**.

Exemple 1: Décomposition organisationnelle	
Description générale du fragment	Lors de cette phase, le Spécialiste GL et le Spécialiste métier collaborent pour réaliser une décomposition organisationnelle des Processus Métier (PM) et Processus Métier Composant (PMC). Cette décomposition donne lieu à un diagramme d'activités décrivant l'organisation, les flux d'information et de responsabilité entre acteurs. Il va aussi permettre de distinguer les Processus métier manuels des Processus métier informatisés (PMI). Le diagramme d'activités pour le PMC permet d'extraire les PMI et de les organiser en tant que cas d'utilisation initiaux dans une première version de la cartographie des cas d'utilisation. Cette activité va aussi permettre au Spécialiste métier et au Spécialiste GL de mettre en évidence des cas d'utilisation complémentaires qui n'avaient pas encore été identifiés par le découpage organisationnel.
Exemple 2: Spécification externe de l'interaction	
Description générale du fragment	Dans un premier temps, le Spécialiste IHM réalise **une projection de la tâche de l'utilisateur,** se basant sur les préconisations ergonomiques sur l'interaction, c'est-à-dire envisager la future interaction à mettre en place. Pour cela, dans le cadre particulier de la conception d'un Système Mixte, il débute par la rédaction d'un scénario projeté abstrait décrivant l'interaction à travers les objets physiques et numériques manipulés par l'utilisateur. La projection abstraite est suivie d'une **projection concrète** qui permet d'identifier les modes (sens) utilisés par l'utilisateur lors de l'interaction et de proposer des types de dispositifs génériques permettant de réaliser cette interaction mixte. Puis les scénarios projetés concrets sont enrichis par une **formalisation sous forme d'un arbre de tâches** représentant la décomposition fonctionnelle des activités de l'utilisateur et du système. Pour terminer cette spécification interactionnelle, chaque tâche interactive de l'arbre de tâches va donner lieu à **la génération d'un diagramme d'interaction mixte.** Ce diagramme peut être créé sous la forme d'un diagramme ASUR.

TABLEAU 25. EXEMPLES DE FRAGMENTS DE METHODE

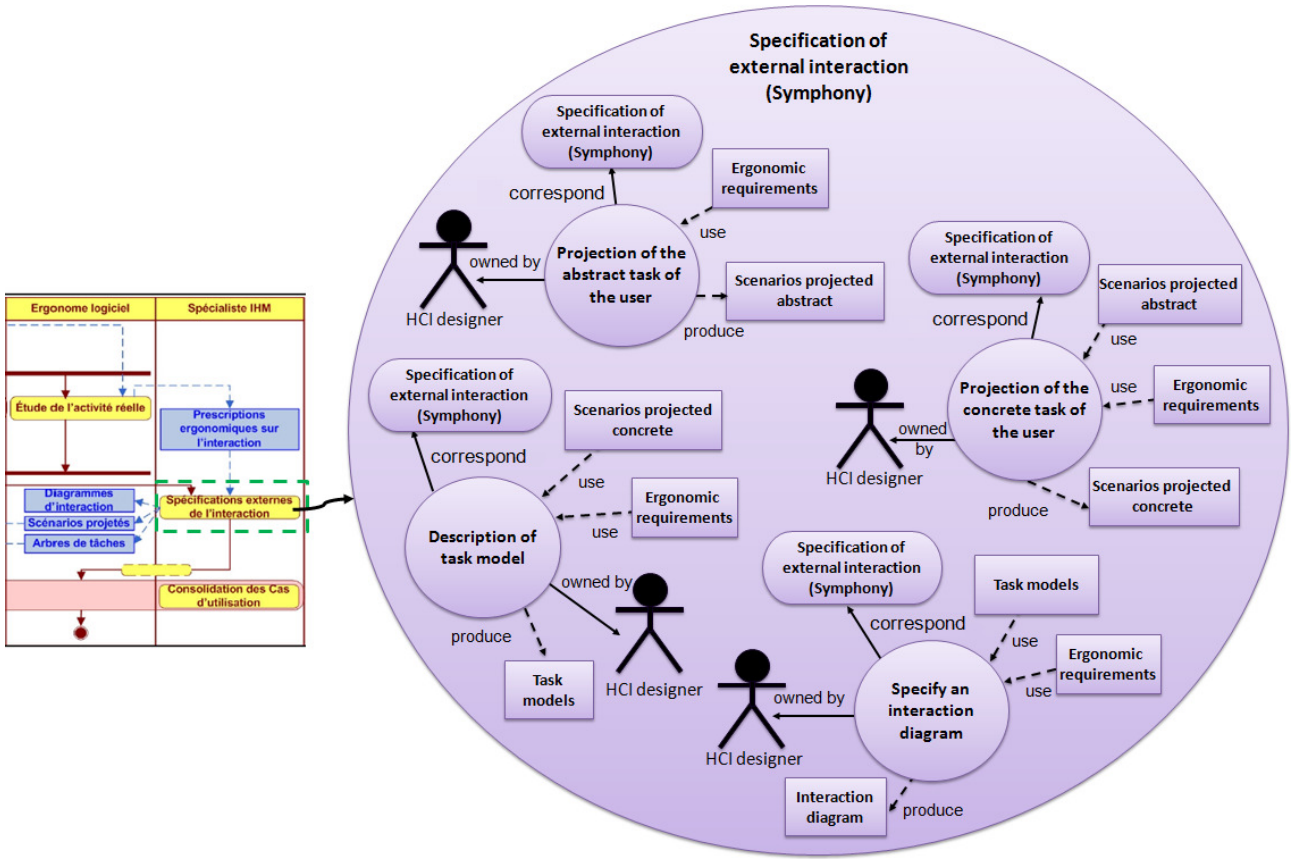
Chacun de ces fragments donnent lieu à un service organisationnel. La figure 71 montre la représentation graphique du service « **Décomposition organisationnelle** ». Pour être intégré dans notre environnement de gestion de modèles, le fragment de méthode à récupérer (par exemple, l'encadré de la figure 71 partie a) doit être analysé et défini en termes de services organisationnels. Il s'agit ici d'avoir un fragment de méthode conforme au méta-modèle de service organisationnel (figure 71 partie b) et décrit suivant le formalisme introduit au tableau 24.



a) b)

FIGURE 71. LA FORMALISATION DU FRAGMENT DE PROCESSUS « DECOMPOSITION ORGANISATIONNELLE » DE LA METHODE SYMPHONY ETENDUE

La figure 72 montre la représentation graphique du fragment de méthode « **Spécification externe de l'interaction** ». Ce fragment correspond dans notre environnement un service organisationnel appelé « **Specification of external interation** » composé de quatre sous-services (un par activité). Comme dans l'exemple précédent, nous proposons en partie à gauche (a) le fragment de méthode à récupérer et à intégrer en tant que service organisationnel.



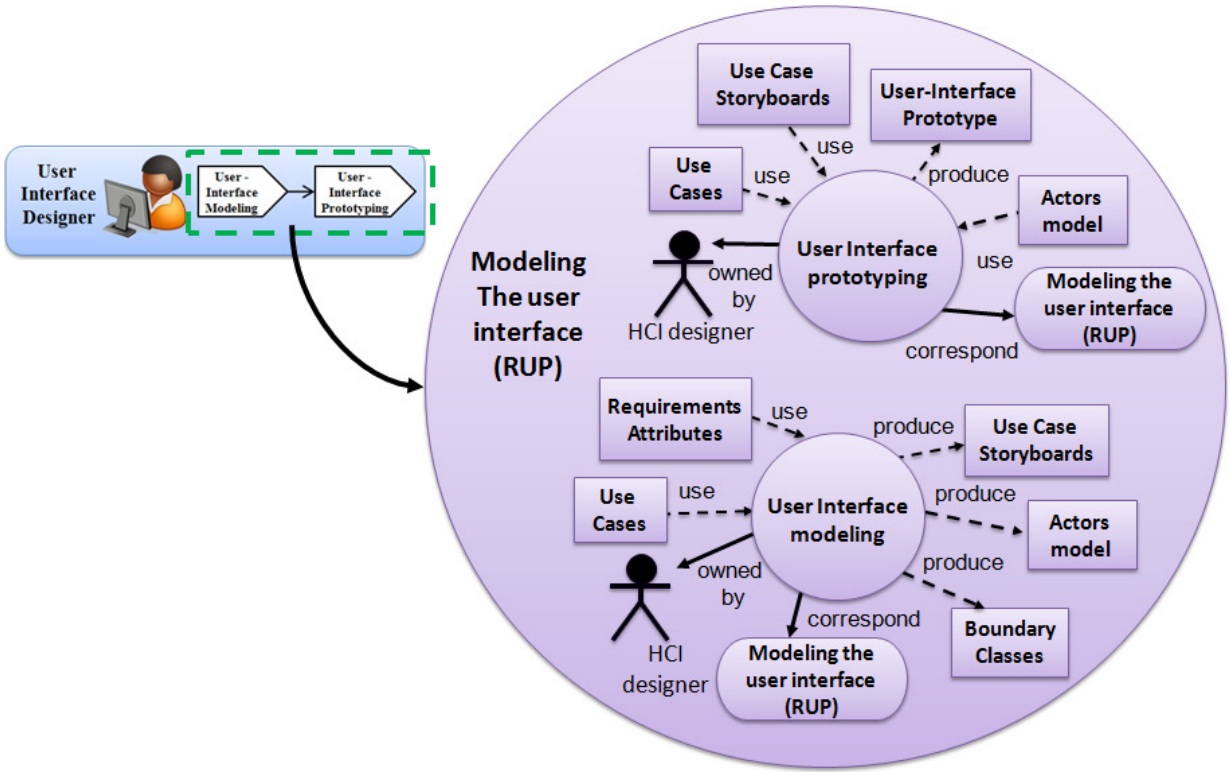
a) b)

FIGURE 72. LA FORMALISATION DU FRAGMENT DE PROCESSUS « SPECIFICATION EXTERNE DE L'INTERACTION »

5.3.2. Les fragments de la méthode RUP [Kruchten, 2000]

Le troisième exemple correspond à une partie de la phase d'élaboration de la méthode RUP initialement proposée par [Jacobson *et al.*, 1999] et étendue par [Kruchten, 2000]. L'exemple correspond à la phase d'élaboration (voir figure 61). Nous nous focalisons sur la modélisation de l'interface utilisateur.

Dans notre approche, les activités de modélisation et prototypage de l'interface de l'utilisateur correspondent à deux services organisationnels. L'un pour spécifier et décrire les interfaces et l'autre pour prototyper les interfaces. La figure 73 montre la formalisation des aspects interactionnels de la phase d'élaboration de la méthode RUP en termes de services organisationnels.



a) b)

FIGURE 73. LA FORMALISATION DU FRAGMENT DE PROCESSUS « MODELING THE USER INTERFACE »

Dans cette section, nous avons illustré par des exemples la formalisation de fragments de processus sous la forme de services organisationnels. La section suivante montre la relation entre les services organisationnels et les services intentionnels.

5.3.3. Le lien entre un service organisationnel et un service intentionnel

Cette section traite la relation entre les buts formalisés par les services intentionnels et les fragments de méthodes structurés sous la forme de services organisationnels. Comme nous l'avons souligné à partir de la description du méta-modèle intentionnel, nous considérons qu'un service intentionnel peut être réalisé par plusieurs services organisationnels.

Concernant l'enchaînement des activités de modélisation du service intentionnel "***Specify Interactional aspects***" d'un système interactif (voir figure 59), un processus qui répond à ce but est une partie de la phase de « **Spécification Interactionnelle des Besoins** » de la méthode Symphony étendue. Ce processus constitue un service organisationnel appelé "***Specification of external interaction***". Il est composé de quatre sous-services (un par activité). Le service organisationnel composite est lié au niveau intentionnel au but "***Specify Interactional aspects***" (figure 74). Toute

méthode capitalisable est donc analysée au travers des buts du niveau intentionnel de manière à identifier des fragments correspondants à des buts.

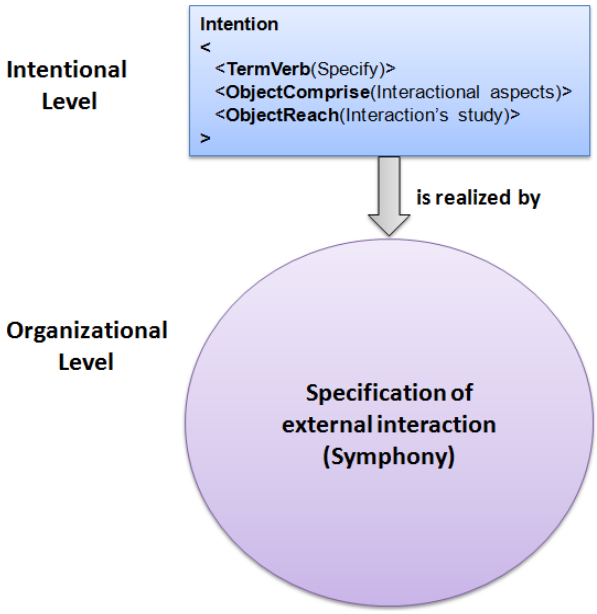


FIGURE 74. RELATION ENTRE UN SERVICE INTENTIONNEL ET UN SERVICE ORGANISATIONNEL

Il est évidement possible de lier les buts intentionnels à d'autres processus existants. Concernant notre exemple, un autre processus possible qui répond au but "***Specify Interactional aspects***" est le fragment de processus de la phase d'élaboration de la méthode RUP décrite dans la section précédente. La figure 75 montre les deux fragments de processus que nous avons analysés et qui répondent parfaitement au but "***Specify Interactional aspects***" pour un système interactif.

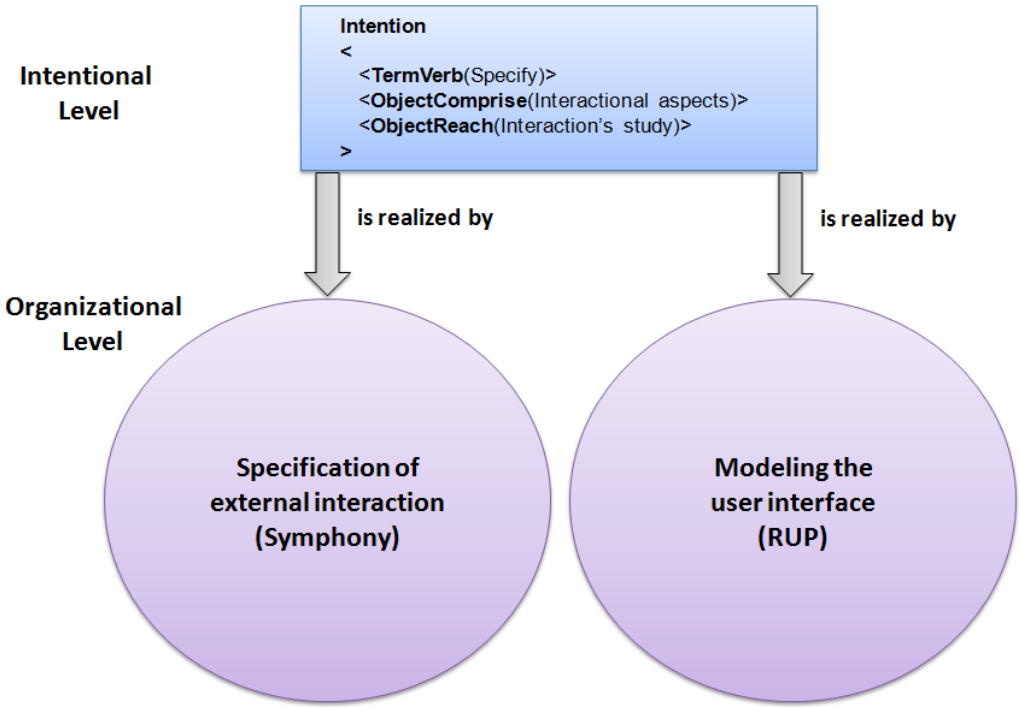


FIGURE 75. RELATION ENTRE UN SERVICE INTENTIONNEL ET PLUSIEURS SERVICES ORGANISATIONNELS

Dans cette section, nous avons montré les principes du méta-modèle de service organisationnel. Nous avons expliqué comme formaliser les fragments de processus sous la forme de services organisationnels et avec des exemples, nous avons montré la relation entre des services intentionnels et organisationnels. La section suivante présente les détails de notre méta-modèle de service opérationnel.

6. LA COUCHE OPERATIONNELLE

6.1. La modélisation d'un service opérationnel

Cette section présente le méta-modèle de la couche opérationnelle. L'objectif de ce méta-modèle est de décrire la structure statique de l'application qui soutiendra la recherche et le couplage des services qui sont requis par la gestion de modèles. La figure 76 montre le méta-modèle de service opérationnel.

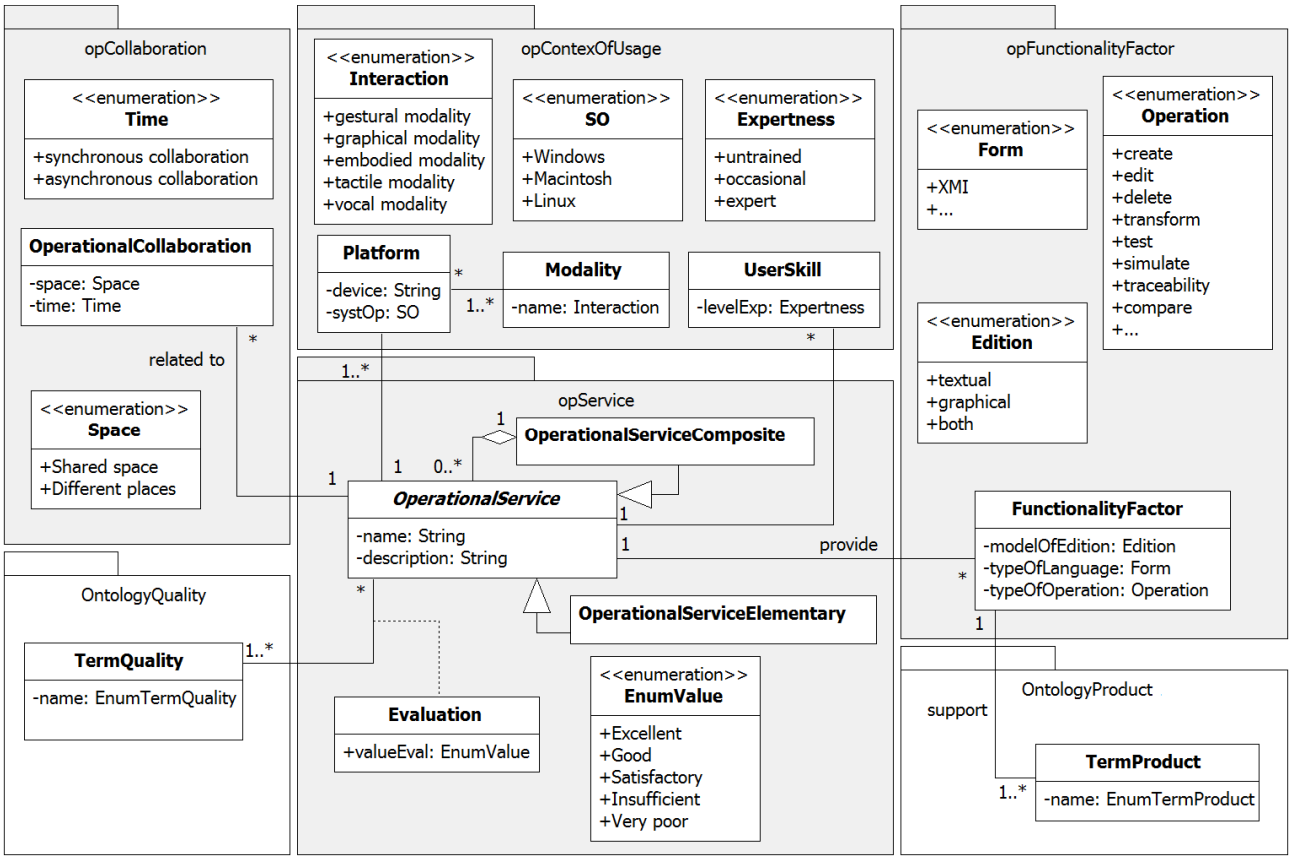


FIGURE 76. MODELE DE SERVICE OPERATIONNEL

Un **service opérationnel** (voir le paquetage « **opService** » dans la figure 76) correspond à une application exécutable composée d'autres services de gestion de modèles. Un service de gestion de modèles est un outil de gestion de modèles offert par un fournisseur. Ces services peuvent être fournis par des outils de gestion de modèles (par exemple : un AGL, un outil de modélisation ou un outil de transformation de modèles, etc.).

Les services opérationnels peuvent porter sur un ou plusieurs modèles (par exemple : le modèle des arbres de tâches, le modèle de classes, le modèle de cas d'utilisation, etc.). De la même

façon que dans les couches intentionnelle et organisationnelle, nous avons utilisé ici l'ontologie des produits « **OntologyProduct** » décrite dans la section 2.2.2.

Nous devons noter que la modélisation de services opérationnels n'est pas nouvelle. Néanmoins, dans notre travail, nous avons voulu augmenter les approches classiques en utilisant les notions telles que : la collaboration, la qualité logicielle et le contexte de l'utilisateur, pour répondre aux besoins que nous avons identifiés en étudiant la méthode Symphony augmentée. Cette étude a montré que divers acteurs (par exemple, spécialistes génie logiciel, spécialistes IHM, ergonomes) travaillent de manière coordonnée sur des modèles différents. Ces conclusions nous amènent à proposer un méta-modèle de services opérationnels associé à divers types de fonctionnalités (section 6.1.1.) et caractérisé par 1) les aspects liés à la conception collaborative (*les aspects collaboratifs du travail de conception*) (section 6.1.2.), c'est-à-dire les types de collaboration dont les concepteurs de modèles pourraient avoir besoin ; 2) le contexte d'utilisation des concepteurs de modèles et les caractéristiques de l'utilisateur (section 6.1.3.) ; 3) un certain nombre de facteurs de qualité logicielle (section 6.1.4.). Par la suite nous décrirons ces trois aspects.

6.1.1. Les fonctionnalités

Un service opérationnel fournit plusieurs fonctionnalités (voir le paquetage « **opFunctionalityFactor** » dans la figure 76). La fonctionnalité d'un service opérationnel définit la façon dont un concepteur de modèles pourra effectuer les opérations sur les modèles (par exemple, texte, graphique ou les deux), les langages supportés pour stocker l'information (par exemple XMI, etc.) et les éventuelles opérations de service (par exemple, la création, l'édition, suppression, etc.).

L'étude des différents outils de gestion de modèles (cf. état de l'art, section 1.2.) a mis en évidence l'ensemble des fonctionnalités offertes par les outils. Nous avons synthétisé les fonctionnalités des outils au moyen de cas d'utilisation. Au niveau opérationnel les fonctionnalités offertes par les outils sont représentées par la figure 77. Ces fonctionnalités sont : la gestion de modèles, la transformation de modèles, la vérification de la cohérence, la simulation et les tests.

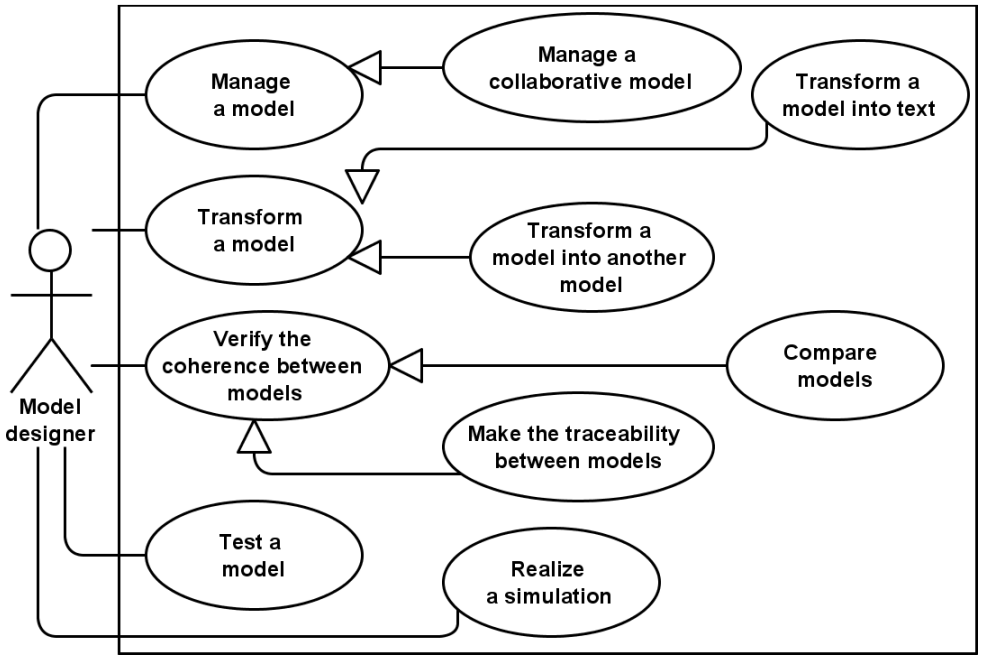


FIGURE 77. DIAGRAMME DE CAS D'UTILISATION DES FONCTIONNALITES ASSOCIEES A LA GESTION DE MODELES AU NIVEAU OPERATIONNEL

Le cas d'utilisation « **manage a model** », englobe toutes les fonctionnalités d'édition de modèles, quelque soit leur nature et leur niveau d'abstraction. En ce qui concerne ce cas d'utilisation, nous l'avons spécialisé pour indiquer la collaboration qui peut exister parmi les concepteurs de modèles, qui doivent effectuer des tâches collaboratives, comme par exemple, l'édition partagée d'un modèle.

Les autres cas d'utilisation de la figure 77 se rapportent à la manipulation des modèles, par exemple : le cas d'utilisation « **Transformer un modèle** » consiste à faire la transformation d'un modèle dans du code ou dans d'autre langage, cette transformation peut être réutilisée par un autre concepteur dans le processus de développement.

6.1.2. Les types de collaboration

Pour supporter les besoins de création d'environnements collaboratifs, un service opérationnel est lié à plusieurs formes de collaboration. La figure 78 montre l'extrait concernant les aspects liés à la collaboration. La collaboration est décrite ici à un niveau opérationnel tel que la centralisation des tâches, l'exécution synchrone, etc.

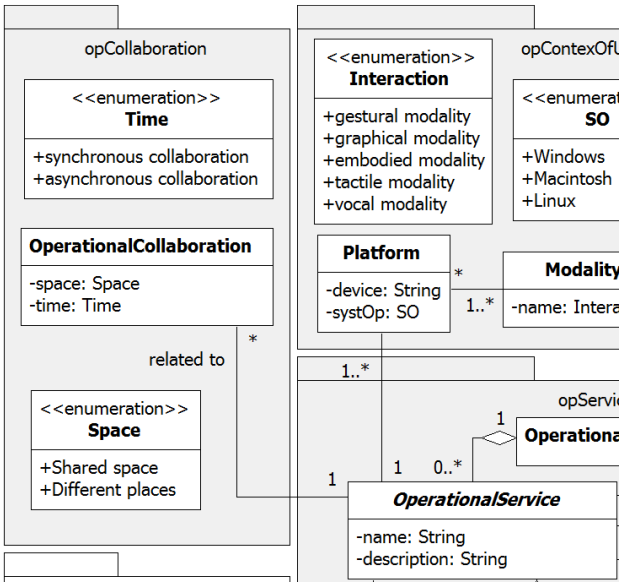


FIGURE 78. EXTRAIT DU META-MODELE DE SERVICE OPERATIONNEL REPRESENTANT LES TYPES DE COLLABORATION

Une **forme de collaboration** est caractérisée suivant le modèle de Denver de Salvador [Salvador *et al.*, 1996] et sur la classification Espace-Temps proposée par Ellis [Ellis *et al.*, 1991]. Le modèle de Denver établit que les utilisateurs peuvent être proches ou éloignés et interagir de façon **synchrone** (temps réel) ou **asynchrone.**

La classification Espace-Temps proposée par Ellis repose sur deux caractéristiques, à savoir **où et quand une action est exécutée** par un des utilisateurs par rapport aux autres utilisateurs. Donc, dans le contexte de notre travail, une collaboration est définie en fonction de deux axes : l'espace (« space »), et le temps (« time »). La figure 79 résume les deux axes considérés.

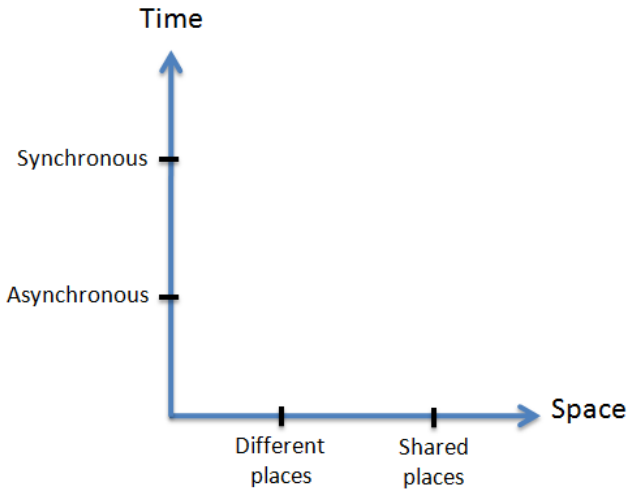


FIGURE 79. CARACTERISATION DE LA COLLABORATION

L'espace prend en compte le fait que les tâches sont centralisées ou distribuées. Un service de modélisation qui supporte une tâche distribuée offre une fonctionnalité de modélisation permettant de travailler à plusieurs concepteurs sur un même modèle.

Les tâches peuvent être exécutées dans des espaces partagés ou privés. Finalement, **le temps** est le moment de la collaboration. Il conduit à distinguer le travail synchrone (même moment de la collaboration réalisée par les différents utilisateurs) du travail asynchrone (moments différents) permettant à chacun de travailler quand il en a la possibilité et/ou la nécessité.

6.1.3. Le contexte d'utilisation

En ce qui concerne le **profil de l'utilisateur** et le **contexte d'utilisation** d'un service opérationnel, la figure 80 considère qu'un service opérationnel peut être exécuté dans différents contextes d'utilisation par des spécialistes qui ont divers niveaux d'expertise. Le contexte d'utilisation est modélisé par le profil de l'utilisateur, la plateforme et la modalité d'interaction. Notons que les « compétences » de l'utilisateur ne relèvent pas de ce niveau (indépendantes de l'outil). C'est au niveau organisationnel que nous exprimons à qui est dédié un service organisationnel (cf. figure 69).

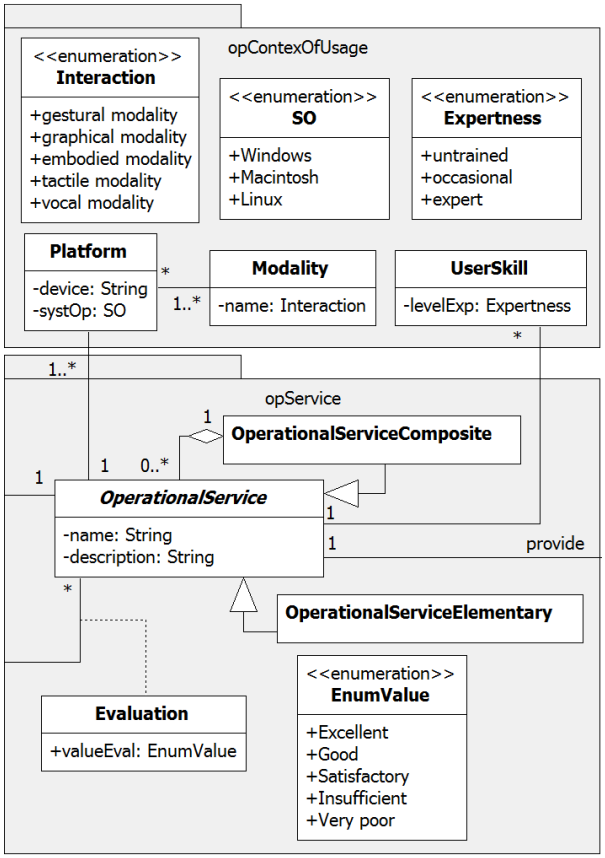


FIGURE 80. EXTRAIT DU META-MODELE DE SERVICE OPERATIONNEL REPRESENTANT LE PROFIL DE L'UTILISATEUR ET LE CONTEXTE D'UTILISATION

Un service opérationnel est caractérisé par **le niveau d'expertise** nécessaire à l'utilisateur. Nous considérons, dans nos travaux, les niveaux d'expertise suivantes : expert, occasionnel et novice.

Le **contexte d'utilisation** est un espace d'information structuré qui inclut les plateformes logicielle et matérielle exigées par le service. Il est relié à une variété de modalités d'interaction. Nous considérons ici, par exemple, la possibilité d'avoir des services de gestion de modèles qui offrent des modalités graphiques, gestuelles, tactiles, vocales, etc.

Nous prendrons comme exemple l'environnement de modélisation ConcurTaskTress (CTTE), un outil basé sur Java réalisé par le laboratoire HIID-ISTI/CNR de Pise (Italie) [Mori *et al.*, 2002]. Cet outil offre une interaction graphique.

6.1.4. La qualité du logiciel

Pour garantir la qualité dans l'utilisation d'un outil de gestion de modèles, nous avons inclus dans notre approche la notion de qualité logicielle. Un fournisseur d'outils souhaitant intégrer un outil en tant que service opérationnel est donc tenu d'en mesurer la qualité. La qualité logicielle est une appréciation globale d'un logiciel basée sur de nombreux indicateurs. Les définitions de qualité logicielle sont données en termes de « conformité aux besoins ». Nous avons intégré dans notre méta-modèle opérationnel l'ontologie des facteurs de qualité « OntologyQuality ».

Les concepts et termes de qualité que nous avons repris sont issues des critères de qualité de FURPS [Grady, 1992], McCall [McCall et *al.*, 1977], Boehm [Boehm et *al.*, 1978], IEEE 1061[IEEE std 1061, 1992], Dromey [Dromey, 1995], ISO 9126 [ISO std 9126, 1991]. La figure 81 montre l'extrait du méta-modèle de service opérationnel qui représente l'aspect de la qualité du logiciel. La figure ci-dessous montre qu'un service opérationnel est catégorisé par au moins un terme de qualité.

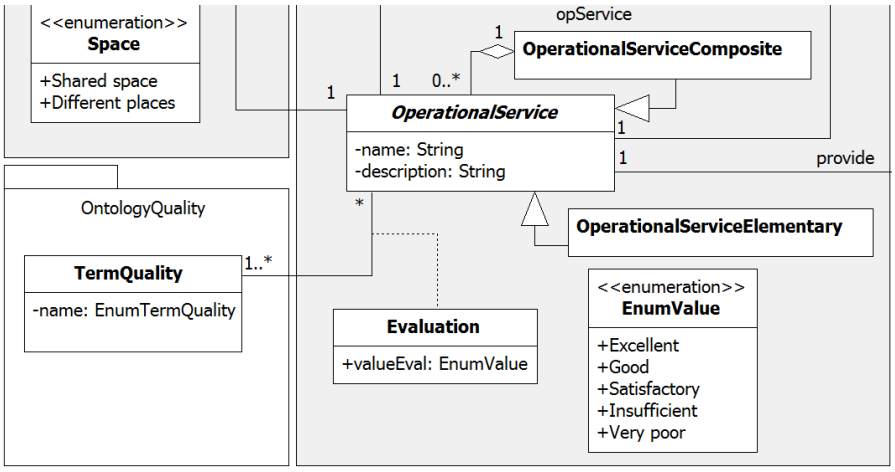


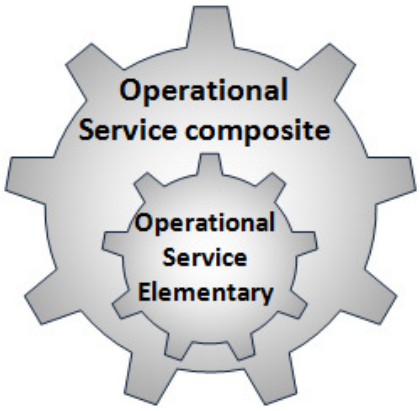
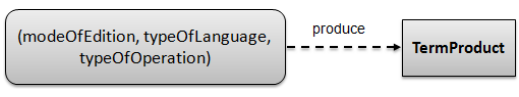
FIGURE 81. EXTRAIT DU META-MODELE DE SERVICE OPERATIONNEL REPRESENTANT L'ASPECT DE LA QUALITE DU LOGICIEL

Pour chaque outil, un critère de qualité est apprécié à l'aide d'une évaluation. Cette évaluation peut être exprimée comme une valeur d'excellence. **La valeur d'évaluation** est évaluée à l'aide d'une classe énumération appelé « **EnumValue** ».

Par la suite, nous présenterons un formalisme de représentation graphique pour les services opérationnels.

6.2. Le formalisme d'expression des services opérationnels

Nous proposons dans le tableau 26 un formalisme d'expression des services opérationnels afin de représenter les objets instances du méta-modèle de services opérationnels qui représentent les outils de gestion de modèles.

Element	Formalism
Opérationnel Service	
Composition of operational services	
TermProduct (Model)	
Functionality Factor	Produced model 

	(modeOfEdition, typeOfLanguage, typeOfOperation)	Used model TermProduct use (modeOfEdition, typeOfLanguage, typeOfOperation)
Operational Collaboration	(space, time)	
Modality	name	
Platform	(device, systOp)	(device, systOp) → name
User Skill	levelExp	
TermQuality (Quality factor)	name	
Evaluation	valueEval	
Association Operational service – elements of meta-model (except TermProduct)	Operational Service — name association → Element of meta-model	

TABLEAU 26. FORMALISME D'EXPRESSION DES SERVICES OPERATIONNELS

6.3. Exemples

6.3.1. La représentation de services opérationnels

Cette section présente des exemples pour illustrer le méta-modèle du service opérationnel. Nos exemples reprennent les descriptions des outils présentés dans le tableau suivant qui correspondent à des outils utilisés par les spécialistes de l'IHM pour réaliser la modélisation des arbres de tâches.

Outil 1 : Concurrent Task Trees Environment (CTTE)	
Description générale de l'outil	L'environnement de modélisation ConcurTaskTress (CTTE) est un outil basé sur Java réalisé par le laboratoire HIID-ISTI/CNR de Pise (Italie) [Mori *et al.*, 2002] qui peut être utilisé pour supporter la conception d'applications interactives mono-utilisateur et coopératives. Cet outil offre un grand nombre de fonctionnalités pour soutenir le développement rapide, l'analyse et la simulation des modèles de tâches. Il est basé sur la notation ConcurTask (CTT) [Paternò, 1997] qui permet aux concepteurs de représenter la structure du modèle de tâches de façon hiérarchique. Cet outil est basé sur un environnement de modélisation graphique et il est indépendant du système d'exploitation utilisé pour le concepteur. Les modèles de tâches générés avec cet outil peuvent être stockés sous le forme d'un fichier XML conforme aux spécifications CTT.
Outil 2 : Multimodal TERESA	
Description générale de l'outil	Multimodal TERESA [Paternò *et al*, 2003] est un environnement de conception d'interfaces utilisateur multimodales. Il est composé d'un ensemble de langages basés sur XML, de transformations entre ces langages et d'un outil d'édition d'interfaces d'utilisateur sous la forme d'arbres de tâches. Le but de TERESA est de fournir un environnement pour développer des interfaces flexibles et multimodales pour un large nombre de plateformes qui soutiennent diverses modalités. Cet environnement prévoit, donc, la composition d'interactions graphiques et vocales. TERESA vise à faciliter les activités des concepteurs, l'un des aspects clés de cet outil est d'offrir une grande souplesse pour la construction d'applications interactives, en fournissant les outils avec différents niveaux d'automatisation. En effet, l'outil est capable de fournir des solutions différentes, allant des solutions complètement automatiques (adaptées pour les concepteurs débutants ou plus experts). En outre, les interfaces utilisateurs générées par l'outil peuvent également être précisées à différents niveaux d'abstraction. Il s'agit d'offrir des vues "différentes" de la même interface utilisateur.
Outil 3 : A graphical task modeling tool (Tamot)	
Description générale de l'outil	Tamot [Lu et al., 2002] est un outil basé sur Java pour la construction de modèles de tâches. Cet outil est basé sur le formalisme Diane+ [Tarby et al., 1996]. Cet outil adresse aux concepteurs qui souhaitent réaliser l'édition et l'analyse de la tâche de l'utilisateur. Les caractéristiques de qualité de cet outil sont : la facilité d'apprentissage, le support pour la création de multiples tâches, le support pour faciliter plusieurs styles (bottom-up et top-down) de modélisation des tâches. Il offre des facilités pour réaliser les modifications, la génération de rapports personnalisables, la capacité de visualiser de multiples perspectives d'un modèle de tâches, l'interopérabilité.
Outil 4 : Kernel of Model for Activity Description environment (K-MADe)	
Description générale de l'outil	K-MADe [Baron et al., 2006] est un outil développé en Java qui permet la conception d'arbres des tâches et facilite la spécification d'un ensemble de variables ayant trait à l'ergonomie. Le but principal de cet outil est de contribuer à la prise en compte de l'ergonomie lors du processus de conception des systèmes interactifs, par l'analyse de l'activité et des tâches. Derrière la spécification de K-MADe il y a beaucoup d'informations destinées à la documentation et à la simulation. Cet outil est destiné aux personnes souhaitant décrire, analyser, et formaliser les activités d'opérateurs humains, d'utilisateurs, dans des environnements informatisés ou non, en situation réelle ou simulée ; sur le terrain, ou en laboratoire. Bien que toutes sortes de profils de personnes soient possibles, cet environnement est plus particulièrement destiné aux ergonomes et spécialistes en IHM.

TABLEAU 27. EXEMPLES D'OUTILS

La figure 82 montre la représentation graphique de l'outil CTTE. Notons que cette représentation correspond au service opérationnel qui offre les fonctionnalités requises pour la gestion des arbres de tâches.

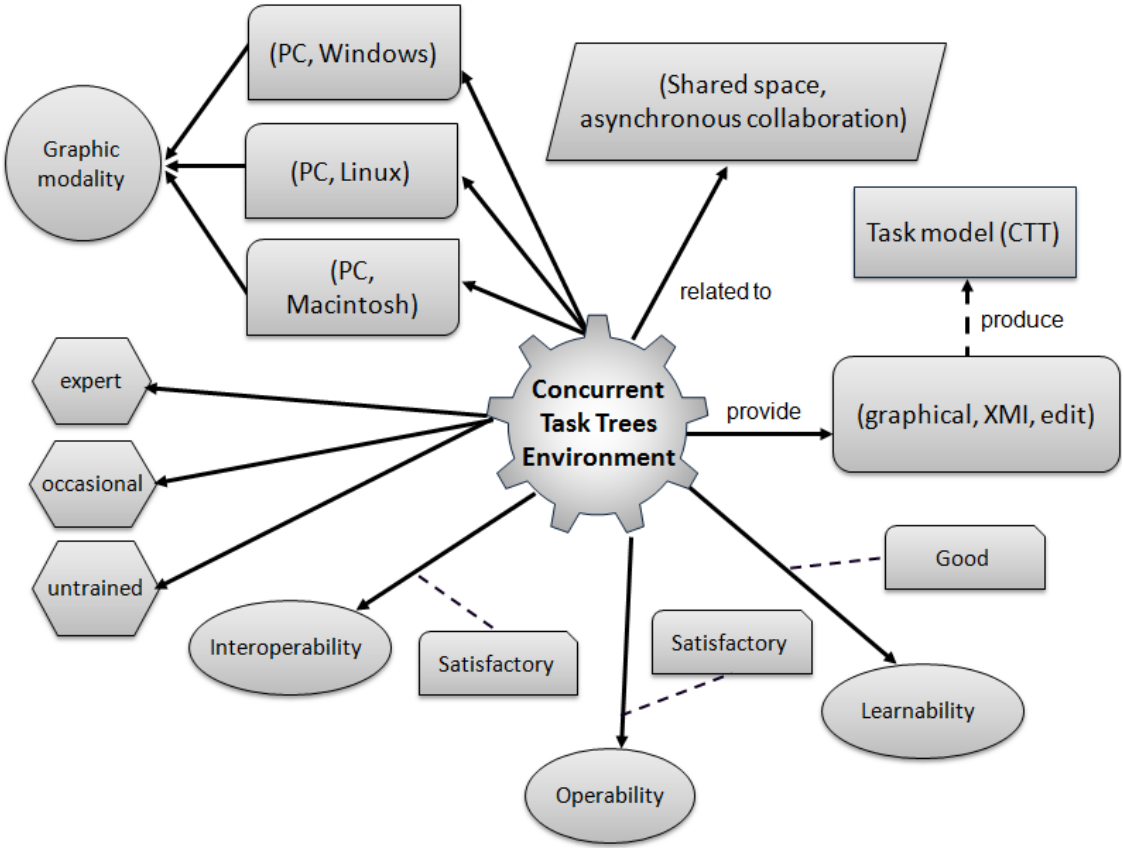


FIGURE 82. REPRESENTATION DU SERVICE OPERATIONNEL « CONCURRENT TASK TREES ENVIRONMENT »

La figure 83 montre la représentation graphique de l'outil K-MADe. Notons que cette représentation correspond au service opérationnel qui offre les fonctionnalités requises pour la gestion des arbres de tâches.

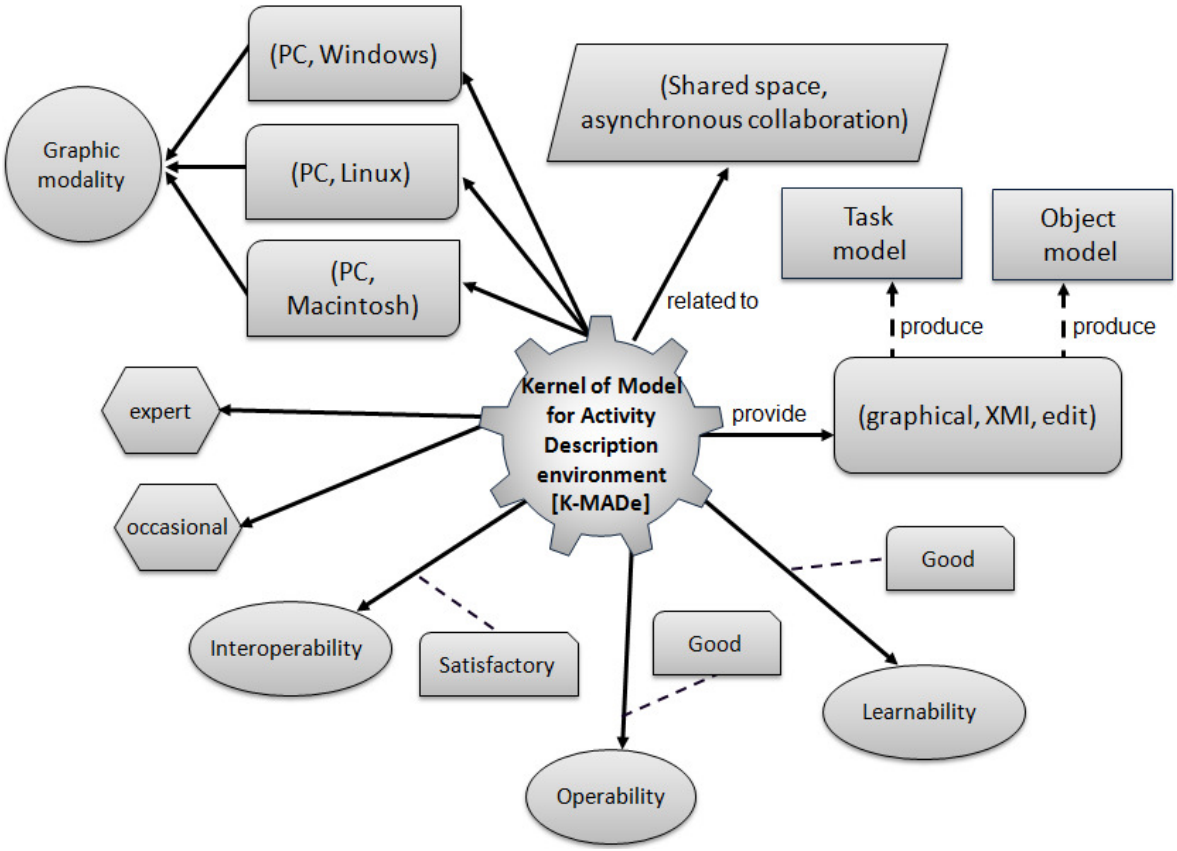


FIGURE 83. REPRESENTATION DU SERVICE OPERATIONNEL « K-MADE »

6.3.2. Le lien entre un service opérationnel et un service organisationnel

Cette section présente un exemple pour illustrer la relation entre un service organisationnel et un service opérationnel. Comme nous l'avons souligné précédemment (section 5.1.), un service organisationnel peut être réalisé par plusieurs services opérationnels. Pour réaliser notre exemple, nous reprendrons les fragments de méthode de la phase de « **Spécification Interactionnelle des Besoins** » de la méthode Symphony étendue. Ces fragments sont formalisés sous la forme de services organisationnels dans la section 5.3.1. (voir figures 71 et 72).

Au niveau opérationnel, l'objectif du spécialiste IHM et de l'ergonome est de choisir un éditeur de modèles offrant une haute conformité avec les arbres de tâches, spécialement les modèles CTT [Paternò, 1997], un langage basé sur des représentations graphiques et utilisé par l'environnement de modélisation CTTE basé sur Java.

La figure 84 montre la relation entre le service organisationnel « D**escription of task model** » et le service opérationnel « **Concurrent Task Trees Environment** ». Notons que dans cette association, l'outil CTTE formalisé sous la forme d'un service opérationnel répond aux besoins du spécialiste IHM en tant que support du service organisationnel « description du modèle de tâches ».

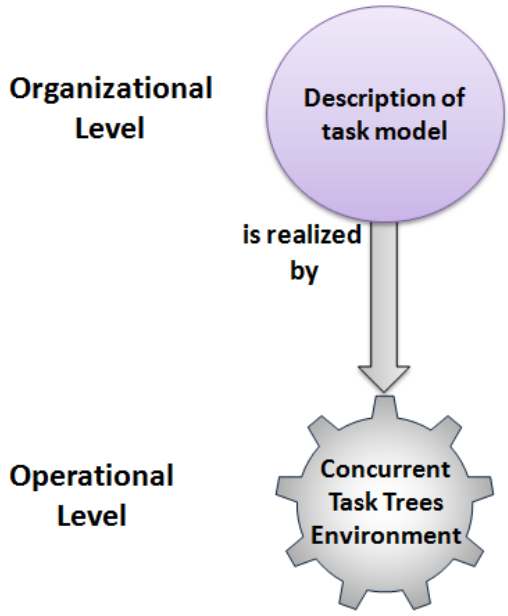


FIGURE 84. RELATION ENTRE UN SERVICE ORGANISATIONNEL ET UN SERVICE OPERATIONNEL

Il est évidement possible de lier les services organisationnels (donc, les fragments de méthodes) à d'autres outils existants. Ces outils doivent être formalisés sous la forme de services opérationnels. Concernant notre exemple, le service organisationnel « **Description of task model** » peut être réalisé par l'usage des outils : **CTTE, TERESA, Tamot, K-MADe,** ou d'autres (cf. figure 85). Notons que par simplification nous présentons les outils sans détailler touts les éléments utilisés pour leur représentation graphique.

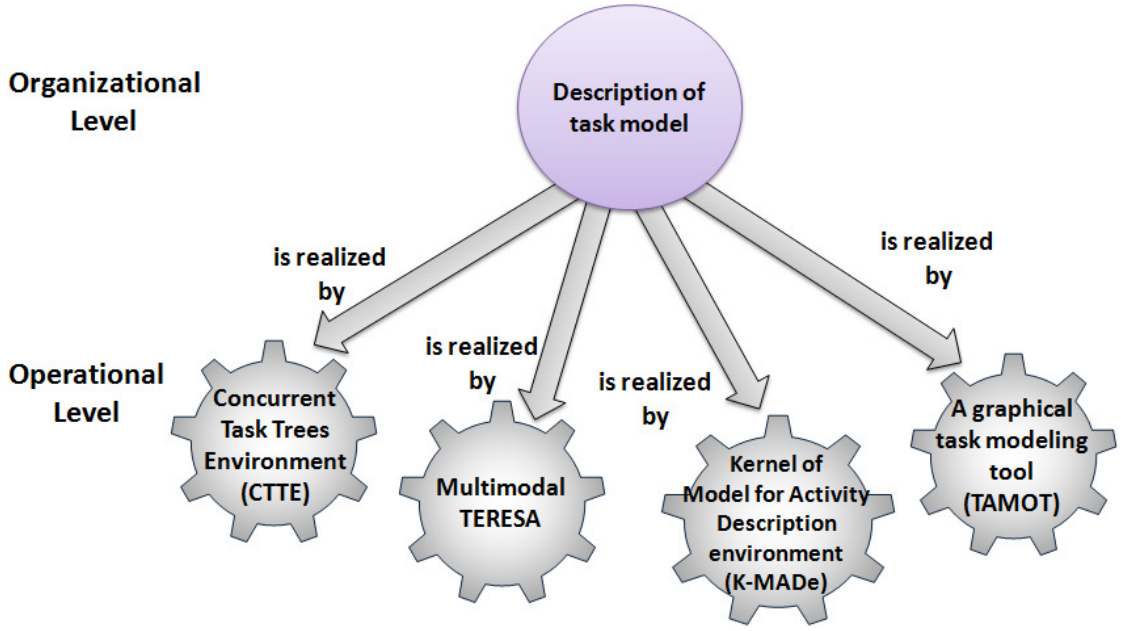


FIGURE 85. RELATION ENTRE UN SERVICE ORGANISATIONNEL ET PLUSIEURS SERVICES OPERATIONNELS

Dans cette section nous avons présenté les principes d'un service opérationnel pour faciliter la création d'environnements de modélisation. Nous avons détaillé le méta-modèle des services opérationnels, puis un formalisme pour la représentation graphique des services intentionnels.

Nous avons montré comment un service organisationnel peut être réalisé par un ou plusieurs services opérationnels, en fonction des compétences spécifiques et des besoins opérationnels des concepteurs de modèles.

7. SYNTHESE

La modélisation de services permet de structurer l'ensemble des connaissances nécessaires pour la description des besoins des concepteurs de modèles, des méthodes adaptées à ces besoins et des outils requis pour assister ces méthodes pour la conception.

Nous avons présenté dans ce chapitre un ensemble des concepts et des principes de notre approche orientée services pour la Réutilisation de Processus et d'Outils de Modélisation qui proposent de faciliter le choix des processus et des environnements de modélisation adaptables aux besoins de concepteurs de modèles. Nos propositions portent sur trois niveaux de services : le niveau intentionnel, le niveau organisationnel et le niveau opérationnel. Les trois niveaux offrent des services de gestion de modèles de natures différentes. Le niveau opérationnel permet de définir l'environnement de modélisation pour les concepteurs de modèles. Le niveau organisationnel garantit la réutilisation des services opérationnels d'une manière coordonnée, mais également, il garantit la création et la gestion des fragments de méthode. Dans cet objectif, le niveau intentionnel permet de définir les buts stratégiques des concepteurs de modèles, qui peuvent être mis en œuvre par des démarches de conception décrites au niveau organisationnel.

Nous avons présenté le méta-modèle de chaque niveau de modélisation, des exemples étaient présentés utilisant des représentations graphiques.

Nous avons montré comment un service intentionnel est réalisé par plusieurs services organisationnels, lui-même est réalisé par plusieurs services opérationnels. Le chapitre suivant, présente une vision globale de l'utilisation de la plateforme à base de services pour la gestion de modèles.

CHAPITRE 4

La vision globale d'utilisation de la démarche

CHAPITRE 4

INTRODUCTION

Le chapitre précédent a présenté notre approche à base de services pour la réutilisation de processus et d'outils de modélisation. Cette approche, structurée en trois niveaux d'abstraction, est soutenue par un outil qui la met en œuvre et la rend utilisable. Ce chapitre est consacré à la présentation de la vision globale de l'utilisation et l'architecture de la plateforme à base de services pour la réutilisation de processus et des outils de modélisation. Il s'agit d'introduire les différents objectifs qui ont guidé la définition de l'approche ainsi que les éléments qui la composent.

Dans un premier temps, nous présentons les fonctionnalités de la plateforme de modélisation. Puis, nous présentons un aperçu de la plateforme en cours d'élaboration qui permettra la gestion de nos trois niveaux de service. Ensuite, nous présentons la vision globale d'utilisation de la plateforme.

1. LA PLATEFORME POUR LA REUTILISATION DE PROCESSUS ET D'OUTILS DE MODELISATION

1.1. Les fonctionnalités de la plateforme

La figure 86 représente les grandes fonctionnalités que nous souhaitons offrir aux fournisseurs de services (sous forme d'un arbre de tâches) qui souhaiteraient utiliser l'approche présentée précédemment. Ces fonctionnalités concernent la création des services dans les trois niveaux d'abstraction.

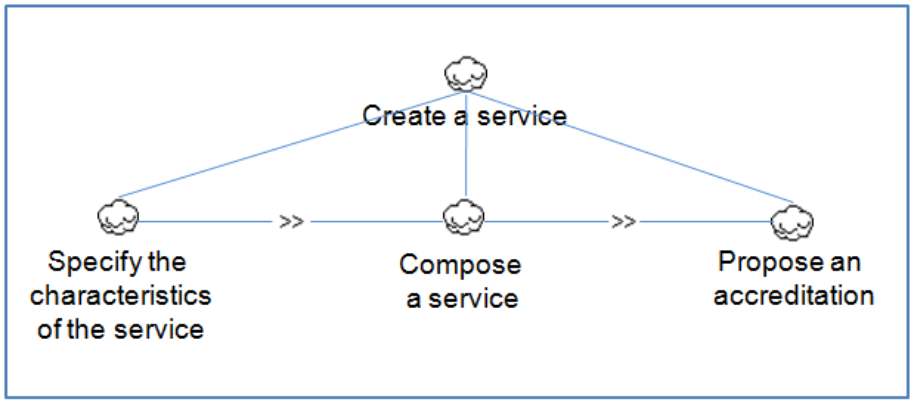


FIGURE 86. FONCTIONNALITES SOUHAITEES POUR LES FOURNISSEURS DE LA PLATEFORME

Le fournisseur peut donc ajouter de nouveaux services appropriés à son poste (par exemple, un fournisseur d'environnements de modélisation fournit un service opérationnel). Il peut réaliser des compositions des services du même niveau (par exemple : un fournisseur d'environnement de modélisation peut fournir un service opérationnel composé de plusieurs éditeurs).

Les fournisseurs peuvent faire des liens entre les services. Il s'agit juste de lier d'autres services du niveau inférieur ou supérieur (peu importe le type) à ce même service. Le lien entre les services doit être autorisé par le fournisseur du service vers lequel le lien est proposé. Par exemple, lorsqu'un fournisseur d'environnements crée un service intentionnel, il doit établir les liens avec les méthodes existantes de la couche organisationnelle. Pour ce faire, le fournisseur propose une accréditation du service intentionnel (l'intention) à tous les ingénieurs de méthode qui offrent les méthodes sous la forme de services organisationnelles. Les ingénieurs de méthodes (les fournisseurs) doivent valider les liens proposés.

La figure 87 représente les fonctionnalités que nous souhaitons offrir aux clients de services sous forme d'un diagramme d'arbres de tâches.

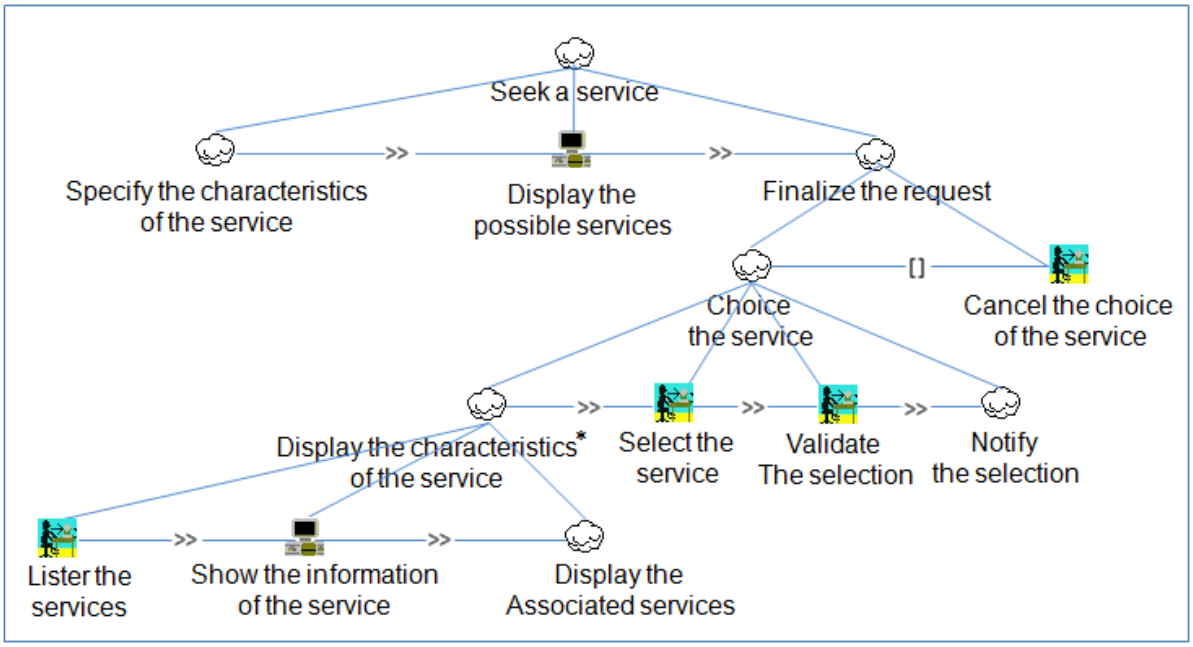


FIGURE 87. FONCTIONNALITES SOUHAITEES POUR LES CLIENTS DE LA PLATEFORME

Le client commence par spécifier les caractéristiques du service qu'il cherche. Puis le système lui affiche les différents services possibles. Enfin la demande est finalisée, soit par le choix d'un service, soit par la suppression du choix. Le choix et validation d'un service permettra définir l'environnement de modélisation requis pour le client.

1.2. L'aperçu de la plateforme

Actuellement nous implémentons un prototype expérimental, en utilisant les modèles de service décrits dans le Chapitre 3. Divers solutions techniques étaient envisageables. Nous aurions pu en particulier nous baser sur les technologies web dont OWL-S. Mais nous avons voulu conserver trois niveaux de services distincts et ne pas nous limiter aux environnements web. Aussi notre solution est basée sur une plateforme de gestion de services « générique ». Nos services pourront ensuite évoquer des services exécutables dans différentes technologies dont OWL-S (voir la figure 88). Le lien entre les services de modélisation et les services exécutables s'effectue en particulier via les paramètres d'entrée et de sortie des services exécutables.

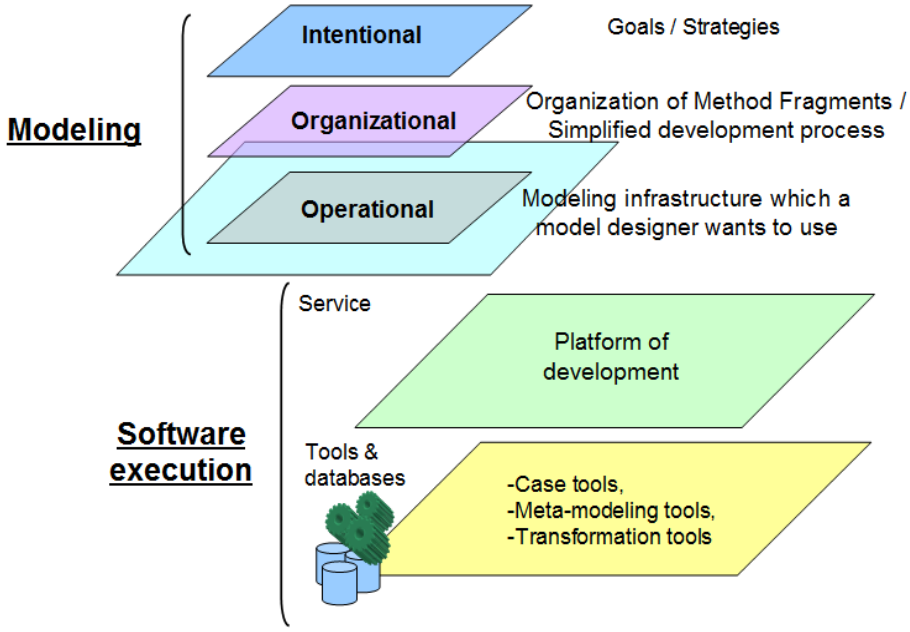


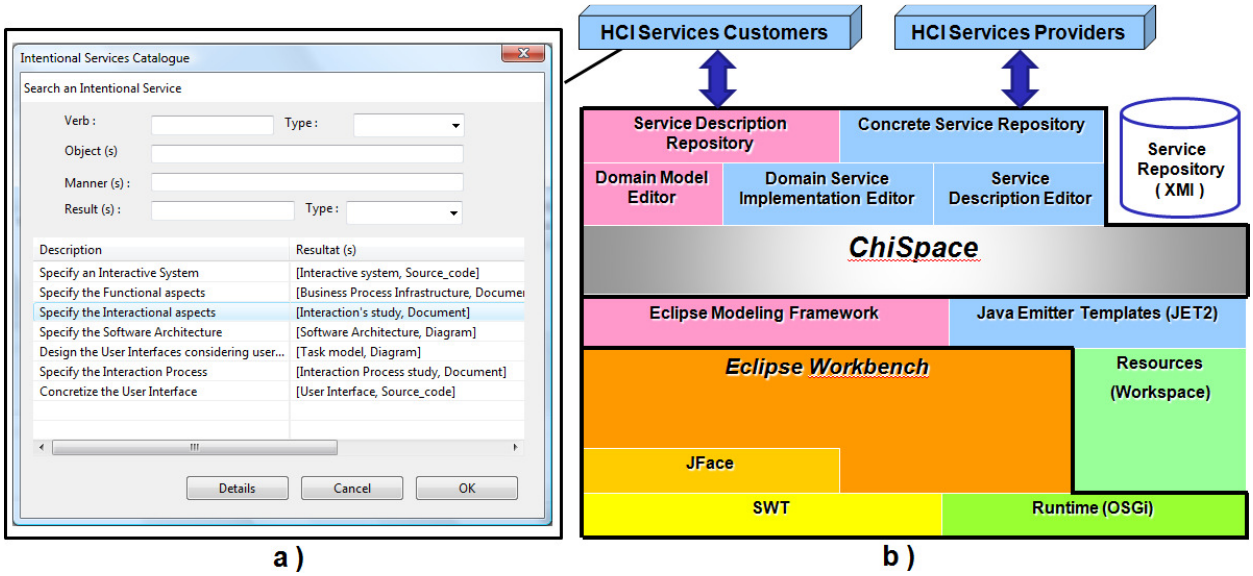
FIGURE 88. PLATEFORME DE GESTION DE SERVICES « GENERIQUE »

Le prototype doit permettre d'effectuer les fonctionnalités que nous avons évoquées dans la section précédente. A ce jour, il est constitué de deux blocs indépendants mais complémentaires.

Le premier bloc (figure 89 partie b) considère la mise en œuvre d'un registre de services avec l'atelier de gestion de composition de services « ChiSpace » [Yu *et al.*, 2008]. L'objectif de cet environnement est de simplifier le travail des développeurs lors de la réalisation d'applications par composition de services. ChiSpace a été implémenté sur « Eclipse Modeling Framework » (Eclipse-EMF) et la technologie JET2 d'Eclipse. L'environnement est composé d'un ensemble d'éditeurs basés sur une perspective personnalisée d'Eclipse [Yu *et al.*, 2008].

Le registre de services correspond à la base de données où sont stockées les descriptions des trois niveaux de services. Ces descriptions sont basées sur les trois modèles de services présentés dans le chapitre précédent.

Le deuxième bloc permet d'ajouter, d'afficher, de sélectionner et de valider les services qui sont stockés dans le registre de services. Il est basé sur « Eclipse Rich Platform » (Eclipse-RCP). Ainsi, Eclipse-RCP est utilisé pour développer les interfaces qui permettent d'utiliser les fonctionnalités de la plateforme pour les clients et les fournisseurs. La figure 89 (partie a) illustre l'interface de recherche des services intentionnels.



a) b)

FIGURE 89. PLATEFORME ORIENTEE SERVICE

La première version de la plateforme présentée ici se focalise sur l'ajout, la composition et la recherche des services intentionnels. Nous considérons les autres fonctionnalités énoncées (par exemple : l'accréditation de services, la sélection et la validation) pour une prochaine version. Les interfaces créées sont basées sur une version antérieure du méta-modèle de services, présenté dans [Pérez-Medina *et al.*, 2009].

1.3. Réalisation de l'ajout, la composition et la recherche d'un service intentionnel

La plateforme réalisée permet (dans la première version) de créer, composer et rechercher des services intentionnels.

L'implémentation a été réalisée par deux stagiaires de DUT informatique de l'IUT2 de Grenoble [De matos, 2009], [Marín, 2009] sur la base des fonctionnalités prévues et présentées dans ce chapitre.

1.3.1. L'ajout d'un service intentionnel

La figure 90 présente l'interface homme-machine de création de services intentionnels. Cette interface permet la spécification des éléments contenus dans un service intentionnel. Il est possible de réaliser la composition de services intentionnels et d'établir les liens avec les services organisationnels. Les boutons « Catalog » à droite de la fenêtre permettent d'accéder à des fenêtres de recherche d'un service existant dans la liste de services correspondant à son type (cf. intentionnel ou organisationnel).

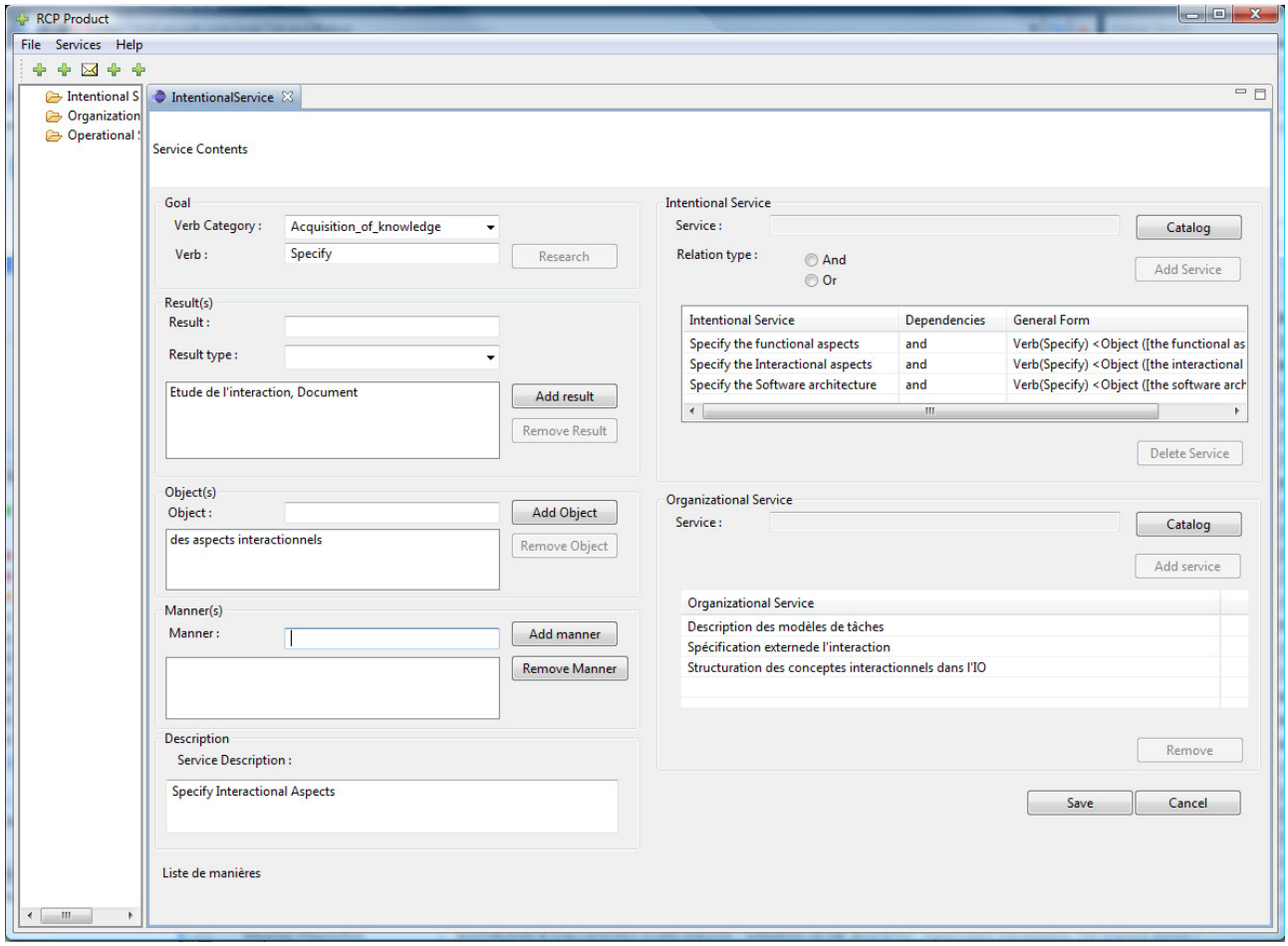


FIGURE 90. INTERFACE D'AJOUT DES SERVICES INTENTIONNELS

1.3.2. La recherche d'un service intentionnel

La figure 91 présente les fonctionnalités de l'outil pour chercher des services intentionnels. La recherche s'effectue suivant les caractéristiques d'un service intentionnel c'est-à-dire suivant les éléments spécifiés dans le méta-modèle tels que le verbe ou l'objet. L'autre particularité de la recherche pour les services intentionnels est l'affichage des services organisationnels qui leur sont liés.

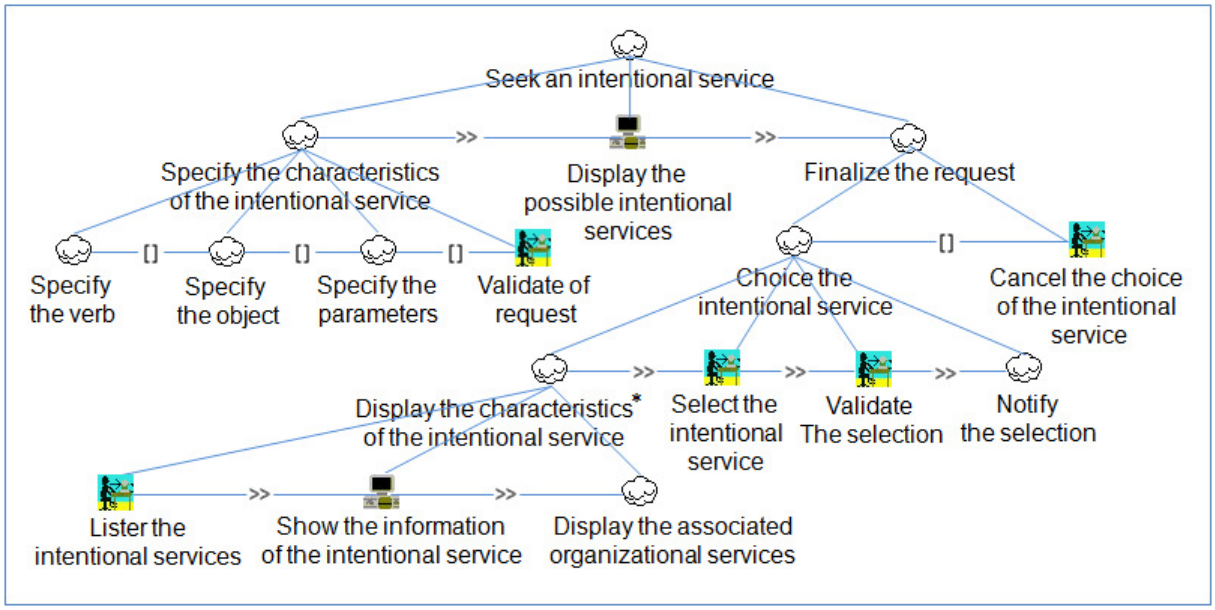


FIGURE 91. RECHERCHE DES SERVICES INTENTIONNELS

Etant donné que chaque type de service est composé d'attributs très différents les uns des autres, il n'était pas possible de concevoir un catalogue général permettant une recherche avancée des services de tout type. Les services d'un même type ont donc un catalogue propre. La figure 92 présente le catalogue des services intentionnels.

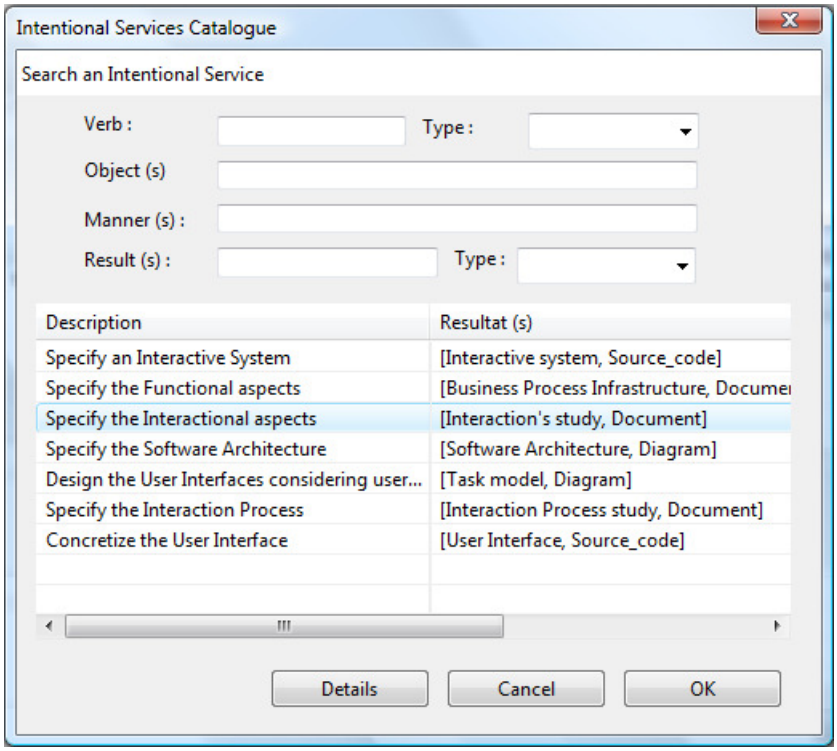


FIGURE 92. INTERFACE DE RECHERCHE D'UN SERVICE INTENTIONNEL

2. LA VISION GLOBALE D'UTILISATION DE LA PLATEFORME

Actuellement, nous ne disposons pas d'un processus d'assistance formalisé d'aide aux clients et fournisseurs de la plateforme pour spécifier, composer et valider les services avant d'être intégrés dans la plateforme. Nos recherches se limitent à offrir une plateforme à base de services pour la construction des environnements de modélisation. Cependant, cette section montre **une vision globale d'utilisation de la plateforme qui est à la fois ascendante et descendante**.

Comme nous l'avons vu lors de la description des modèles de services, les services intentionnels sont liés aux services organisationnels, eux-mêmes liés aux services opérationnels. Par exemple, le spécialiste IHM a besoin d'étudier l'interaction du système. Pour ce faire, il commence par rechercher un service intentionnel qui puisse répondre à ses besoins (voir la figure 93). Une fois le service trouvé, le spécialiste IHM choisit un service organisationnel associé au service intentionnel trouvé. Par rapport à l'activité de spécification externe de l'interaction, le spécialiste IHM, responsable de l'exécution du service organisationnel, a des compétences spécifiques et des besoins. Cette activité qui est représentée sous forme de service organisationnel dans notre approche, est liée à différents services opérationnels, correspondant à des outils pour aider la réalisation de la modélisation des tâches. Ces outils sont des éditeurs de modèles de tâches tels que CTTE, TESERA, K-MADe.

Pour choisir parmi eux, les caractéristiques des services opérationnels sont utilisées. Ici le spécialiste IHM a généralement un Mac, est habitué à travailler avec des modèles graphiques et collabore avec d'autres concepteurs de modèles (ergonomes et spécialistes GL). Il a besoin d'un environnement qui supporte la collaboration asynchrone et permet la réalisation de la tâche de manière distribuée. Il est donc important que 1) l'outil tourne sur plusieurs plateformes dont Mac ; 2) présente les modèles sous forme graphique ; 3) permette la collaboration asynchrone et distribuée. En considérant ces besoins, l'outil CTTE sera sélectionné. La figure 90 représente cette recherche et ces enchaînements.

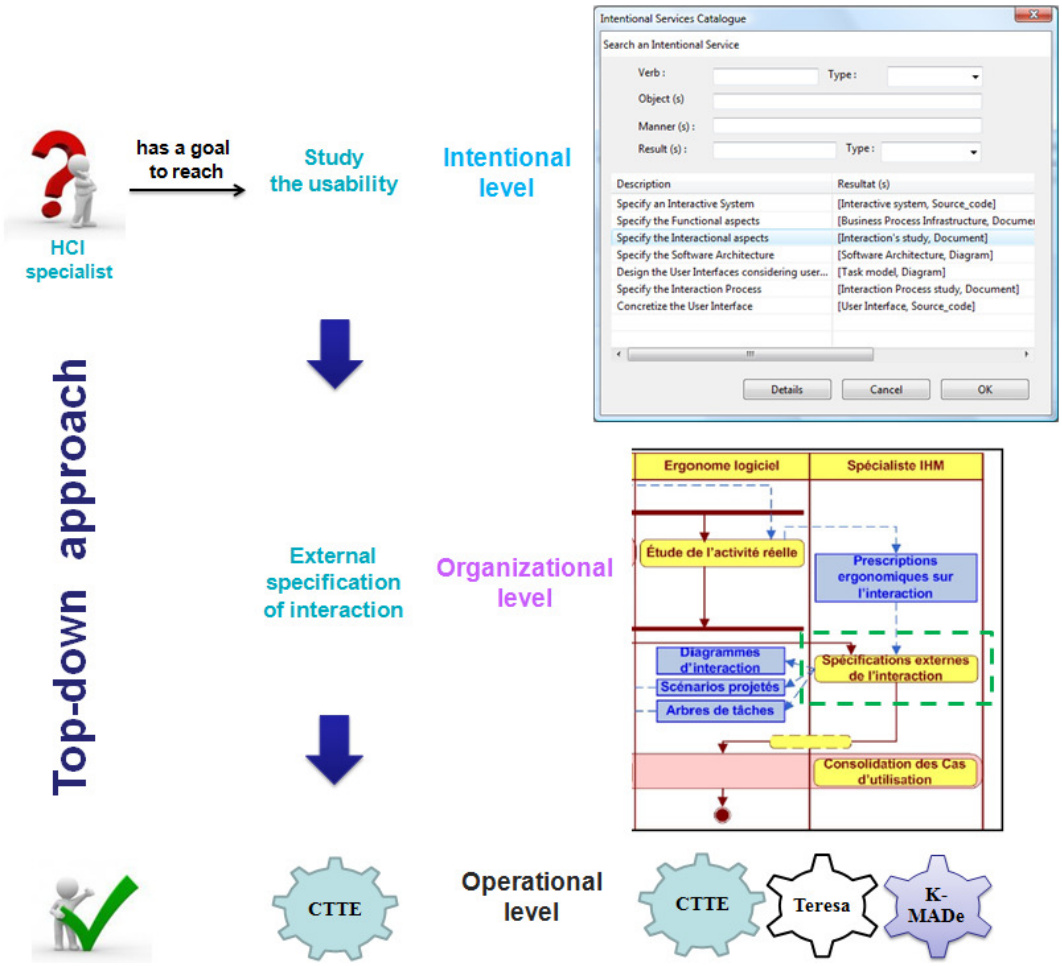


FIGURE 93. ILLUSTRATION DE L'UTILISATION DESCENDANTE DE LA PLATEFORME

Grâce à ces liens, il est ensuite possible de naviguer dans les différents niveaux. Le tableau 28 résume certains scénarios que nous avons considérés, pour les clients de la plateforme.

La vision du client. Pour le client le but est de chercher les services selon ses besoins.		
Le support intentionnel	**Navigation**	**Vision**
De l'intention vers les fragments de méthodes, ensuite les outils Les clients doivent chercher et choisir un service intentionnel selon les objectifs à atteindre. Le but choisi correspond au niveau organisationnel à des services organisationnels. Les clients font alors un choix parmi les services organisationnels associés au service intentionnel. Finalement, les clients peuvent utiliser les services offrant un environnement de modélisation support du processus élu.	Serv. intentionnel ↓ Serv. organisationnel ↓ Serv. opérationnel	Ces exemples correspondent à la vision ***top-down*** dans lequel le choix de l'environnement de modélisation constitue une décision stratégique soutenue par la sélection des objectifs de modélisation.
De l'outil vers les fragments de méthodes, puis les outils Le client connaît parfaitement l'un des outils qu'il souhaite utiliser. Cependant, il a besoin de s'appuyer sur les processus qui suggèrent d'utiliser cet outil. Ces processus suggèrent l'utilisation d'autres outils pour compléter son travail, par exemple : un client connaît	Serv. Opérationnel ↓ Serv. organisationnel	La recherche des services organisationnels est garantie par une vision ***bottom-up***.

parfaitement Visual-Paradigm (http://www.visual-paradigm.com/product/vpuml/) et il doit spécifier l'interaction de l'utilisateur, donc, il explore les services organisationnels qui utilisent cet outil. Il trouve que la méthode proposée par Godet-bar [Godet-bar *et al.*, 2007] suggère qu'utiliser de manière complémentaire l'outil CTT [Paternò, 1997].	↓ Serv. opérationnel	

TABLEAU 28. UTILISATION DE LA PLATEFORME DEPUIS LA PERSPECTIVE « CLIENTS »

De la même manière, le tableau 29 résume certains scénarios que nous avons considérés, pour les fournisseurs de la plateforme.

La vision du Fournisseur. Pour les fournisseurs l'objectif est de rajouter des services et de proposer une accréditation qui permet d'établir les liens avec des services existants de la couche supérieure et/ou de la couche inférieure.

La dynamique du service	Navigation	Vision
De l'outil vers les fragments de méthodes Le fournisseur d'outil ajoute un AGL gérant des modèles de cas d'utilisation, donc il doit proposer une accréditation de l'outil fourni à tous les ingénieurs de méthodes qui gèrent des fragments de méthodes manipulant des cas d'utilisation. Il s'agit d'indiquer pour n'importe quel outil les opérations qu'il réalise sur ces modèles.	Serv. opérationnel ↓ Serv. organisationnel ↓ Serv. opérationnel	Le concepteur de modèles utilise une vision ***bottom-up***, il s'agit de rechercher les services organisationnels et d'établir le lien de l'outil qu'elle offre.
Du fragment de méthode vers les intentions ou les outils L'ingénieur de méthode insère un fragment de méthode, par exemple : « spécification organisationnelle des besoins ». Il doit établir les liens avec les buts existants de la couche intentionnelle (par exemple : « la spécification fonctionnelle d'un système ») et, de la même façon, il doit établir les liens avec les services opérationnels (les outils) qui prend en charge l'édition des modèles manipulés par le fragment de méthode. Si nécessaire de nouveaux objectifs sont ajoutés afin d'enrichir la couche intentionnelle.	Serv. organisationnel ↓ Serv. intentionnel et Serv. organisationnel ↓ Serv. opérationnel	L'ingénieur de méthodes peut utiliser la vision ***bottom-up*** et ***top-down*** pour établir la liaison des fragments de méthode qu'il propose.

TABLEAU 29. UTILISATION DE LA PLATEFORME DEPUIS LA PERSPECTIVE « FOURNISSEURS »

3. SYNTHESE

Dans ce chapitre, nous avons présenté une plateforme à base de services pour la réutilisation de processus et des outils de modélisation. L'outil que nous avons présenté donne le support aux clients et aux fournisseurs dans la création, la recherche et la réutilisation des services intentionnels, organisationnels et opérationnels.

Le prototype expérimental est pour le moment limité à la création et recherche des services intentionnels. Les fonctionnalités permettant la création et recherche des services organisationnels et opérationnels, ainsi que le système d'accréditation des services est en cours de développement. A court terme, il devra être intégré à l'environnement ChiSpace.

Enfin, un effort sur l'ergonomie sera fait, pour faciliter l'appropriation et l'utilisation de la plateforme par les clients et les fournisseurs.

CHAPITRE 5

Conclusion et Perspectives

CHAPITRE 5

Ce chapitre clôture cette thèse en rappelant les contributions et en explicitant des perspectives possibles.

1. LE BILAN DES CONTRIBUTIONS

Les propositions présentées dans le cadre de cette thèse apportent des réponses à plusieurs problématiques de recherche dans le domaine de l'ingénierie de besoins, l'ingénierie des méthodes, l'ingénierie de modèles et la modélisation telles que : la création, la recherche et le stockage des buts, processus et outils de modélisation sous la forme de services. Ces résultats sont considérés dans une approche de gestion de modèles orientée services facilitant le choix des processus et des environnements de modélisation en fonction des besoins des concepteurs de modèles.

L'approche que nous proposons porte sur trois niveaux de services : le niveau intentionnel, le niveau organisationnel et le niveau opérationnel. Les trois niveaux offrent des services de gestion de modèles de natures différentes :

1. Le niveau opérationnel permet de définir des environnements de modélisation adaptés aux concepteurs de modèles.
2. Le niveau organisationnel garantit la réutilisation des services opérationnels d'une manière coordonnée, mais également, il garantit la création et la gestion des fragments de méthodes.
3. Le niveau intentionnel permet de définir les buts stratégiques des concepteurs de modèles, qui peuvent être mis en œuvre par des démarches de conception décrites au niveau organisationnel.

Nous avons montré comment un service intentionnel est réalisé par plusieurs services organisationnels, lui-même réalisé par plusieurs services opérationnels, en fonction des compétences spécifiques et des besoins opérationnels des concepteurs de modèles.

Chaque niveau de service est formalisé par un méta-modèle et un formalisme de représentation graphique conforme au méta-modèle. Les différents méta-modèles ont été illustrés au travers d'exemples concrets issus pour les niveaux intentionnels et organisationnels de fragments de

méthodes existantes (Symphony augmentée et RUP) et pour le niveau opérationnel d'outils de gestion de modèles.

Nos méta-modèles s'appuient sur des ontologies ou des normes existantes dans le domaine des systèmes d'information. Nous avons choisi d'intégrer les ontologies dans nos méta-modèles de manière à disposer de méta-modèles complets et uniformes de chaque niveau d'abstraction. La modélisation de ces ontologies s'appuie sur le concept de catégorisations. Nous avons présenté des patrons pour la catégorisation, puis une adaptation du patron Item-Description que nous avons appelé « Concept-Term », un mécanisme que nous avons défini pour réaliser l'intégration des ontologies dans les spécifications de nos méta-modèles de services.

L'approche orientée services pour la réutilisation de processus et des outils de modélisation est soutenue par un outillage (ou un environnement de modélisation) qui la met en œuvre et la rend utilisable. Nous avons développé un prototype support à nos propositions. Actuellement notre prototype permet d'ajouter et de rechercher des services intentionnels. En outre, nous présentons également un aperçu de la plateforme en cours d'élaboration qui permettra la gestion de nos trois niveaux de service. Finalement, nous avons analysé le fonctionnement global de la solution présentée.

2. LES PERSPECTIVES

Les propositions présentées dans cette thèse peuvent être améliorées à plusieurs niveaux, tant sur l'axe de l'approfondissement des travaux réalisés que sur celui de l'élargissement du domaine de recherche.

Approfondissement des propositions

1. Des travaux à venir incluront la finalisation de la plateforme orientée services. Notre plateforme se base sur des travaux encore en cours dans le domaine de la gestion de services. S'ils semblent nous offrir des possibilités intéressantes pour gérer nos trois niveaux de services, les outils que nous utilisons actuellement souffrent encore de nombreuses imperfections liés au fait que ce sont encore des prototypes (manque de documentation, bugs...). Suivant leur évolution, nous envisagerons de poursuivre le développement avec ces outils ou de choisir des technologies plus matures par exemple pour la gestion des services web qui offrent aujourd'hui des mécanismes de gestion de services aboutis (découverte, orchestration, etc.).

2. Quelque soit la technologie choisie, notre plateforme permettra de tester notre approche afin de valider nos propositions, d'analyser leur impact et leur utilisabilité. Pour ce faire, nous réaliserons des expériences auprès de concepteurs. Même si ce n'est qu'une première étape dans l'évaluation de notre solution, cela nous permettra également d'explorer d'autres fonctionnalités et de les intégrer dans notre solution.
3. Les processus d'usage de la plateforme sont encore très intuitifs. Il s'agira de formaliser d'une part les processus d' « alimentation » de la plateforme en fonction des différents types de fournisseurs (les fournisseurs d'outils de modélisation et les ingénieurs de méthodes) et d'autre part les processus d'usage de la plateforme en fonction des différents types de clients (les chefs de projets et les concepteurs de modèles).

Élargissement du domaine de la recherche

1. Nous considérons très important de créer une ontologie de méthodes. Dans cette ontologie chaque terme correspond à un fragment de méthode existante. Une telle ontologie peut être initialisée à partir des travaux existants mais ne peut vivre et évoluer qu'au travers d'une communauté active permettant de l'alimenter. Nous pensons utiliser le logiciel C-OPEN (COllaborative Pattern Environnement) en cours de développement dans l'équipe SIGMA pour la capitalisation et l'usage de services méthodes sous la forme de patrons collaboratifs.
2. Nous n'avons pas ou peu utiliser des techniques fondamentales des approches à base de services telles que la découverte et la composition dynamique des services. En particulier la composition dynamique de services au niveau organisationnel devrait faciliter l'assemblage cohérent de fragments de méthodes.
3. Des expérimentations poussées doivent nous permettre de valider ou d'invalider l'architecture en couches en trois niveaux de services. Au bout de cette thèse nous avons en particulier des doutes sur l'architecture en couche : la couche intentionnelle est-elle effectivement la couche de haut niveau ou est-elle orthogonale aux deux autres ? mais également sur les niveaux de services : les buts doivent-ils ou non être considérés comme des services à part entière ou comme il est fait généralement comme des descriptifs de services ?

BIBLIGRAPHIE

[**Action IDM, 2007**] Action IDM : Ingénierie dirigée par les modèles. http://www.actionidm.org/, consultation Juin (2007).

[**Agerfalk, 2003**] Agerfalk P.J.: Information systems actability: Understanding Information Techology as a Tool for Business Action and Communication. Doctoral dissertation. Dept. of Computer and Information Science, Linköping University, (2003).

[**Agerfalk et al., 2007**] Agerfalk P., Brinkkemper S., Gonzales-Perez C., Henderson-Sellers B., Karlsson F., Kelly S., Ralyté J.: Modularization Constructs in Method Engineering: Towards Common Ground?, Panel of ME 07, Springer, Geneva, Switzerland, (2007).

[**Allilaire et al., 2004**] Allilaire, F., Idrissi T.: ADT: Eclipse development tools for ATL. In Proceedings of the 2nd European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations (EWMDA-2). Computing Laboratory, University of Kent, Canterbury, UK. England. September (2004), 171-178.

[**Amelunxen et al., 2006**] Amelunxen C., Königs A., Rötschke T., Schürr A.: MOFLON: A Standard-Compliant Metamodeling Framework with Graph Transformations. In: Rensink A., Warmer J., (eds), Model Driven Architecture – Foundations and Applications: 2nd European Conference, Heidelberg: Springer Verlag, 2006; LNCS, Vol. 4066, pp. 361-375, (2006).

[**Annett et al., 1967**] Annett J, Ducan K D.: Task analysis and training design Journal of Occupational Psychology. Vol. 41, pp. 211-221. (1967).

[**Atkinson et al., 2003**] Atkinson C., Kuhne T.: Model-Driven Development: A Metamodeling Foundation. IEEE Software, 20(5) :36–41, (2003).

[**Baida et al., 2004**] Baida Z., Gordijn J., Omelayenko B., Akkermans H.: A Shared Service Terminology for Online Service Provisioning. Proceedings of the Sixth International Conference on Electronic Commerce (ICEC04), Delft, The Netherlands, (2004).

[**Baron et al., 2006**] Baron M., Lucquiaud V., Autard D., Scapin D.L. : K-MADe : un environnement pour le noyau du modèle de description de l'activité. Proceedings of the 18th French-speaking conference on Human-computer interaction (IHM'06), Montreal, Canada, April 18-21 (2006).

[**Barros et al., 2005**] A. Barros, M. Dumas, and P. Oaks, A critical overview of the web services choreography description language (ws-cdl), BPTrends – http://www.bptrends.com /publicationfiles /03-05-WP-WS-DCDL-Barros-et-al.pdf, March (2005).

[**Ben Achour et al., 1998**] Ben Achour C., Rolland C., Souveyet C.: A proposal for improving the quality of the organisation of scenarios collections, Proceedings of the 4th International Workshop on Requirements Engineering Foundation for Software Quality. E. Dubois, A. Opdahl, K. Pohl (Eds), June, Pisa, Italy. (1998).

[**Ben Achour, 1999**] Ben Achour C.: Extraction des Besoins par Analyse des Scénarios Textuels, Thèse du doctorat à l'Université de Paris6. Janvier, (1999).

[**Benatallah et al., 2003**] Benatallah B., Sheng Q-Z., Dumas M.: The Self-Serv environment for web services composition. IEEE Internet Computing, 7(1): 40-48, (2003).

[**Bechhofer et al., 2004**] Bechhofer S., Harmelen F., Hendler J., Horrocks I., McGuinness D., Patel-Schneider P., Stein L.: OWL web Ontology Language Reference. In Mike Dean and Guus Schreiber, Editors, W3C Recommendation (2004).

[**Benatallah et al., 2005**] B. Benatallah, R. M. Dijkman, M. Dumas, and Z. Maamar, Service-oriented software system engineering: Challenges and practices, ch. Service Composition: Concepts, Techniques, Tools and Trend, pp. 48 – 66, Idea Group Inc., (2005).

[**Benjamen, 1999**] Benjamen A.: Une Approche Multi-démarches pour la modélisation des démarches méthodologiques. PhD dissertation. University of Paris 1, Sorbonne, (1999).

[**Berardi et al., 2003**] Berardi D., Calvanese D., De Giacomo G., Lenzerini M., Mecella M.: Automatic Composition of eServices that export their Behavior. Proceedings of the 1st International Conference on Service Oriented Computing (ICSOC'03), Trento, Italy, (2003).

[**Bézivin, 2003**] Bézivin J. : Ecole d'Eté d'Informatique CEA EDF INRIA 2003, Ingénierie des modèles logiciels, (2003).

[**Bézivin J., 2004a**] Bézivin J.: Sur les principes de base de l'ingénierie des modèles. RSTI-L'Objet, 10(4), pp. 145-157, (2004).

[**Bézivin J., 2004b**] Bézivin J.: In Search of a Basic Principle for Model Driven Engineering. CEPIS, UPGRADE, The European Journal for the Informatics Professional, V(2): 21-24, (2004).

[**Bézivin J., 2005**] Bézivin J.: On the unification power of models. Software and System Modeling (SoSym) 4(2), pp. 171-188, (2005).

[**Bézivin et al., 2001**] Bézivin J., Gerbé O.: Towards a Precise Definition of the OMG/MDA Framework. In: Proceedings of the 16th IEEE international conference on Automated Software Engineering (ASE). IEEE Computer Society Press, page 273, San Diego, USA, (2001).

[**Bézivin et al., 2002**] Bézivin J., Blanc X. : MDA vers un nouveau paradigme, Jounal Développeur Référence, 15 juillet 2002, v2.16, pp. 7-11.

[**Bézivin et al., 2004**] Bézivin J., Jouault F., Valduriez P.: On the Need for Megamodels. In: Proceedings of the OOPSLA/GPCE: Best Practices for Model-Driven Software Development workshop, (2004).

[**Bézivin et al., 2005a**] Bézivin J., Jouault F., Rosenthal P., Valduriez P.: Modeling in the Large and Modeling in the Small. In U. Aßmann, M. Aksit, and A. Rensink, editors, Model Driven Architecture, European MDA Workshops : Foundations and Applications, MDAFA 2003 and MDAFA 2004, , June 26-27, 2003 and Linköping, Sweden, June 10-11, 2004, Revised Selected Papers, volume

3599 of Lecture Notes in Computer Science, pages 33–46, Twente, The Netherlands, 2005. Springer.

[**Bézivin et al., 2005b**] Bézivin J., Kurtev I.: Model-based Technology Integration with the Technical Space Concept. In: Metainformatics Symposium, Esbjerg, Denmark, Springer-Verlag, (2005).

[**Bézivin et al., 2006**] Bézivin J., Büttner F., Gogolla M., Jouault F., Kurtev I., Lindow A.: Model Transformations? Transformation Models! MoDELS'2006, pp. 440-453, (2006).

[**Bieber et al., 2001**] Bieber G.: Carpenter J.: Introduction to Service-Oriented Programming (Rev 2.1), Avril (2001).

[**Blanco et al., 2007**] Blanco E., Grebici K., Rieu D.: A unified framework to manage information maturity in design process. In: International Journal of Product Development. Volume 4, Number 3-4, pp. 255-279, (2007).

[**Boehm, 1988**] Boehm B.: A Spiral Model of Software Development and Enhancement, IEEE Computer, Vol. 21 (5), (1988).

[**Boehm et al., 1978**] Boehm B., Brown J., Kaspar H., Lipow M., McLeod G., Merritt M.: Characteristics of Software Quality, Elsevier North Holland, (1978).

[**Bonnet et al., 2003**] Bonnet N., Cleuet F., Salzman, C.: Maîtrise d'ouvrage de projet de système d'information - principes rôles responsabilités facteurs de succès. Number ISBN 2 951 5149 7 2. sous la direction de D. Moisand, AFAI collection pratiques professionnelles. (2003).

[**Booch, 1991**] Booch G.: Object Oriented Analysis and Design with Applications, Benjamin/Cummings, (1991).

[**Booch et al, 1997**] Booch G., Jacobson I., Rumbaugh J.: Unified Modeling Language: version 1.0. Rational Software Coorporation, (1997).

[**Brinkkemper, 1990**] Brinkkemper S.: Formalisation of information systems modelling, Ph. D. Thesis, University of Nijmegen, Thesis Publishers, Amsterdam, (1990).

[**Brinkkemper, 1996**] Brinkkemper S.: Method Engineering: engineering of information systems development methods and tools, Information and Software Technology, Vol. 38, No.4, pp.275-280, (1996).

[**Brinkkemper et al., 1996**] Brinkkemper S., Saeki M., Welke R.: Method Engineering: Principles of method construction and tool support, Chapman &Hall, Springer, pages 336, (1996).

[**Brinkkemper et al., 1998**] Brinkkemper S., Saeki M., Harmsen F.: Assembly Techniques for Method Engineering, Proceedings of the 10th Conference on Advanced Information Systems Engineering, CAiSE'98, Pisa Italy, 8-12 June, (1998).

[**Brinkkemper et al., 1999**] Brinkkemper S., Saeki M., Harmsen F.: Meta-Modelling Based Assembly Techniques for Situational Method Engineering, Information System, Vol. 24 (3), pp. 209-228. (1999).

[**Bubenko et al., 1994**] Bubenko J., Kirikova M.: Worlds' in Requirements Acquisition Modelling, 4th European - Japanese Seminar on Information Modelling and Knowledge Bases, Kista, Sweden, Kangassalo and Wangler (Eds.), IOSpub),(1994).

[**Budinsky et al., 2003**] Budinsky F., Steinberg D., Ellersick R.: Eclipse Modeling Framework: A Developer's Guide. Addison-Wesley Professional, (2003).

[**Caroll, 1995**] Caroll J.: The Scenario Perspective on System Development, in Scenario-Based Design: Envisioning Work and Technology in System Development, Ed J.M. Carroll, (1995).

[**Christensen et al., 2001**] Christensen E., Curbera F., Meredith G., Weerawarana S.: Web Services Description Language (WSDL) 1.1, in W3C Notes, http://www.w3.org/TR/wsdl, (2001).

[**Chung et al., 2000**] Chung L., Nixon B., Yu E., Mylopoulos J.: Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, (2000).

[**Coad, 1992**] Coad P.: Object-Oriented Patterns. Communication of ACM, Volume 35 nro. 9, ACM Press, pp. 152-159, (1992).

[**Cook et al., 2007**] Cook S., Jones G., Kent S., Wills Cameron A.: Domain Specific Development with Visual Studio DSL Tools. Addison-Wesley Professional, 576 pages, (2003).

[**Corby et al., 2008**] Corby O., Dieng-Kunts R., Faron-Zucker C.: Querying the semantic web with the CORESE search engine. In 16th European Conference on Artificial Intelligence (ECAI/PAIS). Pp. 705-709. Valencia, Spain, (2008).

[**Czarnecki et al., 2003**] Czarnecki K., Helsen S.: Classification of Model Transformation Approaches. In OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture, (2003).

[**Dano et al., 1997**] Dano B., Briand H., Barbier F.: A use case driven requirements engineering process. Third IEEE International Symposium On Requirements Engineering RE'97, Antapolis, Maryland, IEEE Computer Society Press, (1997).

[**Dardenne et al., 1991**] Dardenne A., Fickas S., Lamsweerde van A. : Goal - directed concept acquisition in requirements elicitation. Proceedings of the 6th IEEE Workshop System Specification and Design, Como, italy, 14-21, (1991).

[**Dardenne et al., 1993**] Dardenne A., van Lamsweerde A., Fickas S.: Goal-directed requirements acquisition, Science of Computer Programming, 20 (1-2), avril (1993).

[**Deneckere, 2001**] Deneckere R.: Approche d'extension de méthodes fondée sur l'utilisation de composants génériques, PhD thesis, University of Paris 1-Sorbonne, (2001).

[**Deneckere, 2002**] Deneckere R: Using meta-patterns to construct patterns, Proceedings of OOIS'2002 Conférence, Montpellier, France, (2002).

[**Deneckere, 2002**] Deneckere R: Construction de patrons à l'aide de méta-patrons, revue ISI, vol. 7/4, pp. 83-105, (2002).

[**Deneckere et al., 1998**] Deneckere R., Souveyet C.: Patterns for extending an OO model with temporal features. Proceedings of OOIS'98 conference. Springer-Verlag. Paris (France), (1998).

[**Deneckere et al., 2001**] Deneckere R., Souveyet C.: Organising and Selecting Patterns in Pattern Languages with Process Maps, Proceedings of OOIS'2001 Conference, Springer-Verlag, Calgary (Canada), (2001).

[**Deneckère et al., 2008**] Deneckère R., Iacovelli A., Kornyshova E., Souveyet C., « From Method Fragments to Method Services », In Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'08), Montpellier, France, (2008).

[**De Bruijn et al., 2005**] De Bruijn J., Bussler C., Dominque J., Fensel D., Hepp M.: Web service modeling ontology (wsmo) specification, http ://www.w3.org/Submission/WSMO/, June (2005).

[**De matos, 2009**] De matos J.: Mise au point d'un prototype orienté services pour la gestion de modèles. Mémoire de stage. Département d'informatique de l'IUT2 de Grenoble. Promotion 2A. May - Juin (2009).

[**Diaper et al., 2003**] Diaper D., Stanton N.: The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Associates, Mahwah, pp. 568, September 1, (2003).

[**Didonet Del Fabro et al., 2005**] Didonet Del Fabro M., Bézivin J., Jouault F., Breton E., Gueltas G.: AMW: a generic model weaver. In Proceedings of the 1ère Journée sur l'Ingénierie Dirigée par les Modèles (IDM'05). Paris, France, June 2005. Sébastien Gérard, Jean-Marie Favre, Pierre-Alain Muller, Xavier Blanc, Editors Paris, 105-114, France (2005).

[**Didonet et al., 2007**] Didonet Del Fabro M., Jouault F.: Model Transformation and Weaving in the AMMA Platform, In Proceedings of the International Summer School in Generative and Transformational Techniques in Software Engineering (GTTSE'05). Braga, Portugal, July 2005; LNCS, Vol. 4143, pp. 71-77, Springer Verlag, (2007).

[**Dowson, 1988**] Dowson M.: Iteration in the Software Process. Proceedings of the 9th International Conference on Software Engineering (ICSE'98). (1988).

[**Dromey, 1995**] Dromey R.: A model for software product quality, IEEE Transactions on Software Engineering, no. 2, pp. 146-163, (1995).

[**Dubois et al., 1989**] Dubois E., Hagelstein J, Rifaut A.: Formal Requirements Engineering with ERAE, Philips Journal Research, Vol L 43, No 4, (1989).

[**Dupuy-Chessa et al., 2009**] Dupuy-Chessa S., Godet-Bar G., Pérez-Medina J-L., Rieu D., Juras D.: A Software Engineering Method for the Desing of Mixed Reality Systems. The Engineering of Mixed Reality Systems, chapter 15. Dubois E., Gray P., Nigay L (ed.). Springer, pp. 313-334, (2010).

[**Eclipse, 2007**] Eclipse Modeling Project. Official site: http://www.eclipse.org/modeling/, February (2007).

[**Ellis et al., 1991**] Ellis C., Gibbs S., Rein G. : Groupware: Some Issues and Experiences. Journal, Communications of the ACM (CACM), volume 34, numéro 1, pages 38-58, ACM Press, (1991).

[**Ellis et al., 1994**] Ellis C., Wainer J.: A conceptual model of groupware. *CSCW'94*, Chapel Hill, NC, (1994).

[**Etien et al., 2007**] Etien A., Dumoulin C., Renaux E. : Towards a unified notation to represent model transformation. Rapport de recherche RR-6187, INRIA, mai (2007).

[**Favre J-M., 2004a**] Favre J-M. : Foundations Of MetaPyramids Languages Vs Metamodels Episodel I Story Of Thotus The Baboon. Daghstull, Germany, (2004).

[**Favre J-M., 2004b**] Favre J-M.: Foundations Of Model Driven Reverse Engineering Models Episode II Stories Of The Fidus Papyrus And Of The Solarus. Dagshtull, Germany, (2004).

[**Favre J-M., 2004c**] Favre J-M.: Towards a Basic Theory to Model Model Driven Engineering. In Workshop on Software Model Engineering (WISME), joint event with UML'2004, Lisboa, (2004).

[**Favre et al., 2006**] Favre J-M., Estublier J., Blay M.: L'Ingénierie Dirigée par les Modèles : au-delà du MDA. Informatique et Systèmes d'Information. Hermes Science, lavoisier edition, February (2006).

[**Fensel et al., 2002**], Fensel D., Bussler C.: The web service modeling framework WSMF, Electronic Commerce Research and Applications, (2002).

[**Ferguson et al., 2005**] Ferguson D., Stockton M. L.: Service-Oriented Architecture: Programming Model and Product Architecture, IBM Systems Journal 44 nro. 4, pp. 753-780, (2005).

[**Finkelstein et al., 1990**] Finkelstein A., Kramer J., Goedicke M. : ViewPoint Oriented Software Development, Actes de Conférence "Le Génie Logiciel et ses Applications", Toulouse, p 337-351, (1990).

[**Fowler, 1996**] Fowler M.: In Analysis Patterns - Reusable Object Models, number ISBN 0 201 89542 0. Addison-Wesley Publishing Company. (1996).

[**Fowler, 1997**] Fowler M.: UML distillated: applying the standard object modeling notation. Addison-Wesley, Reading, MA, (1997).

[**Franckson, 1991**] Franckson M., Peugeot C.: Specification of the object and process modeling language, ESF Report n° D122-OPML-1.0, (1991).

[**Gamma et al., 1995**] Gamma E., Helm R., Johnson R.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. Addison-Wesley, (1995).

[**García et al., 2004**] García J., Rodríguez J., Menárguez M., Ortín M., Sánchez J., *« Un estudio comparativo de dos herramientas MDA : OptimalJ y ArcStyler »*, Taller de Desarrollo de Software Dirigido por Modelos, DSDM'04 en JISBD'2004, pp. 88-98, November (2004).

[**Godet-Bar et al., 2007**] Godet-Bar G., Juras D., Dupuy-Chessa S., Rieu D., « Vers une méthode de conception de systèmes mixtes : Principes et mise en oeuvre », *RSTI-ISI*. Vol 12, p. 39-66, (2007).

[**Grady, 1992**] Grady R..: Practical software metrics for project management and process improvement, Prentice Hall, (1992).

[**Graham, 1994**] Graham I.: Object Oriented Methods. Addison-Wesley, (1994).

[**Gray et al., 2001**], Gray J., Bapty T., Neema S., Tuck J.: Handling Crosscutting Constraints in Domain-Specific Modeling, Communications of the ACM, pp. 87-93, October (2001).

[**Greenyer, 2006**] Greenyer J.: A study of Model Transformation Technologies: Reconciling TGGs with QVT. – Master Thesis - University of Panderborn. (2006).

[**Gudgin et al., 2007**] Gudgin M., Hadley M., Mendelsohn N., Moreau J.J., Nielsen H.F., Karmarkar A., Lafon Y.: SOAP V.1.2, in W3C Recommendations, http://www.w3.org/TR/soap/, (2007).

[**Guzélian, 2007**] Guzélian G.: Conception de systèmes d'information : une approche orientée services, Thèse de Doctorat soutenue à l'université Paul Cezanne, Marseille, juillet (2007).

[**Guzélian et al., 2004**] Guzélian G., Cauvet C., Ramadour P. : Conception et réutilisation de composants : une approche par les buts. In Actes du XXIIème Congrès INFORSID'04, pages 179–194, Biarritz, (2004).

[**Guzélian et al., 2007**] Guzélian G., Cauvet C.: SO2M: Towards a Service-Oriented Approach for Method Engineering, in: the 2007 World Congress in Computer Science, Computer Engineering and Applied Computing, in the proceedings of the international conference IKE'07, Las Vegas, Nevada, USA, (2007).

[**Hagelstein, 1988**] Hagelstein J.: Declarative Approach to Information Systems Requiremnts, in Knowledge-Based Systems, Vol&, No4, (1988).

[**Harmsen, 1997**] Harmsen F.: Situational Method Engineering. Moret Ernst & Young, (1997).

[**Harmsen et al., 1994**] Harmsen A., Brinkkemper J., Oei J.: Situational Method Engineering for Information System Projects. In Olle T. W. and A. A. Verrijn Stuart (Eds.), Methods and Associated Tools for the Information Systems Life Cycle. Proceedings of the IFIP WG8.1 Working Conference (CRIS'94). pp. 169-194. North-Holland - Amsterdam, (1994).

[**Hassine et al., 2002**] Hassine I., Rieu D., Bounaas F., Seghrouchni O.: Symphony: a conceptual model based on business components, in SMC'02, IEEE International Conference on Systems, Man, and Cybernetics. Volume 2. (2002).

[**Heineman et al., 2001**] Heineman G., Councill W.: In Component-Based Software Engineering : Putting the Pieces Together, number ISBN 0 201 70485 4. Addison-Wesley, Longman Publishing Co., Inc., Boston, MA, USA. 818 pages. (2001).

[**Henderson-Sellers, 2002**] Henderson-Sellers B.: Process meta-modelling and process construction: examples using the OPF. Ann. Software Engineering, 14(1-4), (2002).

[**Henkel et al., 2004**] Henkel M., Zdravkovic J., Johannesson P.: Service-based Processes – Design for Business and Technology. International Conference on Service Oriented Computing (ICSOC'04). New York, USA, (2004).

[**Horvath et al., 2006**] Horvath A., Varro D.: The VIATRA2 Model Transformation Framework. Advanced School on visual modeling techniques, (segravis). UK, (2006).

[**Hug, 2009**] Hug Ch. : Méthode, modèles et outil pour la méta-modélisation des processus d'ingénierie de systèmes d'information. Thèse de Doctorat soutenue à l'université Joseph Fourier, Grenoble, octobre (2009).

[**Hull et al., 2003**] Hull R., Benedikt M., Christophides V., Su J.: E-Services: A Look Behind the Curtain, Proceedings of the 22nd ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'03), ACM, 2003, pp. 1–14, (2003).

[**Hüsemann et al., 2000**] Husemann B., Lechtenborger J., Vossen G.: Conceptual data warehouse modelling. In Jeusfeld et al. [2000], page 6. (2000).

[**Iacovelli et al., 2008**] Iacovelli A., Souveyet C., Rolland C.: Method as a Service (MaaS). RCIS'08: pp. 371-380, (2008).

[**Iacovelli et al., 2008**] Iacovelli A., Souveyet C.: Framework for Agile Methods Classification. MoDISE-EUS'08: 91-102, (2008).

[**IEEE std 1061**] IEEE std 1061: Standard for a Software Quality Metrics Methodology, (1992).

[**IEEE std 729, 1983**] ANSI/IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology, (1983).

[**ISO std 9126**] ISO/IEC International Organization for Standardization: ISO Standard 9126: Information Technology – Software product evaluation – Quality characteristics and guidelines for their use (1991).

[**Jackson, 1995**] Jackson M.: Software Requirements and Specifications – A Lexicon of Practice, Principles and Pejudices. ACM Press, Addison-Wesley, (1995).

[**Jacobson et al., 1992**] Jacobson I., Christenson M., Jonsson P., Oevergaard G.: Object Oriented Software Engineering: a Use Case Driven Approach. Addison-Wesley, (1992).

[**Jacobson et al., 1999**] Jacobson I., Booch G., Rumbaugh J.: The Unified Process A Software Engineering Process Using the Unified Modelling Language, Addison-Wesley Longman, (1999).

[**Jarke et al., 1992**] Jarke M., Pohl K.: Information systems quality and quality information systems. Proceedings of the IFIP 8.2 Working Conference on the impact of computer-supported techniques on information systems development, Minneapolis, NM, june (1992).

[**Jarke et al., 1995**] Jarke M., Pohl K., Dömges R., Jacobs S., Nissen H.W.: Requirements Information Management: The NATURE Approach, Ingénierie des Systèmes d'Informations (Special Issue on Requirements Engineering), Vol.2, No. 6, (1995).

[**Jini, 2010**] The community ressource for jini technology - jini technology, site officiel. Date de consultation: 10 février 2010 http://www.jini.org/wiki/.

[**JMI, 2002**] JMI: Java Metadata Interface, reference implementation. Unisys Corporation. JMI-RI Documentation. CIM Guide. Version 1.3. October (2002).

[**Johnson et al., 1991**] Johnson H., Johnson P.: Task Knowledge Structures : Psychological basis and integration into system design, Acta Psychologica, pp. 3-26, (1991).

[**Johnson et al., 1997**] Johnson R., Woolf B.: The Type Object. Pattern languages of program design 3. Addison-Wesley Software Pattern Series. Boston USA, pp. 47-65, (1997).

[**Jouault F., 2006**] Frédéric Jouault. Contribution à l'étude des langages de transformation de modèles. PhD thesis, Université de Nantes, September (2006).

[**Jouault et al., 2006**] Jouault F., Bézivin J.: KM3 : a DSL for Metamodel Specification. In: Proceedings of the IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS), volume 4037 of Lecture Notes in Computer Science, pages 171–185. Springer, (2006).

[**Juras et al., 2006**] Juras D., Rieu D., Dupuy-Chessa S., Front A.: Conception collaborative pour les Systèmes Mixtes. In *Revue Génie Logiciel num. 77,* GL-IS, p. 31-36, (2006).

[**Kaabi, 2007**] Kaabi R.S.: Une Approche Méthodologique pour la Modélisation Intentionnelle des Services et leur Opérationnalisation, PhD thesis, Université Paris 1-Sorbonne. Réalisé au Centre de Recherche en Informatique - CRI, (2007).

[**Kalnins et al., 2004**] Kalnins A., Barzdins J., Celms E.: Model Transformation Language MOLA. In: Proceedings of Model-Driven Architecture: Foundations and Applications (MDAFA 2004), Linkoeping, Sweden, June 10-11, pp. 14–28 (2004).

[**Karlsson, 2002**] Karlsson F.: Meta-Method for Method Configuration - A Rational Unified Process Case. PhD thesis, Faculty of Arts and Sciences at Linköping University, Linköping, Sweden. (2002).

[**Karlsson, 2005**] Karlsson F.: Method Configuration: Method and Computerized Tool Support. Doctoral dissertation. Dept of Computer and Information Science. Linköping University. (2005).

[**Kleppe et al., 2003**] Kleppe A., Warmer S., Bast W.: MDA explained: The model-driven architecture: Practice and promise; Addison-Wesley, 192 pages. April (2003).

[**Kiczales et al., 1997**] Kiczales G., Lamping J., Menhdhekar A., Maeda Ch., Lopes C., Loingtier J-M., Irwin J.: Aspect-Oriented Programming. In: M. Aksit and S. Matsuoka, editors, Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP), volume 1241 of Lecture Notes in Computer Science, pages 220–242. Springer, Jyväskylä, Finland, June (1997).

[**Kronlof, 1993**] Kronlof K.: Method Integration, Concepts and Case studies, Wiley series in software based systems, John Wiley and sons Ltd., (1993).

[**Kruchten, 2000**] Kruchten P.: In Introduction au Rational Unified Process, number ISBN 2 212 09104 4. Editions Eyrolles, collection Référence, Paris. 282 pages. (2000).

[**Kumar et al., 1992**] Kumar K., Welke R-J.: Methodology Engineering: a proposal for situation-specific methodology construction, dans Cotterman, W.W. et Senn J.A. (Eds.), Challenges and Strategies for Research in Systems Development, John Wiley & Sons Ltd, pp. 257-269. (1992).

[**Kurtev et al., 2003**] Kurtev I., Bézivin J., Aksit M.: Technological Spaces: An Initial Appraisal. In: CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, (2002).

[**Kurtev et al., 2005**] Kurtev I., Bezivin J.: Bridging the MS/DSL Tools and the Eclipse Modeling Framework Software Factory workshop at OOPSLA2005. - 2005.

[**Lacot, 2005**] Lacot X.: Introduction à OWL, un langage XML d'ontologies Web, Ecole Nationale Supérieure des Télécommunications, Paris, Juin. (2005).

[**Lamsweerde, 2000**] Lamsweerde van A.: Requirements Engineering in the Year 00: A Research Perspective, Keynote paper, Proceedings of International Conference on Software Engineering (ICSE'00), ACM Press, (2000).

[**Lee et al., 2005**] Lee Chiung-Hon, Liu A.: Toward Intention Aware Semantic Web Service Systems. IEEE SCC'05, pp. 69-76, (2005).

[**Ledeczi et al., 2001**] Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason C., Nordstrom G., Sprinkle J., Volgyesi P.: The Generic Modeling Environment. In: Proceedings of the IEEE Workshop on Intelligent Signal Processing (WISP), Budapest, Hungary, (2001).

[**Limbourg et al., 2001**] Limbourg Q., Pribeanu C., Vanderdonckt J.: Towards Uniformed Task Models in a Model-Based Approach, 8th International Workshop on Interactive Systems: Design, Specification, and Verification-Revised Papers, pp. 164-182, (2001).

[**Lu et al., 2002**] Lu S., Paris C., Linden K.: Tamot: Towards a Flexible Task Modeling Tool. In Proceedings of Human Factor Conference, Melbourne, Victoria, Australia, (2002).

[**Lyytinen et al., 1989**] Lyytinen K., Smolander K., Tahvainen V-P.: Modelling CASE Environments in sytems Work, CASE'89 conference papers, Kista, Sweden, (1989).

[**Maesano et al., 2003**] Maesano L., Bernard C., Le Galles X.: In Services Web avec J2EE et .NET: conception et implémentations, number ISBN 2 212 11067 7. Eyrolles, Paris. 1056 pages. (2003).

[**Maiden, 1998**] Maiden N.A.M. CREWS-SAVRE: Scenarios for acquiring and validating requirements. Journal for Automated Software Engineering, 5(4):419-446, (1998).

[**Marín, 2009**] Marín J-C.: Mise au point d'un prototype orienté services pour la gestion de modèles. Mémoire de stage. Département d'informatique de l'IUT2 de Grenoble. Promotion Année Spéciale. Juillet - Août (2009).

[**Martin et al., 2005**] Martin D., and all: OWL-S: Semantic markup for web services, version 1.1, Available at http://www.daml.org/services/owl-s/1.1/, (2004).

[**Matula M., 2003**] Matula M.: NetBeans Metadata Repository. SUN Microsystems, March (2003).

[**McCall et al., 1977**] McCall J., Richards P., Walters G.: Factors in Software Quality", Nat'l Tech.Information Service, no. Vol. 1, 2 and 3, (1977).

[**Mecella et al., 2002**] Mecella M., Pernici B.: Building Flexible and Cooperative Applications Based on e-Services. Dipartimento di Informatica e Sistemistica, Universita' di Roma "La Sapienza", Technical Report 21-2002, Roma, Italy, (2002).

[**MediniQVT, 2007**] Medini QVT Medini QVT, Transformation Engineers. (IKV++) site de consultation: http://www.ikv.de/index.php?option=com_content&task=view&id=75&Itemid=77.

[**Meylan, 2006**] Meylan S. Rapport de projet Master 2 Informatique *« Etude et comparaison d'outils de transformation de modèles »*, Université de Franche-Comte, pages 53, Janvier (2006).

[**Mens et al., 2006**] Mens T., Van Gorp P.: A Taxonomy of Model Transformation. Electronic Notes in Theorical Computer Science 152, 125–142 (2006).

[**Minsky M., 1965**] Minsky M.: Matter, Minds, and Models. In: Proceedings of International Federation of Information Processing Congress, New York, United States, vol. 1, pp. 45-49, (1965).

[**Mintzberg, 1982**] Mintzberg H. Structure et dynamique dans les organisations. Editions d'Organisation, Pris, (1982).

[**Mirbel et al., 2006**] Mirbel I., Ralyté J., Situational Method Engineering: Combining Assembly-Based and Roadmap-Driven Approaches. Requirements Engineering, 11(1), pp. 58–78, (2006).

[**Mirbel, 2007**] Mirbel I.: Connecting Method Engineering Knowledge: a Community Based Approach, In Proc. of IFIP WG8.1 Working Conference on Method Engineering, Springer, Geneva, Switzerland., September, pp 176-192, (2007).

[**Mirbel et al., 2009**] Mirbel I., Crescenzo P.: Improving Collaborations in Neuroscientist Community, Rapport de recherche nro. I3S/RR-2009-05-FR, I3S Laboratory, Avril (2009).

[**Mirbel et al., 2009**] Mirbel I., Crescenzo P.: Des Besoins des Utilisateurs à la Recherche de Services WEB : Une Approche Sémantique Guidée Par les Intentions, Rapport de recherche nro. I3S/RR-2009-12-FR, I3S Laboratory, Septembre (2009).

[**Mori et al., 2002**] Mori G., Paternò F., Santoro C. : CTTE :Support for Developping and Analyzing Task Models for Interactive System Design. IEEE Transactions on Software Engineering, pp. 797-813, August (2002).

[**Muller et al., 2005**] Muller P., Fleurey F., Jézéquel J.: Weaving Executability into Object-Oriented Meta-Languages. In: L. Briand and C. Williams, editors, Proceedings of the 8th IEEE/ACM International Conference on Model Driven Engineering Languages and Systems (MoDELS), volume 3713 of Lecture Notes in Computer Science, pages 264–278, Montego Bay, Jamaica, Springer, October (2005).

[**Muller, 1997**] Muller P.: Modélisation objet avec UML. Eyrolles, (1997).

[**Normand, 1992**] Normand V. : Le modèle SIROCO : de la spécification conceptuelle des interfaces utilisateur à leur réalisation, thèse de l'Université Joseph Fourier, Grenoble, (1992).

[**Normand, 1992**] Normand V. : Task modelling in HCI : purposes and means, Rapport de Recherche n° PTI/92-02, Thomson CSF, Division systèmes défense et contrôle, 7 rue des Mathurins, BP 10, 92223 Bagneux Cedex, Juillet (1992).

[**Nigay et al., 2008**] Nigay L., Bouchet J., Juras D., Mansoux B., Ortega M., Serrano M., Lawson L.: Software Engineering for Multimodal Interactive Systems. In Multimodal user interfaces: from signals to interaction. D. Tzovaras (ed.), Lecture Notes in Electrical Engineering, Springer-Verlag, Berlin, pp. 201-218, (2008).

[**Nilsson, 1971**] Nilsson N-J.: Problem Solving Methods in Artificial Intelligence. McGraw Hill, (1971).

[**OASIS-Portlets, 2007**] OASIS: Web Services for Remote Portlets Specification 1.0, http://www.oasisopen.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0. pdf (2003).

[**OASIS-UDDI, 2004**] OASIS: UDDI Version 3.0.2, http://www.oasis-open.org/committees/uddispec/doc/spec/v3/uddi-v3.0.2-20041019.htm, (2004).

[**OASIS-WSBPEL, 2007**] OASIS: Web Services Business Process Execution Language Version 2.0, http://docs.oasisopen.org/wsbpel/2.0/wsbpel-v2.0.pdf, (2007).

[**Olle et al., 1992**] Olle T., Hagelstein J., MacDonald I., Rolland C., Sol H., Van Assche F., Verrijn-Stuart A.: Information Systems Methodology: a Framework for Understanding, Addison-Wesley. (1992).

[**OMG-MDA, 2003**] Object Management Group: MDA Guide Version 1.0.1, Juin 2003. http://www.omg.org/docs/omg/03-06-01.pdf, consultation Juin (2009).

[**OMG-XMI, 2003**] Object Management Group: XML Metadata Interchange (XMI) Specification, Version 2.0, formal/03-05-02, May (2003).

[**OMG, 2006**] Object Management Group: Meta Object Facility (MOF) 2.0 Core Specification, Final Adopted Specification, January (2006).

[**OMG, 2007a**] Object Management Group, Inc. Unified Modeling Language (UML) 2.1.2 Infrastructure, November 2007. Final Adopted Specification.

[**OMG, 2007b**] Object Management Group, Inc. Unified Modeling Language (UML) 2.1.2 Superstructure, November 2007. Final Adopted Specification.

[**OMG, 2008**] OMG. Software & Systems Process Engineering Meta-Model Specification (SPEM), Version 2.0. Formal/2008-04-01, Object Management Group, (2008).

[**OMG-QVT, 2005**] OMG-QVT: Query View Transformation - http://www.omg.org/docs/ptc/05-11-01.pdf.

[**Orriëns et al., 2003**] Orriëns B., Yang J., Papazoglou M-P.: Model Driven Service Composition, in Proc. Of the 1st International Conference on Service Oriented Computing (ICSOC'03), Trento, Italy, (2003).

[**OSGi, 2010**] OSGi Alliance, site officiel. Date de consultation : 10 février 2010. http://www.osgi.org/Main/HomePage.

[**OWL-S, 2004**] OWL-S Coalition, OWL-S Specification. http://www.daml.org/services/owl-s/1.1/, (2004).

[**Pallos, 2001**] Pallos M.: Service Oriented Architecture: A Primer, EAI Journal, December (2001).

[**Papazoglou, 2003**] Papazoglou, M.P.: Service-Oriented Computing: Concepts, Characteristics and Directions, in the Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03), pp. 3-12, Roma, Italy, December 10-12, (2003).

[**Papazoglou et al., 2005**] Papazoglou M-P., Traverso P., Dustdar S., Leymann F.: Service-Oriented Computing : A Research Roadmap, Dahstuhl Seminar 05462, p. 1-29, (2005).

[**Papazoglou et al., 2007**] Papazoglou M-P., van den Heuvel W-J.: Business Process Development Lifecycle Methodology. In communications of ACM, Volume 50 Issue 10, pp. 79-85, (2006).

[**Patil et al., 2004**] Pastil A., Oundhakar S., Sheth A., Verma K.: Semantic Web Services: Meteor-s Web Service Annotation Framework, 13th International Conference on World Wide Web, pp. 553-562, New York, USA (2004).

[**Paternò, 1997**] Paternò F.: ConcurTask Tree: a diagrammatic notation for specifying task models, Proc. of Interact'97, p. 362-369. Sydney, Australia, (1997).

[**Paternò et al., 2003**] Paternò F., Santoro C.: A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms, Interacting with Computers, volume 15. pp. 347-364. Elsevier, (2003).

[**Penserini et al., 2006**] Penserini L., Perini A., Susi A., Mylopoulos J.: From Stakeholder Needs to Service Designs. Requirements Engineering Journal 35, (2006).

[**Pérez-Medina et al., 2007**] Pérez-Medina J-L, Marsal-Layat S.: Transformation et vérification de cohérence entre modèles du Génie Logiciel et modèles de l'Interface Homme-Machine. INFORSID 2007: 382-397, Perros-Guirec, France, May (2007).

[**Pérez-Medina et al., 2009**] Pérez-Medina J-L., Dupuy-Chessa S., Rieu D.: In: D. England et al. (Eds.): TAMODIA 2009, LNCS 5963. Springer-Verlag Berlin Heidelberg, pp. 44 -57 (2010).

[**Pierret-Golbreich et al., 1989**] Pierret-Golbreich, C., Delouis, I., Scapin, D., Un outil d'acquisition et de représentation des tâches orienté-objet, INRIA Rocquencourt, Le Chesnay, FRANCE, (1989).

[**Planet, 2007**] Planet MDE, Official site: http://planetmde.org/index.php?option=com_xcombuilder&cat=Tool&Itemid=47, Sept, (2007).

[**Plihon et al., 1998**] Plihon V., Ralyté J., Benjamen A., Maiden N.A.M., Sutcliffe A., Dubois E., Heymans P.: A Reuse-Oriented Approach for the Construction of Scenario Based Methods. Proc. of the Int. Software Process Association's 5th Int. Conf. on Software Process (ICSP'98), Chicago, Illinois, US, (1998).

[**Potts, 1989**] Potts C.: A Generic Model for Representig Design Methods, Proceedings of the 11th International Conference on Software Engineering, (1989).

[**Prakash, 1994**] Prakash N.: A Process View of Methodologies, 6th Int. Conf. on Advanced Information Systems Engineering, CAISE'94, Springer Verlag, (1994).

[**Prakash, 1999**] Prakash N.: On Method Statics and Dynamics. Information Systems. Vol.34, No.8, pp. 613-637. (1999).

[**Prakash et al, 2003**] Prakash N., Gosain A.: Requirements driven data warehouse development. In Proc. of the 15th Conference on Advanced Informations Systems Engineering (CAISE'03), pp. 13-16, Velden – Australia, June 16-20 (2003).

[**Prat, 1997**] Prat N.: Goal formalisation and classification for requirements engineering, Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, June (1997).

[**Prat, 1999**] Prat N.: Réutilisation de la trace par apprentissage dans un environnement pour l'ingénierie des processus, thèse présenté à l'université de paris1, février (1999).

[**Prat et al, 2002**] Prat N., etand Akoka J.: From Uml to Rolap multidimensional databases using a pivot model. 18èmes Journées Bases de Données Avancées (BDA'02). Every, pp. 21-25, Octobre (2002).

[**Punter et al., 1996**] Punter H., Lemmen K.: The MEMA model: Towards a new approach for Method Engineering. Information and Software Technology, Vol. 38, No.4, pp. 295-305, (1996).

[**Quartel et al., 2004**] Quartel D., Dijkman R-M., van Sinderen M.: Methodological Support for Service-oriented Design with ISDL. Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04). New York, USA, (2004).

[**Ralyté, 1999**] Ralyté J., Reusing Scenario Based Approaches in Requirement Engineering Methods: CREWS Method Base. Proc. of the 10th Int. Workshop on Database and Expert Systems Applications (DEXA'99), 1st Int. Workshop on the RE Process - Innovative Techniques, Models, Tools to support the RE Process (REP'99), Florence - Italy, September (1999).

[**Ralyté, 2001**] Ralyté J.: Ingénierie des méthodes à base de composants, Thèse de Doctorat soutenue à l'université Paris I – Sorbonne, France, Janvier (2001).

[**Ralyté et al., 2001a**] Ralyté J., Rolland C.: An Assembly Process Model for Method Engineering. Proceedings of the 13th CAISE'01, Interlaken, Switzerland, (2001).

[**Ralyté et al., 2001b**] Ralyté J., Rolland C.: An Approach for Method Reengineering. Proc. of the 20th Int. Conf. on Conceptual Modeling (ER'01), Yokohama , Japan , November 2001. H. Kunii, S. Jajodia, A. Solvberg (Eds.), LNCS 2224, Springer-Verlag, pp.471-484 (2001).

[**Ralyté, 2002**] Ralyté J.: Requirements Definition for the Situational Method Engineering, IFIP TC8/WG8.1 Working Conference on Engineering Information Systems in the Internet Context, Kanazawa, Japan, (2002).

[**Ralyté et al., 2006**] Ralyté J., Backlund P., Kühn H., Jeusfeld M. (2006). Method Chunks for Interoperability. Proc. of the 25th Int. Conf. on Conceptual Modelling (ER'06). Tucson, Arizona, USA. LNCS 4215, Springer-verlag, pp. 339-353, November 6-9 (2006).

[**Rieu, 1999**] Rieu D.: Ingénierie des systèmes d'information: bases de données, bases de connaissances et méthodes de conception. Document d'Habilitation de Recherche. Institut National Polytechnique de Grenoble – INPG. Décembre, 08, (1999).

[**Robertson et al., 1999**] Robertson S., Robertson J.: Mastering the Requirements Process, Addison-Wesley Professional edition, (1999).

[**Rolland et al., 1994c**] Rolland C., Prakash N.: A Contextual Approach to modeling the Requirements Engineering Process, (SEKE'94), 6th International Conference on Software Engineering and Knowledge Engineering, Vilnius, Lithuania, (1994).

[**Rolland et al., 1995**] Rolland C., Souveyet C., Moreno M.: An Approach for Defining Ways-Of-Working, in the Information Systems Journal. Special issue: advanced information systems engineering, Volume 20, Issue 4, Elsevier, pp. 337-359, June (1995).

[**Rolland, 1998**] Rolland C.: A Comprehensive View of Process Engineering. Proceedings of the 10th International Conference CAiSE'98, B. Lecture Notes in Computer Science 1413, Springer Verlag Pernici, C. Thanos (Eds), Pisa, ITALY, June (1998).

[**Rolland et al., 1996a**] Rolland C., Plihon V.: Using Generic Chunks to Generate Process Models Fragments, Proceedings of 2nd IEEE International Conference on Requirements Engineering (ICRE'96), Colorado Spring. (1996).

[**Rolland et al., 1996b**] Rolland C, Prakash N.: A proposal for context specific method engineering. In Proc. of the IFIP WG8.1 Int. Conf. On Method Engineering, Chapman&Hall (Pub.), pp. 191-208. Atlanta-USA, August (1996).

[**Rolland et al., 1998a**] Rolland C., Ben Achour C., Cauvet C., Ralyté J., Sutcliffe A., Maiden N., Jarke M., Haumer P., Pohl K., Heymans P.: A Proposal for a Scenario Classification Framework. Requirements Engineering Journal 3 :1, (1998).

[**Rolland et al., 1998b**] Rolland C., Souveyet C., Ben Achour C.: Guiding Goal Modelling Using Scenarios. IEEE Transactions on Software Engineering, special issue on Scenario Management, Vol. 24, No. 12, 1055-1071, Dec. (1998).

[**Rolland et al., 1998d**] Rolland C., Plihon V., Ralyte J.: Specifying the reuse context of Scenario Method Chunks, In Proceedings of the 10th Conference on Advanced System Engineering (CAISE'98). Pisa Italy, 8-12 June, (1998).

[**Rolland et al., 1998e**] Rolland C., Ben Achour C.: Guiding the construction of textual use case specifications. Data & Knowledge Engineering Journal Vol. 25 N° 1, pp. 125-160, (ed. P. Chen, R.P. van de Riet) North Holland, Elsevier Science Publishers. March (1998).

[**Rolland et al., 1999**] Rolland C., Ralyté J., Plihon V.: Method Enhancement by Scenario Based Techniques. In Proc. of the 11th CAiSE'99, pp. 14-18, Heidelberg - Germany, (1999).

[**Rolland et al., 1999**] Rolland C., Prakash N., Benjamen A.: A Multi-Model Vew of Process Modelling, Requirements Engineering Journal (4), pp.169-187, (1999).

[**Rolland et al., 2000**] Rolland C., Prakash N.: Bridging the Gap Between Organisational Needs and ERP Functionality, Requirements Engineering Journal, Vol 5, number 3, pp. 180-193, Octobre (2000).

[**Rolland et al., 2001**] Rolland C., Salinesi C.: Ingénierie des systèmes d'information. Hermès. (2001).

[**Rolland, 2005**] Rolland C.: L'ingénierie des méthodes : une visite guidée, e-TI - la revue électronique des technologies d'information, 1, http://www.revue-eti.netdocument.php?id=726, (2005).

[**Rolland, 2007**] Rolland C.: Capturing System Intentionality with Maps, Conceptual Modeling in Information Systems Engineering, Springer-Verlag, p. 141-158. Berlin, Germany, (2007).

[**Rolland, 2008**] Rolland C., Method Engineering: Towards Methods as Services", In International Conference on Software Process (ICSE-ICSP), Springer-Verlag, Leipzig, Germany, Mai (2008).

[**Royce, 1970**] Royce W.: Managing the development of large software systems; Proceedings of the IEEE WESCON, August, (1970).

[**Saeki et al., 1993**] Saeki M., Iguchi K., Wen-yin K, Shinohara M: A meta-model for representing software specification & design methods. Proc. of the IFIP¨WG8.1 Conference on Information Systems Development Process, Come, pp 149-166, (1993).

[**Saeki et al., 1994**] Saeki M., Wen-yin K.: Specifying Software Specification and Design Methods. Proceedings of Conference on Advanced Information Systems Engineering, CAISE'94, Lecture Notes in Computer Science 811, Springer Verlag, pp. 353-366, Berlin, (1994).

[**Salvador et al., 1996**] Salvador T., Scholtz J., Larson J.: The Denver Model for Groupware Design. Journal, ACM Special Interest Group Computer-Human Interface bulletin (SIGCHI), 1996, volume 28, numéro 1, édition en ligne, ACM Press.

[**Seidewitz E., 2003**] Seidewitz E.: What models mean. IEEE Software, 20(5) :26–32, (2003).

[**Seligmann et al., 1989**] Seligmann P., Wijers G., Sol H.: Analyzing the structure of I. S. methodologies, an alternative approach. In Proceedings of the 1st Dutch Conference on Information Systems, Amersfoort, The Netherlands, (1989).

[**Shlaer et al., 1988**] Shlear S., Mellor S. : Object Oriented Systems Analysis. Prentice-Hall, (1988).

[**Smolander et al., 1991**] Smolander K., Lyytinen K., Tahvanainen V., Marttiin P. : MetaEdit - A Flexible Graphical Environment for Methodology Modelling. Proceedings of the 3th International Conference in Advanced Information Systems Engineering (CAISE'91), Trondheim, Norway, May (1991).

[**Song, 1995**] Song X.: A Framework for Understanding the Integration of Design Methodologies. In: ACM SIGSOFT Software Engineering Notes, Vol. 20, N°1, pp. 46-54, (1995).

[**Song, 1997**] Song X.: Systematic Integration of Design Methods, IEEE Software. (1997).

[**Tarby et al., 1996**] Tarby J-C., Barthet M-F.: The DIANE+ Method., pp. 95-119 dans Computer-Aided Design of User Interfaces. Namur: J. Vanderdonckt (Ed.), Presses Universitaires de Namur. (1996).

[**Tariq et al., 2005**] Tariq N., Akhter N.: Comparison of Model Driven Architecture (MDA) based tools, Karolinska University Hospital; A Thesis Document, pages 74. Sockholm, Sweden. June (2005).

[**Tawbi, 2001**] Tawbi M.: CREWS-L'Ecritoire : un Guidage Outille du Processus d'Ingénierie des Besoins, PhD thesis, University of Paris 1-Sorbonne, (2001).

[**UML, 2000**] Rational Software Corporation, Unified Modelling Language version 1.3. Available at http://www.rational.com/uml/resources/documentation/, (2000).

[**UPnP, 2010**] UPnP Forum, Universal plug and play standards, site official. Date de consultation : 10 février 2010 http://www.upnp.org/standardizeddcps/default.asp.

[**van Lamsweerde, 2000**] van Lamsweerde A.: Requirements Engineering in the year 2000 : A research perspective. 22nd International Conference on Software Engineering, Limerick, Ireland, (2000).

[**Vojtisek et al., 2004**] Vojtisek D., Jzquel J.-M.: MTL and Umlaut NG: Engine and Framework for Model Transformation. ERCIM News, Nro. 58, Special Issue on Automated Software Engineering, 42–45 (2004).

[**Vépa et al., 2006**] Vépa E., Bézivin J., Brunelière H., Jouault F.: Measuring Model Repositories. In: Proceedings of the Model Size Metrics Workshop at the MoDELS/UML 2006 conference, Lecture Notes In Computer Science, Genoava, Italy, Springer, (2006).

[**W3C, 2005**] World Wide Web Consortium. XSL transformations (XSLT) version 2.0, avril (2005).

[**W3C, 2007**] World Wide Web Consortium, http://www.w3.org/TR/2007/REC-xslt20-20070123/

[**W3C-SPARQL, 2009**] W3C-SPARQL : Query Language for RDF, W3C, W3C Recommendation, http://www3.org/TR/rdf-sparql-query/, date de consultation : November (2009).

[**Wellie, 2001**] Wellie M van: Task-based User Interface Design, PhD Thesis - Vrije Universiteit Amsterdam. (2001).

[**White et al., 2008**] White J., Gray J., Schmidt D.: Constraint-based Model Weaving. Transactions on Aspect-Oriented Software Development VI: Special Issue on Aspects and Model-Driven Engineering, Springer-Verlag, pp. 153-190, Heidelberg - Berlin (2008).

[**Wicks, 2006**] Wicks M.: Tool Integration within Software Engineering Environments: An Annotated Bibliography, Technical Report. Ref. HW-MACS-TR-0041, 4th August (2006).

[**Wistrand et al., 2004**] Wistrand K., Karlsson F.: Method components: Rationale revealed, in proceedings of CAISE 04, Springer-Verlag. Riga, Latvia, (2004).

[**Wynekoop et al., 1993**] Wynekoop J., Russo N.: System development methodologies: unanswered questions and the research practice gap. In Proceedings of the 14th Intl. Conf. Inf. Syst., New York, ACM Pub. pp 181- 190, (1993).

[**Yu et al., 1994**] Yu E., Mylopoulos J.: *From ER to AR Modelling Strategic Actor Relationships for Business Process Reengineering*. In Proc. of the 13th International Conference on the Entity-Relatioship Approach (ER'94). Manchester (UK), December 13-16, (1994).

[**Yu et al., 1994**] Yu E., Mylopoulos J.: Using goals, rules and methods to support reasoning in Business Process Reengineering, 27th Hawaii International Conference on System Sciences, Maui, Hawaii, (1994).

[**Yu, 1997**] Yu E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97). Washington D.C., USA. pp. 226-235. (1997).

[**Yu et al., 2008**] Yu J., Lalanda P., Chollet S.: Development Tool for Service-Oriented Applications in Smart Homes. In: Proc. of SCC'08, pp. 239-246. DC USA, (2008).

[**Yue,1987**] Yue K.: What Does It Mean to Say that a Specification is Complete?, Proc. IWSSD-4, Fourth International Workshop on Software Specification and Design, Monterey, (1987).

ANNEXE A. LA METHODE SYMPHONY

Symphony est une méthode de développement élaborée au sein de la société Umanis (http://www.umanis.com/fr) et formalisée au sein de l'équipe SIGMA du laboratoire LSR par Ibtissem Hassine [Hassine *et al.*, 2002]. Elle s'appuie sur les bonnes pratiques de développement orienté objet dont l'objectif majeur est de satisfaire le client en répondant aux exigences fixées par le cahier des charges et au développement rapide des applications. Cette méthode s'appuie sur le langage unifie UML et est construite autour d'un certain nombre de pratiques. Symphony est itérative, orientée utilisateur, pilotée par les cas d'utilisation, s'appuie sur le concept d'objet métier, et adopte un modèle de cycle de vie en Y.

a) Cycle de vie en Y : séparation des aspects fonctionnels et des aspects techniques

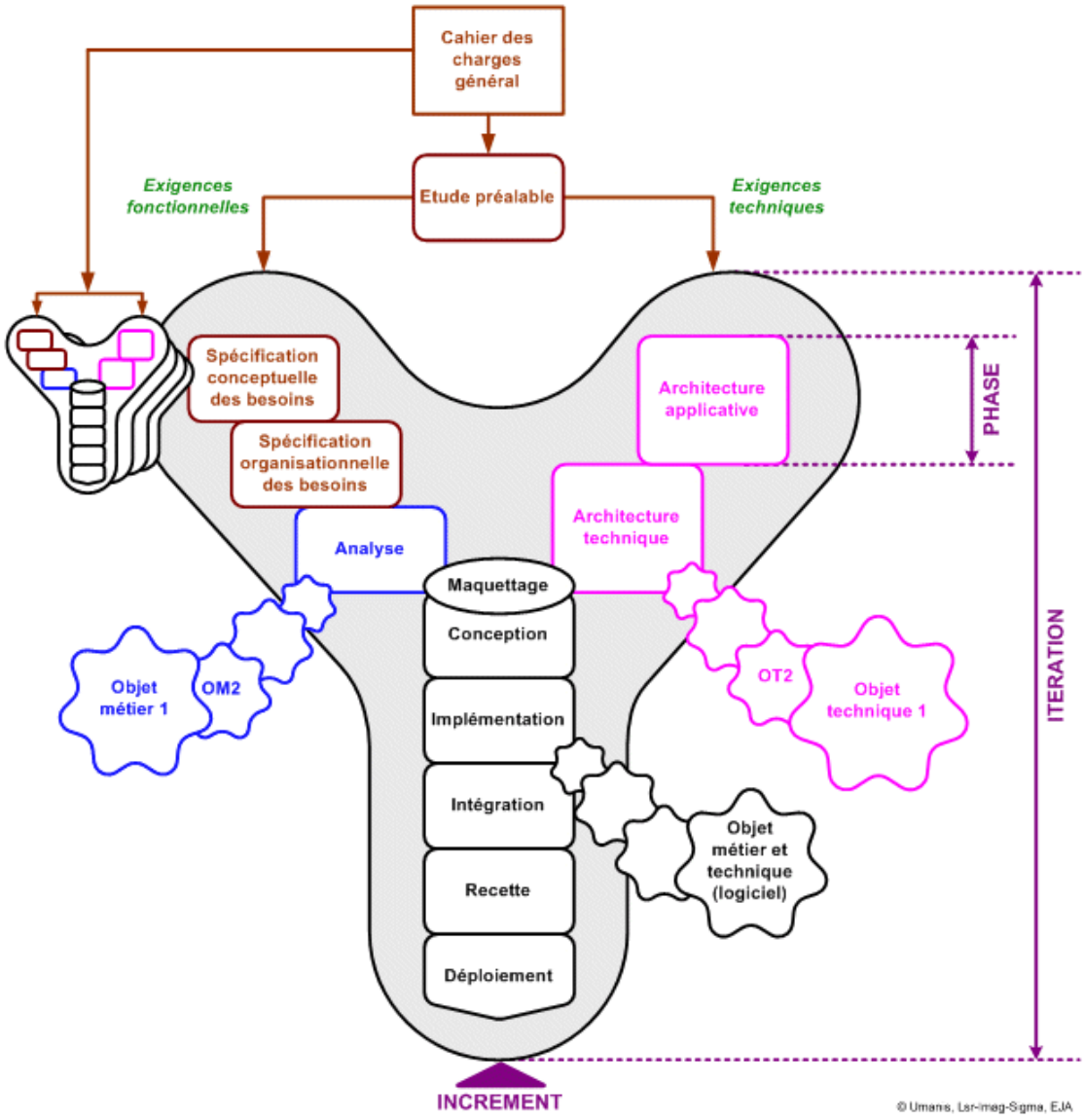


FIGURE 94. CYCLE DE VIE EN Y DE LA DEMARCHE SYMPHONY

Symphony sépare l'étude des besoins fonctionnels de celle des besoins techniques et ceci des le début du cycle de vie des applications. Les deux aspects fonctionnels et techniques sont mis en évidence en adoptant un modèle de cycle de vie en Y [Andre, 1994] [Larvet, 1994] [Rocques et al., 2004] (Cf. Figure 94).

Ce modèle de cycle de vie s'articule autour de trois branches : la branche gauche capitalise la connaissance du métier, la branche droite capitalise un savoir-faire technique et la branche commune consiste à fusionner les résultats des deux branches gauche et droite afin d'assurer la réalisation du système.

b) Une démarche basée sur la modélisation UML

De nombreux formalismes ont été proposés dans le contexte de la conception des systèmes d'information, comme les formalismes orientés objet, en particulier UML (Unified Modelling Langage) [Muller, 1998].

Symphony est une méthode basée sur la modélisation UML, motivée par le fait que ce langage propose un ensemble de modèles adaptés aux différentes phases de développement. Ses différentes représentations graphiques standardisées et synthétiques facilitent les échanges au sein de l'équipe de développement ainsi que l'élaboration de la documentation des systèmes.

c) Une démarche itérative

La démarche Symphony préconise un développement itératif (Cf. Figure 95) dans lequel le développement s'organise à travers un ensemble de cycles de développement en Y (premiers concepts apparus sous le nom de développement en spirale et de développement évolutif par B.W. Boehm [Boehm, 1988]).

Chaque cycle de développement, appelé aussi itération, est centré sur un processus métier ou le raffinement d'un processus métier identifie, décrit et pondère lors de la phase d'étude préalable. Cette phase, précédant le processus de développement, consiste à faire une première reformulation et expression des besoins fonctionnels sous forme de processus métier.

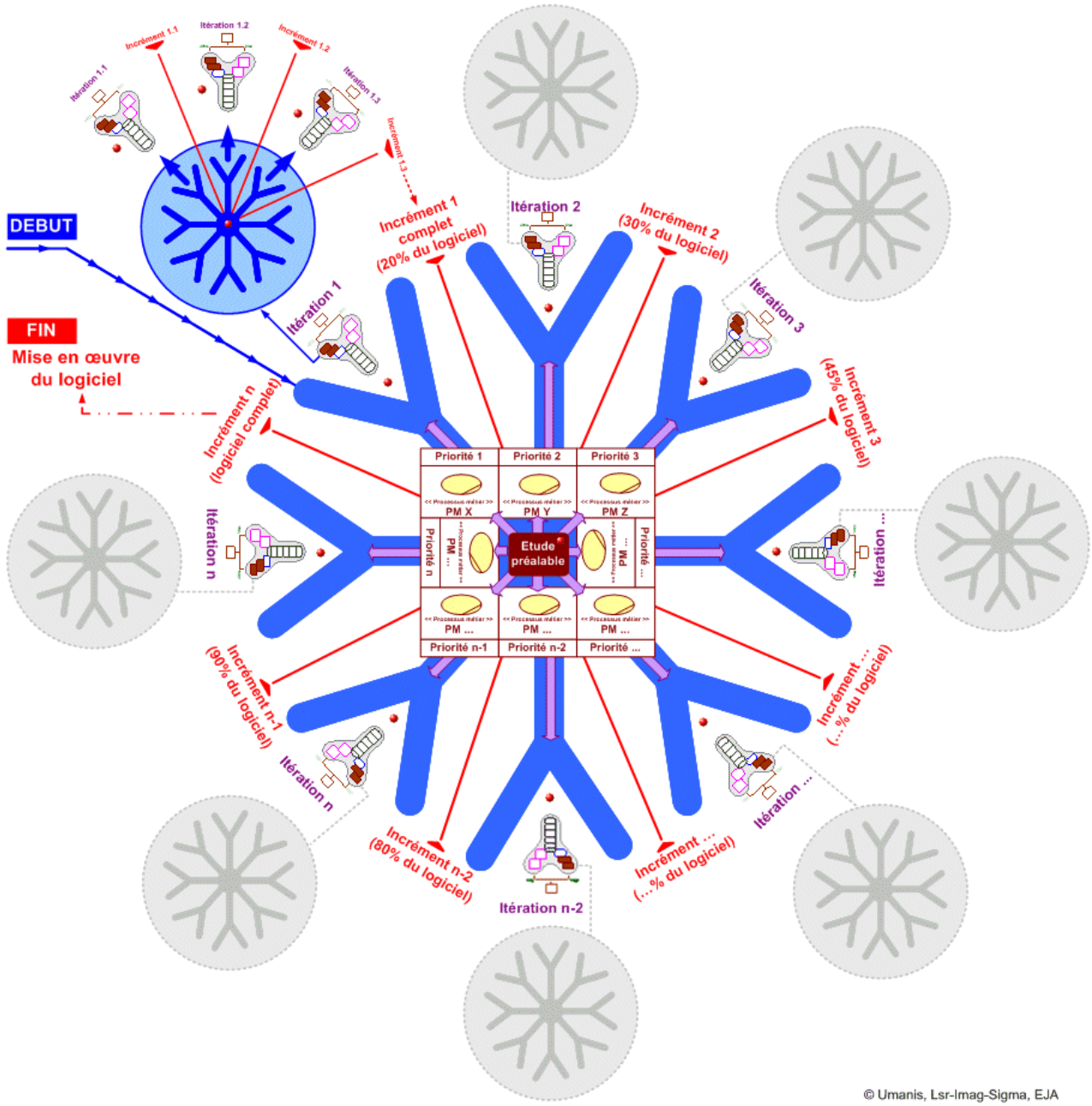


FIGURE 95. PROCESSUS ITERATIF DE SYMPHONY : MODELE EN FLOCONS

d) Une démarche pilotée par les cas d'utilisation

La démarche est orientée utilisateur et pilotée par les cas d'utilisation. En effet, elle intègre les besoins et les usages réels des utilisateurs des les phases amont du développement. Ces besoins sont alors modélisés par des cas d'utilisation qui assurent la cohésion des activités et guident le processus de développement dans son ensemble.

Les modèles de cas d'utilisation décrivent les fonctionnalités complètes du système. A partir de ces modèles, les concepteurs créent une série de modèles de conception et de développement réalisant les cas d'utilisation. Les testeurs vérifient si les composants métier logiciels développés respectent

correctement les cas d'utilisation. Les besoins des utilisateurs sont donc prioritairement traités dans la branche gauche du modèle en Y. Les cas d'utilisation constituent un mécanisme essentiel de traçabilité entre modèles (Cf. Figure 96).

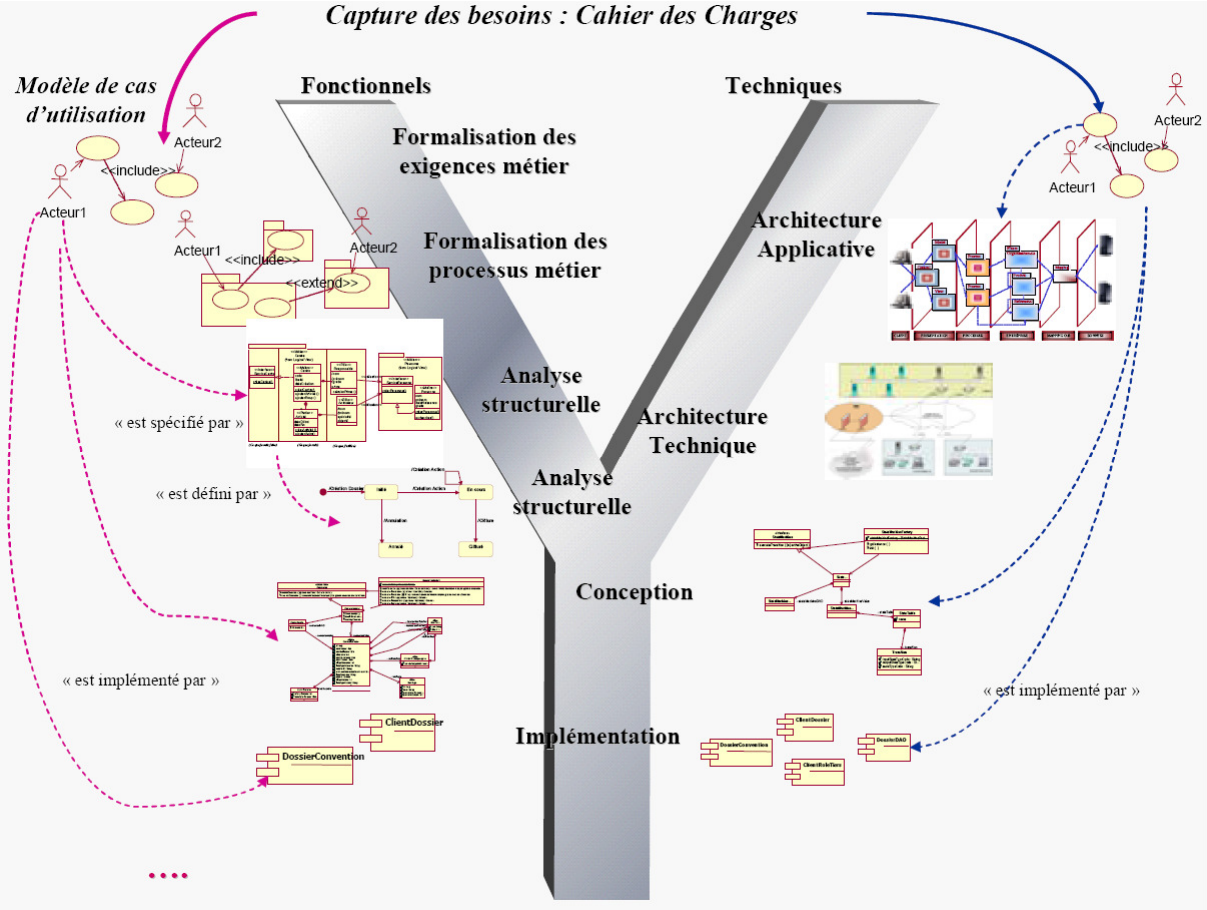


FIGURE 96. MECANISME DE TRAÇABILITE SUR LES CAS D'UTILISATION

e) Une démarche orientée composant métier

La démarche est orientée composant métier car l'application est vue, tant au niveau conceptuel que logiciel, comme un assemblage d'objets métier indépendants et interconnectes. Cette pratique garantit une bonne modularité des spécifications et facilite la réutilisation.

La Figure 96 met en évidence que les composants métier (resp. composants techniques) sont identifies et analyses dans la branche fonctionnelle (resp. technique). Dans la branche centrale du Y, les composants métier conceptuels sont progressivement transformes en composants métier logiciels par intégration des choix techniques spécifiés dans les composants techniques.

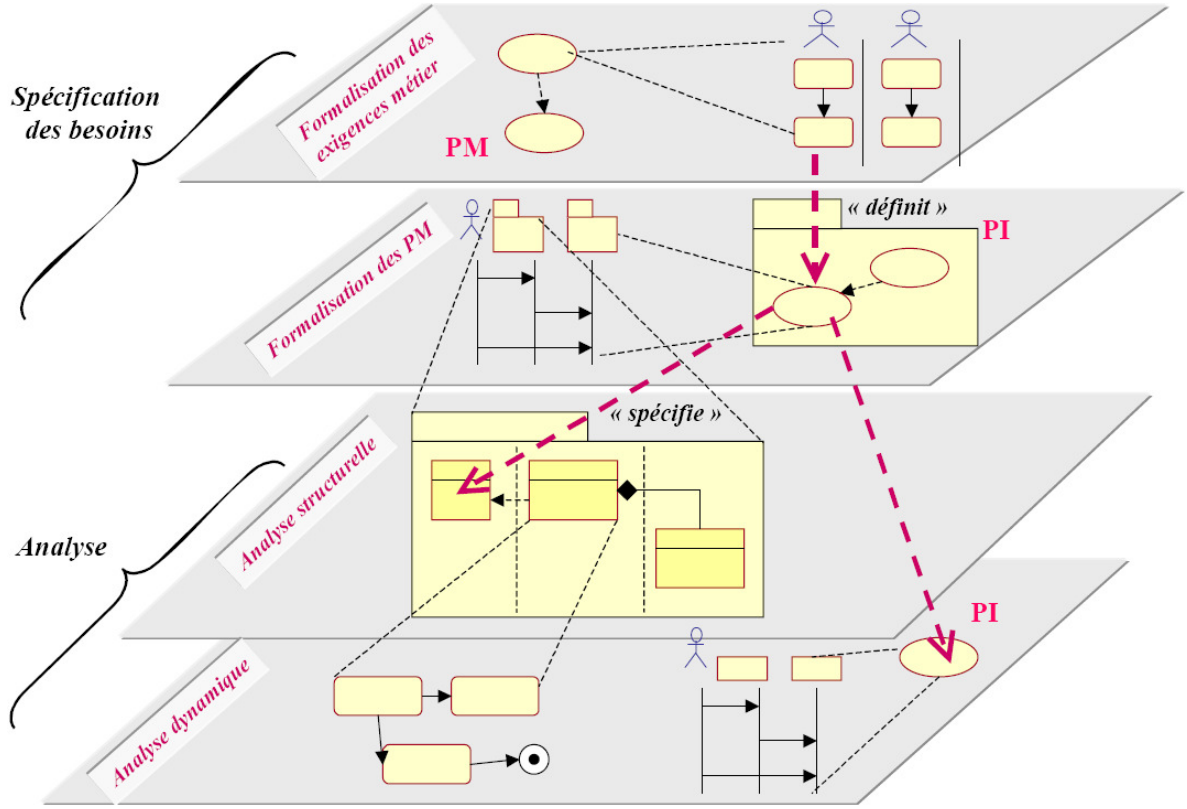


FIGURE 97. TRAÇABILITE PAR LES CAS D'UTILISATION DANS LA BRANCHE GAUCHE DE SYMPHONY

La démarche est basée sur un modèle de composants métier qui permet dès la phase de formalisation des exigences métier, l'identification des composants métier et de leur responsabilité dans le système d'information cible. Un composant métier représente un concept ou un processus du système d'information répondant à des besoins purement fonctionnels. Chaque composant métier est spécifié par une structure tri partie mettant en évidence les services qu'il offre, sa structure interne et les composants auxquels il est lié.

Un composant métier pour un acteur extérieur correspond à une entité qu'il peut manipuler. Cette entité est apte à exécuter des opérations et maîtrise des informations qui lui permettent d'assumer des opérations qu'elle s'engage à réaliser. Les informations forment un réseau de concepts qui a un sens pour le métier.

Le modèle métier de la démarche spécifie trois types de composant métier :

- les composants « processus » permettent de décrire un processus applicatif (par exemple le processus Gestion des Agences, le processus Gestion du Personnel, etc.),
- les composantes « entités » constituent la base des composants métier (par exemple des personnes, des agences, des points de vente, etc.),

- les composants « données » représentent les données de références (par exemple la codification des salaires).

De même que la démarche est basée sur les composants métier, elle l'est aussi sur les composants techniques. Ces derniers représentent des composants non-fonctionnels permettant de définir l'architecture technique de l'application et proposant des solutions à des problèmes techniques récurrents tels que la communication réseau, les connexions d'unités physiques, la persistance, etc.

ANNEXE B. L'ONTOLOGIE DES BUTS

Dans le contexte du méta-modèle SO2M [Guzélian, 2007], les services méthode fournissent des processus et des solutions à des problèmes d'ingénierie des SI. Leur description utilise des ontologies de tâches relatives au domaine de l'ingénierie des SI. Ces ontologies expriment toute la connaissance sur les problèmes de l'ingénierie des SI. Cette annexe présente « **l'ontologie de buts (Lbut)** », une ontologie de tâches qui définit un vocabulaire sur les problèmes d'ingénierie des SI, c'est-à-dire les problèmes qui tracent le développement des SI.

L'ontologie des buts est représentée par un but qui peut être élémentaire ou complexe. Un **but complexe** est composé de buts élémentaires. La relation de composition entre buts exprime une conjonction de buts. La description des **buts élémentaires** est basée sur la formalisation proposée dans [Prat, 1997]. Dans cette formalisation un but comporte un verbe et un objet.

Les verbes expriment des actions plus ou moins complexes de développement et les objets sont des éléments de produit sur lesquels portent les actions. Par exemple, le but : « Construire un modèle d'analyse » est défini par un verbe « Construire » et un objet « un modèle d'analyse ».

L'ontologie **L**but utilise l'ontologie des produits (voir Annexe E) pour définir **l'objet** ou une partie de l'objet d'un but.

En ce qui concerne le **verbe** du but, l'ontologie propose une classification issue des types d'activités que les concepteurs/développeurs mettent en œuvre durant le développement. Les activités usuelles sont l'acquisition de connaissances, la prise de décision, la construction de modèles, la documentation, ….

La figure ci-dessous illustre le méta-modèle de l'ontologie de Buts (**L**but) proposée par Guzélian [Guzélian, 2007].

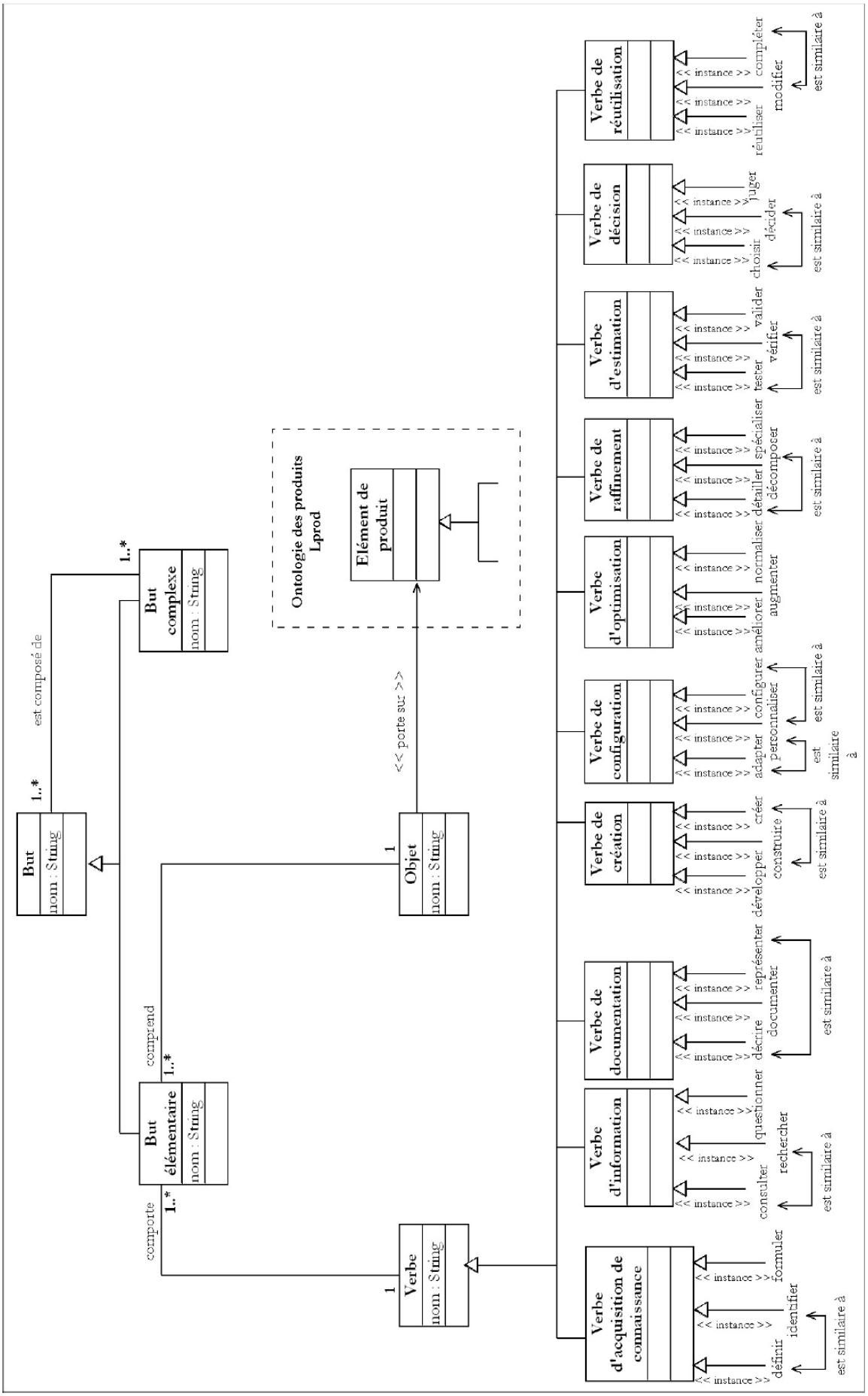


FIGURE 98. L'ONTOLOGIE DES BUTS [GUZELIAN, 2007]

ANNEXE C. L'ONTOLOGIE DES ACTEURS

L'ontologie des acteurs Lact définit un ensemble de rôles pour les acteurs qui ont en charge les problèmes d'ingénierie. Ces rôles correspondent à des fonctions occupées par les acteurs dans le développement de tout système d'information. Les instances de cette ontologie sont utilisées pour indiquer les acteurs concernés par l'utilisation du service.

L'ontologie des acteurs est inspirée des types d'acteurs définis dans le processus unifié associé à UML [Kruchten, 2000]. Cette ontologie est composée de 6 types d'acteurs. Ces acteurs peuvent être des concepteurs, des développeurs, des testeurs, des superviseurs, des architectes, ou des administrateurs. La figure ci-dessous montre l'ontologie des acteurs (**L**act) proposée par Guzélian [Guzélian, 2007].

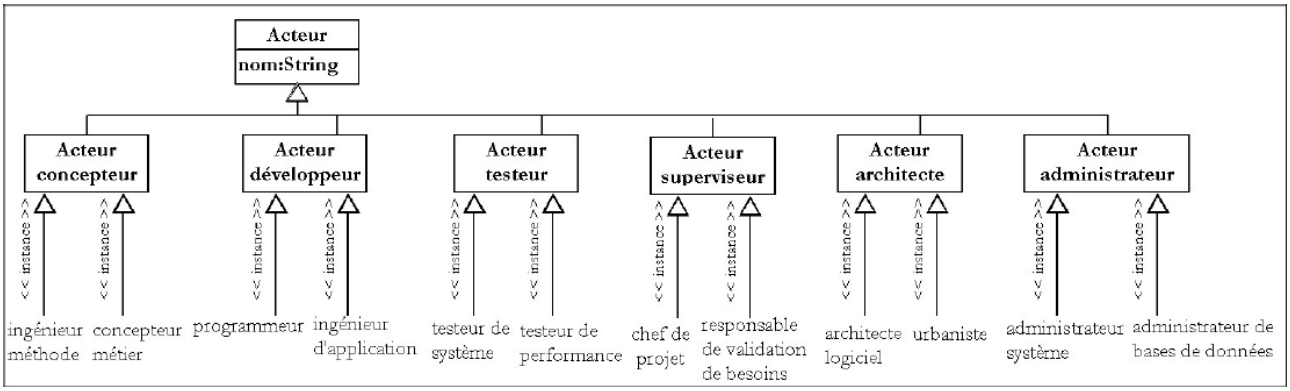


FIGURE 99. L'ONTOLOGIE DES ACTEURS [GUZELIAN, 2007]

ANNEXE D. L'ONTOLOGIE DES PROCESSUS

L'ontologie des processus Lproc définit une terminologie pour décrire les processus de développement. Elle propose une classification des éléments intervenant dans la description de processus. Cette classification comporte trois types d'éléments : Les unités de processus, la construction de contrôle et les artefacts de processus.

- **Les unités de processus** correspondent à des ensembles de tâches plus ou moins complexes. Elles peuvent être mises en œuvre par des acteurs (définis avec l'ontologie d'acteurs **L**act). Elles permettent d'atteindre un but (défini avec l'ontologie des buts **L**but). Les unités de processus correspondent soit à des unités élémentaires (par exemple : activités, opérations, actions ...) soit à des unités complexes. Les unités complexes sont constituées de plusieurs unités élémentaires, elles fournissent un mécanisme de structuration du processus. Trois types d'unités complexes sont proposés :

    - **Les stratégies** correspondent aux approches de développement usuelles utilisées en ingénierie des SI. Par exemple, le développement en cascade, le développement itératif, le développement en spirale... sont considérés comme des stratégies.
    - **Les phases** correspondent à un regroupement temporel des unités de processus ; par exemple, dans le processus unifié [Jacobson *et al.*, 1999], l'initialisation, l'élaboration, la construction, et la transition sont considérées comme des phases.
    - **Les disciplines** correspondent à un regroupement d'unités qui portent sur un même élément de produit ; par exemple, dans un processus de conception de bases de données, il est usuel de définir trois disciplines portant chacune sur un niveau de description particulier de la base de données : le niveau conceptuel, le niveau logique et le niveau physique.

- **Les constructions de contrôle** permettent d'organiser les unités de processus. L'ontologie fournit un ensemble de constructions de contrôle telles que : la séquence, le branchement conditionnel...
- **Les artéfacts de processus** précisent la situation initiale et la situation finale d'une unité de processus. Un artéfact de processus peut aussi être un artéfact utilisé ou généré par une unité de processus. Les artéfacts de processus peuvent être définis avec l'ontologie des produits. Par exemple, l'activité « Identifier les besoins non fonctionnels » permet de générer une liste de besoins non fonctionnels. Cette liste correspond à un artéfact de type document dans l'ontologie des produits.

La figure ci-dessous montre l'ontologie des processus (**L**proc) proposée par Guzélian [Guzélian, 2007].

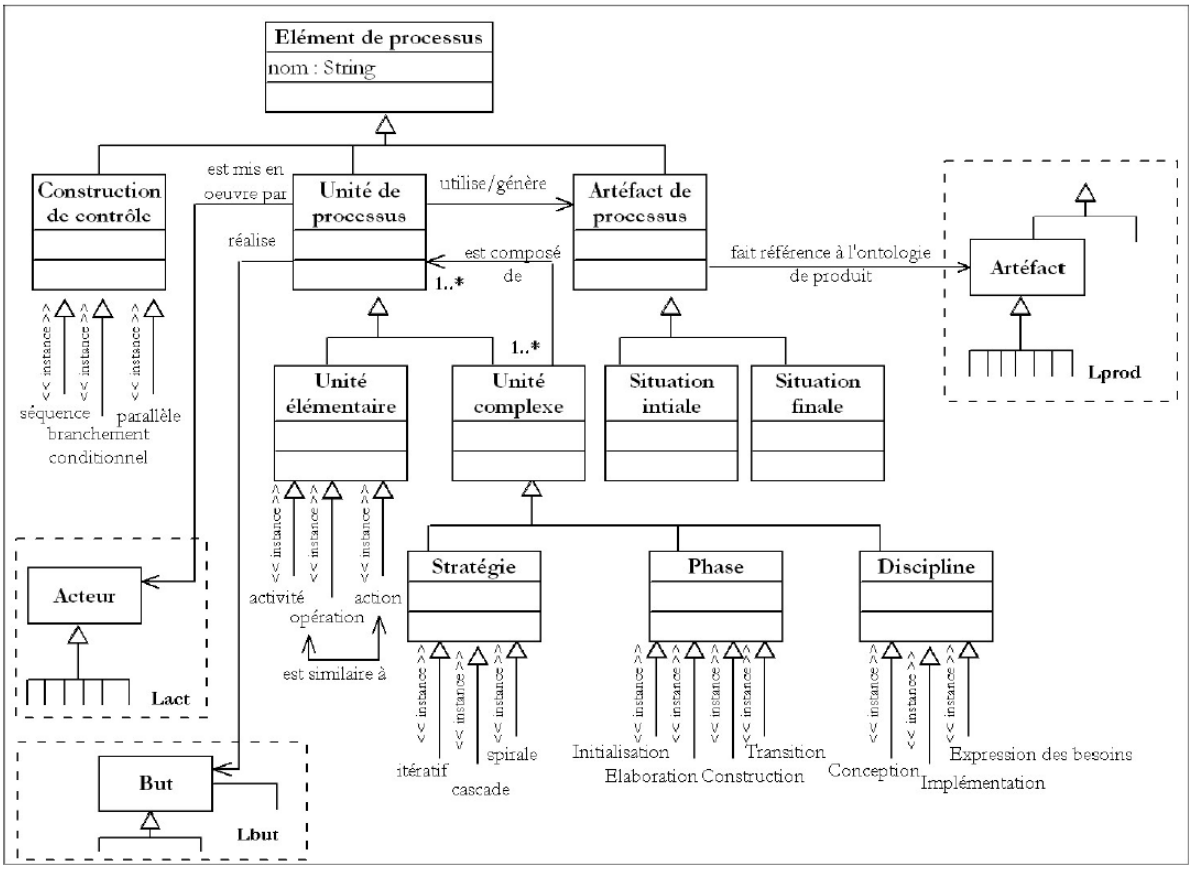


FIGURE 100. L'ONTOLOGIE DES PROCESSUS [GUZELIAN, 2007]

ANNEXE E. L'ONTOLOGIE DES PRODUITS

L'ontologie de Produit (**Lprod**) [Guzélian, 2007] décrit les différents objets qui peuvent être utilisés ou produits au cours du développement d'un système d'information. Elle définit une classification des éléments de produit. Cette classification comporte deux types d'éléments :

- **Les artéfacts**, un artéfact correspond à un objet élaboré, modifié ou utilisé par une unité de processus au cours d'un projet de développement. Pour définir la classification d'un artéfact, nous avons adapté et étendu celle proposée dans le processus unifié [Jacobson et al., 1999]. Dans l'ontologie des produits, les types d'artéfacts retenus sont : les graphes, les méta-modèles, les modèles, les diagrammes, les documents, les codes sources, et les pseudo-codes.
  La notion d'artéfact s'applique à la fois à des unités d'information produites ou modifiées pendant le processus et aussi à des éléments « externes au processus » mais utilisés par le processus. Par exemple, dans le processus unifié, il est usuel de produire et de manipuler des diagrammes et des modèles, d'utiliser le langage UML, de générer des classes en Java et d'élaborer des documents de type glossaire.
  Des éléments de base, correspondent à des notions élémentaires qui sont utilisées dans la définition des artéfacts. On distingue trois types d'éléments de base : des concepts, des contraintes et des liens. Par exemple, un artéfact qui correspond à un diagramme d'activités peut contenir des activités (i.e. des concepts), des liens de séquence (i.e. Des liens) et des conditions de branchement (i.e. des contraintes).

La figure ci-dessous montre l'ontologie des produits (**L**prod) proposée par Guzélian [Guzélian, 2007].

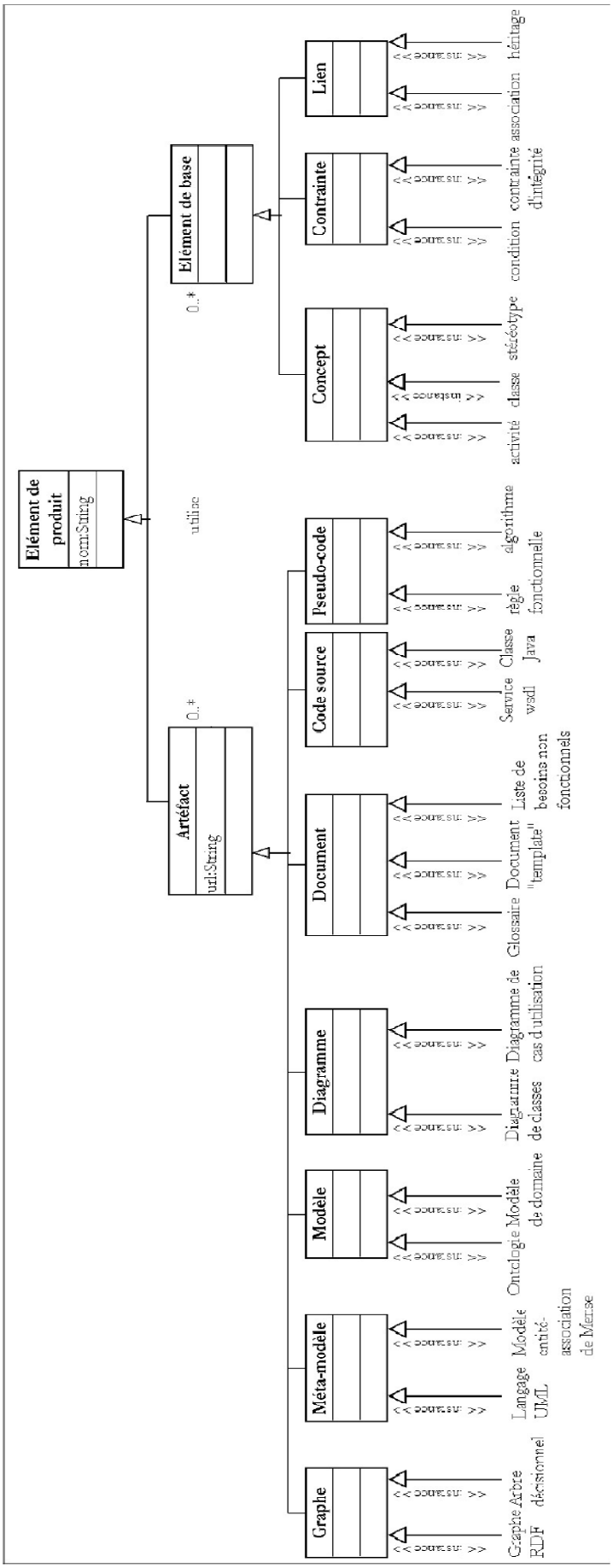


FIGURE 101. L'ONTOLOGIE DES PRODUITS [GUZELIAN, 2007]

ANNEXE F. LE PATRON ITEM-DESCRIPTION

Acteur : Peter COAD [Coad, 1992]

Intention : Ce patron est utilisé lorsque certaines propriétés peuvent s'appliquer à plus d'un objet. Il permet de factoriser des propriétés communes à plusieurs objets.

Indication d'utilisation : Ce patron permet de gérer des objets et leurs propriétés qui sont communes à d'autres objets.

Structure :

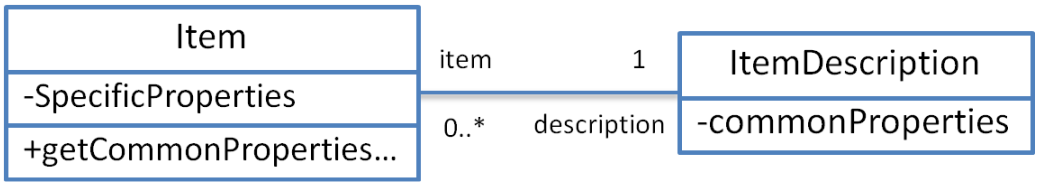


FIGURE 102. STRUCTURE DU PATRON ITEM-DESCRIPTION [COAD, 1992]

Constituants :

Le patron Item-Description propose deux classes : **Item** et **ItemDescription**. La classe **ItemDescription** détient les valeurs des attributs qui peuvent s'appliquer à plus d'un objet de type **Item**. La classe **Item** détient les valeurs de ses propres attributs et diverses méthodes de consultation pour accéder aux valeurs des attributs de la classe **ItemDescription**.

ANNEXE G. LE PATRON TYPE-OBJECT

Acteur : Ralph JOHNSON, Bobby WOOLF [Johnson et *al.*, 1997]

Intention : Ce patron est utilisé lorsque : i) les instances d'une classe doivent être groupées selon des attributs et/ou comportements communs. ii) Une classe doit être spécialisée pour chaque groupe, afin d'implémenter les attributs/comportements communs de ce groupe.

Indication d'utilisation : Ce patron permet de gérer des objets et leurs propriétés qui sont communes à d'autres objets. Les propriétés communes des objets sont séparées des propriétés spécifiques de chaque objet.

Structure :

FIGURE 103. STRUCTURE DU PATRON TYPE-OBJET [JOHNSON ET AL., 1992]

Constituants :

Le patron est composé de deux classes concrètes : **TypeClass** et **Class.** La classe **Class** permet d'instancier des objets, et **TypeClass** permet d'instancier le type des objets. Chaque objet instance de **Class** pointe vers son type.

Avantage :

Les avantages du patron Type-Objet sont :

- La possibilité d'ajouter dynamiquement autant d'objets instances de TypeClass que l'on veut, sans modifier la structure du programme.
- La non prolifération de sous-classes.
- La séparation entre l'objet et son type est cachée au client.
- Le changement dynamique du type d'un objet.
- Un objet peut avoir plusieurs types (variante du patron sur la cardinalité du rôle type à 0..*).

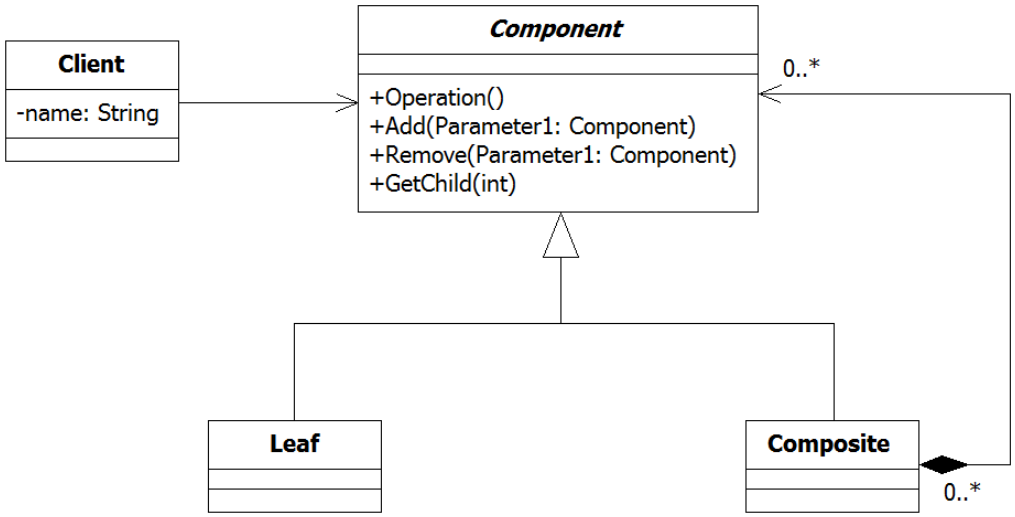
ANNEXE H. LE PATRON COMPOSITE

Acteur : Erich GAMMA, Richard HELM, Ralph JOHNSON, John VLISSIDES [Gamma et *al.*, 1995]

Intention : Le modèle composite compose des objets en des structures arborescentes pour représenter des hiérarchies composant/composé. Il permet au client de traiter de la même et unique façon les objets individuels et les combinaisons de ceux-ci.

Indication d'utilisation : On utilisera le modèle Composite lorsque : (i) On souhaite représenter de hiérarchies de l'individu â l'ensemble, (ii) On souhaite que le client n'ait pas â se préoccuper de la différence entre combinaisons d'objets individuels. (iii) Les clients pourront traiter de façon uniforme tous les objets de la structure composite.

Structure :



Constituants :

Composant : Le composant déclare l'interface des objets entrant dans la composition. Il implémente le comportement par défaut qui convient pour l'interface commune â toutes les classes. Il déclare une interface pour accéder à ses composants enfants et les gérer. Eventuellement, il définit une interface pour accéder à un parent du composant dans une structure récursive, et l'implémente si besoin est.

Feuille : La feuille représente des objets feuille dans la composition. Une feuille n'a pas d'enfants. Elle définit le comportement d'objets primitifs dans la composition.

Composite : Le Composite définit le comportement des composants dotés d'enfants. Il stocke les composants enfants. Il implémente les opérations liées aux enfants dans l'interface Composant.

Client : Le client manipule les objets de la composition à l'aide de l'interface Composant.

Approche Orientée Services pour la Réutilisation de Processus et d'Outils de Modélisation

Résumé : Une méthode de développement de SI permet de modéliser des produits (tout artefact logiciel) en exécutant des modèles de processus. Les concepteurs sont assistés dans leurs tâches par des outils de modélisation.

L'ingénierie des méthodes permet de définir des méthodes de développement en réutilisant et en assemblant des fragments de méthodes existantes. Il s'agit alors de fournir à tout concepteur l'environnement de modélisation adapté à ses nouvelles activités.

L'objectif de nos recherches, est de faciliter le travail des chefs de projets et des concepteurs de modèles en les aidant dans le choix des fragments de méthodes, des modèles et des environnements de modélisation adaptés à leurs besoins. Les contributions de cette thèse s'articulent autour d'une architecture à base de services reposant sur trois niveaux d'abstraction.

Le niveau intentionnel repose sur un modèle de services permettant d'expliciter les besoins des concepteurs et des chefs de projets en termes de gestion de modèles. L'accent est mis sur la modélisation des buts avec l'utilisation d'ontologies issues du domaine des systèmes d'information. Le niveau organisationnel propose un modèle de services facilitant la capitalisation et la sélection de fragments de méthodes réalisant les buts des concepteurs. Le niveau opérationnel s'appuie sur un modèle de services permettant de caractériser et de sélectionner les outils de modélisation adaptés aux fragments de méthodes.

L'ensemble des propositions est accompagné de cas d'études et d'un prototype servant de support et de validation aux travaux réalisés.

Mots-clefs : Modèle, service, gestion de modèles, outils de modélisation.

A Service Oriented Approach for Reuse Process and Modelling Tools

Abstract: An Information System development methodology allows designers to model products (any software artifact) by executing process models. Designers are assisted in their tasks by modeling tools.

Methods engineering permits defining development methods by reusing and assembling fragments of existing methods. The point is then to provide to designers the modeling environment adapted to his/her activities.

The aim of our research is to facilitate the work of project managers and model designers by helping them in choosing methods fragments, models and modeling environments adapted to their specific needs. The contributions of this thesis concern a service-oriented architecture based on three different abstract levels.

The intentional level represents the needs of model designers and project managers in terms of the models management. The emphasis is placed on the modeling goals with the use of ontologies of the information systems the domain. The organizational level proposes a model of services facilitating the capitalization and the selection of method fragments which realize the goals of the designers. The operational level relies on a model of services allowing to characterize and to select the appropriate modelling tools adapted to the methods fragment.

All the proposals are illustrated by study cases and a first version of prototype for supporting and for validating the realized works.

Keywords: Model, service, model management, modeling tools.