Gestion dynamique des règles métiers dans les systèmes d'information dédiés à la conception collaborative

Mohsen Sadeghi

INSTITUT POLYTECHNIQUE DE GRENOBLE

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

T H E S E

pour obtenir le grade de

DOCTEUR DE L'Institut polytechnique de Grenoble

Spécialité : *« Génie Industriel »*

Préparée au laboratoire : Laboratoire des Sciences pour la Conception, l'Optimisation et la Production (G-SCOP)

dans le cadre de ***l'Ecole Doctorale***
«Ingénierie - Matériaux, Mécanique, Environnement, Energétique, Procédés, Production (I-MEP2) »

présentée et soutenue publiquement

par

Mohsen SADEGHI

Le 21/10/2008

Gestion dynamique des règles métiers dans les systèmes d'information dédiés à la conception collaborative

Directeur de thèse :
Frédéric NÖEL

Co-directeur de these :
Khaled HADJ-HAMOU

JURY:

M. Lionel ROUCOULES	Professeur, ENSAM Aix-en-Provence	, Président
M. Jean Pierre NADEAU	Professeur, ENSAM Bordeaux	, Rapporteur
M. Mickaël GARDONI	Professeur, INSA de Strasbourg	, Rapporteur
M. Frédéric NOËL	Professeur, INP Grenoble	, Directeur
M. Khaled HADJ-HAMOU	Maître de conférences, INP Grenoble	, Co-encadrant
M. Chris MCMAHON	Professeur, University of Bath	, Examinateur
M. Ali SIADAT	Maître de conférences, ENSAM Metz	, Examinateur

Remerciements

Tout d’abord, je tiens à remercier les membres du jury pour avoir accepté d’évaluer les résultats de mon travail de recherche. Merci à Lionel ROUCOULES de m’avoir fait l’honneur d’être le président du jury. Je remercie Jean Pierre NADEAU et Mickaël GARDONI, les rapporteurs de ces travaux qui, grâce à leurs points de vues critiques et constructifs, ont su me faire prendre du recul sur mon travail. Je remercie également Ali SIADAT et Chris MCMAHON pour avoir accepté de participer à ce jury.

Parmi eux, je tiens à remercier sincèrement Frédéric NOËL et Khaled HADJ-HAMOU, qui sont à l’initiative de ces travaux, tant pour leurs enseignements scientifiques que pour leurs qualités humaines. Sans excès de modestie, je pense, Frédéric et Khaled, que vous avez fortement contribué, chacun à votre manière et avec des points de vues complémentaires, au succès de ces travaux.

Je remercie également tous les membres du laboratoire G-SCOP et par la même occasion ceux du feu laboratoire 3S pour l’ambiance chaleureuse qui n’en reste pas moins scientifique qu’ils parviennent à créer au quotidien. Je garderai un excellent souvenir de nos échanges.

Je tiens à dire un grand merci à mes amies fidèles qui sont toujours là pour me soutenir.

Je termine par une profonde pensée à mes proches et ma famille pour leur soutien.

Résumé

Ce mémoire contribue à la compréhension des problématiques liées à la cohérence des modèles experts dans un contexte de conception coopérative. Dès lors qu'au moins deux concepteurs coopèrent à un niveau technique de conception, ils partagent de nombreux objets, paramètres, informations ou morceaux de modèles. Cependant ils ne peuvent travailler toujours simultanément et des phases de travail asynchrones sont indispensables. L'objectif de notre travail est de spécifier les mécanismes nécessaires pour mettre en œuvre des règles métier répondant aux cohérences techniques entre concepteurs. On peut imaginer lors de ces phases asynchrones un serveur espionnant l'évolution des modèles de chaque concepteur. Nous définissons des modèles de règles métiers et des mécanismes qui permettent d'évaluer si deux études parallèles divergent au delà de la simple comparaison syntaxique des modèles. Cette mesure de la divergence est alors est utilisé pour notifier en temps voulu les acteurs de la nécessité de provoquer une session de synchronisation et de négociation. Nous proposons des solutions originales pour assister ces activités de conception coopérative. Elles sont basées sur l'utilisation d'un modèle de partage qui est simultanément versionné par les différents collaborateurs. Ces versions sont analysées par des mécanismes syntaxiques et sur la base de contraintes métiers. Ces mécanismes permettent la mesure des évolutions parallèles des versions de modèles afin de communiquer aux différents concepteurs l'état de cohérence de l'environnement de conception et la nécessité de provoquer des tâches collaboratives. Au delà de cette proposition théorique, nous avons développé un démonstrateur qui valide une part des propositions avancées

Mots-clés : conception coopérative, environnement coopératif, modélisation en conception, modèle produit, partage d'information, modèle partagé, cohérence, contraintes métiers, règles métiers, synchronisation, méta-règles, notification

Abstract

This thesis contributes to understand issues related to the consistency of models during collaborative product development. When, two designers work cooperatively to define solutions at the technical level of product design, they share many objects, parameters, or required pieces of information to create a common understanding of design intents. Because of timeframe issues, it may be difficult to schedule simultaneous cooperation between designers, so the asynchronous cooperation is mandatory to allow designers to reflect over different viewpoints, to check incoherencies and to take time to prepare a solution. The asynchronous modification and the maintenance of shared model requires the cooperation of several designers. It is important to find out how shared model evolves during design process, how and when change processes take place and when designers should synchronize the models or should take project review time. The incoherencies need to be detected and to be solved during synchronization process. The coherency maintenance is provided by the relationships between entities of the model and associated constraint control mechanisms. If it is quite easy to compare the differences between two homogeneous models, it is more complicate to extract conflicts due to technical incompatibilities. The objective of our work is 1) to extend the simple comparison method with business rules to take into account expertise knowledge in the conflict detection, 2) to provide a measure of the conflict in order to assist the decision about when cooperation will be required to solve the conflicts. We analyse a range of existing constraints in collaborative product development process and we propose a quite generic model for business rules, which can be used to address conflict management issues. We demonstrate that technical rules were not enough to efficiently measure when conflicts require a synchronisation to ensure the project convergence: organisation rules must be used. Thus we present a model involving the concept of meta-rules to assist this decision. It determines a control mechanism for model evolutions and indicates the cost of changes and their impacts in model evolutions to determine how and when change processes take place (time of models synchronization).

Keywords: cooperative design, cooperative environment, modeling design, information sharing, shared model, consistency, design constraint, business rules, synchronization, meta-rules, notification

Table des Matières

Liste des Figures

Liste des Tableaux

Introduction générale

Le contexte actuel de réduction du cycle de vie des produits incite les équipes de développement de produits à accélérer le cycle de développement, en proposant rapidement des solutions de conception. Pour anticiper les erreurs de conception, la recherche de ces solutions doit prendre en compte, très tôt dans le processus de conception, les contraintes aval de fabrication, d'assemblage, de transport, …

Par ailleurs, ce qui fonde les principes de conception intégrée, les nouveaux produits, par exemple les produits mécatroniques, sont de plus en plus complexes. On parle alors de systèmes, dans lesquels plusieurs technologies issues de domaines d'application différents sont sollicités et coexistent dans un même produit, voire dans un même composant. Les concepteurs issus de ces différents domaines (mécanique, électronique, informatique, …) ont souvent des objectifs antagonistes et ne font pas appel aux mêmes pratiques et méthodes de conception.

Par conséquent, le processus de conception devient essentiellement un processus coopératif dans lequel plusieurs acteurs métier, points de vue, pratiques, méthodes et outils de conception coexistent et convergent vers un objectif commun : proposer des solutions de conception conformes au cahier des charges et respectant les contraintes de chaque métier.

Pour atteindre cet objectif, il devient de plus en plus indispensable de créer des plateformes de conception communes favorisant les liens entre différents domaines d'application et d'assurer l'interface entre plusieurs technologies et pratiques, et ainsi privilégier la coopération. Ces plateformes doivent supporter la multitude de représentations issues des différents métiers et proposer disposer un système d'évaluation multidimensionnel des solutions de conception.

Le travail coopératif en conception de produits implique des problématiques intéressantes tant au niveau de la modélisation du produit qu'au niveau de la modélisation du processus de conception :

- au niveau de la modélisation du produit, le travail coopératif nécessite des supports de représentation des données relatives au produit et un cadre de formalisation et d'échange des différents points de vue métiers. Les différents concepteurs ne pouvant pas toujours intervenir simultanément sur une même représentation du produit, ces supports doivent permettre alors de planifier et de faciliter des réunions de synchronisation des différents travaux. Ceci doit passer par des approches de détection et de résolution de conflits entre les choix effectués par les concepteurs
- au niveau de la modélisation du processus de conception et de la gestion du projet de conception, le problème principal d'un travail collaboratif est d'assurer des pratiques et des raisonnements en conception permettant de converger vers des solutions de conception plus au moins proches et acceptables par toute l'équipe de conception. Ceci passe par un système de jalonnement du processus de conception au travers de sessions de synchronisation permettant aux concepteurs de négocier le plus tôt possible les solutions de conception. Ainsi, à l'issu de chaque jalon de négociation, tous les concepteurs partent d'une même solution provisoire.

Dans beaucoup de situations, le processus de conception est traditionnellement synchrone. En effet, les concepteurs passent beaucoup de temps à résoudre conflits et à valider leurs choix plutôt qu'à avancer réellement dans le processus de conception. D'où l'inconvénient et la complexité de développer et de travailler à partir de modèles génériques et complets. Il serait

alors plus intéressant d'amorcer la conception par des modèles de base et partagés par toute l'équipe.

A l'opposé, dans certaines autres situations de conception, si pour résoudre les conflits, les concepteurs se contentaient uniquement des revues de projet planifiées à priori, il serait alors souvent trop tard pour la synchronisation car beaucoup d'incohérences seraient difficiles à lever. En effet le temps de résolution de tels conflits et de convergence vers une solution partagée par tous les concepteurs pourrait s'avérer très long et très coûteux.

Pour éviter les deux situations extrêmes, nous proposons en plus des revues de projet planifiées à priori sous forme de jalons d'introduire des revues intermédiaires correspondant à des sessions de synchronisation et de négociation. Ses sessions seront définies suivant le niveau de conflits constaté et servent à agréger plusieurs choix locaux de conception (par concepteur) en choix faisant le consensus entre toute l'équipe de conception.

Pour définir ce type de revues de synchronisation dans un contexte de conception coopérative, nous proposons un cadre formel de modélisation du produit, du processus de conception et de son organisation assurant et détectant la cohérence des choix de conception tant au niveau local (individuel) qu'au niveau global (collectif). La modélisation que nous proposons repose essentiellement sur la formalisation des règles et contraintes de conception par métier et inter métiers.

Le cadre de modélisation que nous proposons permettra de répondre à plusieurs questions relatives au processus de synchronisation:

- quand déclencher la session de synchronisation ?
- sur quels objets et sur quels choix portera la synchronisation ?
- comment organiser la session de synchronisation ?

L'apport de ce mémoire est donc de proposer des modèles et outils de gestion dynamique des règles métier dans les systèmes d'informations dédiés à la conception collaborative. Il se décompose en cinq chapitres :

Le Chapitre 1 présente les caractéristiques techniques et essentielles d'un environnement coopératif dédié à la conception. Nous commencerons par définir l'activité de conception ainsi que la dynamique et la complexité du processus de conception qui implique les démarches coopératives. Ensuite, nous montrerons les spécifications techniques et sociales qui caractérisent un environnement global de conception coopérative. Enfin, nous discuterons notre positionnement de recherche ainsi que la problématique à laquelle nos travaux tentent de répondre.

Dans le Chapitre 2, nous donnons une vision globale sur les concepts clés, manipulés en conception, relatifs à la représentation d'un produit. Nous rappelons que dans la littérature, ces concepts ont été modélisés de manières variées dans de nombreux environnements et pour des contextes de conception particuliers. Nous montrerons que ces différents concepts sont couramment utilisés pour définir un produit sous la forme d'un modèle se voulant générique, mais qui continue à évoluer au gré des développements en recherche et des applications industrielles.

L'objectif du Chapitre 3 est de formaliser les concepts présentés dans les chapitres précédents permettant de supporter une activité de conception coopérative. Au cours de ce chapitre, nous adopterons un point de vue technique qui nous permettra de matérialiser nos propositions dans l'environnement coopératif GAM développé au laboratoire G-SCOP. Pour cela, nous présenterons tout d'abord cet environnement et comment y modéliser les divers points de vue des experts de la conception. Notre modélisation repose sur le modèle de produits PPO. Pour

illustrer nos propositions, nous présenterons un scénario de conception coopérative dans l'environnement GAM en utilisant deux exemples de produits : un déclencheur électromécanique et un ventilateur d'ordinateur. Nous terminerons ce chapitre par la présentation de certains problèmes liés à la cohérence de plusieurs versions simultanées d'un modèle, comme la modification en parallèle des mêmes données ou la violation des règles métier.

Dans le Chapitre 4, nous présentons les contraintes métiers et les utiliserons comme des mécanismes de contrôle de la cohérence de modèles dans un environnement coopératif. Ces mécanismes permettront de détecter les conflits techniques lors de l'évolution du modèle partagé dans cet environnement. Pour cela, nous développons un modèle de contraintes capable de supporter la cohérence entre les choix de l'ensemble des experts en facilitant la représentation et l'échange des règles métiers.

Enfin, dans le Chapitre 5, nous détaillons nos propositions de mécanismes de mesure des évolutions parallèles des versions de modèles afin de communiquer aux différents concepteurs l'état de cohérence de l'environnement de conception et la nécessité de provoquer des tâches collaboratives. Nous spécifions les évolutions possibles du modèle partagé par la mise en place de modèles de règles métiers et des mécanismes qui permettent d'évaluer si deux activités parallèles de conception, divergent au delà de la simple comparaison syntaxique de modèles. Cette mesure de la divergence est alors utilisée pour notifier en temps voulu les acteurs de la nécessite de provoquer une session de synchronisation et de négociation.

Nous adressons enfin la conclusion de notre travail de thèse et proposons les perspectives que nous souhaitons lui donner.

1. La conception coopérative

Dans ce chapitre nous développons notre vision sur la conception coopérative, en traitant les caractéristiques d'un environnement supportant cette activité. Avant de spécifier cet environnement, nous donnons une définition générale de l'activité de conception de produit. Nous montrons ensuite à travers cette définition les caractères évolutif et complexe du processus de conception. Ces caractères imposent une forte coopération entre les différentes équipes de la conception, pour une bonne compréhension des résultats de chaque équipe par tous les acteurs concernés, pour assurer ainsi l'intégration de leurs résultats.

1.1. Différentes vues de la conception

Simon [Sim73] définit la conception comme une activité intellectuelle par laquelle sont imaginées quelques dispositions visant à changer une situation existante en une situation préférée. Tichkiewich [Tic97] précise cette définition de la conception de la façon suivante : "La conception consiste à donner un ensemble de propositions permettant de décrire le produit (forme, dimension, moyen d'obtention) et répondant globalement à un cahier des charges". La conception est souvent décrite comme un processus [Ulr00] : un processus de conception est une succession de différentes activités auxquelles sont associées des membres de l'équipe de conception pour créer un nouvel état de la définition du produit. Le processus de conception est également vu comme une agrégation de plusieurs points de vue adoptés par les différents membres de l'équipe sur le produit. Dans ce contexte, Suh [Suh01] considère le processus de conception comme la transformation du besoin, exprimé en termes d'éléments appartenant à l'espace fonctionnel, en la définition d'un produit décrit par des éléments appartenant à l'espace physique. Le processus de conception peut être organisé par la modélisation des flux d'informations qui caractérisent autant la séquence des activités, que la description des caractéristiques du produit. Selon Varcard [Var96] le processus de conception est une succession d'états de représentation du produit.

1.1.1. Représentations pour la conception

La conception serait donc modélisable par la transformation des informations (besoins, exigences et contraintes) en descriptions structurelles qui répondent à ces spécifications. Les informations en conception sont associées à un contexte particulier et évoluent suivant le processus de conception. Elles permettent la mise en place d'un processus de création de connaissances qui participent à la prise de décision. Il s'agit donc en fait d'un procédé de conversion d'informations qui caractérise les besoins et exigences pour un artefact, en connaissance sur le produit [Mis90]. Ces informations et ces connaissances dépendent de l'expérience des acteurs et sont utilisées pour construire les différentes représentations du produit au cours de l'avancement de la conception. Ces représentations, lorsqu'elles sont partagées, conduisent à des objets intermédiaires de la conception.

1.1.1.1. Objets intermédiaires en conception

Le concept d'objet intermédiaire, a été introduit par Jeantet [Jea98] comme "toute entité, physique, graphique ou textuelle, se trouvant entre plusieurs acteurs ou comme production

entre plusieurs étapes dans le cours de l'action de conception". Les objets intermédiaires sont les éléments fondamentaux dans la compréhension du flux d'informations qui caractérise la conception. En effet, les objets intermédiaires constituent les éléments principaux des interactions entre acteurs de la conception.

Jeantet propose trois types d'objets intermédiaires : l'objet de médiation, l'objet de traduction et la représentation. S'ils permettent aux concepteurs d'agir ensemble, ils traduisent aussi la transition d'un état du produit à un autre.

Ils définissent, à la fois, la modélisation du futur produit et le vecteur de la coopération ou de la coordination des acteurs de la conception. Cette modélisation est conceptualisée, et évolue avec la connaissance croissante relative au projet. L'objet est aussi un élément du processus de sa construction.

Les objets intermédiaires peuvent également être caractérisés suivant leur degré d'ouverture [Ste95]. Un objet est ouvert s'il laisse à l'utilisateur une marge de manœuvre au sein de laquelle il peut plus ou moins diverger. En revanche, il sera dit fermé s'il diminue et tend à faire disparaître cette marge de manœuvre. L'objet ouvert incite à un travail d'interprétation, tandis que l'objet fermé transmet une prescription. Par conséquent, les objets ouverts favorisent le processus de conception avec plusieurs concepteurs.

1.1.1.2. Le concept FBS

Le paradigme FBS (Fonction Structure Comportement), décrit dans [Ume90], ressort d'une analyse cognitive des concepts manipulés par les concepteurs. Il permet la représentation de manière explicite des fonctions du produit, de sa structure et de ses comportements internes. Le concept FBS et ses extensions visent à intégrer toutes les données et les connaissances relatives au cycle de la vie de produit. L'approche FBS passe par trois étapes (Figure 1.1) :

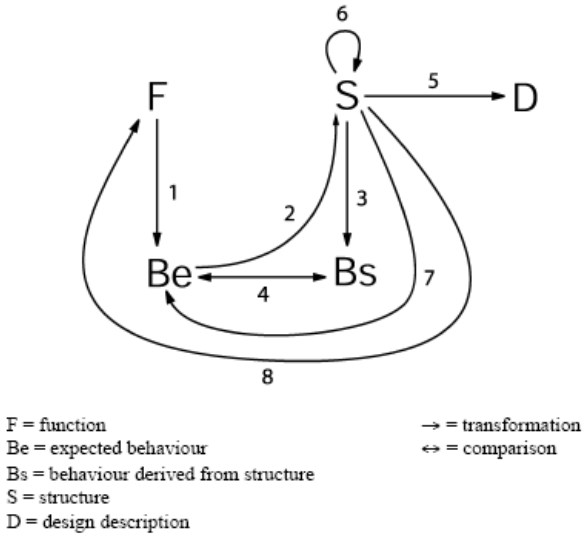


Figure 1.1 Le concept FBS [Ume90]

- **Spécifications fonctionnelles** : cette première étape porte sur la traduction des exigences de produit en fonctions qui génèrent les différentes contraintes qui doivent être remplies en conception. Les spécifications fonctionnelles sont réalisées selon une analyse fonctionnelle. Cette analyse permet d'identifier les principales fonctions du produit, pour satisfaire les contraintes et les relations entre les éléments de produits et les éléments externes.

- **Spécifications structurelle** : la deuxième étape définit les différentes composantes du produit et précise leurs propriétés physiques, géométriques, dimensionnelles, topologiques. C'est en quelque sorte le dispositif créé par le processus de conception pour assurer une fonction donnée.
- **Spécifications comportementales** : dans [Ume90], le comportement est défini comme une suite séquentielle de changements d'états de la structure, les changements d'états étant gouvernés par des lois physiques et permettant de satisfaire des fonctions.

On distingue les comportements *Be* et *Bs* : le comportement *Be* désigne ce qui est attendu de la structure, alors que le comportement *Bs* ce qui en est constaté.

Ces étapes sont caractérisées par des variables, et sont reliées par des processus de transformation. On définit alors des classes d'activités suivant les cheminements entre les différents concepts (Figure 1.2) :

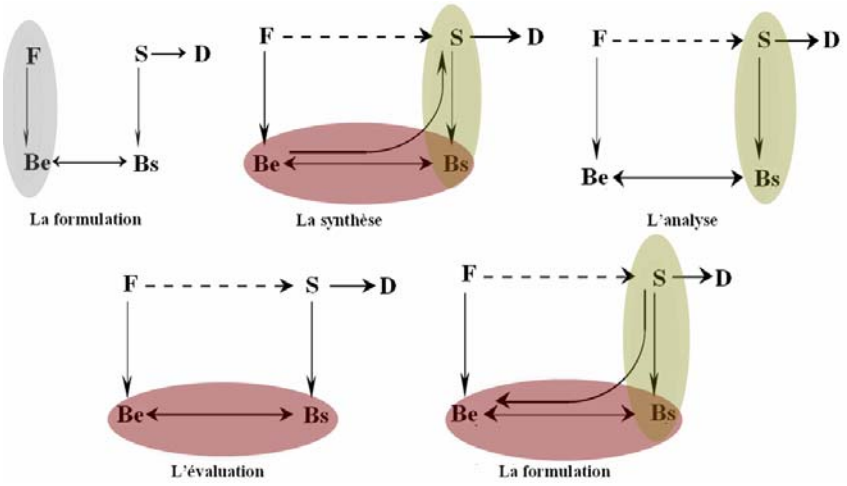


Figure 1.2 Classes d'activités du concept FBS [Ume90]

- la formulation du problème ($F \rightarrow Be$) pour exprimer les fonctions attendues en terme de comportement E du produit,
- la synthèse d'une structure ($Be \rightarrow S$ par Bs) pour répondre à ce problème,
- l'analyse du comportement S de cette structure ($S \rightarrow Bs$),
- l'évaluation ($Bs \leftrightarrow Be$), par comparaison des deux comportements,
- la documentation de la structure du produit ($S \rightarrow D$),
- la formulation de certaines variables de la structure, ou du comportement attendu ou des fonctions visées, ($S \rightarrow \{S', Be', F'\}$).

1.1.2. Dynamique et complexité du processus de conception

Le déroulement du processus de conception consiste alors en un passage d'une situation initiale d'insatisfaction à une situation objective, dans laquelle cette insatisfaction est résolue par la définition du produit. Ce processus est alors considéré comme une résolution de problème [Sim73]. Dans ce contexte, la conception est aujourd'hui l'œuvre de plusieurs acteurs différents, issus de différents métiers. Il s'agit donc d'un processus collectif.

Ces différents acteurs participent conjointement et de façon complémentaire à la résolution du problème de conception. Ils partagent un but commun, qui est celui d'aboutir à la définition du produit. Nous nous trouvons donc dans une situation de coopération [Pru03].

Cette implication d'acteurs différents vise à la prise en compte de nombreux aspects et règles intervenants dans la résolution du problème. Quand un nouveau problème de conception est initié, la solution n'est pas encore connue, surtout si le problème est nouveau pour le concepteur. Avec la progression du travail de conception, les connaissances des choix technologiques et des solutions alternatives augmentent et de nouveaux besoins apparaissent. Par contre la liberté de choix en conception diminue. Par ailleurs l'étude simultanée de différentes solutions demeure limitée. Ce constat largement admis sur le développement de produits est illustré dans la Figure 1.3.

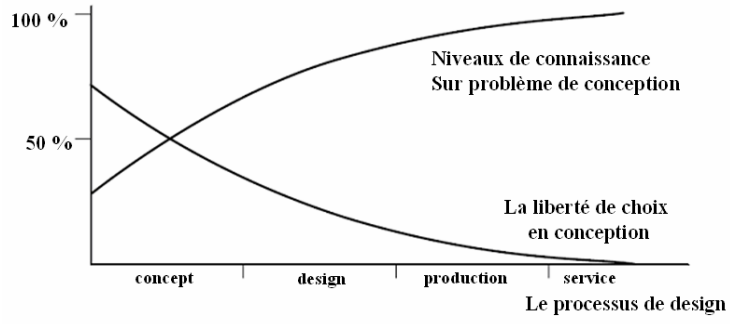


Figure 1.3 Les paradoxes en conception, d'après [Ull97]

Toutefois, au démarrage du processus de conception, les concepteurs partent rarement d'une feuille blanche. La réutilisation des connaissances de conception (acquises dans des projets antérieurs) constitue un point de départ limitant de fait la liberté de choix en conception.

1.2. Conception coopérative

Dans la littérature, on rencontre deux expressions souvent sans distinction : "conception collaborative" et "conception coopérative". Les termes "coopérative" et "collaborative" sont très proches, mais font référence à deux formes d'organisation de travail différentes. Le "Petit Robert" les définit ainsi :

- **Collaborer** : de co « avec » et laborare « travailler », Travailler avec autres personnes
- **Coopérer** : Agir conjointement avec quelqu'un

En effet, deux personnes peuvent collaborer, c'est-à-dire participer à une œuvre commune sans pour autant agir conjointement. Alors que lorsque deux personnes coopèrent, elles agissent conjointement sur un même objet.

La conception collaborative fait référence à un travail d'équipe organisé qui fonctionne suivant un planning impliquant des délais et un partage des tâches. Chaque intervenant sait ce qu'il doit faire et communique, échange ou partage des éléments uniquement pour créer un objet unique de travail. Par contre, en conception coopérative, le travail d'équipe fonctionne sans véritable organisation préalable et chaque intervenant apporte son savoir, son idée en pouvant ou pas s'inspirer sur les apports réalisés précédemment par les autres. L'apport individuel est difficilement identifiable à la fin.

Pour coopérer et collaborer, les acteurs de conception ont besoin de communiquer afin de prendre des décisions ensemble ou afin d'informer les autres de l'avancement de la conception ou de problèmes rencontrés. Ils ont aussi besoin de partager des ressources, qu'elles soient informatiques ou physiques. Finalement, ils ont besoin de planifier leur travail, séquentiellement temporel, distribuer des tâches à chaque membre et d'affecter des rôles

Nous adoptons pour cette thèse le terme de conception coopérative car nos travaux concernent la spécification d'un système d'information pour gérer les connaissances mises en jeu dans des situations de conception coopérative.

1.2.1. Les Principes de la conception coopérative

La conception coopérative utilise la coopération pour améliorer les processus de conception multidisciplinaire en intégrant les différentes vues expertes. Ceci se formalise par des environnements multi vues pour la mise en cohérence des activités de conception. Les recherches dans ce domaine consistent :

- à définir et mettre en œuvre des structures pour l'intégration des activités multidisciplinaires et multiculturelles de conception et des processus de décision dans un contexte coopérative
- à développer des méthodes et des outils de construction, de partage, de manipulation et d'adaptation d'information qui constituent les supports intégrateurs de la conception coopérative
- à formaliser les besoins des utilisateurs de produit, les connaissances des experts, et les règles métiers pour les rendre disponibles au plus tôt dans le cycle de vie de produit
- à formaliser les activités à l'interface de plusieurs acteurs, éventuellement par l'analyse des pratiques

Dans un but de structuration et d'intégration des activités multidisciplinaires, les activités de conception peuvent être réorganisées. L'interaction entre acteurs peut être le paradigme central de cette réorganisation. Cette nouvelle organisation de la conception oblige les experts de la conception projet à communiquer, à échanger des informations avec des coopérateurs qui peuvent être de cultures et de métiers différents. Les différents experts de la conception doivent donc investir cette dimension culturelle dans la mise en place d'une approche de la conception coopérative pour prendre en compte les exigences de chaque métier.

1.2.2. Outils informatiques support à la coopération

Les systèmes coopératifs sont des systèmes permettant à plusieurs acteurs de travailler ensemble via une infrastructure informatique. Ces systèmes s'intéressent aux moyens permettant de supporter les activités coopératives de la conception. De nombreux travaux dans ce domaine ont cherché à mieux comprendre les fondements des activités des acteurs de la conception, en vue de leur fournir des supports informatiques plus appropriés à la coopération.

Nous allons voir dans cette partie quels sont les aspects techniques et les supports informatiques pour assister les concepteurs dans leur activité de conception. Nous listons les typologies et les fonctionnalités des outils coopératifs pour répondre au mieux à certaines exigences de la conception coopérative et également les problèmes induits par l'utilisation de ces novelles technologies.

1.2.2.1. Conception coopérative assistée par ordinateur

Depuis 1984, un domaine de recherche, appelé Computer Supported Cooperative Work (CSCW) [Gru94], littéralement Travail Coopératif Assisté par Ordinateur (TCAO), étudie la conception, la construction et l'utilisation des systèmes coopératifs. Ce domaine « regroupe l'ensemble des techniques et des méthodes qui contribuent à la réalisation d'un objectif commun à plusieurs acteurs, séparés ou réunis par le temps et par l'espace, à l'aide de tout dispositif interactif faisant appel à l'informatique, aux télécommunications et aux méthodes de conduite de groupe ». Dans ce domaine, la communauté de recherche CSCWD « Computer Supported Cooperative Work in Design » étudie les façons de travailler en coopération à travers l'utilisation de l'informatique, dans un contexte de conception de produits industriels. Elle définit les fonctionnalités des logiciels à remplir pour répondre mieux à différents aspects de la conception coopérative. Les travaux dans le domaine du CSCWD utilisent sciences humaines et sociales pour éclairer l'étude des activités coopératives de la conception. Ceci fournit des cadres permettant de comprendre l'activité collective et d'analyser les concepts nécessaires pour la conception d'outils ou d'environnements informatiques pour le support de ces activités.

1.2.2.2. Théorie de l'activité pour le développement des systèmes coopératifs

La conception coopérative est une activité complexe, où il existe des interactions entre humains et matériels. Le développement et la mise en œuvre des outils de la conception coopérative nécessitent une étude approfondie des activités des concepteurs. Il est souvent considéré que pour mieux comprendre ces activités coopératives, l'étude de la théorie de l'activité humaine [Hal02] [Kut96] est l'un des principaux thèmes qui permettent de mettre en évidence des éléments particuliers caractérisant les activités des concepteurs, et de mieux comprendre la place et le rôle que peuvent y occuper les systèmes informatiques.

Ainsi, Engeström [Eng01] a défini un modèle simple du concept d'activité (Figure 1.4). Ce modèle est utilisé pour définir les caractéristiques des outils coopératifs de la conception. Ce modèle considère une activité comme un phénomène collectif, dirigé vers un objet idéal qui la motive et autour duquel est constitué un groupe de sujets interagissant jusqu'à sa réalisation.

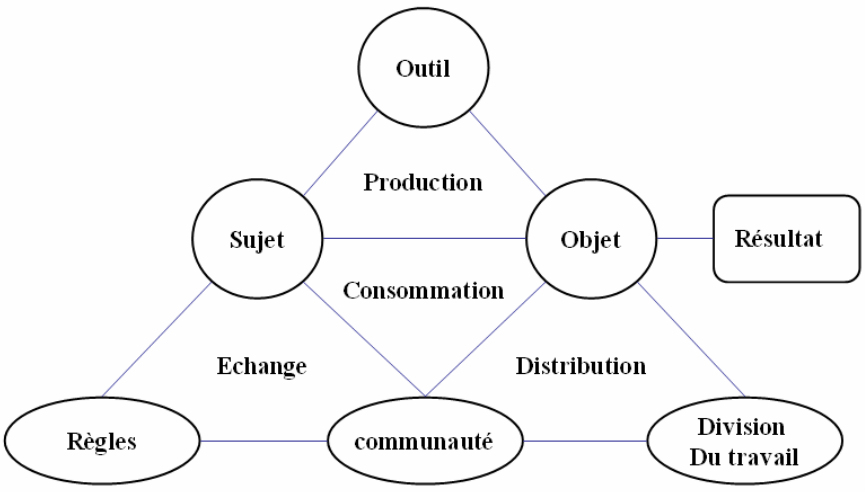


Figure 1.4 Structure de l'activité humaine, d'après [Eng01]

L'objet de l'activité fait référence à l'espace de problème vers lequel l'activité est dirigée. Le sujet réalise l'activité en transformant cet objet et en le représentant avec des outils. Il est intéressant de souligner la place particulière de l'outil au sein de l'activité : en tant qu'artefact médiateur et moyen de transformation de l'objet en représentation de produit, l'outil permet, et il limite tout à la fois, en restreignant cette transformation au cadre imposé par sa nature même ainsi que par les fonctionnalités qu'il propose. De plus, l'outil informe le sujet sur l'état de l'activité. Le sujet fait partie d'une communauté qui englobe l'ensemble des sujets partageant le même objet. La division du travail, artefact qui médiatise la communauté et l'objet, se rapporte à la répartition des tâches entre les membres de la communauté, aux éléments qui définissent le pouvoir et le statut de chacun des membres. Enfin, les règles représentent les règlements, normes et conventions, implicites ou explicites, qui régissent et contraignent les actions et interactions des sujets au sein de la communauté.

Les sujets de l'activité participent à cette évolution et en sont généralement les acteurs : ce sont les sujets qui font évoluer l'activité, en réponse à de nouveaux besoins, aux contradictions qui naissent dans l'activité. L'évolution du système informatique doit donc pouvoir être réalisée par ses utilisateurs.

La théorie donne aussi une vision des mécanismes d'apprentissage, dans une succession de trois étapes indispensables pour tout environnement informatique supportant une activité de conception collaborative :

- coordination : chacun exécute son propre rôle et les actions consignées,
- coopération : problème commun, chacun essaye de trouver des façons acceptables de le conceptualiser,
- communication réflexive : reconceptualiser les différentes interactions entres acteurs en relation avec les objets d'activités partagées

1.2.2.3. Typologies des outils de conception coopérative assisté par Ordinateur

Les outils informatiques de coopération sont construits autour de quatre cadres [Sal95] : cadre espace-temps, cadre conceptuel, cadre fonctionnel, cadre organisationnel.

Le cadre espace-temps

Le cadre espace-temps définit tous les dispositifs techniques permettant de mettre en relation et coopération des acteurs de la conception. Le cadre prend en compte la localisation physique des acteurs et le moment où ils participent à la conception. Ce cadre basé sur une classification des systèmes de CSCWD, a été proposée à l'origine par Johansen [Joh88] afin de mieux définir différentes interactions entre des acteurs impliqués dans processus de conception coopérative.

Quatre catégories (Tableau 1-1) peuvent être distinguées selon que le travail des acteurs se déroule dans un même espace ou dans des espaces différents, en même temps (synchrone) ou en temps différé (asynchrone). Ces catégories présentent les outils pouvant être utilisés en fonction de la matrice espace/temps.

Tableau 1-1 Matrice espace-temps dite de [Joh88]

	Temps		
Espace		**Même temps (synchrone)**	**Temps différé (asynchrone)**
	Même espace de travail	° Salle de réunion équipée d'un ordinateur pour chaque membre du groupe avec une même et unique interface graphique ou avec un seul ordinateur et un vidéo projecteur ° Travail à travers un tableau ° Logiciels qui permettent le vote électronique (pour la prise de décisions)	° Des agendas électroniques pour pouvoir fixer une date ° Les newsgroups électronique ou les forums de discutions ° Centre de ressources partagées un même espace disque sur le réseau local. ° Gestion de suivi de projet ° Les outils de planning
	Espaces de travail différent	° Vidéoconférence ° Chat ° Partage d'applications ° Transfert de fichiers ° Les éditeurs multi utilisateurs ° Les systèmes de tableaux blancs	° Le courrier électronique ° Calendrier partagé ° Base de données commune ° Systèmes de gestion des données techniques ° Les systèmes de flux de travail

Le cadre conceptuel

Le cadre conceptuel est défini par un ensemble d'éléments référentiels qui permettent aux acteurs de se comprendre, d'échanger des informations et produire conjointement un parasitage sur le concept de travail. En conception, le cadre conceptuel permet de repositionner les questions sur le concept adapté à une bonne représentation du produit. En effet, il doit : formaliser des objets partages, intégrer et formuler ensemble des règles et des connaissances de chaque métier qui permettent de se comprendre et de coopérer dans le processus de conception.

Le concept d'objet intermédiaire de la conception, contribue à ce cadre en permettant d'avoir la globalité des points de vue des métiers dans la conception. Avec ce nouveau regard, l'objet intermédiaire :

- constitue un bon vecteur de coopération, l'objet intermédiaire doit concerner le contenu même du futur produit. Plus ce rapport au contenu sera fort, plus l'efficacité de l'objet en tant qu'instrument de coopération sera grande,
- doit être représenté la nature, la forme, le contenu de l'objet influent sur la forme de la communication entre les acteurs de la conception,
- doit être ouvert (autant que possible). En d'autres termes, il doit laisser une marge de manœuvre à leurs utilisateurs, mais aussi traduire une forte présence du futur produit afin d'être un moyen de coopération efficace.

Le cadre fonctionnel

Le cadre fonctionnel définit la structuration des données partagées, la modélisation du support du travail et des opérations de production, l'organisation du processus de travail par l'identification des tâches et des rôles et le choix des modes de communication et des protocoles de conversation. En résumé, ce cadre définit quatre espaces avec des fonctionnalités différents pour les outils de CSCWD comme la montre la Figure 1.5 :

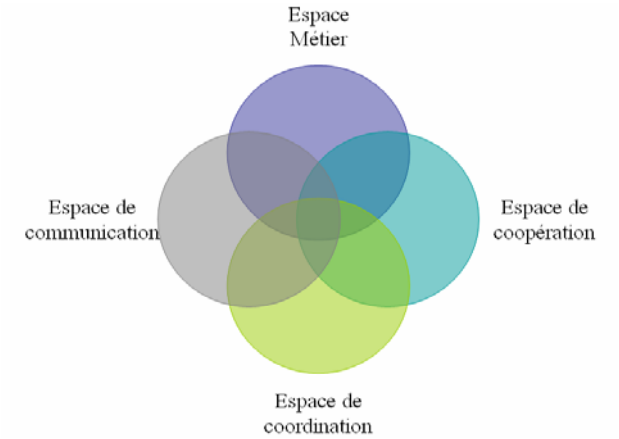


Figure 1.5 Les espaces fonctionnels [Sal95]

Espace de coopération

L'espace de Coopération désigne les objets qui résultent d'une activité de groupe. C'est dans cet espace que les données, le cadre conceptuel et les objets intermédiaires de la conception sont situés et accessibles selon certaines règles de partage à tous les acteurs. Il s'agit donc ici essentiellement des données partagées et des outils permettant d'obtenir et de manipuler de telles données.

Les problèmes spécifiques aux espaces de coopération sont définis sous trois angles différents celui de l'hétérogénéité des données, celui de leur partage et celui de leur disponibilité :

- **Hétérogénéité des données** [Elm99] [Bre86] : dans un processus de conception, des données et des objets intermédiaires de la coopération ne sont pas conformes à une référence unique donc l'intégration ou le simple échange des données n'est pas une tâche facile si les différents intervenants de la conception (producteurs ou consommateurs des informations) ne s'entendent pas sur la sémantique des données. Dès lors les principales questions posées sont : (1) comment résoudre les problèmes liés à l'hétérogénéité ?, (2) comment tenir compte des préférences du concepteur dans la conception de l'espace de coopération ?, (3) comment donner la possibilité aux concepteurs d'exprimer leurs préférences ? et (4) comment évaluer la qualité de l'espace de coopération ?
- **Partage des données** : un espace de coopération nécessite que les données soient partagées entre ses différents utilisateurs. Les données sont partagées par leur circulation entre différents acteurs selon une procédure bien déterminée (c'est le rôle des logiciels de gestion de processus workflow), où les données peuvent être partagées à tout moment par tous les acteurs qui disposent d'un droit de regard sur les données. Cela entraîne des problèmes de cohérence des données. En effet, les accès concurrents aux mêmes données ou les évolutions parallèles des données peuvent conduire à des incohérences.

- **Disponibilité des données** : dans un environnement de conception coopérative la possibilité de travailler en groupe ne doit pas perturber l'activité personnelle d'un concepteur particulier. Ainsi chaque acteur d'une coopération doit pouvoir accéder aux données qui lui sont nécessaires. Par exemple, un acteur ne doit pas travailler sur des données alors que quelqu'un d'autre les a verrouillées. La disponibilité des données est conditionnée par les fonctionnalités de contrôle des données partagées.

Espace de coordination

L'espace de coordination est le lieu privilégié pour l'assignation de tâches et de rôles aux différents acteurs de la conception. Il a pour but de coordonner les acteurs afin de réaliser une œuvre commune. Cet espace permet de définir la collaboration dans le temps. Il exprime les relations entre les utilisateurs et leurs activités. La coordination assure l'efficacité, dans la réalisation de la tâche, du groupe. Elle permet la production collective d'un groupe en assurant la synchronisation relative au travail commun, en gérant, en particulier, les conflits qui apparaissent.

Dans l'espace de coordination, il est nécessaire d'assurer l'avancement du projet, l'intégration des travaux et des règlements menés par les acteurs dans les différentes disciplines et les cohérences entre les actions et les ressources et objets communs de conception.

Lors du lancement d'un projet, les intervenants planifient des revues de projet et des jalons bien précis. Au cours des revues de projet, les partenaires font le bilan de l'état d'avancement du projet. Ils vérifient la cohérence des résultats obtenus lors de l'intégration de leurs travaux, et décident des suites à donner. De telles fonctionnalités doivent donc aussi être disponibles dans les environnements de conception coopérative.

Les concepteurs ont besoin d'informations techniques pour prendre des décisions, or, ces informations peuvent être détenues par un partenaire éloigné. Dans le but d'augmenter la réactivité et la rapidité des prises de décisions, un acteur doit pouvoir communiquer avec le détenteur de l'information qui lui est nécessaire. Ce type de communication correspond à des réunions techniques quasi quotidiennes entre deux partenaires. Il faut donc fournir des fonctionnalités équivalentes dans un contexte de conception coopérative pour mettre en relation des acteurs. Étant donné que ces réunions sont impromptues, les données nécessaires à la prise de décision doivent être disponibles.

On peut imaginer que les acteurs de la conception partagent leurs données hétérogènes, et que tous travaillent en parallèle avec leurs propres compétences sur ces données partagées. Lorsque ces données évoluent, la synchronisation doit permettre la fusion des évolutions. Ainsi le processus de synchronisation doit tenir compte des changements effectués par chaque participant sur les parties communes pour guider le processus de prise de décision collective.

Ce mécanisme, souvent associé aux environnements partagés asynchrones, permet de comparer et de fusionner différentes versions des mêmes données, tout en garantissant que la fusion ultérieure des différentes versions sera cohérente.

Espace de Communication

L'espace de communication permet de décrire formellement une communication entre les participants dans un groupe de travail. Il offre aux acteurs de l'espace de coordination la possibilité d'échanger des informations, des savoirs et savoir-faire. Cet espace doit permettre des échanges aussi bien formels qu'informels. Plus cet espace sera riche, plus facile sera la création de l'esprit d'équipe primordial pour une collaboration réussie. La notation de cet espace doit être générale, afin de pouvoir décrire n'importe quel type de communication

nécessaire entre intervenants de conception coopérative. Cet espace est composé de quatre éléments : les intervenants, l'objectif (message), la forme et la structure de la communication :

- **les intervenants de la communication** : l'espace de communication, définit un ensemble d'intervenants selon leur rôle et leur capacité à interagir pendant le processus de conception. Cet espace catégorise des intervenants pour différents types de participation à la communication. Par exemple :
  - *Un animateur* : l'intervenant qui peut établir et gère la communication
  - *Des participants* : l'ensemble d'intervenants qui participent a la communication
- **l'objectif et le message de la communication** : après avoir défini les participants, il est nécessaire d'assembler les objectifs de la communication. Il faut définir l'objectif de la communication en sachant que l'information et la communication sont un ensemble d'activités spécifiques qui participent à atteindre des résultats, des objectifs et un but.
- **la forme de la communication** : La forme de la communication est définie en fonction des intervenants et des objectifs. Dans un environnement de coopération chaque intervenant peut avoir recours à des formes de communication différentes qui peuvent être regroupées dans les familles suivantes [Med04] :
  - *Communication diffusée* : Cette communication s'effectue entre les acteurs de la conception. Sa caractéristique principale est que l'information est diffusée à tous les intervenants de la communication. Ce type de communication est équivalent à une communication publique.
  - *Communication privée* : Ce type de communication, concerne l'information strictement échangée entre deux acteurs. Certains acteurs ont le droit d'établir une communication privée avec un autre acteur.
  - *Communication réglementée* : Cette communication est établie entre les acteurs de la conception avec des droits de participation et de consultation de la communication en fonction du rôle des intervenants. Par exemple, Dans cette classe de communication, il est indispensable d'établir des droits de lecture et d'écriture sur le matériel support employé au cours de la séance de communication.
- **La structure de la communication** : La structure de la communication est définie en fonction des formes de communication et peut être réalisée par des outils informatiques.

Espace métier

L'espace métier est le lieu des activités spécifiques à un métier. Comme cela a été observé maintes fois dans les différents expériences de conception coopérative, cet espace reste indispensable aux acteurs pour qu'ils puissent avoir un temps de réflexion, de calcul, de modification, d'interprétation des informations, de structuration des informations et des données qu'ils vont devoir partager et présenter aux autres intervenants de la coopération [Mec06].

Le cadre organisationnel

Le cadre organisationnel regroupe les éléments (espace, temps, conceptuel et fonctionnel) qui caractérisent la structuration du travail coopératif pour un bon déroulement de la coopération entre acteurs. On définit deux types d'organisations pour un environnement de coopération :

L'organisation statique

Elle concerne en général la structure relativement stable dans le temps d'une organisation de travail coopérative. Les fonctions techniques de l'organisation statique dans un environnement coopératif sont :

- la création et la destruction des projets, des activités liées à ces projets et des tâches à accomplir et à coordonner au sein de ces projets,
- l'ajout ou suppression des participants ou données qui peuvent être les résultats d'un projet, d'une tâche ou d'une activité,
- le contrôle et la synchronisation des données liées aux tâches afin de respecter les délais spécifiés,
- la modification des rôles des participants,
- le contrôle de la sécurité de l'environnent de coopération : en effet l'appartenance d'un participant à un projet, activité ou tâche et le rôle qu'il y joue détermine les droits qu'il possède sur ces ensembles ainsi que les droits qu'il possède sur les données.

L'organisation dynamique

Elle concerne en général une instance organisation très évolutive dans le temps. Elle permet de redéfinir des connexions et des données manipulées (communication ou coordination). Les fonctions techniques de l'organisation dynamique dans un environnement coopératif complétant les fonctions des organisations statiques sont :

- l'identification des intervenants qui se connectent ou se déconnectent de la plateforme de coopération,
- la création ou destruction d'une séance de coopération, l'entrée d'un participant dans une séance, la sortie d'un participant d'une séance,

1.2.3. Relations de confiance dans les activités collaboratives

La structure technologique mise en place ne doit pas perturber le travail des acteurs de la conception. Lors de la mise en place d'environnements coopératifs, il faut donc avoir conscience que l'aspect social est au moins aussi important que l'aspect technique. En effet, il est clair que les solutions techniques ont des effets non négligeables sur la dimension sociale de l'environnement coopératif [Hee04].

Une conception coopérative ne peut être efficace qu'à la condition que les acteurs partagent leurs informations et leurs connaissances. Ceci nécessite une certaine confiance mutuelle et des liens sociaux bien établis (les échanges informels) qui se développent avec le temps. En effet, avec le temps, les acteurs apprennent à se connaître et à travailler ensemble. Cette confiance s'impose difficilement dans un contexte de conception distribuée. Les acteurs mettent beaucoup plus de temps à se connaître et à travailler ensemble.

L'environnement mis en place pour la conception coopérative doit par conséquent reproduire cette confiance, ce qui est très important pour une collaboration distante efficace. Il faut que les membres d'une équipe se connaissent. Chacun doit pouvoir obtenir des informations sociales, organisationnelles et globales relatives à la tâche et aux activités en cours. De la sorte, chaque individu connaît les activités de tous les autres membres du groupe et ainsi, il peut coordonner ses activités.

1.3. Du principe de coopération à des environnements coopératifs

Nous avons commencé ce chapitre en définissant la conception, sa dynamique et sa complexité qui imposent la coopération.

Nous avons présenté par la suite des éléments de spécifications d'un environnement global de conception coopérative, des aspects techniques et des aspects sociaux à prendre en considération pour la mise en place d'un environnement coopératif répondant aux besoins des concepteurs. Nous allons préciser notre travail dans le cadre de ces spécifications.

1.3.1. Vers une structuration des revues de projets

L'organisation du développement de produit est une activité collaborative et distribuée. Comme montré dans la Figure 1.6-a, chaque expert dans un ou plusieurs domaines a son propre point de vue technique pour concevoir et réaliser un produit. Il utilise des ontologies spécifiques pour représenter un produit. La discussion autour de ces ontologies intervient au cours des revues de projet. Parmi les activités collaboratives de développement de produit, la revue de projet est une des activités les moins formalisées. Nous cherchons donc à contribuer à sa formalisation et à sa structuration car elle nous semble être un élément critique pour la justification des décisions de conception.

La formalisation des revues de projet soulève plusieurs problèmes dont :

- la synchronisation des différents modèles experts,
- la définition d'un cadre de représentation des différents points de vue du produit (langage commun) pour permettre l'hétérogénéité dans la formalisation,
- l'échange d'information
- la résolution de conflits entre acteurs métiers lorsque par exemple un choix technique du bureau d'étude s'avère impossible à réaliser en fabrication ou lorsqu'une demande de modification au BE entraîne à son tour d'autres modifications imprévues.

Plusieurs sujets de Recherche sont alors ouverts (Figure 1.6-b) :

- Comment définir une représentation commune partagée pour gérer des liens entre les vues métiers différentes ?
- Comment comprendre et modéliser les interactions entre experts métiers pour suivre la cohérence des informations ?
- Comment décider des revues de projet et des points de synchronisation ?

L'environnement coopératif auquel nous souhaitons contribuer doit répondre à ces questions.

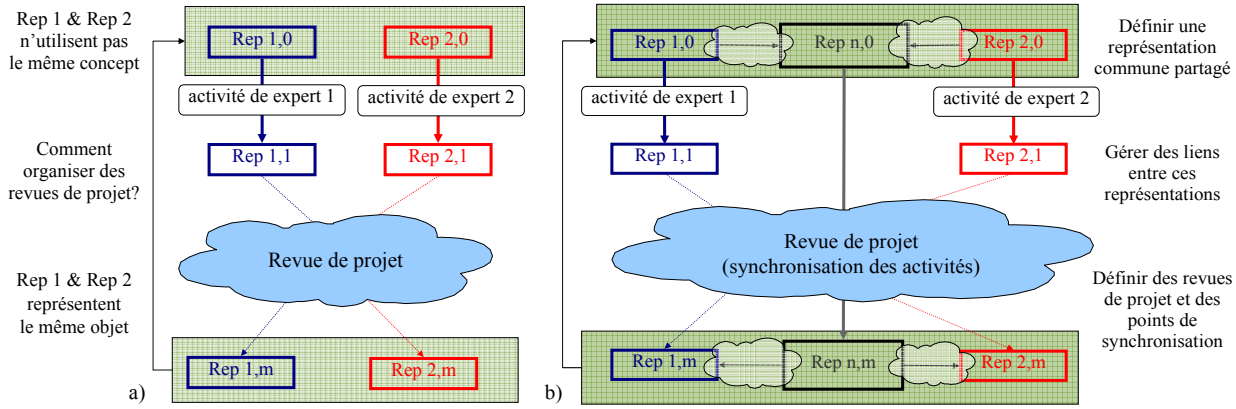


Figure 1.6 Formalisation des revues de projets de conception

Nous avons discuté précédemment trois cadres pour la spécification d'un environnement coopératif de la conception pour constituer un espace virtuel d'échanges pour améliorer l'efficacité de l'équipe de conception dans son ensemble.

En ce qui concerne le cadre espace-temps, notre travail s'insère dans le cas du mode de travail "asynchrone" avec un espace de coopération commun qui se base sur un modèle partagé.

Le modèle partagé assure l'intégration des données avec des descriptions hétérogènes, leur partage et leur disponibilité pour tous les acteurs de la conception.

Dans le cas du mode "asynchrone", chaque acteur dispose d'une copie du modèle commun. Chaque copie évolue en parallèle et ainsi les versions modélisées divergent progressivement. Dans le mode "asynchrone" les concepteurs copient une partie du modèle partagé dans un espace privé (espace métier) afin d'obtenir une version stable pendant leur travail. Chaque concepteur modifie l'objet considéré selon son expertise et ses propres critères. Pour assurer l'émergence d'une solution commune leurs résultats doivent donc être fusionnés. Chaque expert est responsable de son espace privé. A la fin, chaque concepteur doit intégrer des modifications dans la base de données partagée.

Nous verrons dans les chapitres suivant les différents travaux qui participent à la formalisation du cadre conceptuel, du cadre fonctionnel et du cadre organisationnel. Ces travaux concernent les méthodes de modélisation de processus de la conception. Une difficulté dans ces travaux est le maintien de la compatibilité et de la cohérence des différentes représentations. En conséquence, l'identification des impacts issus d'une modification passe par une identification manuelle par des experts. Ce processus manuel ne peut pas assurer un cheminement complet de toutes les modifications du modèle partagé. Par conséquent, une approche efficace pour dépister les modifications de modèle partagé est nécessaire pour améliorer de manière significative le processus de synchronisation.

Le plus souvent, les acteurs travaillent indépendamment, modifiant des versions alternatives des représentations du produit. Tous les conflits produits exigent la négociation et la propagation des décisions de modifications vers les différentes représentations des experts pour résoudre le conflit. Ceci est habituellement réalisé lors de la synchronisation entre les différentes versions. L'assistance pour la gestion de tels conflits requière :

- le stockage des représentations publiques et privées,
- le contrôle de l'intégrité des informations produites,
- une assistance à la résolution des conflits identifiés.

Nous verrons que certains conflits syntaxiques peuvent être automatiquement identifiés par des outils mettant à jour l'information dans une vue quand une autre vue est éditée. Toutefois, la plupart du temps les conflits ne seront pas automatiquement résolus. Par conséquent des mécanismes sont nécessaires pour détecter et évaluer ces conflits afin de notifier les acteurs concernés.

Dans le cadre de nos travaux, nous pensons que la gestion de conflits pourrait être améliorée si le système de conception coopérative emploie un système temps réel pour détecter les conflits structuraux ou sémantiques entre les différentes représentations. Dans tous les cas, il est nécessaire de modéliser les interactions entre les entités du modèle partagé. L'évolution de chaque entité ne peut être considérée d'une manière indépendante de celle des autres entités du modèle. Le changement d'état (modifiée ou non modifiée) d'une entité peut être obtenu par modification directe de l'entité ou indirectement suite à la modification de certaines autres entités du modèle. La modélisation des relations entre les objets nous permet dans un premier temps de connaître l'impact de changement d'état d'une entité sur le reste du modèle, ensuite d'informer les bonnes personnes des changements qui ont lieu et qui les concernent.

La mesure de l'incohérence est très importante pour assister le maintien de la cohérence du modèle partagé. Il y a deux solutions imaginables : a) comparer les graphes d'informations associés à chaque copie pour résoudre des conflits syntaxiques (même valeur pour le même attribut), b) mise en jeu de contraintes métiers qui définissent des conflits complexes mettant en œuvre des règles expertes.

La mesure de la violation de chaque contrainte peut servir à quantifier l'incohérence globale du modèle. Si nous considérons le modèle global comme un réseau de contraintes et de relations entre les entités de conception, les conflits s'expriment par l'identification des contraintes violées au cours de la conception. Dans cette approche, lors de la connexion d'une expertise à l'environnement partagé, un espace privé géré dans l'environnement de partage est alloué à l'expertise. Chaque expertise intervenant dans le processus collaboratif dispose d'un tel espace. Les modifications effectuées par chaque expertise sont capitalisées dans ces espaces privés sans modification de l'espace public. En fonction des interactions entre entités des messages de notification sont envoyés à l'ensemble des acteurs concernés. Ainsi les acteurs sont informés de l'existence de travaux impactant les leurs sans pour autant que leur jeu de données partagées soit modifié. Il est alors possible d'imaginer des mécanismes de mesure du cumul des interactions permettant de décider le moment opportun d'une synchronisation des modèles. Le résultat de la synchronisation est alors partagé au travers de l'espace commun.

1.3.2. Démonstrateur des fonctionnalités proposées

Nous utiliserons deux systèmes : un déclencheur électromécanique et un ventilateur d'ordinateur ; exemples des produits pour construire un scénario de conception coopérative. Lors de la conception de tels produits, des experts électrotechniciens et des experts mécaniciens apportent plusieurs compétences et connaissances spécifiques. Le premier domaine de compétence est l'application des principes scientifiques et du savoir-faire d'une manière qui assure un développement optimal du produit. L'électrotechnique conduit à des études électromagnétiques, souvent réalisées en parallèle. L'électromagnétisme est une théorie des champs où les quantités physiques (ex. un courant électrique ou un champ magnétique) sont définies sur tout l'espace. Afin de simuler ces phénomènes, les électrotechniciens utilisent des outils mathématiques. Pour le mécanicien, le problème associé avec cet aspect de continuité de l'électromagnétisme est qu'il est difficile de connaître où et comment un effort agit sur une structure.

Comme la plupart des domaines d'ingénierie, dans la conception d'un produit les électrotechniciens adoptent des symboles afin de communiquer succinctement des informations. Ces symboles ont évolué et sont appris tout au long de la formation d'un ingénieur. La plupart de ces symboles ne seront pas compris par un mécanicien ou, sont interprétés autrement.

Un deuxième domaine est la connaissance et l'exploitation des aspects comme des structures et des solutions existantes, et des technologies. Le passage d'une fonction à une structure doit passer par les principes de la physique et un ingénieur fait appel à son expérience afin d'intégrer des aspects de technologie qui ont fait leurs preuves. En fait, il est fort probable qu'un mécanicien et un électrotechnicien, en travaillant séparément, réaliseraient deux structures qui satisfont à priori la même fonction. Dans un projet de conception coopérative il faut pouvoir prendre en compte des solutions possibles provenant de domaines culturellement différents. Nous verrons à travers ces exemples, comment adapter un environnement coopératif pour exprimer les différentes représentations de produit, les interactions entres des acteurs qui ont des connaissances différents. Cet environnement doit conduire à gérer les incohérences et permettre la synchronisation entre acteurs.

2. Modélisation pour la conception collaborative

Le chapitre précédant a défini le contexte de l'étude. Pour notre étude, il est nécessaire de disposer de représentations du produit. Dans ce chapitre, nous faisons le point des modélisations utilisées en conception qui permettent la collaboration entre acteurs. Nous montrons que dans la littérature, ces concepts ont été modélisés de manières variées dans de nombreux environnements et pour des contextes de conception particuliers. Nous montrerons que ces différents concepts sont couramment utilisés pour définir des produits selon des modèles dits génériques, mais qui continuent d'évoluer au gré des développements en recherche et des applications industrielles.

2.1. Modélisation en conception

2.1.1. Modélisation de l'activité de conception

Un modèle de la conception doit définir l'ensemble des concepts et des relations qui décrivent les informations attachées au développement d'un produit. Plusieurs approches de modélisation existent aujourd'hui et diffèrent selon leurs domaines d'applications et leurs niveaux de détails. La plupart de ces approches essaient de représenter le produit sous forme d'un modèle produit [Kim93].

Dans [Isa00], les auteurs définissent le modèle produit comme étant un "réservoir" d'informations et de données relatives au produit qui vise à supporter les activités de conception tout au long de son cycle de vie. En effet le modèle produit permet une modélisation fine et multi niveaux du produit suivant plusieurs concepts (Tableau 2-1).

A l'instar des concepts de structure et de fonction, la notion de comportement a été introduite comme un élément fondamental de modélisation d'un produit [Ger90]. En effet d'une part l'objet physique (concept de structure) est conçu et réalisé pour satisfaire des besoins déterminés par l'expert humain et d'autre part, la fonction se rapporte à un besoin exprimé ou à l'utilité d'un produit, c'est-à-dire à la raison de son existence. Le comportement quant à lui est défini comme la totalité des propriétés d'un objet et émerge comme le résultat d'interactions entre la structure de l'objet et son environnement. En effet, les fonctions que doit remplir l'objet donnent lieu à un ensemble de propriétés comportementales permettant à ces fonctions d'être réalisées. Les comportements sont satisfaits par plusieurs propriétés structurelles. Les concepteurs définissent les propriétés structurelles de façon à ce que les comportements actuels de l'objet satisfassent les comportements attendus, et par conséquent satisfassent les fonctions attendues. Ainsi, les nombreux acteurs métier impliqués dans le processus de conception ont des connaissances et des exigences différentes sur le produit à réaliser, en technologie, en tolérancement, ou en fabrication, etc. La connaissance métier est un moyen donné aux acteurs pour décrire le produit dans sa propre vue métier. Dans [Tic97] [Bri01], le modèle produit est défini comme "une structure multi-vues dépendant des activités de conception et des différents métiers intervenant sur le produit durant son cycle de vie".

Tableau 2-1 Concepts fondamentaux pour un modèle de produit

concept	les objectifs	entités
Fonctionnel	- Fonctions à remplir par l'objet - Flux fonctionnels	- Fonctions - Liaisons fonctionnelles - Surfaces fonctionnelles
Structurel	- Décomposition des ensembles - Position des ensembles	- Objets technologiques - Positions - Architecture
Volumique	- Définition géométrique - Caractérisation des matériaux	- Géométrie/ aménagements - Matériaux, Paramètres - Surfaces géométriques
Comportemental	- Caractérisation des phénomènes physiques	- Comportement - État fonctionnel - Paramètre
Multi points de vue (structurelle, fonctionnelle, comportemental)	- Modes de fabrication - Modes d'assemblage - Tolérances, - Traitements.	- Procédés - Assemblages - Traitements - Propriétés technologiques

La maîtrise du cycle de vie du produit et de son processus de conception impose de structurer les informations pertinentes qui caractérisent le produit, son environnement et les événements générés pendant les phases de conception et d'industrialisation. Les modèles de conception sont donc un moyen de représentation du cycle de vie de produit par la formalisation des informations associées (Figure 2.1) [Rob05] :

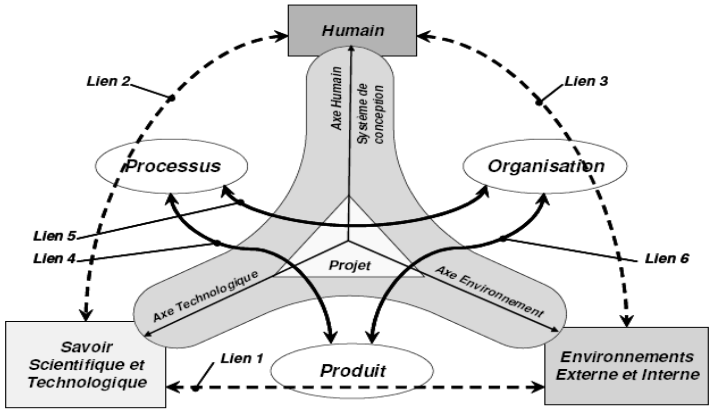


Figure 2.1 Modèle de description du système de conception, d'après [Rob05]

- aux descriptions fonctionnelles, structurelles et comportementales des éléments du produit tout au long de son cycle de vie,
- au processus suivi ou à suivre pour l'élaboration de la conception d'un produit,
- aux moyens humains, matériels et logiciels mis en œuvre pour le processus de conception,
- à l'environnement dans lequel évolue le produit, à l'organisation et la structure de l'entreprise (rôles, équipes) et les relations entre ses différentes composantes (objectifs, contraintes, ressources, etc.) dans cet environnement pour la conduite de ses projets,

– à l'interaction entre les différents acteurs du processus de conception caractérise la connaissance utilisée, générée et échangée tout au long du processus de conception.

Nous établirons ainsi la nécessité de disposer de modèles pour la description des produits manufacturés, des ressources utilisées pendant les différentes phases de conception, des processus mis en place ou envisagés pour la mise en œuvre de l'une de ces phases.

Nous donnons dans les parties qui suivent l'organisation de ces concepts dans les différents outils couramment utilisés pour définir un produit sous la forme d'un modèle dit générique, mais qui continue à évoluer au gré des développements en recherche et des applications industrielles. Nous présentons huit principaux modèles qui ont des éléments importants dans notre contexte : les modèles CAO, les outils PDM/ SGDT, le modèle FBS-PPR, le modèle STEP, le modèle CPM, le modèle produit de MOKA, le modèle CODEMO et le modèle PPO. Modèles dont les abréviations seront explicitées ci-après

2.1.2. Les modèles de conception collaborative

2.1.2.1. Les modèles CAO (Conception Assistée par Ordinateur)

De nombreux outils de CAO permettent de supporter plusieurs phases du processus de conception et de représenter les différents aspects du modèle de conception (conceptuel, détaillée, fabrication, assemblage, etc.). En CAO, l'efficacité en particulier de la modélisation géométrique est un gain appréciable, et les facilités de modification des modèles permettent d'élaborer des solutions dans des délais restreints. La CAO représente essentiellement l'aspect structurel et n'offre pas de solution pour les fonctions et comportements (FBS). Caractérisant des découpages structurels les entités CAO de type pièce, Assemblage et Feature ; les Fonctions sont éventuellement traduites par des contraintes qui lient les entités CAO ; cependant le concept de comportement demeure absent des modèles CAO. Le modèle de référence est une représentation 3D de la géométrie du produit conçu. Une fois achevé, ce modèle sert de base commune aux activités de conception. La Figure 2.2 montre que le modèle CAO est au cœur de la conception collaborative faisant intervenir plusieurs applications métiers.

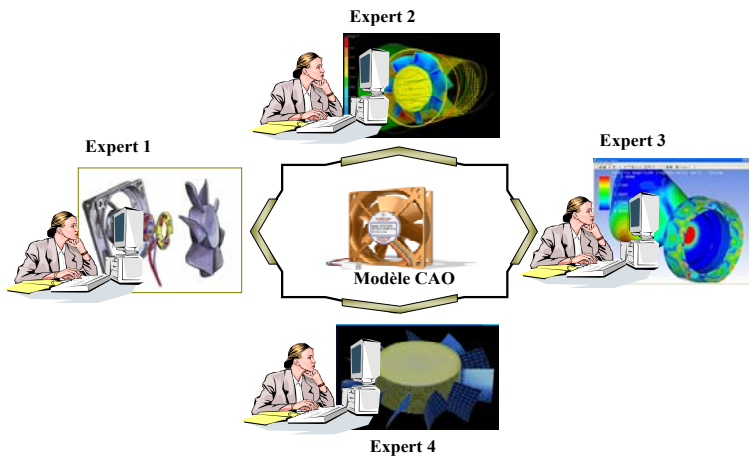


Figure 2.2 Le modèle CAO est cœur du travail collaboratif

Les éditeurs CAO semblent converger vers un choix unique d'une représentation de la géométrie essentiellement B-Rep et arbre de features. Chaque application métier est alors un outil qui doit :

- associer des informations métiers au modèle 3D,
- reconnaître les caractéristiques métiers dans le modèle géométrique,
- proposer des analyses expertes sur la base du modèle construit dans les deux étapes précédentes.

Il est alors légitime de reprocher à cette approche :

- **le choix d'un modèle de référence incomplet** : (ne définissant que de la géométrie) autour duquel de nombreuses informations et reconstruction sont souvent le préalable à chaque expertise métier,
- **la redondance des informations** : à construire dans le cadre d'expertises différentes mais partageant plus que la géométrie. Il n'est pas rare de devoir affecter le matériau à une même pièce dans des modules métiers distincts,
- **l'hétérogénéité des informations à maintenir** : le modèle de conception pour une seule expertise est généralement maintenu dans de multiples fichiers, correspondant aux diverses applications métiers, avec des formats incompatibles. En plus des formats de fichiers différents, chaque outil fonctionne dans son propre espace de travail avec ses propres commandes et procédures,
- **la nécessité de traduction des modèles** : pour un outil qui utilise des modèles créés par d'autres outils, il est en effet nécessaire d'appliquer des transformations à ces modèles pour obtenir un modèle compatible et utilisable. Ces transformations, limitées souvent à une traduction de formats de conception, nécessitent également une conversion manuelle des informations de conception. Ces traductions se traduisent souvent par des pertes d'informations, des pertes de temps et une complexité pour garder la trace de la correspondance entre les différentes représentations,
- **un manque de synchronisation entre modèles** : les modifications apportées par l'un des outils ne sont pas forcément disponibles dans d'autres outils à moins que le concepteur prenne sur lui de transformer de manière explicite ces modifications dans le bon format. En conséquence, il est difficile de propager entre les outils les changements progressifs apportés aux modèles de conception,
- **un défaut de réutilisabilité** : la réutilisation d'un modèle de conception est également rendue plus difficile car le concepteur doit extraire les différentes représentations de chaque outil de travail dans lequel elles sont définies.

Un environnement CAO efficace devrait permettre aux concepteurs d'intégrer différents outils et représentations en formalisant les relations entre ces representations.

2.1.2.2. Les outils PDM/ SGDT

Les solutions PDM (Product Data Management) [Mci95] en français SGDTs (Systèmes de Gestion des Données Techniques) [Ran95] permettent le partage des informations provenant de divers experts métiers dans une base de données unique. Les SGDTs récents (Agile, PDMWorks, Windchill, TeamCenter, SmartTeam, etc.) proposent une gestion collaborative des informations.

Les fonctionnalités des systèmes PDM se divisent généralement en deux catégories, fonctions d'utilisateurs et fonctions d'utilité, et sont résumées dans la Tableau 2-2 :

Tableau 2-2 Les principales fonctionnalités d'un PDM/SGDT

Les fonctions d'utilisateur	les fonctions d'utilité
•Gestion de données et de documents la fonctionnalité qui permet aux utilisateurs de stocker, de récupérer et de partager des documents. Ceci inclut l'autorisation d'accès aux informations, le support pour versionement/révisionement des documents/pièces, la définition et la maintenance des relations entre les pièces et les documents et d'autres données connexes. **•Gestion de workflow et de processus** Ceci inclut la support pour la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqué dans la réalisation d'un processus métier. Il permet décrire les interactions sous forme d'échange d'informations entre divers acteurs les tâches à accomplir entre les différents acteurs d'un processus, les délais à respecter, les modes de validation. **• Gestion de décomposions structurelle du produit** Il permet définir une vue unique du produit sous forme d'arbre d'assemblage/sous-assemblage/pièce. Ceci inclut les liens entre des données de définition du produit contenu dans la processus de conception et la structure du produit **• Gestion intégrée des articles et des listes de pièces** la fonctionnalité qui permet de charger et de gérer une structure de classification réutilisable pour des articles et des listes de pièces standard (par exemple, des informations relatives aux pièces et aux fournisseurs) **• Gestion de projet** Fournit des fonctions de gestion de projet pour définir un réseau hiérarchique des tâches et sous-tâches nécessaires pour compléter un projet. Cette structure peut se rapporter au données de produit, au documents et leurs modifications, ce qui permet de voir comment un projet se déroule en termes de la situation d'avancement,	**•Communications et notification** Cette fonctionnalité permet aux utilisateurs de savoir le plus rapidement possible quand un produit est prêt à être transformés a la prochaine tâche et mais aussi à travers lesquelles l'information est la plus up-to-date. La notification des actions se fait automatiquement. **•Le transfert et la traduction de données** Toutes les données sont stockées et accessibles sous le contrôle du système de PDM. Un utilisateur n'aurait pas besoin de savoir l'endroit où les données sont stockées. Le système garde la trace de l'emplacement effectif des objets reçus et permet aux utilisateurs d'y accéder, s'ils ont le droit permission. Dans un système de PDM prédéfinis traducteurs peut être utilisé pour gère la conversion des données entre les applications. **•Administration** Gestion de rôle est souvent traitée au sein de l'administration du système. Un cycle de vie des produits comporte de nombreuses fonctions propres aux besoins des entreprises et au rôle de chaque acteur qui intervient à des temps spécifiques du processus de conception. Gestion de rôle permet de visualiser l'ensemble des droits attribués aux différents rôles telles que la mise en place des comptes d'utilisateurs, des groupes et des droits d'accès. **•L'intégration d'applications** L'intégration avec d'autres applications (l'éditeur de texte, CAD / CAM et ERP systèmes) est un élément important d'un système PDM. L'objectif est de définir les différents modes de couplage entre SGDT et logiciels métier.

Le modèle produit partagé dans un SGDT se limite à une décomposition structurelle sous forme d'arbre d'articles : un article est un assemblage, un sous-assemblage ou une pièce. Cette décomposition hiérarchique est en général définie en début de projet de conception et elle est difficilement modifiée et adaptée en cours de projet. Cette décomposition hiérarchique est indépendante des modèles fonctionnels, comportementales, et géométriques.

Les SGDTs forment un cadre global pour la gestion intégrée des documents, des applications et des processus qui contribuent à la définition d'un produit. Ils permettent de gérer d'une manière "optimale" le cycle de développement d'un produit en assurant la traçabilité de tous les documents. Les SGDTs se limitent à une approche de haut niveau, les détails de la conception ne sont pas accessibles directement, ce qui réduit leur efficacité. En effet dans un contexte de conception collaborative, l'analyse des activités parallèles des différents experts ne permet pas d'identifier le détail des changements progressifs de tous les éléments de conception.

2.1.2.3. Le modèle FBS-PPRE

Labrousse [Lab08] propose le modèle FBS-PPRE (Functional Behaviour Structure – Product Process Resources, External effect) pour l'intégration du modèle produit avec le modèle processus. Le modèle FBS-PPR est construit autour des notions de produit, de processus, de ressource et d'effet extérieur (PPRE) couplées dans un modèle générique basé sur les concepts de fonction, de comportement et de structure (FBS) (Figure 2.3). Il vise à améliorer

l'intégration et la traçabilité des éléments des processus d'entreprise pour analyser et extraire les éléments réutilisables dans le processus de conception. La notion de processus permet de modéliser la dynamique des informations et permet de définir de manière cohérente les comportements des différents objets manipulés.

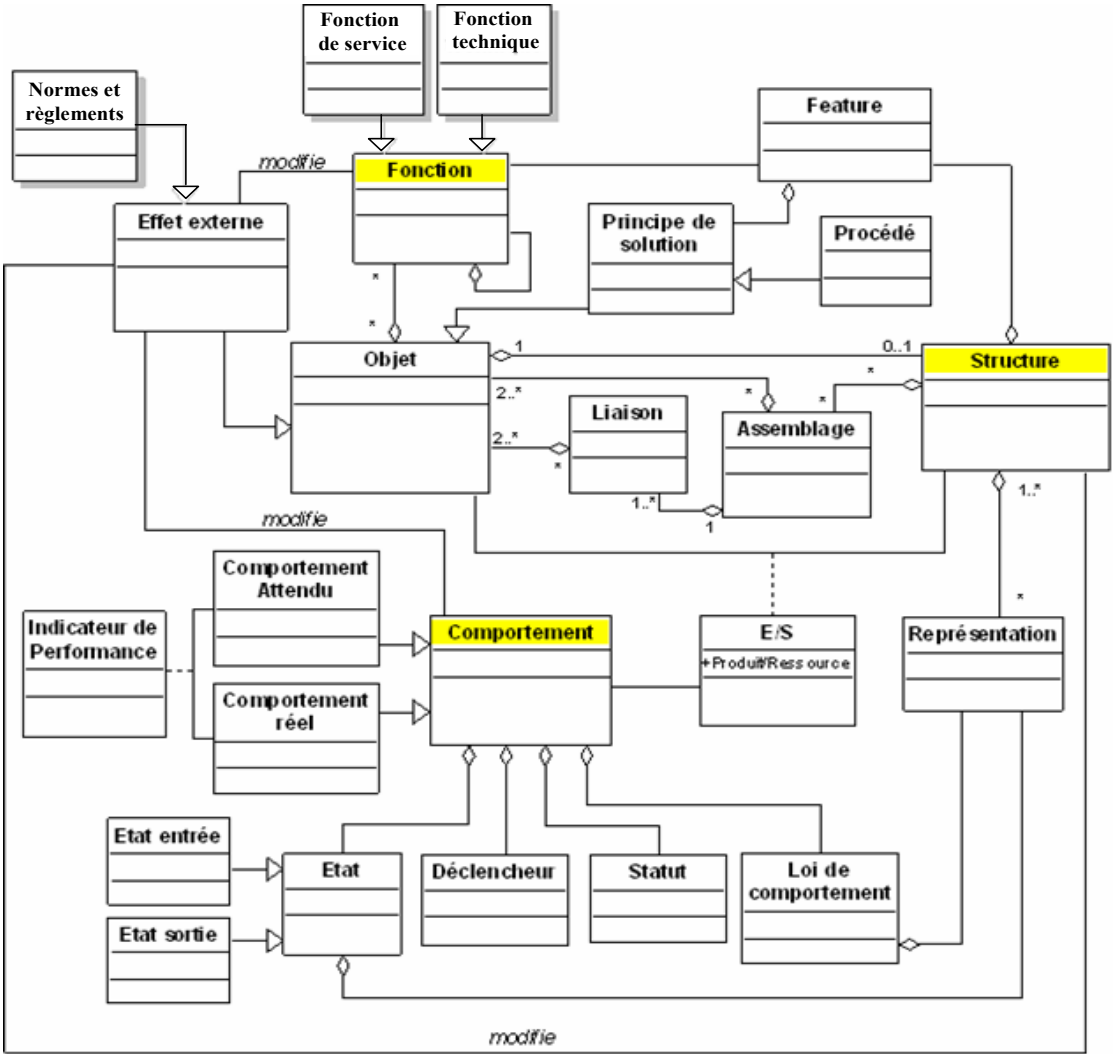


Figure 2.3 Diagramme de classe du modèle FBS-PPRE, d'après [Lab08]

Les notions connexes à FBS et intégrées dans le modèle FBS-PPRE sont :

- **état** : correspond à un instant et renvoie à des représentations de la structure.
- **indicateur de performance** : permet d'évaluer la satisfaction des objectifs (par exemple, le ratio entre les variables du comportement réalisé et du comportement attendu). Par exemple le coût, le délai, la durée de vie, etc.
- **produit, processus et ressource** : il est utile de distinguer les différents rôles pour les objets d'entreprise.
- **effet externe ou de contraintes** (normes, règlements, clients, etc.)

Le modèle FBS-PPRE ne fait pas de distinction à la base entre Produit, Processus et Ressource qui sont généralisés en « Objet ». Les concepts de produit, Processus et Ressources apparaissent alors à travers des rôles que jouent les objets les uns par rapport aux autres.

2.1.2.4. Le modèle STEP

Le modèle ou standard STEP (Standard for the Exchange of Product Model Data) [Ste08], est le premier standard international pour l'échange, la présentation et l'archivage de façon unique des modèles de données et des informations des systèmes hétérogènes (CAO (conception assistée par ordinateur), FAO (fabrication assistée par ordinateur), IAO (Ingénierie assistée par ordinateur) et PDM (Product Data Management)). L'objectif de la norme STEP est de définir une représentation non ambiguë du produit couvrant tout son cycle de vie. Pour cela, STEP propose des langages, des modèles de données et des méthodes pour définir un modèle intégré pour la gestion des informations de l'entreprise. La Figure 2.4 présente les différents aspects de la technologie STEP :

Figure 2.4 Structure de la norme STEP

- **Méthodes de description de l'information** : EXPRESS (ISO94)[1], est le langage de modélisation des données du standard STEP. Son objectif principal est de décrire les modèles d'informations dans le domaine technique, en vue de l'échange de données représentant de façon non ambiguë ces informations. Il permet la modélisation des classes d'objets (dénommées ENTITY) et les relations entre elles dans une application métier.
- **Méthodes de mise en œuvre** : ces sont des méthodes, prédéfinies et normalisées, de représentation d'instances conformes au modèle décrit par le langage EXPRESS. Les méthodes de mise en œuvre appliquent la méthode d'échange de données à travers la notion de fichier neutre une population d'instances d'entités conformes a un model défini en EXPRESS et une méthode de partage des instances basée sur la définition des fonctions d'accès normalisées (une interface d'accès sous le nom Standard Data Interface, SDAI).
- **Modèles de données** : le modèle de données regroupe trois parties : 1) les ressources intégrées : fournissent des informations génériques et indépendantes du contexte (type de produit, type de l'application, processus de fabrication, etc.) pouvant être utilisées par différentes applications pour définir un produit générique (la partie 45 de STEP : matériaux, la partie 47 : tolérances, etc.). 2) les ressources d'application : fournissent des informations relatives à un domaine particulier. Par exemple, le modèle pour les applications électriques (partie 103 de STEP). 3) les protocoles d'applications : l'ISO fournit une méthodologie pour intégrer les besoins de l'utilisateur par spécialisation des données techniques de son métier. Cette méthodologie impose le passage par les phases de spécification du besoin, de finition de l'AAM (Application Activity Model), de définition de l'ARM (Application Reference Model) et de définition de l'AIM (Application Interpreted Model) pour donner la définition EXPRESS du protocole d'application dans sa globalité. La Figure 2.5 liste quelques protocoles d'applications et montre la diversité des domaines applicatifs couverts.

[1] ISO: International Organization for Standardization

- **Test de conformité** : l'ISO propose des normes définissant les tests de référence pour les difficultés rencontrées dans les interfaces du protocole d'application. Il définit pour chaque API (Application Programming Interface), une politique de tests pour les interfaces et les critères pour établir leur conformité.

Partie 201 : Dessin technique explicite
Partie 202 : Dessin technique associatif.
Partie 203 : Conception mécanique 3D avec gestion de configuration
Partie 207 : Conception des outils et gammes pour l'emboutissage
Partie 208 : Processus de modification, du cycle de vie d'un produit.
Partie 209 : Analyse par éléments finis de structures métalliques et composites
Partie 213 : Plans de procédés de contrôle numérique pour gammes d'usinage
Partie 214 : Données pour la conception mécanique d'automobiles
Partie 219 : Gammes pour machines à mesurer
Partie 222 : Conception et fabrication de structures composites
Partie 223 : Conception et fabrication de pièces moulées
Partie 224 : Formes fonctionnelles utilisées lors de l'élaboration de gammes de fabrication.

Figure 2.5 Quelques protocoles d'applications du standard STEP

Le modèle STEP est très étendu. Il n'est pas envisageable que des concepteurs rédigent des modèles directement en STEP. Il est donc réserver à des exports depuis des applications logicielles qui embarquent un driver STEP.

2.1.2.5. Le modèle CPM

Le NIST[2] propose un modèle générique, dit Core Product Model (CPM) [Sud05a], qui contient les éléments fondamentaux pour représenter tout produit. Le CPM a été conçu comme une représentation des informations relatives au développement de produits et peut être vu comme une base pour le développement de la nouvelle génération d'outils de CAO. Ce modèle fournit une représentation de produits manufacturiers (Artifacts) contenant une large variété de concepts d'ingénierie de produits au delà de la géométrie. Cette représentation inclut les notions de forme, fonction, comportement, matériaux, décompositions physique et fonctionnelle, la mise en correspondance entre les concepts de fonction et de forme ainsi que plusieurs autres relations entre les différents concepts.

Le CPM est défini par un diagramme de classe UML qui représente le produit. Comme illustré par la Figure 2.6, on constate cinq catégories de classes :

- **Core Product Model** : représente le plus haut niveau d'abstractions.
- **Common Core Object** : est la classe de base pour toutes les classes d'objets qui fournissent la couche de base du modèle.
- **Common Core Relationship** : décrit les associations entre entités.
- **Core Entity** : représente des entités réelles comme Artifact, Form, Feature.
- **Core Property** : est la classe de base à partir de laquelle Fonction, Flux, Forme, Géométrie et Matériaux sont spécialisés.

[2] NIST: National Institute of Standards and Technology

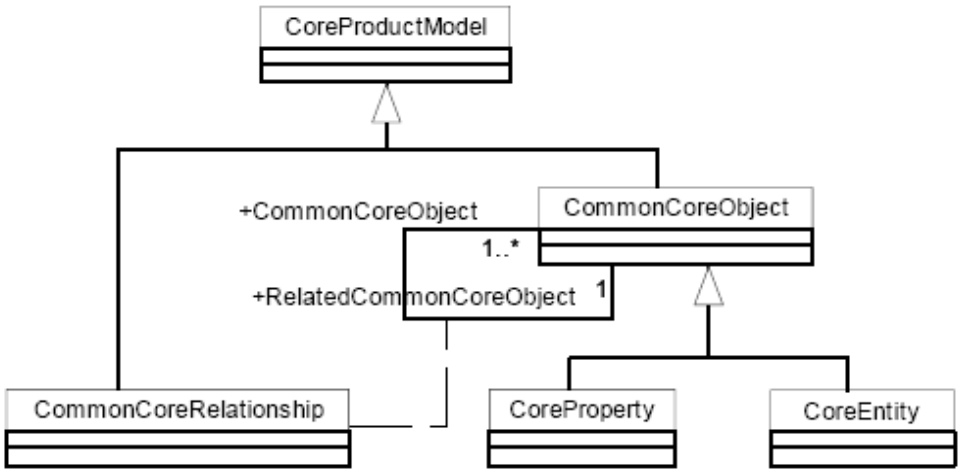


Figure 2.6 Catégories de classe du modèle CPM [Sud05a]

La Figure 2.7 montre le diagramme UML étendu du modèle CPM. Le modèle met en place, d'une manière plus ou moins explicite, les concepts de Fonction, Structure et Comportement :

- la **Fonction** est représentée par la classe « Function ». Le modèle met en place une forme particulière de fonction appelée fonction de transfert de flux (fluide, énergie, information, etc.). Elle est représentée par la classe Transfer Function, ainsi il peut s'agir soit d'une transmission ou d'une conversion de flux entre deux bornes entrée et sortie.
- le **Comportement** est représenté par la classe « Behavior » qui représente comment un « Artefact » réalise sa « Fonction ». Ce comportement observé est souvent évalué et analysé pour vérifier que l'Artefact assure bien sa Fonction.
- la **Structure** du produit est définie conjointement à partir des notions d' « Artefact », de « Feature » et de « Form ». Un « Artefact » représente une entité distincte du produit, qui peut être un « Composant » un « Part », un « Sous-assemblage » ou un « Assemblage ». Une « Feature » est une portion de la forme globale d'un Artefact à laquelle est attachée une « Fonction » spécifique. Ainsi, un « Artefact » peut avoir des « Features » de conception, des « Features » d'analyse, des « Features » de fabrication, etc. Le type de chaque « Feature » dépend de la « Fonction » qu'elle remplie. La forme (Form) définit les caractéristiques physiques d'un « Artefact », qui sont représentées en termes de propriétés géométriques (description spatiale) et physiques (matériaux).

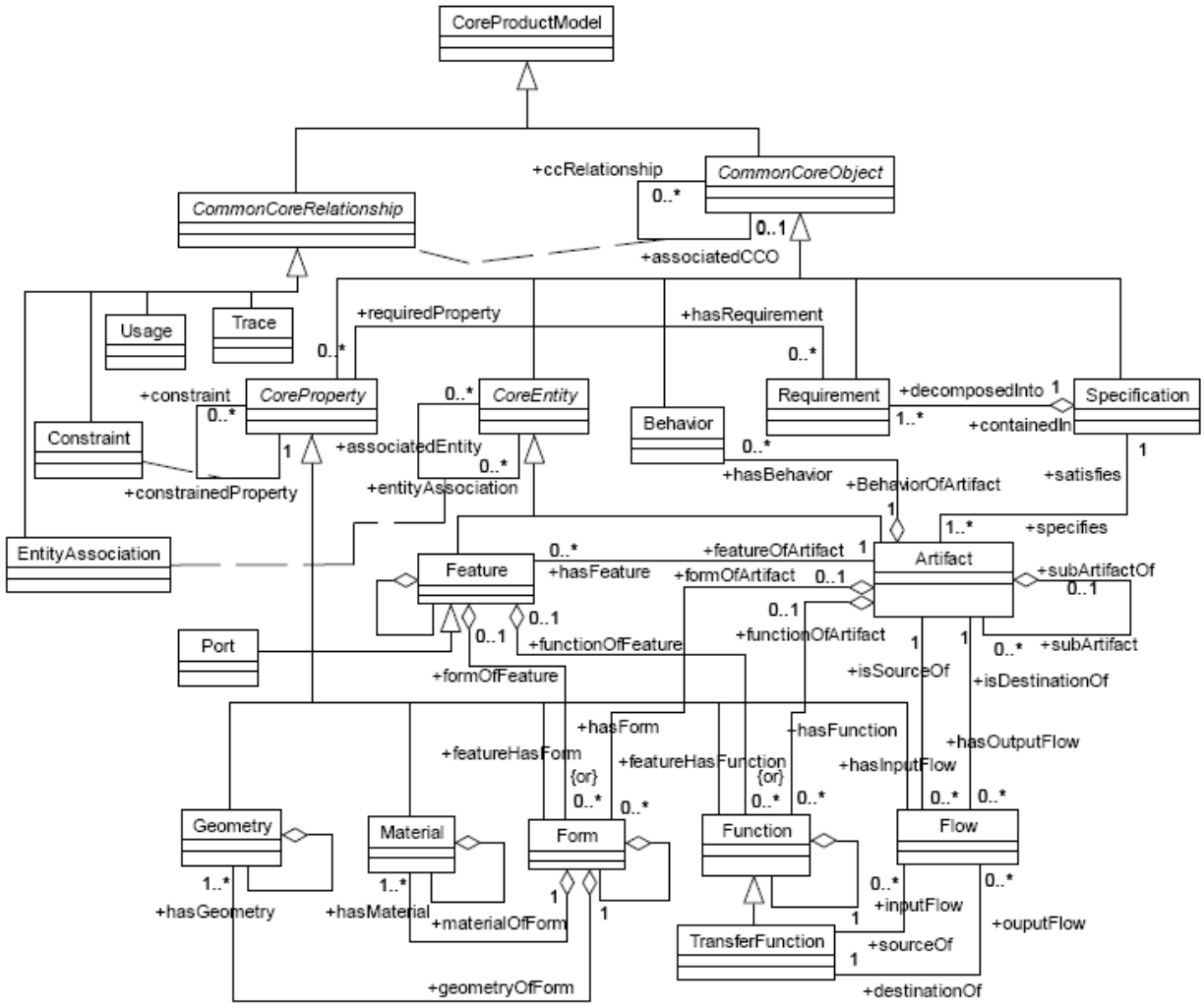


Figure 2.7 Diagramme de classe du modèle CPM d'après [Sud05a]

Des extensions du modèle CPM ont été développées. Nous pouvons citer :

- Open Assembly Model (OAM) pour représenter les données d'assemblage et pour faciliter leurs échanges [Sud05b].
- Product Semantic Representation Language (PSRL) pour le développement d'une représentation formelle des informations sur le produit [Pat05].
- Design-Analysis Integration projet pour une architecture conceptuelle de données pour l'intégration spatiale et fonctionnelle de la conception et pour supporter la conception basée sur la conception orientée analyse (analysis-driven design) et l'analyse opportuniste (opportunistic analysis) [Fen03].
- Product Family Evolution Model pour la représentation de l'évolution des familles de produits et la prise en compte de la rationalité des modifications [Wan03].

Le CPM fournit un modèle de base pour la gestion produit, ce modèle est simple, générique, non-propriétaire, indépendant de tout processus de développement de produit, non lié à un domaine particulier, facile à étendre vers des domaines spécifiques et capable de capturer une large part du flux de données partagées tout le long du cycle de vie du produit.

2.1.2.6. Le modèle produit de MOKA

MOKA [Sto01] (Methodology and tools Oriented to Knowledge based engineering Applications) est une méthode issue d'un projet européen ESPRIT/AIT[3]. Le modèle MOKA partage les motivations du modèle CPM et exploite STEP, KIF (Knowledge Intergange Format) et les efforts liés à la création d'un modèle intégré de produit pour explorer des applications de type KBE (Knowledge Based Engineering). Le langage de modélisation MOKA, basé sur le formalisme UML, est créé pour représenter l'ingénierie des connaissances au niveau des utilisateurs afin de les déployer dans des applications KBE comme le prototype des modèles spécifiques à un domaine d'application.

Le modèle produit de MOKA, représenté par la Figure 2.8, s'appuie sur cinq vues du produit :

- **la vue structure** définit une décomposition hiérarchique de la structure du produit sous forme de pièces, d'assemblages et d'un ensemble de caractéristiques (Features). La structure peut être physique, logique ou conceptuelle dans n'importe quel étape de la conception ;
- **la vue fonction** définit la décomposition fonctionnelle du produit et les principes de la solution de conception ;
- **la vue comportement** interprète un modèle d'état des divers états du produit et de la transition d'un état à un autre ;
- **la vue technologie** interprète les connaissances concernant une technologie associée au cycle de vie de produit. Celles-ci comprennent les informations sur le processus de fabrication et les autres processus qui interviennent à tout moment du cycle de vie.
- **la vue représentation** interprète toute autre connaissance concernant la géométrie de la structure (la dimension, la forme et le positionnement) et la représentation alternative de la structure du produit.

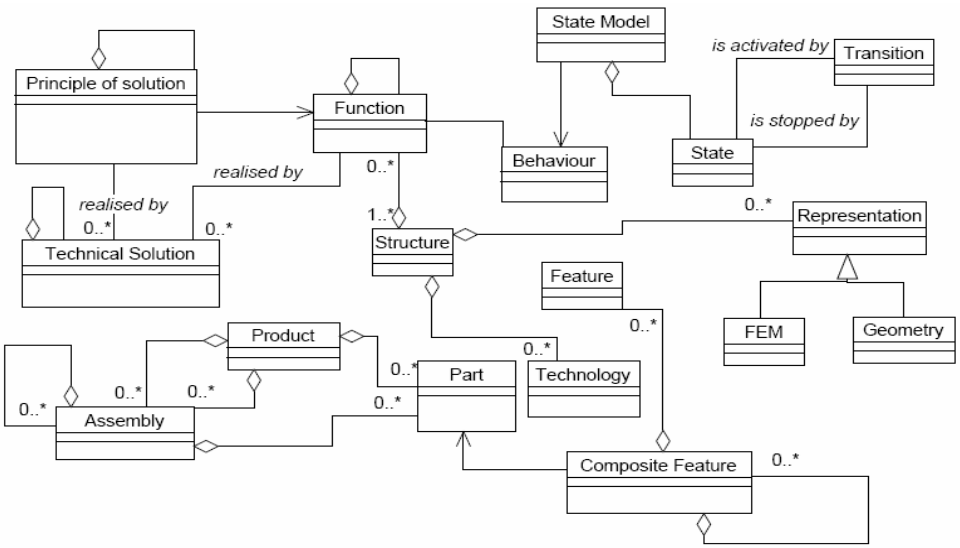


Figure 2.8 Le méta-modèle produit de MOKA [Sto01]

[3] Projet européen ESPRIT : en 1998, les objectifs on été de définir une méthodologie de réalisation pour des applications KBE (Knowledge Based Engineering)

Le projet MOKA propose également un modèle de processus de conception moins structuré. Ce modèle inclut les catégories de connaissances extraites des experts qui sont renseignées dans des fiches appliquées nommés ICARE (Illustration, Contrainte, Activité, Règle, Entité). Une fiche ICARE définit les éléments de connaissances comme suit (Figure 2.9) :

- **Illustration** : permet d'enregistrer les expériences déjà vécues ou des expériences particulières,
- **Contrainte** : permet d'enregistrer les éventuelles limitations imposées sur le produit,
- **Activité** : permet de décrire le processus de conception,
- **Règle** : permet de traduire les contraintes dans le processus de conception.
- **Entité** : elle permet de décrire le produit. Elle peut être structurelle ou fonctionnelle.

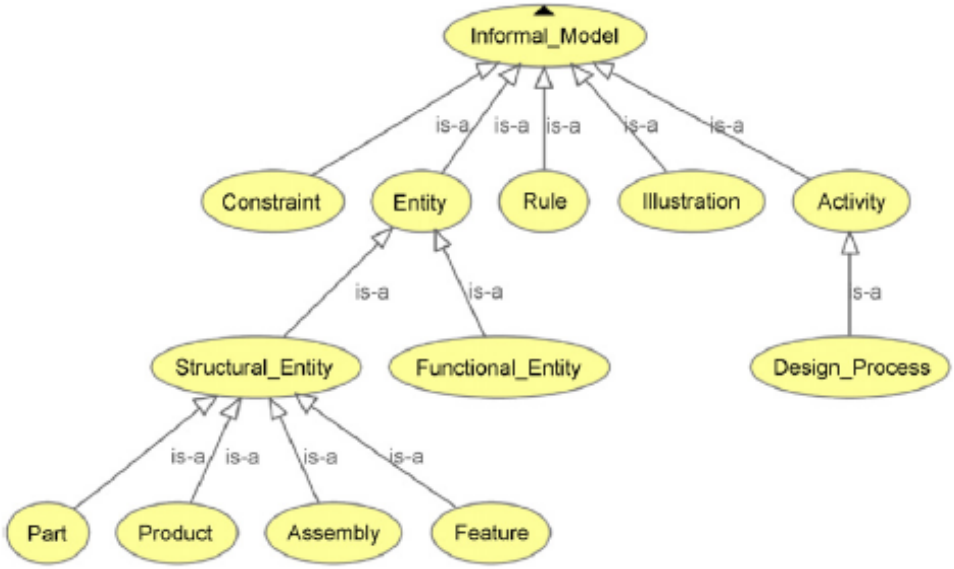


Figure 2.9 Le modèle informel ICARE de MOKA [Sto01]

Le modèle produit est basé sur les fiches Entités et Contraintes. Les liens hiérarchiques entre les instances de chaque vue sont déterminés en analysant les relations qui relient les fiches entre elles.

2.1.2.7. Le modèle du système CODEMO

Le système CODEMO (Co-Operative DEsign MOdeler) [Rou99] est un support pour la conception coopérative intégrée dans un environnement informatique favorisant la conception assistée par ordinateur afin de créer des décompositions multi-vues du produit en cours de conception.

La Figure 2.10 montre l'architecture du modeleur CODEMO.

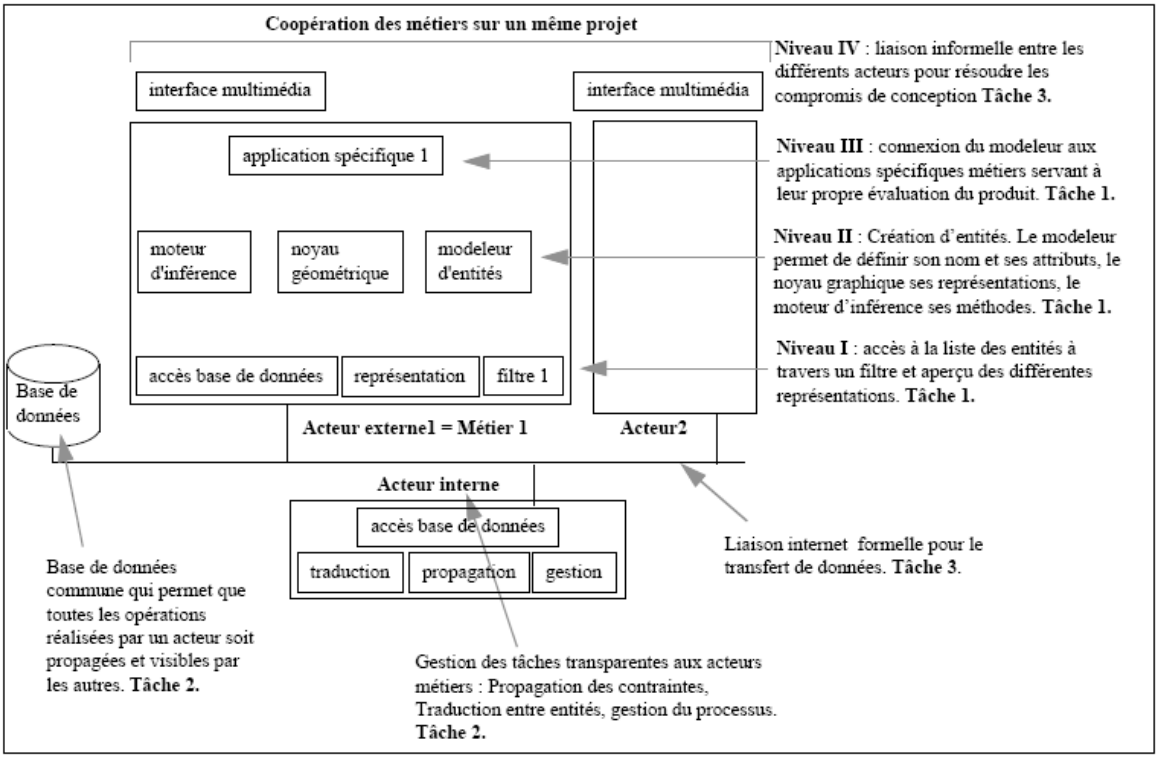


Figure 2.10 Architecture du modeleur de conception intégrée CODEMO [Rou99]

Le modeleur fournit une interface graphique pour créer de manière dynamique un Modèle Produit, ses vues Technologique, Ossature, Géométrique et Métier [Rou99] :

- La vue **Technologique** représente la décomposition structurelle du produit déduite du modèle fonctionnel du produit.
- La vue **Ossature** représente le produit par ses surfaces fonctionnelles.
- La vue **Géométrique** se construit à partir des deux vues Technologique et Ossature.
- La vue **Métier** représente les connaissances des acteurs métier sur le produit en créant leur propre vue.

Dans chaque vue, les acteurs ont la possibilité de voir la structure des données à travers les décompositions multi-vues. Ces données peuvent être lues et modifiées pour faire évoluer la conception, et assurer ainsi la compréhension du produit par les acteurs.

Le modeleur de conception intégrée CODEMO permet d'avoir plusieurs représentations des données du produit (Figure 2.11), entre autre une représentation graphique, pour que chaque acteur en ait une vision compréhensible. Le modeleur fournit une base de données commune qui permet que toutes les opérations réalisées par un acteur soit propagées et visibles par les autres. Il gère la cohérence du modèle de données (composants, liens et relations) et de la connaissance qui lui est associée. Ceci consiste à propager les contraintes dans le Modèle Produit et à faire apparaître les compromis lorsqu'une relation est violée.

Les acteurs se connectent sur un projet de conception particulier à partir de leur propre plate-forme de travail et y introduisent leurs propres données et leurs propres contraintes issues de l'évaluation qu'ils en ont faite avec leurs outils spécifiques.

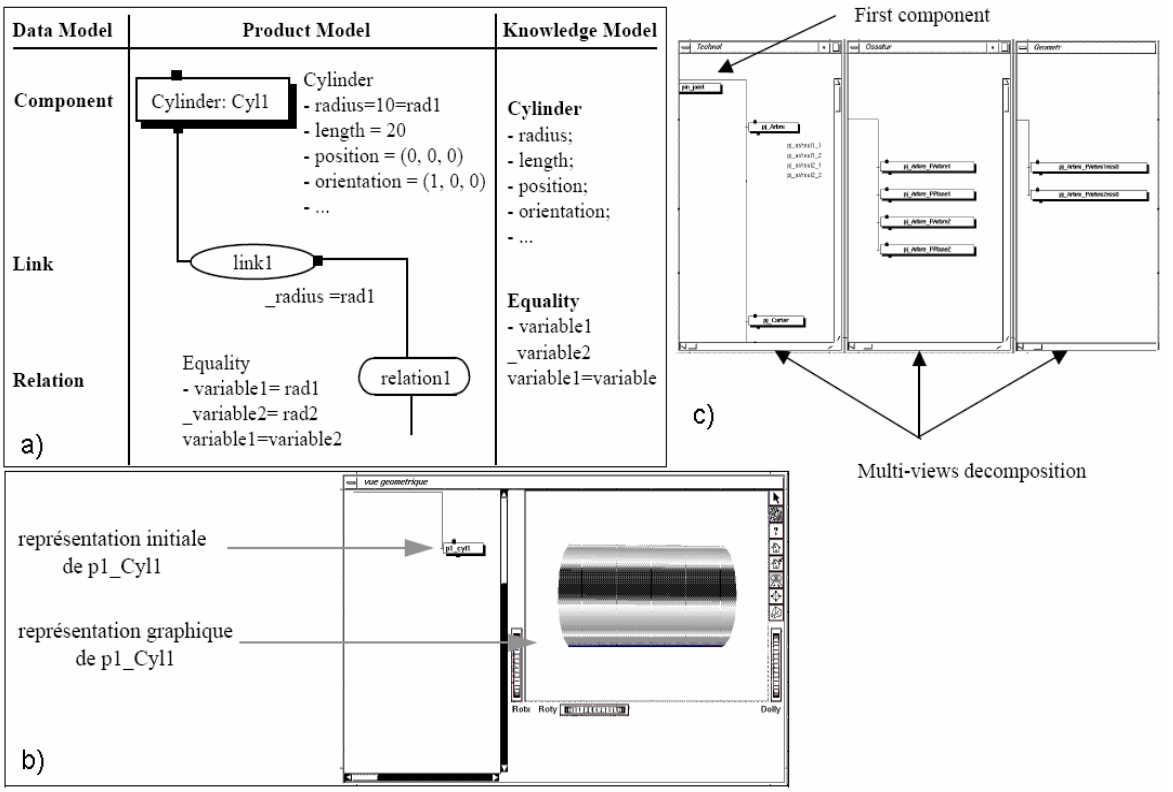


Figure 2.11 Décompositions multi-vues dans CODEMO

En résumé, le modeleur de conception intégrée CODEMO doit s'insérer dans une plateforme plateau projet afin d'assister les différents acteurs dans la gestion collaborative des informations de conception.

2.1.2.8. Le modèle PPO

Le modèle PPO (Produit Processus Organisation) a été développé dans le cadre du projet IPPOP[4] [Ipp08]. Le modèle PPO est défini comme un modèle commun pour lier les modèles existants pour la conception. Il intègre des modèles Produit, Processus et Organisation (Figure 2.12).

Le modèle PPO a deux caractéristiques principales :

– c'est en modèle simple qui traite des concepts de base pour la modélisation. Son diagramme de classe, illustré par la Figure 2.12, montre : (i) quatre classes non abstraites pour la définition du produit : la classe Composant, la classe Interface, la classe Fonction et la classe Comportement ; (ii) quatre classes non abstraites pour la définition de la planification de projet de conception : la classe Projet, la classe Acteur, la classe Matériel, la classe Logiciel et la classe Information ; (iii) trois classes sont utilisées pour définir l'organisation et les capacités de l'entreprise : la classe Centre de Décision, la classe Cadre de Décision et la classe Cadre de Conception. Les trois sous modèles (Produit, Processus et Organisation) sont principalement intégrés autour de la classe Projet

[4] IPPOP : Le projet IPPOP (Intégration Produit – Processus - Organisation pour l'amélioration de la Performance en ingénierie) est supporté par les ministères de l'industrie et de la Recherche dans le cadre du Réseau National des Technologies Logiciels. Il a été labellisé le 10 décembre 2001 pour une durée de 36 mois et a eu pour objectif d'intégrer les connaissances liées au produit et au processus pour contribuer à l'augmentation du patrimoine technologique de l'entreprise et à la maîtrise de la conduite de l'activité de conception.

– c'est un modèle extensible, il est possible de le spécialiser, pour être un langage de partage et d'échange d'informations tout au long du processus de développement du produit. Chaque expert peut compléter l'ensemble des attributs de chaque classe des sous modèles Produit, Processus ou Organisation pour avoir une description détaillée de tout le modèle et adaptée à ses besoins de partage.

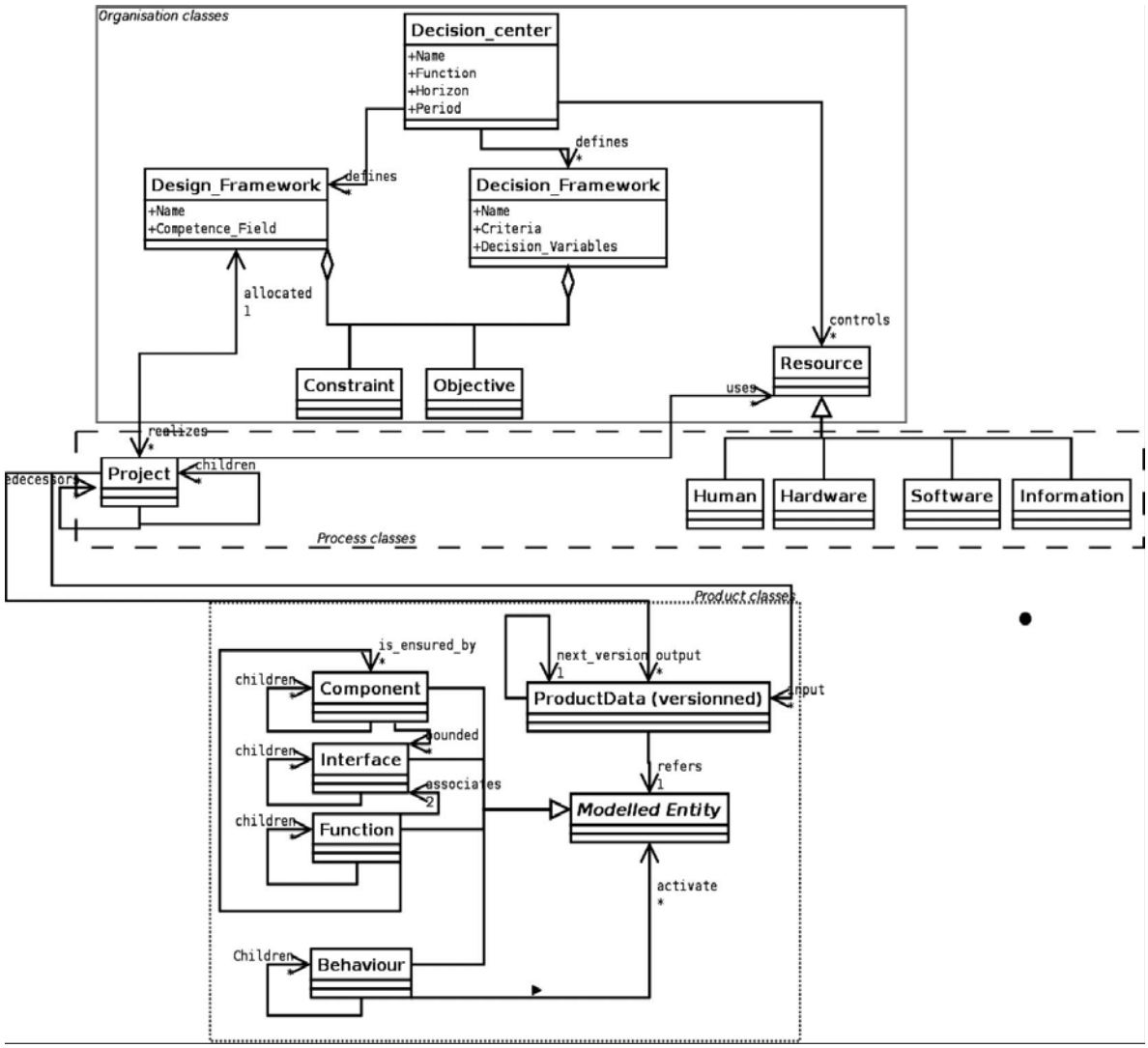


Figure 2.12 Diagramme de classes du modèle PPO [Noe07]

Dans nos travaux, nous utiliserons le sous-modèle Produit du modèle PPO. Le modèle produit de PPO permet la représentation du produit à l'aide d'entités de base : Composant, Interface, Fonction et Comportement avec des qualificatifs qui permettent d'envisager la description de familles de produit (Alternative) mais aussi de différencier les données partagées par l'ensemble des acteurs (Commun) de celles spécifiques à un métier (Vue). La Figure 2.13 décrit le diagramme de classes du modèle produit de PPO.

– un **composant** permet de donner une représentation structurelle du produit, il peut être divisé en sous-composant pour présenter un assemblage, un sous-assemblage ou une pièce.

– une **interface** est le lien entre un composant et un élément extérieur (interface d'un autre composant, fonction, etc.)

- une **fonction** représente les relations entre composants via leurs interfaces, elle définit un objectif à atteindre par le produit à concevoir
- un **comportement** définit l'état du modèle dans les différentes phases du cycle de vie du produit. Un comportement est défini par déclaration des composants, interfaces et fonctions.

Le modèle produit de PPO est une extension du modèle « composant » lien relation issus de CODEMO. La description de chaque entité peut être affinée par un qualificatif : « Alternative », « Commun » ou « Vue ». Un Composant Commun (CC) est un composant dont la description est la même pour tous les experts de conception, un Composant Alternative (AC) est décomposé en composantes alternatifs, et un Composant Vue (VC) est décomposé en plusieurs représentations spécifiques des expertises différentes.

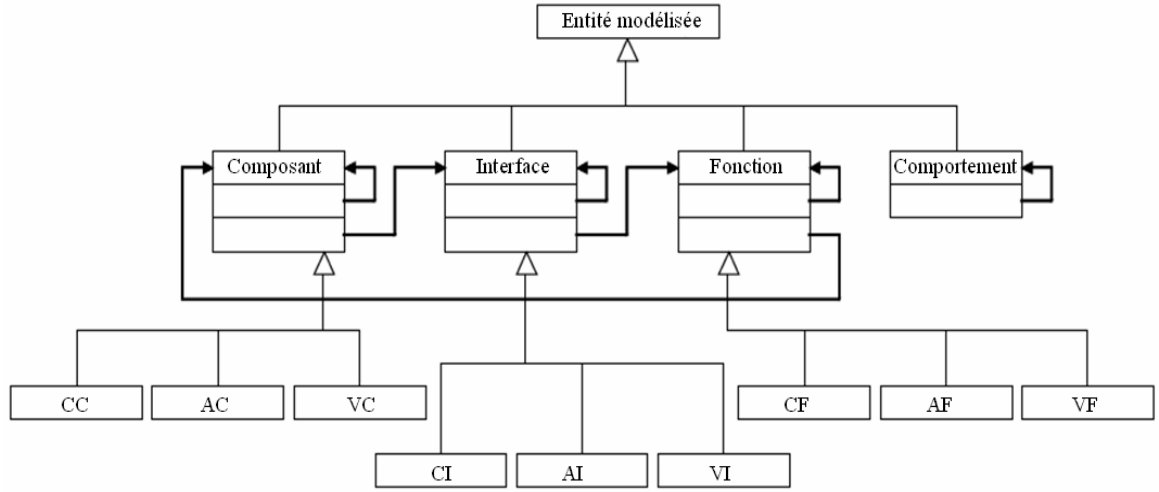


Figure 2.13 Digramme de classes du modèle produit dans PPO

L'usage de modèle produit de PPO est adapté au partage d'informations dans des conceptions innovantes ou novelles pour l'entreprise où la structure du produit n'est pas figée à l'origine du projet.

2.1.3. Synthèse des principaux modèles de conception collaborative

La présentation dans ce chapitre d'un ensemble de modèles a été l'occasion d'exposer les concepts couramment utilisés en conception collaborative. Maintenant, nous mettons en évidence les principales dimensions de développement de ces modèles.

La plupart des modèles décrivent la représentation des produits à travers trois descriptions : structurelle, fonctionnelle et comportementale. Néanmoins chaque cadre de modélisation utilise ses propres concepts internes.

Nous remarquons également que les modèles et outils décrits précédemment couvrent globalement trois dimensions de l'activité de conception collaborative :

- la dimension activités et conduite de projet (processus) qui vise l'élaboration du modèle produit : des activités sont caractérisées par les données entrées, les résultats qu'elles doivent fournir en sortie, des ressources nécessaires à leur accomplissement, des conditions et contraintes à satisfaire, etc.

- la dimension organisation de l'entreprise (rôles, équipes, etc.) qui fait référence à la description des ressources comme moyens utilisés par l'entreprise pour la conduite de ses projets. La conduite du processus de conception ne peut réussir sans une mobilisation suffisante et bien maîtrisée des efforts et des ressources de l'organisation.
- la dimension intégration des données produit, intégration du processus et intégration des multipoints de vue métiers ainsi que transformation et échanges des informations entre les différents métiers en décrivant le déroulement du développement du produit.

Nous synthétisons ces dimensions en trois grandes catégories que nous détaillons par la suite : 1) Modèles intégrés de conception, 2) Multipoints de vue métiers et modèles de représentation du produit et 3) Prise en compte de l'ensemble des modèles produit, processus et organisation. Ensuite nous abordons les solutions technologiques supports à la conception collaborative.

2.1.3.1. Modèles intégrés de conception

Avant d'introduire la notion de conception intégrée, nous définirons les principes de l'ingénierie séquentielle et simultanée [Bou94] [Leb03] à l'origine de l'ingénierie intégrée.

La conception séquentielle est basée sur une décomposition séquentielle du processus de conception, centrée autour de la notion de phase (Figure 2.14). Chaque phase correspond à un métier mis en œuvre par certaines activités menées par leurs acteurs pour la résolution d'un problème de conception. Les résultats d'une phase (ses outputs) sont le point de départ pour la phase suivante (ses inputs). En effet, le déroulement effectif du processus de conception selon un modèle séquentiel ne pourrait avoir lieu sans tenir compte des contraintes métiers des phases suivantes. Par conséquent ce modèle conduit à un problème de communication entre les différentes phases lorsqu'il y a incompatibilité entre le produit d'une phase et le point de départ de la suivante. Dans ce contexte, le cycle de développement du produit itère des boucles de retro-conception, ce qui n'assure pas son accélération.

Afin d'anticiper et de réduire d'éventuels problèmes de conception séquentielle, les nouvelles organisations proposent ingénierie simultanée (Figure 2.14). Ce modèle permet également de prendre en compte simultanément les contraintes relatives à l'ensemble des phases. Ce modèle se base sur l'hypothèse d'un déroulement parallèle des activités de conception. Sont également définis et pris en compte simultanément la fabrication du produit, son système de production, sa maintenance et tous les éléments de son cycle de vie. La communication prescriptive entre phases, dans le cas d'une conception simultanée, correspond à des revues de projets. La représentation du processus de conception, selon le modèle simultané, permet de réduire le cycle de développement du produit. Les revues de projet sont les seules moyens d'assurer la convergence des phases vers un objectif commun. Cette nouvelle organisation du travail oblige les acteurs du projet à communiquer et à échanger des informations avec des collaborateurs qui peuvent être de métiers et de cultures différents.

La diversité des environnements utilisés d'une part et les problèmes de transfert et d'échanges de connaissances et de données d'autre part, nécessite l'intégration des différentes phases, afin que chacune d'elle puisse tenir compte des éléments issus des autres et favoriser ainsi la collaboration, le pilotage des échanges et le maintien de la cohérence entre phases. Il s'agit de formaliser les connaissances propres aux métiers associés aux phases, afin que les acteurs de la conception soient à même de coopérer et d'intégrer leurs connaissances dans l'exécution de leurs tâches, et donc de prendre en compte les contraintes issues d'autres phases.

Dans la littérature, on rencontre de nombreuses définitions pour le concept de conception simultanée, ou ingénierie concourante.

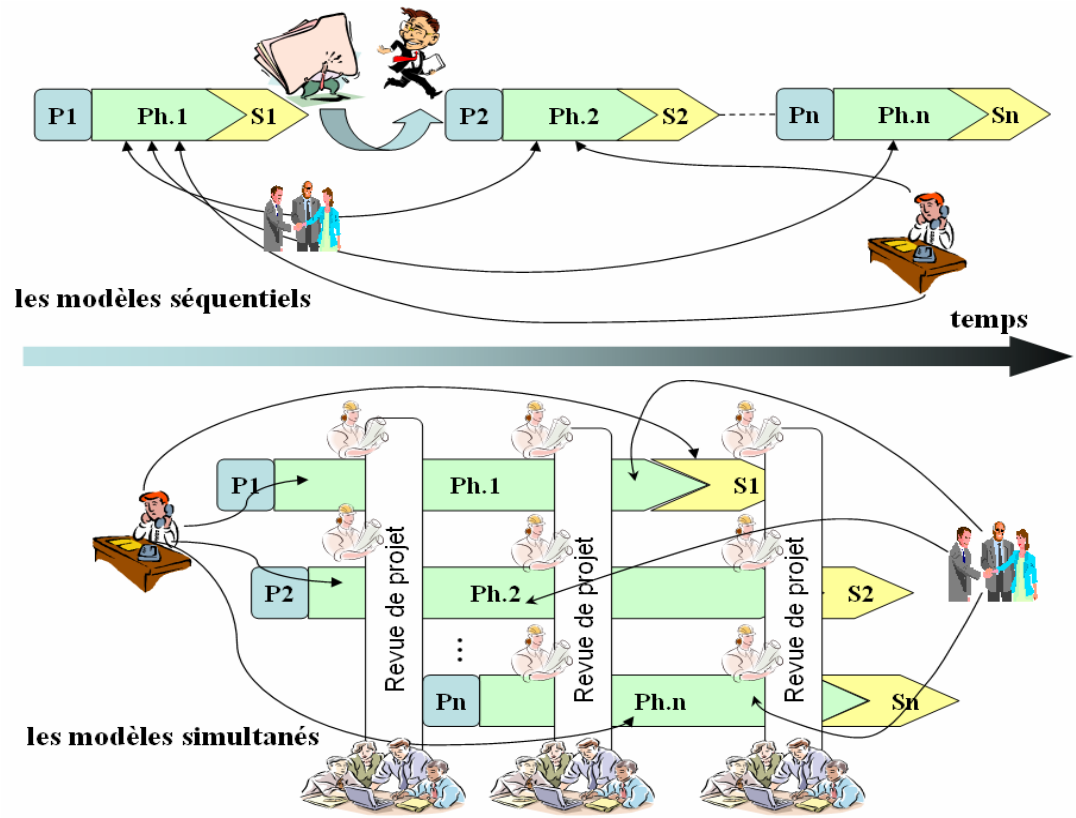


Figure 2.14 Les modèles séquentiels et simultanés

L'ingénierie concourante est proposée comme la solution permettant d'achever plus rapidement et efficacement des processus de développement de produits innovants. Deux notions apparaissent au centre de la philosophie de l'ingénierie concourante : l'intégration et la concurrence [Bad00].

Ceci conduit à un processus de conception intégrée [Tic97] qui envisage la réalisation d'un processus de conception dans un contexte d'ingénierie concourante [Lan96]. Il met en œuvre l'intégration d'acteurs différents, qui coopèrent pour atteindre un objectif commun. Dans ce contexte, les différents aspects d'un problème de conception sont modélisés et résolus par différents acteurs, chacun d'eux intervient en utilisant son propre système d'expertise. Dans la résolution de leurs problèmes de conception, les concepteurs utilisent différentes ressources (modèles, applications, outils logiciels, règles, données, etc.) qui sont intrinsèquement hétérogènes. Les ressources d'ingénierie étant utilisées à des emplacements géographiquement distribués, ces ressources hétérogènes doivent être intégrées pour faciliter l'interopérabilité et la collaboration dans un environnement distribué.

Grâce aux avancées en informatique réseaux et en technologie d'informations distribuées, il est devenu communément admis que l'activité de conception peut être distribuée à travers des environnements réseaux. Ces réseaux informatiques sont alors utilisés comme des éléments facilitateurs des activités collaboratives de conception. Toutefois, la distribution des problèmes de conception et des organisations à travers des réseaux informatiques, est souvent vécue par les concepteurs comme une contrainte complémentaire.

Plusieurs solutions informatiques allant dans cette direction ont été proposées. Dans [Noe07], les auteurs constatent que les modèles et solutions informatiques intégrés actuels doivent :

- fournir l'information à la bonne personne et au bon moment. Optimiser le flux d'information entre les différentes phases de la conception devient un point clé pour réagir et participer efficacement à la conception d'un nouveau produit complexe.
- éviter la perte d'information et/ou la mauvaise compréhension des besoins utilisateurs. Une bonne structuration des données joue un rôle crucial dans la mise en place d'un mécanisme viable qui permet de garder une trace des informations relatives au produit.
- éviter la redondance de données, faciliter et automatiser leur gestion en couplant disponibilité, efficacité et sécurité. Bien que travaillant à la conception du même produit, les informations ne sont pas formalisées et sauvegardées de la même manière par tous les ingénieurs. Fournir un modèle de référence partagé par tout le monde dans l'entreprise est une solution courante pour éviter la redondance et l'incompatibilité des informations.
- assurer la cohérence lors de l'intégration et de l'évolution du processus de conception. Eviter la redondance concourt à cet objectif mais ne s'avère pas suffisant. La gestion de la cohérence ne peut simplement s'appuyer sur la création d'un modèle commun unique. Des conflits peuvent se produire entre certaines définitions du produit chaque fois que des règles métier ne sont pas formalisées pour relier certains paramètres de la définition du produit. L'absence d'expression de ces règles donne l'impression que ces paramètres sont déconnectés et les systèmes informatiques ne les prennent pas en compte.

2.1.3.2. Multiplicité des points de vues métiers et modèles de représentation du produit

Chaque acteur de la conception développe une vue du produit et travaille sur des données représentant un aspect particulier. La Figure 2.15 montre un exemple du produit « Ventilateur d'ordinateur » selon plusieurs vues et modèles de représentation : chaque modèle décrit un point de vue particulier du produit à réaliser.

Selon [Cha97], une vue métier est générée par la description du produit faite en appliquant le savoir-faire et en prenant en compte les règles d'un métier particulier. En effet, la vision d'un objet de conception est tout à fait subjective est dépend des préjugés, de la culture de chacun et certainement de plusieurs autres facteurs difficilement identifiables. Une vue peut être de deux natures, soit commune à plusieurs acteurs d'une même expertise ou à des expertises différentes, ou propre à un acteur particulier / expertise donnée.

Dans [Bel94], les auteurs associent les points de vue relativement aux expertises métier qui interviennent au cours du cycle de vie de produit et listent des expertises fondamentales pour la conception intégrée : point de vue cinématique, point de vue technologue, point de vue calcul, point de vue fabrication, montage et maintenance, point de vue commercial, sécurité et utilisateur final, etc.

Au niveau informationnel, le point de vue d'un acteur sur un objet technique se traduit par l'expression de ses besoins en information sur l'objet pour réaliser une fonction du produit. Ainsi, les difficultés résident dans la prise en compte des points de vue des acteurs lors de la conception du modèle conceptuel afin qu'il soit possible d'élaborer les vues répondant aux besoins de chaque acteur.

La modélisation des points de vue permet, en se greffant sur un modèle unique d'éviter l'élaboration de plusieurs modèles indépendants. L'objectif est la gestion des points de vue dans le modèle global représentant les besoins utilisateurs en information sur l'objet technique. Cela afin de palier d'une part aux problèmes d'analyse et de conception puis d'évolution d'un modèle centralisé multipoints de vue de l'objet technique. Tout cela permet

la traçabilité de l'évolution de ce modèle au regard de l'apparition des différents métiers tout au long du cycle de vie du produit. De plus, d'un point de vue informationnel, l'approche multi-points de vue apporte les intérêts suivants :

- la centralisation des connaissances et des règles métiers.
- la cohérence et non redondance des modèles de la conception d'un produit.

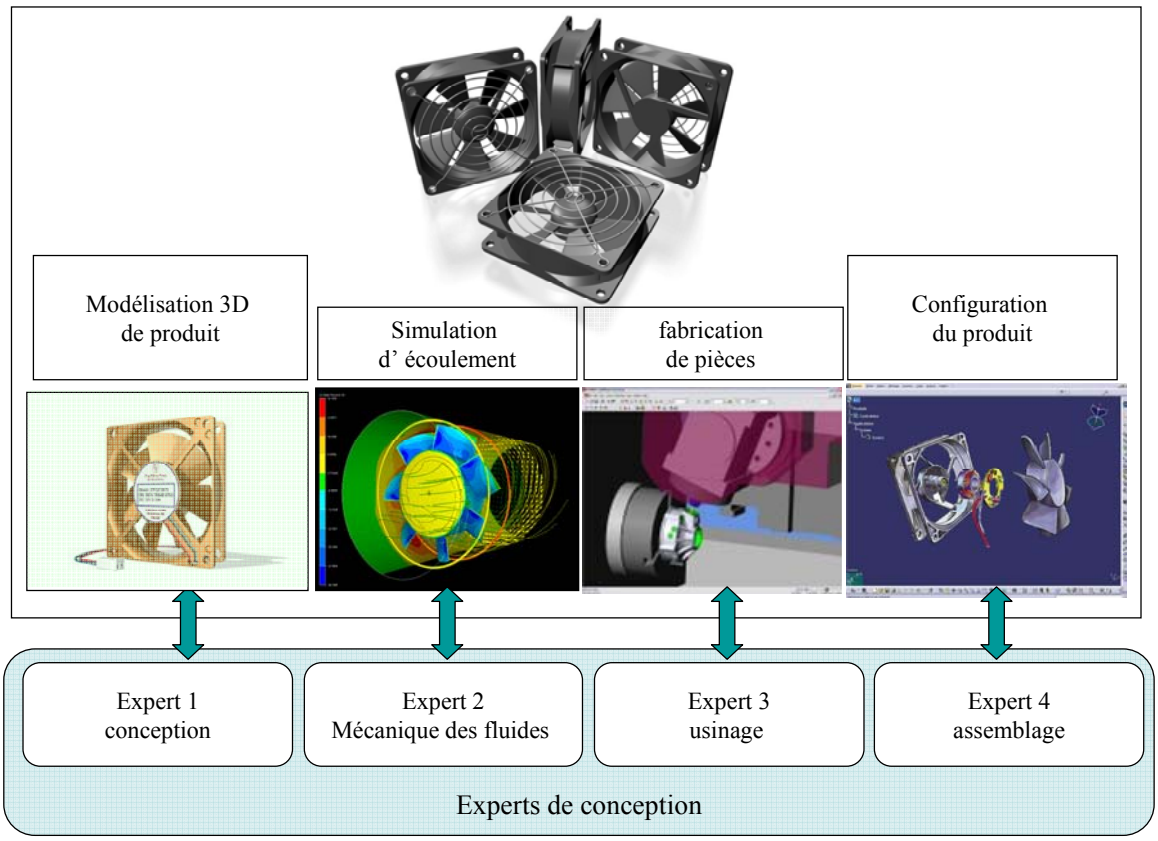


Figure 2.15 Points de vue métiers et modèles de représentation d'un ventilateur

La problématique d'un système de conception collaborative réside dans l'intégration, au plus tôt, dans les processus de développement de produit de la dimension métier en situation de travail collaborative, non seulement d'un point de vue solutions opérationnelles, mais également d'un point de vue processus de mise en œuvre.

2.1.3.3. Prise en compte de l'ensemble des modèles de produit de processus et d'organisation

L'état de l'art sur les modèles de conception nous montre que trois modèles de base (produit, processus et organisation) sont nécessaires pour s'assurer de manière effective et efficace l'avancement d'un projet de conception. En effet la réalité du contexte global de déroulement de la conception montre que ces trois dimensions produit, processus et organisation sont étroitement liées ; ceci signifie que nous ne pouvons traiter une dimension de manière individuelle sans prendre en compte les autres.

La description du produit ou d'un composant ne suffit pas ; le concepteur a besoin de connaître également le processus et les organisations mises en œuvre pour la réaliser.

En effet, le modèle de processus décrit la dynamique des activités de conception ; il permet de décrire le cycle de vie des produits relativement aux activités dans lesquelles ils sont manipulés, et par conséquent, il justifie ou du moins explique l'existence ou l'évolution des produits et des organisations au cours de la conception.

Ainsi, la prise en compte simultanée des trois modèles, permet de décrire les produits et les organisations non pas de façon absolue, mais relativement aux activités dans lesquelles ils sont manipulés. Par conséquent l'intégration de ces modèles apparaît inéluctable pour une description complète du produit et du contexte de son développement.

Un système de conception intégrée est souvent basé sur l'intégration de certaines des deux dimensions évoquées précédemment, à savoir la modélisation intégrée du produit et la multi représentation des vues métiers. Ainsi, la prise en compte conjointe des modèles de produit, processus et organisation a suscité plusieurs études en recherche et au niveau du monde industriel. De nombreux travaux se sont attachés à comprendre et à exploiter les relations :

- entre les modèles ou les applications de la conception à construire et les processus de conception que ces derniers vont faciliter, guider, voire entièrement automatiser,
- entre les processus de conception qui sont mis en œuvre dans une organisation et les objectifs de cette organisation.

Dans des environnements évolutifs, les organisations ont besoin, d'une part d'intégrer les nouvelles solutions applicatives, et d'autre part de guider la mise en œuvre de leurs activités et l'usage des solutions technologiques dans un environnement intégré. La finalité est de construire des structures flexibles qui puissent s'adapter le plus rapidement et le plus aisément possible aux changements organisationnels.

2.2. Le support technologique

2.2.1. Structure globale

La délocalisation des activités de conception, et l'émergence des nouvelles technologies de l'informatique (OMG, CORBA, DCOM, Java RMI, etc.), laissent entrevoir un support efficace à la conception collaborative entre acteurs distribués. De par la diminution des contacts entre acteurs, le problème de conception dans un tel mode se complexifie. La Figure 2.16 montre l'architecture d'un support technologique de la conception collaborative. L'efficacité d'un tel environnement peut être mesurée en termes de complétude, d'indépendance, d'ouverture et d'évolutivité. En résumé, la plateforme d'intégration devrait soutenir les besoins suivants :

- **la plate-forme technologique devra être un système ouvert** afin d'une part d'assurer son évolution pour assister l'intégralité du processus de conception et d'autre part d'en faciliter l'acceptation de la part du groupe des experts issus de divers secteurs disciplinaires participant au projet. La plateforme ouverte est un élément essentiel pour le support d'une démarche de conception collaborative. L'ouverture permettrait de faciliter l'intégration de supports informatiques existants.
- **intégrer les outils spécifiques des disciplines différentes** : Il devrait être possible d'intégrer les différents types d'outils de différentes disciplines. Ceci nécessite le développement d'un modèle central de partage et d'une couche d'intégration dynamique qui définit des interfaces et des configurations de communication.

- **le partage des données et l'intégration des points de vue métier** : une plateforme intégrative doit gérer la duplication de l'information entre les outils, la synchronisation et le maintien de sa cohérence. Dans les outils spécifiques de la conception, certaines informations communes se retrouvent sous différentes représentations. Cette information indique une relation entre les différentes vues de la conception. Bon nombre de mécanismes d'intégration devraient permettre la spécification de ces informations et la représentation des interactions entre différents types d'informations utilisables lors de la définition des points de vue métier
- **gestion d'un modèle commun** : il est nécessaire de disposer de fonctionnalités telles que le stockage des modèles, la manipulation de versions et variantes de modèles, la gestion de demande de modification, la gestion des conflits, etc.
- **l'architecture de la plateforme** : doit assurer l'interopérabilité entre différents outils. L'architecteur peut être orientée vers une approche dirigée par les modèles [Fle06b]. Ce mouvement vers des plateformes fondées sur des Modèles est particulièrement intéressant pour la conception des systèmes d'informations distribués.

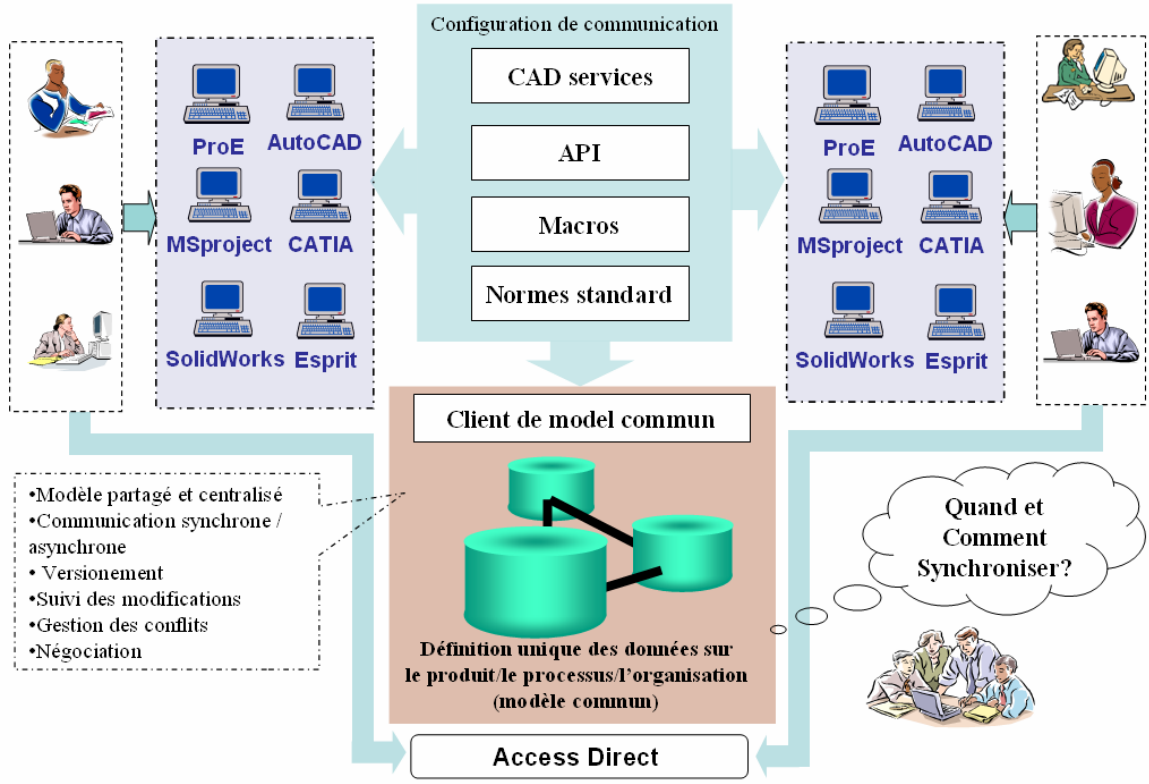


Figure 2.16 Exemple de support technologique de la conception collaborative

2.2.2. Partage et échange des informations

Il existe deux grandes approches pour partager et échanger les informations dans un environnement de conception (Figure 2.17) :

- **échange direct** (création de l'interface entre deux applications métiers) : les informations concernant deux métiers peuvent être échangées en faisant une configuration de communication (protocole d'échange) directe entre deux applications métiers du style

CAD services, APIs, etc. L'inconvénient majeur de cette approche est qu'il est nécessaire de développer des interfaces de communication pour chaque couple d'outils métiers. Dans le cas d'intégration de plusieurs outils, le nombre d'interfaces nécessaires devient ingérable.

- **échange indirect** (intégration des applications métiers), les informations peuvent être stockées en un point unique. Pour éviter les répétitions, le modèle commun permet un partage rapide de l'application, ce qui garantit la cohérence des informations. Nous faisons l'hypothèse que les objets manipulés lors d'un échange indirect sont stockés dans une base commune ou partagée entre les acteurs.

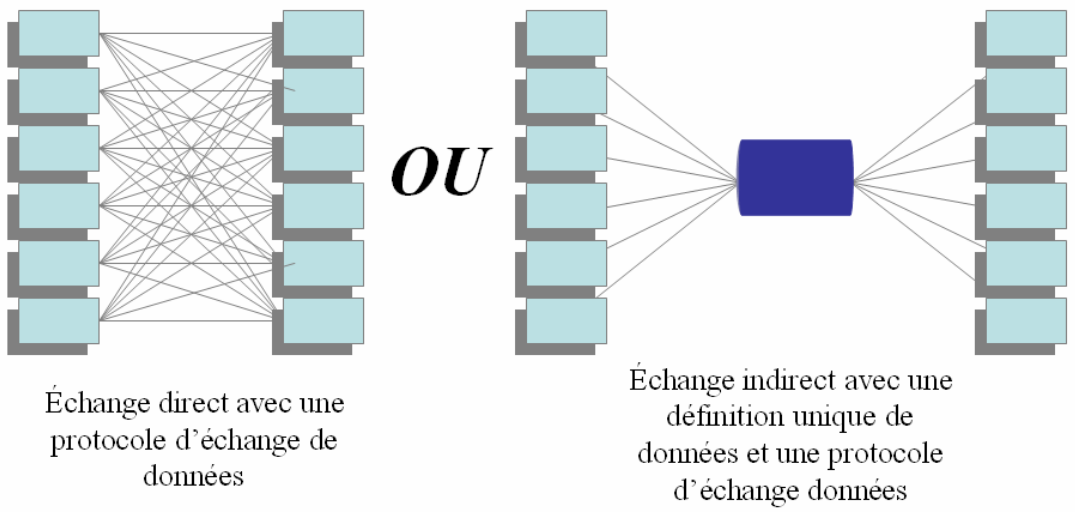


Figure 2.17 Configurations possibles pour l'échange de données

Dans le cas de l'environnement proposé Figure 2.16, nous privilégions des échanges indirects, et donc le partage via un modèle commun. Toutefois, la Figure 2.16 propose deux modes d'accès au modèle commun :

- une **interface d'accès direct** entre acteur et la base commune. Chaque acteur traite les données projet par une modification directe de la base commune. Cette interface permet aux acteurs de créer dans le modèle partagé les différents composants du modèle partagé et les entités relatives à un domaine particulier, d'agir sur le modèle partagé courant (changement d'état des entités, changement des valeurs des attributs, appels de méthodes, etc.), et de consulter l'état du modèle partagé à un instant donné,
- une **interface d'accès indirect** des acteurs à la base commune par l'intermédiaire de sa propre application métier. Il s'agit d'une interface de dialogue entre un outil d'expertise particulière et le modèle commun afin de sauvegarder des nouvelles données et d'interroger les données déjà enregistrées. Les données sont d'abord rapatriées et traitées dans une base privée où sont stockés seulement les objets propres à l'application métier ; ensuite elles sont fusionnées dans la base commune pour intégrer l'ensemble des modifications.

Quelque soit la configuration adoptée, les acteurs ont besoin de coordonner leur accès à la base partagée. La coordination peut être soit synchrone, lorsque les utilisateurs de la base partagée sont amenés à exécuter leurs opérations sur les objets communs d'une façon concurrente en temps réel, ou asynchrone lorsque, les concepteurs travaillent d'une façon indépendante sur les parties du projet dont ils sont responsables. Dans ce dernier cas, ils ont

besoin d'interagir à des intervalles de temps réguliers ou d'une façon spontanée pour fusionner les différents travaux.

2.2.3. Interfaces d'échange d'informations des outils CAO avec l'extérieur

Les interfaces d'échange sont un moyen de dialogue entre les outils CAO avec eux même ou une base de données partagée. Sont reconnues [Bet08] quatre technologies potentielles permettant aux outils experts de communiquer avec l'extérieur, que ce soit avec d'autres outils experts ou avec le modèle partagé : les CAD Services, les APIs Natives, les Macro-commandes et les formats standards d'échanges.

2.2.3.1. Les CAD services

Le standard CAD Services est une interface normalisée conçue pour les systèmes de CAO mécanique qui permet l'interopérabilité des outils de CAO, FAO et IAO. Ce standard permet d'avoir un environnement distribué de l'interface de la conception du produit. Cette spécification permet l'échange de la géométrie et la topologie des données de CAO vers des outils d'analyse et les applications de fabrication. L'objectif est de donner, aux concepteurs, la possibilité d'intégrer les meilleurs logiciels, quel que soit le fournisseur, dans une grande variété d'applications IAO au travers d'interfaces inter processus supportée via CORBA. Ces interfaces standards permettent la conception du produit dans un environnement qui inclut une variété de systèmes CAO, indépendamment de l'outil CAO choisi, et offrent la possibilité de communiquer avec plusieurs outils CAO à la fois.

2.2.3.2. Les APIs Natives

Une API Native (Application Programming Interface) est en quelque sorte une spécification d'interface de programmation qui propose des fonctions ou méthodes particulières destinées aux utilisateurs qui désirent personnaliser les services offerts par un logiciel. La plupart des logiciels CAO offrent une API basée sur une approche orientée objet. En général, l'API se présente sous forme de classes d'objets utilisables dans des langages orientés objet comme le langage C++. C'est le cas de CATIA, SolidWorks, Pro-Enginieer ainsi que de nombreux autres codes de CAO. Afin de communiquer avec le modèle CAO au moyen des API, une interface graphique est indispensable ; elle permet d'afficher toutes les actions à exécuter sur le modèle de la structure en cours de construction.

2.2.3.3. Les Macros

Les Macros sont le moyen le plus simple pour l'utilisateur final pour faire de la programmation sur des logiciels CAO. Ces programmes sont basés sur des langages macros qui, de fait, sont des APIs natives simplifiées. Le mécanisme le plus simple pour construire des macros consiste à activer un enregistreur de macros et d'exécuter l'action souhaitée. L'enregistreur de macros convertit les actions réalisées en code exécutable. Une fois les actions enregistrées, le code peut être modifié pour l'adapter à un contexte plus général.

Les macros sont donc une forme d'API d'ouverture du code. En général moins étendues que les fonctions des API natives, les macros offrent tout de même un accès aux fonctionnalités utilisateurs. Les macros enregistrées sous format textuel peuvent être crées par un outil extérieur, puis chargées dans le code de CAO pour exécution.

2.2.3.4. Les standards d'échanges

Le moyen le plus classique de partager de l'information entre les différents outils participant au cycle de vie d'un produit, est la communication par échanges de fichiers. Cette méthode est pratiquée par la plupart des ingénieurs concepteurs. Les fichiers informatiques se substituent alors progressivement à leurs homologues papiers : plans, gammes, fiches techniques, etc. Les sorties produites par un programme ou module sous forme de fichier peuvent facilement devenir des entrées pour un autre module par l'intermédiaire de ce même fichier. On obtient une chaîne de traitements, chaque traitement utilisant les données du précédant, les échanges de données se faisant par le biais de fichiers intermédiaires. Certains de ces fichiers ont un format standard appelé format neutre (qui possède un mode de description de données indépendant du système de départ et du système d'arrivée). Les entités de départ doivent être écrites sous forme d'entités du modèle neutre. Les formats neutres les plus connus sont STEP, IGES, SET, VDA et des formats basés sur XML.

Un problème des normes standards réside dans la phase de duplication du modèle d'origine. En effet la duplication pose les problèmes de versionement et de mise à jour du fichier standard et des fichiers origines. Le risque de perte d'informations pendant l'échange de données entre le modèle d'origine et le format standard est un second problème récurant. Parfois il n'y a pas de correspondance entre les entités récentes du code CAO et la norme toujours en retard vis a vis des innovations des distributeurs CAO. Les nouveaux concepts ne peuvent être enregistrés dans le modèle standard. Aucune norme ne propose aujourd'hui une méthode d'enregistrement de l'historique de construction des fonctions CAO, même si cette fonction semble vouloir émerger dans STEP : les features demeurent spécifiques à chaque code de CAO.

2.2.4. Version support et suivi de modification

2.2.4.1. Systèmes existants

Les représentations multi-vues du produit dans des environnements collaboratifs doivent permettre aux experts de travailler ensemble à différents niveaux du processus de développement de produit. L'environnement collaboratif exige le maintien de cohérence des différents modèles construits au cours du processus de conception ; cette cohérence est obtenue par trois principaux composants :

- un modèle commun qui intègre tous les autres modèles,
- un système de versionement qui permet d'une part de décrire des alternatives, des choix et des solutions et d'autre part de suivre les évolutions de l'information,
- un système de synchronisation, qui détecte les incohérences et les conflits entre les modèles. Il doit fournir aux acteurs un support pour observer, configurer et résoudre des conflits entre les différentes vues métier.

Comme la conception évolue au fil du temps, de nombreuses révisions, variantes, ou alternatives de la conception, peuvent être créées. La gestion de versions trace ces différences et assure que les différentes versions du modèle de conception sont cohérentes avec les attentes des concepteurs. Alors que la gestion de versions a été utilisée avec succès dans certaines disciplines de l'ingénieur, notamment en génie logiciel (RCS : Revision Control System, CVS : Concurrent Versions System), nous constatons peu de systèmes de conception mécanique qui supporte la gestion de versions pour des modèles de conception. En effet, les modèles de conception mécanique présentent des caractéristiques qui rendent difficile la

gestion de versions et qui doivent être considérées lors du développement d'un système de versionement :

- les applications métier développées à l'origine par des concepteurs spécialistes, ne se préoccupent que de la technologie métier et beaucoup moins des problèmes de collaboration.
- les éléments d'un modèle en conception s'organisent hiérarchiquement par des relations spécifiques dans chaque modèle,
- il y a de multiples représentations pour un objet de conception,

Pour tenir compte de ces caractéristiques, les approches adaptatives de versionement, doivent permettre de décrire les caractéristiques des objets de conception ainsi que les relations entre ces objets. La plupart des bases de données en conception sont basées sur des approches parallèles de copies multiples pour coordonner l'évolution de la conception (Figure 2.18).

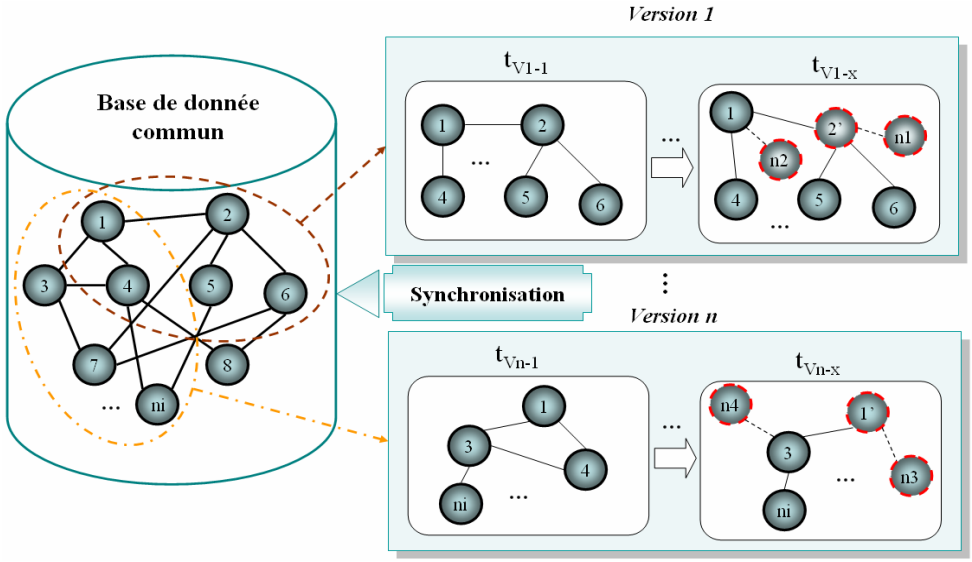


Figure 2.18 Evolutions de la conception par les versions parallèles

Cela se traduit par trois étapes :

- **Copier** : les utilisateurs réservent des copies d'objets qu'ils veulent modifier dans leurs applications locales. Les autres utilisateurs potentiels n'accèdent pas à ces copies tant qu'elles ne sont pas déposées de nouveau dans la base partagée.
- **Modifier** : plusieurs applications acquièrent simultanément des copies du même objet et les modifient indépendamment les unes des autres, créant ainsi des versions parallèles pour le même objet.
- **Fusionner** : pour coordonner le dépôt des nouvelles versions dans la base partagée, on opère au préalable une fusion des versions (réconciliation). Cette réconciliation génère une version spécifique pour chaque objet modifié et cette version est amenée à être déposée dans la base.

2.2.4.2. Conséquence en conception

Ces approches sont fondées sur l'isolement des différentes copies d'un objet de conception puis leur fusion. Chaque version représente l'état de l'objet à un moment donné de

l'historique du projet de conception. Ces approches sont introduites afin de garantir que les concepteurs ne perturbent pas les travaux de leurs collègues en phase synchrone.

Il est alors nécessaire de supporter la gestion et la fusion des différentes copies. Lors d'un projet de conception collaborative, il n'est pas rare que deux concepteurs accèdent simultanément à un même objet de conception (par exemple, un composant du produit), et travaillent par la suite de façon isolée (en mode asynchrone). Chaque concepteur modifie l'objet considéré selon ses propres contraintes et critères. Pour assurer l'émergence d'une solution commune, leurs résultats doivent donc être fusionnés. Avant de les fusionner dans la base partagée, chaque concepteur doit valider ses modifications au regard de celles proposées par les autres concepteurs.

Le plus souvent, les concepteurs travaillent de manière indépendante, modifiant des versions alternatives. Tous les conflits qui apparaissent exigent la négociation et la propagation des décisions de modifications vers les différentes vues des autres concepteurs. Ceci est habituellement réalisé lors de la phase de synchronisation entre les différentes versions. Ainsi, la modification d'une entité d'une vue spécifique peut être cause de conflits avec les entités d'autres vues indiquées dans le modèle partagé. En conséquence, l'identification des impacts issus d'une modification passe par une identification manuelle par les concepteurs. Ce processus manuel n'assure pas un cheminement complet de toutes les modifications du modèle partagé. Par conséquent, une approche efficace pour dépister les modifications de modèle commun est nécessaire pour améliorer de manière significative le processus de synchronisation.

Si des confits apparaissent, les concepteurs doivent négocier la fusion de leurs modifications, donnant ainsi, lieu à une nouvelle version de l'objet. En résumé, le conflit est défini comme un désaccord entre au moins deux experts de la conception dont les points de vue ou objectifs apparaissent contradictoires.

Certains conflits syntaxiques peuvent être automatiquement identifiés par des outils mettant à jour l'information dans une vue quand une autre vue est éditée. Toutefois, la plupart du temps les conflits ne seront pas automatiquement détectés et résolus. Par conséquent de nouveaux mécanismes sont nécessaires pour détecter et évaluer ces conflits afin de notifier les acteurs concernés. Beaucoup de travaux sur la détection et la notification de conflits dans le domaine des bases de données existent dans la littérature ; on distingue les bases de données actives et les bases de données distribuées.

Habituellement, les systèmes de base de données actives utilisent les règles de type "ECA : Événement-Condition-Action". Si la condition de la règle ECA est satisfaite alors des actions associées seront exécutées. La complexité de l'environnement de conception, la grande hétérogénéité des outils de conception et le manque d'interopérabilité entre ces outils limitent l'application de ces concepts aux modèles partagés en conception collaborative.

D'autre part, les systèmes de bases de données actives et distribuées proposées dans la littérature n'utilisent aucun mécanisme de détection de conflits complexes et de notification pour contrôler les dépendances entre le modèle partagé et les contraintes définies par le concepteur. La gestion de tels conflits requière une assistance pour :

- **Identifier et mesurer une évolution conflictuelle d'un environnement collaboratif** : connaître les sources d'incohérence et de conflit ainsi que les étapes d'évolution des versions vers l'incohérence entre elles, dans le but quand synchroniser. Nous avons constaté que si la durée d'évolution des versions excède une limite difficile à cerner, la synchronisation des versions devient ingérable.

- **Analyser les vraies raisons d'un conflit** : 1) identifier les différents types d'incohérences et de conflits ; conflits liés aux violations d'une ou plusieurs règles métiers, aux méthodes, aux valeurs, etc., 2) utiliser les méthodes et outils spécifiques d'analyse de conflits en conception pour définir les différentes causes de conflits et le classement de ces causes.
- **Gérer les comportements d'un conflit** : marquer les différents types de comportements d'un conflit dans le processus de conception collaborative.
- **Assister la résolution les conflits inter experts** : 1) évaluer le positionnement et l'importance des acteurs concernant le conflit dans une réunion de projet de conception, 2) résoudre le conflit par une stratégie adaptée qui peut être un processus automatique ou un processus semi-automatique avec une phase de négociation.

La gestion de conflits pourrait être améliorée si le système de conception intégrée emploie un système temps réel pour détecter les conflits structuraux ou sémantiques entre les différentes vues. Dans tous les cas, il est nécessaire de modéliser les interactions entre les entités du modèle partagé.

2.3. Conclusion

Pour atteindre l'objectif d'une gestion intégrée, complète et performante de toutes les données relatives au développement d'un produit, les modèles intégratifs doivent s'appuyer sur des outils et des modèles de données capables de capturer, de représenter ces données et de les échanger entre les différents acteurs intervenants dans toutes les phases du cycle de vie du produit. De ce fait, la gestion des modèles de produit est un moyen fondamental pour toute activité de collaboration autour du produit.

Nous avons vu que selon le modèle de conception choisi, celui-ci ne se prêtera pas à décrire le même aspect du processus de conception. Ce processus parcourt en effet un espace multidimensionnel, et chaque modèle n'est apte à décrire que certaines de ces dimensions. Il existe beaucoup d'exemples de modèles produit/processus différents, mais peu d'outils logiciels les utilisant. Si on détaille ces outils dans une optique de réutilisation de l'existant, on constate que chaque implémentation a des spécificités qui font qu'elle est dédiée à l'objet ou un contexte de conception pour lequel elle a été créée. En conséquence, elle ne peut donc pas être réutilisée en tant que telle.

L'analyse des tendances d'évolution des environnements collaboratifs de conception, nous a permis de connaître les directions d'améliorations des environnements intégrés :

- prendre en compte conjointement et d'une façon cohérente les modèles de gestion de projet de conception,
- prendre en charge la forte hétérogénéité du contexte global de conception ; ceci concerne l'hétérogénéité des points de vue métiers et donc l'hétérogénéité de leurs modèles de représentation,
- prendre en compte le contexte sociologique de l'entreprise ; sans changement profond des règles et des habitudes des acteurs.

L'intégration, la fusion, la vérification et la cohérence des données créées ou modifiées pendant les différentes phases du cycle de vie du produit sont aussi des fonctionnalités importantes dans un système de conception collaborative. Ces fonctionnalités ne peuvent pas se réaliser sans tenir compte la gestion de la cohérence, de la redondance et la propagation de contraintes et de règles métiers, en parallèle et en interaction avec l'évolution des modèles de

la conception. Cette démarche impose un traitement de l'ensemble des règles métiers des acteurs de la conception pour prendre en compte leurs contraintes métiers, afin de garder la cohérence de la conception. Les règles métiers peuvent être vues comme un savoir accumulé grâce à l'expérience du concepteur. D'après des modèles étudiés, on observe un manque d'identification, de formalisation et d'intégration des règles métiers entre les différents experts par des modèles de conception.

A tout stade du processus de conception, toute solution envisagée par les concepteurs doit donc être compatible avec les connaissances liées aux différents métiers. Toutes ces connaissances ne sont pas forcement exprimées ni intégrées avec les différents modèles de la conception. De plus, les concepteurs ont besoin d'une méthode pour exprimer des règles et des contraintes propres à leurs métiers et pour les partager entre les différents métiers à divers stades du projet.

Il est donc nécessaire de mettre en place une démarche qui facilite l'identification et l'intégration de ces règles métiers. Elles doivent être prises en compte simultanément durant la phase de développement du produit.

3. Scénario d'usage dans un environnement de conception coopérative

L'objectif de ce chapitre est de formaliser les concepts présentés dans les chapitres précédents permettant de supporter une activité de conception coopérative. Au cours de ce chapitre, nous adopterons un point de vue technique qui nous permettra de matérialiser nos propositions dans l'environnement coopératif GAM que nous présentons à la section 3.2. Nous présentons comment y modéliser les divers points de vue des experts. La modélisation repose sur le modèle PPO implémenté dans GAM. Pour illustrer nos propositions, nous présenterons un scénario de conception coopérative en utilisant deux exemples de produits réels. Nous terminerons ce chapitre par la présentation de certains problèmes liés à la cohérence de plusieurs versions simultanées d'un modèle, comme la modification parallèles des mêmes données ou la violation des règles métiers.

3.1. Supports nécessaires pour un environnement coopératif

3.1.1. Technologie pour un environnement coopératif

La collaboration entre experts de la conception est souvent réalisée à l'aide de systèmes PDM (Product Data Management), permettant aux experts de partager des fichiers issus des logiciels (fichiers natifs). Les infrastructures PDM peuvent être étendues à un plus grand nombre d'intervenants afin d'accélérer le développement produit tout en préservant la précision et la sécurité de l'information. Les systèmes PDM s'appuient sur une approche de haut niveau en proposant une structuration des informations relatives à la conception. Cependant, les détails de la conception ne sont pas visibles réduisant ainsi leur effectivité lors de l'analyse de la conception complète d'un produit ou lors de la mise en œuvre des changements progressifs des éléments de la conception. L'atome de description d'un PDM est l'article et il est caractérisé par un ensemble de fichiers dont le contenu n'est pas maîtrisé. Les systèmes PDM ne prennent pas en charge des situations de coopération pendant lesquelles l'expert doit en détail partager des paramètres du produit.

La difficulté de coopération est alors de créer des liens entre les différents modèles souvent hétérogènes (se référant à diverses ontologies, les ontologies se révèlent ne jamais être uniques), à un niveau de granularité fine (à l'intérieur des fichiers). Ces liens se réaliseront, en particulier, dans un environnement coopératif qui permet de créer un modèle commun, homogène et partagé pour assurer l'intégration des différents modèles hétérogènes. Suite à cette intégration, les questions principales qui peuvent se poser sont :

- comment suivre l'évolution du modèle partagé ?
- comment trouver des incompatibilités syntaxiques et techniques lors de l'évolution du modèle partagé ?

Nous pouvons mettre en évidence que l'extraction des incompatibilités syntaxiques entre les modèles est une tâche plus facile que l'extraction des incompatibilités techniques. Ces dernières nécessitent notamment :

- d'intégrer les règles métiers de chaque concepteur dans l'espace commun pour tenir compte, pendant la comparaison des différentes versions du modèle partagé, des connaissances des concepteurs dans la détection de conflits techniques ou pour éviter des incompatibilités techniques,
- de fournir une mesure du conflit en vue d'aider à la prise de décision au moment où une coopération nécessite une revue de projet pour résoudre les conflits. Des experts des différents services travaillent de manière asynchrone et parallèle, avec des outils hétérogènes (spécifiques à chaque service). Lors de la revue de projet, chaque expert arrive avec ses dossiers spécifiques, contenant le résultat des travaux de chaque service. L'objectif étant d'échanger de manière informelle les informations provenant de chaque service. A la fin de cette revue de projet, chaque acteur a en tête les axes de travail sur lesquels il va devoir travailler, et repart avec un document textuel résumant les décisions prises pendant la revue de projet. Suite à cette revue, chaque acteur repasse en mode asynchrone, et travaille sur les axes spécifiques de son service (fixés lors de la revue de projet) (Figure 3.1). Jusqu'à présent, nous ne connaissons pas de système capable de déclencher des revues de projet de manière automatique.

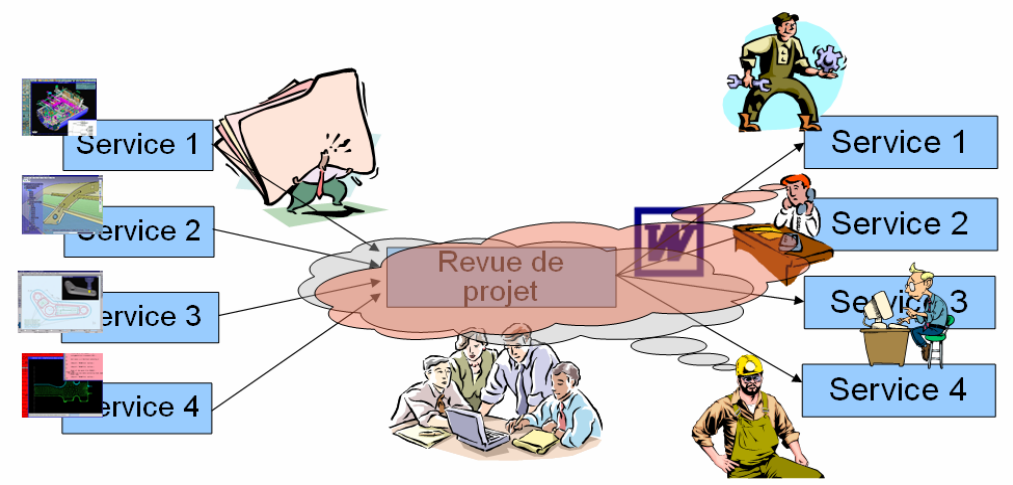


Figure 3.1 Définition des revues de projet et des points de la synchronisation

Pour intégrer ces contextes dans le domaine de la conception coopérative, l'objectif des travaux de cette thèse est de proposer un ensemble de méthodes générales et de démarches à base de cas permettant l'intégration des règles métiers et visant la mesure de conflits et d'incompatibilités entres modèles. Pour illustrer nos propositions nous utilisons un modèle commun homogène qui nous permettra de montrer les problèmes liés à l'évolution d'un modèle. Nous supposons que la phase d'intégration de modèles hétérogènes dans un modèle commun est déjà réalisée, et nous nous intéressons à la manipulation du modèle partagé et à son évolution. Pour traiter ce problème, nous utilisons l'environnement coopératif GAM [Gam08] formalisé et structuré sur la base d'une approche de type IDM (Ingénierie Dirigée par les Modèles) [Hal05]. Nous utiliserons l'environnement GAM pour implémenter le modèle commun et les supports que nous proposerons pour l'environnement coopératif de la conception. En soit GAM, nous permet une implémentation plus rapide des modèles à prendre en charge et au finale une architecteur plus générique que dans des développements usuels.

3.1.2. Approches par modèles (IDM)

L'Ingénierie dirigée par les modèles (IDM) [Hal05] et [Fav06] ou Model Driven Engineering (MDE) [Sch06] propose des pistes qui permettent aux informaticiens de générer tout ou partie d'une application informatique à partir de modèles. Dans le contexte de l'IDM, les modèles sont utilisés comme des éléments de la conception, de la discussion ou de la documentation. Le principe de l'IDM réside dans l'exploitation de modèles comme des entrées du processus de développement d'un logiciel. En pratique, cela nécessite de formaliser les modèles de logiciels à un haut niveau d'abstraction et de traduire ces modèles sur des solutions techniques spécifiques le plus automatiquement possible. Dans la terminologie de l'IDM, ces automates sont regroupés sous le terme de transformations de modèles. Les deux originalités de l'IDM sont donc d'une part des modèles plus formels à haut niveau sémantique, et d'autre part des programmes de transformations de modèles.

3.1.2.1. Intérêt de l'approche basée sur les modèles

L'approche par modèles présente de nombreux avantages. Les modèles constituent un moyen utile pour représenter des activités humaines. Les modèles sont les outils privilégiés des ingénieurs. L'utilisation des modèles favorise la communication entre les acteurs dans le cadre de la représentation et du support d'activités ou de processus coopératifs de la conception et du développement d'applications [Hal05]. La Figure 3.2 montre les trois principaux acteurs indispensables à tout développement logiciel utilisant des modèles.

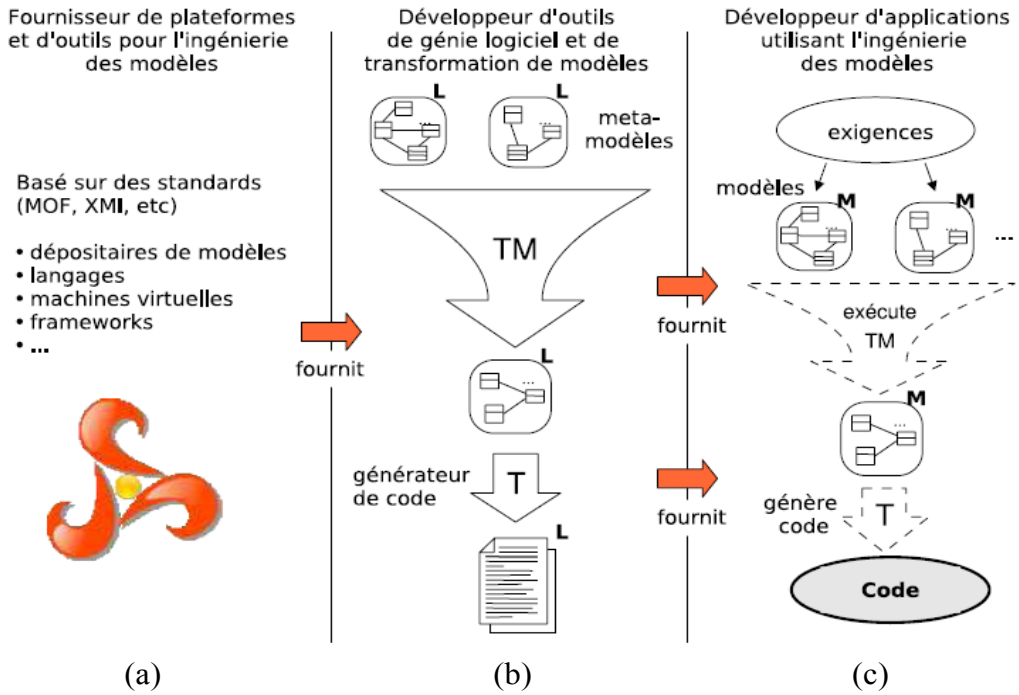


Figure 3.2 Trois principaux acteurs de l'ingénierie des modèles [Fle06b].

La Figure 3.2-c présente un processus d'ingénierie dirigée par les modèles (IDM). Les Figure 3.2-b et Figure 3.2-a détaillent les pré-requis de ce processus de développement : les environnements et outils indispensables à l'exploitation productive des modèles :

– **Ingénierie dirigée par les modèles** [Kab06] : La Figure 3.2-c présente un processus de développement d'applications utilisant l'ingénierie des modèles. L'idée est que le

développeur d'applications commence par modéliser son système à partir des exigences clients avec des modèles adaptés mettant en œuvre des niveaux d'abstraction élevés. Lors de cette phase de modélisation, il utilise plusieurs langages (ou méta-modèles), fournis par le développeur d'outils de génie logiciel, afin de capturer au mieux les différents aspects du système. Pour passer de ces modèles d'analyse à un modèle de conception, le développeur d'applications édite les modèles et utilise des transformations sur étagère qui permettent de composer et de raffiner les modèles de l'application.

- **Outils de génie logiciel** [Sri06] : La Figure 3.2-b présente les développeurs d'outils de génie logiciel et de transformations de modèles. Ces développeurs utilisent les normes et les plateformes génériques d'ingénierie des modèles pour créer des outils de génie logiciel destinés à un domaine d'application particulier. Le rôle de ces développeurs est de définir des langages ou méta-modèles spécifiques, de proposer des éditeurs permettant d'éditer les modèles correspondants et de rendre ces modèles exploitables en fournissant des transformations de modèles.
- **Plateformes de modélisation** [Mar05] : La Figure 3.2-a représente les fournisseurs de plateformes qui sont à la base de l'ingénierie des modèles. Leur rôle est de définir les briques de base des techniques fondées sur les modèles. Ainsi, ils doivent, par exemple, définir précisément ce que l'on entend par modèle, méta-modèle, transformation, et les relations qui existent entre ces éléments. Une fois les éléments de base, leurs sémantiques et leurs relations clairement définies, l'objectif est de proposer des outils et des plateformes qui implémentent ces normes. L'environnement de modélisation GAM présenté dans la suite de ce chapitre est un exemple particulier d'une telle plateforme.

Un environnement pour l'ingénierie des modèles doit permettre de définir des langages de modélisation et des transformations de modèles. Dans la pratique actuelle, des formalismes hétérogènes sont utilisés pour la définition de la structure et de la sémantique des langages et des transformations de modèles. L'hétérogénéité des formalismes utilisés est un obstacle à la fiabilité du processus de développement car le non interopérabilité entre ces formalismes rend difficile toute vérification relative à la cohérence entre les différents artefacts d'un projet [Bur04]. La première contribution de l'environnement coopératif est un langage dédié à l'ingénierie des modèles qui permet d'exprimer de façon homogène les différents artefacts d'un projet, tout en respectant des points de vue différents.

L'ensemble des acteurs représentés dans la section précédente partage l'utilisation des modèles comme base pour le processus de développement. Pour chacune de ces approches de développement, l'utilisation de modèles nécessite la définition de langages de modélisation et de transformations de modèles. En fin de compte, l'ingénierie des modèles est la discipline qui s'intéresse à ces problèmes. Dans la section suivante nous définissons avec précision ce que l'on entend par modèle dans le contexte de l'ingénierie des modèles.

3.1.2.2. Modèle, Méta-modèle, Méta-méta-modèle

Cette section présente les concepts de base de l'ingénierie des modèles (modèle, méta-modèle, méta-méta-modèle) et les relations qui existent entre ces différents concepts [Met08] :

- **Modèle (Le système)** : La première notion importante est la notion de Modèle. Quelle que soit la discipline considérée, un modèle est une abstraction d'un système construite dans un but précis. On dit alors que le modèle représente le système. Un modèle est une abstraction dans la mesure où il contient un ensemble restreint d'informations sur un système.

- **Méta-modèle (le langage de modélisation)** : Afin de rendre un modèle utilisable il est nécessaire de préciser le langage de modélisation dans lequel il est exprimé. On utilise pour cela un méta-modèle. Le méta-modèle représente les concepts du langage de méta-modélisation utilisé et la sémantique qui leur est associée. En d'autres termes, le méta-modèle décrit le lien existant entre un modèle et le système qu'il représente. Un modèle est dit conforme à/ compatible avec un méta-modèle ; le modèle est une instance de méta-modèle.
- **Méta-méta-modèle (le langage de la méta-modélisation)** : De la même manière qu'il est nécessaire d'avoir un méta-modèle pour interpréter un modèle, pour pouvoir interpréter un méta-modèle il faut disposer d'une description du langage dans lequel il est écrit : un méta-modèle pour les méta-modèles. C'est naturellement que l'on désigne ce méta-modèle particulier par le terme de méta-méta-modèle. En pratique, un méta-méta-modèle détermine le paradigme utilisé dans les méta-modèles et les modèles construits.

La base de cette approche est représentée de manière simplifiée sur la Figure 3.3 [Fle06]. Cette figure représente les trois niveaux de modélisation M1 (modèle), M2 (méta-modèle) et M3 (méta-méta-modèle) ainsi que les différentes relations qui existent entre eux. Les trois relations de base de l'ingénierie du modèle sont : la conformité (χ), la représentation (μ) et l'appartenance (ε). Cela permet de mettre en évidence les relations qui existent entre modèles et langages : un modèle est exprimé dans un langage de modélisation (c'est-à-dire qu'il appartient à ce langage) et est conforme au modèle qui représente ce langage. Le méta-méta-modèle représente le langage dans lequel il est exprimé, il est donc conforme à lui même. Cette représentation permet de stopper la déclination aux trois niveaux M1, M2 et M3.

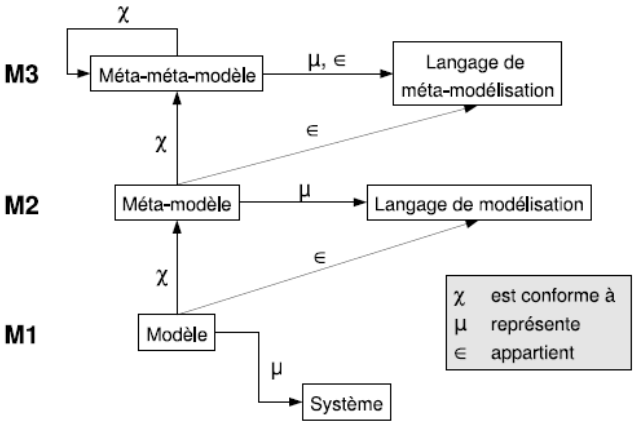


Figure 3.3 Niveaux de modélisation [Fle06].

En pratique, un produit industriel comme un « Ventilateur d'un PC » définit le Système réel, et la façon de représenter et de comprendre ses fonctionnalités et ses composants définit le Modèle du produit. Le modèle est décrit via un langage avec des concepts définis, par un Méta-modèle (PPO, STEP, CPM, …). Notons qu'un méta-modèle n'est pas un modèle d'un modèle. Selon l'IDM, un méta-modèle est un modèle qui définit le langage d'expression d'un modèle. Dans l'exemple précédent, les concepts « fonction » ou « composant » qui accompagnent la représentation du Ventilateur d'un PC constituent le méta-modèle utilisé pour réaliser le modèle de produit. Le modèle de produit doit être conforme à ce langage pour être lisible et utilisable.

Les principaux langages de méta-modélisation existants dans le domaine d'IDM sont MOF, CMOF, EMOF et ECore. MOF (Meta-Objet Facilities) a été normalisé par l'OMG lors de la construction d'UML. EMOF (Essential MOF) et CMOF (Complete MOF) sont deux

évolutions de MOF définies lors de la normalisation de UML. CMOF contient un ensemble de concepts plus important que EMOF qui lui ne contient que les concepts essentiels à la méta-modélisation. Enfin, le langage ECore n'est pas une norme mais le langage de méta-modélisation défini par IBM et utilisé dans l'EMF (Eclipse Modeling Framework).

3.1.3. Lien entre représentation en conception et IDM

Chaque concepteur, possède sa propre vision de l'univers dans lequel il est sensé opérer. Ainsi pour développer son raisonnement face à un problème de conception donné, ce dernier manipule un ensemble de concepts dont la sémantique est spécifique à son domaine d'application.

Un schéma d'expertise permet de regrouper ces concepts et de les organiser d'une manière cohérente, il permet ainsi à travers sa structure et la sémantique de ses éléments de formaliser les connaissances des experts qui demeurent jusqu'alors implicites, comme illustré dans la Figure 3.4. Les schémas sont des représentations organisées de l'expérience de la conception. Ils peuvent entraîner des distorsions cognitives, c'est-à-dire des interprétations erronées des informations extérieures [Bet08]. Un schéma d'expertise fournit les moyens pour créer un ou plusieurs modèles de représentations expertes. La structure d'un schéma se résume en un ensemble de concepts et un ensemble de règles pour associer ces concepts.

Dans tous les cas, un schéma S est défini par un dictionnaire qui regroupe l'ensemble des concepts, et une grammaire caractérisée par un ensemble des règles reliant ces concepts. Une représentation est une concrétisation des pensées du concepteur, et permet de représenter l'activité courante de conception, selon son point de vue. La création d'une représentation, passe par l'instanciation d'un ensemble de concepts parmi ceux du schéma, tout en suivant les règles qui gouvernent le dictionnaire du schéma. Chaque représentation doit être conforme au schéma dont elle découle.

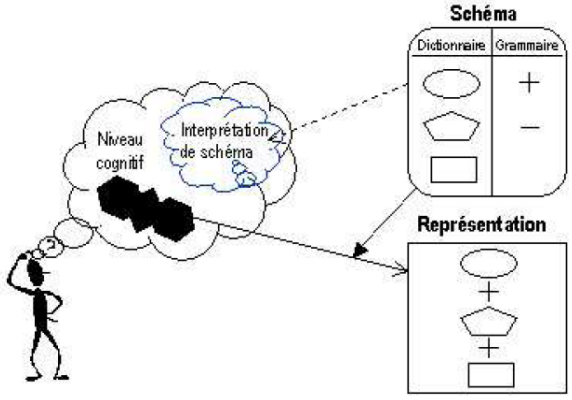


Figure 3.4 Différents nivaux de formalisation [Bet08]

Comme le décrit la Figure 3.5, la formalisation de l'activité de conception se décline à trois niveaux : le niveau cognitif, le niveau schéma, et le niveau représentation. Sur la Figure 3.5, l'étape de modélisation matérialisée par une projection du niveau cognitif vers le niveau représentation, est à la charge de l'individu. En effet, tout expert possède sa propre interprétation du dictionnaire qu'il utilise; en conséquence il n'y a aucune raison pour que deux personnes différentes aboutissent à deux représentations identiques. Ceci explique la multiplicité des représentations utilisées en conception. La représentation experte devient alors un moyen privilégié pour interagir, non seulement avec les acteurs du même domaine,

mais aussi avec les différentes organisations de l'entreprise. La création d'interactions entre deux concepteurs nécessite la présence d'un ou plusieurs objets communs partagés.

Dans la Figure 3.5, l'objet partagé est interprété comme $Op1$, par le concepteur 1, et comme $Op4$, par le concepteur 2 et $Op1 \neq Op4$. Ceci résulte du caractère multi points de vue de l'activité de conception, ou chaque concepteur possède ses propres connaissances et ses propres pratiques concernant les problématiques de conception auxquelles il est confronté. Ainsi, les concepteurs 1 et 2 doivent associer diverses représentations à $Op1$ et $Op4$, conformes aux dictionnaires utilisés.

Supposons que le concepteur 1 crée $R1 \rightarrow S1$ (*R1* conforme à *S1*) et le concepteur 2 crée $R2 \rightarrow S2$.

Les deux schémas $S1$ et $S2$ ne sont pas appariés, l'interaction entre le concepteur 1 et le concepteur 2 n'est donc pas formalisable. L'interaction devient possible si le concepteur 1 et le concepteur 2 utilisent le même schéma Sp, supposé, au moins, partiellement partagé par les deux concepteurs 1 et 2. Malgré tout, la polysémie du schéma Sp impose le plus souvent que : $Rp1 \neq Rp2$.

Mais cette fois-ci, les différences de représentations ne sont pas un barrage à la compréhension par le concepteur 1 de $Rp2$ et réciproquement par le concepteur 2 de $Rp1$.

Pour gérer les interactions potentielles entre acteurs de conception, les connections possibles entre les représentations mises en jeu doivent être modélisées. Dans ce cas, même si $R1$ et $R2$ ne sont pas identiques, nous savons a priori que ces deux représentations font référence aux mêmes objets. Il existe des liens d'appariement entre $R1$ et $R2$, même si ces liens ne sont pas formalisés.

Comme décrit par la Figure 3.5, les interactions doivent être identifiées au niveau cognitif et formalisées au niveau des schémas. L'identification des concepts partagés doit être réalisée, d'abord, lors d'une discussion ouverte entre experts métiers.

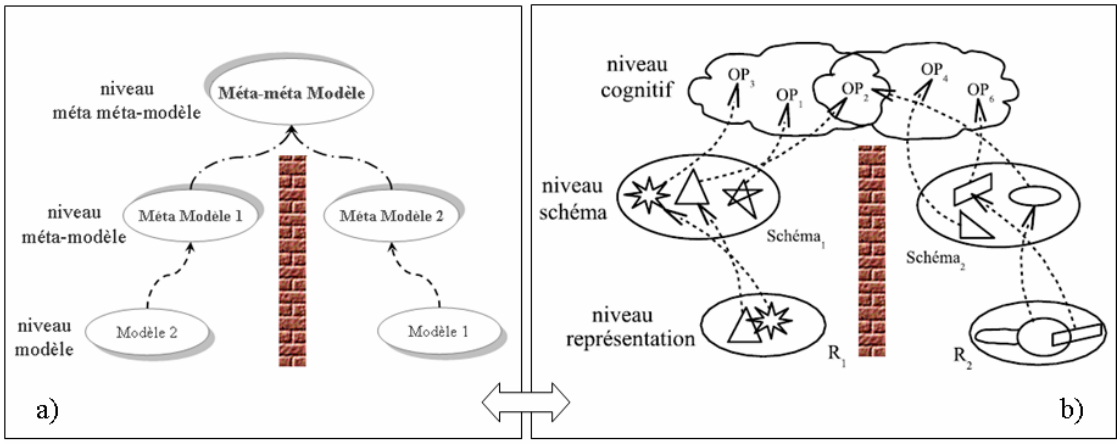


Figure 3.5 Liens entre les différents nivaux de représentation en conception et IDM

A ce niveau encore très général, le domaine de la conception est soumis aux mêmes préoccupations que celui du génie logiciel (Figure 3.5) auquel nous faisons référence ; la question de l'interopérabilité y est centrale. Le niveau cognitif en conception usuellement non formalisé interprète autant les schémas que les représentations. Il s'identifie au niveau méta-modèle du génie logiciel. La définition d'un schéma peut être ramenée simplement à un

ensemble d'éléments connectés par une certaine structure représentée par un méta-modèle. En pratique, une représentation particulière appelée modèle doit être conforme au méta-modèle.

Pour formaliser les interactions, les concepteurs doivent créer ensemble, un méta-modèle commun correspondant à un schéma partagé *Sp* (Figure 3.6), et les règles d'appariement entre le méta-modèle d'une application métier et le méta-modèle commun. L'objectif est donc de définir le méta-modèle commun et partagé, et les règles de traduction qui les relient aux autres méta-modèles. En conception coopérative, l'interopérabilité entre plateformes formalisées par un méta-modèle commun, outils de production, d'analyse et de conception (CAO, DAO, PDM) n'est pas encore mature. La difficulté demeure l'échange d'informations entre les différents outils tout en conservant la cohérence des données qu'elles représentent.

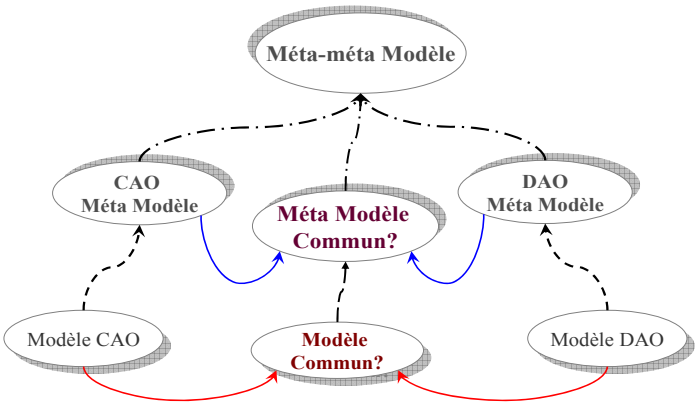


Figure 3.6 Définition d'un méta-modèle commun et partagé

Dans cadre de cette thèse, nous supposons (Figure 3.7) :

- l'existence d'un espace et d'un langage commun qui permettent de partager et de représenter les données des différents métiers,
- l'intégration et la synchronisation de sources de données hétérogènes (les applications métiers) dans cet espace commun sont disponibles ; d'autres travaux [Che07] tentent d'apporter des réponses à cette problématique.
- deux ou plusieurs experts modifient en parallèle l'espace commun, ce qui conduit à différentes versions du modèle commun partagé. Les conflits techniques apparaissent lors de la synchronisation et de la comparaison de ces différentes versions.

On s'interroge alors sur l'assistance à la synchronisation de ces modèles ?

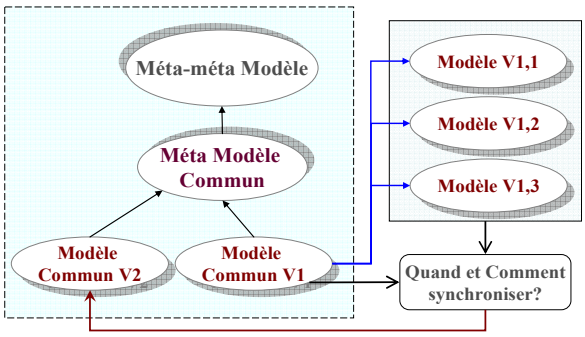


Figure 3.7 Synchronisation des différentes versions du modèle commun

Nous avons introduit précédemment les différents supports qui autorisent l'interaction entre applications métiers. Nous avons choisi le méta-modèle produit PPO comme schéma partagé (méta-modèle commun) qui permet d'établir les interactions entre concepteurs. Ce méta-modèle est restreint à un nombre limité de concepts pour représenter un produit, un processus, ou l'organisation. Cette légèreté le rend facilement compréhensible par toute personne impliquée dans un projet de conception. En contrepartie, il est en premier approche insuffisant pour décrire tous les détails de l'activité de conception. Cependant, chaque expertise métier est susceptible d'étendre l'environnement de collaboration, en enrichissant les concepts qu'elle souhaite partager avec les autres expertises. Cette déclaration ne nécessite pas d'adaptation lourde de l'environnement.

L'environnement GAM, développé au laboratoire G-SCOP est un prototype mettant en œuvre les différents niveaux de formalisation explicités dans cette section. Nous utilisons la plateforme GAM [Gam08] pour ces fonctions de méta-méta-modélisation. Cette plateforme fournit un moyen pour représenter les modèles et méta-modèles des différents métiers dans un environnement unique.

3.2. Plateforme GAM

GAM [Gam08] est une plateforme de modélisation dédiée à la formalisation des activités de la conception coopérative. Elle met en œuvre l'approche de l'Ingénierie Dirigée par les Modèles. Elle propose un méta-méta-modèle permettant de créer de nouveaux méta-modèles et d'instancier simultanément les différents modèles sans compilation informatique. Un mécanisme de versionement intégré permet de retrouver l'évolution simultanée des modèles et des méta-modèles.

La plateforme GAM est disponible en mode client-serveur, ce qui permet de prendre en charge le travail collaboratif quel que soit le nombre d'experts travaillant sur le projet. Cette technique permet aux utilisateurs de travailler de manière simultanée sur un modèle partagé modifiant des versions alternatives. La Figure 3.8 montre l'architecture enrichie de l'environnement qui gère les différents niveaux de modélisation. Dans ce qui suit, nous présenterons les différentes composantes de cette architecture: 1- Graph modeller, 2- GAMM3, 3- GAM-toolkitk, 4- GAM-PPO, 5- GAM-Constraint, 6- GAM-Diff. Le travail effectué dans cette thèse a en particulier permis la spécification et tests préliminaires du module GAM-Constraint.

3.2.1. Modeleur de graphes : « Graph modeler »

Les modèles et les méta-modèles, comme formalismes de représentation des concepts et leurs relations, sont définis par des graphes et une sémantique associée aux nœuds et aux liens de ces graphes. Les nœuds correspondent aux différents types d'informations pouvant être des propositions, des expressions ou des propriétés représentés par experts. L'étiquette associée au lien indique le type de relation entre deux nœuds. Le Graph Modeler de la plateforme GAM permet de représenter et de gérer toute sorte d'information sous la forme d'un graphe. Chaque élément de ce graphe a une identification unique, GUID (General Universal IDentifier) qui identifie chaque élément dès sa création. La gestion de ce graphe permet de définir les différentes fonctionnalités de l'environnement GAM : par exemple la gestion du versionement des fichiers GAM, l'analyse des différences entre deux graphes, etc., sont des fonctions développées à ce niveau.

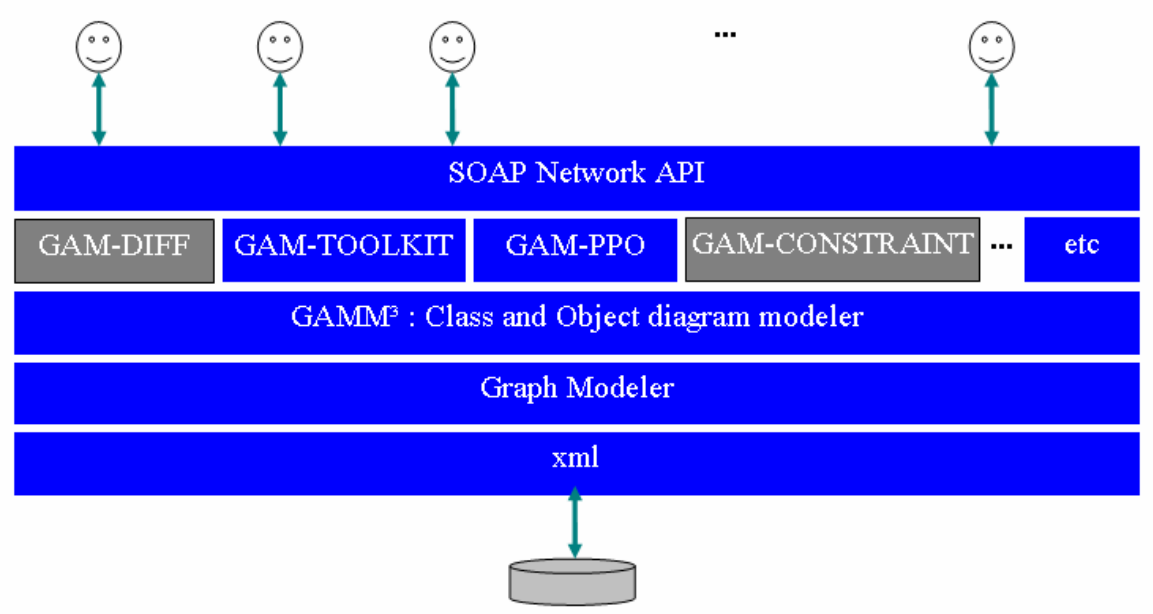


Figure 3.8 Architecture de la plateforme GAM

3.2.2. GAMM3

Le package GAMM3 offre une surcouche fixant une sémantique spécifique pour des graphes de Graph Modeler.

La définition de cette sémantique est ici considérée comme la définition d'un méta-méta-langage. Le package GAMM3 fournit aussi des règles de grammaire du méta-méta-langage. Le méta-méta-modèle issu de GAMM3 est utilisable pour définir des schémas / méta-langages spécifiques. Les packages GAM-Toolkit et GAM-PPO en sont deux exemples.

Dans les parties suivantes nous expliquerons les différents méta-modèles associés à l'environnement GAM que nous envisagerons d'implémenter dans le contexte de la conception coopérative

3.2.3. GAM-Toolkit

Le package GAM-Toolkit fournit un formalisme pour décrire des interfaces graphiques pour l'utilisateur (ou GUI pour Graphical User Interface) associée à une application logicielle. Pour définir une GUI, le package GAM-Toolkit fournit un méta-modèle dans lequel des éléments graphiques (bouton, fenêtre, etc.) et des éléments de contrôle d'une interface peuvent être définis afin de construire un modèle pour l'interface. La composante GAM-Toolkit dispose également d'un moteur d'exécution des modèles d'interfaces.

Dans nos travaux, nous utiliserons un modèle d'interface "GAM-Project Editor" conforme à GAM-Toolkit (Figure 3.9). Le modèle GAM-Project Editor est une interface graphique simple pour décrire, manipuler et sauvegarder des modèles et des méta-modèles de l'environnement GAM. Il permet aussi la visualisation et l'édition des modules GAM-Toolkit et GAM-Project Editor eux-mêmes.

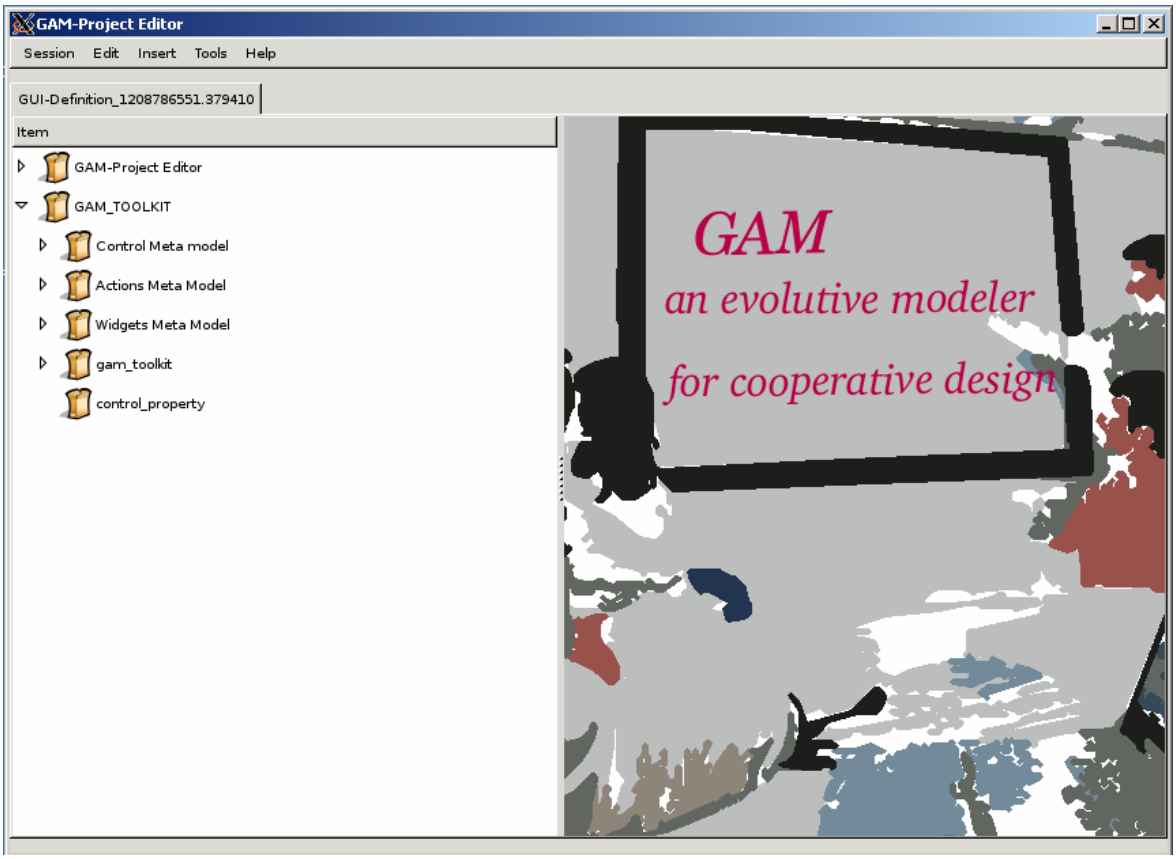


Figure 3.9 GAM-Project Editor : Interface graphique de GAM

3.2.4. GAM-PPO : Modélisation du produit avec GAM

Après l'analyse des modèles de partage d'information de conception (chapitre 2), nous avons porté notre choix sur le modèle PPO (Produit Processus Organisation) qui permet notamment la représentation du produit à l'aide d'entités de base : Composant, Interface, Fonction et Comportement et un ensemble de qualificatifs : Alternative, Commun et Vue. En effet, les entités permettent d'envisager la description d'une famille de produit (Alternative) mais aussi de différencier les données partagées par l'ensemble des acteurs (Commun) de celles spécifiques à un métier (Vue). Le module GAM-PPO est une implémentation du modèle produit comme un modèle PPO dans l'environnement GAM. GAM-PPO peut alors être instancier pour représenter des produits spécifiques. Pour illustrer l'utilisation de GAM-PPO, nous présenterons deux exemples de produits réels. Dans ce rapport, nous illustrerons nos propositions par deux exemples de produits : le cas d'un Relais électromagnétique et celui d'un Ventilateur de PC.

3.3. Modélisation de produits avec GAM-PPO

Les deux exemples qui suivent permettent par la suite d'illustrer les outils développés. Nous présenterons les cas du relais électromagnétique et du ventilateur d'un ordinateur. Pour chaque cas, nous présentons le fonctionnement du système, son processus de conception et enfin la modélisation partagée.

3.3.1. Exemple de relais électromagnétique

Pour illustrer l'utilisation de l'environnement coopératif, nous prenons ici un premier exemple concernant la conception d'un relais électromagnétique.

3.3.1.1. Présentation du système

Le but est de concevoir un relais électromagnétique (Figure 3.10) qui couple de nombreux métiers (mécanique, électrique, magnétique et thermique, fabrication) tout en étant relativement simple. Nous choisissons cet exemple de produit pour sa capacité à mettre en relief la coopération et la communication entre les différents métiers concernés.

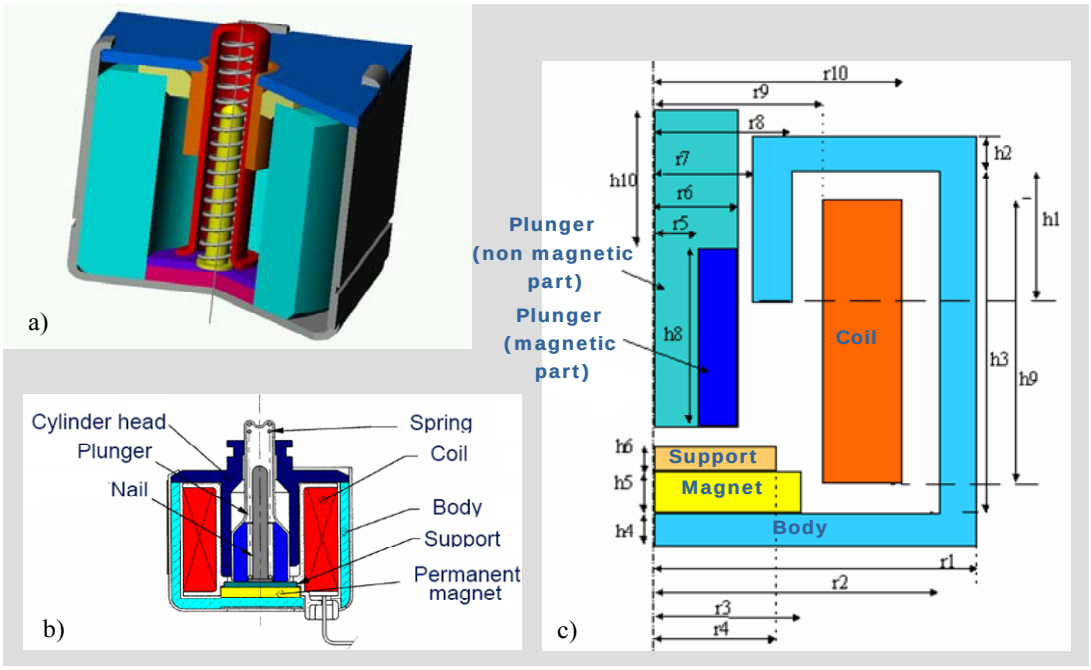


Figure 3.10 Exemple de relais électromagnétique

Nous partons d'une situation où les experts définissent un choix de structure de produit pour concevoir le relais :

- les mécaniciens travaillent autour d'un modèle CAO (Figure 3.10-a). Ils s'intéressent notamment à la fabricabilité des différents composants (Figure 3.10-b), leur assemblage et leur résistance mécanique,
- les électriciens travaillent à partir d'un schéma géométrique simplifié (Figure 3.10-c), pour construire un modèle analytique qui représente les éléments utiles au passage du flux magnétique. Ils utilisent des logiciels propres à leur domaine, comme Flux2D pour analyser le flux magnétique avec une configuration géométrique donnée.

La fonction du système est assurée par le couplage du fonctionnement des systèmes mécanique et électromagnétique. Le percuteur (Plunger) du relais électromagnétique est retenu en position basse par une force magnétique créée par l'aimant (Magnet). Lorsque le courant électrique parcourt la bobine (Coil), il crée un flux magnétique qui va s'opposer à celui de l'aimant. La force retenant le percuteur est ainsi affaiblie. La force prédominante agissant sur le percuteur est alors celle du ressort (Spring) le poussant vers le haut. Le percuteur se décolle de l'aimant et accélère rapidement sous l'action prépondérante du ressort.

3.3.1.2. Processus de conception

Habituellement chaque expert a son propre modèle et la coopération entre les expertises n'est pas formalisée. Ici nous imaginons l'existence d'un modèle partagé accessible aux différents experts.

Après la définition des spécifications et les choix de structure du déclencheur, la phase coopérative de conception peut initier un modèle commun. On s'intéresse alors au dimensionnement en commun de la structure pour trouver une solution technique qui répond aux besoins communs et respecte les contraintes métiers et le cahier des charges fonctionnel.

La Figure 3.11 montre l'utilisation du méta-modèle produit PPO pour représenter le modèle produit électromagnétique. Il est constitué d'une arborescence de description du produit (relais) dans lequel le principal composant CC Assemblage est décomposé en plusieurs sous-composants CC : la culasse (Cylinder-head), le corps (Body), le noyau (Plunger), la ressort (Spring), etc. Chaque composant CC possède une ou plusieurs interfaces VI (View Interface) ; par exemple le noyau a trois interfaces : mécanique, électrique et matériel. On visualise aussi une décomposition fonctionnelle du produit. La fonction principale CF est décomposée en deux sous-fonctions CF : la CF "positionner et fixer" et la CF "mouvement". Par exemple, le concepteur choisit de réaliser l'assemblage de la culasse (cylinder head) sur le corps (Body) en ajoutant la sous-fonction AF culasse-corps (head-body) sur la fonction CF positionner et fixer (positioning and fixturing). Cette sous-fonction correspond à la mise en position et la fixation de la culasse sur le corps. Cette sous-fonction est de type "Alternative" car deux solutions de fixation sont envisageables. Ces deux solutions sont les sous-fonctions CF, le collage (sticking) et le vissage (screwing) de la culasse sur le corps. A partir de ce niveau dans l'arborescence fonctionnelle, deux fixations alternatives peuvent être étudiées séparément.

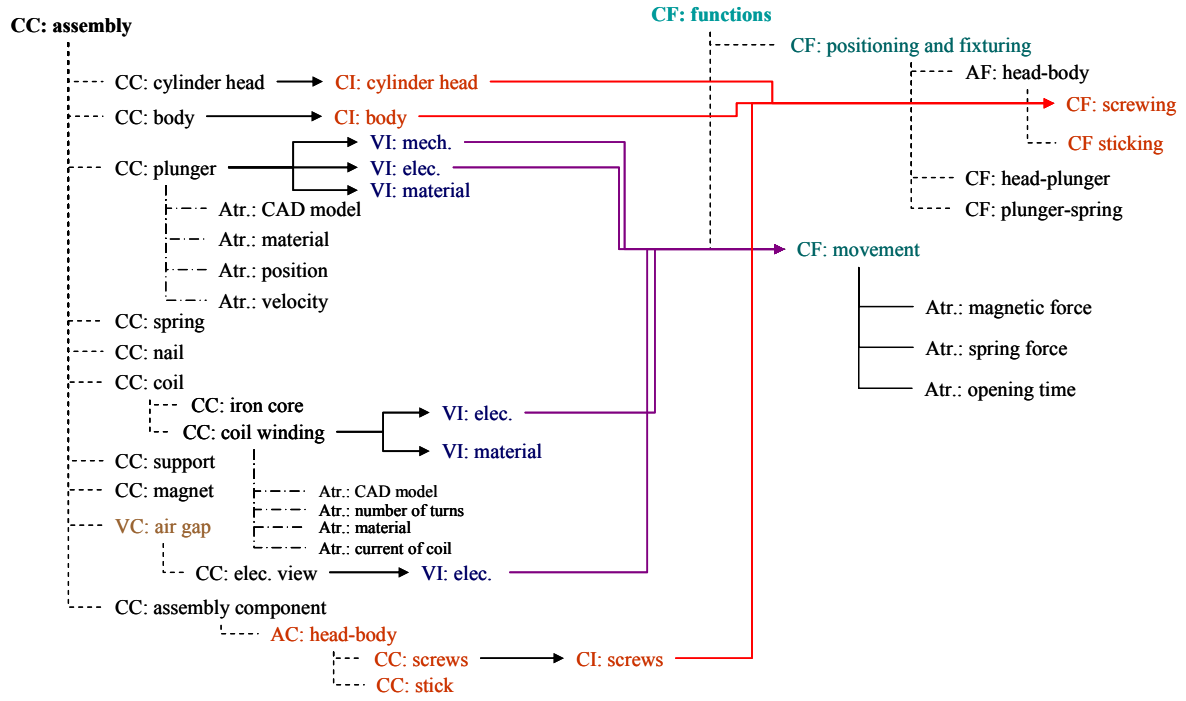


Figure 3.11 Instanciation de modèle produit de PPO pour décrire le modèle du relais

Dans le modèle produit PPO, les composants sont liés à leurs interfaces et les fonctions couplent deux interfaces. Ces différents liens permettent d'affiner la description du produit en permettant de définir quelle interface appartient à quel composant et quelle partie d'un composant est impliquée dans une fonction. Par exemple, les composants vis sont insérés dans la description du produit comme étant une solution de la fonction visser et les interfaces vis se retrouvent dans le modèle comme des relations entre la fonction vissage (Screwing) et les composants vis (Screws) pour réaliser la sous-fonction fixer culasse-corps.

Les différentes vues expertes sont réalisées par l'héritage des attributs des composants, des interfaces ou des fonctions propres aux experts.

3.3.1.3. Modélisation partagée

La plateforme GAM permet aux utilisateurs de créer des méta-modèles et de les instancier ensuite pour obtenir des modèles différents. Nous pouvons alors déclarer le méta-modèle du produit dans l'environnement GAM pour ensuite traiter et représenter les différents modèles produit. Comme illustré dans la Figure 3.12, nous retrouvons trois experts qui travaillent en parallèle sur la conception du relais électromagnétique : le mécanicien, l'électricien et le fabricant. Les trois experts utilisent l'environnement GAM pour construire le modèle du relais conforme au modèle PPO déclaré précédemment. Ils ont leur propre espace de travail (espace métier) pour continuer à travailler en parallèle sur le modèle du relais. Chaque espace de travail correspond à une vision du produit évoluant selon les souhaits de l'expert. Nous obtenons alors, à un instant t, trois différentes versions du modèle du relais. A l'aide d'un processus de synchronisation, chaque expert intègre les différents éléments mécaniques, électriques, magnétiques ou de fabrication définis par ses outils d'expertises spécialisés dans la version PPO de son espace métier.

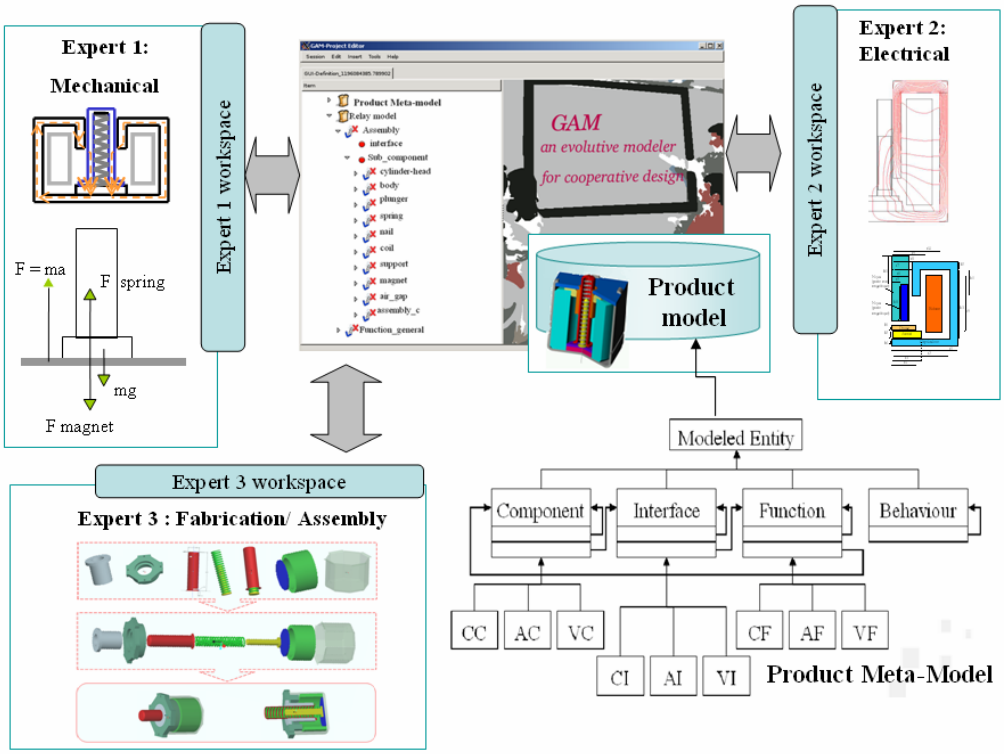


Figure 3.12 Implémentation du méta-modèle produit dans l'environnement GAM

Les Figure 3.13 et Figure 3.14, représentent des détails respectivement du méta-modèle et du modèle du relais dans GAM.

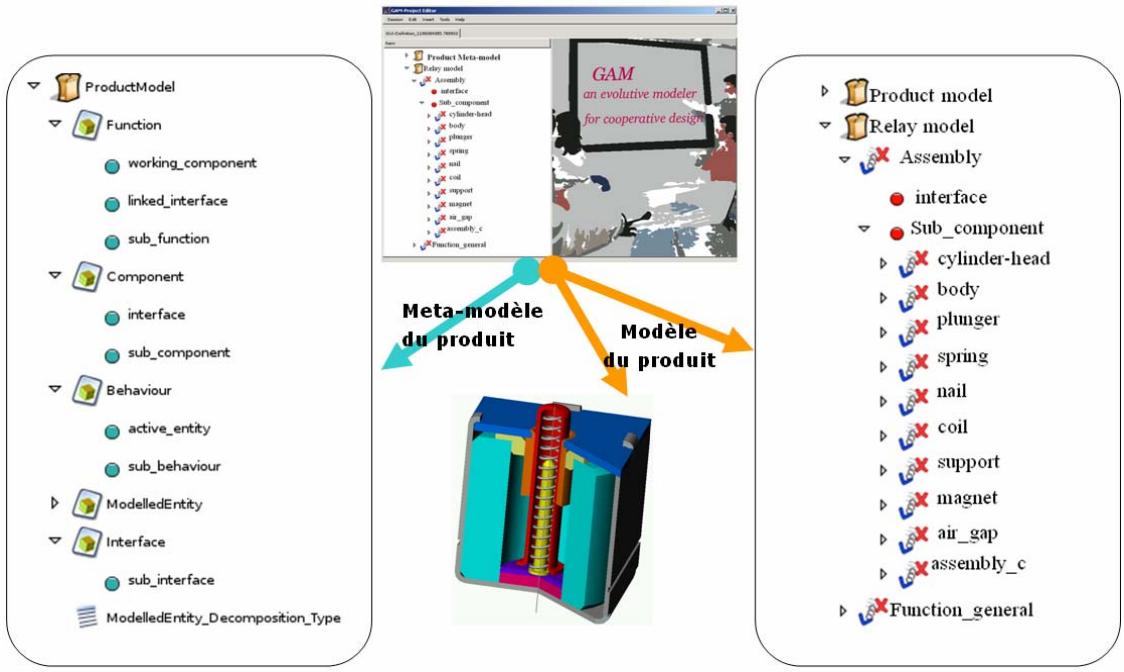


Figure 3.13 Représentation du méta-modèle et du modèle produit du relais dans GAM

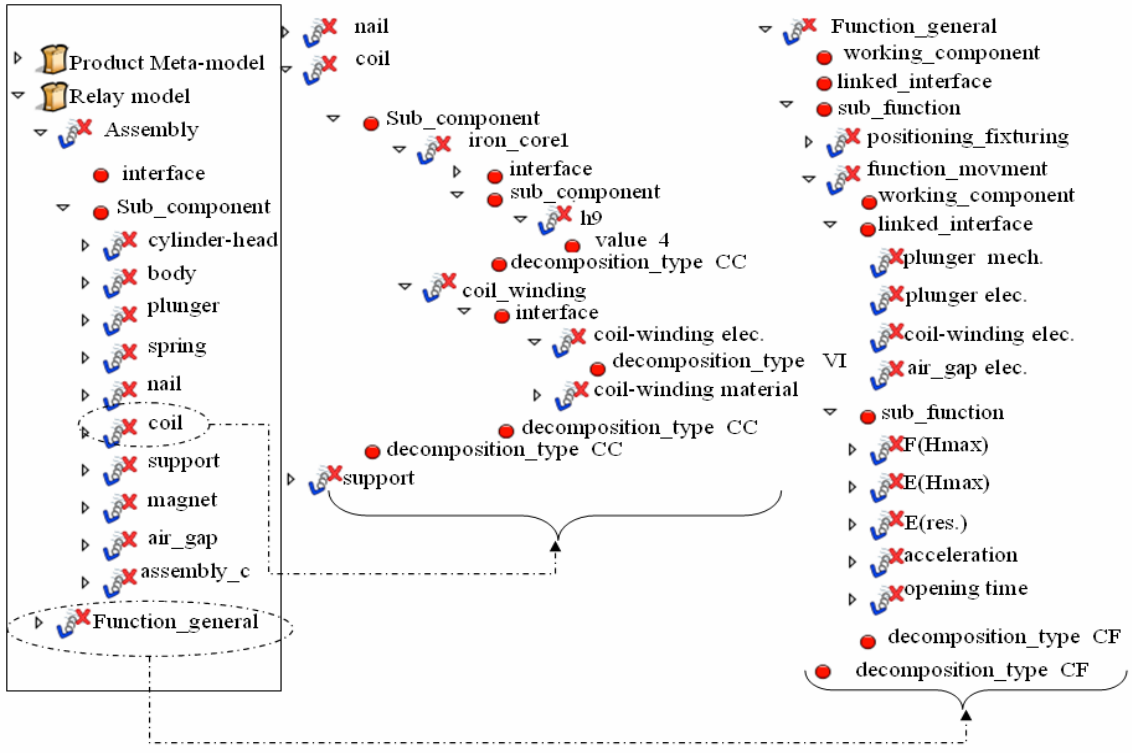


Figure 3.14 Détail du modèle produit du relais dans GAM

Sur la Figure 3.14, le composant bobine est décrit comme constitué d'un fil conducteur isolé et enroulé autour d'un noyau. Le noyau se décompose en deux partis un noyau métallique (iron core) et un fil en cuivre (coil winding) qui sont des composants type Common (CC). La

hauteur du noyau est représentée par la variable h_9, et sa valeur est égale à 4 cm. Le composant fil en cuivre possède deux interfaces, une interface liée aux caractéristiques électriques (coil-winding elec.) et une autre liée aux caractéristiques matériaux (coil-winding material).

La décomposition de la fonction principale apparaît sous la forme de deux sous-fonctions : positionnement (positioning and fixturing) et mouvement (movment). Des composants du relais sont liés à la fonction mouvement avec plusieurs interfaces : plunger elec., plunger mech., coil-winding elec. et air-gap elec. La fonction mouvement est une Fonction type Common (FC) caractérisée par cinq variables définies par des équations mécanique et/ou électrique pour modéliser le mouvement du noyau (plunger) :

– $E(H\ \max)$: l'énergie cinétique en arrivant en position haute,

– $F(H\ \max)$: force exercée en position haute,

– *Opening time* : temps de réaction du relais,

– $E(res)$: énergie de percussion minimum,

– *Acceleration* : une accélération de déclenchement en position bas).

La Figure 3.15 montre les liens que les experts peuvent établir entre leur propre application métier et l'environnement GAM. La formalisation de ce lien ne fait pas l'objet de notre travail et est ici supposée acquise. En utilisant l'environnement coopératif GAM, ils peuvent partager ainsi les éléments communs du modèle produit, par exemple l'épaisseur de la culasse (h_9), et enfin négocier les choix de chaque acteur.

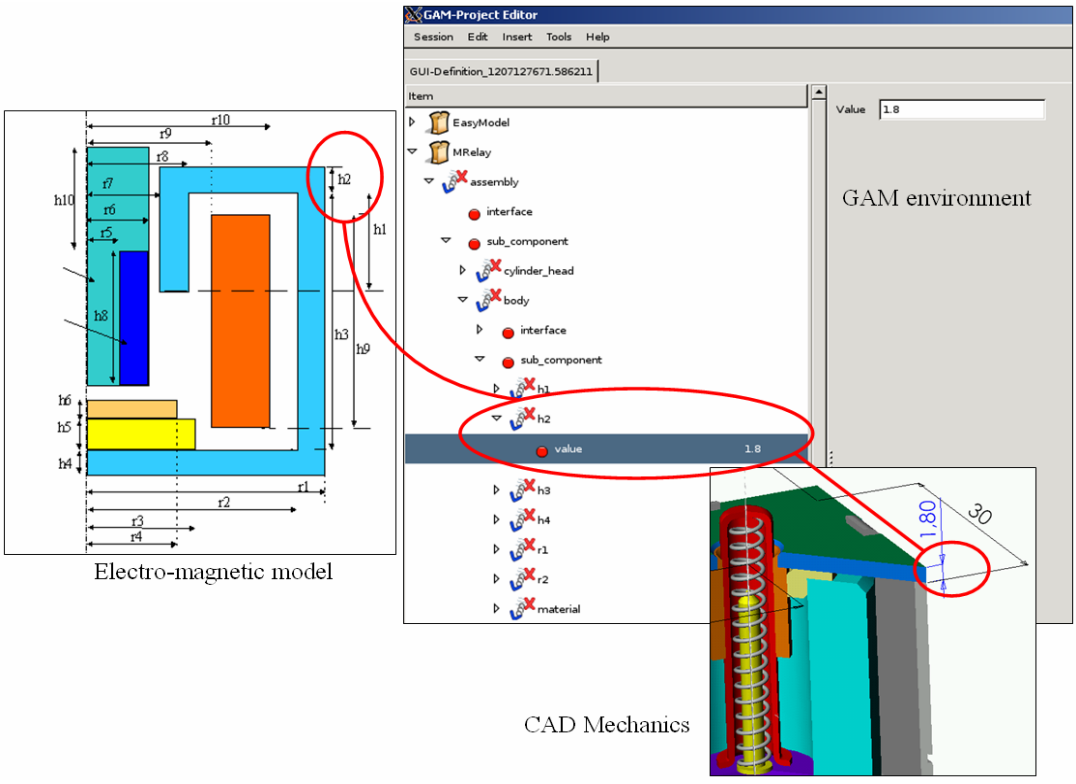


Figure 3.15 Partage des éléments communs du modèle produit

Le principe de cet environnement coopératif est de laisser les acteurs travailler avec leurs propres applications existantes (par exemples un outil CAO mécanique pour les mécaniciens et un outil d'analyse du flux magnétique par éléments finis pour les électriciens), et d'autre part offrir un moyen de partage des données entre ces différentes applications.

Au fur et à mesure du processus de conception du relais électromagnétique, les acteurs peuvent créer et modifier des données relatives à un paramètre du modèle. Dès qu'un acteur apporte une modification au produit il peut ainsi modifier les données associées dans GAM pour que tous les autres acteurs puissent avoir accès à la modification et étudier ensuite ses implications en utilisant leur application spécifique.

3.3.2. Exemple de ventilateur d'ordinateur

Comme deuxième exemple, nous présentons le cas de la conception d'un ventilateur d'ordinateur qui nécessite également l'intervention de plusieurs experts et disciplines : la mécanique, l'électricité, l'électromagnétique, la thermique et la fabrication.

3.3.2.1. Présentation du système

A l'heure actuelle, la majorité des ordinateurs possèdent un dispositif de refroidissement à air associé aux éléments d'ordinateur, comme l'alimentation, le microprocesseur (CPU) et le processeur graphique (GPU) de la carte graphique.

Le CPU est le composant principal de l'ordinateur. C'est à ce niveau que sont traitées toutes les données. La technologie faisant augmenter rapidement la vitesse des nouveaux CPU, il devient donc nécessaire de prêter attention à la dissipation thermique, ce qui amène rapidement à des problèmes de refroidissement du CPU. Le système de refroidissement par ventilation (fan cooling system) est le plus couramment utilisé sur les PC. Comme illustré par la Figure 3.16-a, il se décompose souvent en deux grandes parties :

- un radiateur (Heat-Sink) fixé sur le CPU qui est composé d'un métal à forte conductivité thermique comme le cuivre ou l'aluminium formé par des ailettes. Il offre une surface de contact entre le CPU et l'air. La chaleur émise par le CPU passe par le radiateur et est ensuite dissipée dans l'air,
- un ventilateur (Cooling-Fan) ajouté sur le radiateur qui accélère le flux d'air sur le radiateur, et donc améliore le transfert thermique.

La Figure 3.16-b montre des composants principaux d'un ventilateur de PC. Un ventilateur est un moteur à courant continu qui entraîne une hélice (Propeller). Il crée un vent artificiel par transformation de l'énergie électrique en énergie de rotation de l'hélice.

Le moteur est un moteur sans balais, constitué :

- d'un rotor à l'origine de la circulation d'un flux magnétique créé par des aimants permanents,
- d'un stator bobiné relié à un collecteur rotatif qui est la partie rotative du ventilateur et
- d'un circuit imprimé P.C.B. (Printed Circuit Board) qui est une plaque destinée à regrouper tous les composants électroniques qui contrôlent la vitesse du moteur.

Ce type de moteur ne contient aucun collecteur tournant et donc pas de balais. Par contre un capteur de position (capteur à effet Hall) et un système électronique de commande sont installés sur le P.C.B. pour assurer la commutation du courant dans les enroulements de stator.

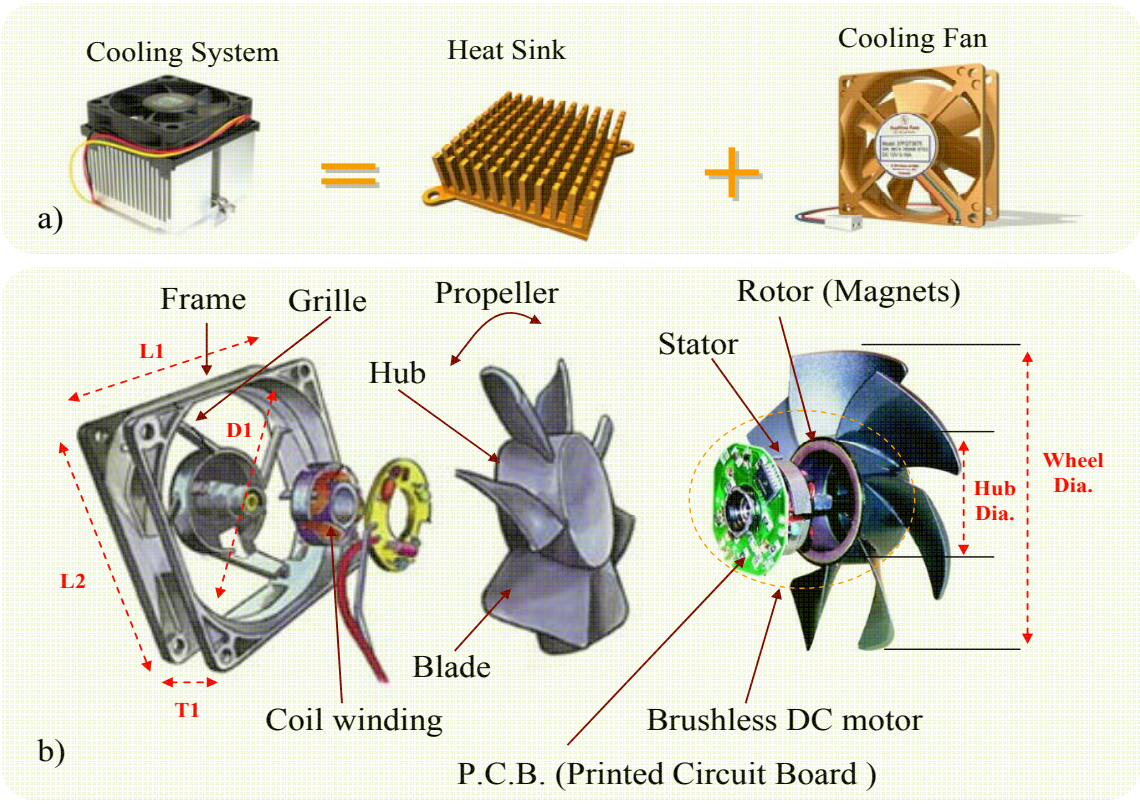


Figure 3.16 Exemple de ventilateur d'ordinateur

3.3.2.2. Processus de conception

La conception du ventilateur s'effectue en plusieurs phases. La première phase traite des modèles analytiques qui illustrent les différents concepts mécaniques et électriques du ventilateur. La deuxième phase d'étude traite de l'amélioration des caractéristiques structurelles du ventilateur par l'utilisation des techniques d'analyse par éléments finis. Ces techniques sont utilisées pour simuler les effets thermiques ou les vibrations des structures aux hautes fréquences et d'analyser le flux magnétique avec une structure géométrique proposée. En conséquence, une analyse des flux et des mouvements des composants mécaniques CFD (Computational Fluid Dynamics) est effectuée. Au terme de cette analyse, on obtient les températures résultantes sous forme de tracés surfaciques sur le composant électrique (CPU) et sous forme de tracés de section à travers le fluide et les solides. Le concepteur peut également analyser des vecteurs de vitesse (vitesse du moteur électrique) qui facilitent la visualisation des schémas de flux à travers lesquels le ventilateur souffle de l'air sur le dissipateur thermique (radiateur).

Les acteurs concepteurs essaient ensemble d'optimiser les différents paramètres de conception pour définir un ventilateur efficace qui soit moins bruyant (diminuer la bruit et les vibrations), consomme moins d'énergie et souffle plus d'air.

3.3.2.3. Modélisation partagée

En utilisant le modèle PPO (Figure 3.17), les experts représentent leurs points de vue et partagent les éléments communs de la conception.

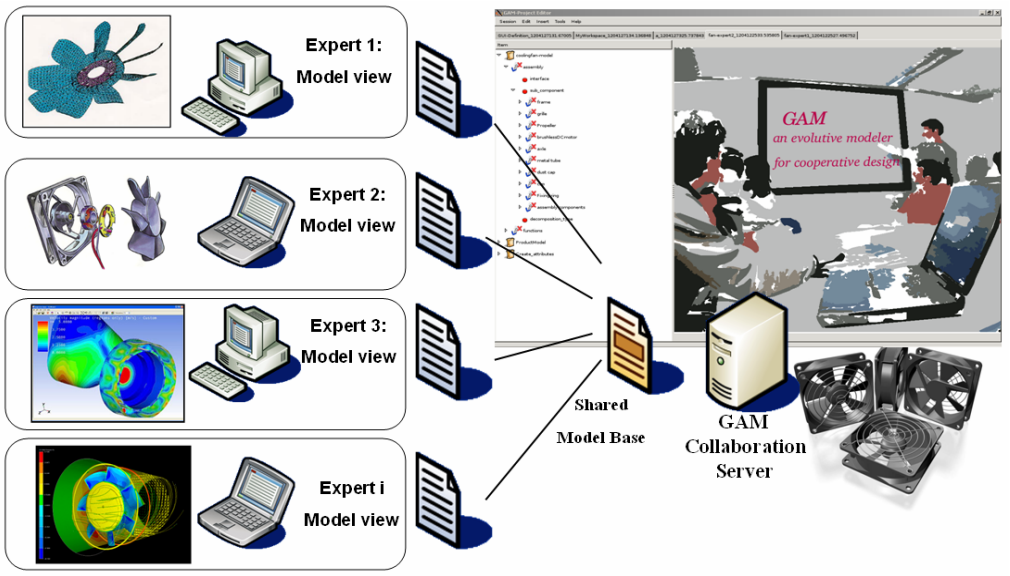


Figure 3.17 Conception d'un ventilateur d'ordinateur dans GAM

La Figure 3.18 montre une représentation hiérarchique du modèle produit du ventilateur. Ce modèle est en relation avec d'autres modèles tels que le modèle géométrique, le modèle matériaux, le modèle de tolérancement, etc. Sur la Figure 3.18, le ventilateur assemblé (assembly) est composé de plusieurs pièces, regroupées en sous-composants (sub-components) : frame, grille, propeller, brushlessDCmotor, axel, etc (Figure 3.18-a). Chaque sous-composant peut aussi se décomposer en d'autres sous-composants, par exemple le sous-composant moteur sans balais (brushlessDCmotor) est constitué de trois sous-composants : un rotor, un stator et un circuit imprimé P.C.B.

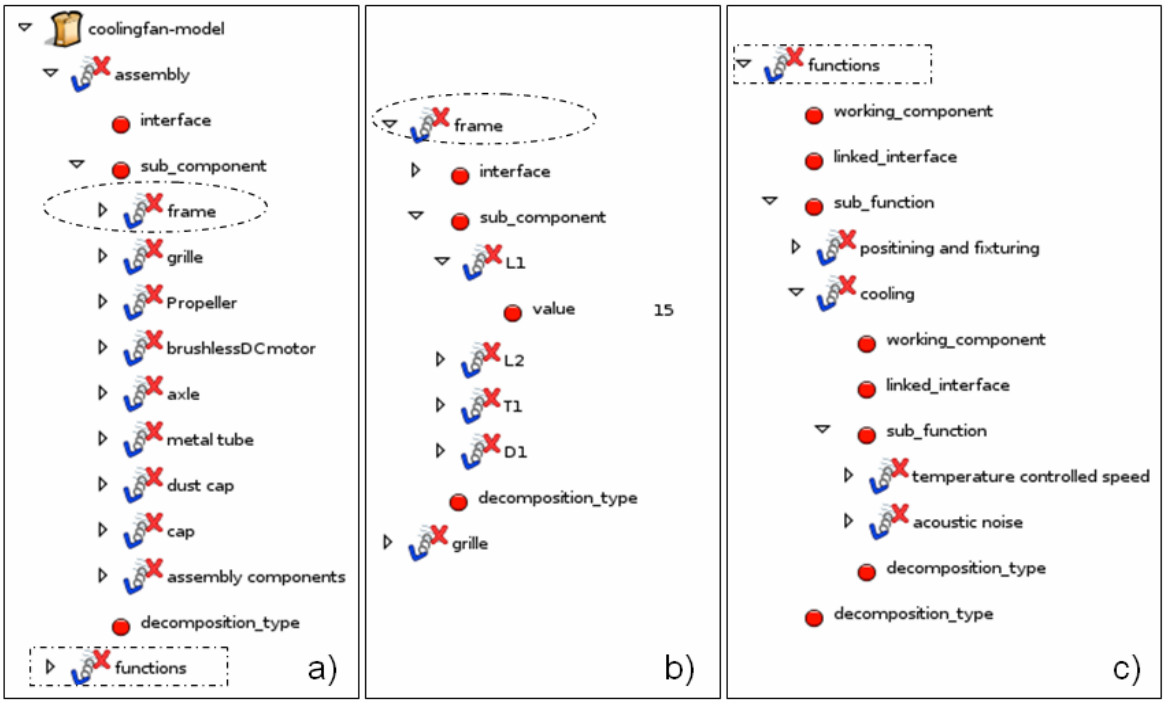


Figure 3.18 Modèle produit PPO du ventilateur

Chaque expert peut réaliser son point de vue en spécialisant les composants , interfaces ou fonctions via de nouveaux attributs. La Figure 3.18-b illustre les attributs géométriques du composant frame sous les noms de variables partagées : L_1 (longueur), L_2 (largeur), T_1 (hauteur), D_1 (diamètre du trou).

Enfin, les experts définissent l'ensemble des fonctions principales qui vont caractériser le ventilateur. Puis, ils décomposent les fonctions en sous-fonctions jusqu'à la description complète du ventilateur (Figure 3.18-c). Le modèle représente l'arborescence des fonctions en montrant la relation père-fils entre fonctions. Cette arborescence est conçue pour permettre ensuite une analyse de vérification de la consistance des relations entre les fonctions. Par exemple le ventilateur a comme fonction père la fonction "Ventiler" (cooling). Pour réaliser cette fonction, il faut réaliser la sous-fonction contrôle de vitesse du ventilateur (temperature controlled speed).

3.4. Problème d'incohérence

3.4.1. Origines des incohérences

Les exemples de conception présentés à la section 3.3 impliquent plusieurs acteurs et conduisent souvent à des ensembles de représentations incohérentes entre elles. Différentes définitions ont été données pour le terme incohérence ou conflit en conception coopérative :

- un désaccord entre les concepteurs sur les composants du produit et/ou sur leur évolution [Con98],
- une incompatibilité entre deux décisions concernant la conception ou les objectifs des concepteurs [Kle00],
- une divergence d'objectifs [Cas00].

Dans tous les cas, une source de conflits apparaît lorsque plusieurs concepteurs proposent des représentations du produit ou d'une de ses parties et que ces propositions sont incompatibles. Il ne faut pas en conclure que plusieurs propositions ne peuvent coexister au même instant, mais plutôt qu'il est nécessaire d'organiser le suivi et la mise en cohérence de ces systèmes.

Dans l'environnement coopératif GAM, les acteurs peuvent travailler de manière indépendante et asynchrone sur des versions différentes (Figure 3.19).

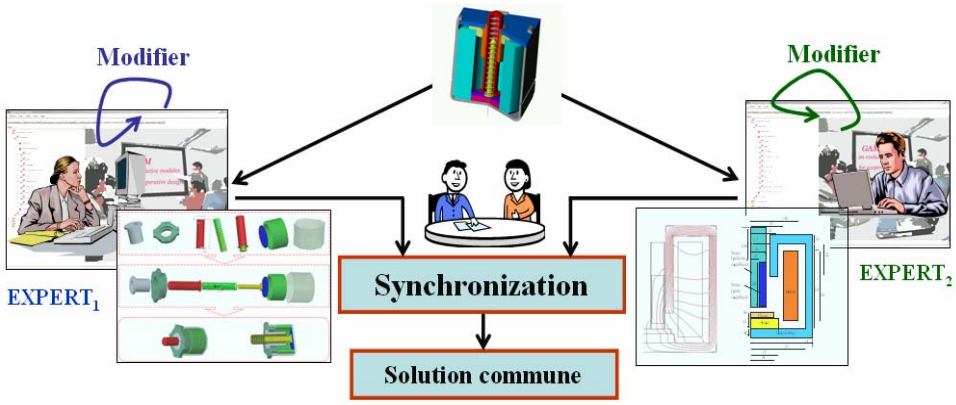


Figure 3.19 Synchronisation de différentes versions

La cohérence des données entre ces versions demeure problématique. Selon les définitions précédentes de conflit, les sources majeures de conflit qui se manifestent pendant la synchronisation de ces versions sont :

- **la modification concurrente de données communes** : les conflits dans ce cas correspondent aux incompatibilités syntaxiques entre deux versions concernant des modifications différentes des données communes. Ce type de conflit peut être détecté par une comparaison de versions différentes,
- **la violation des regèles métiers** : une part importante de la conception coopérative consiste à communiquer et à échanger des données, des informations et des connaissances, entre concepteurs ayant des points de vue et des objectifs différents et spécifiques à leurs domaines d'expertise (mécanique, électricité, fabrication, marketing, …). Lorsque ces points de vue et ces objectifs sont mis en commun, de telles divergences (des points de vue différents) conduisent fréquemment à des conflits. Pour prendre en compte ces différents points de vue et règles métiers qui interviennent en avancement de la conception, l'intégration de l'ensemble des exigences et des règles métiers est une tâche importante de la conception. Cette intégration permet d'éviter des conflits entre leurs exigences.

Prenons deux exemples (Figure 3.20), issus du cas de la conception du relais électromagnétique et qui illustrent l'intérêt de la modélisation des contraintes techniques en conception coopérative. L'épaisseur de la culasse et l'entrefer entre la culasse et le noyau sont deux paramètres géométriques contraints par les deux domaines de la mécanique et de l'électrotechnique.

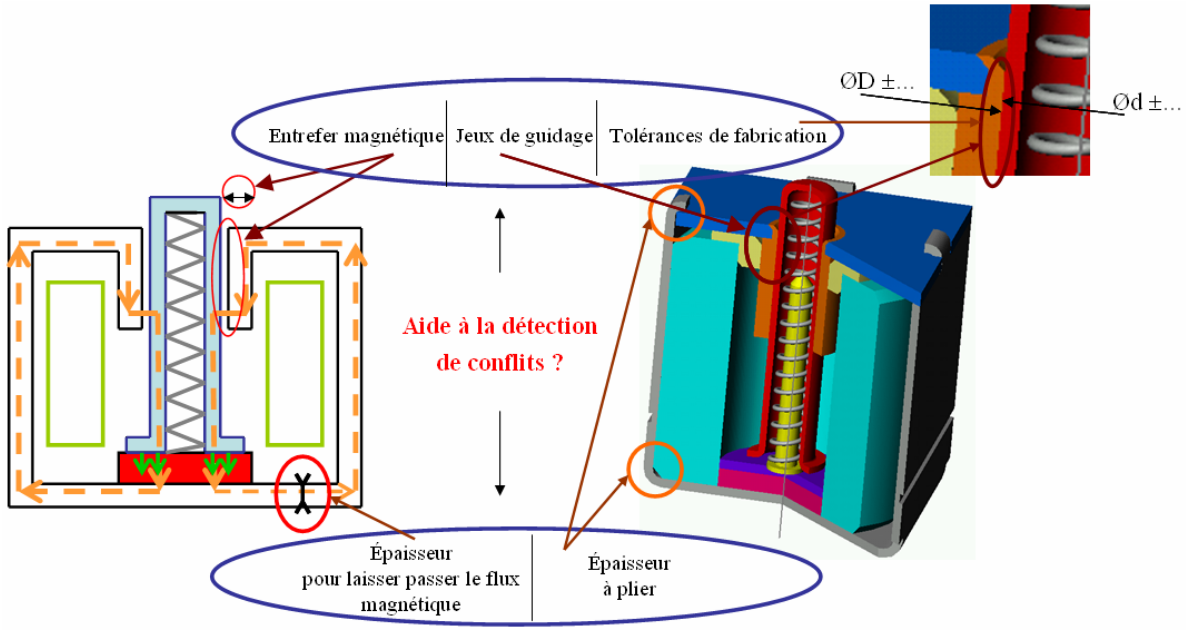


Figure 3.20 Intérêt de modéliser les contraintes techniques

En ce qui concerne l'épaisseur de la culasse :

- **du point de vue mécanique**, il ne faut pas que l'épaisseur de la culasse soit trop importante pour permettre son pliage lors de sa fabrication mais il faut également qu'elle ne soit pas trop petite afin d'assurer une certaine rigidité de la structure,

- **du point de vue électrotechnique**, la culasse constitue une partie du circuit magnétique et son épaisseur influence le flux magnétique.

En ce qui concerne l'entrefer :

- **pour l'électrotechnicien**, cet entrefer représente une grande résistance dans le circuit magnétique et il doit être suffisamment petit pour ne pas empêcher le flux magnétique,
- **pour le mécanicien**, si l'entrefer est trop petit, le frottement entre le noyau et la culasse gêne le mouvement de la partie centrale ; il doit être suffisamment grand pour permettre au clou de glisser mécaniquement en prenant en compte aussi les tolérances de fabrication.

De telles contraintes, considérées comme des informations importantes de la conception, doivent être modélisées et échangées en parallèle avec le modèle produit et au fur et à mesure de l'avancement du projet de conception. Il faut que chaque acteur de la conception se demande quelles sont les influences et les contraintes imposées par son propre métier. Ainsi, en prenant en compte les différentes contraintes de chaque métier avec le modèle produit, les acteurs peuvent aboutir à un compromis afin soit d'éviter tous les conflits ou pour négocier et formuler des intervalles de variations tolérables par tous les métiers.

Les experts doivent disposer d'une configuration des événements pour vérifier quand et comment les conflits sont détectés, surveillés, stockés, présentés, et éventuellement automatiquement résolus. Ceci passe par des méthodes vérifiant la violation des contraintes et définissant l'importance relative des conflits. Tracer les évolutions de chaque modèle est aussi un élément important pour simplifier l'identification des conflits et leurs causes, cela simplifie l'accès à l'information associée et la prise de décision. La prise de décision peut être assistée par un support de négociation entre experts affichant les causes de conflits et formalisant les conséquences des décisions.

Pour résoudre le problème de cohérence entre les acteurs métiers dans l'environnement coopératif GAM, nous proposons deux composantes complémentaires : GAM-Constraint et GAM-Diff. Ces deux composantes sont des supports à la cohérence de l'ensemble des experts, facilitant ainsi la représentation et l'échange des règles métiers et le processus de synchronisation des versions avec le modèle partagé.

3.4.2. GAM-Diff

Le module GAM-Diff que nous utilisons, fournit un méta-modèle pour représenter la différence entre les modèles. Un algorithme générique compare deux modèles dans l'environnement GAM et retourne un modèle de différences conforme au méta-modèle de différences. Le mécanisme de comparaison permet d'identifier les modifications (ajout, suppression, modification d'éléments) du modèle de départ afin de représenter et traiter les différences entre le modèle de départ et le modèle évolué (la version).

Le module GAM-Diff est donc un outil d'aide pour la synchronisation des versions parallèles d'un même modèle. Les experts utilisent ensuite ces modèles de différences pour choisir les modifications qu'ils accepteront ou qu'ils refuseront lors de la mise à jour du modèle partagé. Le module GAM-Diff assure surtout la vérification de la cohérence syntaxique. Ce module est présentée et développée en détails dans les chapitres suivants.

3.4.3. GAM-Constraint

Le modèle partagé PPO est statique, le suivi de son évolution est laissé à la propre initiative des concepteurs. Ce mode est loin d'être suffisamment productif. En effet, les acteurs ont besoin d'un environnement dynamique pour gérer l'évolution du modèle partagé et pour maintenir la cohérence de la conception.

La caractéristique de cet environnement dynamique est le fait qu'il intervient comme un acteur de la conception. Le module GAM-Constraint est envisagé pour définir la dynamique de modèle.

Les contraintes sont issues des relations qui apparaissent entre les éléments du modèle produit pour maintenir la cohérence du modèle. Les contraintes définissent en quelques sortes les règles d'évolution du modèle partagé. Nous nous basons sur la déclaration et la vérification du réseau de contraintes définies sur un modèle produit. Les contraintes réduisent l'espace d'évolution des représentations du produit, mais permettent de maintenir la cohérence dans l'environnement de conception et imposent des règles d'évolution. Ceci est présenté et développé en détails dans les chapitres suivants.

3.5. Conclusion

Dans ce chapitre, nous avons présenté un environnement coopératif (GAM) pour modéliser les divers points de vue des experts de la conception. Nous utilisons une implémentation du méta-modèle PPO, permettant de décrire les différents aspects du produit à concevoir à différents nivaux d'abstraction (composant, interface, fonction, comportement).

Dans cet environnement, chaque expert importe au moins une partie du modèle partagé. Ils éditent de manière asynchrone leur copie du modèle partagé sans que les autres acteurs soient informés des modifications en cours chez leurs collaborateurs. Une fois ces tâches asynchrones finalisées une étape de fusion doit être assistée selon les modifications proposées dans les différentes versions. Cette étape illustre certains problèmes de cohérence de versions lors de la modification concurrente de données communes ou lors de la violation des règles métiers. Pour aider les experts à maintenir la cohérence globale de modèle, la suite du document consiste à développer d'une part, un modèle de contraintes (module GAM-CONSTRAINT) support à la cohérence entre l'ensemble des experts en facilitant la représentation et l'échange des règles métiers, et d'autre part, une méthode basée sur (GAM-DIFF) qui fournit un moyen d'obtenir la différence syntaxique entre modèles conformé au même méta-modèle afin de réaliser la synchronisation et de conserver la traçabilité des modifications sur des versions construites en parallèle.

4. Modélisation des contraintes métiers pour la cohérence des modèles

Nous venons de présenter un environnement de partage d'informations dans un cadre collaboratif. Dans ce chapitre, nous présentons l'utilisation des contraintes métiers pour la gestion de la cohérence dans le modèle partagé. A chaque étape de la conception, tout expert participant au travail collaboratif doit impérativement travailler sur un modèle cohérent. Nous partons ainsi du principe que tout modèle partagé est cohérent et qu'une modification, si elle est exécutée toute seule dans un modèle partagé cohérent, laissera toujours ce modèle partagé dans un autre état cohérent. Cette fonctionnalité est généralement assurée par la vérification des contraintes qui sont des assertions logiques des experts de la conception devant toujours être vérifiées par les données du modèle partagé.

Nous présentons notre approche pour la représentation et la vérification des contraintes métiers dans le cadre d'un modèle partagé qui supporte les activités de conception collaborative. Dans un premier temps, nous rappelons l'intérêt de la modélisation des contraintes en conception et définissons ensuite un cadre d'utilisation des contraintes pour assurer la cohérence d'un modèle partagé. La suite est consacrée à la représentation et à la vérification des contraintes dans l'environnement GAM qui supporte et abrite le modèle partagé. Nous montrons enfin la mise en œuvre de l'approche proposée ainsi qu'un exemple d'application.

4.1. Intérêt des contraintes en conception

L'activité de conception de produit apporte une réponse aux besoins exprimés sous forme d'un cahier des charges qui vise à définir les spécifications de base d'un produit. Sachant que nous sommes dans un contexte de conception collaborative, tous les métiers intervenant dans le développement du produit doivent exprimer leurs besoins et contraintes, il devient de plus en plus difficile de satisfaire totalement tous les objectifs du cahier des charges. Lors de la conception d'un produit, il y a des choix à faire sur la structure, les fonctions, les dimensions, les matériaux, … pour répondre aux différents besoins exprimés par tous les intervenants. Un choix de conception dépend d'un autre choix, d'une loi physique, d'une règle métier, des besoins client, d'un usage, d'une technologie de fabrication, de la connaissance et l'expérience des acteurs métiers. Par conséquent, les choix sont délicats à faire, les paramètres sont difficiles à évaluer, les décisions difficiles à prendre, car ils sont fortement interdépendants et encore plus interdépendants dans le cas d'une collaboration. Les différents éléments de la conception sont souvent communs à plusieurs métiers et à plusieurs acteurs, donc peuvent être traités en même temps. Les objectifs des acteurs étant parfois antagonistes, les choix et les décisions divergent souvent. Dans ce cas, les différentes interactions entre choix, ainsi que les acteurs qui font ses choix doivent être orientées vers la solution la plus adaptée à la difficulté rencontrée lors du processus de conception.

Afin de réduire les incohérences en conception, les décisions que les acteurs ont à prendre doivent être conformes aux connaissances explicitement exprimées et qui permettent la

compréhension des choix et des contraints imposées par chaque acteur. Plusieurs questions sont alors soulevées :

- comment considérer simultanément l'ensemble des connaissances et raisonnements de tous les experts lors du processus de conception ?
- comment aboutir à des solutions cohérentes de conception en considérant l'ensemble des règles formelles et informelles ?
- comment formaliser des règles d'expertises ?

Il faut donc pouvoir évaluer plusieurs choix et prendre en compte toutes les contraintes des intervenants de la conception. Ces contraintes peuvent être :

- les exigences essentielles du cahier des charges pour assurer l'intégrité des solutions techniques,
- les connaissances ou les raisonnements des experts pour définir par exemple les limitations de ressources, les limites de validité du modèle d'analyse, les spécifications concernant l'analyse des phénomènes physiques, etc.,
- les flux d'informations entre d'experts afin de maintenir la coopération interdisciplinaire.

Il faut donc pouvoir :

- exprimer d'une manière la plus complète possible les contraintes que les acteurs doivent respecter,
- représenter tous les conflits techniques qui peuvent se produire en analysant les contraintes exprimées par les acteurs,
- rechercher les choix possibles qui prennent en compte toutes les contraintes plus ou moins importantes à satisfaire.

Les cas d'utilisation des contraintes en conception sont :

- la représentation du problème de conception sous la forme d'un problème de satisfaction de contraintes (CSP),
- Les techniques d'optimisation en conception sous contraintes,
- Le contrôle de cohérence de la conception par les contraintes d'intégrité.

4.1.1. Représentation du problème de conception sous la forme d'un CSP

Un problème de satisfaction de contraintes, CSP pour Constraint Satisfaction Problem, est un problème modélisé sous la forme de contraintes. Une contrainte est « une relation entre un ensemble de variables, chacune prenant ses valeurs dans un domaine continu ou discret. Une contrainte est donc définie sur un ensemble de variables» [Fis00]. La résolution d'un problème modélisé sous la forme de contraintes consiste à trouver une ou plusieurs solutions de telle sorte à ce qu'on affecte une et une seule valeur à chaque variable tout en respectant toutes les contraintes simultanément.

En général un CSP est un problème définit par le triplet (V, D, C) tel que [Gel98] :

- Un ensemble fini de variables $V = \{V_1, V_2, ..., V_n\}$ prenant leurs valeurs dans des domaines $D = \{D_1, D_2, ..., D_n\}$ continus ou discrets tel que chaque variable $V_i \in V$ dispose d'un domaine associé de valeurs possibles $D_i \in D$,
- Un ensemble fini de relations C, appelées contraintes, ou chaque contrainte $C_j \in C$ est défini sur un sous-ensemble de variables.

Il existe plusieurs techniques pour résoudre un problème CSP [Com99]. Les techniques les plus utilisées sont le filtrage. Le filtrage consiste à déduire à partir des contraintes les valeurs impossibles des variables. C'est-à-dire, à chaque itération les contraintes sont utilisées pour filtrer les domaines en éliminant les valeurs qui ne seront jamais solutions relativement aux choix effectués.

Deux classes de CSP sont envisagées selon la séquence de recherche des solutions :

- Le CSP statique [Fle06] suppose que le nombre de variables et de contraintes ne varient pas durant la séquence de CSP,
- Le CSP dynamique [Mou04] est simplement une séquence de CSP où chaque élément de la séquence diffère du précédent par l'ajout et/ou le retrait de certaines variables et/ou contraintes.

Dans la littérature, plusieurs travaux concernant l'utilisation des CSP dans le domaine de la conception ont été développés : les systèmes CAOs [Mar05], les modèles de produits [Hor99] et de processus de conception [Has95], la conception simultanée [Gup96], la gestion de workflow [Su03], ont fait l'objet de modèles applicatifs.

Les contraintes en conception définissent un ensemble de règles élémentaires issues de la modélisation des connaissances et des raisonnements des experts concernant la conception d'un produit [Fis00]. Elles sont spécifiées sous forme de relations continues ou discrètes entre des variables définissant les paramètres et les choix de conception. Le modèle de processus de conception étant une logique d'évolution du modèle produit, il est considéré comme un ensemble d'alternatives de tâches et de méthodes de conception modélisées sous forme de contraintes. Nous pouvons considérer, en général, une tâche de conception comme un processus augmente ou diminue le nombre de variables. Ce processus permet d'affiner la définition du produit par l'instanciation de variables ou par génération de nouvelles contraintes pour explorer des voies de conception particulières. Il s'agit là, selon le vocabulaire de la programmation par contraintes, de choix qui s'accompagnent de la création de nouvelles variables, d'instanciation de variables et de pose de contraintes dynamiques [Yan98].

L'hétérogénéité des variables et des contraintes combinée avec la dynamique des contraintes représentées par les concepteurs durant les différentes phases augmentent la complexité du réseau de contraintes. Dans ce contexte, l'implémentation des différents algorithmes de CSP devient difficile puisque l'importance du temps de calcul n'est pas compatible avec l'activité de conception et que plusieurs acteurs travaillent sur les mêmes éléments donc sur les mêmes modèles de contraintes. Il est difficile d'envisager plusieurs concepteurs exécutant ensemble et en même temps la propagation de contraintes sur le même modèle. Nous notons également l'importance du contrôle du modèle de contraintes directement par les concepteurs : nous n'avons pas seulement besoin de propager les contraintes en aveugle mais aussi du suivi de l'état de contraintes. Ceci permet à un acteur, avant d'exécuter la propagation, de valider ou de remettre en cause ses choix par rapport aux contraintes des autres acteurs.

4.1.2. Technique d'optimisation en conception sous contraintes

Les techniques d'optimisation utilisent les outils de la programmation par contraintes afin de résoudre des problèmes de conception. Ces techniques ont été utilisées dans plusieurs domaines d'application de la conception : la conception automobile [Foi99], aéronautique [Ben01], électronique [Cou99], etc. Dans un contexte de conception collaborative, l'optimisation multidisciplinaire doit permettre aux concepteurs d'incorporer les effets des contraintes imposées par chaque métier en même temps. L'optimum global ainsi trouvé est meilleur que la configuration trouvée en optimisant chaque discipline indépendamment les unes des autres : on prend en compte les interactions entre disciplines [Che02]. La conception est effectuée par équipes, relevant d'un domaine précis. Chaque équipe s'attache à obtenir un point de conception acceptable [Gup93]. L'optimisation suppose une formulation des différents aspects du problème de conception suivant trois éléments [Che02] :

- **Contraintes** : une contrainte est une condition qui doit être satisfaite pour obtenir un point admissible. Au delà de la traduction des phénomènes physiques, ces contraintes représentent des limitations de ressources, des exigences des cahiers des charges, ou des limites de validité du modèle d'analyse.
- **Objectifs** : un objectif est une fonction qui doit être optimisée. Par exemple, le concepteur choisit de maximiser le profit, le rayon d'action, ou de minimiser le poids total. D'autres problèmes traitent de multi-objectif et de multi-acteurs, souvent antagonistes. La solution consensuelle sera alors choisie par tous les acteurs.
- **Modèles** : les concepteurs établissent des modèles afin de représenter le comportement des contraintes et des objectifs selon les variables de conception. On peut avoir des modèles expérimentaux, comme une analyse régressive sur le coût du produit, ou bien des modèles analytiques, comme les éléments finis. Le caractère multi-disciplinaire du problème complique fortement le choix des modèles et leur implémentation.

Une fois que l'on a choisi les variables de conception, les contraintes et les objectifs, le problème général peut s'exprimer de la manière suivante [Cho96] :

trouver x qui minimise $f(x)$

sous contraintes $g(x) \leq 0$, $x_{LB} \leq x \leq x_{UB}$

Où f est une fonction objectif, x est le vecteur représentant les variables de conception, g est le vecteur définissant les contraintes de conception, et x_{LB} et x_{UB} sont les bornes, respectivement inférieur et supérieur, des variables de conception. En optimisation sous contraintes, l'objectif est à la fois de satisfaire les contraintes du problème et d'obtenir en même temps la meilleure solution de conception selon un critère ou un objectif.

Le problème est ensuite résolu en appliquant les techniques d'optimisation appropriées. Cela passe par des algorithmes à base de gradient, des algorithmes génétiques, etc. La plupart de ces méthodes demandent un grand nombre d'évaluations des objectifs et des contraintes. Chaque méthode est assez coûteuse en temps de calcul. Il faut préciser enfin qu'il n'existe pas, pour ce type de problème fortement combinatoire, de solution miracle garantissant de trouver l'optimum global en un temps raisonnable.

4.1.3. Contrôle de cohérence de la conception par contraintes d'intégrité

Dans beaucoup d'entreprises, le processus de conception coopérative d'un produit fait appel à des connaissances, des ressources et des équipements géographiquement distribués [Wan02]. La coopération et l'intégration en conception distribuée sont réalisées par l'utilisation de bases de données partagées qui forment une représentation multi-dimensionnelle de l'ensemble des paramètres impliqués dans le processus de conception. Un des problèmes majeurs dans ces bases de données partagées est le maintien de la compatibilité et de la cohérence des différentes vues des experts [Car99]. Par conséquence l'identification des impacts issus d'une modification passe par une identification manuelle par les experts. Ce processus manuel n'assure pas une propagation complète de toutes les modifications de la base partagée. Par conséquent, une approche efficace pour dépister les modifications de modèle partagé est nécessaire pour éviter des conflits entre experts en conception distribuée [Cer03].

Dans le domaine de l'informatique, la cohérence d'une base de données partagée est assurée par la définition de contraintes d'intégrité [Dou99], qui sont des assertions définies sur la base de données qui doivent être vérifiées à la fin de chaque modification. Un état d'une base de données partagée est cohérent si toutes les contraintes d'intégrité sont satisfaites [Cho05]. De nombreux cadres comme les contraintes d'intégrité, ont été développés pour maintenir la cohérence de la base de données dans le domaine de l'Ingénierie.

Ram [Ram97] propose un modèle orienté objet pour la conception collaborative. Ce modèle est basé sur la définition des objets de la conception et des méta-objets associés à ces objets. Les méta-objets représentent les contraintes de la conception. Chaque objet de la conception est associé à un espace de contraintes qui est composé d'une ou plusieurs contraintes. Tout état interne de l'objet de conception est contrôlé par ses contraintes. Ce modèle fournit un environnement flexible qui sépare l'environnement de définition des contraintes de l'environnement de la conception. Ceci permet la définition et la modification des contraintes sans modification de l'application de conception. L'approche proposée exige la programmation manuelle en C++ pour capturer les objets et les contraintes de la conception.

Goonetillake [Goo02] décrit un cadre de modélisation basé sur le concept de Contrainte Version Objet (CVO) pour fournir un mécanisme d'intégrité entre les différentes versions d'un même objet. Chaque CVO contient un ensemble de contraintes d'intégrité qui doivent être satisfaites pour une version d'un objet particulier. Les composants de l'objet et les CVOs sont distribués dans l'ensemble du système. Les CVOs sont classés en local pour les composants des objets situés dans un espace de travail local et en global pour des configurations globales des objets complexes situés dans l'espace de travail global. Cette classification facilite la vérification de la cohérence des données et facilite aussi la gestion des contraintes.

Roller [Rol02] propose le concept d'Active Semantic Network (ASN). L'ASN est une base de données active, distribuée, et orientée objet qui prend en charge la définition des contraintes de la conception en les évaluant par un système à base de règles. Les contraintes sont utilisées pour modéliser n'importe quel type de dépendance entre les données du produit. Le modèle de mesure de conflit est basé sur la vérification des contraintes de la conception et des règles de type ECA (Event, Condition, Action), pouvant être employées pour la résolution des problèmes de la cohérence en conception collaborative. Quand un événement se produit sur une entité dans certaines conditions, une action est automatiquement exécutée propageant de nouveaux événements. Dans certains cas, des règles générales peuvent être employées pour

répondre automatiquement aux événements. Puisque certaines actions ne sont pas automatisées, l'expert peut être impliqué dans la propagation de contraintes par des procédures de notification.

Dans les travaux de Wang [Wan04], un « linkage universelle modèle » est développé pour représenter l'information liée à la conception de produits mécaniques dans une forme distribuée. Il intègre les contraintes géométriques traditionnelles de la géométrie et des contraintes non géométriques de la conception. Il fournit une représentation des contraintes pour la conception détaillée et l'optimisation de la conception. La représentation des contraintes de type intervalle et les méthodes de résolution sont étudiées.

Toutefois, ces recherches sont uniquement fondées sur la représentation des contraintes numériques et ne concernent pas d'autres types de contraintes. Nous ne connaissons pas de mécanismes de contrôle d'un réseau de contraintes pour suivre les modifications asynchrones des versions d'un modèle partagé de la conception.

4.2. Spécifications pour la formalisation de contraintes en conception collaborative

Dans le cadre de cette thèse, nous prenons en compte les contraintes métiers comme des contraintes d'intégrité dans un environnement coopératif de conception qui permettent de définir les conflits techniques lors de l'évolution d'un modèle partagé. En effet, dans une activité de conception coopérative, il faut déterminer où chaque contrainte métier sera stockée et vérifiée. Cette décision dépend du type de contrainte manipulée, du mode de mise à jour du modèle partagé et du moment où la mise à jour est exécutée. Dans ce cadre, nous proposons d'adapter cette approche à l'origine du domaine de l'informatique au domaine de l'ingénierie, et évidemment pour les systèmes coopératifs de la conception.

La question principale qu'il faut se poser est : comment peut-on gérer les contraintes d'intégrité dans un modèle partagé de conception pour assurer la propriété de cohérence entre différentes experts ?

Nous nous sommes intéressés à la recherche de solutions et à la vérification de contraintes en présence de modifications parallèles des versions d'un modèle partagé, visant:

- à vérifier les contraintes le plus tôt possible dans le processus de modifications parallèles du modèle partagé, de manière à minimiser les retours sur des modifications conflictuelles en cas de violation de la cohérence,
- à ne pas imposer à l'expert de spécifier la stratégie de vérification des contraintes, mais que celle-ci soit déterminée automatiquement par le système,
- à spécifier les contraintes métiers comme des contraintes d'intégrité de façon globale et déclarative. Une analyse syntaxique des contraintes et des modifications permet de réduire l'ensemble des contraintes à vérifier, en déterminant un ensemble de contraintes risquant d'être violées par une modification.

La mesure de la violation de chaque contrainte peut servir à quantifier l'incohérence globale du modèle partagé. Si nous considérons le modèle global comme des relations entre les entités de conception et comme un réseau de contraintes associées à ces entités, les conflits s'expriment par l'identification des contraintes violées au cours de la conception.

Nous donnons dans les parties qui suivent la représentation et l'intégration de notion de règles métiers dans les différents modèles de la conception coopérative. L'objectif est de définir la

structure d'un modèle de règles métiers qui se veut générique, mais qui continuera à évoluer au gré des développements en recherche et des applications industrielles. Nous présenterons d'abord la notion de contrainte et de règle métier dans la conception pour décrire ensuite leurs cas d'utilisation dans la conception coopérative.

Pour pouvoir présenter ces contraints dans un environnement coopératif de conception, nous présentons un méta-modèle de contraintes qui permet de décrire le modèle de contraintes appliquées aux modèles de la conception. Nous avons implémentés ce modèle de contraintes dans l'environnement GAM. A travers les deux exemples du relais et du ventilateur, nous faisons le lien entre le modèle de contraintes et le modèle produit de PPO.

4.3. Structure du modèle de contraintes

Dans notre travail, nous nous basons sur la déclaration et la vérification du réseau de contraintes définies sur un modèle partagé. Les contraintes réduisent l'espace d'évolution des représentations, mais permettent de maintenir la cohérence dans l'environnement de conception et imposent des règles d'évolution. Les contraintes dans un processus de conception collaborative sont ici classées selon l'origine des entités de chaque modèle partagé (Figure 11-a) :

- contraintes produit/produit qui représentent des rapports entre les entités définies par le modèle de produit : par exemple, des contraintes au sujet de dimension, de la géométrie, de la tolérance, etc.,
- contraintes processus/processus qui représentent des rapports entre les entités définies par le modèle de processus : par exemple, les contraintes de charges des ressources,
- contraintes organisation/organisation qui représentent des rapports entre des entités définies par le modèle d'organisation : par exemple, contraintes de prise de décision dans des unités d'organisation,
- contraintes produit/processus qui représentent des relations entre les entités du modèle de produit et du modèle de processus: par exemple, la contrainte qui spécifie que des pièces sont impliquées lors d'une tâche spécifique : disponibilité de modèles pour la tâche,
- contraintes processus/organisation qui représentent des relations entre les entités du modèle de processus et du modèle d'organisation: par exemple, des contraintes au sujet du choix d'un responsable de tâche en fonction des rôles de chacun,
- contraintes organisation/produit qui lient la définition du produit à l'organisation de l'entreprise. En particulier chez des assembleurs des produits complexes, la nature des composants du produit dépend du choix des fournisseurs et donc de l'organisation du système de production dans son ensemble.

Une telle classification aide à choisir le système de gestion de contraintes pour maintenir la définition, l'évaluation et la résolution des contraintes. Nous pouvons utiliser le lien Processus-Organisation pour l'identification des acteurs de la collaboration. La propagation via le lien Produit-Processus définit l'impact des solutions choisies sur les activités.

Dans ce document, nous nous concentrons toutefois sur les contraintes produit/produit déclarées sur le modèle produit de PPO.

Les contraintes sont caractérisées selon leurs positions dans l'espace 3D présenté par la Figure 4.1 [Bet06]. Dans cette figure :

– l'axe « contexte » réfère l'étendue des groupes de personnes concernées par la définition des contraintes. Parce que notre architecture du modèle partagé sépare l'espace de travail en espaces privés et un espace commun, des contraintes peuvent être divisée en trois catégories : 1) les contraintes locales ne concernent que les entités de l'espace privé, 2) les contraintes globales impliquent des entités dans différentes vues, 3) les contraintes universelles concernent toutes les vues et concepteurs. Le contexte d'une contrainte permet une première évaluation de son importance.

– l'axe « formalisation » identifie le degré de formalisation. C'est un indicateur des méthodes qui sont employées dans la définition, l'évaluation et la résolution de contraintes. La contrainte peut être implicite ou explicite. Ainsi si la contrainte est formalisée d'une manière explicite, le système procède à une résolution automatique, dans le cas contraire la résolution n'est que manuelle ; le système fait appel aux concepteurs pour propager les modifications.

– l'axe « degré » identifie le degré prioritaire de violation et de dépendance de contraintes en cas de conflit. Si la contrainte est forte, le système doit envoyer immédiatement un message d'alerte aux acteurs. Ceci donne donc lieu à une notification immédiate. Si la contrainte est faible, le système stocke un message d'alerte pour un envoie ultérieur. Nous appelons ceci une notification différée.

La position d'une contrainte dans ce cube est un indicateur de modélisation des règles métiers pour maintenir la cohérence d'un modèle partagé.

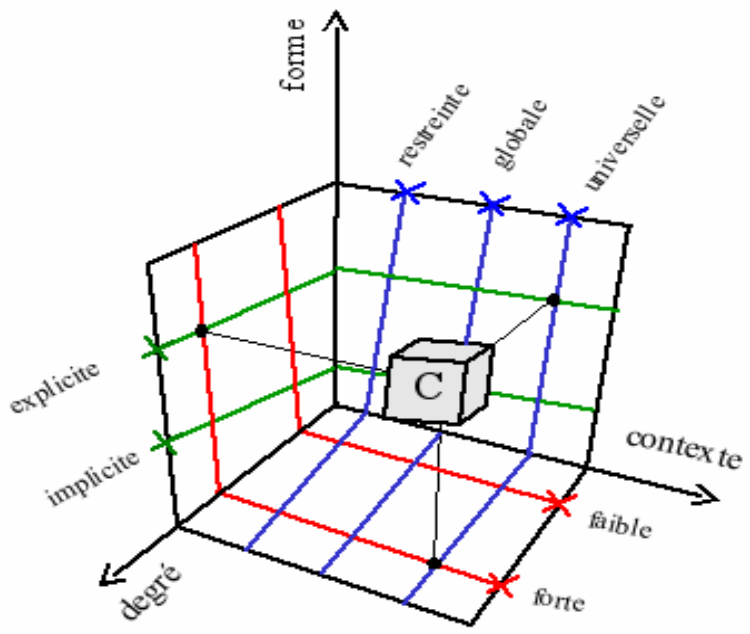


Figure 4.1 Caractérisation des contraintes en conception [Bet06]

Le modèle produit est représenté par l'ensemble des points de vue liés aux différents métiers de la conception. Ces points de vue sont caractérisés par des paramètres de conception afin de préciser la définition conceptuelle du produit. Ces paramètres peuvent être des paramètres structurels, fonctionnels, comportementaux ou technologiques qui sont définies par des variables de conception. Ces variables peuvent être de types différents : symboliques ou numériques, discrètes ou continues, et mixtes. Par exemple : dans la modèle produit de relais, nous pouvons considérer la variable « body-material» représentant le choix de matériau de

culasse (body) comme symbolique puisque son domaine correspond à l'ensemble des noms des matériaux : « body-material»= {'iron', 'steel', 'aluminum'}. Les variables numériques peuvent être : 1) discrètes qui se définissent à partir de listes d'entiers ou de réels, ou bien à partir d'intervalles de valeurs entières ; par exemple la variable « nb-of –spring» représentant le nombre de spires du ressort est un variable numérique discrètes, ou 2) continues ; par exemple la variable "opening-time" est une variable réelle et continue, "opening-time"ε[0, ∞[.

Les contraintes de la conception sont les relations qui restreignent les possibilités de variation des paramètres de conception du modèle produit. Elles limitent l'espace de solutions de configuration en autorisant des combinaisons de paramètres et en interdisant d'autres. C'est à travers elles que les choix de conception transitent pour supprimer les alternatives incompatibles avec les choix effectués.

Les types de contraintes pouvant liées les paramètres d'un modèle du produit sont généralement très divers. Certaines sont locales liées à un métier particulier et d'autres sont globales considérées par au moins deux métiers. Leurs types peuvent être de plusieurs natures [Yan98] : numérique (équation ou inéquation), booléenne, ensembliste, algorithmique, etc.

Dans cette étude, nous avons envisagé une classification des contraintes fondée sur une organisation hiérarchique des contraintes en deux classe de contraintes simples et de contraintes complexes. Cette classification facilite la définition des contraintes à faire apparaître sur des éléments d'un modèle produit :

Les contraintes simples

Les contraintes simples sont celles qui sont prises isolément et représentées par les expressions simples. Il est possible, en étudiant le contenu des contraintes simples, de distinguer quatre types de contraintes:

- **la contrainte mathématique (math. equation) :** Les expressions mathématiques sont essentielles pour calculer des paramètres de conception ou pour représenter des lois physiques et des résultats d'expérimentations. Dans l'exemple du relais, le concepteur représente une loi physique pour calculer la force exercée en position haute du noyau, $F(H_{max})=k\ (L0-L2)-mg$,
- **la contrainte du contrôle de valeur (one value controle)** contrôle les valeurs des variables continus ou discrètes. Pour des variables continus, cette contrainte représente une borne acceptable pour une variable qui soit inférieure limitant dans un intervalle donné. Par exemple, « le temps de déclenchement de relais doit avoir une valeur comprise entre 1ms et 3,5 ms». Pour des variables discrètes, cette contrainte représente une liste de valeur acceptable ou inacceptable pour une variable. Par exemple : le matériau choisi pour la culasse doit être un des matériaux représenté dans cette liste : {'iron', 'aluminum', 'steel'},
- **la contrainte entre deux valeurs (two value relation)** décrit une relation entre deux variables numériques, qui sont liés par un opérateur d'égalité ou d'inégalité. Par exemple, énergie cinétique en arrivant en position haute (E(Hmax)) doit être Supérieure a énergie de percussion minimum (Emin = 0.12 J), $E(Hmax)=1/2\ K\ [(L0-L1)2-(L0-L2)2] + mg\ (L1-L2) \geq 12$ J,
- **la contrainte métier non-formalisé (R-implicite)** : il s'agit d'une représentation implicite de la relation entre les paramètres de la conception qui ne peut pas être exprimée comme une expression mathématique ou logique. Elle doit être considérée par les experts

lors de la vérification des contraintes. Par exemple, le concepteur définit une contrainte métier qui signifie « un traitement thermique rend plus difficile le perçage d'une pièce »,

- **la contrainte d'importance d'objet (object importance)** décrit l'importance d'un paramètre de conception dans le processus de conception. Lorsqu'un expert fait une modification sur ce paramètre, il faut que les autres experts soient d'accord entre eux. Par exemple, la contrainte sur paramètre d'entrefer entre la culasse et le noyau,
- **la contrainte de compatibilité (Compability)** permet de définir les combinaisons de valeurs autorisées pour un ensemble de variables. Les contraintes de compatibilité exprimées en extension sont représentées par des listes de n-uplets indiquant quelles sont les valeurs compatibles entre elles. Par exemple, indiquer que l'épaisseur de la culasse, n'est compatible qu'avec un certain type de matériaux, variable symbolique. Le contrainte va être décrit, en extension, par une liste de couple de valeurs (épaisseur, matériau) : {([0.3, 0.5], 'Iron'), ([0.5, 0.7], 'Steel'), ([0.7, 0.9], 'Aluminium').

Les contraintes complexes

Les contraintes complexes considèrent la combinaison des contraintes simples (élémentaires) avec les liaisons logiques des concepteurs. Elles ont donc une structure différente qui doit les associer à d'autres contraintes. Pour le moment nous n'en utilisons qu'une :

- **La contrainte de type algorithmique, SI… ALORS …SINON…FSI** : cette type de contrainte s'exprime de manière suivante : si (condition) alors <instructions1;> sinon <instructions2;> fsi permet d'effectuer les <instructions1;> lorsque la condition est satisfaite et d'effectuer les <instructions2;> lorsque la condition n'est pas satisfaite. Par exemple :

SI 1/ « 0.3< épaisseur de culasse <0.4 » ALORS
Type-Usinage=X
SINON (1)
Type-Usinage=Y
FIN SI

Dans les sections suivantes, nous allons décrire les implémentations de ces contraintes dans l'environnement de GAM pour ensuite modéliser les contraintes imposées par différents experts lors de la conception du relais et du ventilateur.

4.4. Implémentation du modèle de contraintes dans l'environnement GAM

Les contraintes peuvent être définies au niveau du modèle (Figure 4.2), ou de façon globale au niveau du schéma. Ce modèle de contraintes est lié au modèle produit avec des liens spécifiques, et représente des contraintes sur différents types d'objets du modèle produit. La combinaison du modèle de contraintes et du modèle produit, permet d'avoir l'ensemble des informations sur un modèle intégré. Cette approche permet d'exprimer de manière uniforme les contraintes portant sur des objets du modèle produit. L'expression des contraintes est simple, et s'intègre plus facilement au modèle produit.

Nous maintenons l'expression des contraintes de la conception dans un modèle séparé pour différencier les objets de la conception et des contraintes associées. Cette différentiation facilite la phase de déclaration et de vérification des contraintes et permet de réutiliser des contraintes comme une source de connaissance.

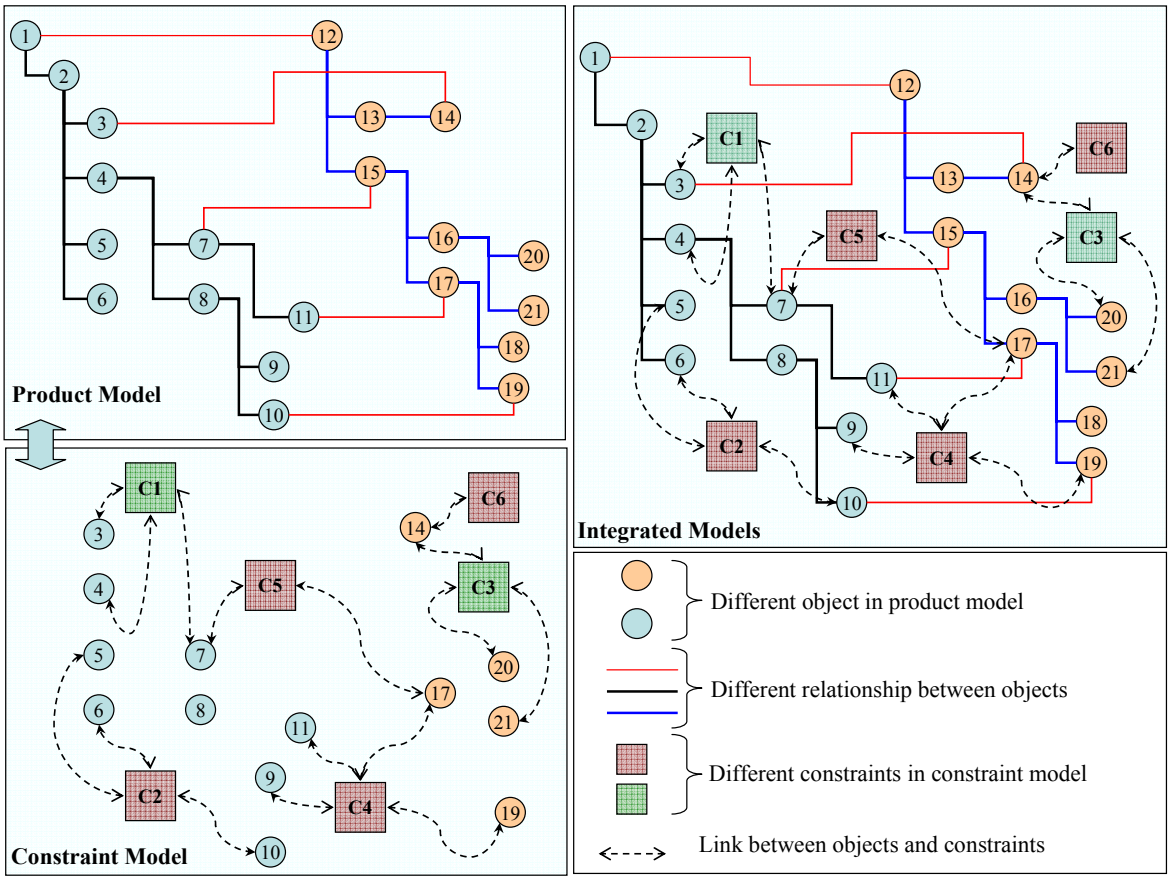


Figure 4.2 Intégration du modèle produit et du modèle de contraintes

Le modèle de contraintes que nous proposons est basé sur le méta-modèle défini dans la Figure 4.3. Ce méta-modèle est organisé en paquet «GAM Constraint» contenant des classes définissant la syntaxe abstraite des expressions des contraintes. Une contrainte est définie par :

- un attribut «objet» qui permet d'effectuer une liste des entités du modèle de produit concernées par des contraintes. Cet attribut est utilisé pour lier les contraintes associées aux entités du modèle produit,
- un attribut «type» qui représente le type de contrainte déclarée,
- un attribut «info.» attribut fournit des informations implicites sur la contrainte,
- un attribut «activation» qui est un attribut de type booléen. La valeur "vrai" indique que la contrainte doit être vérifiée et la valeur «faux» indique que la contrainte ne doit pas être vérifiée lors du processus de vérification des contraintes,
- un attribut «expression » qui définie une expression symbolique de la contrainte qui doit être compatible avec le type annoncé.

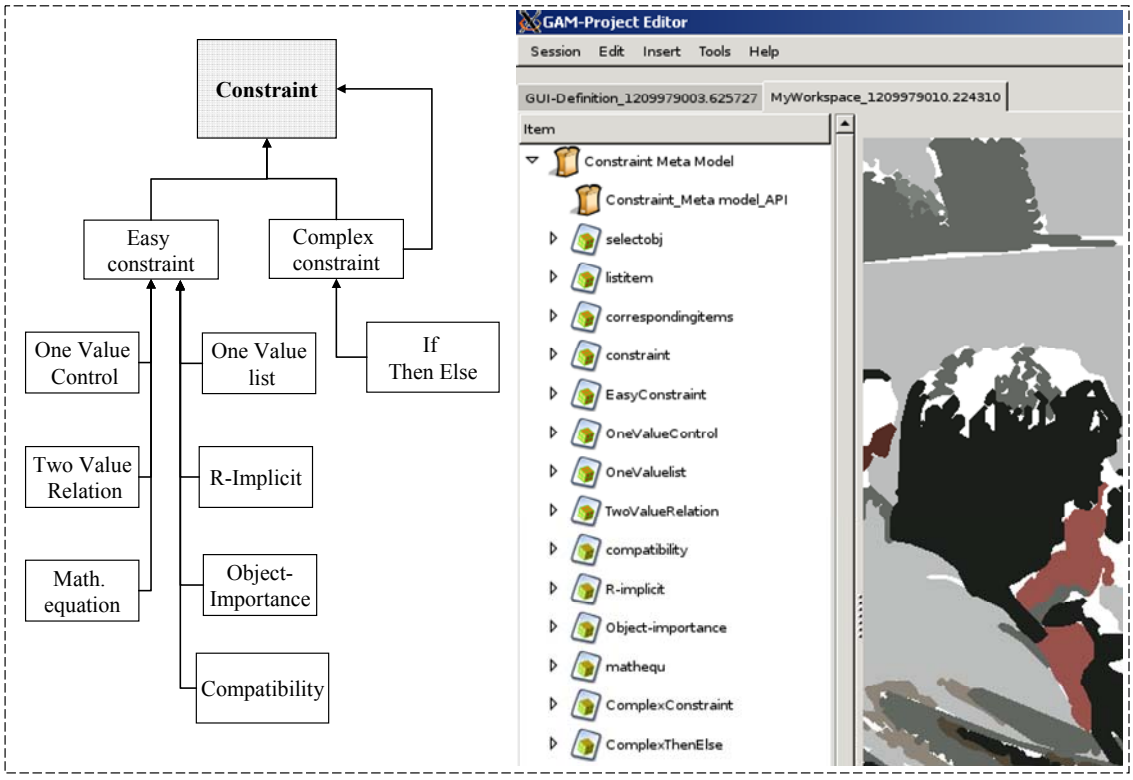


Figure 4.3 Méta-modèle de contraintes dans GAM

La Figure 4.4 présente l'architecture de modélisation de contraintes. L'information peut être capturée grâce à une interface graphique utilisateur et exploitée par un processeur spécifique.

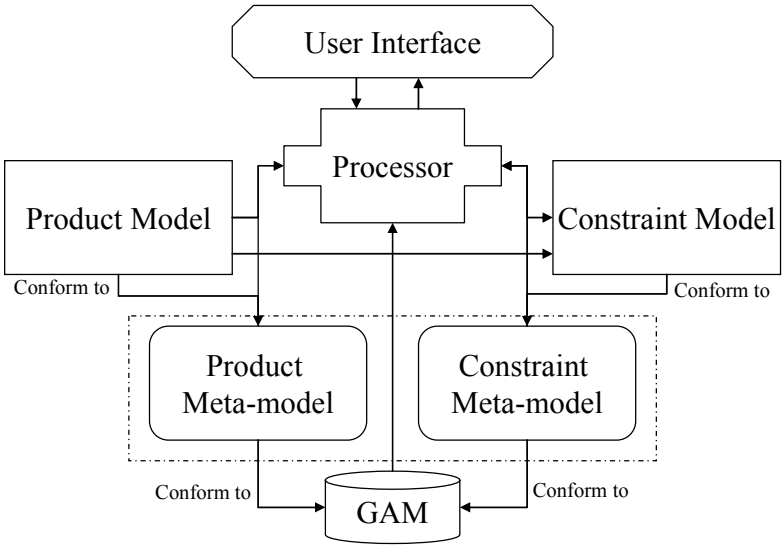


Figure 4.4 Architecture de modélisation des contraintes dans GAM

Ce processeur récupère le méta-modèle, le modèle de produit et également le méta-modèle de contraintes afin d'obtenir les informations nécessaires pour la représentation d'une contrainte. L'objectif est de générer un modèle de contraintes conforme à son méta-modèle. Ce modèle de contrainte est utilisé dans le processus de vérification des contraintes.

Prenons l'exemple du relais électromagnétique, qui est présenté dans le chapitre 3. Cet exemple intègre des éléments mécaniques, électriques, magnétiques et des aspects de

fabrication qui sont très couplés. Les communications aux étapes différentes de la conception ont souligné plusieurs difficultés en travaillant dans un environnement distribué, entre acteurs de cultures différentes. Lors de la conception de ce produit, les acteurs considèrent les différentes contraintes. Afin de remplir le cahier des charges produit pour chaque paramètre d'entrée et de sortie, il faut que tous les acteurs se demandent quelles sont les influences et les contraintes imposées par leur métier. Normalement, de telles contraintes font partie des discussions informelles au fur et à mesure d'un projet de conception, les acteurs essayant d'aboutir à un compromis. Nous allons modéliser quelques exemples de contraintes considérées lors de la conception du relais.

La Figure 4.5, représente l'exemple de modèle de contrainte associe à modèle produit de relais sous forme d'un arbre.

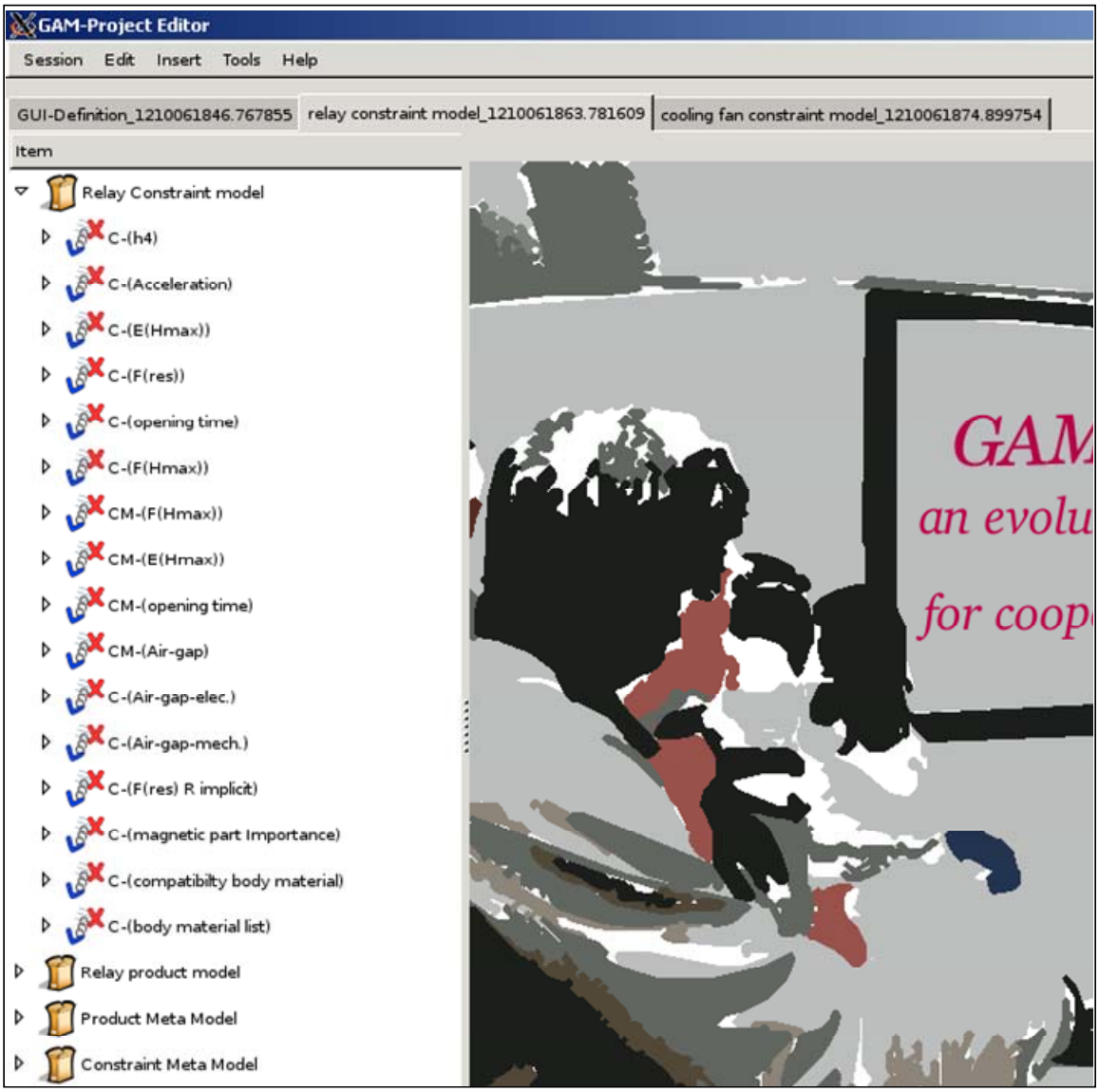


Figure 4.5 Exemple du modèle de contraintes du produit Relais

Ce modèle est liée au modèle produit de relais, au méta-modèle de produit PPO et au méta-modèle de contraintes. Le modèle de contraintes contient différents types de contraintes considérées par les acteurs de la conception (Figure 4.6 et Figure 4.7) :

- la contrainte « C-(h4) » est une contrainte du contrôle de valeur pour la variable continu de « h4 » qui représente l'épaisseur de culasse. La contrainte C-(h4) définit un intervalle de valeur acceptable pour l'épaisseur de culasse entre [2,10]. Si la variable d'épaisseur contient une valeur qui se retrouve en dehors de cet intervalle, la contrainte est violée.
- la contrainte « C-(F (res) R implicit) » est une contrainte de la relation implicite qui signifie la relation implicite entre la variable F(res) et les paramètres de la composant de l'aimante. Cette contrainte montre l'importance des paramètres du dimensionnement de l'aimante pour calculer de la force résiduelle, F (res). Lors de la modification d'un de ces paramètres, cette contrainte doit être déclenchée pour attirer l'attention des experts.
- la contrainte « CM-(E(Hmax)) » est une contrainte mathématique qui représente l'énergie cinétique en arrivant en position haute. Cette contrainte se représente avec l'équation suivante : $E(H_{max}) = \frac{1}{2} k\left[(L_0 - L_1)^2 - (L_0 - L_2)^2\right] + mg(L_1 - L_2)$, cette contrainte dépend de la constante d'élasticité du ressort (K), la longueur du ressort détendu (L0), la longueur du ressort en position bas (L1), la longueur du ressort à en position haute (L2), le poids de noyau (m), et la gravité (g) : on reconnait le potentiel du ressort et celui de la masse. Lors de la modification de un de ces paramètres, cette contrainte détermine la nouvelle valeur de E (Hmax).
- la contrainte « C-(body material list) » : est une contrainte du contrôle de valeur pour la variable discrètes et symbolique de l'objet « material » qui représente le matériau de la culasse. Cette contrainte représente une liste de valeurs acceptables pour la variable « material ». Le matériau choisi pour la culasse doit être un des matériaux représentés dans cette liste : {'iron', 'aluminum', 'steel'}. Si la variable « material » contient une valeur en dehors de cette liste, la contrainte est violée.

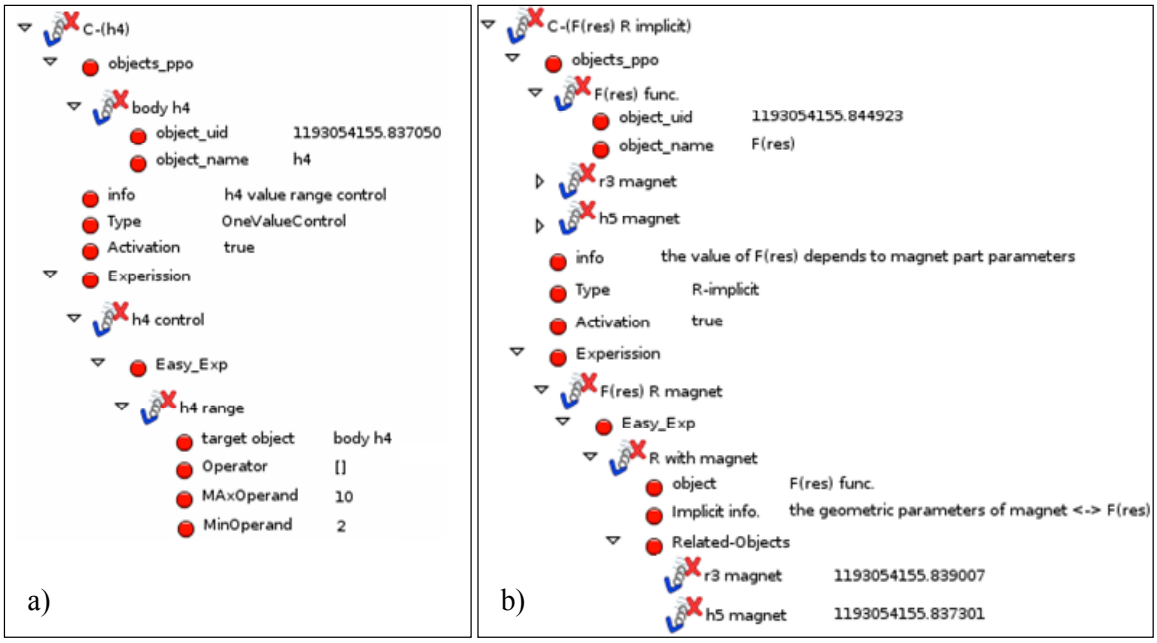


Figure 4.6 Exemples de contraintes

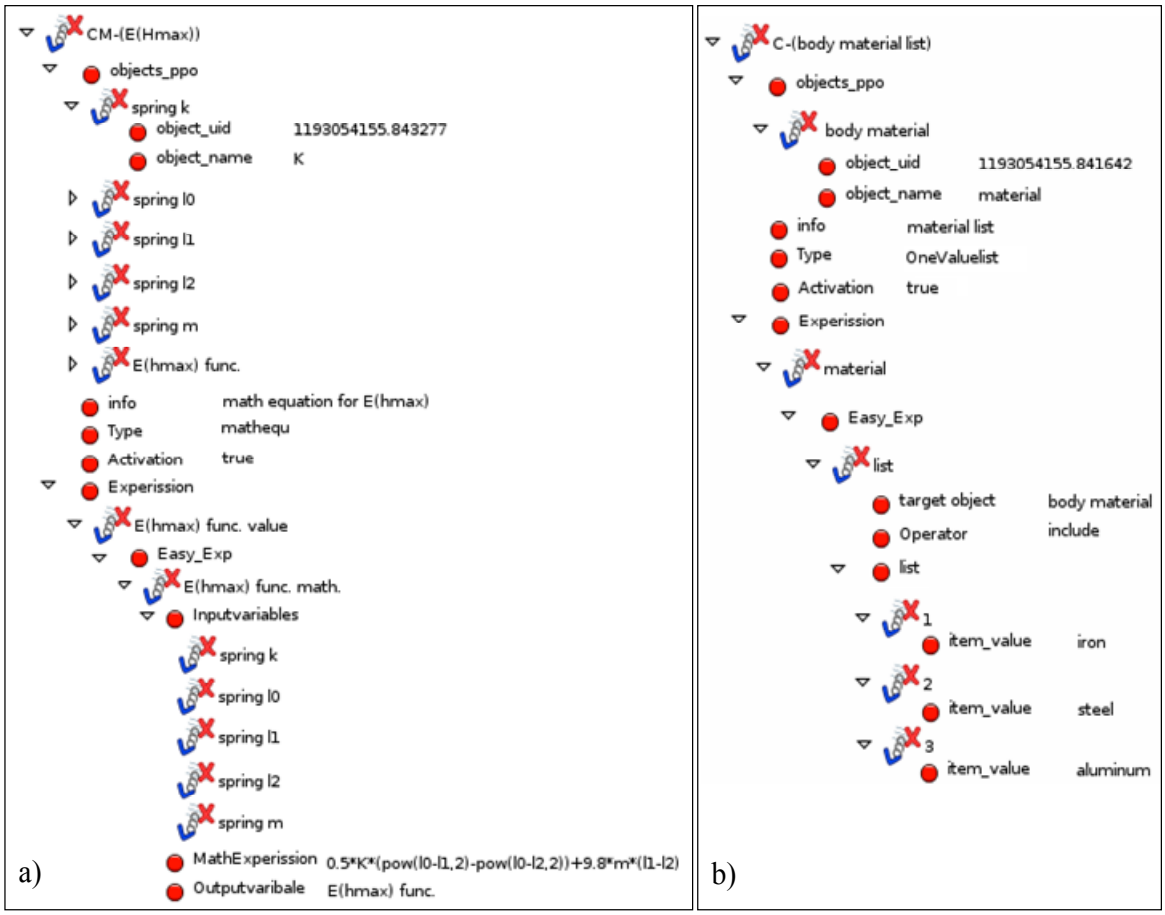


Figure 4.7 Exemples de contraintes

4.5. Vérification des contraints et cohérence des modèles

Un modèle commun partagé est cohérent si toutes les contraintes sont satisfaites. Une phase de synchronisation qui met à jour les objets du modèle partagé doit conserver la validité des contraintes du modèle. La vérification de l'ensemble de ces contraintes, à l'issue de toute mise a jour, est très coûteuse et réduit la disponibilité du modèle partagé. Pour résoudre ce problème, notre travail se situe dans l'approche des méthodes de détection. La solution que nous proposons s'adapte à tout mécanisme de gestion de contraintes utilisant les principes suivants :

Les contraintes métiers sont déclarées de façon globale et déclarative.

- Une analyse syntaxique des versions de modèle partagé et des modèles contraintes permet de réduire l'ensemble de contraintes à vérifier, en déterminant l'ensemble de contraintes risquant d'être violées par une modification,
- La vérification des contraintes est faite automatiquement.

L'analyse des contraintes et des modifications a pour objectif de réduire le nombre de contraintes à vérifier après l'exécution d'une modification du modèle. Dans le chapitre suivant, nous présentons une analyse purement syntaxique des modifications effectuées par chaque acteur. Cette analyse permet de détecter, pour chaque contrainte et chaque

modification, les impliquées. Par la suite, l'analyse d'une contrainte détermine les objets impliqués par la contrainte, alors que l'analyse d'une modification détermine les objets touchés par la modification. Intuitivement, une modification risque de provoquer la violation d'une contrainte lorsqu'elle manipule des structures de modèle manipulées par la contrainte. Ces analyses permettent donc de déterminer, lors de l'analyse d'une modification, l'ensemble des contraintes à vérifier.

4.6. Conclusion

Nous avons présenté, l'utilisation des contraintes et règles métiers comme un mécanisme de vérification de la cohérence des modèles. Le résultat de la phase de contrôle permet de définir les sessions de synchronisation. Ce résultat sera analysé et évalué par d'autres règles métiers (notion de méta-règle) dans le chapitre suivant pour déterminer QUAND synchroniser.

5. Contrôle de l'évolution du modèle partagé pour définir les points de synchronisation

Nous venons de proposer un modèle de représentation des contraintes, nous détaillons maintenant nos propositions de mécanismes de mesure des évolutions parallèles des versions de modèles afin de communiquer aux différents concepteurs l'état de cohérence de l'environnement de conception et la nécessité de provoquer des tâches collaboratives. Nous spécifions les évolutions possibles du modèle partagé par la mise en place de modèles de règles métiers et des mécanismes qui permettent d'évaluer si deux activités parallèles de conception divergent au delà de la simple comparaison syntaxique de modèles. Cette mesure de la divergence est alors utilisée pour notifier en temps voulu les acteurs de la nécessité d'une session de synchronisation et de négociation.

5.1. Évolution des modèles

La Figure 5.1 présente un scénario d'évolution d'un modèle M_{b1} partagé par trois experts. Chaque expert récupère une copie de ce modèle partagé pour l'exploiter dans un espace de travail local. L'édition concourante de plusieurs copies locales permet à plusieurs experts de travailler simultanément sur un même modèle à différents endroits et à différents moments. Sur la Figure 5.1 nous supposerons que l'expert E_1 intervient en premier en modifiant sa copie locale du modèle jusqu'à l'instant t_{m0}. Ensuite les deux experts E_2 et E_3 modifient à leur tour, leur propre copie. Afin d'éviter des conflits, les deux experts E_2 et E_3 travaillent simultanément sur leur copie pendant la période $[t_{m1}, t_{m2}]$ et pourraient avoir connaissance des modifications apportées par l'expert E_1. De plus, ils pourraient être au courant des modifications effectuées par chacun durant cette même période. Enfin, si l'expert E_1 reprend le travail à l'instant t_{m3}, il doit connaître à son tour les modifications apportées par les deux experts E_2 et E_3.

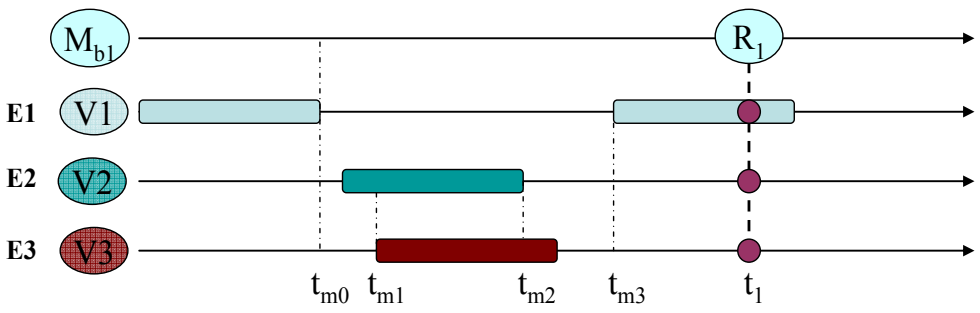


Figure 5.1 Exemple de scénario d'évolution de modèles

Quand les experts travaillent sur leur copie locale de manière asynchrone, il est impossible d'avoir les informations sur les évolutions des copies privées. S'ils continuent de travailler sur leur copie sans aucun mécanisme de contrôle des évolutions des modèles ou sans aucun mécanisme de mesure des divergences, l'intégration des changements effectués par chaque expert au modèle partagé, devient une tâche difficile. Les experts ont besoin d'un mécanisme qui contrôle les évolutions de leur copie locale par rapport au modèle partagé pour définir les sessions de synchronisation et d'identifier les moments opportuns pour synchroniser. Par exemple, dans la Figure 5.1, à l'instant t_1, un mécanisme de contrôle informe les experts qu'ils doivent synchroniser leurs copies avec le modèle partagé. De plus, les experts ont besoin d'un outil informatique pour assister cette phase de synchronisation et de négociation. Cet outil doit être capable d'une part de comparer la copie d'un modèle avec l'original pour représenter les différences et les changements apportés par un expert, et d'autre part d'analyser ces différences pour détecter et évaluer les incohérences et les conflits entre experts.

Nous proposons ici, un cadre conceptuel pour contrôler les évolutions des copies d'un modèle partagé et résoudre les problèmes liés à la cohérence et à l'évolution des modèles. Ce cadre conceptuel peut être intégré à un environnement coopératif existant ayant pour but de recueillir, de présenter et de distribuer les changements effectués par chaque expert sur le modèle partagé. Dans le cadre de nos travaux, nous optons évidemment pour l'environnement GAM.

Pour supporter l'évolution de ces modèles, nous proposons dans ce chapitre une démarche caractérisée par :

- un modèle de différence qui repose sur le concept de distance entre un modèle partagé et sa version (*Diff*). Ce modèle est utilisé pour représenter les évolutions apportées par chaque expert métier par rapport au modèle de base,
- un indicateur d'évaluation des modèles de différences (méta-règle) qui permet de définir des valeurs limites pour la notion de distance et qui détermine les moments de synchronisation des versions avec le modèle partagé.

5.2. Gestion des évolutions des modèles de la conception

La gestion des évolutions des modèles de la conception permet de gérer les modifications des différents éléments traités par les différents experts. Il permet en particulier de gérer les versions du modèle partagé et d'assurer la traçabilité et l'intégration de ces modifications afin de maintenir la cohérence du modèle partagé.

Le processus de conception est évolutif, les modèles de produit et les contraintes évoluent suite à des créations, des suppressions ou des modifications fréquentes des éléments. Chaque expert travaille sur une copie locale du modèle partagé (produit et contraintes) et à certaines périodes du processus de conception, synchronise sa copie pour être à jour (Figure 5.2). La démarche de gestion des évolutions des modèles est basée sur un ensemble de mécanismes visant à fournir des informations sur les modifications effectuées : quand, comment, où et qui a procédé à un changement dans le modèle partagé. Il est important de savoir d'une part comment les modèles de produit et de contraintes évoluent au cours du processus de

conception et d'autre part, comment et quand la demande de synchronisation est prise en compte.

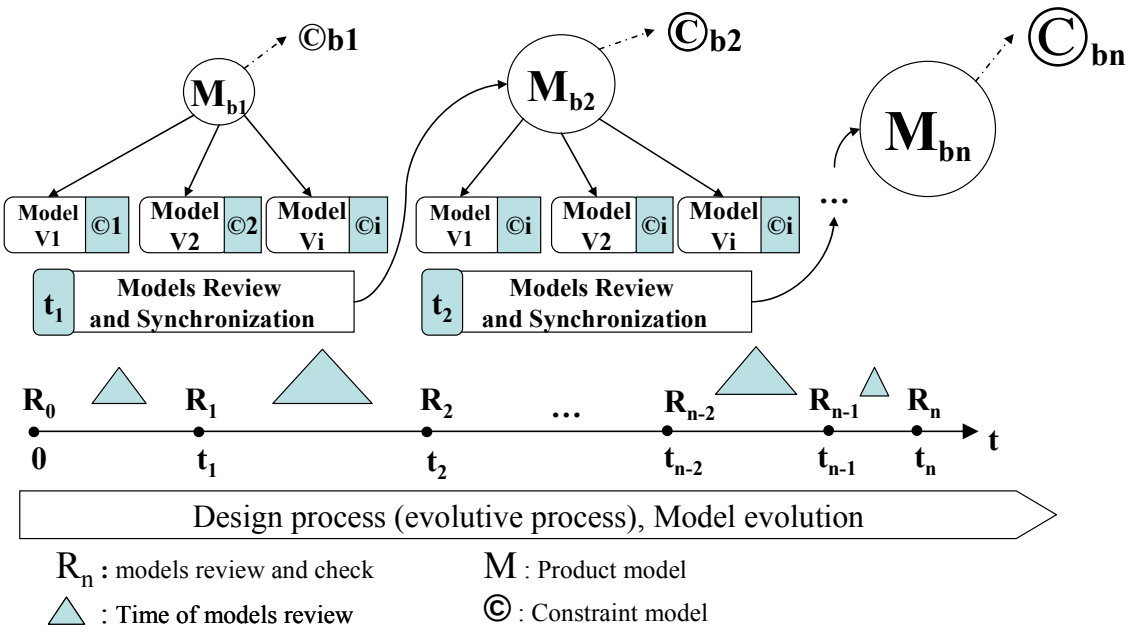


Figure 5.2 Processus d'évolution des modèles produit et contraintes

Les modèles de produit et de contraintes sont utilisés pour stocker et échanger les données de la conception. Les experts produisent les différentes versions du modèle partagé pendant une phase de conception plus au moins longue. La détection de changements entre le modèle partagé et les versions modifiées, est une fonction importante. Chaque expert doit connaître les modifications apportées par les autres experts sur le modèle. L'intégration des modifications et des contraintes de la conception apportées par chaque métier, introduit de nouvelles exigences pour faire évoluer le modèle commun. Au cours d'une session de synchronisation, tous les changements validés seront mis en œuvre dans la future version du modèle partagé.

La Figure 5.3, montre la dynamique de l'évolution du modèle partagé au cours de l'avancement du projet de la conception. Dans les premières phases d'un nouveau projet de conception, les experts ont moins de connaissances et de contraintes relatives aux solutions de conception, ce qui induit plus de degrés de liberté dans les choix de solutions de conception. Au fur et à mesure de l'avancement du processus de conception, les choix effectués et les contraintes imposées par chaque métier limitent les degrés de liberté sur les choix qui restent à faire. Les évolutions de l'ensemble des modèles de produit et de contraintes déterminent l'état d'avancement du processus de conception jusqu'à la solution finale. Toute évolution des modèles satisfaisant l'ensemble des contraintes métiers représente une solution acceptable.

Lors de l'évolution des modèles, les experts se retrouvent parfois dans des situations conflictuelles :

– pour satisfaire les objectifs de la conception, les experts ne disposent pas de la même connaissance et des mêmes principes concernant la compréhension du produit et des problématiques liées à sa conception,

– la complexité du processus de conception durant la prise en compte des besoins de la clientèle, le couplage fonctionnel entre deux métiers, la prise en compte des contraintes et durant le choix de méthodologie de la conception.

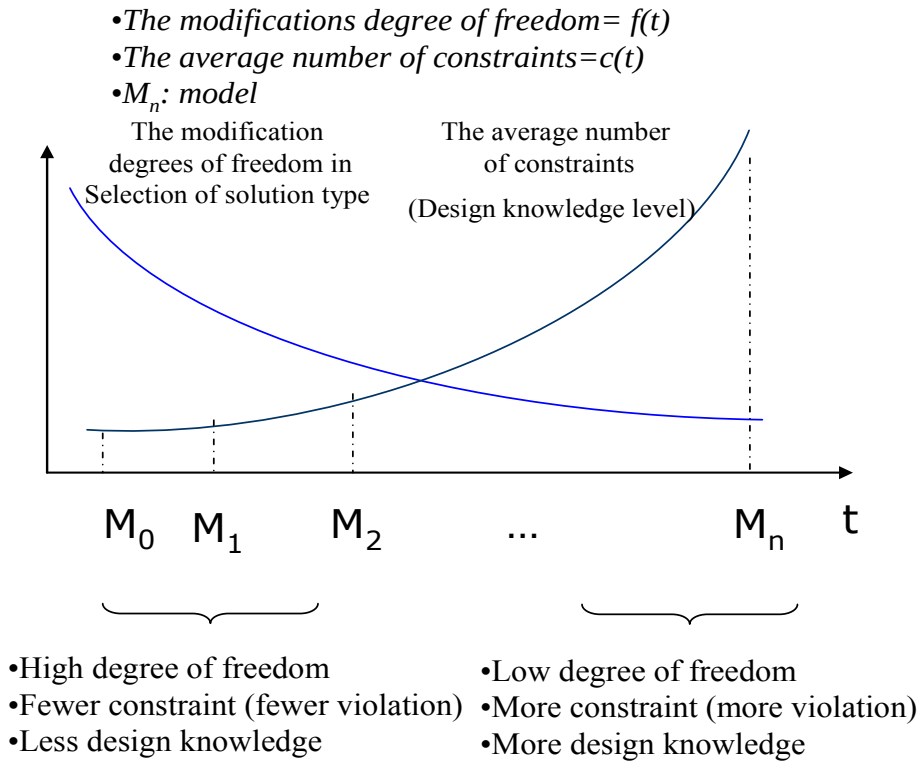


Figure 5.3 Evolution des contraintes et des degrés de liberté en conception

Pour faciliter l'avancement de la conception, les experts ont besoin de détecter les conflits, de comprendre leurs causes et leurs mécanismes, et enfin de se donner les moyens pour anticiper ces conflits et d'engager ainsi l'action pour les résoudre.

5.3. Gestion des évolutions des modèles dans GAM

La synchronisation nécessite la représentation des différences entre chaque version (copie du modèle) et le modèle de base ($Diff_i$) pour ensuite les évaluer afin de définir les changements nécessaires à apporter au modèle partagé (Figure 5.4). Les évolutions des versions sont propagées tout en respectant les contraintes de chaque expert en assurant la cohérence du modèle partagé.

En conséquence, il est nécessaire de tenir compte des mécanismes qui évaluent les évolutions des versions afin de garantir la cohérence globale du modèle partagé. Pour atteindre cette exigence, les mécanismes de l'évaluation doivent fournir les informations suivantes :

– l'identification des modifications est faite sur chaque version,

– l'identification de l'origine des conflits est faite de manière automatique lors de l'analyse des modifications.

Ces mécanismes servent à fournir les informations nécessaires pour la phase de synchronisation et de négociation.

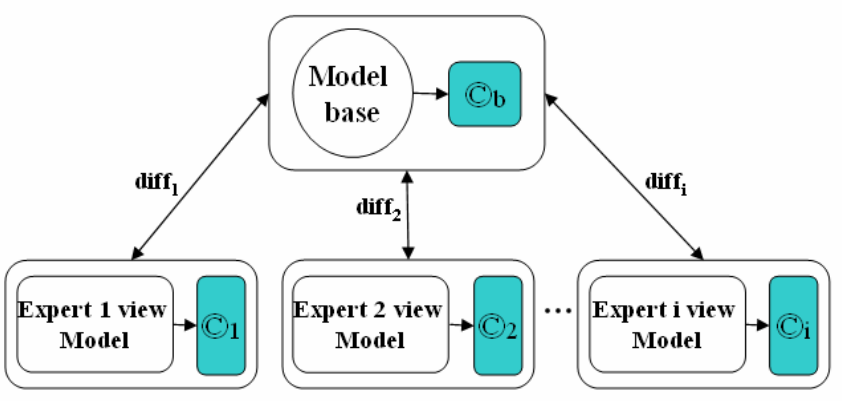


Figure 5.4 Versions parallèles d'un modèle partagé

5.3.1. Représentation des différences

Les experts travaillent dans l'environnement coopératif de manière asynchrone sur différentes parties du modèle global, donc les différentes versions du modèle partagé peuvent apparaître à tout moment au cours du processus de conception. Le versionnement permet :

- de revenir à un état amont dans le processus de conception,
- à plusieurs experts de travailler sur le modèle en même temps,
- de conserver la trace et l'historique des modifications, par exemple les modifications effectuées pour passer d'une version au modèle partagé,
- l'identification des différences entre les versions afin de choisir les modifications acceptables,
- aux experts d'intégrer les différentes modifications.

5.3.1.1. Notion de différence

Pour gérer les différentes versions, il faut identifier les différences entre ces versions et le modèle original (modèle partagé) afin de synchroniser les versions avec le modèle partagé. Le processus de synchronisation requiert deux étapes principales :

- la première étape, consiste à calculer la différence entre deux versions noté *Diff* , qui permet l'extraction des modifications faites sur une copie privée,
- la deuxième étape, permet l'intégration de *Diff* dans le modèle partagé.

La Figure 5.5 montre le processus de calcul de *Diff* . D'une part (M_b, C_b) représentent les modèles de base partagés respectivement de produit et de contraintes. D'autre part (M_i, C_i) représente la copie de l'expert i, avec $i = 1,...,n$ et n étant le nombre d'experts impliqués dans le processus de conception collaborative.

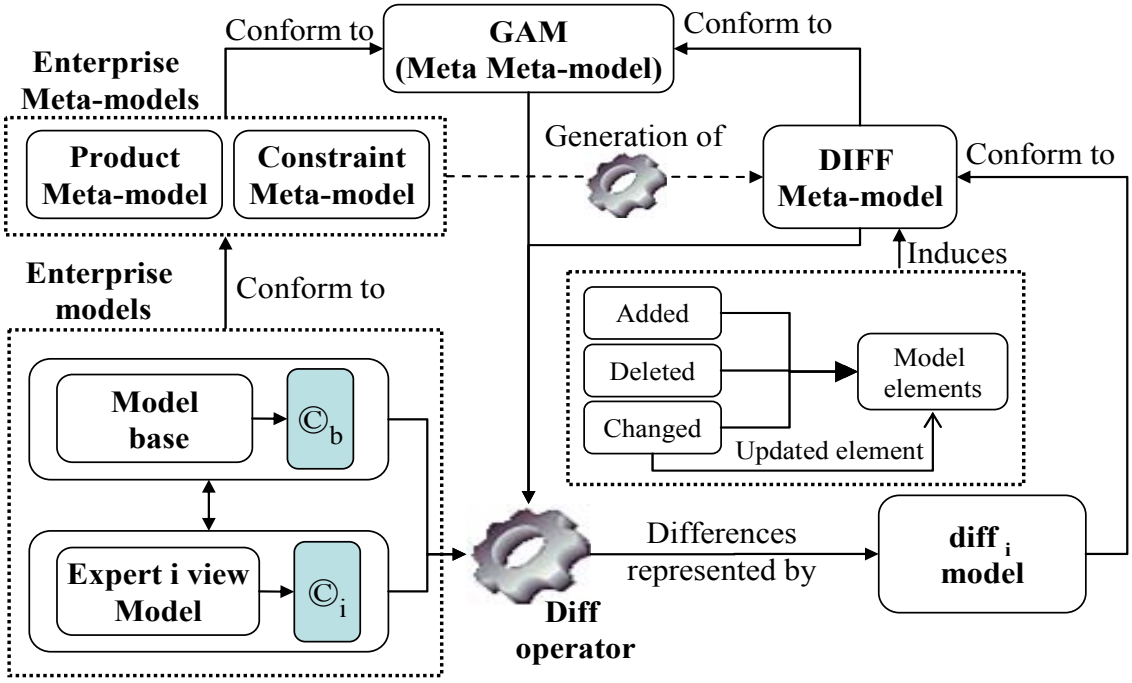


Figure 5.5 Processus de calcul des différences entre modèles

Les modèles (M_b, C_b) et (M_i, C_i) sont conformes à des méta-modèles (Produit et Contrainte). Leur différence est conforme à un autre méta-modèle (Méta-modèle *Diff*). Cette difference est automatiquement générée en utilisant les méta-modèles (produit et contraint). En particulier, le nouveau méta-modèle est retenu pour fournir un modèle de différence (le modèle $Diff_i$) afin d'exprimer les modifications effectuées sur le modèle de base et représentant la version de l'expert i.

Le premier pas vers cet objectif est de comprendre les types d'évolutions ainsi que leurs conséquences sur le modèle partagé. Les évolutions possibles d'un modèle sont de plusieurs natures :

- **Ajouts** : de nouveaux éléments sont ajoutés à la version locale d'un expert et qui ne sont pas présents dans le modèle partagé,
- **Suppressions** : des éléments existants dans le modèle partagé sont supprimés dans la version locale,
- **Modifications** : des éléments déjà existants dans le modèle partagé sont mis à jour dans la version locale d'un expert. Par exemple, la modification de la valeur d'un attribut.

L'opérateur *Diff* utilise des algorithmes de comparaison de modèles pour la détection des correspondances et les différences entre deux modèles en comparant tous les éléments et leurs propriétés. L'opérateur *Diff* peut être formalisé par :

$$Diff_i = (M_i - M_b; C_i - C_b), \qquad \text{avec } i = 1,...,n$$

$$Diff_{Totale} = \sum_{i=1}^{n} Diff_i$$

Le $Diff_i$ est le modèle de différence et représente toutes les différences entre le modèle de l'expert i (une version) et le modèle de base partagé. Ces différences concernent à la fois

celles du modèle produit et du modèle de contraintes. Le $Diff_{Totale}$ est une agrégation de tous les modèles de différences permettant d'évaluer les modifications apportées par tous les experts.

Cet opérateur est adapté aux caractéristiques et typologies de la structure des modèles existantes dans l'environnement coopératif. En effet, dans le contexte de nos travaux, les données d'un modèle sont représentées par un graphe. Les nœuds et les arcs représentent respectivement les objets et les relations entres eux.

Les algorithmes de comparaison utilisent ces informations pour identifier des différences afin de visualiser le résultat sous la forme du modèle de différence. Le contenu des modèles de différence est essentiel pour la compréhension des correspondances et des différences entre deux versions de modèles pour ensuite communiquer et négocier ces modifications.

L'opérateur $Diff_i$ identifie la différence entre le modèle de base (M_b, C_b) et la version (M_i, C_i) de l'expert i. Les évolutions possibles d'un modèle (ajout, suppression et changement) sont réalisées par un ensemble d'opérateurs. Chaque opérateur est associé à un type d'évolution. Par exemple, l'opérateur « Renommer Elément » est de type changement qui caractérise le changement du nom d'un élément du modèle de base dans le modèle version.

Afin d'intégrer l'ensemble des modifications ($Diff_{Totale}$), apportées par tous les experts, dans la modèle partagé, on est naturellement tenté d'écrire l'opérateur suivant :

$$(M_b, C_b)_{new} \leftarrow (M_b, C_b) + Diff_{Totale}$$

$$(M_b, C_b)_{new} \leftarrow (M_b, C_b) + \sum_{i=1}^{n} (M_i - M_b; C_i - C_b)$$

Le $(M_b, C_b)_{new}$ représente le modèle partagé final après l'intégration des modifications de chaque expert. En pratique, l'application de cet opérateur est très difficile du fait que les experts peuvent apporter des modifications aux mêmes objets du modèle partagé, nous avons alors souvent :

$$\bigcap_i Diff_i \neq \varnothing$$

L'opération précédente conduit donc nécessairement à des incohérences et il est impossible de prendre en compte toutes les modifications dans le modèle partagé final. Ces incohérences et ces conflits doivent être identifiés, enregistrés et suivis jusqu'à leur résolution. À cette étape, la vérification de la cohérence est fondamentale et l'intégration des modifications nécessite la mise en place d'outils d'aide à la synchronisation et à la négociation pour que chaque expert prenne part à la discussion, à la résolution de conflits et enfin aux choix des modifications à apporter au modèle partagé.

Dans la suite de ce chapitre, nous présentons une approche de comparaison de modèles (base et version) et de détection de changements sur le modèle version afin de les représenter dans un modèle de différence qui n'est autre qu'une instance du méta-modèle $Diff$.

5.3.1.2. Détection des changements

Les changements se produisent parfois sur les mêmes éléments et peuvent également être la cause de violation de contraintes. Ces changements sont alors en conflit. Dans ce cas, il ne peut pas être décidé automatiquement quels changements sont à prendre en compte dans le

modèle partagé pour éviter ces conflits. Les experts doivent négocier pour résoudre les conflits.

L'approche de détection des changements repose sur l'idée que l'évolution d'un modèle peut avoir des impacts sur le contenu du modèle Figure 5.6 :

– en modifiant ou en éliminant des éléments du modèle, les caractéristiques des éléments ou les relations existantes entre éléments ou entre leurs caractéristiques. Par exemple, si un expert décide de supprimer un objet (Obj-4) ou une contrainte (C4), ou de modifier la valeur d'un objet (Obj-5).

– en ajoutant de nouveaux éléments, de nouvelles caractéristiques ou de nouveaux liens. Par exemple, si un expert décide d'ajouter un objet (Obj-12) ou une contrainte (C5).

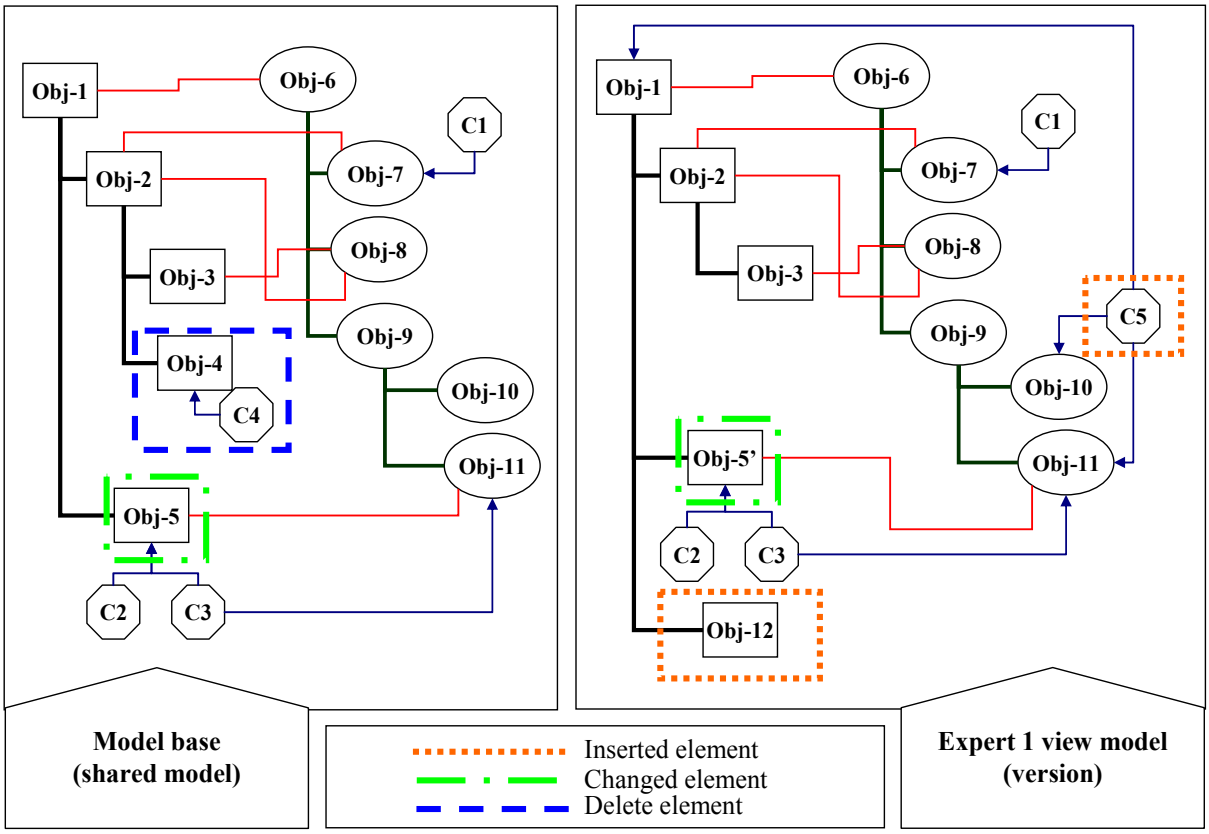


Figure 5.6 Approche de détection des modifications apportées par les experts

Les impacts des changements apportés au modèle peuvent être soit directs, en modifiant la structure ou la caractéristique du modèle ou indirects, en violant les règles métiers. Par conséquent, pour détecter les différents changements ainsi que leurs impacts lors de l'évolution d'un modèle, il faut :

– trouver les différences syntaxiques qui se profilent à partir de la comparaison des graphes d'informations associées au modèle de base et à chaque version,

– mettre en jeu les contraintes et règles métiers à l'origine des changements indirects.

I. Comparaison des graphes d'informations

Les graphes sont considérés comme une approche très riche pour la modélisation en ingénierie. Un modèle à base de graphe est défini par un ensemble de sommets qui représentent les éléments du modèle, et un ensemble d'arcs qui modélisent les différentes relations entre ces éléments. Les éléments et les relations peuvent être décrits par un ensemble d'identificateurs, c'est-à-dire qu'il est possible d'associer une ou plusieurs valeurs symboliques ou numériques à chaque sommet ou chaque arc afin de spécifier les différents types d'éléments et de relations.

Les objets de conception dans les modèles produit et contraintes sont décrits en termes d'entités (composants, interfaces, fonctions, etc.) et de relations entre ces entités. Par conséquent, afin de comparer ces objets, nous proposons de les modéliser par des graphes multi-identificateurs.

Par exemple, chaque objet des modèles produit ou de contraintes est décrit par :

- un "**UID**" (Unique IDentifier) comme identifiant unique, (propriété native de l'environnement GAM),
- un "**Classifier**" qui représente un ensemble d'objets, déclaré comme des UID de class en méta-modèle et signifie le "Type" d'objet qui est un ensemble de caractéristiques communes à des objets,
- un "**Nom**" qui l'identifie de manière unique et textuelle,
- des "**Attributs**" qui sont des éléments de la structure des objets. Ces sont des variables définies dans le "Classifier" et qui décrivent ces objets.

La Figure 5.7 décrit la sauvegarde textuelle d'un objet du modèle produit de l'exemple du Relais. Il est caractérisé par un "UID" de valeur "1193054155.826032", un "Nom" égal à "Cylinder_Head", un "Classifier" définit par "UID" de la classe "Composant" dans la méta-modèle du produit et qui représente le "Type" de l'objet dans la modèle produit, et trois "Attributs" qui caractérisent cet objet : "interface", "sub_component" et "decompostion_type".

```
<Class name="Component" abstract="false" leaf="false" uid="1156792173.9700861- 11
    56792173.9700840-1156792183.9592700">
  <Attribute name="interface" multiplicity="0..*" mode="mandatory" unique="false"
    ordered="false" classifier="1156792173.9700861-1156792173.9700840-
    1156792183.9631250" />
  <Attribute name="sub_component" multiplicity="0..*" mode="mandatory" unique="false"
    ordered="false" classifier="1156792173.9700861-1156792173.9700840-
    1156792183.9592700" />
  <Specialise>
    <Class ref="1156792173.9700861-1156792173.9700840-1156792183.9616649" />
  </Specialise>
</Class>
```

a)

```
<Object-uid="1193054155.826032",name="cylinder_head",classifier="1156792173.
    9700861-1156792173.9700840-1156792183.9592700">
  <Attribute name="interface">
  <Attribute name="sub_component" />
  <Attribute name="decomposition_type" value="CC" />
</Object>
```

b)

Figure 5.7 Exemple de description sur GAM d'un objet d'un modèle produit

Par conséquent, la Figure 5.6 montre que le modèle version de l'expert 1 présente des changements par rapport au modèle de base. Les algorithmes de comparaison de modèles utilisent les informations associées à chaque élément des deux modèles à comparer pour identifier les différences entre elles. L'identifiant unique "UID" est un élément important pour la comparaison de modèles. Nous pouvons distinguer les différentes opérations d'ajout, de suppression ou de modifications effectuées sur la structure du modèle de base pour aboutir à la structure du modèle version d'un expert, en comparant les "UIDs" des éléments entre les deux modèles de la manière suivante :

- **Ajout d'éléments** : les nouveaux éléments à ajouter sont des éléments qui n'ont pas encore cet UID dans la modèle de base. Nous noterons ce type de différence par $New(e^2)$, où e^2 représente l'élément ajouté au modèle version d'un expert et qui n'existe pas dans le modèle de base. Dans l'exemple de la Figure 5.6, les éléments "Obj-12" et "C5" sont deux nouveaux éléments ajoutés au modèle version de l'expert 1 et qui ne figurent pas dans le modèle partagé.
- **Suppression d'éléments** : les éléments supprimés sont des éléments ayant des UID dans le modèle de base mais pas dans la modèle version de l'expert. Nous noterons ce type de différence par $Delete(e^1)$, où e^1 représente l'élément supprimé du modèle version de l'expert et qui existe dans le modèle de base. Dans l'exemple de la Figure 5.6, les éléments "Obj-4" et "C4" sont deux éléments supprimés et par conséquent n'apparaissent pas dans la modèle version de l'expert.
- **Modification d'éléments** : les éléments modifiés sont des éléments qui existent et ont les mêmes UIDs dans les deux modèles de base et de l'expert mais qui ont soit des noms différents ou qui ont au moins une valeur de l'un de leurs attributs est différente. Nous noterons ce type de différence avec : (1) $Change(e, c^n, n^1, n^2)$, où e représente l'élément existant avec un même UID dans les deux modèles, c^n représente le changement de type modification de nom, n^1 est le nom de l'élément dans le modèle de base et n^2 est le nom de l'élément dans le modèle version de l'expert, (2) $Change(e, c^v, a^n, v^1, v^2)$, où e représente l'élément existant avec un même UID dans les deux modèles, c^v représente le changement de type modification de valeur de l'attribut a^n, v^1 correspond à la valeur de l'attribut a^n dans le modèle de base et v^2 sa valeur dans le modèle version de l'expert. Dans l'exemple de la Figure 5.6, l'élément "Obj-5" du modèle de base a été modifié dans le modèle version de l'expert et apparaît avec un autre nom "Obj-5'". Les deux objets "Obj-5" et "Obj-5'" sont identifiés avec un même UID.

II. Vérification des contraintes associées

Nous pouvons définir à priori des structures de contrôle suite aux différents changements sur les modèles. Ces structures de contrôle peuvent être mises à contribution pour propager les effets d'un changement en vérifiant les contraintes relatives aux éléments modifiés. Certaines contraintes sont considérées comme un obstacle pour le processus de conception. En effet, elles peuvent entraîner des "conflits interdisciplinaires", par exemple entre les concepteurs qui travaillent dans le domaine mécanique et ceux qui travaillent dans le domaine électronique. Les actions de coopération entres les différents concepteurs sont poussées par un sentiment de recherche des causes des conflits, qui tend à favoriser une organisation de synchronisation dirigée par l'ensemble des experts. Par contre, l'intégration des modifications exige une compréhension plus large des rapports entre ces modifications et les règles représentées par chaque acteur métier.

Les modifications génèrent des conflits lorsque ces derniers deviennent la cause de violation de contraintes interdisciplinaires. La vérification des contraintes interdisciplinaires devient fondamentale pour assurer la cohérence du modèle partagé après l'intégration des différentes versions.

Quand une opération de modification porte sur un modèle, les algorithmes de comparaison des graphes sont utilisés pour déterminer quelle partie et quels éléments du modèle sont effectivement modifiés. Si une opération de modification est effectuée sur un objet x, nous pouvons alors trouver dans le modèle d'autres objets à travers la relation de dépendance qu'ils ont avec l'objet x. Toutes les contraintes liées à tout objet dans cette relation de dépendance doivent alors être représentées et vérifiées. Pour évaluer la notion de cohérence, des fonctions telles que $find(x)$ et $evaluate(x)$ doivent être définies pour la modélisation et la propagation aux autres éléments des effets d'un changement d'un élément. La fonction $find(x)$ utilise les différences syntaxiques présentées précédemment pour trouver les éléments modifiés ainsi que les contraint qui leurs sont associées. Sur la base des résultats de la fonction $find(x)$, la fonction $evaluate(x)$ est utilisée pour vérifier la consistance des contraintes associées aux objets modifiés. La mise en œuvre de la fonction $evaluate(x)$ dépend essentiellement de la nature des contraintes traitées. Par exemple, la fonction $find(x)$ retrouve un élément x pour lequel la valeur d'un attribut a été modifiée et retrouve ensuite la contrainte qui définit le domaine acceptable pour les valeurs de cet attribut. La fonction $evaluate(x)$ fait appel à un algorithme, spécifique à ce type de contrainte, pour vérifier si la contrainte n'est pas violée, c'est-à-dire si la nouvelle valeur affectée à l'attribut appartient au domaine acceptable. La fonction $evaluate(x)$ retourne une valeur du statut de la contrainte : violer ou non-violer.

5.3.1.3. Descriptions des changements

Nous présentons les différences entre le modèle de base et chaque modèle expert dans un modèle de différences propre à chaque version. Notre objectif est de capturer les modifications effectuées par chaque expert sur les modèles produit et contraintes et de les formaliser ensuite par un modèle de différences $Diff_i$ propre à chaque acteur métier. Dans un premier temps, nous définirons une représentation des différences. Afin de supporter efficacement l'évolution des modèles, nous proposons de construire un méta-modèle de différences que nous notons $DIFF$. Ce méta-modèle utilise précisément les propriétés réflexives des représentations d'objets dans les modèles produit et contraintes.

I. Le méta-modèle de différences

Le méta-modèle de différences $DIFF$ permet de représenter et de visualiser les différences entre la version d'un modèle produit ou d'un modèle de contraintes avec l'original (partagé). La Figure 5.8 décrit la structure générale du méta-modèle $DIFF$ et ses différents liens avec le méta-modèle du produit et le méta-modèle contraintes. Les algorithmes proposés pour la détection des changements utilisant le méta-modèle $DIFF$ afin d'enregistrer les résultats.

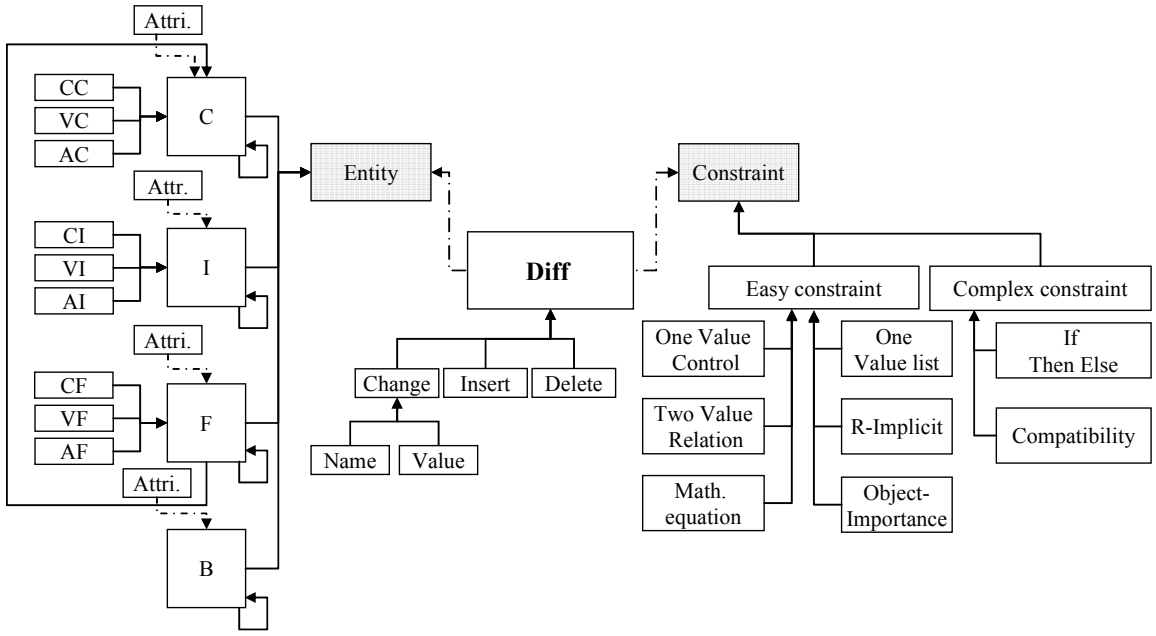


Figure 5.8 Structure générale du méta-modèle de différences

La Figure 5.9 montre les différentes classes du méta-modèle *DIFF* avec leurs attributs dans l'environnement GAM:

- La classe "**objectinfo**" est une classe générale déclarant les éléments nécessaires pour l'identification et la représentation d'un objet sur lequel porte un opérateur de changement. Les attributs "object name", "object UID" et "object type" décrivent l'identité d'un objet. L'attribut "operation type" représente le type de changement adressé à un objet.
- La classe "**object-value-modification**" enregistre des informations sur la modification de valeur d'un objet et des contraintes relatives à cet objet. Par exemple : l'attribut "value on side1" représente la valeur d'un objet dans la modèle de base avant la modification, l'attribut "value on side2" représente la valeur du même objet dans le modèle versionné après la modification, et l'attribut "related constraints" représente toutes les contraintes associées à l'objet ainsi leurs états (violée ou non-violée). L'attribut "related constraints" est associé à une classe du même nom.
- La classe "**object-name-modification**" donne des informations sur le changement de nom d'un objet,
- La classe "**object-insert-delete**" est utilisée pour représenter un objet ajouté ou supprimé dans modèle, ainsi que toutes les contraintes associées à l'objet ajouté ou supprimé,
- La classe "**constraintdiff**" décrit des opérations d'ajout, d'activation, de désactivation ou de modification d'une contrainte dans le modèle versionné,
- La classe "**results**" résume toutes les données quantitatives relatives à l'ensemble des modifications effectuées par un expert sur le modèle produit et son modèle de contraintes associé. Nous définissons plusieurs attributs correspondants aux différents types de modifications :
  - Sur le modèle produit :
    - nombre d'opérations de modification de valeurs d'attributs,
    - nombre d'opérations de modification de noms d'éléments,

- nombre d'éléments ajoutés,
- nombre d'éléments supprimés,

- Sur le modèle de contraintes :
  - nombre de contraintes violées,
  - nombre de contraintes ajoutées,
  - nombre de contraintes supprimées,
  - nombre de contraintes modifiées.

Figure 5.9 Les classes du méta-modèle de différences avec les attributs associées.

La présentation des différences dans GAM permet de visualiser et de comprendre les différents changements effectués pour préparer la fusion. En effet, les contributions individuelles sur le modèle partagé sont propagées à tous les experts, offrant une vue d'ensemble de ce qui est en cours de réalisation et qui pourrait être utile pour la prochaine version du modèle partagé. Les modèles de différences permettent aux experts, lors de la phase de synchronisation, de mener à bien la négociation en listant clairement les conflits à lever.

II. Exemple de relais

Pour l'exemple du relais, la Figure 5.10 montre un cas d'utilisation du module GAM-Diff pour identifier les différences entre le modèle du concepteur 1 et le modèle de base. Pour intégrer son point de vue, le concepteur 1 récupère en local une copie du modèle produit partagé et du modèle de contraintes partagé. Le travail de conception de l'expert 1 consiste à apporter des modifications sur ces deux modèles conformément à son expertise et à son point de vue. A la fin d'une session de travail, le module GAM-Diff fournit à l'expert 1 une comparaison via le modèle *Diff* de sa version actuelle du modèle (produit et contraintes) avec le modèle de base (produit et contraintes). Ce module offre aux différents concepteurs d'une part, des informations sur les opérations de conception en termes d'insertions, de suppressions, ou de changements de propriétés des éléments apportées par l'expert 1 à sa propre version du modèle produit, et d'autre part des informations sur les contraintes ajoutées, modifiées ou violées suite aux modifications apportées par l'expert 1. Dans ce qui suit, nous présentons quelques exemples de changements apportés au modèle du relais et en déduisant quelques éléments du modèle de différences.

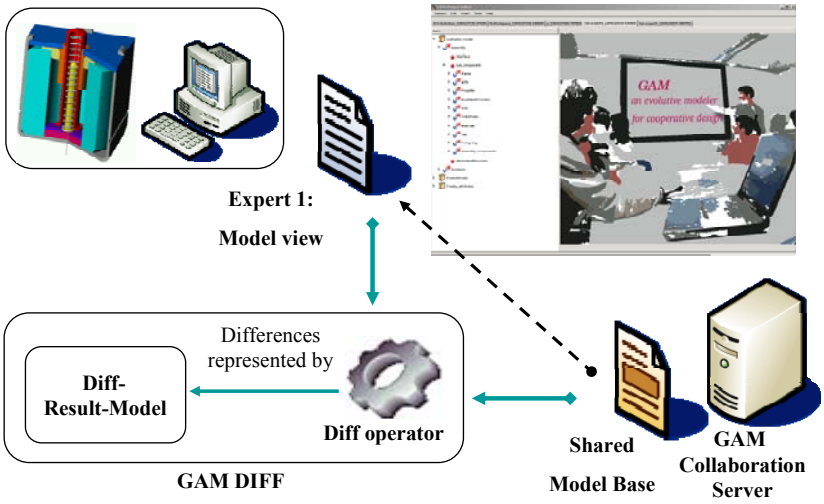


Figure 5.10 Cas d'utilisation du module de différences dans GAM

La partie (a) de la Figure 5.11 montre un exemple de visualisation des différences constatées entre le modèle version de l'expert 1 et le modèle partagé. Nous constatons 4 modifications de valeurs d'attributs, 7 opérations d'ajout et de suppression d'éléments, 2 modifications de contraintes et l'ajout d'une nouvelle contrainte.

La Figure 5.11-b présente un exemple de modification de la valeur d'un attribut de fonction $F(H_{\max})$. Sa valeur dans le modèle de base étant "14.2" (value on side 1), l'expert 1 lui affecte la valeur "22.2" (value on side 2) dans sa version locale du modèle. Cette opération a des conséquences sur deux contraintes. En effet, l'attribut $F(H_{\max})$ apparaît dans deux

contraintes. La contrainte (Constraint-1) est une contrainte de type relation entre deux attributs ("TwoValueRelation") : $F(H_{max}) > F(res)$ qui définit une relation conditionnelle entre la valeur de $F(H_{max})$ et valeur de $F(res)$. La valeur de $F(res)$ étant de "15", le résultat de vérification de cette contrainte renvoie l'état "not violated" pour signifier que la contrainte n'est pas violée. La contrainte (Constraint-2) signifie que la modification de la valeur de $F(H_{max})$ affecte la valeur d'autres attributs reliés par une contrainte de type équation mathématique.

La Figure 5.11-c montre l'exemple d'insertion du nouvel élément "fixing-part" dans le modèle de l'expert 1. Cet élément est de type composant. Le modèle de différences montre aussi la localisation exacte de l'insertion de cet élément. Il est ajouté comme sous composant d'« assembly» à la dixième place des sous-composants.

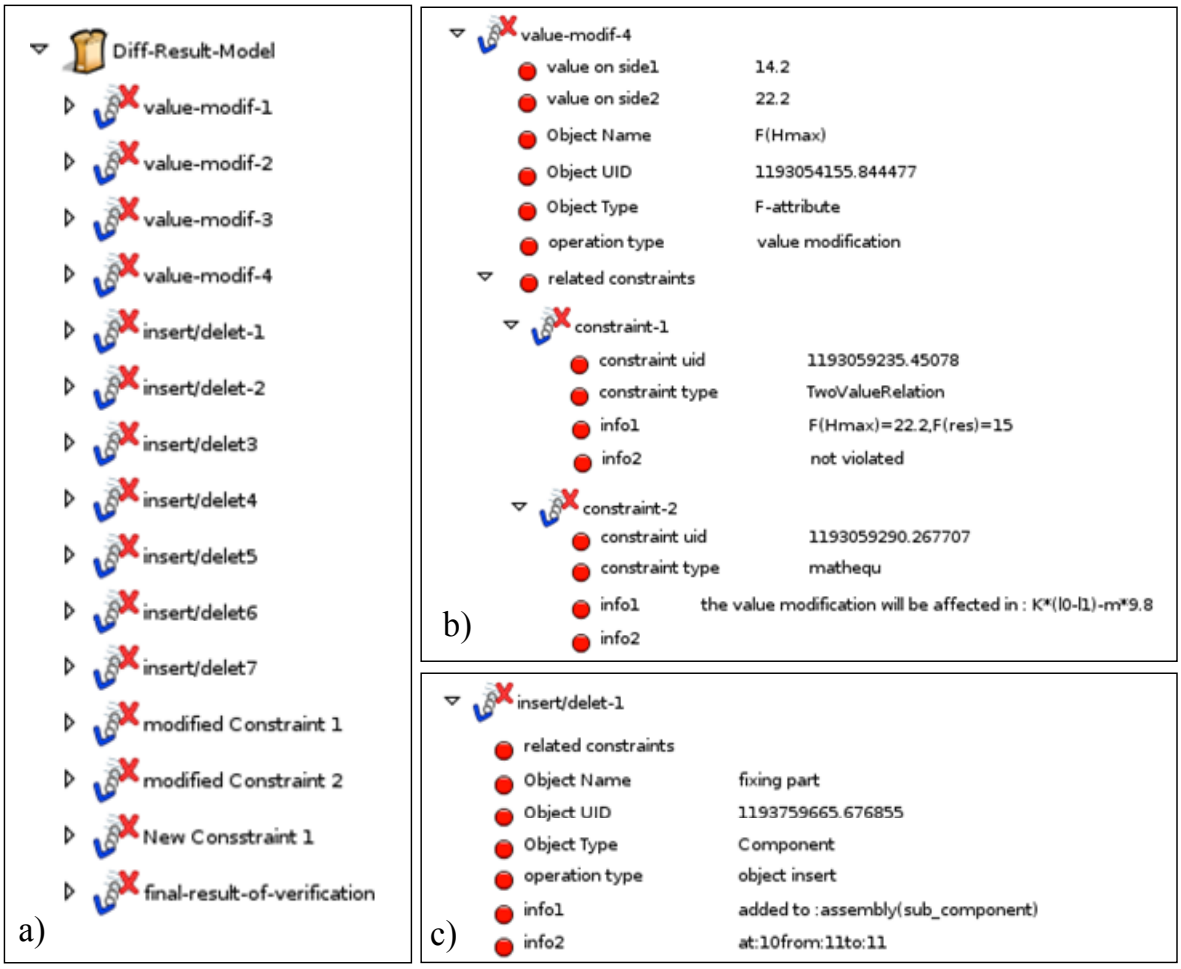


Figure 5.11 Exemple d'un modèle de différences

La Figure 5.12 illustre un autre exemple de modification de la valeur de l'attribut matériel d'un composant. Sa valeur dans le modèle de base est "value on side 1" = "steel" et la valeur choisie par l'expert 1 est "value on side 2" = "iron". La fonction $find(x)$ montre que cette modification affecte deux contraintes utilisant l'attribut *Material* dans leur définition.

La première contrainte est de type compatibilité qui définit un ensemble de couples de valeurs compatibles entre l'attribut *Material* et l'attribut h_2. La nouvelle valeur de l'attribut

Material n'étant pas compatible avec la valeur actuelle de h_2, la fonction *evaluate*(x) montre que cette contrainte est violée.

La deuxième contrainte affectée par la modification est une contrainte de type "OneValueList", qui définit une liste de valeurs possibles pour l'attribut *Material* : $Material \in \{steel, alum, iron\}$. La nouvelle valeur de l'attribut *Material* étant "iron", la fonction *evaluate*(x) retourne la valeur False qui indique que cette contrainte n'est pas violée.

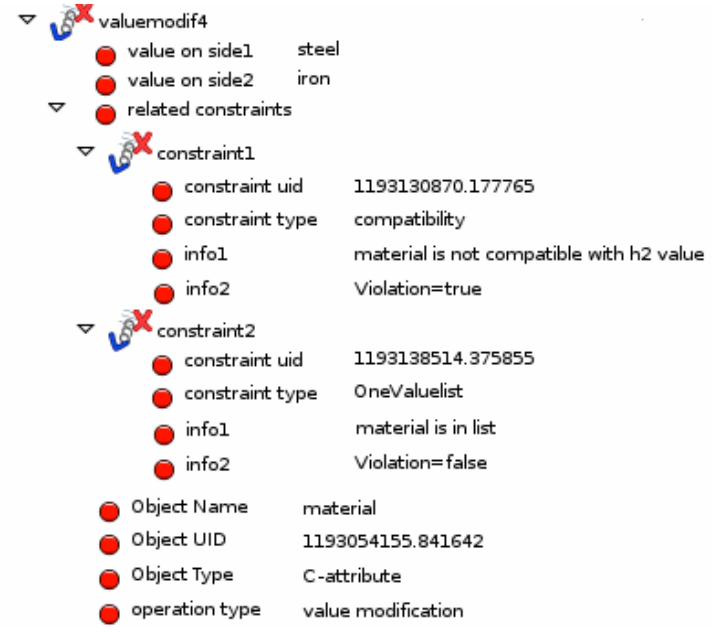


Figure 5.12 Exemple de modification de la valeur d'un attribut et ses conséquences

La Figure 5.13-a, présente un exemple de modification d'une contrainte existante. La Figure 5.13-b montre un exemple d'ajout par l'expert 1 d'une nouvelle contrainte.

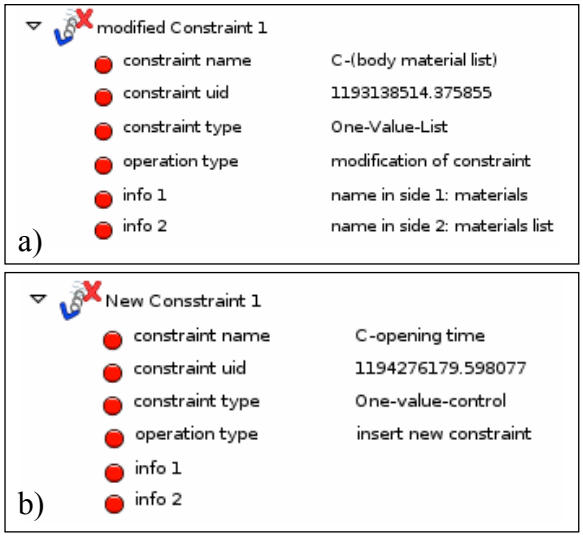


Figure 5.13 Exemples de modification et d'ajout de contraintes

La Figure 5.14 synthétise dans un tableau les données quantitatives du modèle de différences. Les modifications apportées par l'expert 1 au modèle du relais comprennent :

- Concernant le modèle produit :
  - nombre d'opérations de modification de valeurs d'attributs = 4,
  - nombre d'opérations de modification de noms d'éléments = 0,
  - nombre d'éléments ajoutés = 7,
  - nombre d'éléments supprimés = 0,
- Concernant le modèle de contraintes :
  - nombre de contraintes violées = 2,
  - nombre de contraintes ajoutées = 1,
  - nombre de contraintes modifiées = 2.

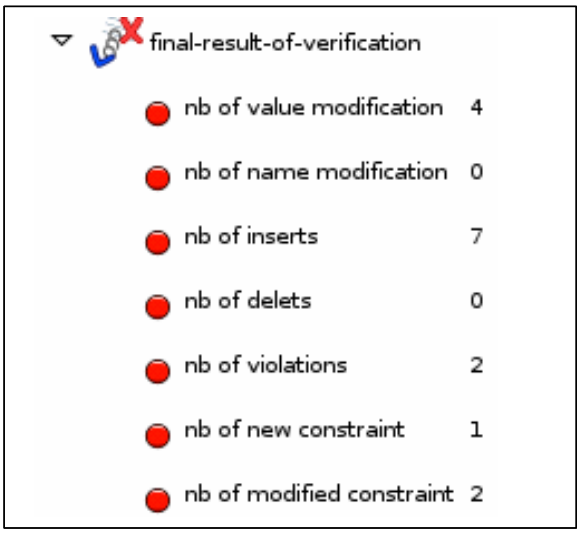


Figure 5.14 Synthèse des modifications apportées par un expert

Le module GAM-Diff fournit ce type de résultat entre le modèle de base et chaque modèle versionné. Les experts utilisent ses informations ainsi que les causes de conflits lors des sessions de synchronisation de leurs différentes versions avec le modèle partagé afin de minimiser les incohérences. Par exemple dans la Figure 5.6 les éléments et attributs créés en M_1 ne sont pas touchés par une opération en M_2 parce qu'un ID unique est défini pour chaque élément ou attribut créé. Donc, toutes les opérations effectuées en M_1 sont valides. Par contre les opérations de modification ou de suppressions d'éléments ou d'attributs sont souvent source de conflits. Un conflit se produit soit :

- lorsqu'un élément ou un attribut est supprimé en M_1, mais pas en M_2,
- lorsqu'un même attribut se voit affecté des valeurs différentes en M_1 et M_2, ou lorsqu'un changement apporté en M_1 ou en M_2 viole une contrainte.
- ou enfin lorsqu'une modification en M_1 viole une contrainte en M_2, si la modification est propagée dans M_2 (et réciproquement).

5.3.2. Contrôle de l'évolution des modèles en utilisant les modèles de différences

Les versions des experts évoluent avec le temps (le long du processus de conception) et les différences entre le modèle de base et ces différentes versions augmentent progressivement, comme illustré dans la Figure 5.15.

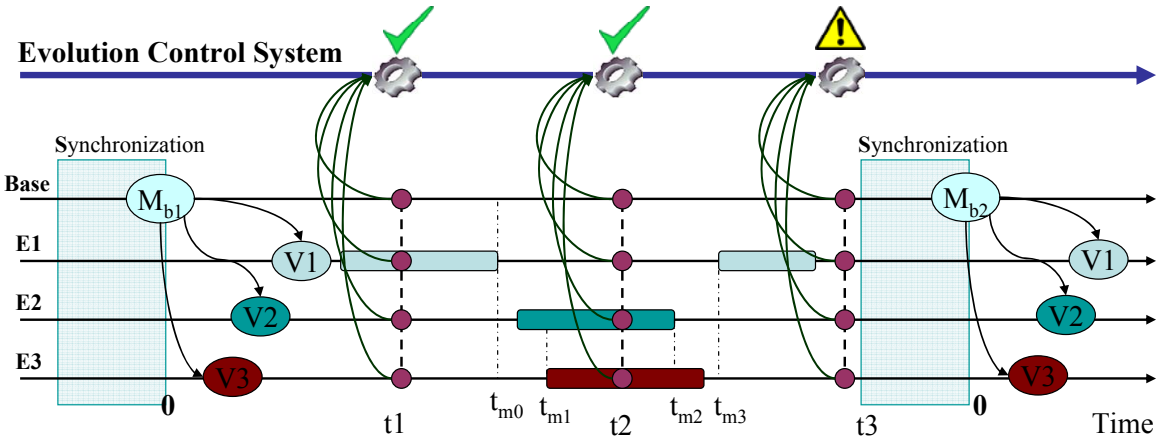


Figure 5.15 Scénario d'évolution de modèles suivant le modèle de différences

Si les experts continuent d'apporter des changements à leurs versions sans aucun mécanisme de contrôle, les différences s'accentueront, la divergence entre les versions augmentera au cours du temps et par conséquent la phase de synchronisation des versions deviendra plus difficile.

Pour rendre les sessions de synchronisation plus utiles et plus faciles à conduire, il est nécessaire d'intégrer au processus de conception collaborative un système qui permet de :

- contrôler de façon périodique $(t_1, t_2, t_3 ...)$ les évolutions des différentes versions. Si lors d'un contrôle le nombre de changements et d'incohérences devient important, le système peut notifier aux experts la nécessité de planifier une session de synchronisation. Les différences de toutes les versions doivent être mesurées et agrégées afin d'obtenir un degré de divergence de l'ensemble des versions par rapport au modèle partagé. Les sessions de synchronisation sont déclenchées lorsque ce degré de divergence dépasse une certaine limite à définir. L'objectif principal serait donc d'avoir un système de contrôle du processus de conception révélant ce qui se passe réellement lors du travail des concepteurs, et identifiant quand les concepteurs doivent synchroniser leurs versions,
- faciliter la phase de synchronisation des versions avec le modèle partagé en utilisant, d'une part un mécanisme de négociation des choix effectués par les acteurs métiers, et d'autre part un mécanisme de propagation de contraintes pour la résolution des conflits. Dans certains contextes, pour aboutir à des solutions réalistes de conception il est parfois possible de violer certaines contraintes. Il est alors nécessaire d'exprimer ce type d'information lors de la synchronisation des versions.

Au final, l'objectif d'intégration de ce système est de répondre à deux questions :

- quand synchroniser ? et
- comment synchroniser ?

5.3.2.1. Modélisation pour le contrôle de l'évolution des modèles

La Figure 5.16 explicite un scénario de conception collaborative avec deux experts. Pour travailler en mode local, chaque expert récupère une copie du modèle produit et du modèle de contraintes associé. Le mécanisme de contrôle calcule, à certaines dates (t_1, t_2, t_3), la divergence Δ_1 de la version 1 par rapport au modèle partagé et la divergence Δ_2 de la version 2 par rapport au même modèle partagé. Une divergence agrégée Δ_{final} est ensuite calculée sur la base des deux divergences Δ_1 et Δ_2. La divergence totale Δ_{final} et d'autant plus critique qu'elle se rapproche d'une divergence limite Δ_{limit} prédéfinie. La notion de divergence ou de dissimilarité correspond aux différences donc à la distance entre deux modèles.

Pour gérer cela, nous introduisons le concept de "méta-règles" qui définit les conditions sur les évolutions des versions en analysant l'impact des modifications parallèles sur le modèle de base. En d'autres termes, le système contient des "méta-règles" indépendantes du domaine et qui modélisent de manière explicite la stratégie de déclanchement d'une synchronisation. Elles sont chargées de contrôler l'évolution des modèles des experts. Nous mettons en place:

- un observateur des historiques et des évolutions des versions,
- un système de calcul de la différence de chaque modèle métier par rapport au modèle commun (méta-règle),
- des structures de contrôle utilisant des méta-règles, et
- des actions à déclencher (notification pour session de synchronisation) lors de la violation des méta-règles.

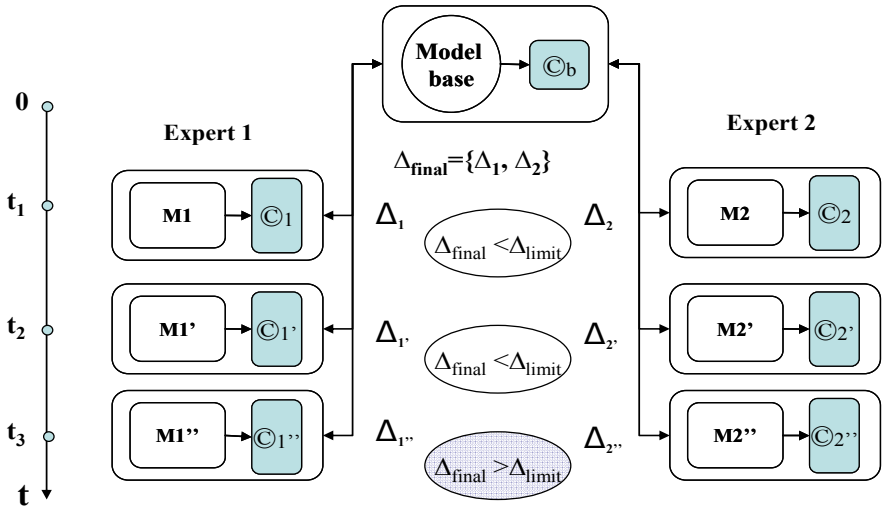


Figure 5.16 Evolution du processus de conception collaborative avec deux experts

Le concept de "méta-contrainte" caractérise les relations ou les conditions entre les contraintes métiers permettant d'obtenir des solutions réalistes lors de la phase de la synchronisation.

Nous devons donc proposer :

- un cadre conceptuel pour les méta-règles afin de décrire, de concevoir et de faire évoluer des modèles et d'avoir ensuite un système de contrôle dynamique et automatique,
- un cadre conceptuel pour les méta-contraintes afin de décrire les stratégies de négociation et de propagation des contraints et d'avoir ensuite un système de synchronisation.

Ces cadres ont formalisés, déployées et testées encore une fois au sein de l'environnement GAM.

I. Méta-règles

La Figure 5.17 montre un exemple de scénario d'implémentation de méta-règles dans l'environnement GAM. Deux experts 1 et 2 travaillent de manière indépendante et asynchrone sur leurs versions du modèle produit. À certaines dates, le système de contrôle observe le statut des différences Δ_{final} entre les versions des deux experts et le modèle partagé en définissant la fonction suivante :

$$\Delta_{final} = \delta_{syntaxique} + \delta_c$$

$\delta_{syntaxique}$ est une fonction d'évaluation des différences syntaxiques entre les versions des experts et le modèle de base. La fonction δ_c est une fonction de vérification des contraintes métiers correspondants aux modifications effectuées. Les méta-règles définissent les limites et les conditions sur le Δ_{final}. Si le système de contrôle détecte des cas de violation de la méta-règle ($\Delta_{final} > \Delta_{limit}$), un message de notification doit être envoyé à tous les experts pour organiser une session de synchronisation.

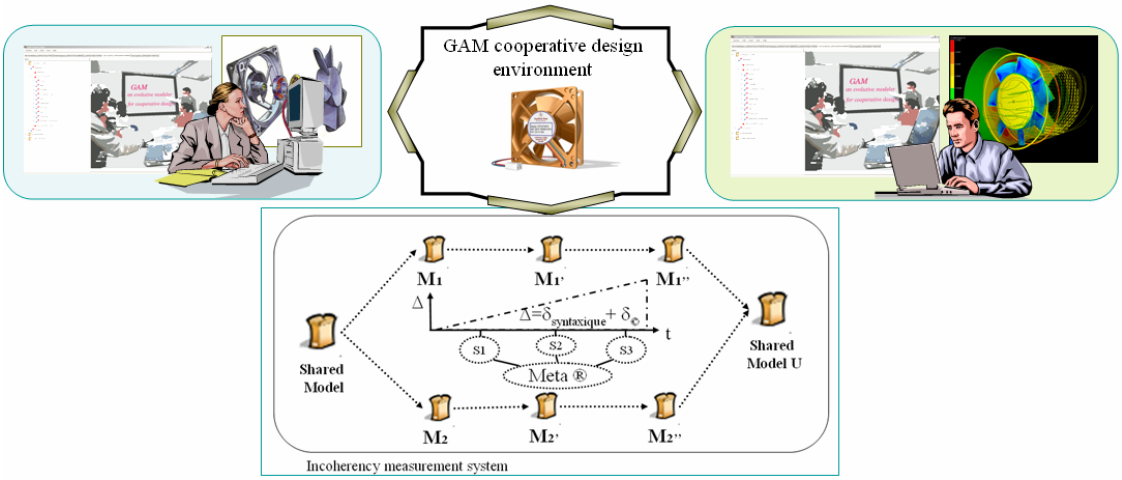


Figure 5.17 Scénario d'implémentation de méta-règles dans GAM.

Dans ce contexte, les méta-règles définissent les points clés qui permettent de gérer le fonctionnement de l'environnement coopératif en lui conférant une plus grande autonomie par récupération du travail des différents experts et synthèse des résultats. La structuration de cet espace d'autonomie passe par la définition des règles communes aux experts ainsi que les méthodologies employées pour l'analyse, la supervision et l'apprentissage dans des environnements de conception évolutifs et/ou distribués.

La Figure 5.18, représente la structure du système de contrôle. Ce système permet aux acteurs de définir le modèle de méta-règles pour évaluer les effets de l'ensemble des modèles de différences.

En effet, les méta-règles sont des règles particulières qui permettent de gérer le flux dynamique des évolutions parallèles des versions en effectuant une analyse "temps réel". Les méta-règles peuvent être classées en plusieurs groupes qui définissent : le coût de modification d'un élément ou de violation d'une contrainte, la durée d'évolution des modèles, les conditions sur la violation de certaines règles métiers, les stratégies de choix de règles mais aussi pour définir l'ordre d'activation des autres règles.

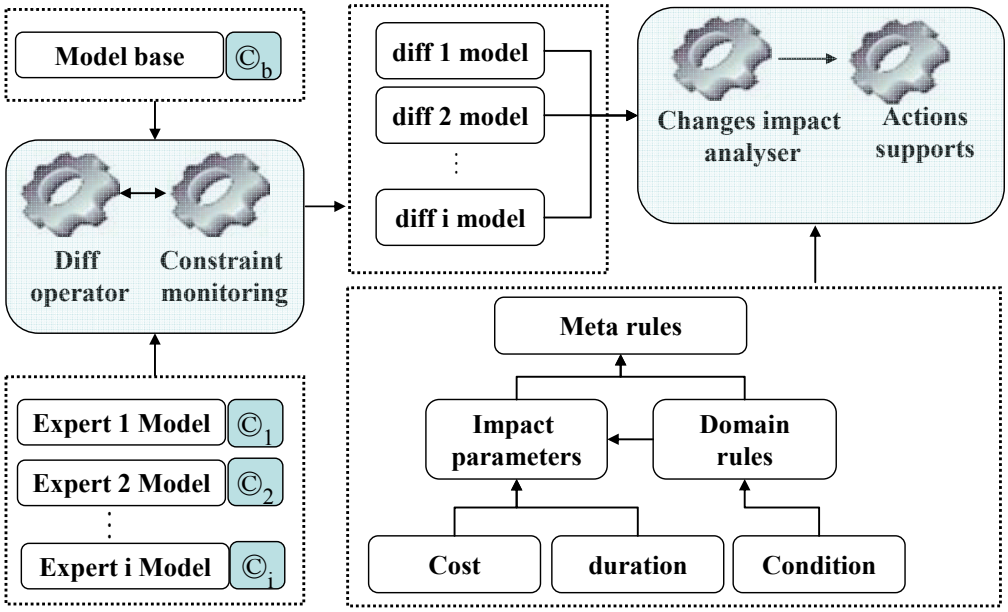


Figure 5.18 Structure du système de contrôle des modifications

Nous définissons la fonction d'analyse des effets de modification comme :

$$\Delta_{final} = f_1(Diff_i) + f_2(C_i | Diff_i)$$

$$f_{contrôle} = f_{M-R}(\Delta_{final})$$

Les fonctions f_1, f_2 et $f_{contrôle}$ se présentent de la manière suivante :

La fonction f_1

Cette fonction évalue le coût des modifications totales pour des différences syntaxiques en utilisant les résultats des modèles de différences. Comme nous l'avons expliqué précédemment, les différents changements apportées par chaque expert sont classés en plusieurs catégories : le nombre de modifications de valeurs d'attributs, le nombre d'éléments supprimées, ajoutés, etc (Figure 5.19). Si nous posons l'hypothèse que le coût de changements dépend du type de changement et de l'expert, alors nous pouvons affecté un facteur de poids λ_j (j étant la catégorie de changement) prenant en compte l'importance de chaque catégorie de modification et un autre facteur de poids μ_i (i étant l'expert) prenant en compte l'importance de chaque expert dans l'évaluation finale. Le paramètre nb_{ji} définit le nombre de modification de type j apportées par l'expert i.

La fonction f_1 est une fonction de λ_j, nb_{ji} et de μ_i définie par les experts pour évaluer le coût total de leurs modifications :

$$f_1 = F(\lambda_j, nb_{ji}, \mu_i)$$

Par exemple, la fonction d'agrégation F pourrait s'exprimée sous la forme d'une moyenne pondérée :

$$f_1 = F(\lambda_j, nb_{ji}, \mu_i) = \sum_i \left(\mu_i \sum_j \left(\lambda_j . nb_{ji} \right) \right)$$

Avec :

$\Delta_i = \sum_j \left(\lambda_j . nb_{ji} \right)$, le coût total de toutes les modifications apportées par l'expert i.

$\delta_j = \sum_i \left(\mu_i . nb_{ji} \right)$, le coût total des modifications de catégorie j apportées par tous les experts.

		μ1	μ2	…	μi	
	Operation on model/ experts	Expert 1	Expert 2	…	Expert n	
λ1	nb of value modification	4	3		2	δ1
λ2	nb of name modification	0	1		3	δ2
λ3	nb of inserted items	7	0		4	δ3
λ4	nb of deleted items	1	3		1	δ3
λ5	nb of violated constraint	2	5		3	δ4
λ6	nb of new constraint	1	0		2	δ5
λ7	nb of modified constraint	2	0		1	δ7
		Δ1	Δ2	…	Δn	

Figure 5.19 Synthèse des modifications apportées par tous les experts

La fonction f_2

C'est une fonction qui regroupe trois catégories de règles :

- **Les règles d'évocation de contexte** : sont les règles suivant les actions sur le modèle de produit ou le modèle de contraintes. Elles peuvent donner une orientation dynamique à l'exécution des actions prédéfinies. Par exemple, si les contraintes C_1, C_2 et C_5 sont violées alors vérifier la contrainte C_6.
- **Les règles de relation d'ordre :** sont les règles agissants sur des catégories de règles et permettant d'utiliser certain règles avant d'autres, en fonction de la situation, et de préciser la manière d'utiliser ces règles. Une telle méta-règle s'écrirait par exemple : SI "règle R_1 a été appliquée ET SI "règle R_2 a été appliquée" ALORS "conséquence". Ces méta-règles spécifient l'ordre dans lequel peuvent être activées certaines règles, notamment quand celles-ci arrivent aux mêmes conclusions. Exemple : SI les règles R_1

ET R_2 ont comme actions $A_a, A_b, ..., A_i$ ALORS activer en priorité les règles qui ont dans leurs prémisses $A_a, A_b, ..., A_i$.

- **Les règles de désactivation** : sont les règle agissants sur certaines règles et permettant de désactiver certaines autres règles en fonction de la situation, par exemple : SI la contrainte C_1 est violée ALORS désactiver la contrainte C_3.
- **Les règles d'action** : sont des règles qui permettent d'exécuter certaines actions en fonction de la situation, par exemple : SI la contrainte C_1 est violée ALORS envoyer un message de notification.

La fonction $f_{contrôle}$

La fonction $f_{contrôle}$ est un ensemble de méta-règles f_{M-R} qui définissent les conditions sur l'évaluation globale Δ_{final} (f_1 et f_2). La Figure 5.20 montre quelques exemples de méta-règles qui sont considérées pour le contrôle de l'évaluation des modèles. La règle "Rule1" contient une limite qui contrôle la valeur de la fonction f_1, si la valeur de la fonction f_1 dépasse le seuil 25 alors envoyer un message de notification. La règle "Rule2" est une méta-règle sur l'état de la contrainte de "c-air-gap" qui signifie dans cette exemple l'importance de cette contrainte (contraint dure), les experts doivent toujours respecter cette contrainte. En cas de violation de cette contrainte, le système de contrôle envoie un message de notification aux experts. Ce type de méta-règle contrôle l'évolution des modèles. S'il détecte une violation de méta-règle, il notifiera les experts, selon un mode spécifique.

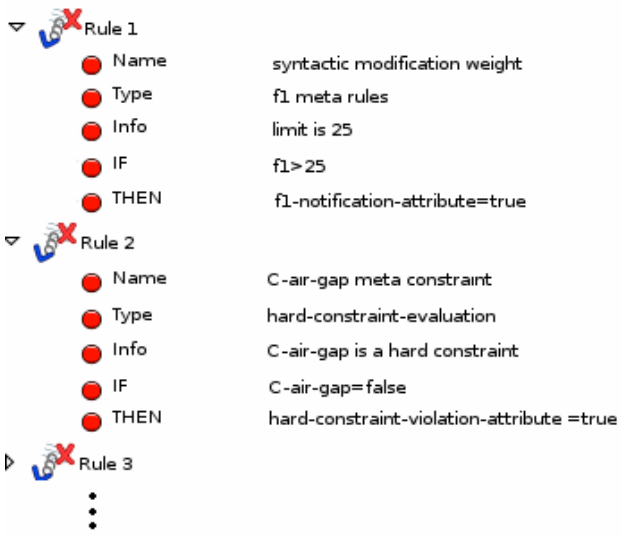


Figure 5.20 Exemples de méta-règles de contrôle d'évolutions de modèles

Avec la définition des différentes catégories de méta-règles, le système de contrôle fournit les méthodes de vérification des évolutions des modèles. La Figure 5.21 montre le concept de système de contrôle en utilisant les méta-règles définies précédemment. Ce système vérifie ces méta-règles à certaines dates. Si une méta-règle est violée, alors le système affichera un message de notification à tous les experts participant à la conception coopérative. Ce message de notification permet aux experts de planifier une session de synchronisation. En principe, en utilisant ce système de contrôle dans un projet de conception coopérative, les concepteurs peuvent avoir deux types de revues de projet (synchronisation) (Figure 5.22) :

- la revue de projet R_i fixée et planifiée à priori, qui permet la synchronisation des modèles des experts à des périodes de temps prédéfinies pour toute la période du projet,
- la revue de projet T_i notifiée et non planifiée, qui pourrait avoir lieu entre deux revues de projet planifiées. Cette revue pourrait être fixée si, après vérification du statut des méta-règles à certaines dates intermédiaires (dans notre cas, la périodicité de vérification des méta-règles est fixée), le système de contrôle détecte la violation d'une méta-règle. Dans ce cas, le système envoie un message de notification pour organiser la session de synchronisation, donc une revue de projet. Bettaieb [Bet06] appelle ce mécanisme, une notification anticipée. Notre travail a permis sa mise en œuvre effective. En effet, le système de contrôle informe les concepteurs de l'importance de la divergence entre leurs différentes versions du modèle et suggère une revue de projet avant même la revue planifiée R_i.

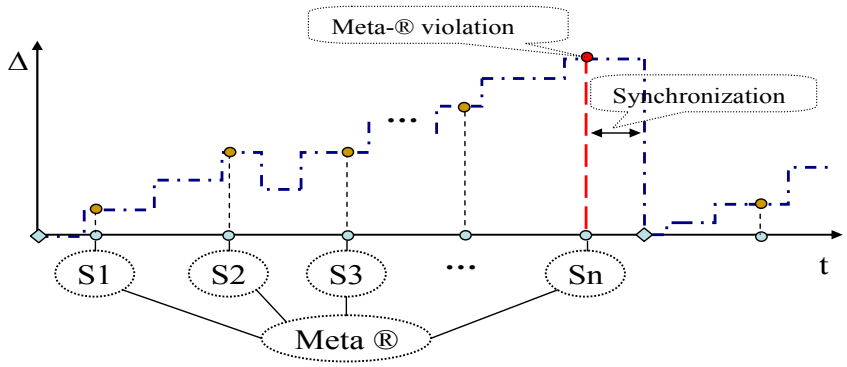
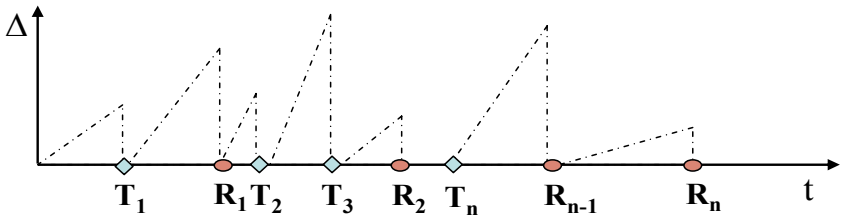


Figure 5.21 Système de contrôle d'évolution de modèles



R: fixed time for models review during the life of the project
T: suggested time for models review (meta-rule violation)

Figure 5.22 Revues de projet pour la synchronisation de modèles

II. Méta-contraintes

Comme pour les méta-règles, les méta-contraintes sont des contraintes qui définissent des relations portant sur les différentes contraintes métiers définies par les experts. Elles permettent de gérer et guider l'étape de propagation des contraintes métiers afin d'aboutir à des solutions de conception acceptables et réalisables. Le contrôle du processus de propagation des contraintes permet de générer tous les chemins possibles menant à une solution. Ces méta-contraintes représentent les connaissances des experts sur les stratégies de propagation des contraintes, donc de recherche de solution. Les méta-contraintes permettent alors de déterminer (Figure 5.23) :

- **les dépendances entre les contraintes** : ce type de méta-contrainte se présente par la définition des règles de dépendances des contraintes métiers. Si certaines contraintes métiers sont violées alors d'autres doivent impérativement être satisfaites, ou bien de nouvelles contraintes qui n'appartenaient pas au modèle initial doivent être activés.
- **les priorités entre les contraintes métiers** : ce type de méta-contrainte détermine les priorités et les importances des contraintes métiers lors de la résolution d'un problème,
- **les degrés de violation des contraintes** : ce type de méta-contrainte se présente par la définition de règles sur les niveaux de violation des contraintes métiers, sachant que certaines contraintes peuvent être plus ou moins fortement violées,

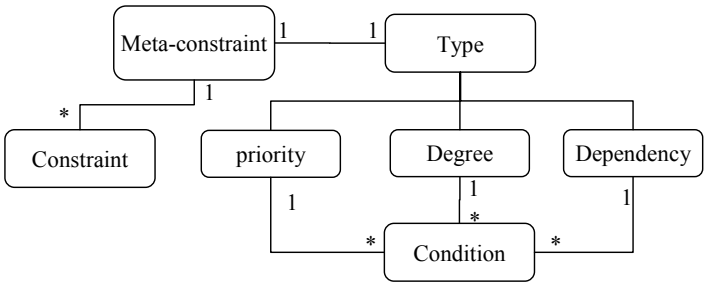


Figure 5.23 Méta-modèle de contraintes associé au modèle produit

5.3.2.2. Expérimentation des résultats dans la salle MEXICO

Du point de vue modélisation, nous avons mis en évidence plusieurs méta-règles et méta-contraintes pour aider les concepteurs dans leur travail collaboratif. Les expérimentations menées avec des experts nous apportera, sans doutes, des éléments supplémentaires pour compléter les outils d'assistance au processus de conception collaborative.

La Figure 5.25, montre la salle MEXICO (Moyen EXpérimentaux pour l'Intégration en COnception) installée au Laboratoire G-SCOP. Elle contient des équipements capables d'enregistrer un scénario de travail coopératif. Nous utilisons cette salle pour expérimenter un travail de conception coopérative entre plusieurs experts en utilisant l'environnement GAM. Nous disposons dans la salle MEXICO un serveur capable de supporter l'infrastructure du système d'information partagé via GAM (Figure 5.24).

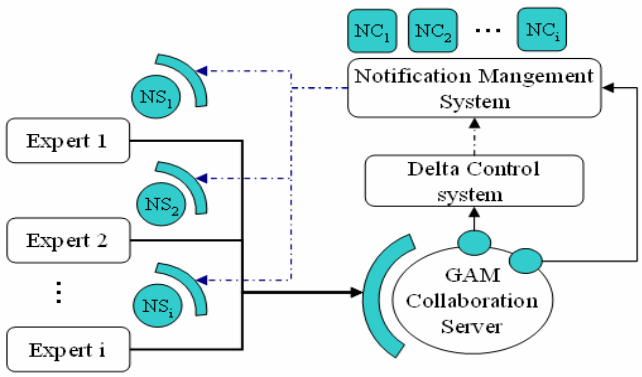


Figure 5.24 Système de contrôle et de notification intégré à l'infrastructure GAM

Toutes les données sont centralisées sur ce seul serveur (GAM Collaboration Server), ce qui simplifie les contrôles de sécurité et la mise à jour des données. Il est capable de servir plusieurs experts simultanément.

Le serveur "GAM Collaboration Server" est composé de trois parties :

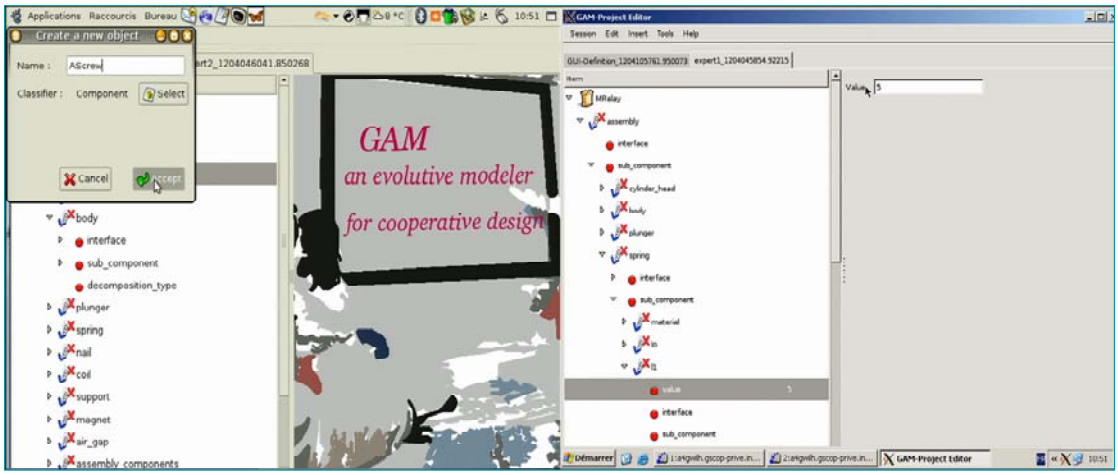
- l'application GAM qui peut être appelée par les experts à travers le réseau,
- le système de contrôle des différences (Delta Control System) qui contrôle l'évolution parallèle des versions des experts,
- le gestionnaire de notification (Notification Management System) qui gère les connexions à tous les postes de travail des experts et assure la transmission immédiate de toutes les alertes. Il est connecté à plusieurs serveurs de notifications (NS_i) qui sont associés aux experts. Ces serveurs permettent de distribuer une notification aux experts en réponse à des déclencheurs spécifiques associés au compte de chaque expert en fonction de leurs préférences. Chaque serveur de notification est à l'écoute, prêt à répondre à la notification envoyée par le gestionnaire de notification, dès qu'une notification lui parvient, il la traite et envoie une réponse de confirmation.

Des postes clients sont installés dans la salle MEXICO pour permettre la connexion des experts au système d'information (Figure 5.25-a & Figure 5.25-b). La salle est aussi équipée d'un appareil d'acquisition vidéo des postes de travail collaboratif (Figure 5.25-c & Figure 5.25-d). Cet appareil est un mixeur vidéo 4 canaux qui a pour objectif d'enregistrer les sessions de travail des postes clients et de les juxtaposer sur une même séquence vidéo. Il possède une sortie quad qui offre les quatre sources d'entrée (vidéo/audio) sur un seul et unique moniteur de sortie.

Figure 5.25 Expérimentation des résultats dans la salle MEXICO

Nous avons construit un scénario qui illustre, via l'exemple du Relais, l'intérêt d'utiliser le système de contrôle des évolutions dans un projet de conception coopérative. Le scénario se déroule comme suit :

- Expert 1 et Expert 2 se connectent au serveur de collaboration GAM. Chaque expert récupère une copie du modèle de base pour continuer de travailler sur le modèle produit du Relais. En utilisant les équipements de l'enregistrement vidéo, nous arrivons à capturer les travaux parallèles des deux experts sur un écran pour voir comment évoluent leurs modèles (Figure 5.26).
- Le système de contrôle des évolutions, contrôle à certaines périodes de temps les changements réalisés par chaque expert ainsi que leurs impacts. Il génère les modèles des différences et vérifie ensuite les contraintes et les méta-règles associées aux modèles. Après un certain temps de travail des experts, lors du contrôle des modèles, le système s'aperçoit de la violation d'une méta-règle qui définit la condition limite sur les différences syntaxiques et métiers.



Expert 1 Expert 2

Figure 5.26 Captures d'écran des travaux parallèles de deux experts

- Le gestionnaire de notification se charge de notifier les deux experts qu'ils doivent planifier une session de synchronisation et de négociation (Figure 5.27). Chaque expert reçoit un message de notification sur son écran de travail.

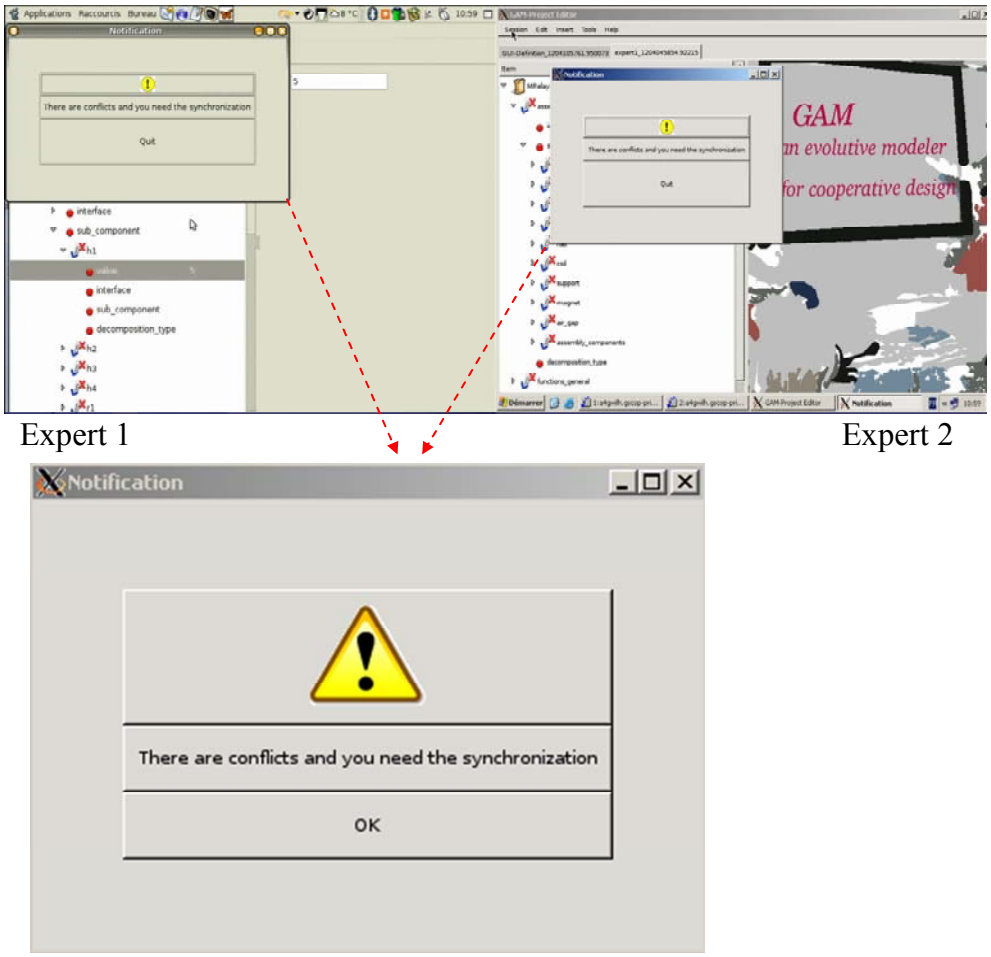


Figure 5.27 Système de notification pour une session de synchronisation et de négociation

– Comme illustrer dans la Figure 5.28, les deux experts peuvent utiliser les résultats des modèles de différences en parallèles lors de la session de négociation et de synchronisation.

Figure 5.28 Utilisation des modèles de différences lors de la négociation

Nous avons donc formalisé et structuré ces étapes qui demeurent encore essentiellement informelles.

5.4. Conclusion

Dans ce chapitre, nous avons décrit le problème de la cohérence entre les différents modèles des experts qui partagent un modèle de base. Le concept de différences *Diff* est présenté comme une technique qui vise à représenter les différences entre les versions et le modèle partagé, ce qui permet aux experts de prendre connaissance des modifications réalisées par leurs collègues. Afin d'arriver à piloter les moments opportuns pour une fusion des différentes évolutions envisagées, nous avons proposé un système de contrôle d'évolution qui contrôle les évolutions des différentes versions en vérifiant les impacts des modifications de chaque évolution.

Nous avons défini des mécanismes de mesure du cumul des impacts permettant de décider le moment opportun d'une synchronisation des modèles. Ces mécanismes sont représentés par la définition des méta-règles qui décrivent les conditions et les limites sur l'évolution des modèles.

Conclusion générale

Dans ce document nous avons traité plusieurs problématiques liées à la cohérence des modèles experts dans un contexte de conception coopérative de produits manufacturiers. Dès lors qu'au moins deux concepteurs coopèrent à un niveau technique de la conception, ils partagent de nombreux objets, paramètres, informations ou morceaux de modèles. Cependant ils ne peuvent travailler toujours simultanément et des phases de travail asynchrones sont indispensables. L'objectif de notre travail est de spécifier les mécanismes nécessaires pour mettre en œuvre des règles métier assurant la cohérence technique entre concepteurs. En imaginant un serveur espionnant l'évolution des modèles de chaque concepteur. Nous avons introduit des modèles de règles métiers et des mécanismes qui permettent d'évaluer si deux études parallèles divergent au delà de la simple comparaison syntaxique des modèles. Cette mesure de la divergence basée sur la notion de méta-règle est alors utilisée pour notifier en temps voulu les acteurs de la nécessité de se synchroniser. Nous avons proposé des solutions originales pour assister des activités de conception coopérative. Elles sont basées sur l'utilisation d'un modèle de partage qui est simultanément versionné par les différents collaborateurs. Ces versions sont analysées par des mécanismes syntaxiques et à base de contraintes métiers. Au delà de cette proposition théorique, nous avons développé un démonstrateur qui valide une part des propositions avancées.

Ce travail ouvre la voie à la concrétisation d'un véritable environnement coopératif de la conception. Pour atteindre cet objectif, nous envisageons les perspectives à moyen terme suivantes :

- ***extension de l'approche de modélisation de contraintes sur le modèle de processus et le modèle d'organisations proposé dans le modèle PPO :*** le chapitre 4 propose un modèle de contrainte à intégrer au modèle produit. Néanmoins la définition et l'intégration des modèles de contraintes au modèle de processus et modèle d'organisations est une extension logique de notre travail. La difficulté principale sera de vérifier si les types de contraintes envisagées au chapitre 4, sont suffisants pour gérer la cohérence dans ces domaines spécifiques,
- ***définition d'un service de synchronisation :*** le processus de synchronisation requiert d'abord le calcul de différences entre deux versions permettant l'extraction des modifications puis l'intégration des différences dans le modèle partagé. Le chapitre 5 ébauche les méthodes d'extractions des différences et les caractéristiques d'un mécanisme de contrôle de l'évolution pour définir les points de synchronisation. La formalisation de l'étape d'intégration des différences est maintenant requise pour finaliser l'assistance à la synchronisation,
- ***intégration d'un moteur de propagation des contraintes*** ce moteur peut servir à analyser l'espace des solutions d'un problème de conception. Il permettra aux concepteurs, la génération de solutions parmi l'ensemble des choix possibles lors de la phase de synchronisation et de négociation. Il ne s'agit pas de créer un nouvel outil de propagation de contraintes mais d'inter-opérer avec des outils existants,
- ***formalisation de la notion de méta-contraintes :*** La formalisation de méta-contraintes permet de gérer et guider l'étape de propagation des contraintes afin d'aboutir à des solutions de conception acceptables et réalisables. Un modeleur de méta-contraintes est requis et devrait être spécifié sur la base de ce que nous avons initialiser dans ce document,

- ***optimisation des performances des modules proposés et le passage à grande échelle :*** des optimisations peuvent être apportées à notre prototype, notamment en ce qui concerne l'amélioration de certaines modules comme GAM-DIFF (afin de générer les éléments du méta-modèle de différences de manière automatique en se basant sur le méta- modèle des deux modèles à comparer) et GAM-Constraint (pour ce qui concerne la définition des règles métiers et du moteur de raisonnement). Ceci permettra d'améliorer l'efficacité de GAM et de travailler sur des problèmes de taille industrielle. De même l'intégration de ce prototype au sein d'une infrastructure du marché permettra le passage à l'échelle en effectuant des tests grandeur réelle. Il serait intéressant d'intégrer cet outil et les méthodes proposées au niveau industriel dans un outil PDM ou PLM afin notamment de faciliter la coopération entre les différents acteurs et la mise à jour des informations relatives aux produits aux processus et aux ressources.

Au-delà de ces perspectives qui tendent à finaliser la démarche entreprise, nous nous devons de rappeler que la démarche introduite prend en compte les différents situations de distribution dans l'espace et dans le temps telles que décrites par la matrice de Johansen [Joh88] dans un cadre conceptuel qui s'est volontairement limité au cadre du modèle PPO issu d'IPPOP, dans un cadre fonctionnel plutôt associé à des espaces de coopération, avec un cadre organisationnel assez statique. De ce contexte il sera logique de viser des perspectives qui s'affranchissent de ces limites en développant des axes de recherches nouveaux comme par exemple :

- la prise en compte de contextes organisationnels incertains et fortement évolutifs,
- une meilleure interaction avec les espaces métiers (problèmes d'interopérabilité dans des modèles hétérogènes),
- une formalisation des espaces de communication et de coordination qui demeurent encore fortement informels,
- Une ouverture à d'autres cadres conceptuels par exemple en tentant l'expérience avec STEP comme modèle partagé.

Au final, nous espérons que ces travaux contribuent à une pratique effective des méthodes de conception intégrée.

Bibliographie

[Bad00] Badham M., Couchman R., Implementing concurrent engineering, *Human Factors and Ergonomics in Manufacturing*, Vol.10 (3), pp.237-249, 2000.

[Bel94] Belloy P., Intégration de connaissances métiers dans la conception : un modèle pour les pièces mécaniques, application à l'usinage et à l'estampage, *Thèse de doctorat, Université Joseph Fourier, Grenoble*, 1994.

[Ben01] Bensana E., A constraint based environment to solve configuration problems: application to HALE UAV design, *CEAS Conference on Multidisciplinary Aircraft Design and Optimization*, Cologne, Germany, 25-26 June 2001.

[Bet06] Bettaieb S., Contribution à la spécification d'un environnement de conception collaborative intégrateur d'expertises métier hétérogènes, Thèse de doctorat de l'Institut Polytechnique de Grenoble, 2006.

[Bet08] Bettaieb S., Noël F., A generic architecture to synchronise design models issued from heterogeneous business tools: towards more interoperability between design expertises, *International Journal of engineering with computers*, Vol.24 (1), 2008.

[Bou94] Bourdichon P., L'ingénierie simultanée et la gestion d'informations, *Ed. Hermès Sciences, Collection : Techniques de l'information*, 1994.

[Bre86] Breitbart Y., Olson P., Database integration in a distributed heterogeneous database system, *In Proc. of the Second IEEE Conference on Data Engineering*, 1986.

[Bri01] Brissaud D., Tichkiewitch S., Product models for life-cycle, *Annals of the CIRP*, Vol.50 (1), pp105-108, 2001.

[Bur04] Burmester S., Giese H., Tool integration at the meta-model level: the Fujaba approach, *International Journal on Software Tools for Technology Transfer (STTT)*, Vol.6 (3), pp 203 – 218, 2004.

[Car99] Carballo J. A., Director S. W., Constraint Management for Collaborative Electronic Design, *Proceedings of the 36th ACM/IEEE conference on Design automation*, Louisiana, United States, 1999

[Cas00] Castelfranchi C., Conflict Ontology, book chapter of *Computational Conflicts, Conflict Modelling for Distributed Intelligent Systems*, Ed. Springer, pp. 21-40, 2000

[Cer03] Ceroni J. A., Velasquez A. A., Conflict detection and resolution in distributed design, *Journal of Production Planning & Control*, Vol. 14 (8), pp.734–742, 2003

[Cha97] Chapa E., Outils et structure pour la coopération formelle et informelle dans un contexte de conception holonique, *Thèse doctorat, Institut National Polytechnique de Grenoble, 1997.*

[Che02] Chen L., Lin L., Optimization of product configuration design using functional requirements and constraints, *Research in Engineering Design*, Vol.13, pp. 167–182, 2002

[Che07] Chettaoui H., Noel F., An environment for collaborative design: a new approach to CAD tool interoperability, *Proceedings of the International Conference on Product Lifecycle Management*, 2007

[Cho05] Chomicki J., Marcinkowski J., On the Computational Complexity of Minimal-Change Integrity Maintenance in Relational Databases, Book Chapter of

Inconsistency Tolerance, Lecture Notes in Computer Science Book Series, Vol.3300, , pp.119-150, 2005

[Cho96] Chong E.K.P., Zak S.H., An introduction to Optimization, *Ed. John Wiley & Sons*, 1996.

[Com99] Comon H., Dincbas M., A Methodological View of Constraint Solving, *International journal of Constraints*, Vol.4 (4), pp.337-361, 1999.

[Con98] Cointe C., Aide à la gestion de conflits en conception concourante dans un système distribué, *Thèse de Doctorat Sciences Spécialité Informatique. Montpellier II : Université des Sciences et Techniques du Languedoc*, 1998.

[Cou99] Coutel C., Contribution méthodologique à la conception sous contraintes de dispositifs électromagnétiques", *Thèse de doctorat, Institut national polytechnique de Grenoble, Grenoble,* 1999.

[Dou99] Doucet A., Fauvet M.-C., Using database versions to implement temporal integrity constraints, Book Chapter of *Constraint Databases and Applications, Lecture Notes in Computer Science Book Series*, pp.219-233, 1999.

[Elm99] Elmagarmid A., Rusinkiewicz M., Management of Heterogeneous and Autonomous Database Systems, *Ed. Morgan Kaufmann*, 1999.

[Eng01] Engeström Y., Expansive Learning at Work: toward an activity theoretical re-conceptualization, *Journal of Education and Work*, Vol.14 (1), pp.133-156, 2001

[Fav06] Favre J., Estublier J., L'ingénierie dirigée par les modèles, *Ed. Lavoisier, Hermès-science*, 2006.

[Fen03] Fenves S.J., Choi Y., Master product model for the support of tighter design-analysis integration, *National Institute of Standards and Technology, Gaithersburg, MD, 20899, NISTIR 7004*, p. 7004, 2003.

[Fis00] Fischer X., Stratégie de conduite du calcul pour l'aide à la décision en conception mécanique intégrée ; application aux appareils à pression, *Thèse de doctorat, Ecole nationale supérieure d'arts et métiers*, 2000.

[Fle06a] Flener P., Pearson J., Static and Dynamic Structural Symmetry Breaking, Book chapter of *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science Book Series*, pp.695-699, 2006.

[Fle06b] Fleurey F., Langage et méthode pour une ingénierie des modèles fiable, *Thèse de doctorat, Université de Rennes 1, Équipe Triskell – IRISA,* 2006.

[Foi99] Foisel R., Drogoul A., Des écosystèmes artificiels d'aide à la conception : l'exemple du projet CAROSSE, *Rapport de Recherche, n° 029, LIP6*, Paris, 1999.

[Gam08] Gam: http://www.g-scop.inpg.fr/GAM/

[Gel98] Gelle E., On the generation of locally consistent solution spaces in mixed dynamic constraint problems, *Thèse de doctorat, Ecole polytechnique fédérale de Lausanne*, 1998.

[Ger90] Gero J-S., Design prototypes: A knowledge representation schema for design, *Journal of Artificial Intelligence*, Vol.11 (4), pp.26-36, 1990.

[Goo02] Goonetillake J. S., Carnduff T.W., An integrity constraint management framework in engineering design, *Computers in Industry, Special issue: CSCW in design*, Vol. 48 (1), pp. 29 – 44, 2002

[Gru94] Grudin J., Computer-supported cooperative work: its history and participation, *IEEE Computer* Vol.27 (5), 19-26, 1994.

[Gup93] Gupta A., Tiwari S., Distributed Constraint Management for Collaborative Engineering Databases, *Proceedings of the Second International Conference on Information and Knowledge Management*, Washington, USA, 1993.

[Gup96] Gupta L., Chionglo J.F., A constraint-based model of coordination in concurrent design projects, *5th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1996.

[Hal02] Halverson C.A., Activity Theory and Distributed Cognition: Or What Does CSCW Need to DO with Theories?, *Computer Supported Cooperative Work*, Vol.11, pp.243-267, 2002.

[Hal05] Halin G., Kubicki S., Architecture dirigée par les modèles pour une représentation multi-vues du contexte de coopération, *Proceedings of the 17th international conference on Francophone sur l'Interaction Homme-Machine*, 2005.

[Has95] Hashemian M., A Constraint-Based System for Product Design, *Concurrent Engineering*, Vol.3(3), pp.77-186, 1995.

[Hee04] Heerwagen J., Kampschroer K., Collaborative knowledge work environments, *Building Research and Information*, Vol. 32 (6), pp. 510-528, 2004.

[Hor99] Horvath L., Rudas I.J., "Constraint based modeling of automatic shape generation processes", *IEEE International Conference on Systems, Man, and Cybernetics*, 1999.

[Ipp08] Ippop, http://projects.opencascade.org/IPPOP, 2008.

[Isa00] Isaksson O., Trends in product modelling - an endrea perspective, *In Proceedings of produktmodeller*, Linkoping, Sweden, 2000.

[Jea98] Jeantet A., Les objets intermédiaires dans la conception. Eléments pour une sociologie des processus de conception. Sociologie du Travail 3 (1998), pp. 291–316

[Joh88] Johansen R., Charles J., GroupWare: computer support for business teams, *Ed. free press*, New York, USA, 1988.

[Kab06] Kabanda S., Adigun M., Extending model driven architecture benefits to requirements engineering », *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, Somerset West, South Africa, 2006.

[Kim93] Kimura F-L., Kjellberg F., Product modelling, *Annals of the CIRP*, Vol.42(2), pp.695-706, 1993.

[Kle00] Klein M., Towards a systematic repository of knowledge about managing collaborative design conflicts, Book chapter of *Artificial Intelligence in Design, Ed. Boston. Dordrecht: Kluwer Academic*, pp.129-146, 2000.

[Kut96] Kuutti, K. , Activity Theory as a Potential Framework for Human-Computer Interaction Research, In. Nardib (ed.), *Context and Consciousness: Activity Theory and Human-Computer Interaction*, pp.17-44, Cambridge: MIT Press, 1996.

[Lab08] Labrousse M. , Bernard A., FBS-PPRE: an Enterprise Knowledge Lifecycle Model, book chapter of *Methods and Tools for Effective Knowledge Life-Cycle-Management, Ed. Springer Berlin Heidelberg,* pp.285-305, 2008

[Lan96] Lander S., Corkill D., Designing integrated engineering environments: blackboard-based integration of design and analysis tools, *Concurrent Engineering*, Vol.4 (1), pp.59–71, 1996.

[Leb03] Leboterf, G., L'ingénierie : concevoir des dispositifs dans des environnements complexes et évolutifs, *Education permanente*, vol.157, pp.53-61, 2003

[Mar05] Martínez M., L.and Félez J., A constraint solver to define correctly dimensioned and over dimensioned parts, *Computer-Aided Design journal*, Vol.37 (13), pp.1353-1369, 2005

[Mar05] Marvie R., Duchien L., Les plates-formes d'exécution et l'IDM, in: L'ingénierie dirigée par les modèles, Au-delà du MDA, *Ed. Hermès Science*, 2005

[Mci95] Mcintosh K., Engineering Data Management: A Guide to Successful Implementation, *Ed. McGraw-Hill*, 1995.

[Mec06] Mechekour E., Etude des aspects informels en conception collaborative à distance ; Propositions pour des outils supports aux activités synchrones, *Thèse de doctorat, Laboratoire Sols, Solides, Structures de Grenoble (3S), INPG*, 2006

[Med04] Medelez O. E., Conception d'un environnement collaboratif multimédia pour des séances d'apprentissage du raisonnement clinique gérées à distance, Thèse *de doctorat, Laboratoire d'Informatique Medicale (LIM), Universite de Rennes I*, 2004.

[Met08] Meta model : http://www.metamodel.com/, 2008

[Mis90] Mistree F., Smith W.F., Decision-Based Design: A Contemporary Paradigm for Ship Design, *Transactions of the Society of Naval Architects and Marine Engineers*, Vol. 98, 1990.

[Mou04] Mouhoub M., Solving Dynamic CSPs, Book Chapter of *Advances in Artificial Intelligence, Lecture Notes in Computer Science Book Series, pp. 504-509*, 2004.

[Noe07] Noël F., Roucoules L., The PPO design model with respect to digital enterprise technologies among product life cycle, *Journal: International Journal of Computer Integrated Manufacturing*, Vol.21 (2)http://www.informaworld.com/smpp/title~content=t713804665~db=all~tab=issueslist~branches=21 - v21, pp.139 – 145, 2008.

[Pat05] Patil, L., Dutta, D., Ontology-based exchange of product data semantics, *IEEE Transactions on Automation Science and Engineering*, Vol.2 (3), pp.213- 225, 2005

[Pru03] Prudhomme G., Zwolinski P., Integrating into the design process the needs of those involved in the product life cycle, *Journal of Engineering Design*, Vol.14 (3), pp.333-353, 2003.

[Ram97] Ram D., Vivekananda N., Constraint Meta-object: a new object model for distributed collaborative design, *IEEE Transactions on System, Man, and Cybernetics- part A: systems and humans,* Vol. 27 (2), pp. 208 – 221, 1997.

[Ran95] Randoing J.-M., Les SGDT, Collection : Techniques de l'information Editeur : Hermes Sciences Publicat. (21 octobre 1995),

[Rob05] Robin M., Rose V., Modelling collaborative knowledge to support engineering design project manager. *In 17th IMACS World Congress, Scientific Computation, Applied Mathematics and Simulation*, 2005.

[Rol02] Roller D., Eck O., Advanced database approach for cooperative product design, *Journal of Engineering Design*, Vol.13 (1), pp.49-61, 2002

[Rou99] Roucoules L., Méthodes et connaissances : contribution au développement d'un environnement de conception intégrée, *Thèse de doctorat, Laboratoire Sols, Solides, Structures de Grenoble (3S), INPG*, 1999

[Sal95] Salaü I., La conception distribuée : Théorie et méthodologie, *Thèse de doctorat en automatique, Nancy, Université Henri Poincaré*, 1995

[Sch06] Schmidt D. C., Model-Driven Engineering, *IEEE Computer*, Vol.39(2), pp25-31, 2006.

[Sim73] Simon H., The structure of ill-structured problems, *Journal of Artificial Intelligence*, Vol.4, pp.181-20, 1973.

[Sri06] Sriplakich P., Blanc X., Supporting Collaborative Development in an Open MDA Environment, *Proceedings of the 22nd IEEE International Conference on Software Maintenance*, Philadelphia, USA, 2006.

[Ste08] STEP, http://www.steptools.com, 2008

[Ste95] Stephane M., Jeantet A., Les objets intermédiaires de la conception, chapitre du *communicationnel pour concevoir, pp.21-41 in, edited by J. Caelen and K. Zreik*, 1995.

[Sto01] Stokes M., Editor, Managing Engineering Knowledge; MOKA: Methodology for Knowledge Based Engineering Applications, *Ed. Professional Engineering Publishing*, London, 2001.

[SU03] Su S.Y.W., Meng J., Dynamic Inter-Enterprise Workflow Management in a Constraint-Based E-Service Infrastructure, *Electronic Commerce Research*, Vol.3(1&2), pp. 9-24, 2003.

[Sud05a] Sudarsan R., Fenves S.J., A product information modeling framework for product lifecycle management, Computer-Aided Design, Vol.37 (13), pp.1399-1411, 2005.

[Sud05b] Sudarsan R., Mehmet B., Information models for product representation: core and assembly models, *International Journal of Product Development*, Vol.2 (3), pp. 207 – 235, 2005.

[Suh01] Suh N.P., Axiomatic Design: Advances and Applications, *Oxford University Press*, 2001.

[Tic97] Tichkiewitch S., Methodology and product model for integrated design using a multiview system, *Annals of CIRP*, Vol.46 (1), 1997.

[Ull97] Ullman D.G., The Mechanical Design Process, 2nd edition, McGraw-Hill International Editions, Singapore, 1997.

[Ulr00] Ulrich K.T., Eppinger S.D., Product design and development, *Second edition, McGraw Hill International editions*, 2000.

[Ume90] Umeda Y., Takeda H., Function-behaviour, and structure, *Applications of Artificial Intelligence in Engineering V (Gero, J.S., Ed.),* pp.177-193, 1990.

[Var96] Varcard P., Aide à la programmation des outils en conception de produit, *Thèse de doctorat, ENSAM*, 1996.

[Wan02] Wang L., Shen W., Collaborative conceptual design – state of art and future trends, *Computer Aided Design*, Vol. 34 (13), pp.981-996, 2002.

[Wan03] Wang F., Fenves S.J, Towards modeling the evolution of product families, *International design engineering technical conferences and computers and information in engineering conference*, Chicago, Illinois, 2003.

[Wan04] Wang Y., Nnaji B. O., UL-PML: constraint-enabled distributed product data model, *International Journal of Production Research*, Vol. 42 (17), pp.3743-3763, 2004.

[Yan98] Yannou B., Les apports de la programmation par contraintes en conception, Chapitre de *conception de produits mécaniques : méthodes, modèles et outils, Ed. Hermès,* ouvrage collectif PRIMECA dirigé par M. Tollenaere, pp 457-486, 1998