UNIVERSITÉ DE GRENOBLE

HABILITATION A DIRIGER DES RECHERCHES

DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Sophie DUPUY-CHESSA

préparée au sein du **Laboratoire d'Informatique de Grenoble**

Modélisation en Interaction Homme-Machine et en Système d'Information : A la croisée des chemins

HDR soutenue publiquement le **1er décembre 2011**,
devant le jury composé de :

Mme Régine LALEAU
Professeur à l'Université Paris-Est Créteil, Président
Mme Mireille BLAY-FORNARINO
Professeur à l'Université de Nice, Rapporteur
M. Eric DUBOIS
Professeur au Centre Henri Tudor, Luxembourg, Rapporteur
M. Jean VANDERDONCKT
Professeur à l'Université Catholique de Louvain, Belgique, Rapporteur
Mme Gaëlle CALVARY
Professeur à Grenoble INP, Membre
M. Yves LEDRU
Professeur à l'Université Joseph Fourier, Membre
Mme Dominique RIEU
Professeur à l'Université Pierre-Mendès France, Membre

Il avait fait alors une grande démonstration de sa découverte à un congrès international d'astronomie. Mais personne ne l'avait cru à cause de son costume . . . L'astronome refit sa démonstration en 1920, dans un habit très élégant. Et cette fois-ci tout le monde fut de son avis.

Antoine de Saint-Exupéry

Remerciements

Ce travail est le reflet de contributions diverses, issues de nombreuses personnes de mes sphères autant professionnelle que personnelle. C'est avec toute ma gratutide que je leur adresse mes plus sincères remerciements.

Je tiens tout d'abord à remercier les trois rapporteurs de ce document, Mireille Blay-Fornarino, Eric Dubois et Jean Vanderdonckt d'avoir apporté leur point de vue et leurs commentaires sur mon travail de recherche. Je remercie également Régine Laleau pour avoir accepté de présider ce jury. Je suis heureuse que cela soit possible cette fois.

Mes remerciements vont ensuite à mes "directeurs" : Yves Ledru qui m'a appris à mener des recherches ; Dominique Rieu qui m'éclaire autant en sciences qu'en valeurs humaines ; Gaëlle Calvary qui ouvre sans cesse le champ des possibles et nous pousse à faire opérer la magie ;-)

Viennent ensuite mes étudiants : David, Guillaume, Jorge, Eric, Alfonso, Mario, Nicolas, Ludovic ... Mon travail et le plaisir que j'ai eu à l'accomplir leur doivent beaucoup.

Je n'oublie pas mes autres collègues, en particulier mes deux équipes qui m'apportent deux ambiances, deux approches, deux élans. Je remercie également Lydie, camarade en recherche depuis toujours, et Nadine qui m'a ouvert les portes du monde des sciences expérimentales.

Je voudrais ici souligner le soutien jamais mis en défaut de mes collègues de l'IUT. Je pense à Gaëlle, Nourry, Jérôme, Caro et Christine avec qui il est si agréable de travailler et qui savent si bien alléger le travail administratif. Je pense aussi Agnès qui partage mes bureaux et mes humeurs aussi bien à l'IUT qu'au labo.

C'est aussi au-delà de Grenoble que j'ai parfois trouvé l'inspiration. A tous les autres, tous ceux et toutes celles que j'ai croisés en conférence, en réunion de travail ou autre et qui m'ont apporté ces petits déclics qui permettent de progresser ou donnent envie de continuer.

Enfin le plus grand des mercis à mes proches, en tout premier lieu mes parents et mon mari, dont le soutien et la patience au quotidien n'ont jamais connu de faille. Ils sont derrière chacun de mes pas. Il y a, bien sûr, mes filles pour leur compréhension, mais également pour leurs incompréhensions qui me rappelent tous les jours l'essentiel. Enfin il y a mes copines, "nounous" officielles et officieuses (Marie, Pat, Fab ...), toujours prêtes à me dépanner. Qu'ils sachent tous que sans leur aide, ce travail n'aurait jamais vu le jour.

Table des matières

Introduction

1

Nous devons nous y habituer : aux plus importantes croisées des chemins de notre vie, il n'y a pas de signalisation.

Ernest Hemingway

Sommaire

Mes activités de recherche m'ont menée à suivre divers chemins : ceux du Génie Logiciel (GL) et des systèmes d'information (SI) dès la thèse, puis celui de l'informatique ubiquitaire en post-doctorat qui m'a dirigée vers l'Interaction Homme-Machine (IHM). Ce cheminement a toujours été motivé par une direction clairement tracée : ne pas limiter un système à son point de vue technique, mais considérer aussi ses utilisateurs et leurs contextes de travail. C'est cette croisée des chemins de l'IHM, des SI et du GL que nous allons tenter de baliser dans ce mémoire.

1.1 Contexte

Les progrès technologiques comme la miniaturisation des microprocesseurs et capteurs, le succès des technologies communicantes, aiguillés par l'objectif "d'accès à l'information par tous, n'importe où, n'importe quand, pour des besoins personnalisés", ouvrent un vaste de champ de possibilités. L'utilisateur peut désormais évoluer dans un environnement façonnable fusionnant des entités réelles et virtuelles. La "barre à franchir" pour la conception est déplacée vers le haut. En effet, ces systèmes, dits ubiquitaires, doivent relever des défis majeurs dans leur développement et leur usage.

Du point de vue de l'IHM, les défis sont liés à l'adaptation au contexte d'usage, à la prise en compte de la cohabition entre mondes réel et virtuel, à des interactions multiples et complexes entre ces mondes (variété des dispositifs, multimodalité, utilisabilité). La multiplicité des possibilités d'interaction (les gestes 3D, le suivi du regard, le suivi de la position de l'utilisateur, etc.) et d'adaptation peut se voir comme un facteur de souplesse avec sa compagne immédiate, la complexité.

De manière similaire, la conception d'un SI devient plus complexe. Le rôle des acteurs change. Le client d'un service est de plus en plus au cœur des processus métier. C'est ce que (Front et al. 2009) appelle "le syndrome de la pompe à essence" : le client qui demandait autrefois à l'employé de la station service de lui faire le plein d'essence, doit aujourd'hui le faire lui-même. D'acteur externe au processus, il devient acteur interne presqu'au même titre que les acteurs de l'organisation. Il n'est plus seulement utilisateur du SI ; il est aussi utilisateur du système informatique, complexifiant ainsi sa conception. Cette difficulté s'accompagne de celle des organisations : les organisations ont des périmètres de plus en plus larges (on parle d'organisation étendue, d'organisation virtuelle, d'organisation ouverte) et qui évoluent souvent (elles peuvent n'être conçues que pour un projet).

L'IHM et les SI traitent de problèmes complémentaires, avec dans les deux cas, l'objectif de fournir des systèmes adaptés aux besoins. L'IHM, en se basant sur l'ergonomie, cherche à fournir des systèmes utiles et utilisables. L'utilité détermine si le logiciel satisfait les exigences fonctionnelles et opérationnelles des utilisateurs alors que l'utilisabilité concerne la qualité de l'interaction (facilité d'utilisation, facilité de mémorisation, satisfaction de l'utilisateur, performances, adéquation au profil de l'utilisateur). L'approche SI complète cette vision en plaçant l'utilisateur dans son cadre professionnel afin de fournir une interaction située par rapport à l'organisation. Il me semble pertinent de rapprocher les points de vue de ces deux communautés afin de réaliser des systèmes adaptés aux utilisateurs et à leur contexte de travail.

1.2 Enjeux et Objectif Principal

Le couplage des visions de l'IHM et des SI a pour enjeu le plus évident **une meilleure adéquation des systèmes aux problèmes et aux besoins**. Un deuxième enjeu est **une plus grande productivité lors des phases de**

conception et de maintenance en maîtrisant mieux les pratiques communautaires mais aussi celles conjointes aux deux domaines.

Ces deux enjeux trouvent souvent des solutions au sein de méthodes de conception ou de certains de ses composants (processus, modèles ou outils). D'après Seligmann (Seligmann et al. 1989), une méthode est un ensemble de moyens d'investigation constitué par « a way of thinking, a way of modeling, a way of working and a way of supporting ». Cela signifie que toute méthode est définie par une approche ou un paradigme, comporte des modèles pour définir le produit, propose des processus ou démarches et doit être soutenue par des outils logiciels.

Mon objectif sera de trouver des méthodes ou composants de méthodes permettant de prendre en compte la complexité des SI actuels ainsi que les avancées en terme d'interaction.

1.3 Approche

Les prémices des méthodes de développement sont apparus dans les années 60 avec des manuels de conduite de projet pour analyser les applications de gestion. Dans les années 70 qui marquent le point de départ du génie logiciel, c'est le développement des méthodes d'analyse et de conception proprement dites. Elles peuvent être classées globalement en deux catégories :

- **les méthodes dites "classiques"** qui ont connu des approches cartésiennes (SADT (Marca et McGowan 1988)), systémiques (Merise (A. Rochfeld 1985)) ou (objet telles que OMT (Rumbaugh et al. 1991) ou OOSE (Jacobson et al. 1992)). Dans tous les cas, ces méthodes se basent sur le suivi d'un modèle de processus formalisé ("the way of working") qui s'appuie sur des modèles de produit ("the way of modeling").
- **les méthodes agiles** (eXtreme Programming (Beck 1999), Scrum (Schwaber et Beedle 2001) etc.). En novembre 2001, le Manifeste de Développement Logiciel Agile est publié, regroupant quatre valeurs ainsi que douze principes associés (Highsmith et Fowler 2001). La première valeur est l'aspect humain et communicatif avec la mise en avant de l'interaction entre les personnes plutôt que les processus et les outils. La deuxième valeur, la livraison d'un produit opérationnel, est la priorité vis-à-vis d'une documentation parfois excessive. La troisième valeur est la collaboration avec le client : elle est devenue primordiale et ne doit pas se restreindre à une simple négociation de contrat. La dernière valeur est la réactivité face au changement qui est préférée au suivi strict d'un processus qui ne mène parfois pas à la réponse aux demandes réelles du client.

Si les méthodes agiles proposent des techniques de gestion de projet intéressantes, elles n'offrent pas de réels guides pour les concepteurs et les développeurs dans leur travail, ni ne permettent une traçabilité des choix de conception. Le premier point est particulièrement bloquant dans le cadre de systèmes interactifs complexes pour lesquels les concepteurs sont peu formés et ont besoin de guidage. Le deuxième point me semble limitant pour envisager la compréhension et la maintenance du système.

Suivant les "bonnes pratiques" de l'agile, j'ai cherché à augmenter l'implication des utilisateurs en me basant sur les pratiques du domaine de l'IHM qui étudie depuis longtemps comment réaliser des systèmes utiles et utilisables. J'ai opté pour une approche "classique" qui apporte une structuration plus forte et une meilleure maîtrise des méthodes. En particulier, les modèles consituent un point d'ancrage possible entre les domaines de l'IHM et des SI. Ainsi, ils vont me servir de levier afin de :

- faciliter la coordination entre IHM et SI : chacun des domaines propose des modèles spécifiques qui permettent de prendre en compte leurs spécificités, principalement l'utilisateur pour l'IHM et l'organisation pour les SI ; leur mise en cohérence permet d'envisager la conception d'un système adapté aux besoins des utilisateurs dans le contexte de leur organisation. L'objectif est de coordonner des modèles et des processus différents afin de fournir une méthode de conception qui concilie les exigences issues des deux communautés.
- traiter une problématique commune suivant deux points de vue (SI et IHM) différents. Il s'agit de proposer une solution spécifique à chacun des domaines, mais en s'appuyant sur une approche commune (i.e. les modèles).

Cette utilisation des modèles me mène aussi à soulever des problèmes et proposer des solutions relatives à la gestion de modèles. Néanmoins ces questions sont toujours envisagées d'un point de vue des futurs utilisateurs des solutions. Aussi dans mes propositions, j'ai tenté de mettre en œuvre une bonne pratique de l'IHM qui se répand aussi dans le domaine du Génie Logiciel (GL) au sein du courant du GL empirique, celle de l'approche expérimentale. Ainsi, j'envisage autant que possible des évaluations de mes solutions par des utilisateurs concernés.

L'originalité de ce travail réside donc dans la fertilisation croisée entre trois domaines, l'IHM, les SI et le GL. En premier lieu, le domaine de l'IHM, nous apprend à considérer l'utilisateur final dès les premières phases d'un projet logiciel. Celui des SI prend en compte le contexte organisationnel. Enfin le Génie Logiciel fournit des techniques et outils sur lesquels appuyer mes propositions.

1.4 Organisation du mémoire

Ce mémoire est organisé en cinq chapitres. Le chapitre 2 dresse un tour d'horizon des travaux en ingénierie des modèles de manière générale, mais aussi dans mes deux domaines d'intérêt que sont l'IHM et les SI. Ce panorama permet d'expliquer et de justifier les verrous et mes objectifs associés.

Les chapitres 3, 4 et 5 présentent mes contributions pour lever les verrous identifiés au chapitre 2. Le chapitre 3 a pour objectif de faire le pont entre la conception de l'IHM et celle des SI, en proposant une méthode de conception alliant pratique de l'IHM et des SI pour la conception des systèmes interactifs complexes.

Le chapitre 4 décrit une problématique commune aux deux domaines, que j'aborde avec des objectifs spécifiques : l'auto-explication d'une interface pour l'IHM et celle d'un processus métier pour les SI.

Mes propositions dans les domaines de l'IHM et des SI m'ont aussi menée à identifier des problèmes génériques liés à la modélisation et à développer des pratiques et des outils pour sa gestion. Le chapitre 5 a pour objet de présenter deux de ces contributions : un cadre théorique pour la conception d'environnements de modélisation adaptés, et des outils pour la qualité des modèles et des langages.

Enfin le chapitre 6 dresse le bilan de mes contributions et identifie des directions de recherche à court et moyen termes pour des chemins communs.

Ingénierie des modèles

2

Ce qui est simple est faux. Ce qui est compliqué est inutilisable.

Paul Valéry

Sommaire

Ce premier chapitre décrit la voie existante pour la gestion de modèles, en particulier les chemins annexes suivis en IHM et en SI, mais aussi les obstacles qu'il reste à franchir.

2.1 Modèles et Langages

La modélisation est l'un des éléments communs utilisés en ingénierie de l'IHM et des SI. Elle était initialement utilisée à des fins de communication entre les concepteurs de systèmes et les parties prenantes. De nos jours, l'industrie tend à utiliser de plus en plus les modèles pour des tâches non contemplatives telles que la simulation, la génération de cas de tests ou de parties de code. Cette vision fut initiée à la fin des années 2000 par l'Object Management Group (OMG) avec l'initiative MDA (Model-Driven Architecture). Le MDA, basé sur le standard UML, promeut l'utilisation de modèles comme des artefacts essentiels dans le développement de logiciels. Il s'est ensuite étendu au Model-Driven Engineering (MDE) ou Ingénierie Dirigée par les Modèles (IDM en français) qui s'intéresse à tous types de langages de modélisation.

Suite à l'approche objet des années 80 et de son principe du "tout est objet", l'IDM promeut le principe du « tout est modèle ». Le concept central de l'IDM est la notion de modèle : le code source n'est pas considéré comme l'élément central d'un logiciel, mais comme un élément dérivé de la fusion et du «tissage» d'éléments de modélisation. Le but est de rendre explicit et d'encapsuler le savoir-faire au sein d'opérations automatisables sur les modèles. L'IDM met clairement l'accent sur le pouvoir génératif des modèles, ce qui ne reflète pas toute la richesse des travaux en ingénierie des modèles abordés dans cette section.

2.1.1 Modèles

Définitions

Il n'existe pas de définition universelle de ce qu'est un modèle. Dans (Kleppe et al. 2003), "un modèle est une description (d'une partie) d'un système écrit dans un langage bien formé. Un langage bien formé est un langage avec une forme bien définie (syntaxe), un sens (sémantique), qui convient pour une interprétation automatique par un ordinateur." Cette définition, si elle peut convenir dans le cadre de l'IDM, n'est clairement pas adéquate dans le cadre d'un processus de développement au sein duquel les modèles ont aussi un rôle de communication entre les acteurs du projet. La même remarque peut être faite avec les définitions de (Bézivin et Gerbé 2001, Seidewitz 2003, Favre 2004). Un modèle y est défini comme une simplification d'un système construit avec un but en tête, qui doit être capable de répondre aux questions à la place du système actuel (Bézivin et Gerbé 2001).

On déduit de cette définition la première relation entre le modèle et le système qu'il représente, appelée "représentation De" et notée avec le symbole μ sur la figure 2.1 (partie représentation générale). La partie exemple de la figure illustre qu'un modèle est une représentation d'un système, ici un système bancaire, selon un point de vue particulier (diagramme de classes, diagramme d'objets, diagramme de séquences).

Si les définitions précédentes ont l'avantage d'aboutir à des relations claires entre le modèle et le système étudié, elles oublient l'observateur présent dans la définition de Minsky (Minsky 1965) : « pour un observateur B, M est un modèle de l'objet O, si M aide B à répondre aux questions

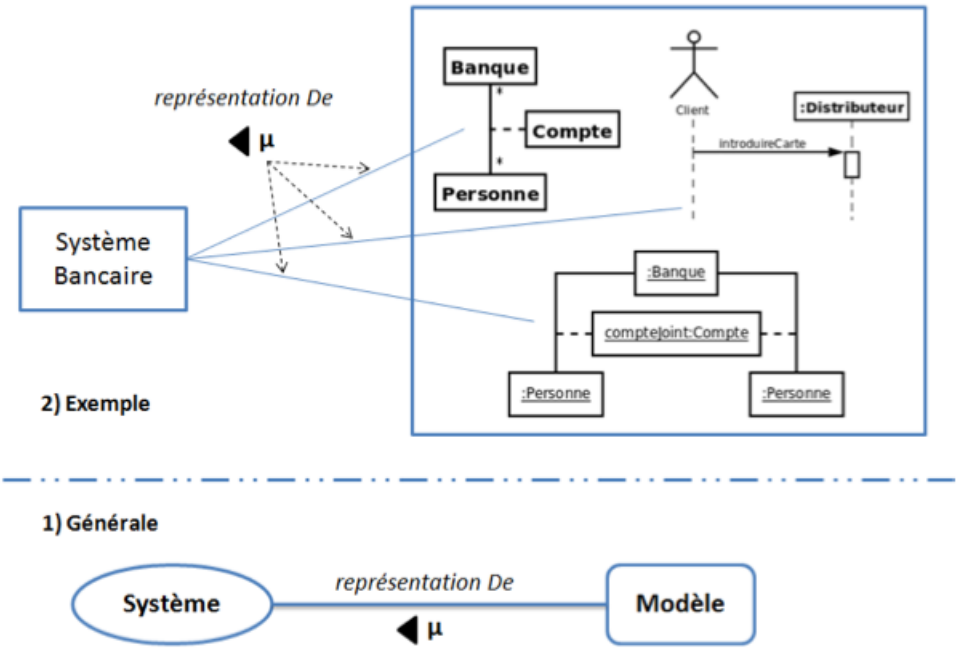


FIGURE 2.1 – *Relation entre Système et Modèle adaptée de (Favre et al. 2006) dans (Perez-Medina 2010)*

qu'il se pose sur O ». Cette définition met en évidence un point important qui est l'observateur à qui s'adresse le modèle. Cet observateur peut être un ordinateur pour correspondre aux définitions de l'IDM, mais ce n'est pas nécessairement le cas. L'observateur peut aussi être l'un des acteurs du processus de développement.

On peut retenir de ces définitions 1) qu'un modèle est une abstraction d'un objet et qu'en tant que tel, il ne doit pas représenter tout l'objet O ; 2) qu'un modèle est destiné à un observateur.

Qualité des modèles

Si la définition d'un modèle est globalement acceptée, il n'y pas de consensus pour caractériser ce qui fait un "bon" modèle.

En considérant la définition de la qualité proposée par l'ISO 9000, (Moody 2005) définit la qualité d'un modèle comme "the totality of features and characteristics of conceptual model that bear on its ability to satisfy stated and implied needs".

En considérant les modèles comme un type particulier de produit logiciel, la qualité peut être structurée en termes de critères/sous-critères jusqu'à l'obtention de métriques mesurables. Ainsi la qualité d'un diagramme de classes UML peut être définie comme un ensemble de buts tels que l'analyse, la prédiction ou la génération de code qui peuvent être décomposés en caractéristiques comme la cohérence, la complexité ou l'esthétique (Lange et Chaudron 2005). Par exemple, dans (Lange et Chaudron 2005), la complexité est une des caractéristiques de la communication qui peut être mesurée par des métriques comme la profondeur dans l'arbre d'héritage, la dynamicité et le nombre de classes par cas d'utilisation (Fig.

2.2). Si cette structuration de la qualité en arbre a l'avantage d'être pragmatique, les relations entre les critères sont souvent basées sur le jugement. Ainsi l'ISO et l'IEEE ont des hiérarchies différentes d'attributs de qualité des logiciels.

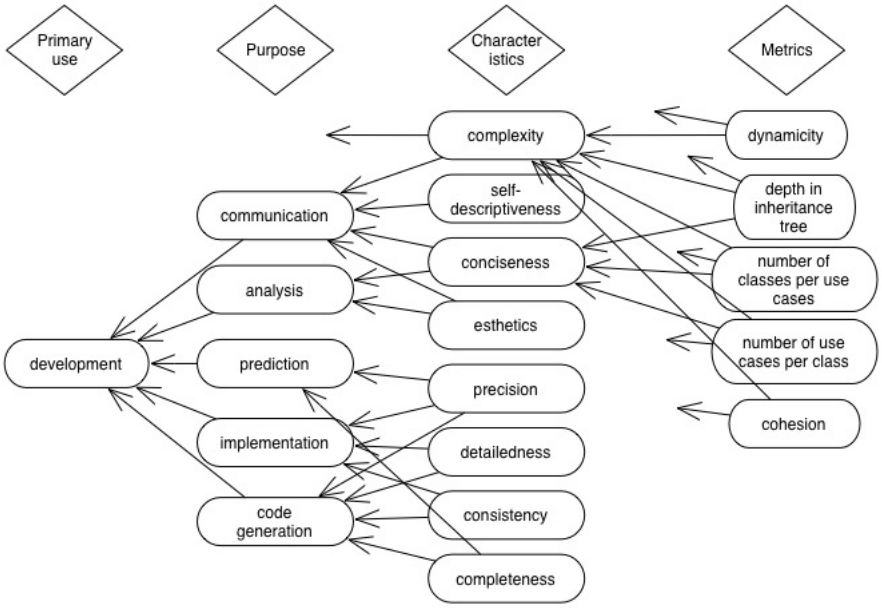


FIGURE 2.2 – *Référentiel de qualité des modèles UML de Lange (Lange et Chaudron 2005)*

De manière plus générale, (Mohagheghi et al. 2009) identifie 6 classes d'objectifs de qualité tels que l'exactitude qui définit l'inclusion des éléments pertinents et des relations entre eux et l'inclusion des assertions du domaine ou la complétude c'est-à-dire le fait d'avoir toutes les informations nécessaires pertinentes et suffisament détaillées pour le but de la modélisation.

Une deuxième approche consiste à s'appuyer sur les caractéritiques d'un langage pour étudier un modèle suivant des checklists. Ainsi pour les modèles UML 2.0, Unhelkar (Unhelkar 2005) classifie les objectifs de la qualité d'un modèle en syntaxique (avec une focalisation sur l'exactitude), en sémantique (c'est-à-dire la complétude, la cohérence et la représentation du domaine) et en esthétique (pour l'apparence et l'aide à la compréhension).

Enfin une dernière approche pour structurer la qualité des modèles consiste à s'appuyer sur l'étude des signes et de leur signification. Les frameworks suivants cette approche sont appelés sémiotiques. Ils ont été initiés par Lindland (Lindland et al. 1994) pour évaluer la qualité de n'importe quel type de modèle. Dans (Lindland et al. 1994), la qualité est détaillée en trois aspects :

- **Syntaxique**. La qualité syntaxique vérifie dans quelles mesures un modèle correspond aux constructions de son langage sans considérer le sens. Son but est d'obtenir la correction syntaxique.
- **Sémantique**. Elle indique le lien d'un modèle avec son domaine ou les connaissances des spécialistes du domaine. Ses objectifs sont la validité et la complétude. La validité signifie que toutes les asser-

tions du modèle sont correctes et pertinentes pour le problème. La complétude signifie que le modèle contient toutes les assertions correctes et pertinentes du domaine.

– **Pragmatique**. La qualité pragmatique est liée à l'interprétation du modèle de l'audience. Son but est la compréhension.

Ce référentiel a été validé par une étude empirique (Moody et al. 2003) et a donné lieu à de nombreux travaux complémentaires. En particulier, il a été étendu par Krogstie pour détailler la qualité pragmatique (Krogstie 1998; 2003) (Fig. 2.3). La qualité pragmatique dépend des utilisateurs du modèle. Ainsi il existe la qualité sociale pragmatique pour les parties prenantes, la qualité physique pour les concepteurs et la qualité technique pour les acteurs techniques (i.e. les outils). (Krogstie 1998; 2003) définissent aussi la qualité organisationnelle pour évaluer dans quelles mesures le modèle remplit les objectifs de la modélisation et la qualité empirique qui comprend les moyens de faciliter la compréhension tels que la lisibilité et l'agencement.

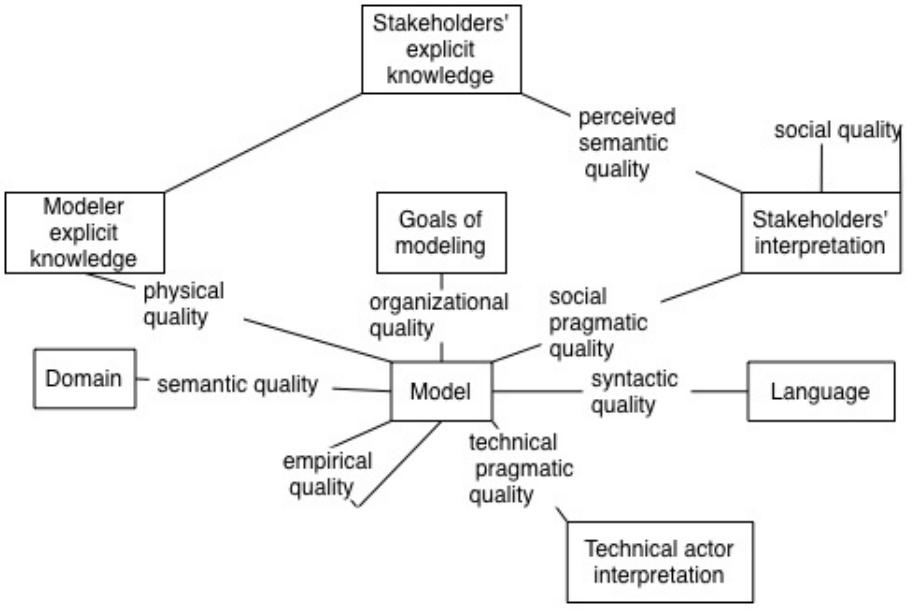


FIGURE 2.3 – *Référentiel sémiotique de Krogstie pour la qualité des modèles (Krogstie 2003)*

On peut aussi remarquer qu'il n'existe pas de modèle "bon" intrinsèquement : la qualité d'un modèle dépend du point de vue utilisé (celui des outils, celui des concepteurs, etc).

Dans les travaux qui viennent d'être présentés, les caractéristiques de qualité d'un modèle sont de deux ordres : celles portant sur la qualité des modèles en eux-même (par exemple, le nombre de cas d'utilisation par classe) et celles portant sur la qualité induite par le langage de modélisation auquel se conforme le modèle (par exemple, le choix de symboles appropriés pour faciliter la lisibilité). Comme dans le travail de Krogstie et al., je considère que les caractéristiques du langage sont un levier intéressant pour aboutir à des modèles de qualité. C'est pourquoi nous nous

intéressons par la suite aux langages de modélisation ainsi qu'à leur qualité.

2.1.2 Langages

Définitions

Un langage peut être défini suivant deux points de vue complémentaires, nommés définitions en extension et définition en intension.

Le langage est défini en extension quand on décrit l'ensemble de toutes les instances existantes. Dans cette vision, "a language is the set of all utterances of L" (Kleppe 2007). L'IDM utilise la relation « appartient A » (noté ϵ sur la figure 2.4) pour exprimer la relation entre le modèle et le langage dans lequel il est écrit (Favre et al. 2006).

Dans ce cas, un modèle est créé conformément à un méta-modèle, et ce modèle est, en même temps, une instance d'un langage. La relation qui unit un méta-modèle à un modèle est appelée « est conforme A » et noté χ sur la figure 2.4. Dans la figure, les modèles de classes, d'objets et de séquences sont conformes au méta-modèle UML.

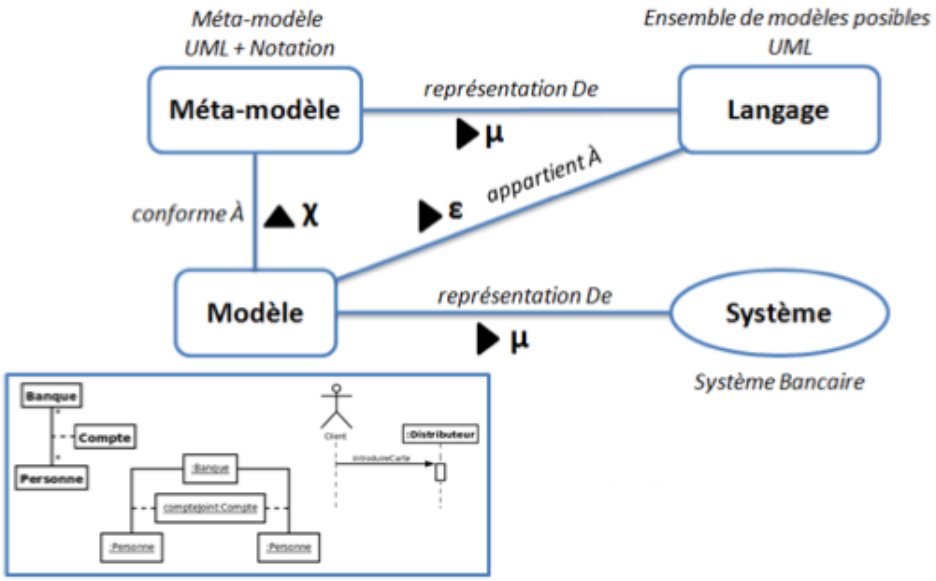


FIGURE 2.4 – *Relation entre Système, Modèle, Méta-Modèle et langage adaptée de (Favre et al. 2006) dans (Perez-Medina 2010)*

Un méta-modèle est un modèle d'un langage de modélisation (Favre et al. 2006). Le méta-modèle représente donc le langage. Le modèle auquel est conforme un méta-modèle est appelé méta- méta-modèle (par exemple le Meta-Object Facility, MOF) (OMG 2007b;a).

Si (Favre et al. 2006) présente la relation de conformité d'un modèle à son méta-modèle, l'exemple pris, celui d'une carte géographique, comprend aussi la conformité à une notation graphique non décrite par le méta-modèle. Dans (Perez-Medina 2010), la figure a donc été adaptée pour préciser ce point.

Cette vision, complémentaire à celle présentée en Fig. 2.4 correspond à une définition en intension (description des propriétés communes aux

instances possibles) d'un langage de modélisation. Ainsi un langage L est défini par une syntaxe abstraite, une syntaxe concrète et une sémantique.

Dans ce cas, la distinction entre méta-modèle et notation est faite en différenciant la syntaxe abstraite décrite par un méta-modèle et la syntaxe concrète pouvant être graphique ou textuelle. La syntaxe abstraite capture le vocabulaire et la taxonomie (i.e. les concepts) du langage (Fondement et Baar 2005) alors que la syntaxe concrète décrit la notation, c'est-à-dire la représentation des éléments du langage. Une séparation claire entre syntaxe abstraite et syntaxe concrète est une technique pour gérer la complexité de la définition d'un langage de modélisation car elle permet de définir les éléments d'un langage indépendamment de leur représentation.

La description du langage est complétée par une sémantique. La description de la sémantique correspond aux moyens de communiquer une compréhension des éléments linguistiques du langage en direction d'une ou plusieurs autres entités (personnes ou machines). Elle fait partie d'un langage pour permettre à son concepteur de communiquer sa compréhension du langage. Outre sa description en langue naturelle, la sémantique peut être définie au moins de 4 manières différentes (Kleppe 2007) :

- dénotationnelle c'est-à-dire en construisant des objets mathématiques (appelés dénotations ou sens) qui représentent le sens d'un modèle.
- opérationnelle en décrivant comment un modèle est interprété comme une séquence d'étapes de calcul. Cette séquence est souvent décrite sous la forme d'un système d'états-transition qui montre l'exécution état par état.
- translationnelle c'est-à-dire en traduisant le modèle dans un autre langage qui est bien compris. Cette approche a en particulier été utilisée pour préciser la sémantique d'UML en traduisant ses modèles dans des langages formels tels que Z (Dupuy et al. 2000), Object-Z (Kim et Carrington 1999) ou B (Laleau et Mammar 2006).
- pragmatique en fournissant des outils qui exécutent le modèle. De tels outils sont appelés implémentations de référence.

Suivant la définition de la syntaxe et de la sémantique, Fraser classifie les langages(Fraser et al. 1994) en informels, semi-formels et formels. (Fraser et al. 1994) parle de langages informels quand ils n'ont ni syntaxe, ni sémantique précisément définies ; de langages semi-formels si leur syntaxe est précisément définie, mais leur sémantique décrite en langue naturelle, de manière informelle et de langages formels qui possèdent une syntaxe et une sémantique définies rigoureusement. Le standard actuel de ce type de langage est UML. Par exemple, la syntaxe d'un langage formel peut être définie par une grammaire ou un méta-modèle alors que sa sémantique sera décrite de manière dénotationnelle, opérationnelle ou translationnelle. Des exemples de langages formels sont certains langages de programmation, Lustre (Caspi et al. 1987) ou Z (Spivey 1992) .

L'IDM nous fournir aujourd'hui les briques de base (syntaxe abstraite et concrète) pour créer facilement de nouveaux langages. On voit ainsi apparaître des langages répondant aux problèmes spécifiques à un domaine limité. Ce peut-être un langage spécifique au domaine de l'assurance ou à celui de la banque. On nomme ces langages, Domain Specific Languages (DSL) contrairement aux langages généraux, tels qu'UML, qui doivent être

utilsables pour plusieurs domaines d'applications. Cette distinction est orthogonale à toute autre classification. Par exemple, les langages peuvent être visuels ou textuels ; orientés-objet, orientés événements, etc. Plus spécifiquement, (Mohagheghi et al. 2009) définit un Langage de Modélisation Spécifique au Domaine (DSML) comme un langage de modélisation habituellement visuel qui est utilisé pour modéliser un domaine particulier.

Si les règles de définition d'un DSML sont aujourd'hui connues, qu'en est-il de l'utilité et de l'utilisabilité d'un tel langage ? Pour exprimer la problématique autrement, quelle est la qualité d'un langage ? Comment la définir et comment la mesurer ?

Qualité des langages

La qualité des langages est généralement assez difficile à appréhender. La plupart des travaux existants étudie un langage particulier en réalisant des expériences avec des utilisateurs. Ces évaluations n'en sont néanmoins pas simples car il ne s'agit pas d'évaluer la qualité d'instances du langage (par exemple, un diagramme de classes pour un système bancaire), mais le langage (par exemple, le diagramme de classes en UML). Ainsi Siau et Tian (Siau et Tian 2001) utilise une approche basée sur le modèle de traitement de l'information GOMS (Goals Operators Methods and Selection Rules (Card et al. 2000)) pour évaluer des diagrammes UML. Les auteurs mesurent le temps d'exécution pour réaliser certains des diagrammes UML pour déterminer leur complexité. De manière plus générique, (Aranda et al. 2007, Patig 2008) proposent des protocoles expérimentaux applicables à n'importe quel langage pour évaluer leur facilité de compréhension.

Une approche plus automatique pour évaluer la qualité d'un langage est d'étudier son méta-modèle. Dans (Mohagheghi et Aagedal 2007), l'hypothèse est qu'un méta-modèle complexe conduit à un pouvoir d'expression plus grand et donc à des modèles plus petits. Suivant la même approche, (Rossi et Brinkkemper 1996) propose une méthode de calcul de la complexité conceptuelle théorique. Dans ce cas, l'hypothèse (non démontrée) est qu'il existe une dépendance intrinsèque entre les méta-modèles et la facilité d'apprentissage du langage : "the more complex a metamodel, the harder the method is to learn". Ce travail a permis de comparer plusieurs langages de modélisation orientés objet à partir de leur métamodèle et conclut qu'ils deviennent plus complexes au fil du temps. En utilisant les mêmes règles de calcul, Siau et Cao cité dans (Krogstie 2003), aboutissent à des résultats similaires : UML est de 2 à 11 fois plus complexe que n'importe quel autre langage de modélisation orienté-objet.

Les approches précédentes se focalisent sur la syntaxe abstraite d'un langage. Un point de vue complémentaire est d'étudier la qualité de la syntaxe concrète. La Théorie de la Physique des notations (Moody 2009) donne 9 principes pour concevoir des notations visuelles cognitivement efficaces. Nous citerons à titre d'exemples la clarté sémiotique, qui illustre aussi le lien entre syntaxe concrète et abstraite, la tranparence sémantique, qui traite en partie du lien entre syntaxe concrète et sémantique et la discriminabilité perceptive qui est entièrement focalisée sur la notation :

– **La Clarté sémiotique**. La clarté sémiotique traite essentiellement de

la nécessaire correspondance 1-1 entre les éléments de la syntaxe abstraite et ceux de la syntaxe concrète.

- **La Transparence sémantique**. Elle définit dans quelle mesure la signification d'un symbole peut être déduite de son apparence. Les symboles doivent donc fournir des indices sur leur sens (la forme implique le contenu). Ce concept est proche de celui d' "affordance" en interaction homme-machine : l' "affordance" cherche la transparence dans les actions possibles pour l'utilisateur alors que la transparence sémantique vise la facilité de compréhension des concepts. La transparence sémantique n'est pas un état binaire, mais un continuum allant de la compréhension immédiate de la signification du symbole sans explication jusqu'à une interprétation différente ou oppposée à son sens.
- **La Discriminabilité perceptive**. La première phase de traitement de l'information chez l'être humain est la perception sensorielle. Pour un schéma graphique, cette perception est visuelle : reconnaissance des formes, des couleurs... Cette activité bénéficie d'une puissance de calcul importante (une partie du cerveau dédiée représentant plus d'un quart du cerveau) grâce notamment à un fonctionnement massivement parallèle. Le fait de pouvoir discerner facilement chaque type d'éléments graphiques par rapport aux autres est donc primordial. Cette propriété se nomme la discriminabilité perceptive.

De manière complémentaire à ces travaux, des référentiels étudient les langages dans leur globalité (syntaxe et sémantique) et identifie 5 caractéristiques pour un langage de modélisation (Krogstie 2003) (Fig. 2.5) :

- **la convenance au domaine** : les concepts du langage sont suffisament puissants pour exprimer n'importe quoi dans le domaine, mais pas plus. Cette caractéristique est liée à la sémantique du langage.
- **la convenance aux connaissances des acteurs** : le langage doit être approprié pour les acteurs de la modélisation (i.e. les concepteurs de modèles). Ainsi il vaut mieux baser un langage sur l'expérience avec d'autres langages précédemment utilisés (par exemple, baser UML sur OMT).
- **la convenance à la capacité d'externaliser les connaissances** : il ne doit pas y avoir d'assertions dans les connaissances explicites des participants qui ne peuvent être exprimées dans le langage. Cette caractéristique est aussi relative à la qualité de la sémantique.
- **la convenance à la capacité de compréhension** : le langage doit être accessible aux parties prenantes. Le langage ne doit pas avoir de trop nombreux concepts ; les concepts doivent être distinguables les uns des autres etc.
- **la convenance à l'interprétation par des acteurs techniques** : la syntaxe et la sémantique du langage doivent être suffisament formelles pour que les acteurs techniques c'est-à-dire les outils de modélisation puissent automatiser certains traitements sur les modèles.

On retiendra de ces travaux que la qualité d'un langage de modélisation dépend de la cible du langage (concepteurs de modèles, parties prenantes ou outils). En fonction de ces cibles, certaines caractéristiques doivent être mises en avant. Par exemple, un langage dédié à l'expression des besoins a pour premier objectif de répondre au besoin de convenance

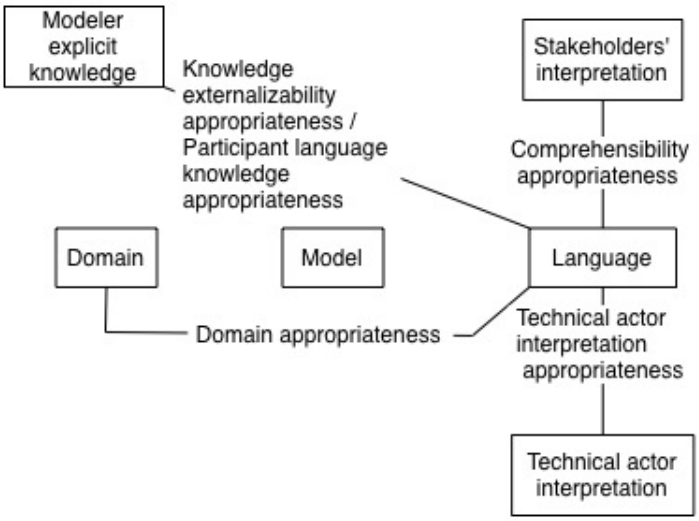


FIGURE 2.5 – *Référentiel sémiotique de Krogstie pour la qualité des langages (Krogstie 2003)*

à la capacité de compréhension des parties prenantes alors qu'un langage de conception doit permettre une interprétation par les acteurs techniques.

2.1.3 Les Savoir-Faire en ingénierie des modèles

L'ingénierie des modèles vise à gérer un grand nombre de modèles pour exprimer séparément les préoccupations des parties prenantes, des concepteurs, des architectes, etc. Pour cela, elle s'appuie principalement sur leur syntaxe abstraite en proposant des outils de transformations, de tissage ou de vérification de cohérence.

D'autres outils basés sur la sémantique d'un langage existent aussi. C'est l'appanage des méthodes formelles qui permettent de raisonner rigoureusement, à l'aide de la logique mathématique. Ainsi précision, abstraction et formalismes permettent raisonnement, vérifications, tests et preuves supportés par des outils de simulation, de preuve, de model-checking ou de tests.

Nous nous focaliserons ici sur les outils de l'IDM que nous utilisons dans nos travaux.

Transformations de modèles

Le principe de transformation a été introduit en MDA afin de produire un modèle spécifique à la plateforme (PSM) à partir d'un modèle indépendant à la plateforme (PIM) et d'un modèle de description de la plateforme (PDM). Le PIM est lui-même issu d'un modèle d'analyse indépendant des préoccupations informatiques, le Computational-Independent Model (CIM). Ce principe a été généralisé en IDM dont l'une des idées clef est de « considérer toutes les opérations génératives de la même manière », par des transformations.

Une transformation peut être vue comme un ensemble de règles qui décrivent comment un ou des modèles sources sont transformés en mo-

dèles cibles (Fig. 2.6). Une transformation est représentée par la relation "EstTransforméEn". Le modèle produit peut être du code, des cas de test, d'autres modèles, etc. Ainsi il existe plusieurs types de transformation : modèles vers modèles et modèles vers code (ou modèles vers texte) (Czarnecki et Helsen 2003).

La relation "EstTransforméEn" consiste donc à prendre des éléments d'un ou plusieurs modèles pour générer d'autres éléments de modèle(s). Cette génération représente un savoir-faire métier. Les transformations de modèles sont une manière de capitaliser le savoir-faire. De la même manière que les modèles ne sont pas une nouveauté, la notion de transformation n'est pas née avec l'IDM : les compilateurs de code ou encore XSLT sont aussi des illustrations de transformations. Néanmoins avec l'IDM les transformations sont devenues des objets de premier ordre : elles sont explicites, manipulables en dehors de tout autre code et capitalisables.

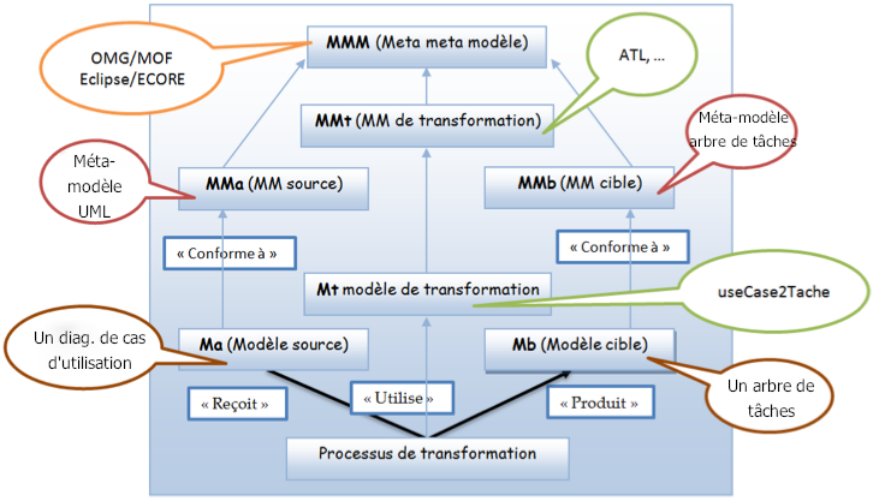


FIGURE 2.6 – *Le principe de transformation de modèles adapté de (Perez-Medina 2010)*

Les règles de transformation (Mt modèle de transformation) sont établies entre le méta-modèle source et le méta-modèle cible, c'est-à-dire entre l'ensemble des éléments du modèle source et celui du modèle cible. Le processus de transformation prend en entrée un modèle conforme au méta-modèle source et produit en sortie un ou plusieurs autre(s) modèle(s) conforme(s) au méta-modèle cible, en utilisant les règles préalablement établies. La transformation d'un modèle source Ma (par exemple, un diagramme de cas d'utilisation conforme à son méta-modèle MMa, UML) vers un modèle cible Mb (par exemple, un modèle d'IHM, un arbre de tâches, conforme à son méta-modèle MMb) est réalisée par le modèle de transformation Mt (conforme à son méta-modèle MMt qui peut être celui d'un langagel de transformation comme Atlas Transformation Language, ATL (Allilaire et Idrissi 2004)). Tous les méta-modèles utilisés dans le processus de transformation doivent être conformes à une spécification commune, par exemple le MOF, proposé par l'OMG.

Les transformations sont basées sur des langages. L'OMG a émis un ensemble de spécifications pour les langages de transformation de modèles : le standard QVT (Query View Transformation). (OMG 2005) est une

spécification de l'OMG pour les langages de transformation et de manipulation de modèles. La partie Query permet de sélectionner des éléments sur un modèle grâce au langage de contraintes OCL (OMG 2010). Une vue est un modèle à part, avec éventuellement un méta-modèle restreint spécifique à cette vue. Anfin une Transformation permet de transformer un modèle en un autre. Pour ces transformations, QVT définit plusieurs types de syntaxes : déclarative, impérative, hybride (QVT-relation, etc.) ainsi qu'une sémantique (QVT-Core). Les principes de QVT ont été implémentés dans de nombreux langages comme ATL (Allilaire et Idrissi 2004) ou Viatra2 (Csertan et al. 2002).

Tissage de modèles et vérification de cohérence

L'IDM ne se limite pas aux transformations de modèles. En effet, de nombreux modèles sont produits de manière indépendante par différents concepteurs. Il est alors nécessaire de faire des liens entre ces modèles ou tout au moins de vérifier leur cohérence.

Tissage de modèles (Fabro et Jouault 2007) propose une opération sur les modèles appelée « le tissage de modèles ou méta-modèles ». Le tissage précise les différents types de relations (i.e. les liens) entre les éléments de modèles (ou méta-modèles). Ce lien propose alors la relation entre le modèle source et le modèle cible. Elle permet également de voir quelles règles sont utilisées pour réaliser cette transformation. Afin d'expliquer le tissage de modèles, nous considérons un système d'information simple pour une bibliothèque décrit dans (Fabro et Jouault 2007). La figure 3.11 montre un exemple d'un modèle de tissage qui capture les relations entre un méta-modèle avec des structures fixes et des relations référentielles : « Foreign Keys » (représentant une clé étrangère en base de données relationnelle, de référence : « R1 ») et un méta-modèle qui contient des structures imbriquées (« Nested ») (représentant une base hiérarchique de données en XML, de référence : « X1 »). Ainsi, les deux schémas représentent la même information, mais des structures de données différentes sont utilisées. Une opération de tissage est spécifiée pour saisir les liens entre les deux schémas avec toutes les informations sémantiquement pertinentes.

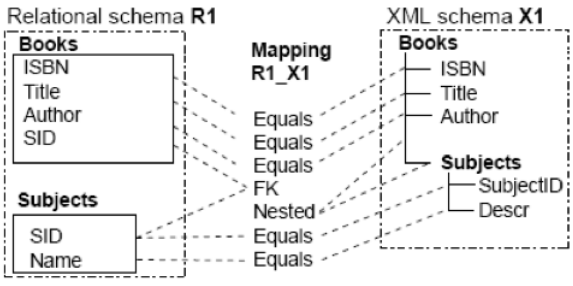


FIGURE 2.7 – *Exemple de tissage entre modèles (Fabro et Jouault 2007)*

Vérification de cohérence entre modèles La vérification de cohérence entre modèles a été identifiée dans les années 90 en ingénierie des besoins (Finkelstein et al. 1992) comme un requis pour permettre à différents

acteurs du développement d'un logiciel d'utiliser leur langage habituel tout en gardant une cohérence entre les modèles représentant les besoins. Néanmoins l'objectif n'est pas forcément de proscrire toute incohérence (Balzer 1991). Ainsi les incohérences peuvent être tolérées temporairement tout au long de la construction des modèles. Suivant ce principe, (Blanc et al. 2008) propose l'approche Praxis. Praxis recherche les incohérences plutôt que de vérifier la cohérence d'un modèle. Une règle d'incohérence définit un patron qui ne doit pas être présent dans un modèle. Si ce pattern est présent alors le modèle possède une incohérence. D'un point de vue conceptuel, une règle d'incohérence est la négation d'une règle de cohérence. La construction des modèles peut alors être surveillée afin de présenter en temps réel un diagnostic récapitulant les incohérences présentes dans les modèles.

Transformations de modèles, tissage et vérification de cohérence sont des techniques de gestion des savoir-faire qui peuvent être utilisées dans différents domaines et sur différents types de modèles. Dans la section suivante, nous appréhendons l'ingénierie des modèles au sein des domaines de l'IHM et des SI.

2.2 Ingénierie des modèles en IHM et SI

Cette section présente une vision synthétique des travaux basées sur l'ingénierie des modèles dans les domaines de l'IHM et des SI.

2.2.1 Ingénierie des modèles en IHM

La communauté IHM a une longue expérience des modèles, expérience qui a débuté bien avant que l'IDM n'existe en tant que domaine. Selon (Coutaz 2010), dans les années 80, les grammaires (que nous nommerions aujourd'hui méta-modèles) étaient les bases formelles de la génération d'interfaces textuelles ou graphiques ; plus récemment, les transformations étaient implémentées par du code au sein des générateurs d'interfaces et le mapping se limitait à l'expression de correspondances entre les éléments de l'IHM et ceux du code métier. La conception d'interfaces basée sur les modèles est donc une pratique de longue date, qui met à profit aujourd'hui les pratiques de l'IDM. Cette section résume ces pratiques que nous avions étudiées dans (Pérez-Medina et al. 2007).

Modèles et langages en IHM

De nombreux travaux (Paterno 1999, Furtado et al. 2004, Mahfoudhi et al. 2006, Raneburger 2010, Vanacken et al. 2007, Calvary et al. 2003, Dubois 2009) ont adopté une approche basée sur les modèles pour concevoir des IHM. La plupart ont pour modèle de base, l'arbre de tâches qui représente une décomposition des activités de l'utilisateur et du système. La dimension verticale de l'arbre correspond à la hiérarchie des tâches, la dimension horizontale à leur agencement séquentiel. Par exemple, la fig. 2.8 utilise le formalisme des ConcurTask Trees (Paterno et al. 1997) pour représenter les tâches nécessaires à la réservation d'un chambre d'hotel. Il

peut se lire comme suit, en partant de la tâche "HotelReservation". L'utilisateur sélectionne le type de chambre souhaité ("SelectRoomType"). C'est une tâche d'interaction entre l'utilisateur et le système qui se décompose en deux sous-tâches "SelectSingleRoom" et "SelectDoubleRoom". L'opérateur de décomposition est le choix ce qui signifie que l'utilisateur effectue l'une ou l'autre de ces tâches. Une fois le type de chambre choisi, l'utilisateur peut effectuer sa réservation ("MakeReservation"). "MakeReservation" est une tâche abstraite qui est consituée de deux sous-tâches, la tâche système "ShowAvailability" puis la tâche d'interaction "SelectRoom".

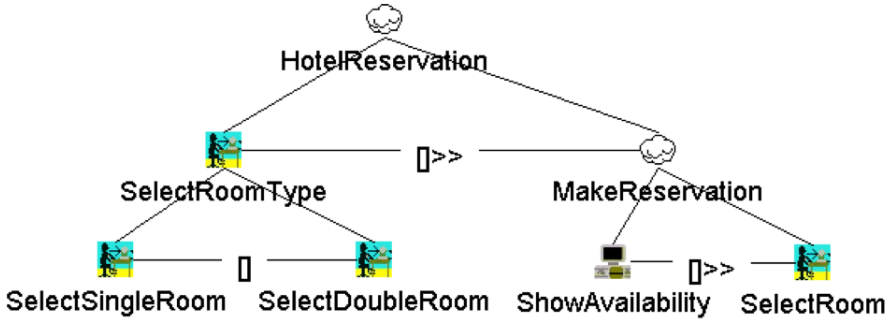


FIGURE 2.8 – *Exemple d'arbre de tâches (Paterno 2003)*

Ce modèle est complété par un modèle du domaine qui décrit les objets manipulés par les tâches. Il peut aussi l'être par d'autres modèles suivant les préoccupations : par exemple, des modèles d'interaction (Dubois 2009, Coutrix et Nigay 2006, Chalon et David 2004) qui représentent les objets phyisques et réels manipulés, ou des modèles de plateforme et d'utilisateur pour les systèmes d'adaptation au contexte (Calvary et al. 2003, Vanacken et al. 2007).

Dans le cadre des modèles d'interaction, l'attention a été principalement portée sur le support aux trois dimensions d'un modèle (Beaudouin-Lafon 2004) : 1) le pouvoir descriptif (ou de classification) qui est la capacité de décrire une grande variété d'interfaces et de les classifier ; 2) le pouvoir génératif qui aide les concepteurs à créer de nouvelles solutions ; 3) le pouvoir comparatif pour aider à estimer des alternatives de conception. Ces dimensions étant plutôt centrées vers le concepteur que vers l'instrumentation des modèles, une ingénierie des modèles basés sur la définition d'un langage avec une syntaxe et une sémantique, n'a pas nécessairement été mise en place.

Ce n'est pas le cas pour de nombreux langages de descriptions d'interfaces (UIDL), comme UsiXML [1], UIML [2] ou XIML [3]. Vanderdonckt (Vanderdonckt 2008) et Coutaz (Coutaz 2010) notent même une profilération de ces UIDL qui engendre le besoin de définir un standard. Ainsi des processus de standardisation sont initiés auprès d'OASIS pour UIML ou auprès du W3C pour UsiXML.

1. www.usixml.org
2. http ://www.uiml.org
3. www.ximl.org

Gestion des savoir-faire en IHM

Un des maître-mot en IHM est le centrage utilisateur. La conception centrée utilisateur (Norman et Draper 1986) est un paradigme définissant 12 principes clés qui mettent l'utilisateur à une place centrale au sein des processus de développement. De manière similaire, ISO 13407 (ISO 1999) promeut l'utilisabilité comme l'un des aspects très importants de la conception d'IHM, mais ne s'appuie pas spécifiquement sur des modèles.

En contre-pied à la conception centrée utilisateur, Constantine et al (Constantine et al. 2003) ont défini l'approche centrée-usage comme "a systematic, model-driven approach to visual and interaction design for user interfaces in software and Web-based applications" où "the center of attention is not users per se but usage, that is, the tasks intended by users and how these are accomplished". Ils positionnent clairement le centrage-usage en opposition au centrage utilisateur en spécifiant leurs différences fondamentales (Fig. 2.9).

Conception centrée *utilisateur*	**Conception centrée *usage***
Focalisation sur l'utilisateur : expérience et satisfaction de l'utilisateur	Focalisation sur l'usage : amélioration des outils permettant la réalisation de tâches
Dirigée par les données des utilisateurs	Dirigée par les modèles
Implication substantielle de l'utilisateur : étude de l'utilisateur, conception participative, retour utilisateur et évaluations par les utilisateurs	Implication sélective de l'utilisateur : modélisation exploratoire, validation des modèles, évaluations structurées de l'ergonomie
Description des utilisateurs et de leurs caractéristiques	Modèles des relations de l'utilisateur avec le système
Modèles de conception réalistes ou descriptifs	Modèles de conception abstraits, permettant de retarder la prise en compte des détails concrets de l'interface
Conception par prototypage itératif	Conception par modélisation
Variété de processus informels ou non spécifiés	Processus systématique et spécifié

FIGURE 2.9 – *Différences fondamentales entre conception centrée usage et conception centrée utilisateur d'après (Constantine et al. 2003)*

Si conception centrée usage et conception centrée utilisateur semblent s'opposer, des travaux issus de Caméléon (Calvary et al. 2003, Sousa et al. 2007) montrent qu'elles sont compatibles. On y retrouve mis en œuvre un ensemble de notions proches de la conception centrées utilisateur, telles que le profilage des utilisateurs, et de la conception centrée usage, telles qu'une approche dirigée par les modèles pour la génération l'interface.

La plupart de ces travaux qui suivent une conception basée sur les modèles suit une approche descendante : les modèles sont raffinés par transformations jusqu'au code. Néanmoins, le cadre Caméléon (Calvary et al. 2003) qui sert de référence pour les interfaces adaptables au contexte d'usage, prévoit des transformations descendantes, mais aussi ascendantes pour permettre la rétro-conception. Comme le montre la Fig. 2.10, il est constitué de 3 parties :

- sur la gauche, un ensemble de modèles ontologiques qui donne lieu à des modèles archétypaux et observés. Les modèles ontologiques représentent les dimensions clés pour traiter un problème donné ; ils spécifient les concepts inhérents au domaine et leurs relations. Les modèles archétypaux sont créés par des concepteurs et servent de base au processus de conception. Les modèles observés sont des

modèles effectifs qui guident l'adaptation au contexte lors de l'exécution.
- sur la droite, un processus de développement qui explique comment produire une interface pour un contexte d'usage archétypal. Ce processus n'est pas le seul et tout niveau d'abstraction peut être un point d'entrée du processus.
- en bas à droite, un processus d'éxécution qui montre comment les interfaces utilisateur et l'infrastructure d'éxécution peuvent coopérer pour viser un autre contexte d'usage.

Ce cadre est utilisé, entre autres, pour nommer les différents niveaux d'abstraction des modèles employés :
- le niveau tâches-concepts répertorie, pour chaque tâche, les concepts qu'elle manipule. Il correspond au Computational-Independent Model (CIM) du MDA
- l'interface abstraite (AUI) structure l'IHM en espaces de travail et spécifie l'enchaînement entre espaces. C'est le niveau Platform-Independent Model (PIM) du MDA.
- l'interface concrète (CUI) réifie les espaces de travail en fenêtres ou canevas ; leur contenu ainsi que leur enchaînement en objets d'interaction (aussi appelés widgets ou interacteurs). Les boutons, champs texte, menus déroulants, etc. sont des exemples d'interacteurs. Ces préoccupations sont de l'ordre du Platform Specific Model (PSM) dans la terminologie MDA.
- l'IHM finale (FUI) est une version exécutable ou interprétable de l'IHM concrète qui correspond au niveau code.

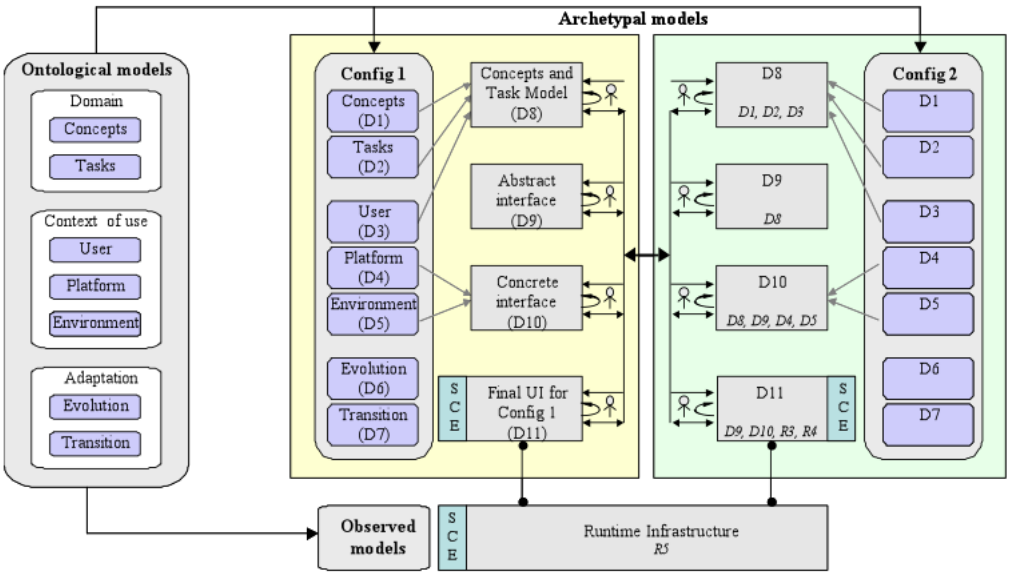


FIGURE 2.10 – *Cadre de référence Caméléon (Calvary et al. 2003)*

Les transformations envisagées dans Caméléon peuvent être soit verticales, c'est-à-dire proposer un changement de niveau d'abstraction vers le bas (réification) ou vers le haut (abstraction) ; soit horizontales en restant au même niveau d'abstraction (traduction). Ces transformations peuvent faire appel à des liens de tissage qui les aident à transformer le modèle de manière adéquate (par exemple, les concepts manipulés par une tâche

sont groupés ensemble afin de demeurer ensemble après transformation) (Sottet et al. 2008).

Ce travail est le reflet de la tendance en IHM où le principe de transformation est utilisé très souvent alors que ceux de tissage ou de gestion de cohérence ne sont guère usités.

2.2.2 Ingénierie des modèles en SI

La communauté SI a une longue et forte culture de l'utilisation des modèles. En effet, dès les années 70, les premiers modèles de données, comme le modèle Entité/Association font leur apparition. Depuis l'ingénierie des SI s'est toujours fortement appuyée sur des modèles que ce soit pour représenter des produits du processus du développement (par exemple, un modèle de base de données), ou le processus qui a mené à la réalisation des produits.

Langages et Modèles en SI

Les informations à représenter dans le cadre de la modélisation d'un SI ne se limite pas à décrire le système à développer sous forme de modèles de produits. Elles doivent aussi permettre de représenter l'organisation entre les acteurs d'un processus ainsi que leurs buts. Aussi le domaine des SI s'intéresse à représenter les processus par des modèles. C. Rolland définit de tels modèles de la façon suivante (Rolland 2005) : "Un modèle de processus prescrit une manière de faire, une démarche méthodologique pour atteindre la cible souhaitée. Il décrit à un niveau abstrait et idéal, la façon d'organiser la production du produit : les étapes, les activités qu'elles comprennent, leur ordonnancement, et parfois les critères pour passer d'une étape à une autre. Il joue le rôle de moule des processus d'ingénierie." Ces modèles peuvent servir à représenter des processus métier aussi bien que des processus de développement, qui sont les processus métier des informaticiens.

A titre d'exemple, la figure 2.11 présente le modèle global de processus de la méthode Symphony (Hassine 2005) proposée par la société Umanis et formalisée par l'équipe SIGMA du LIG. Ce processus débute par une étude du cahier des charges qui permet d'identifier les processus métier qui seront traités itérativement par des cycles de vie en Y. Dans un cycle en Y, la branche gauche correspond à l'étude des besoins fonctionnels ; la droite à celle des besoins techniques (contraintes logicielles, architectures logcielle et technique, etc) et la branche centrale à la mise en commun des deux précédentes branches pour fournir un système. Chaque branche est constituée de phases telles que la spécification conceptuelle des besoins ou l'analyse, qui se décomposent en activités décrites sous forme de diagramme d'activités UML. On parle de modèle de processus orienté activités.

Comme celui de l'exemple de Symphony, la plupart des modèles de processus sont orientés activités c'est-à-dire qu'il décrivent le processus sous forme des activités exécutées pour produire un produit et sur leur ordonnancement. Mais d'autres catégories de modèles de processus ont été identifiées dans (Rolland 2005). Par exemple, il existe des modèles orien-

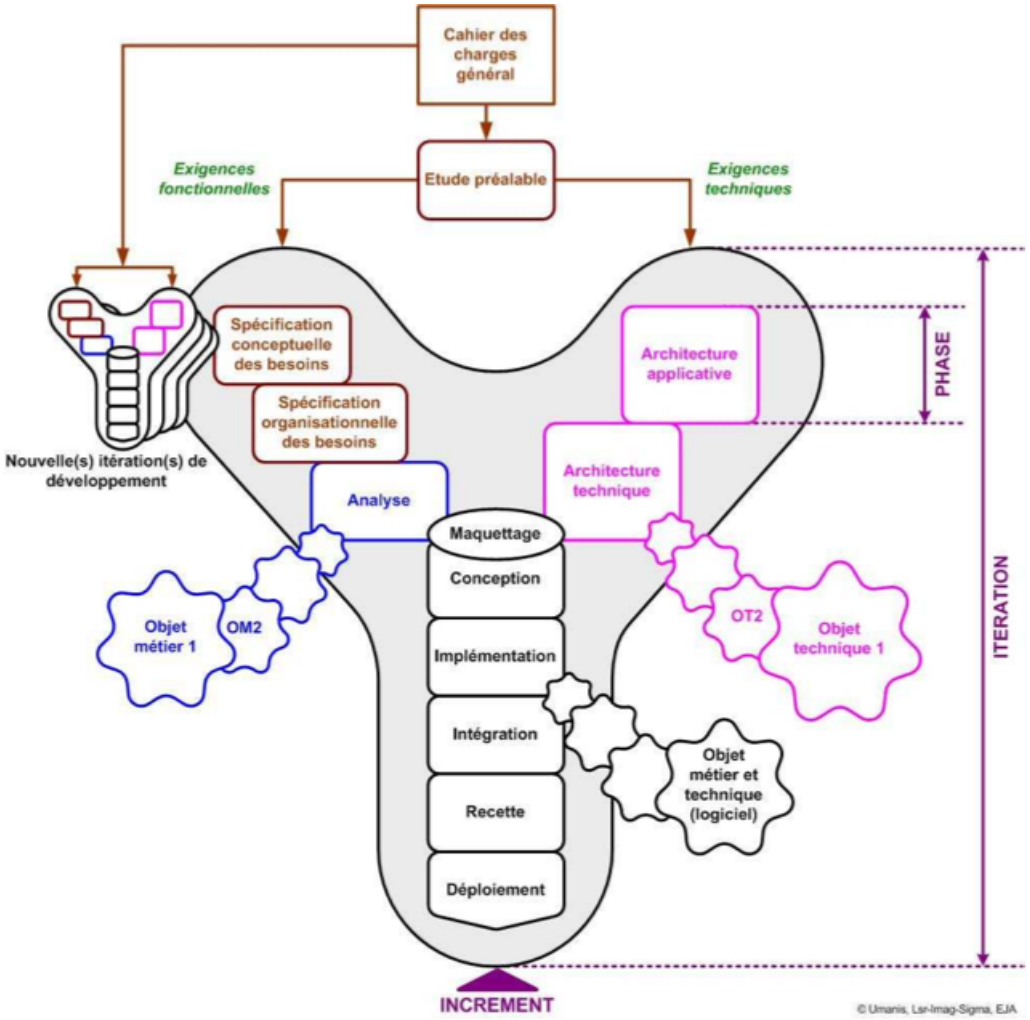


FIGURE 2.11 – *Exemple de modèle de processus issu de (Hassine 2005)*

tés but qui se focalisent sur le choix d'une stratégie parmi un ensemble de possibilités pour atteindre une intention ou but. L'exemple typique de ce genre de modèles sont les MAPs (Rolland et al. 1999). A titre d'illustration, la figure 2.12 représente la MAP pour la construction d'un diagramme de cas d'utilisation. Les éléments entourés sont des buts qui peuvent être atteints en suivant des stratégies représentées par les flèches. Par exemple, le but "Conceptualiser un cas d'utilisation" peut être réalisé en considérant d'abord le cas normal, par inclusion, par abstraction, ou par extension.

Ainsi en modélisation des SI, trois niveaux d'abstraction sont généralement considérés, chacun correspondant à des langages de modélisation différents (Front et al. 2009) :

- le niveau intentionnel décrit les besoins des acteurs d'une organisation au sein d'une activité. Les langages de modélisation sont alors généralement orientés buts ou stratégies comme les MAPs ou KAOS (van Lamsweerde 2001).
- le niveau organisationnel représente l'organisation à mettre en place pour répondre aux objectifs. Les éléments importants à modélisation dans ce cadre sont les processus métier et les activités des différents

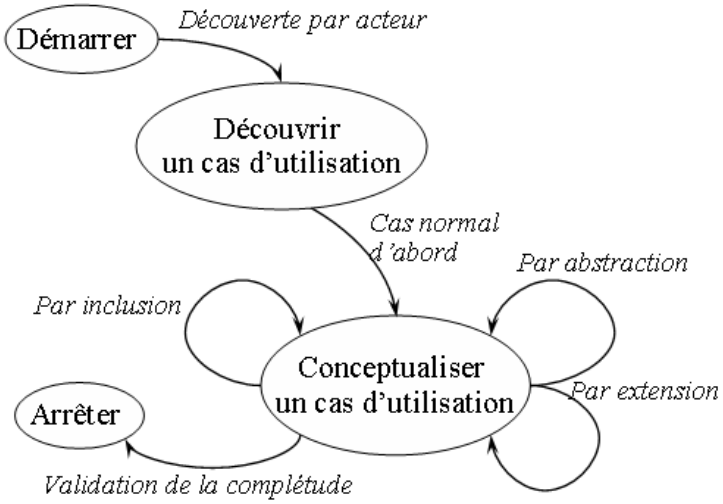


FIGURE 2.12 – *Exemple de MAP issu de (Rolland 2005)*

acteurs. Les langages couramment utilisés dans ce cadre sont les diagrammes d'activités UML ou les langages issus des standards BPM (Business Process Management), comme BPMN2 (OMG 2011).
– le niveau opérationnel décrit les solutions informatiques. Les langages de modélisation utilisés sont ceux du Génie Logiciel dont le standard UML.

La modélisation quasi-systématique des produits ou des processus d'une méthode de conception est donc l'une des caractéristiques de l'approche en SI. Contrairement à l'IHM, il n'existe pas plétaure de langages spécifiques au SI. Les standards tels qu'UML, sont néanmoins adaptés à des besoins spécifiques tels que la réutilisation ou la personnalisation de modèles. Par exemple, la variabilité, qui traduit la capacité d'un système à être changé, personnalisé et configuré en fonction d'un contexte, a été introduite dans les modèles UML pour permettre d'obtenir une spécification adaptable (Clauss 2001, Ziadi et Jézéquel 2005, Saidi et al. 2007).

Gestion des savoir-faire en SI

Le domaine des SI s'est particulièrement intéressé à la manière d'utiliser les modèles dans le cadre de la conception. Ainsi de nombreuses méthodes de conception (A. Rochfeld 1985, Kruchten 2003, Palmer et Felsing 2002, Hassine 2005) s'appuyant sur des processus clairement définis, ont été proposées et expliquent quand et comment utiliser les modèles. Au sein de ces processus, les approches développées en IDM apportent assez naturellement des réponses aux besoins des systèmes d'information en faisant en sorte que les artefacts de la conception deviennent des éléments inhérents au système. Même si les solutions existantes privilégient les transformations de modèles pour raffiner les modèles de l'expression des besoins jusqu'au code, d'autres besoins émergent comme celui de tisser des liens ou de vérifier la cohérence entre modèles pour gérer la collaboration au sein d'équipes de projet regroupant des compétences diverses. Si les outils de l'IDM ne sont pas suffisamment matures pour supporter

totalement ces besoins, les techniques sous-jacentes sont tout de même envisagées en SI.

De plus, le domaine des SI utilise un concept complémentaire à ceux de l'IDM pour capitaliser des modèles et les réutiliser. Il s'agit des patrons (design patterns). Un patron exprime et capitalise un problème récurrent à résoudre, propose une solution possible à ce problème et offre un moyen d'adapter la solution à un contexte spécifique. Si les patrons sont répandus dans de nombreux domaines dont celui de l'IHM (Dearden et Finlay 2006), ils se démarquent en SI par l'utilisation systématique de modèles pour décrire la solution du problème considéré. Les plus connus de ces patrons sont ceux de (Gamma et al. 1995) qui donnent des solutions sous forme de diagrammes de classes et/ou de séquences. Par exemple, le patron "Composite" qui explique comment structurer des classes de manière à ce que les clients ignorent la différence entre des objets composés et des objets individuels propose la solution donnée en Fi. 2.13.

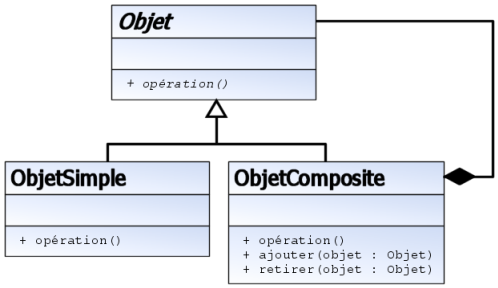


FIGURE 2.13 – *Solution du patron "Composite" issu de (Gamma et al. 1995)*

Le domaine des SI est donc un grand utilisateur des solutions de l'IDM, mais il contribue aussi à développer des techniques complémentaires de gestion de modèles telles que les patrons de conception.

2.2.3 Bilan

Les domaines de l'IHM et des SI s'appuient tous deux sur la modélisation pour comprendre le futur système et gérer les savoir-faire. On peut noter que les pratiques en matière de modèles sont plus standardisées dans le domaine des SI et que le recours à la modélisation y est plus systématique. Les modèles de l'IHM sont centrés sur l'utilisateur, ses tâches et des modèles de l'interfaces. Ceux des SI représentent le système mais aussi les buts des acteurs et leur organisation. La modélisation traduit bien les préoccupations et spécificités de chacun des domaines : l'utilisateur pour l'IHM et l'organisation pour les SI.

2.3 CONCLUSION

2.3.1 Bilan et Verrous

Cette présentation du domaine de l'ingénierie des modèles a mis en évidence sa maturité : les concepts de langages, modèles, transformations ... sont précisément définis. Il est donc possible de créer des langages, des modèles, de transformer des modèles. Nous avons pu constater dans

notre rapide tour d'horizon de la modélisation en IHM et en SI de **la diversité des préoccupations** abordées par les modèles (Favre et al. 2006) : ils peuvent servir à décrire différentes facettes d'un même système (pour moi, principalement l'IHM et l'organisation) à différents niveaux d'abstraction (tâches et concepts, AUI ou CUI en IHM). Dans mon contexte, il s'agit de prendre en compte les spécificités de l'IHM et des SI afin de proposer des systèmes adaptés aux entreprises et à leurs employés. Notre premier verrou est donc **la gestion de la séparation des préoccupations entre IHM et SI** tout en garantissant la cohérence entre les deux domaines. Pour ce faire, il faut gérer la traçabilité et la cohérence des modèles de l'IHM et des SI à différents niveaux d'abstraction. Ce point est nécessaire pour réaliser un pont entre les deux domaines.

Outre cette nécessaire coordination entre les deux domaines, l'utilisation de modèles induit des problèmes communs identifiés dans le cadre de l'IDM, mais que nous n'aborderons pas nécessairement du point de vue technique, mais de celui des utilisateurs (finaux ou concepteurs).

Ainsi l'un des défis pour l'IDM est la gestion des évolutions des modèles qu'impliquent l'adaptation aux technologies ou l'adaptation des besoins (Barbier 2007 ?) : "How do Model Driven Development techniques and tools make upstream models (whose fluctuation cycles are those of requirements) perpetually consistent and inline with downstream models (which are subject to technological changes and improvements) ? ". Une solution à cette question peut être celle des "modèles interactifs" proposée par (Krogstie et Jørgensen 2002) : "With interactive models, the Information System makes the models available to the users at runtime, and the behaviour of the system is partly controlled by the models. By altering the models the users can thus modify the behaviour of the system to fit their needs." En considérant les modèles interactifs, il est possible d'envisager le défi de l'« **expert-user modelling** » présenté l'année dernière aux assises du Groupement De Recherche Génie de la Programmation et du Logiciel (Albert et al. 2010) : comment donner aux experts d'un domaine les moyens de construire ou de faire évoluer leur propre système informatique en utilisant des techniques de modélisation ?

Un des verrous à ce problème est **la dichotomie entre la conception et l'exécution** (Coutaz 2010). Il est nécessaire de la supprimer pour que les modèles soient le reflet du code et le code celui des modèles. L'intérêt est de permettre aux analystes ou aux utilisateurs finaux de comprendre et faire évoluer leur système en modifiant les modèles afin d'obtenir un résultat plus conforme aux besoins des utilisateurs ou de l'entreprise.

Ce point met en évidence le besoin de disposer de modèles de bonne qualité du point de vue de leurs utilisateurs. En IDM, l'accent a été mis sur l'ingénierie des transformations de modèles alors qu'un intérêt limité a été porté à la production de modèles simples et intuitifs (Barbier 2007 ?). **La qualité des modèles** a bien été abordée, mais elle demeure problématique (Favre et al. 2006, Vanderdonckt 2008).

La qualité des modèles est partiellement liée à celle des langages de modélisation. En IHM, (Vanderdonckt 2008) note que "For instance, a UIDL suffering from no semantics may suffer from incorrection, lack of expressiveness, and lack of separability. A UIDL suffering from no stylitics may suffer from stylistic incompleteness and, therefore, from lack of

expressiveness." Aussi comme le souligne (Favre et al. 2006), un des verrous demeurant pour l'IDM est la "définition de méta-modèles rigoureux, séparant la sémantique du langage et sa syntaxe et offrant des mécanismes de validation et de raisonnement en adéquation avec la nature et la complexité des systèmes à modéliser". Le deuxième défi commun à l'IHM et au SI est donc **la qualité d'un langage de modélisation**.

Dans les deux cas (qualité des modèles et qualité des langages), l'un des verrous est la capacité à **évaluer les propositions**. Si le Génie Logiciel s'ouvre au domaine des expérimentations avec l'essor du "Génie Logiciel Empirique", il a encore besoin de développer des approches efficaces pour conduire des études empiriques et définir des métriques servant à évaluer langages et modèles. Ce point de l'évaluation est déterminant pour permettre à des experts d'un domaine de créer et/ou de modifier leurs propres modèles. Les langages et modèles doivent donc être étudiés du point de vue des utilisateurs de langages et de modèles et non pas seulement des outils.

On peut remarquer que parmi les verrous présentés, la gestion des préoccupations est directement liée au fait que nous considérions les domaines de l'IHM et des SI lors de la conception. Elle nécessite donc une solution commune qui permettent des pratiques de modélisation conjointes. Les autres verrous (l'évaluation de la qualité des langages et des modèles et la suppression de la dichotomie entre conception et exécution) sont les mêmes pour nos deux communautés.

2.3.2 Objectifs et Contributions

L'exploration du premier défi (la diversité des préoccupations) doit permettre de lier de manière harmonieuse les spécificités des domaines de l'IHM et des SI. J'ai donc cherché à faire le pont entre les modèles de l'IHM et ceux des SI. En complément des approches d'architecture logicielle, nous avons opté pour une démarche méthodologique d'une part et d'autre part, nous avons mis en place des langage et outil, inspirés de la programmation par aspects, pour gérer la connexion entre les points de vue IHM et SI.

Néanmoins pour ne pas limiter les concepteurs à nos solutions, nous avons cherché à leur faciliter le travail de choix des processus, des modèles et des outils à utiliser pour concevoir leur système. Notre solution vise à offrir un guidage dont le but est de permettre aux concepteurs de concevoir un environnement de modélisation adapté à leurs besoins. Même si le besoin est issu du couplage entre IHM et SI, cette problématique et notre solution ne sont pas spécifiques à ces domaines. Aussi nous les présentons comme une contribution pour la gestion de modèles. Mais contrairement à l'IDM, nous n'abordons pas les problèmes liés à la gestion de modèles spécifiquement d'un point de vue technique, mais plutôt vis-à-vis des utilisateurs qui dans le cas des méthodes de développement sont des concepteurs, et de leur organisation. Ainsi la conception d'un environnement de modélisation n'est pas étudiée du point de vue de l'intégration d'outils existants, mais de celui des concepteurs et de leurs méthodes de travail dans le but de les guider. Les autres verrous liés à la modélisation ont été abordés de manière similaire.

Concernant le défi de l'expert-user modelling, la première étape me semble être de favoriser la compréhension du système par les experts du domaine ou les utilisateurs finaux : l'hypothèse est qu'en comprenant mieux, ils devraient être capables d'envisager avec plus de facilité les évolutions futures. Dans mon travail, la compréhension s'appuie sur des explications du système fournies par des modèles. Aussi j'ai travaillé le concept d' **auto-explication d'un système basée sur les modèles** qui fait référence au fait qu'un système puisse expliquer son fonctionnement par des modèles sans intervention humaine. J'aborde cette question aussi bien dans le domaine de l'IHM que dans celui des systèmes d'information, mais en considérant une approche commune qui cherchent à mettre en avant la spécificité de chacun des domaines. Dans le cadre de l'IHM, des modèles fournissent des explications pour des utilisateurs finaux alors que dans le cadre des SI, les modèles doivent servir à abstraire une représentation d'une organisation complexe. Nous suivrons donc des chemins parallèles entre IHM et SI...

Le dernier verrou qui est identique pour nos deux communautés est celui de l'évaluation de la qualité de langages. Dans ce cadre, plus qu'une approche commune, j'envisage un chemin commun basé sur la capitalisation collaborative des connaissances des concepteurs et des évaluateurs. Il s'agit de proposer des guides méthodologiques, issus des pratiques des concepteurs et évaluateurs, basés sur des protocoles expérimentaux et/ou des métriques pour évaluer un langage ou un modèle. Aussi je propose 1) la mise en place d'un outil de capitalisation de connaissances collaboratifs qui permet de collecter les bonnes pratiques d'évaluation des langages et des modèles, 2) l'intégration de métriques de mesure de qualité au sein d'un environnement de (méta-)modélisation. Ce travail débute, mais il me semble prometteur pour aboutir à des langages et des modèles de qualité.

2.3.3 Résumé des verrous et Contributions

Défi	Verrous	Contributions	Caractéristiques
Diversité des préoccupations	- Cohérence entre IHM et SI - Traçabilité entre niveaux d'abstraction	- processus de conception - langage pour la connexion - intergiciel de gestion de la connexion	- Problème de mise en commun - Solution conjointe
Expert-User Modelling	- Dichotomie entre modèles et code - Qualité des modèles	- Auto-exlication des IHM - Auto-explications des chorégraphies de services	- Problème commun - Solution sprécifique
Gestion des modèles	- Guidage pour la conception - Evaluation de la Qualité des langages et des modèles	- approche pour la réutilisation de processus et d'outils de modélisation - capitalisation collaborative des connaissances - outil d'évaluation de métriques de qualité	- Problème commun - Solution commune et indépendante d'un domaine

Les chapitres suivants présentent nos contributions pour chacun des défis : tout d'abord la connection des préoccupations des deux domaines, puis un premier pas vers la modélisation par les experts d'un domaine

grâce à l'auto-explication, enfin nos contributions en termes de pratiques et outils pour la gestion de modèles. Ce dernier chapitre regroupe nos propositions pour la conception d'environnements de modélisation et celles pour la qualité des langages et des modèles, qui sont des problématiques communes abordées conjointement aux deux domaines.

3 Gestion des préoccupations de l'IHM et de l'espace métier ou le problème de mise en commun

---

Les hommes construisent trop de murs et pas assez de ponts.

Isaac Newton

Sommaire

Ce premier chapitre traite de la gestion des préoccupations entre l'IHM et l'espace métier à différents niveaux d'abstraction. Il explique comment nous avons fait le pont entre les deux domaines au niveau conceptuel et technique.

3.1 Contexte

Tout système interactif est composé d'un noyau fonctionnel, appelé aussi espace métier, et d'une interface utilisateur. Le problème de pont entre les deux espaces a été abordé de nombreuses fois, soit du point de vue architectural (Coutaz 1987, Krasner et Pope 1988), soit d'un point de vue processus (Barthet et Tarby 1996, Sousa et Furtado 2003, Constantine et al. 2003, Fox et al. 2008) avec parfois un lien explicite entre les modèles des deux domaines (Corlett 2000, Nunes et e Cunha 2001). Mais aucun de ces travaux ne traite de la conception de systèmes d'information ubiquitaires pouvant avoir des interfaces complexes, par exemple tangibles ou adaptables au contexte, du point de vue des modèles. Ainsi si de nombreux UIDL existent (cf. section 2.2.1), la question de leur utilisation conjointe avec des langages tels qu'UML pour représenter le noyau fonctionnel n'est que rarement posée.

Nous nous sommes intéressés à ce lien pour les systèmes de réalité mixte qui sont des systèmes interactifs combinant mondes réel et virtuel (Dubois et al. 1999). Ce terme engloble aussi bien les systèmes de réalité augmentée qui superposent au monde réel des informations du monde virtuel, que des interfaces tangibles où le monde virtuel est manipulé à partir d'objets du monde réel. Des langages spécifiques aux systèmes de réalité mixte traitent généralement du problème de l'organisation des dispositifs et des flux de données entre ces dispositifs. Les plus courants comptent les langages ASUR (Dubois 2009), IRVO (Chalon et David 2004) ou le modèle de l'interaction mixte de C. Coutrix (Coutrix et Nigay 2006). Ces langages souvent peu formalisés sont utilisés indépendamment de la conception du noyau fonctionnel. Seul, ASUR, a été intégré dans un processus de conception de l'IHM (Dubois 2009) qui ne couvre pas le noyau fonctionnel. Le guidage pour la conception des systèmes de réalité mixte est donc faible soit pour la conception de l'interaction, soit pour son lien avec l'espace métier.

La complexité que nous voulons aborder est donc double. Il s'agit de guider les concepteurs dans la conception d'interfaces complexes, tout en faisant le pont avec la conception de l'espace métier. Ainsi nous avons proposé une méthode de conception, Symphony augmentée, qui étudie à la fois un processus de conception coordonné entre IHM et noyau fonctionnel et un langage de modélisation de l'espace interactionnel qui complète les travaux précédents. Nous détaillons dans les sections suivantes, ces propositions ainsi que leurs outils de support.

3.2 Approche

En prenant en compte les composants d'une méthode (à savoir les produits, les processus et les outils (cf. section 1.1)), nous avons cherché à savoir comment construire une méthode de conception pour les systèmes mixtes. On constate, à travers la littérature (Gulliksen et al. 2005, Sousa et Furtado 2003, Constantine et al. 2003, Fox et al. 2008), que de nombreuses méthodes intégrant à la fois les dimensions métier et IHM étendent des démarches ou méthodes existantes en Génie Logiciel. Nous adoptons ce point de vue, qui permet de profiter d'une méthode formalisée, utilisée et

maîtrisée, dont on connaît les forces et les faiblesses. Sur ce noyau orienté métier (dont on conserve au mieux les processus et modèles) doivent se greffer de manière cohérente les préoccupations, modèles et pratiques de l'IHM.

La méthode Symphony

Le noyau de notre méthode de développement pour les systèmes mixtes est la méthode Symphony qui est une méthode de développement itérative, incrémentale, basée sur l'utilisation de composants dès les premières phases du processus. Ainsi Symphony s'appuie sur des composants conceptuels tripartites, appelés Objets Métier, qui doivent permettre d'améliorer la réutilisation des composants et un couplage faible entre ces derniers.

D'un point de vue processus, Symphony prône un découpage organisationnel de la réalisation d'un système suivant les processus métier. Suite à l'étude préalable, les processus métier sont prioritisés et répartis au sein des itérations. Chaque itération suit sur un cycle en Y qui est connu pour être bien adapté pour les projets utilisant des nouvelles technologies car il permet de lever les incertitudes techniques caractéristiques de ce type de projet. Il nous a donc paru pertinent pour les systèmes de réalité mixte où les incertitudes techniques peuvent être importantes.

Extension de Symphony

Nous avons étendu Symphony pour y inclure les pratiques de conception des systèmes de réalité mixte. Les extensions couvrent toutes les branches de la méthode initiale afin d'aboutir à une méthode de développement complète, mais doivent respecter certains principes (Dupuy-Chessa et al. 2007, Godet-Bar et al. 2007a; 2010) :

- Ne pas bouleverser les pratiques des acteurs des processus de développement, en particulier en laissant aux spécialistes de l'IHM leurs outils et modèles,
- Prévoir des activités collaboratives de conception permettant aux spécialistes IHM, aux ergonomes et aux spécialistes du noyau fonctionnel de se synchroniser sur des objectifs communs ou des modèles communs,
- Garantir une traçabilité et une cohérence entre les modèles utilisés par les différents spécialistes intervenant dans la méthode,
- Fournir des outils de support à la méthode, depuis les guides méthodologiques jusqu'à la génération du code. De plus, ces outils doivent permettre la mise en œuvre des principes mentionnés ci-dessus.

A partir de ces principes, nous avons réalisé Symphony étendue avec les hypothèses suivantes : 1) le processus facilite la collaboration entre les acteurs des domaines de l'IHM et du génie logiciel ; 2) il permet aux concepteurs de produire des modèles cohérents ; 3) il permet d'assurer une certaine qualité du code produit.

Nous ne nous attarderons pas ici sur la traçabilité des modèles au sein de chaque domaine, qui nous avons formalisé pour l'évolution des Objets Symphony au sein du processus de Symphony étendue. Sachez néanmoins que cette traçabilité a été gérée par des transformations de modèles

et des vérifications de cohérence avec la mise en œuvre de techniques de l'IDM.

Dans la suite de ce chapitre, nous présentons un processus de conception collaboratif qui intégre les pratiques du génie logiciel et de l'IHM pour garantir les deux premiers principes. Ensuite, nous verrons les modèles communs issus des activités collaboratives, faisant ainsi le pont entre espaces métier et interactionnel. Nous présentons ici les résultats actuels de ces travaux qui ont été adaptés suite à des évaluations présentées en section 3.5.

3.3 Processus de Symphony Etendue

3.3.1 Présentation globale

Le processus de Symphony étendue pour le développement des systèmes de réalité mixte s'appuie sur un cycle de vie en Y (Fig. 3.1). Ce cycle s'applique itérativement à chaque processus métier du système en cours de développement (Dupuy-Chessa et al. 2009) :

- La branche gauche (fonctionnelle) correspond à la traditionnelle tâche de modélisation du domaine et des besoins, indépendamment des aspects techniques. Pour la conception des systèmes de réalité mixte, cette branche a été étendue avec un processus proposé par (Renevier 2004). Elle inclut des scenarii d'interaction, une analyse de la tâche, le choix des modalités d'interaction représentées avec des modèles ASUR par exemple, et des maquettes. Nous respectons donc bien notre premier principe qui est la prise en compte des pratiques de l'IHM en utilisant un processus et des modèles existants. Néanmoins en analyse, cette branche se termine par la structuration des espaces métier et de l'interaction avec des Objets tripartites de Symphony que nous avons étendus pour l'IHM (cf. section suivante). Si le modèle d'Objets Symphony ne correspond pas à un modèle existant pour l'IHM, il est, a priori, proche des connaissances des informaticiens et permet une forte cohérence entre espace métier et espace interactionnel.
- La branche technique (droite) permet aux concepteurs d'étudier les architectures techniques et applicatives. Elle combine aussi toutes les contraintes et tous les choix techniques en relation avec la sécurité, l'intégration à l'existant, etc. Nous avons limité notre travail pour les systèmes de réalité mixte aux choix techniques les concernant c'est-à-dire le choix des dispositifs d'interaction et le choix de l'architecture globale.
- La branche centrale intègre les branches fonctionnelle et technique en un modèle de conception qui réunit le modèle d'analyse et l'architecture technique. Elle montre comment les différents éléments du futur système sont structurés et distribués sur les divers dispositifs.

Nous ne détaillerons pas plus le processus de Symphony Etendue dont une description plus complète peut être trouvée dans (Dupuy-Chessa et al. 2009) ou dans (Godet-Bar 2009) pour une version actualisée de la branche gauche. Par la suite, nous nous concentrons sur ce qui fait l'originalité de

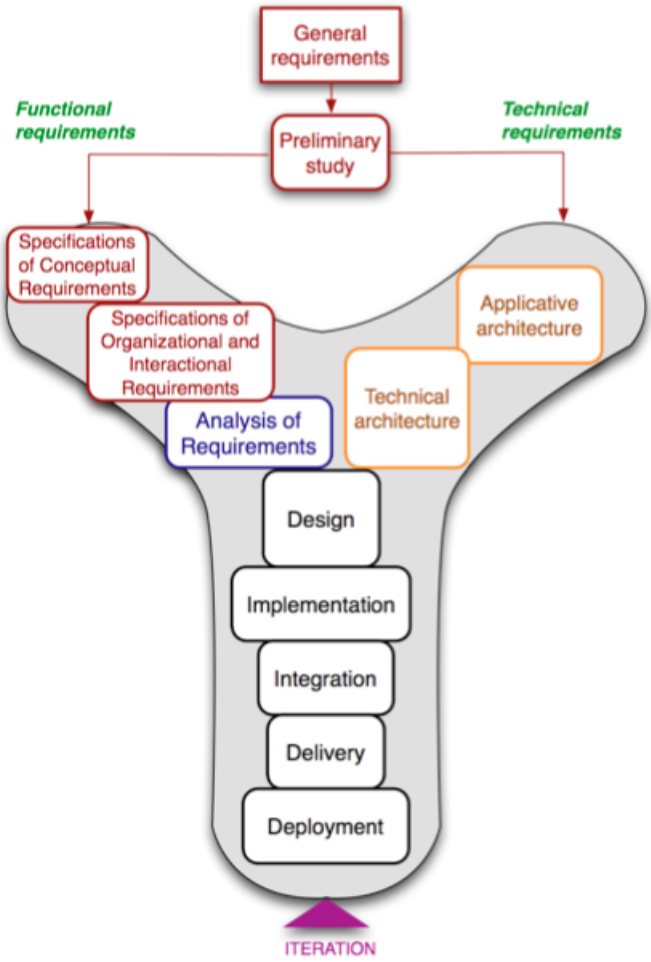


FIGURE 3.1 – *Cycle de Symphony étendue issu de (Dupuy-Chessa et al. 2009)*

ce travail à savoir la gestion des activités collaboratives en nous basant sur la dernière version de Symphony étendue.

3.3.2 Activités collaboratives

Coopérations

Les activités collaboratives de Symphony Etendue ont lieu entre des spécialistes du GL, qui étudient le noyau fonctionnel, ceux de l'IHM et les ergonomes qui analysent l'IHM. Ces collaborations ne sont pas forcément médiées par la machine, contrairement aux travaux sur les collecticiels qui sont des outils permettant un travail collectif, collaboratif et à distance. Elles sont donc potentiellement beaucoup plus larges et les collecticiels ne peuvent être qu'un moyen de résoudre certains problèmes pratiques de collaborations. Nous proposons donc plutôt de reprendre l'une des taxonomies des collaborations proposées par Grebici et al. (Grebici et al. 2005). On distingue deux types de collaboration élémentaires, les coopérations et les coordinations. Nous définissons une coordination comme une activité collaborative qui met en œuvre une décomposition du travail en activités, la description du planning associé, l'affectation d'acteurs à ces activités, un objectif propre étant fixé à chaque acteur ou activité. Une coopération

se définit comme une activité collaborative qui met en œuvre une action commune des différents acteurs partageant, au moins partiellement, un objectif commun et coordonnant leurs actions de manière autonome, sans processus prédéfini.

Après avoir utilisé ces deux types de collaborations pour Symphony Etendue (Godet-Bar et al. 2007a, Dupuy-Chessa et al. 2009), nous avons finalement opté pour n'avoir que des coopérations. Ce choix a été dicté par les résultats d'une expérimentation (Dupuy-Chessa et al. 2011) que nous décrivons par la suite, mais aussi par le fait que toutes les activités du processus ont maintenant pour vocation de produire un artefact utilisé lorsque nécessaire comme support aux activités de collaborations entre acteurs.

L'étape du processus de Symphony qui illustre le mieux les activités collaboratives est la phase de spécification organisationnelle et interactionnelle des besoins (Fig. 3.1) qui doit déterminer le "qui fait quoi et quand". Le but est de considérer les choix organisationnels en détaillant comment les acteurs internes au processus métier travaillent. Nous avons étendu cette phase (Fig. 3.2) pour y inclure la spécification de l'interaction, à partir du choix de style d'interaction effectué durant la phase de "Spécification Conceptuelle des Besoins". Elle débute par une activité commune de décomposition des processus métier. Puis les spécialistes de chaque domaine modélisent leurs préoccupations ensemble ou séparément. Par exemple, pour l'espace d'interaction, l'ergonome et le spécialiste IHM spécifient les besoins en termes d'interaction. C'est là qu'ils utilisent les arbres de tâches, les modèles ASUR et des maquettes. A la fin de cette coopération, le spécialiste IHM décrit l'espace d'interaction sous forme d'Objets tripartites Symphony. Ces Objets serviront de base à l'activité commune ("coopération") de connexion entre espaces métier et interactionnel avec le spécialiste GL. Toutes les activités de collaboration sont bien des activités où les acteurs travaillent ensemble à produire des artefacts communs. De plus, on peut noter que ces activités communes débutent et terminent la phase, ce qui permet des points de synchronisation forts, mais qui ne contraignent pas le travail de chacun durant la phase.

Influence de l'IHM sur l'espace métier

Un des résultats intéressants de l'intégration des pratiques de conception des systèmes mixtes dans un cycle logiciel a été de montrer comment des choix d'interaction adoptés par l'ergonome et le spécialiste IHM peuvent déclencher une évolution de l'espace métier (Godet-Bar et al. 2008; 2010). Cette évolution peut concerner tous les artefacts produits au cours de l'étude des besoins fonctionnels (Fig. 3.3).

Par exemple, un système de réalité augmentée gère souvent des données topologiques pour positionner l'utilisateur ou des éléments dans un environnement en trois dimensions superposé au monde réel. Si on considère que l'espace d'interaction a besoin de gérer le plan 3D d'un logement, ces données peuvent servir 1) à augmenter les connaissances du métier pour prendre en compte, par exemple, les plans d'architecte d'un logement, 2) à proposer de nouvelles activités comme l'annotation d'un dommage constaté lors d'un état des lieux par des notes (textuelles ou vo-

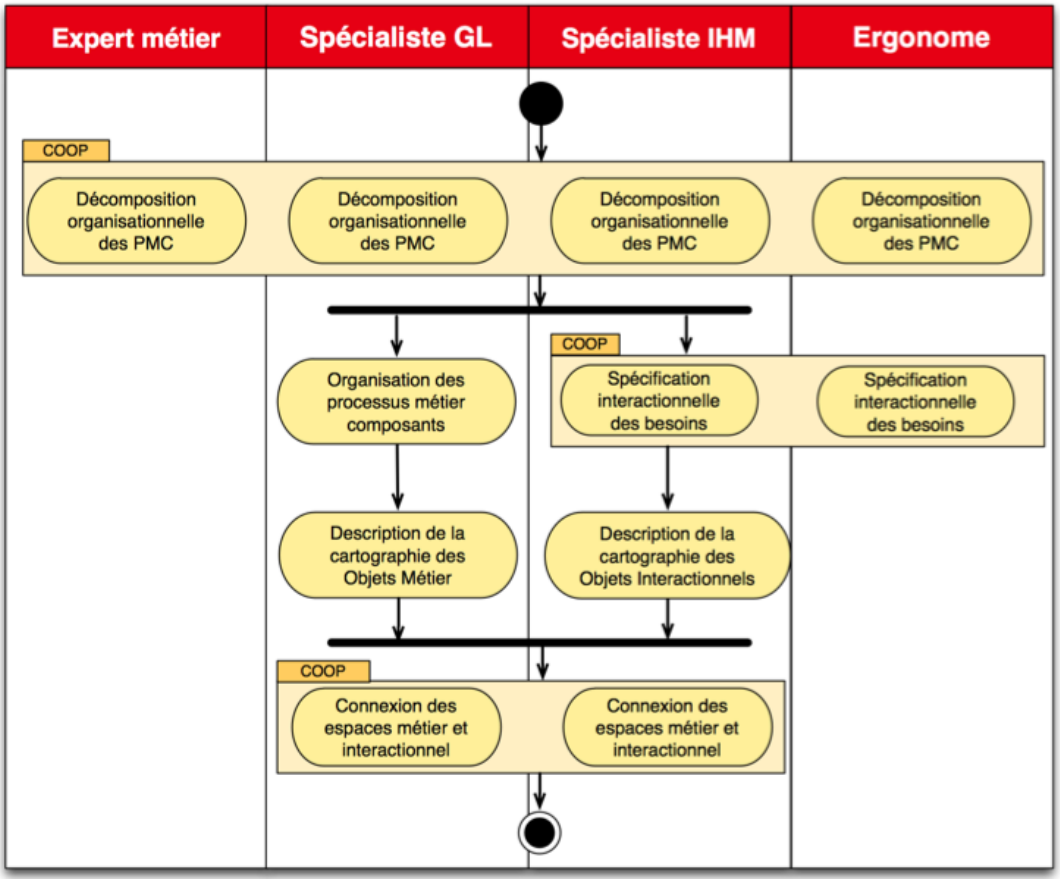


FIGURE 3.2 – *Phase de spécification organisationnelle et interactionnelle des besoins issue de (Godet-Bar 2009).*

cales), 3) à réaliser de nouveaux cas d'utilisation comme par exemple, une estimation d'un dommage basé sur son emplacement et ses dimensions réels, 4) à proposer de nouveaux processus tels que des visites virtuelles de logements. Ces évolutions montrent l'intérêt d'une conception coordonnée entre espaces métier et interaction. Toutefois, le processus doit être suffisament détaillé pour être utilisable par les acteurs de la méthode.

3.3.3 Outillage

Pour décrire le processus de Symphony étendue de manière précise, nous avons choisi de le spécifier sous forme de patrons. Dans ce cas, les patrons correspondent pour la plupart à des fragments de méthode de développement proposés dans une étape de développement et réutilisables par d'autres processus. C'est à ce titre que l'on peut les qualifier de patrons de processus. L'ensemble des patrons de processus d'une méthode forme ainsi un guide méthodologique pour la mise en œuvre de la méthode.

Par exemple, le cycle proposé en Fig. 3.1 représente la solution préconisée par le patron racine de Symphony Etendue "Cycle de Développement Symphony augmentée". Chacune des phases du cycle peut elle-même être décrite par un patron, créant ainsi un système de patrons inter-reliés. Suivant ce principe, la branche gauche de Symphony Etendue a été décrite par Guillaume Godet-Bar par une vingtaine de patrons, rendus accessibles

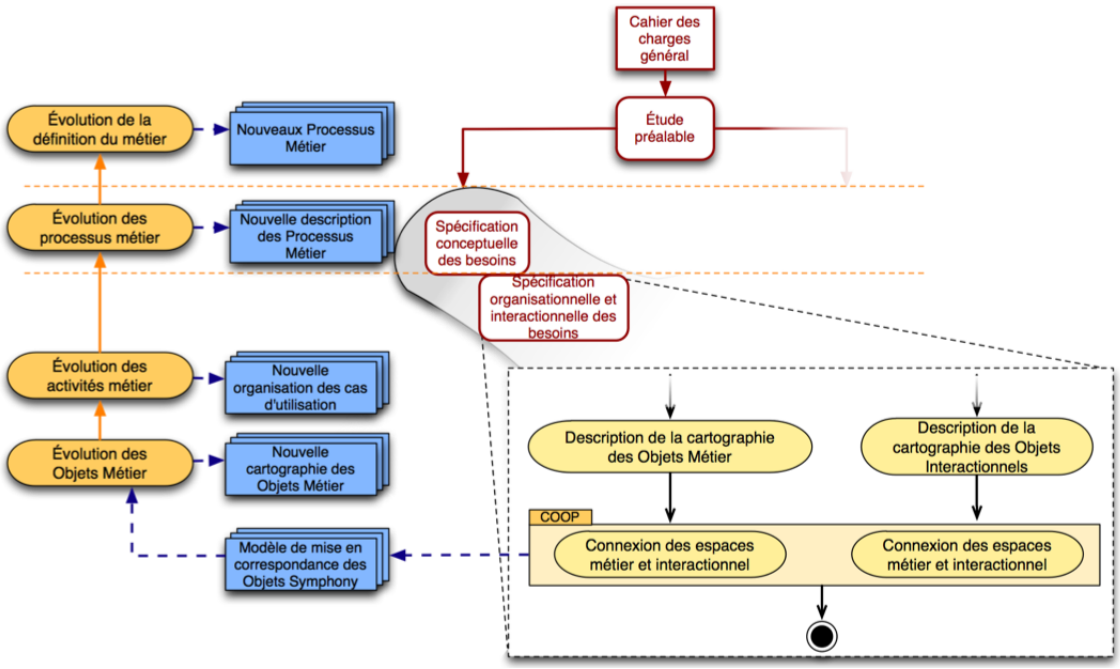


FIGURE 3.3 – *Cycle de Symphony issu de (Godet-Bar 2009)*

sous la forme d'un site web à l'adresse suivante :
http ://iihm.imag.fr/godetg/PatternSystem/index.html

3.4 Les modèles pour la collaboration

Les modèles communs sont des éléments importants pour favoriser la collaboration entre les différents domaines. Ce sont eux qui apportent la vision commune du système. Nous abordons ici seulement les modèles qui supportent le travail collaboratif entre les différents membres de l'équipe de développement, en laissant volontairement de côté ceux qui servent à communiquer avec les parties prenantes.

3.4.1 Modèles communs

Nos premiers modèles communs sont des modèles informels, qui servent à capturer les besoins de haut niveau lors des phases d'étude préalable et de spécification conceptuelle des besoins. Ce sont des maquettes et des scenarii tels que décrits par Carroll (Carroll 1997) ou les cas d'utilisation essentiels de Constantine (Constantine et al. 2003). Les scenarii ont un rôle particulièrement important puisqu'ils ont été identifiés comme un point de jonction privilégié entre les domaines du GL et de l'IHM (Sutcliffe 2005) et qu'ils sont ensuite affinés dans des modèles plus formels tels que les diagrammes de séquences.

Cependant ces modèles sont de haut niveau et ne sont pas adaptés à une mise en correspondance des concepts de l'IHM et de l'espace métier losque l'analyse des différents acteurs a été menée. Aussi nous introduisons les Objets Symphony, qui sont identifiés dans chacun des domaines avant d'être connectés puis affinés.

3.4.2 Objets Symphony

Les Objets Symphony se structurent en Objets Métier pour le noyau fonctionnel et en Objets Interactionnels pour la partie IHM. Nous décrivons dans cette section, ces deux types d'Objets ainsi que la façon de les lier.

Objet Métier

Les Objets Métier de Symphony (Hassine et al. 2002) se divisent en trois types : les Objets Métier Processus, qui permettent de décrire un processus applicatif ; les Objets Métier Entité, qui matérialisent les concepts applicatifs ; les Objets "Données de référence" servant à la nomenclature. Si nous considérons un exemple fonctionnellement simple d'une application sur table augmentée, qui consiste à afficher sur des cercles concentriques les membres d'une équipe de recherche, l'espace métier décrit l'organisation de l'équipe de recherche et fournit des méthodes pour ajouter/enlever ou modifier les membres. Ainsi il existe deux Objets Métier Entité "Team" et "Member" et un Objet Métier Processus "ManageTeamMembers" (cf. Fig. 3.4).

Les Objets sont décrits comme des paquetages tri-partites. La partie gauche du paquetage décrit les services fournis par l'objet, en utilisant une classe "Interface". La partie centrale du paquetage décrit l'implémentation de ces services (classe "Master") ainsi que l'éventuelle sous-division en concepts complémentaires (classes "Part") qui n'est pas illustrée sur la Fig. 3.4. Enfin la partie droite (classes "Role") décrit les services requis par l'objet pour garantir son comportement. Les objets sont liés par des relations de dépendance appelées "use" qui ne peuvent exister qu'entre un rôle et un objet. Ici elle permet de préciser que le processus «ManageTeamMembers» fait appel au concept de «Team» qui est lui-même lié à celui de "Member" (toujours par une relation "use" entre le rôle d'un objet et l'interface de son objet lié).

La structure tri-partite des Objets doit permettre d'avoir des Objets indépendants assurant une modularité forte. De plus la logique applicative doit se concentrer sur les Objets Métier Processus, permettant ainsi aux Objets Métier Entité d'être plus facilement réutilisables.

Objet Interactionnel

Nous avons suivi une structuration identique pour l'espace d'interaction : ainsi nous avons proposé le concept d'Objet Interactionnel (OI) (Godet-Bar et al. 2007b). Par symétrie avec l'espace métier, l'espace interactionnel est constitué d'Objets Interactionnels Processus, d'Objets Interactionnels Entité et d'Objets Interactionnels Données de référence (Godet-Bar 2009) :

- Les OI «Entité» décrivent les concepts centraux de l'espace interactionnel ; ils représentent des concepts ayant un lien fort avec un ou plusieurs concepts du domaine de l'application. Par exemple, des avatars, sous forme de photos, qui représentent les membres de l'équipe ou des boîtes de propriétés qui apportent des informa-

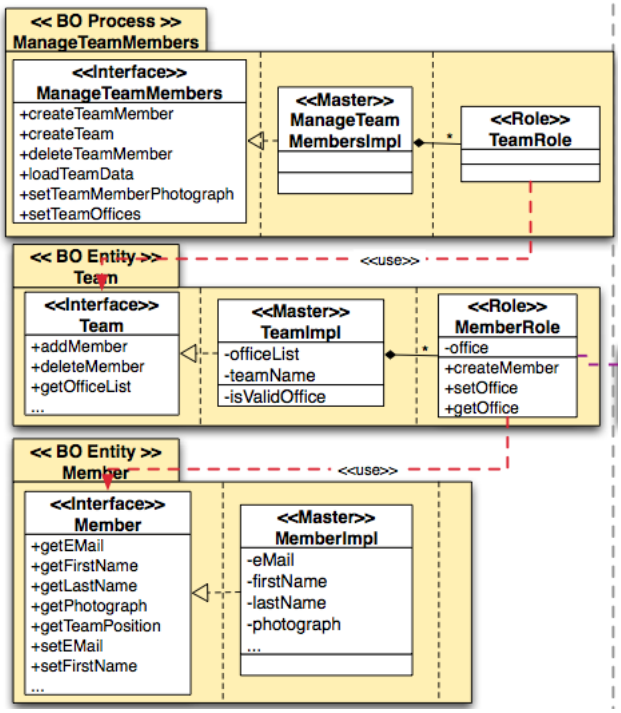


FIGURE 3.4 – *Exemple d'Objets Métier issu de (Godet-Bar et al. soumis)*

tions supplémentaires sur ces membres sont des Objets Interactionnels Entité.

- Les OI « Processus » prennent en charge la mise en œuvre des règles applicatives de gestion des Objets Interactionnels Entité et de construction de l'espace d'interaction, indépendamment des règles métier. Comme pour les Objets métier, la dépendance entre Objet Interactionnel Processus et Objet Interactionnel Entité est matérialisée par une relation d'utilisation ("use"). Dans l'exemple simple de l'affichage des membres d'une équipe par cercles concentriques, il existe un Objet Interactionnel Processus "ManageAvatarScene" qui doit, en particulier, gérer l'affichage des différents membres sans que les représentations (avatars) de ces derniers ne se chevauchent (Fig. 3.5).
- Les OI «Données de référence» représentant les données de base manipulées au sein de l'espace interactionnel (par exemple, des codes de couleur).

Les Objets Interactionnels ont été conçus en complément des modèles pour la modélisation des systèmes de réalité mixte, en proposant un point de vue plus plus centré sur les informaticiens habitués à la notion d'objet et de composant. Ils peuvent être utilisés dans d'autres cadres que celui de la réalité mixte. Toutefois ils peuvent paraître complexes pour des interfaces simples de type formulaires.

Par rapport au référenciel Caméléon, les Objets Interactionnels peuvent être considérés comme des AUI. Même s'ils ne donnent pas de détails sur les espaces de travail et leur réification, ils permettent toutefois de structurer l'espace interactionnel (OI Entité) et les enchaînements au sein de celui-ci (OI Processus). Les CUI ne sont pas modélisés en tant que tels, mais leurs informations sont contenues dans des maquettes.

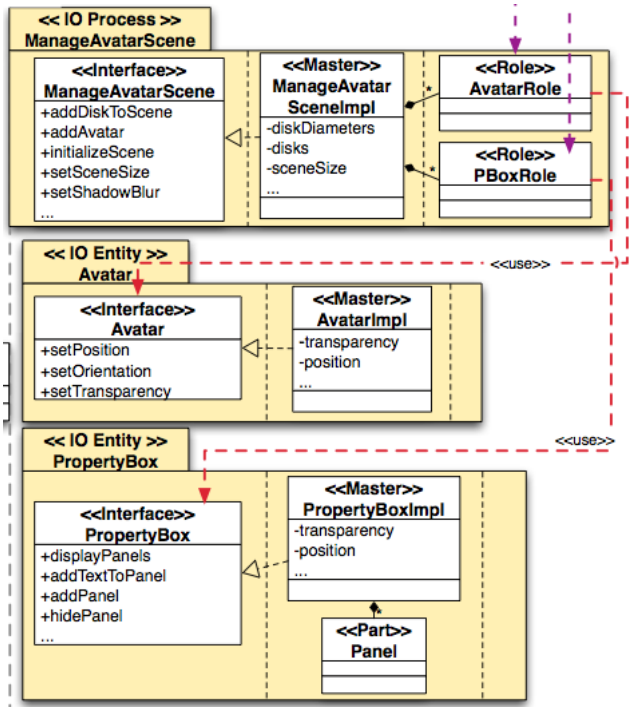


FIGURE 3.5 – *Exemple d'Objets Interactionnels issu de (Godet-Bar et al. soumis)*

Correspondance entre espace d'interaction et espace métier

L'un des intérêts des Objets Interactionnels est la mise en correspondance des espaces métier et interactionnel, à l'aide de la relation « Représente ». Cette relation est ensuite raffinée sous la forme d'une classe de contrôle nommée « Translation ». De par son rôle de médiateur entre fonctionnalités et interface, cette classe partage des caractéristiques avec la facette « Contrôle » d'un agent PAC (Coutaz 1987). Les liens architecturaux sont donc explicites au niveau des modèles. Ainsi nous avons trois espaces conceptuels distincts (Fig. 3.6) : le métier sur la gauche, l'interaction sur la droite et la traduction au milieu. Les Objets Processus de chaque espace gère leurs Objets Entitiés. La traduction répercute les modifications entre les espaces métier et interactionnel.

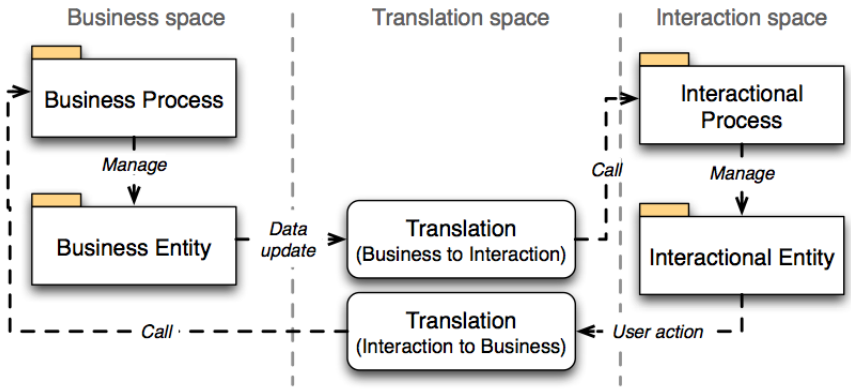


FIGURE 3.6 – *Architecture Symphony issu de (Godet-Bar et al. soumis)*

Pour l'exemple de l'affichage des membres d'une équipe, une classe "MemberTranslation" est ajoutée pour exprimer qu'un avatar et une boîte de propriétés sont des représentations dans l'espace interactionnel d'un membre de l'espace métier (Fig. 3.7).

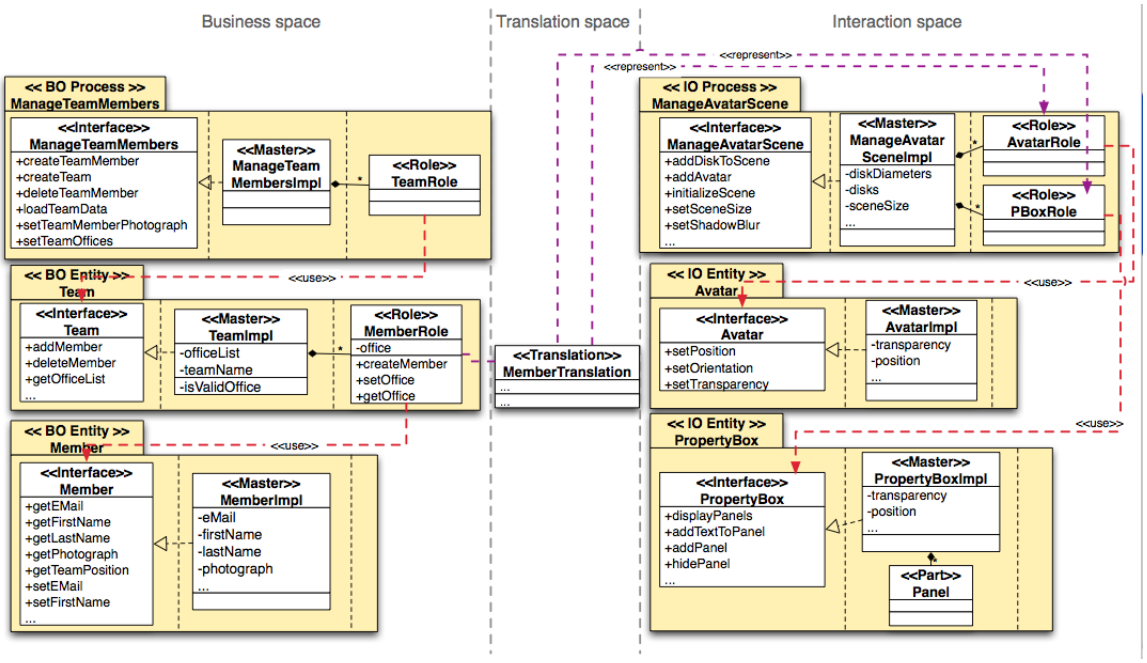


FIGURE 3.7 – *Exemple de "Translation" issu de (Godet-Bar et al. soumis)*

Outre l'existence structurelle de cet Objet "Translation", nous avons introduit des concepts pour exprimer son comportement qui traduit le lien sémantique entre les espaces métier et interactionnel. Par exemple, la création d'un nouveau membre doit déclencher la création d'un avatar dans l'espace interactionnel.

La première étape consiste à définir quel type d'événement peut déclencher une modification (ici la création d'un membre). Nous avons adopté une grammaire simple inspirée du paradigme de la Programmation par Aspects (Kiczales et al. 1997) pour représenter les caractéristiques d'un événement de connexion, nommé "Symphony Advice". La description de l'Objet Métier (resp. Interactionnel) qui a un impact sur l'espace d'interaction (resp. métier) est alors annoté par un "advice" (fig. 3.8). Dans l' "advice", la notion "after"/"before" correspond, à l'instar des concepts de la programmation par aspects, au fait de réaliser la traduction respectivement après ou avant l'exécution de la méthode sur laquelle se branche la translation. Ici l'exécution de la méthode "createMember" de l'Objet Interactionnel Entité "MemberRole" déclenche un événement de connexion identifié par le nom "createAvatarFromMember". "createAvatarFromMember" correspond au nom de la méthode qui effectuera, depuis la classe "MemberTranslation", la traduction du membre vers son avatar.

Une fois qu'un événement de connexion est identifié, il reste à décrire la réaction à cet événement c'est-à-dire les opérations effectuées par la méthode déclenchée. La figure 3.9 décrit le comportement de la méthode "createAvatarFromMember". Quand un Objet Métier "Member" est créé (à travers son rôle "MemberRole"), un nouvel Objet Interactionnel "Avatar"

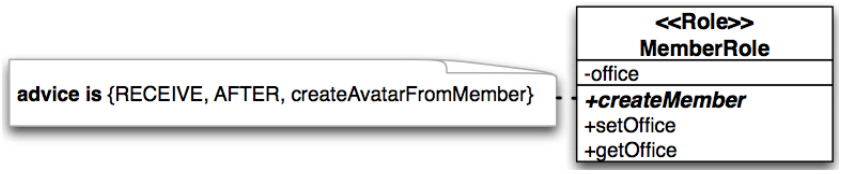


FIGURE 3.8 – *Exemple d' "Advice" issu de (Godet-Bar et al. soumis)*

est instancié pour représenter le nouveau membre dans l'espace interactionnel. La méthode de création d'un avatar ("addAvatar") est gérée par le Objet Interactionnel Processus "ManageAvatarScene". Cet Objet Processus doit maintenir des règles telles que le fait que les Objets Avatar et Boîte de Propriétés doivent se déplacer de concert. Toute translation passe par les objets processus afin de garantir le respect des processus métier ou interactionnel, qu'un appel direct aux Objets Entité ne pourrait pas maintenir avec facilité (comme par exemple un déplacement de concert de deux objets). Le comportement de la translation (et donc l'appel à "addAvatar") n'est donc pas précisé dans l'advice.

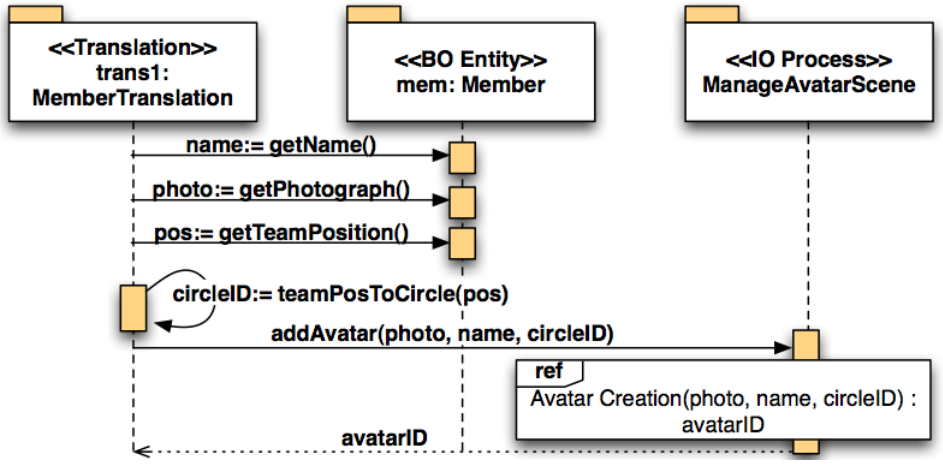


FIGURE 3.9 – *Translation "createAvatarFromMember" issue de (Godet-Bar et al. soumis)*

Les Objets "Translation" sont les seuls éléments à gérer des liens entre les espaces d'interaction et métier. Il n'existe pas de relation directe entre les deux espaces conceptuels, ce qui nous paraît être une solution en adéquation avec les architectures logicielles telles que PAC. En complément de ces architectures, nous représentons la dynamique des enchaînements entre les espaces d'interaction et métier.

De plus, la description en Objets Symphony n'est pas qu'un point de vue technique sur la structuration de l'application. C'est le moyen de mettre en cohérence la vision des spécialistes de l'IHM et du noyau fonctionnel. Les Translations sont réalisées de manière conjointe par des informaticiens spécialistes en IHM ou en GL : d'abord ébauchées lors de la phase de Spécification Organisationnelle et Interactionnelle des besoins, elles sont affinées pendant l'Analyse des besoins avant d'être adaptées pour représenter la structure de l'application lors de la Conception. Les Objets de la spécification ne sont pas détaillés : ce ne sont pas des objets tri-partites, mais seulement des concepts liés entre eux par des relations

d'utilisation ou de translation. En Analyse, les Objets sont décrits au niveau de détails des exemples illustrant cette section. Enfin ils sont affinés pour prendre en compte les technologies choisies dans la branche droite de Symphony, comme par exemple l'intergiciel Sonata décrit dans la section suivante.

3.4.3 Sonata : un intergiciel entre IHM et métier

La conception puis l'implémentation d'applications structurée en Objets Symphony suit des règles systématiques. Pour cette raison, nous avons développé deux profils UML qui décrivent les concepts des Objets Symphony, aux niveaux spécification et analyse, et un profil pour annoter les diagrammes de classes UML avec des "advices". Cette formalisation du langage permet d'automatiser une part importante du cycle de développement et de l'implémentation.

L'automatisation du processus a été abordée dans la section 3.3.3. Je présente ici Sonata (Godet-Bar et al. soumis), un intergiciel qui facilite la structuration en Objets Symphony au niveau du code.

Introduction à Sonata

La structuration forte des Objets Symphony permet de proposer l'intergiciel Sonata (Godet-Bar et al. soumis). Sonata permet de connecter aisément les "briques" logiques des applications structurées en Objets Symphony, tout en facilitant leur réutilisation. Le code d'implémentation de l'intergiciel est disponible en ligne à l'adresse :
http ://github.com/ggodet-bar/Sonata/

Afin de réaliser cet objectif, Sonata s'appuie pour l'essentiel sur le principe d'utilisation de conventions, concernant la structuration, le nommage ou l'organisation des objets, en lieu et place des traditionnelles configurations au cas par cas. L'élaboration de solutions perd ainsi une certaine souplesse mais gagne en lisibilité et efficience. Cette approche s'applique naturellement aux Objets Symphony, dont la structure interne et l'organisation sont systématiques.

Sonata a été implémenté en Java en incluant des mécanismes de la programmation par aspects pour gérer les connexions entre Objets. Je ne décris pas ici dans le détail tous les éléments de Sonata. Je me contente d'introduire ceux gérant les Objets Symphony et leurs connexions.

Objets Symphony en Sonata

Certains concepts des Objets Symphony sont purement conceptuels et ne correspondent à aucune fonctionnalité. Par exemple, les notions d'Objets Métier ou Interactionnel sont purement organisationnelles.

Mais six autres concepts ont été opérationnalisés et sont gérés par Sonata : 1) les Objets Processus ; 2) les Objets Entités ; 3) les "Roles" des Objets ; 4) les classes "Translation" ; 5) les "Advices" et 6) les connexions Symphony. Les 5 premiers éléments correspondent aux concepts que nous avons présentés dans la section précédente. Le dernier, les connexions Symphony, rassemble les Objets Entité et les classes "Translation" dans

une description de tous les éléments responsables d'une association donnée entre un ou plusieurs Objets Symphony sources et un Objet cible médiée par une classe "Translation".

Ces constructions sont intégrées à l'intergiciel Sonata comme des classes abstraites Java, des interfaces et/ou gérées par classes utilitaires. Par exemple, les Objets Symphony, Entité ou Processus, sont des interfaces de marquage. Les interfaces de marquage sont des classes abstraites vides (elles ne déclarent ni méthodes ni attributs), dont la fonction est de permettre la classification des classes les implémentant. Les interfaces de marquage reconnues par Sonata permettent également à l'intergiciel de tisser des services récurrents au sein des classes implémentant ces interfaces. Ce mécanisme est décrit en Fig. 3.10, dans le cas de la conception d'un Objet Symphony Entité. Le fait d'implémenter l'interface "EntityObject" permet à Sonata de tisser dans l'objet "Master" des services d'identification (deux méthodes, void setID(int id) et int getID(), encapsulant un attribut privé int id). De même, le fait d'implémenter l'interface SymphonyRole permet à Sonata de tisser dans chaque objet "Role" des services identifiant les instances d'Objets Symphony collaborateurs.

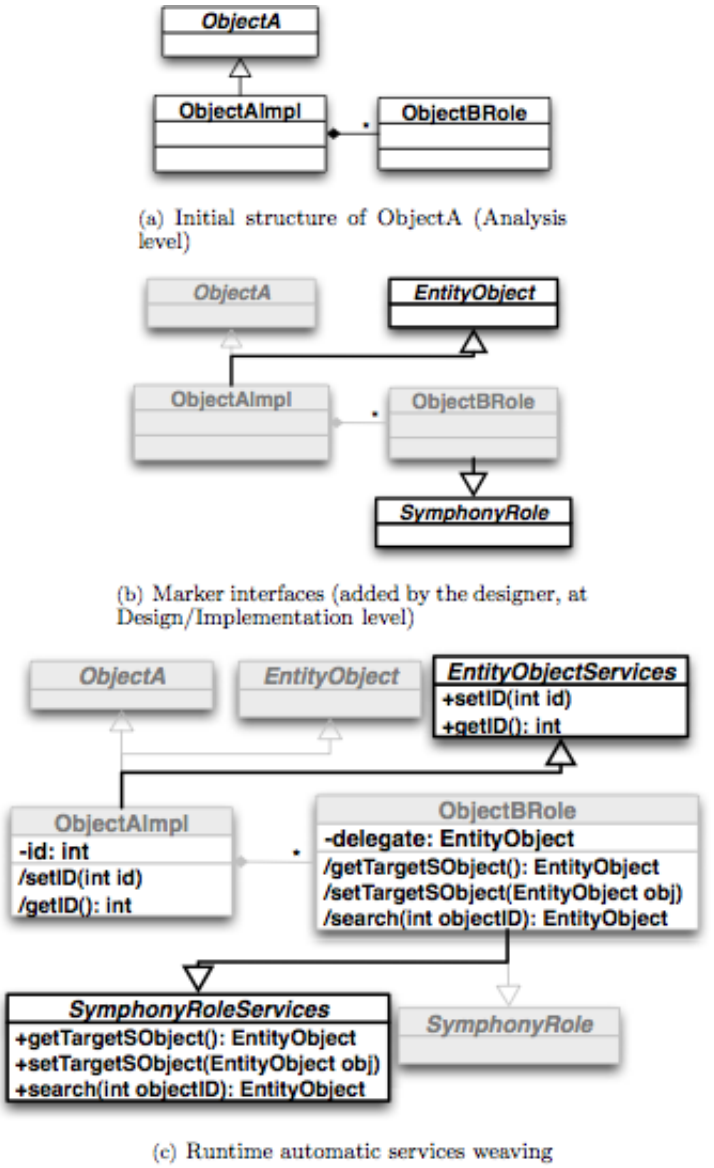
Connexion entre espaces dans Sonata

Sonata gère aussi la connexion entre espaces métier et d'interaction, représentée au niveau conceptuel par des "Translations". Considérant que les connexions doivent être intégrées transversalement au reste du code fonctionnel, l'approche de la programmation orientée aspects semble adaptée.

Dans la section précédente, un mécanisme d'annotation ("Advice") permettant d'associer la méthode d'une classe Translation à un appel de méthode déclenché dans l'espace métier ou interaction a été présenté. Cette formalisation est utilisée pour générer automatiquement les aspects permettant de lier les Objets Symphony et Translation. Ces classes, générées automatiquement, n'ont pas vocation à être modifiées par les développeurs.

Il reste à coder la traduction en elle-même. Une classe abstraite "ConectionTranslation" peut être implémentée à cet effet. Par exemple, la classe "MemberTranslation" implémente "ConectionTranslation" pour gérer toutes les traductions entre les instances de l'Objet Métier "Member" et celles de l'Objet Interactionnel "Avatar". Le code de cette classe est le seul élément de toute l'application où les implémentations de "Member" et de "Avatar" sont appelées toutes les deux, réduisant ainsi le couplage entre les espaces métier et d'interaction à un minimum acceptable.

Reprenons le comportement décrit en Fig. 3.9 qui décrivait la translation "createAvatarFromMember" de la classe "MemberTranslation" déclenchée après un appel de la méthode "createMember" de la classe "MemberRole". La Fig. 3.11 résume les principaux événements de l'exécution quand cette méthode "createMemberRole" est appelée. La partie du diagramme de séquence qui est située au dessus du cadre pointillé correspond à l'ensemble des appels déclenchés dans l'espace métier pour céer un nouveau membre. Les appels à l'intérieur de l'encadré en pointillé sont tissés à l'exécution ; ils correspondent au déclenchement de l'événement de connexion. Une demande est envoyée à l'"Invoker" qui est une



(a) Initial structure of ObjectA (Analysis level)

(b) Marker interfaces (added by the designer, at Design/Implementation level)

(c) Runtime automatic services weaving

FIGURE 3.10 – *Etapes du tissage de services au sein des Objets Symphony (Godet-Bar et al. soumis)*

des classes de l'intergiciel Sonata créée pour gérer les connexions. L' "Invoker" crée et appelle une instance de la classe "MemberTranslation" pour créer un Objet Interactionnel "Avatar".

Sonata offre donc un support au codage de la connexion entre Objets Métier et Objets Interactionnels. Le code des Objets Symphony n'est jamais influencé par celui de l'autre domaine. Tout le couplage est mis en œuvre par des "Translations", gérées par des mécanismes spécifiques à Sonata.

3.5 Evaluations

Nous avons cherché à évaluer la méthode Symphony Etendue (son modèle de processus et les Objets Symphony) ainsi que Sonata. Notons que ces évaluations sont partielles et qu'elles ne peuvent prétendre à des ré-

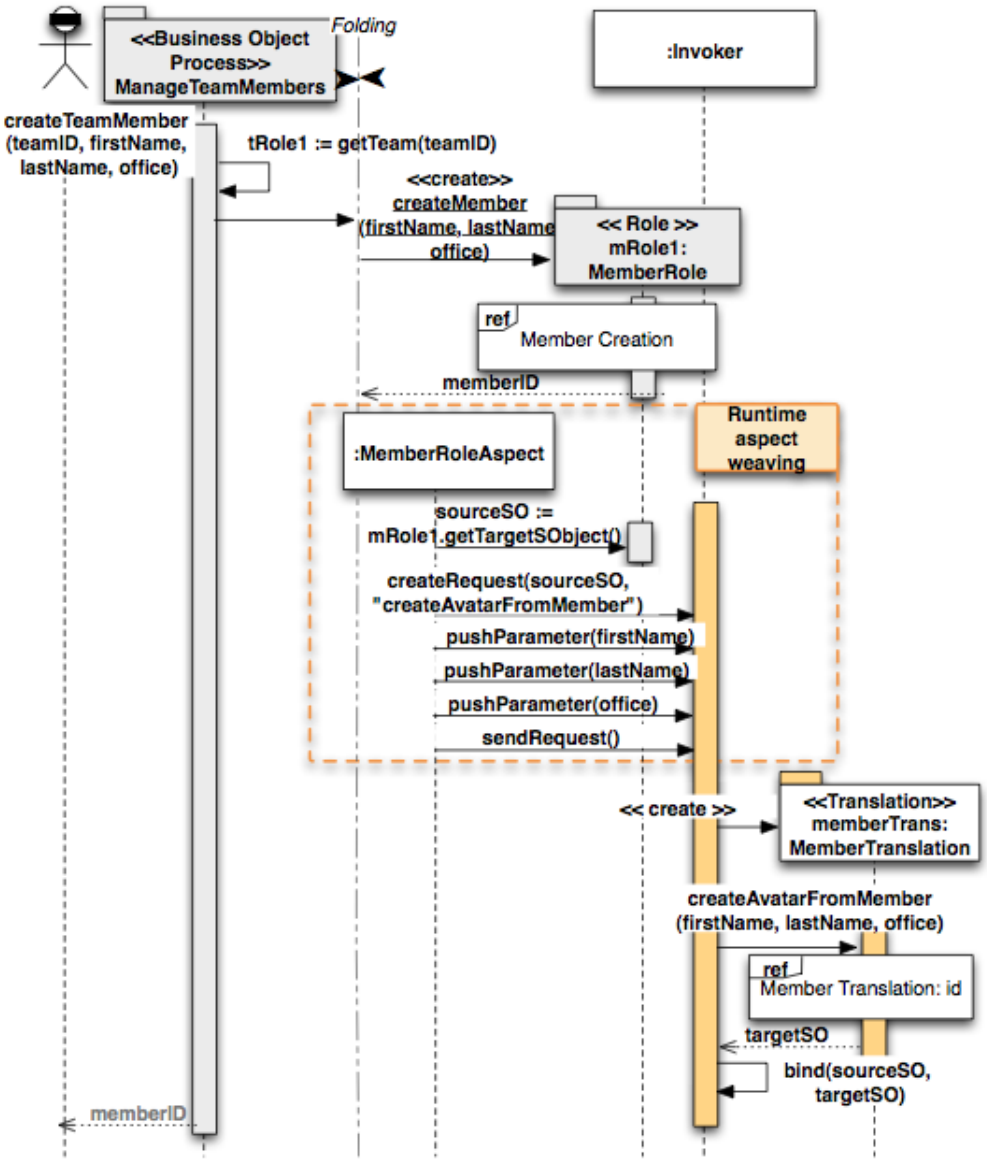


FIGURE 3.11 – *Tissage à l'exécution réalisé par Sonata (Godet-Bar et al. soumis)*

sultats quantitatifs significatifs. Elles nous ont néanmoins permis de faire évoluer nos propositions.

3.5.1 Etudes de cas

Tout d'abord, nous avons évalué le processus et les modèles de produits de Symphony étendue. L'étude d'un modèle de processus n'étant pas suffisante pour juger de la convenance des activités, nous avons cherché à récolter des informations sur l'utilisation de la méthode. Nous avons donc opté pour une approche qualitative, qui a pour but de parvenir à une compréhension fine du sujet étudié. Ainsi des échantillons petits mais ciblés sont utilisés. Les informations collectées sont nombreuses et souvent révélatrices de cas plus généraux même si leurs conclusions se limitent aux cas étudiés.

Dans un premier temps, nous avons réalisé 3 études de cas utilisant Symphony étendue. Bien que le nombre de ces études ainsi que les intervenants (les concepteurs de la méthode) amènent des biais certains, ces cas d'études ont permis de montrer la couverture du modèle de processus et des Objets Symphony :

- la première était un état des lieux augmenté pour lequel l'IHM était un système de réalité augmentée. Elle avait pour caractéristiques d'avoir un noyau fonctionnel assez complexe et une interface en réalité augmentée.
- la deuxième s'appuyait sur une interface tangible pour visualiser les membres d'une équipe et leurs relations (cf section précédente). Elle avait un noyau fonctionnel simple, mais devait modéliser un type de système de réalité mixte non encore traité avec les Objets Symphony (i.e. une interface tangible).
- la troisième était une interface graphique pour la visualisation de tests de sécurité pour les aéroports. Ici la logique aplicative et l'interface graphique sont relativement complexes.

Ensuite nous avons tenté de lever le biais des intervenants en testant Symphony étendue auprès d'autres concepteurs spécialistes en IHM ou en Génie Logiciel. Ce cas d'étude a été réalisé avec l'aide de Nadine Mandran, ingénieur CNRS au LIG et spécialiste en expérimentations. Il avait pour objectif de recueillir des commentaires concernant deux de nos hypothèses de conception de Symphony étendue (cf section 3.2) : 1) le processus facilite la collaboration entre les acteurs des domaines de l'IHM et du génie logiciel ; 2) il permet aux concepteurs de produire des modèles cohérents. Une description détaillée de l'expérimenation a été publiée dans (Dupuy-Chessa et al. 2011). Je n'en présente ici qu'un résumé.

L'expérience a été menée auprès de 4 binômes de concepteurs (un spécialiste IHM et un spécialiste GL) pendant une semaine. Les concepteurs avaient pour consignes de suivre le processus de Symphony étendue pour concevoir un système interactif. Le modèle de processus se limitait aux phases les plus collaboratives c'est-à-dire les deux premières phases de Spécification Conceptuelle des besoins et de Spécification Organisationnelle et Interactionnelle des besoins dans leur version présentée dans (Dupuy-Chessa 2009). A chaque séance de travail, seul ou en binôme, les concepteurs devaient noter leurs activités, leur durée, les buts, leurs résultats et les problèmes éventuels. La séance de travail qui consistait à metre en commun les Objets Métier et Interactionnel a été réalisée lors d'une séance filmée afin de mieux maîtriser les résultats de cette activité collaborative clé.

La méthode a été perçue comme intéressante et safisfaisante. La collaboration est apparue utile à 5 concepteurs sur 7 (un des concepteurs était absent lors de la clôture de l'expérimentation). Elle a été citée comme l'un des éléments qui réduisent les erreurs grâce à une meilleure compréhension entre les spécialistes de différents domaines. L'un des points positifs notés est la séparation des préoccupations entre IHM et métier. Les participants ont aussi apprécié la vision commune, qui est facilitée par l'utilisation de modèles communs.

Ainsi la majorité des participants (6/7) pense que la méthode peut rendre la conception plus efficace en terme de temps grâce à la vision

commune donnée par les scenarii et grâce à l'utilisation de niveaux d'abstraction appropriés. Toutefois un des spécialistes en IHM a pointé la nécessité de vouloir travailler de manière conjointe.

Le principal problème du processus est qu'il peut être long. Des remarques suggèrent d'adapter la méthode à la taille du projet, en omettant certaines étapes d'analyse du métier ou des utilisateurs lorsque le projet est une application simple destinée à un seul utilisateur, par exemple. De plus, la durée du projet et des échanges collaboratifs varie beaucoup d'un groupe à l'autre. En particulier, les collaborations avaient parfois lieu pour des objectifs inappropriés c'est-à-dire que des concepteurs travaillaient ensemble pour réaliser les modèles de l'un des domaines au lieu de travailler séparément.

Notre deuxième hypothèse portait sur la cohérence des modèles produits. Au terme de l'expérience, seuls deux groupes avaient utilisé les Objets Symphony. Un groupe n'avait pas suffisament avancé en termes de conception, l'autre groupe était divisé : le spécialiste IHM avait utilisé les Objets Interactionnels alors que le spécialiste GL avait voulu conserver des modèles UML standards. Un point que nous n'avions pas envisagé est donc la résistance au changement dans les pratiques de modélisation. Pour les 2 autres groupes qui ont utilisé des Objets Symphony, ils ont été amenés à effectuer des ajustements (ajout/suppression/renommage d'objets) afin d'obtenir un modèle d'Objets Symphony cohérent entre l'espace interactionnel et l'espace métier. 3 des 4 spécialistes interrogés (le dernier était absent lors de l'interview final) considèrent que les Objets Symphony sont "un pont en Génie Logiciel et IHM, un bon point de synchronisation" (Dupuy-Chessa et al. 2011).

Ce retour d'expérience nous a amenés à simplifier le processus 1) pour ne conserver que les activités qui produisent des modèles indispensables à la suite du processus, les autres activités devenant optionnelles ; 2) pour n'inclure que des coopérations dont le but est plus évident (la production d'un modèle commun). C'est ce modèle de processus simplifié qui a été présenté dans la thèse de Guillaume Godet-Bar (Godet-Bar 2009) et dans ce mémoire.

On peut néanmoins noter la difficulté d'évaluer des modèles de processus ou de produits. Si les évaluations qualitatives nous apportent des informations intéressantes sur le processus, elles n'ont pas de valeur statistiques et peuvent difficiliement être reproduites. Une évaluation exhaustive de la méthode aurait nécessité de comparer le développement à l'aide de Symphony étendue avec d'autres méthodes de développement. La mise en œuvre d'une telle expérience aurait toutefois été longue et complexe. Il est aussi difficilement envisageable d'adopter une approche quantitative qui serait chronophage et pas nécessairement plus prolifique en retours d'utilisation.

3.5.2 Evaluation technique

Nous avons aussi cherché à évaluer l'intérêt de la méthode Symphony étendue d'un point de vue technique : la structuration en termes d'Objets Symphony obtenue à l'issue de l'utilisation de la méthode favorise-t-elle la qualité des systèmes en résultant ? (Céret et al. 2010) présente nos résul-

tats à cette question. Nous avons réalisé 3 versions de la même application modélisée avec des Objets Symphony : la première avec une architecture MVC standard, la deuxième avec un MVC amélioré (Eckstein 2007), la dernière avec Sonata. Nous avons évalué le code de ces 3 applications à travers de métriques. Nous nous sommes concentrés sur le couplage entre composants et la complexité avec l'hypothèse qu'étant donné la même couverture fonctionnelle et la même interface, les variations de ces propriétés reflèteraient la qualité de la structuration du code ou de l'intergiciel. Les résultats ont montré que : le couplage était faible entre les Objets dans toutes les implémentations, ce qui tend à montrer leur modularité ; si un couplage existe entre 2 Objets Symphony, il est concentré sur la relation de dépendance entre les classes Role et Interface des objets collaborant ; en corollaire, l'instabilité des Objets, c'est-à-dire la fragilité qu'un Objet génère au sein d'un système, en tant qu'entité à la fois requise par d'autres mais aussi dépendante d'autres, se concentre sur les classes Role ; la structuration n'empêche pas de produire du code complexe, néanmoins en respectant les bonnes pratiques de programmation, la complexité des composants est plus qu'acceptable puisqu'elle donne des résultats en dessous des seuils fixés pour l'expérimentation.

Bien que la méthode Symphony étendue n'ait été évaluée que de manière partielle, nous pouvons néanmoins en tirer des résultats encourageants : elle parait remplir ses objectifs premiers à savoir faciliter le pont entre GL et IHM pour des systèmes de réalité mixte, tant au niveau conceptuel que technique.

3.6 Apports et Perspectives

3.6.1 Contributions

Les travaux présentés dans ce chapitre montrent l'intérêt de coupler les points de vue complémentaires de l'IHM et des SI. Notre principale contribution sur ce sujet est la proposition d'une méthode de développement pour les SI, pouvant avoir des interfaces graphiques mais aussi des interfaces de réalité mixte. Cette méthode est "complète" dans la mesure où elle propose un modèle de processus qui s'appuie sur des modèles de produits et qui est outillé par un guide méthodologique mais aussi par un intergiciel.

Le modèle de processus démontre l'avantage d'une conception conjointe de l'IHM et du noyau fonctionnel en mettant en évidence les évolutions fonctionnelles qui peuvent être envisagées en considérant de nouvelles techniques d'interaction.

Les produits s'appuient sur des langages spécifiques à chacun des domaines afin de conserver les habitudes de travail des concepteurs, mais aussi sur des langages communs afin de favoriser la communication. Parmi ces langages, nous avons proposé les Objets Interactionnels en complément des Objets Métier de Symphony pour faciliter la mise en correspondance des espaces interactionnel et métier tant au niveau conceptuel qu'au niveau du code. Nous pensons que la structuration d'une application interactive en Objets Symphony apporte les avantages suivants :

– La conceptualisation du système sous forme d'Objets indépendants

et interconnectés encourage leur modularité et leur réutilisation ainsi que celle de leur spécification.
- Les dépendances entre les Objets à l'intérieur d'un même espace conceptuel sont faciles à gérer : chaque relation est médiée par une classe "Role" qui porte tout le poids du couplage entre deux Objets.
- Il y a un découplage fort entre les Objets Interactionnels et les Objets Métier : il n'existe pas de relation directe entre les deux espaces conceptuels ; il y a une dépendance indirecte, qui est gérée par des classes spécifiques, les "Translations" qui ne peuvent impacter que l'un ou l'autre des espaces (mais pas les deux).
- les classes "Translation" sont les seuls éléments du code à gérer la relation entre les espaces d'interaction et métier.

Enfin le codage est facilité par la mise à disposition d'un intergiciel, Sonata, qui permet de coupler les objets d'un système. De plus, en respectant les conventions de structuration, Sonata peut aussi être utilisé indépendamment de Symphony étendue.

La méthode Symphony étendue, ainsi que l'outil Sonata, sont matures et accessibles sur internet, en open source. A ce jour, plusieurs applications ont été développées à l'aide de la méthode dans un contexte académique. Elles ont permis de faire évoluer le processus pour une meilleure collaboration et une plus grande efficacité, tout en respectant les principes que nous nous étions fixés. A présent que la méthode Symphony étendue acquiert une forme de stabilité, il serait intéressant de la confronter à des développements en entreprise.

3.6.2 Perspectives

Les Objets Symphony et leur outil de support répondent à nos objectifs initiaux (cf. section 3.2). Néanmoins le nombre et la complexité des modèles à manipuler peuvent être un frein à l'adoption d'une telle approche. Il se pose alors la question de la complexité et donc de la qualité du langage des Objets Symphony, que nous avons seulement survolés dans nos évaluations. Nous avons pu noter la difficulté de réaliser de telles expérimentations, nous montrant ainsi le réel besoin de support à ces pratiques. Nous avons aussi constaté l'utilité non seulement, de métriques, qui permettent d'obtenir rapidement des éléments d'évaluation, mais aussi d'une approche qualitative lors de la réalisation de l'étude de cas avec la méthode Symphony.

Une autre limite à la méthode concerne la conception de l'IHM. Symphony étendue s'appuie sur un processus 1) qui manque de flexibilité même s'il comporte des chemins optionnels comme par exemple, celui pour concevoir un système de réalité mixte ; 2) qui ne traite pas tous les types d'interfaces dont en particulier les IHM adaptables. D'autres processus doivent être envisagés pour couvrir des pratiques telles que celles préconisées dans le cadre de référence Caméléon pour les interfaces adaptables. Dans ce cas, les Objets Interactionnels aussi pourraient nécessiter d'être plus flexibles, par exemple en y introduisant de la variabilité comme celà a déjà été réalisé pour les Objets Métier (Saidi et al. 2009).

3.7 Principaux résultats et encadrements

Principales publications

- G. Godet-bar, D. Rieu and S. Dupuy-Chessa, Towards the Integration of HCI Practices and Business Evolution in the Symphony Method, Information and Software Technology, Elsevier, vol. 5, num. 52, pages 492-505, 2010.
- Guillaume Godet-Bar, Sophie Dupuy-Chessa, et Dominique Rieu. Sonata : Flexible connections between interaction and business spaces. System and Software, en phase de 2ème révision.

Outils disponibles

- Site web décrivant Symphony :
  http ://iihm.imag.fr/godetg/PatternSystem/index.html
- Intergiciel Sonata :
  http ://github.com/ggodet-bar/Sonata/

Encadrements

- Guillaume Godet-Bar, Spécification et outillage d'une méthode de conception des systèmes de réalité mixte, thèse co-encadrée avec Dominique Rieu, 2006-2009.
- Eric Céret, Comparaison de plateformes pour l'implémentation de systèmes interactifs, stage de M2Pro co-encadré avec Guillaume-Godet-Bar, 2008.
- David Juras, stage de mémoire CNAM co-encadré avec Dominique Rieu, 2005-2006 (mémoire non soutenu).
- Jorge-Luis Pérez-Medina, Spécifications techniques de la démarche Symphony dans le cadre des systèmes de réalité augmentée, M2R co-encadré avec Dominique Rieu, 2006.
- Cyril Vachet, Spécifications et outillage d'une démarche de développement de systèmes mixtes, M2R co-encadré avec Dominique Rieu, 2005.

Projets

- projet local BQR INPG, 2006-2007
- projet local IMAG COCOVI (Conception Collaborative et Validation pour les systèmes Interactifs post-WIMP), 2006-2007

4 L'Auto-explication, une problématique commune gérée spécifiquement

Les grandes personnes ne comprennent jamais rien toutes seules, et c'est fatigant, pour les enfants, de toujours et toujours leur donner des explications.

Antoine de Saint-Exupéry

Sommaire

L'auto-explication est un problème commun à l'IHM et aux SI que déclinons dans chacun des domaines. Nous suivrons donc deux chemins parallèles : le premier en IHM pour répondre à d'éventuelles questions des utilisateurs sur les interfaces et le deuxième en SI pour comprendre l'exécution d'une chorégraphie de services.

4.1 Auto-Explication

Le terme "auto-explication" est issu du domaine de l'apprentissage. C'est une stratégie qui consiste pour l'apprenant à produire pour lui-même des explications de ce qu'il étudie. En apprentissage, l'hypothèse est que les apprenants qui produisent le plus d'auto-explications comprennent mieux l'objet d'étude que les autres.

Mon hypothèse est qu'un système auto-expliqué améliore la compréhension du système par ses utilisateurs. Cette compréhension accrue doit permettre une meilleure utilisation du système mais aussi favoriser ses évolutions.

L'auto-explication d'un système fait référence à la capacité de ce système à expliquer son fonctionenement sans intervention humaine. Ce concept est proche de celui de supervision qui est une technique de suivi et de pilotage. Mais contrairement à ce dernier qui surveille le bon fonctionnement d'un système ou d'une activité, l'auto-explication n'a pas de notion de "bon". L'objectif est simplement d'expliquer pour favoriser la compréhension.

En considérant que les modèles réalisés lors de la conception contiennent des informations utiles à la compréhension, je propose de les utiliser pour formuler les explications. Les modèles ne se limitent pas à la conception et peuvent aussi servir à l'exécution pour l'auto-explication. Il ne s'agit pas ici de rétro-ingénierie car les modèles utilisés pour expliquer le système ne sont pas issus du code, mais font partie de ceux réalisés lors du processus de conception. Il s'agit de fournir lors de l'exécution d'un système des explications issues des modèles auparavant limités à la documentation.

J'ai décliné cette approche en IHM et en SI, en espérant à terme pouvoir unifier les solutions. En IHM, l'auto-explication propose de fournir des réponses aux questions que peut se poser l'utilisateur face à une interface (Frey et al. 2010). En SI, l'objectif est d'expliquer le déroulement d'un processus métier (Cortes-Cornax 2011).

4.2 Auto-Explication des IHM

4.2.1 Contexte

L'auto-explication en IHM a pour objectif de améliorer l'utilisabilité d'un logiciel en traitant deux problèmes. Le premier provient du fait que les choix des concepteurs ne sont pas toujours évidents ou adaptés aux utilisations finales du logiciel. Le deuxième est relatif à la génération automatique d'interfaces qui aboutit à un défaut de qualité (Myers et al. 2000). Le palliatif envisagé est de permettre aux utilisateurs de disposer d'explications sur l'IHM pour qu'ils la comprennent mieux et qu'ils la manipulent mieux.

Ainsi l'auto-explication d'une IHM vise à fournir à l'utilisateur final des informations sur la justification de l'IHM (quel est son but ?), les choix de conception (pourquoi est-elle structurée dans ces espaces de travail ? quelle est l'utilité de ce bouton ? ...), son état courant (pourquoi ce menu est-il grisé ?) ainsi que l'évolution de son état (comment puis-je activer

cette fonctionnalité ?). C'est la problématique de la thèse d'Alfonso Garcia-Frey, qui a débuté en octobre 2009 encadrée avec Gaëlle Calvary. Ce travail a lieu dans le cadre du projet européen ITEA UsiXML [1] qui a pour objectif de définir, valider et standardiser un langage de description d'interfaces utilisateur (UIDL) qui permet de mettre en œuvre le concept "$\mu 7$" : multiterminal, multiutilisateur, multilingue/culture, multi-organisation, multicontexte, multimodalité et multiplateformes. Nos propositions pour l'auto-explication s'appuient sur ce langage que nous contribuons à définir.

4.2.2 Approche

Le langage UsiXML s'appuie sur le cadre de référence Caméléon (section 2.2.1). Ainsi le langage est constitué actuellement d'une dizaine de méta-modèles inter-connectés. UsiXML définit des méta-modèles de domaine et de tâches, mais aussi un méta-modèle **QOC** (Question, Options, Criteria (MacLean et al. 1991)) pour guider le processus de transformations à travers des options de conception. QOC documente les raisons justifiant un choix de conception : les Questions identifient les questions clé de la conception ; les Options fournissent les réponses possibles aux Questions et les Critères permettent l'évaluation et la comparaison des Options. Il est ainsi possible en UsiXML de décrire l'IHM avec des modèles, mais aussi de justifier les choix de conception. Nous allons nous appuyer sur les modèles de UsiXML pour fournir des explications aux utilisateurs.

Nous proposons de considérer les critères de QOC comme des critères de qualité. Les concepteurs d'IHM doivent définir leurs IHM en utilisant les modèles de UsiXML. Quand un choix de conception s'offre à eux, ils doivent l'exprimer sous la forme d'une question, d'options et de critères. Les options seront les alternatives qui pourront être proposées à l'utilisateur et les critères fourniront les justifications des différents choix en termes de qualité. La qualité peut être basée sur des recommandations, mais aussi sur des évaluations du système. Ainsi la Fig. 4.1 montre comment un concepteur peut avoir le choix entre 2 interacteurs différents pour la saisie d'une date et peut justifier chacune des options par des modèles de qualité, représentant ici des critères ergonomiques. Il reste à embarquer ces modèles à l'éxécution afin de pouvoir les utiliser pour répondre aux questions des utilisateurs.

Comme nous l'avons mentionné précédemment, le méta-modèle de QOC existe en UsiXML. Mais la notion de critère est trop limitée pour l'auto-explication. Par exemple, le concepteur doit pouvoir décrire le critère de tolérance aux fautes qu'il a choisi en expliquant d'où est issu ce critère (i.e. de la norme ISO9241-110) et comment il le mesure. Pour exprimer la complexité des critères de qualité qui peuvent justifier les options de conception, nous avons conçu un méta-modèle de qualité que nous décrivons par la suite.

1. http ://itea.defimedia.be/

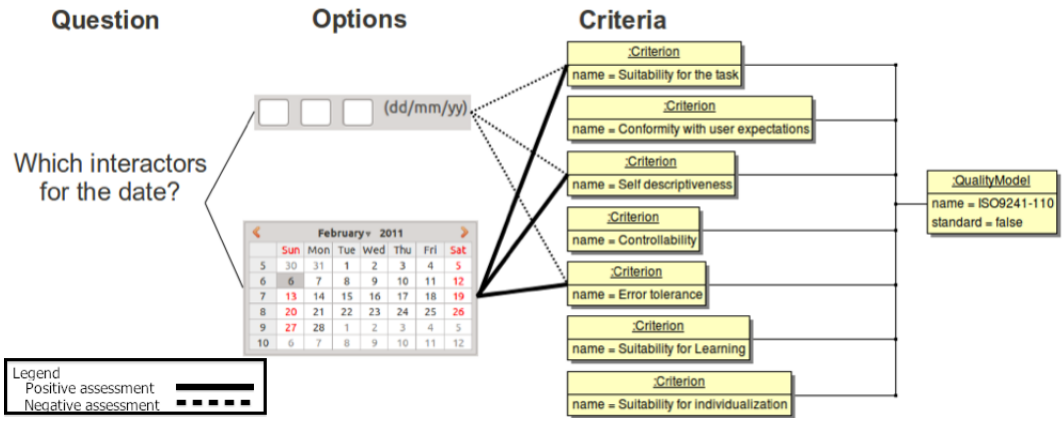


FIGURE 4.1 – *Exemple d'options de conception issu de (Frey et al. 2011b)*

4.2.3 QUIMERA - Méta-modèle de qualité

Différents méta-modèles et modèles de qualité existent dans des domaines divers (McCall 1977, ISO 2001, Mehmood et al. 2009, Mohagheghi et Aagedal 2007, Dromey 1996). Notre méta-modèle, qui nous avons baptisé QUIMERA, s'appuie sur ces différents travaux en tentant de couvrir la qualité de n'importe quel type de produit, l'évaluation de la qualité et ses différentes perspectives (Frey et al. 2011b).

Perspectives de Qualité QUIMERA (Fig. 4.2) a été conçu pour couvrir les 4 perspectives de qualité présentées dans (Carlier 2006) :

- La qualité attendue est la qualité dont le client a besoin. Elle est définie à travers la spécification du système étudié.
- La qualité souhaitée est le degré de qualité que l'expert de qualité désire atteindre pour la version finale du système. Elle est dérivée de la qualité attendue.
- La qualité obtenue est celle réalisée par une implémentation donnée du système. Elle est issue de la qualité souhaitée.
- La qualité perçue correspond à la perception de la qualité par le client une fois que le système a été livré.

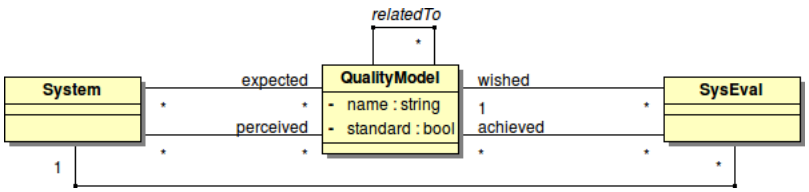


FIGURE 4.2 – *Quatre perspectives de qualité dans QUIMERA (Frey et al. 2011b)*

Dans notre cas, l'entité "System" représente le produit à étudier. "SysEval" est une représentation d'une évaluation particulière de ce produit. On peut noter l'attribut "Standard" qui dénote le fait que le modèle considéré soit la représentation d'un standard tel que ISO9146 ou les critères ergonomiques de Bastien et Scapin (Bastien et Scapin 1993). Dans ce cas, toutes les parties du méta-modèle ne seront pas instanciées : par exemple, un standard est défini dans l'absolu ; il n'est pas associé à un système

particulier et n'a pas d'évaluation. Il peut être ré-utilisé en définissant un nouveau modèle de qualité qui lui sera lié par l'association "RelatedTo" et qui définira les parties non instanciées du standard pour le système à évaluer.

La Figure 4.3 montre le méta-modèle de qualité en détail, mais sans les perspectives de qualité. Un modèle de qualité est composé de critères. Un critère est nommé et comme pour un patron de conception, il est caractérisé par la description du problème qu'il traite et son contexte d'application (par exemple, interfaces web). Les critères peuvent être récursivement associés à d'autres critères au travers de la classe "CriterionAssociation". Par exemple, le critère de "Gestion des erreurs" de (Bastien et Scapin 1993) a pour sous-critère "Protection contre les Erreurs". Des recommandations spécifiques peuvent être spécifiées pour chaque critère. Une recommandation est une affirmation qui caractérise le critère. Elle peut porter sur n'importe quel élément de modèle UsiXML ou du code, que nous nommons artefact. Par exemple, la protection contre les erreurs a pour recommandations de griser les commandes non disponibles, de fournir la liste de valeurs possibles ...

Des évaluations peuvent être spécifiées (classe "EvaluationMethod") par des métriques et/ou des pratiques (des patrons ou anti-patrons / de processus ou de produit). Dans le cas de métriques, la mesure est donnée par un résultat numérique ("NumericalResult) qui peut être compris entre des limites spécifiées par le concepteur. Pour les pratiques, le résultat a une valeur booléenne indiquant si la pratique est respectée ou non. Par exemple, on peut chercher à évaluer si (oui ou non) la liste des valeurs possibles est bien proposée. Les métriques et les pratiques sont évaluées directement sur des artefacts à travers des recommandations.

Le concepteur peut ajuster l'importance de chaque critère de qualité pour le système étudié en donnant des poids à chaque recommandation. Il est ensuite possible de réaliser de la qualité courante du système. Quand un résultat ne satisfait pas les attentes du concepteur, le concepteur pourra opérer une transformation ou un ensemble de transformations qui auront été enregistrées pour réaliser une variante (de meilleure qualité) du système.

Validation Nous avons instantié QUIMERA sur deux études de cas, issus de domaine différents (Frey et al. 2011a). La première étude de cas est relative au domaine premier du méta-modèle, i.e. l'utilisabilité. Ainsi les critères ergonomiques de Bastien et Scapin (Bastien et Scapin 1993) ont été définis dans une instance de QUIMERA. La deuxième étude de cas a été réalisée par E. Céret qui n'avait pas participé à la conception de QUIMERA. Elle traduit au sein du méta-modèle l'évaluation de la qualité du code qui avait été réalisée lors de l'évaluation des applications implémentées avec les Objets Symphony, décrits en section 3.5 (Céret et al. 2010). Le modèle résultant comporte 39 classes dont certaines représentant les résultats obtenus par des outils de calcul de métriques logicielles pour l'évalation du code. Ces études de cas semblent montrer que QUIMERA est bien adapté pour décrire les différentes facettes de la qualité d'un système.

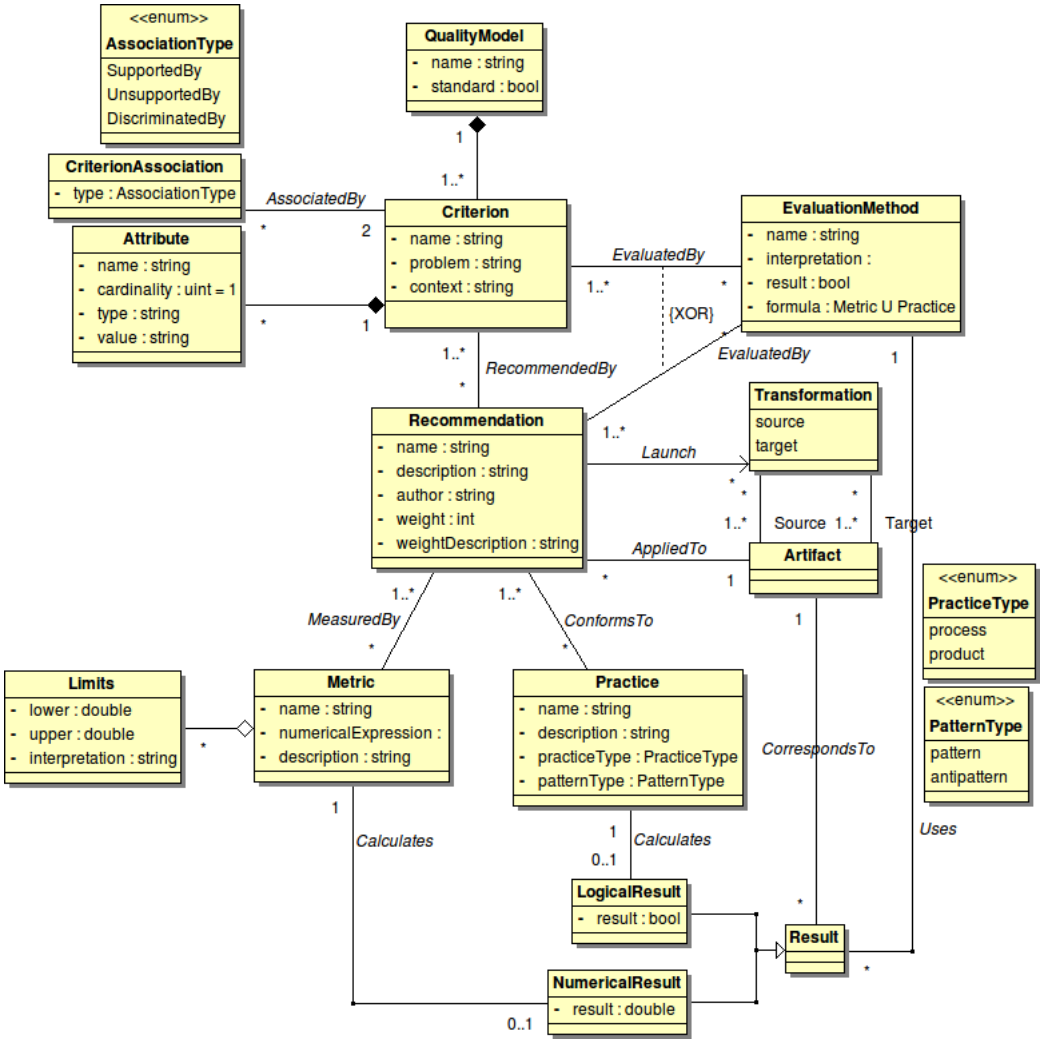


FIGURE 4.3 – *Méta-modèle de Qualité - QUIMERA (Frey et al. 2011b)*

4.2.4 Bilan

QUIMERA constitue une contribution conceptuelle qui ne se limite pas au domaine de l'IHM. En particulier, nous envisageons de l'utiliser pour structurer les critères de qualité des langages de modélisation.

Sa définition, comme celle des autres méta-modèles du langage UsiXML, a renforcé notre expérience en terme de définition de langages. Nous avons remarqué qu'il est difficile de mettre en pratique certaines recommandations pour la définition d'un méta-modèle. Ainsi au sein du projet UsiXML, il est recommandé de fournir des méta-modèles complets, la complétude étant définie comme la capacité d'un méta-modèle d'abstraire toutes les caractéristiques d'un domaine par l'identification des concepts et des relations appropriés. Si cette propriété est intéressante, elle est difficile à appréhender lors de la création d'un méta-modèle et son respect mériterait d'être évalué après la création. Il nous est donc apparu le besoin d'évaluer les méta-modèles.

En termes d'auto-explication, notre proposition est pour l'instant théorique. Nous travaillons actuellement à la réalisation d'un outil qui per-

mettra d'expliciter à la conception les choix et à l'éxécution de générer les explications basées sur les modèles QOC et de qualité. Cet outil permet actuellement de suivre une chaîne de transformations de modèles allant d'un arbre de tâches au code. Il reste à ajouter les modèles QOC et de qualité dans la chaîne de transformations. Ces différents modèles seront l'entrée d'un service d'auto-explication qui à partir d'un élément d'une interface et d'une question génèrera la réponse à la question en se basant sur les modèles UsiXML. Les modèles seront ainsi disponibles lors de l'exécution pour nous permettre d'explorer de nouveaux concepts tels que l'adaptation guidée par la qualité, effaçant ainsi les limites entre concepteurs, systèmes et utilisateurs finaux.

4.3 Auto-explication des chorégraphies de services

4.3.1 Contexte

En SI, je m'intéresse à l'auto-explication dans le cadre de processus métier inter-entreprises. Un processus métier est défini comme "un ensemble finalisé d'activités orienté vers la production d'un résultat qui a une valeur, une importance, pour le client" (Hammer et Campy 1993). Un processus propose une vision horizontale de l'entreprise contrairement à la vision verticale qui se concentre sur les fonctions de l'entreprise (vente, production, achat ...) plutôt que sur la manière de rendre un résultat au client.

L'implémentation des processus métier des entreprises fait de plus en plus souvent appel à des approches basées sur les services. Une entreprise peut avoir besoin de services fournis par d'autres organisations ou d'autres entités (personnes physiques, logiciels, entreprises, etc.) pour réussir à remplir les exigences de ses clients. Du point de vue du client, une seule entreprise lui rend le service même si ce service nécessite de coordonner différents services fournis par d'autres organisations. On parle alors de composition de services.

La composition de services a fait l'objet de nombreux travaux de recherche (Milanovic et Malek 2004, Koehler et Srivastava 2003) et industriels (Oracle dernière consultation 07/2011) ayant conduit à des efforts de standardisation (OASIS 2003, W3C 2005). Différentes approches de composition apparaissent comme l'orchestration et la chorégraphie de services. L'orchestration de services définit l'enchainement des appels aux services du point de vue d'un participant. La chorégraphie trace les échanges des messages entre les différents participants d'un point de vue global. Il s'agit d'un contrat métier où les participants se sont mis d'accord sur l'ordre et la manière d'interagir. Cette vision globale complète la vision locale de chaque participant qui définit son propre processus interne (orchestration). D'ailleurs seules les orchestrations s'exécutent, les chorégraphies ne sont qu'une représentation globale des processus locaux.

Dans le cadre du travail de Mario Cortès-Cornax (M2R puis thèse depuis septembre 2010, co-encadrés avec D. Rieu), nous nous intéressons particulièrement au concept de chorégraphie qui est proche conceptuellement de celui de processus métier. Un exemple simple de chorégraphie est l'achat en ligne d'un ordinateur à monter. Si le fournisseur peut four-

nir la commande, il envoie une confirmation au client et réalise la livraison. Sinon il peut contacter ses sous-traitants pour lui fournir les pièces manquantes. Ce n'est qu'une fois qu'il aura obtenu toutes les pièces manquantes et qu'il aura réalisé l'ordinateur qu'il enverra la confirmation et la commande. De point de vue du client, seul le résultat est visible ; le fait que le fournisseur ait fait appel ou non à ses fournisseurs externes relève de sa gestion du processus de commande.

L'un des problèmes de ce type de processus métier est de comprendre comment ils se déroulent : ont-ils recours à des services externes ? un service est-il bloquant ? etc. L'objectif de notre travail est de permettre à une chorégraphie de services d'expliquer ses exécutions à partir de ses modèles sans intervention humaine. C'est dans ce sens que nous utilisons le terme "auto-explication de chorégraphies de services".

4.3.2 Approche

Pour expliquer une chorégraphie de services, nous proposons d'établir un passage graduel entre le monde de la modélisation des processus métier et celui de l'implantation sous forme de services. Notre approche est basée sur les principes de vues et de niveaux d'abstraction. Les différents niveaux doivent permettre de fournir des représentations, plus ou moins détaillées, aux différents acteurs d'un processus métier. Les représentations sont issues des modèles, qui décrivent une chorégraphie avec plusieurs niveaux de détails. Il faut donc définir différents niveaux d'abstraction et garantir des transitions progressives entre eux.

Au niveau le plus bas, les modèles sont le reflet de l'implémentation alors qu'au niveau le plus haut, ils doivent correspondent à des concepts des processus métier. Nous avons identifié 3 niveaux d'abstraction qui ont leur équivalent en MDA (Fig. 4.4) : le niveau conception (PSM) qui correspond à une description en langage WS-CDL, qui est la recommandation du W3C pour la spécification des chorégraphies de services ; le niveau analyse (PIM) qui ne retient que les concepts sans entrer dans les détails d'une technologie particulière et le niveau domaine (CIM) qui ne représente que les éléments de base d'une chorégraphie c'est-à-dire les participants, les rôles et les services. Pour chaque niveau, nous avons conçu un méta-modèle qui définit le périmètre des informations disponibles.

Pour garantir une traçabilité forte entre les différents niveaux, nous avons utilisé des règles de construction simples : un méta-modèle de niveau N (par exemple celui d'analyse) doit contenir les même éléments ou des éléments raffinés du méta-modèle de niveau N+1 (par exemple celui de domaine). Ainsi le méta-modèle d'analyse contient les classes "Role", "Participant" et "Relationship" qui se retrouvent telles quelles dans le méta-modèle de domaine.

L'objectif est de faire le pont entre le monde opérationnel des services et celui de l'organisation de l'entreprise, en proposant un langage pour les chorégraphies de services. Notre approche prise dans le sens descendant contribute à la définition d'une chorégraphie de services par un processus de raffinement de modèles. Dans le sens ascendant, elle permet d'expliquer à différents niveaux de détails une chorégraphie. On utilise donc un seul langage utilisé différemment suivant les niveaux d'abstraction.

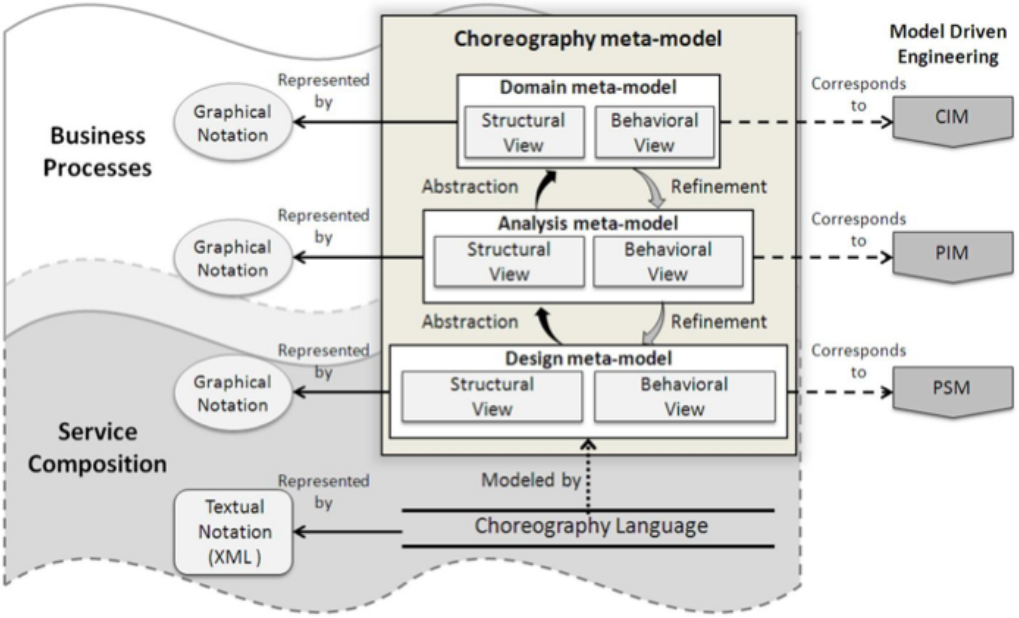


FIGURE 4.4 – *Langage pour la chorégraphie de services issu de (Cortes-Cornax et al. 2011a)*

Ce langage pourrait d'ailleurs être la nouvelle version du langage BPMN, BPMN2 (OMG 2011), qui permet désormais de représenter les chorégraphies. Nous n'avons pas pu nous y référer lors de notre travail qui a débuté en février 2010, toutefois nous étudions actuellement la possibilité de rapprocher notre solution de BPMN2.

4.3.3 Langage de modélisation de chorégraphies de services

Le langage de modélisation des chorégraphies de services est défini par une syntaxe abstraite spécifiée par des méta-modèles, une syntaxe concrète i.e. une notation graphique pour les méta-modèles de domaine et d'analyse et d'une sémantique décrite en langue naturelle.

Méta-modèles Nous avons réalisé trois méta-modèles (Fig. 4.4) chacun correspondant à un niveau d'abstraction. Il y a le méta-modèle de conception, celui d'analyse et celui de domaine. Chaque méta-modèle est constitué de deux vues différentes : une vue structurelle où nous retrouvons la partie plutôt statique de la définition des éléments (participant, rôle, relation etc) de la chorégraphie et une vue comportementale qui définit les réactions des différents éléments par rapport aux actions d'autres éléments.

Nous nous contentons ici de présenter le méta-modèle d'analyse (Fig. 4.5) qui illustre bien notre proposition sans tomber dans la simplicité de celui de domaine, ni dans la complexité de celui de WS-CDL. On peut néanmoins remarquer la traçabilité avec le méta-modèle de domaine dont les classes sont représentées sans couleur de fond.

Du point de vue structurel, un participant qui joue un rôle doit fournir

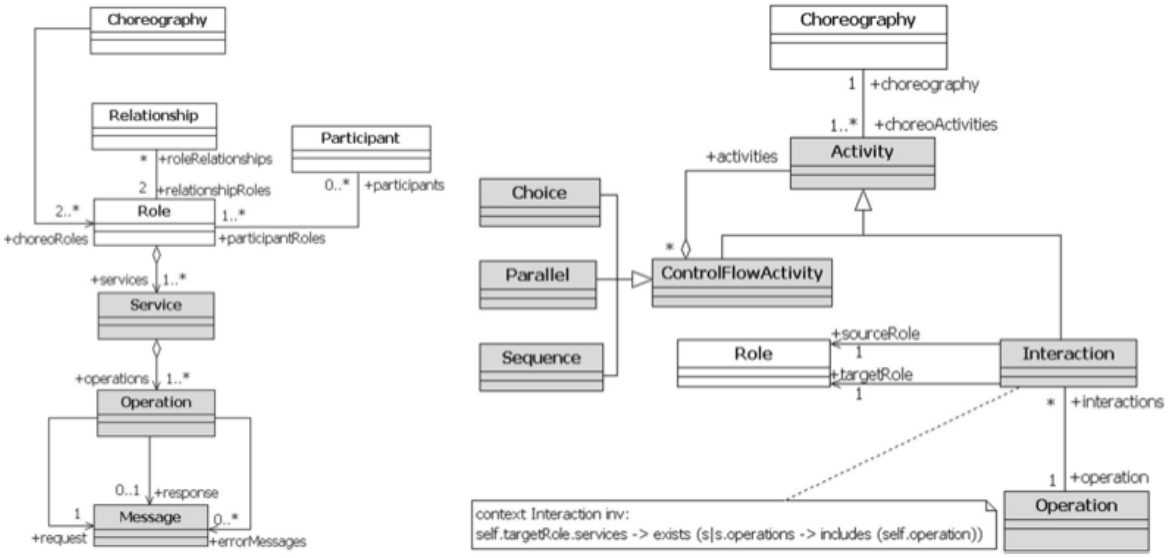


FIGURE 4.5 – *Méta-modèle d'analyse des chorégraphies de services issu de (Cortes-Cornax et al. 2011a)*

les services ("Services") définis pour ce rôle pour respecter la chorégraphie. Un service est considéré comme une manière d'accéder à des opérations définies ("Operation"). Chaque opération définit un message de demande ("Request") et optionnellement un message de réponse ("Response") ou des messages d'erreur ("Error"). Par exemple, pour l'achat de matériel en ligne, nous identifions les rôles client, fournisseur, sous-traitant et soumissionnaire. Le fournisseur offre le service de "ManagerOrderRequest". La demande est un message de commande et la réponse est l'accusé de réception de ce message.

Du point de vue comportemental, une chorégraphie est un ensemble ordonné d'interactions entre les rôles. Nous devons donc trouver des mécanismes pour les composer et les ordonner. Nous introduison la classe "Activity" comme une généralisation des activités "ControlFlowActivity" et des interactions "Interaction". Une activité "ControlFlowActivity" peut être un choix, une exécution paralèlle ou séquentielle. Quand une opération est appelée dans une interaction, ce doit être une opération définie comme étant un des service du rôle cible ("TargetRole").

Syntaxe concrète Pour définir notre notation graphique, nous nous sommes basés sur les principes énoncés dans (Moody 2009) pour la «physique» des notations. Nous avons essayé d'appliquer certains d'entre eux, en sachant que ces principes s'influencent mutuellement. Nous avons appliqué les principes suivants :

- Principe de gestion de la complexité. Pour gérer la complexité de la notation, nous avons utilisé différents niveaux d'abstraction et nous avons séparé les représentations structurelle et comportementale.
- Principe d'intégration cognitive qui permet de faire le lien entre les différentes représentations. La représentation d'un rôle est la même dans les visions structurelle et comportementale.

– Principe d'expressivité visuelle. Pour utiliser plus efficacement les capacités visuelles, nous avons joué sur les formes mais également sur la couleur.
– Principe d'économie graphique. Pour respecter les capacités cognitives des concepteurs (7+-2 symboles pour une représentation) ; par exemple, nous avons limité le nombre de symboles dans la partie structurelle.

Des tableaux (Cortes-Cornax 2010) ont été réalisés afin de faire la correspondance entre le méta-modèle et les éléments graphiques choisis. La vue statique utilise une icônographie proche de celle des diagrammes de classes UML alors que la vue dynamique s'inspire des diagrammes de séquences UML. Cette notation nous permet d'obtenir des modèles comme celui de la Fig. 4.6 qui correspond à modèle de domaine statique pour l'exemple de l'achat d'ordinateur sur internet.

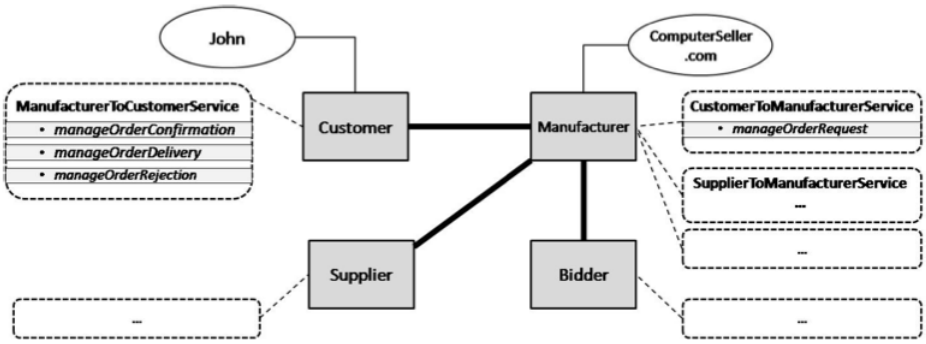


FIGURE 4.6 – *Exemple de modèle statique de chorégraphie issu de (Cortes-Cornax et al. 2011a)*

Validation

Le langage de modélisation pour les chorégraphies de services a été évalué lors d'une expérimentation conçue et réalisée avec l'aide de Nadine Mandran (Cortes-Cornax 2011). L'expérimentation avait pour objectifs d'évaluer deux hypothèses : 1) le méta-modèle proposé représente bien les concepts de la chorégraphie de services ; 2) notre notation est plus appropriée pour les chorégraphies que celles de BPMN2. Elle a eu pour sujets, 4 spécialistes des services et 4 spécialistes des processus métier.

Elle a permis d'aboutir aux conclusions suivantes :
– des problèmes d'incompréhension des concepts sont apparus car les termes utilisés et la façon de nommer (par exemple, ParticipantType, RoleType, RelationshipType) étaient trop proches de WS-CDL.
– la séparation entre les vues structurelles et comportementales n'était pas assez marquée.
– la vue structurelle du niveau domaine a paru un peu pauvre en se limitant aux participants, aux rôles et à leurs relations.
– la représentation de la vue comportementale par une extension des diagramme de séquences d'UML est apparue intuitive et facile à mettre en œuvre. Néanmoins les sujets ont noté une éventuelle difficulté à l'utiliser pour des processus complexes.

Ces remarques nous ont permis de faire évoluer les méta-modèles et la notation (Cortes-Cornax 2011). De plus, la sortie du standard BPMN2 nous amène aussi à revoir notre proposition pour en être plus proche.

4.3.4 Bilan

Le langage pour les chorégraphies de services a permis de mieux définir le concept de chorégraphie en identifiant les concepts principaux (ceux du méta-modèle de domaine), de ceux de détails (ceux du méta-modèle d'analyse) et de ceux relatifs à une technologie (ceux du méta-modèle de conception). Cette compréhension a abouti à l'identification des besoins pour la modélisation des chorégraphies et à l'évaluation du potentiel de la nouvelle version de BPMN pour les chorégraphies (Cortes-Cornax et al. 2011b).

Les niveaux d'abstraction et les vues semblent des techniques de méta-modélisation intéressantes pour aboutir à des modèles compréhensibles. Les évaluations menées tendent à valider ces techniques. Elles montrent aussi la faisabilité et l'utilité des pratiques expérimentales pour la conception de langage de modélisation.

Enfin il est nécessaire faire le lien entre les implémentations qui exécutent des orchestrations de services et notre langage afin de pouvoir visualiser de réelles exécutions de chorégraphie. Nous envisageons de développer un environnement de suivi de chorégraphies qui permettrait de 1) concevoir des chorégraphies à différents niveaux d'abstraction ; 2) de changer de niveaux d'abstraction facilement par transformation de modèles et 3) de visualiser l'exécution des services à différents niveaux. Dans le cadre de chorégraphies complexes, le passage à l'échelle risque d'être limité par nos représentations. En plus des techniques de méta-modélisation, nous devons envisager d'utiliser des techniques de visualisation comme le préconise (Mosser et al. 2010) pour la visualisation de composition de fragments d'orchestration. Nous espérons ainsi permettre aux chorégraphies s'auto-expliquer afin de comprendre plus facilement l'exécution d'un processus métier et réduire la boucle qui existe entre les phases descendante de conception et celles ascendantes de pilotage du système.

4.4 Apports et Perspectives

4.4.1 Contributions

Nos contributions pour l'auto-explication sont, pour l'instant, théoriques. Nous avons défini les modèles conceptuels répondant à nos besoins : nous avons identifié et formalisé les informations pour expliquer un système.

Dans le cadre de l'auto-explication des IHM, nous augmentons les modèles utilisés pour expliciter les choix de conception et leurs évaluations à l'exécution alors que pour les chorégraphies de services, nous nous sommes focalisés sur la facette fonctionnelle des services à différents niveaux d'abstraction.

Les approches ou techniques développées dans chacun des domaines sont transposables dans l'autre domaine. L'approche d'argumentation dé-

veloppée en IHM est assez facilement applicable à la modélisation des processus métier. Ainsi il serait possible d'expliquer les choix d'organisation induits par le processus. De même, les techniques de modélisation (niveaux d'abstraction et vues) développées dans le cadre des chorégraphies de services pourraient aussi être utilisées pour définir des explications plus ou moins détaillées sur l'IHM.

4.4.2 Perspectives

Les solutions envisagées pour l'IHM et pour les SI sont complémentaires et il est possible d'envisager à plus long terme des auto-exlications portant autant sur les aspects organisationnels qu'interactionnels d'un système. Avec le travail que nous avons initié, nous pouvons envisager de fournir de chorégraphies de services s'auto-expliquant aussi bien sur le processus métier qu'elles représentent, que sur les IHM proposées par les services. Une explication de l'organisation serait donnée par les modèles relatifs à la chorégraphie alors que les services mis en œuvre embarqueraient les modèles pour l'auto-explication de l'interface.

Mais il nous reste un enjeu important à traiter aussi bien en IHM qu'en SI : c'est de rendre les modèles vivants à l'exécution. Cette piste devrait être abordée sous peu dans le cadre des thèses d'Alfonso Garcia-Frey et de Mario Cortès-Cornax.

Outre l'apport pour l'auto-explication, les méta-modèles que nous avons définis, nous permettent d'identifier des besoins dans ce cadre. Ainsi nous avons noté le besoin :

- d'outils de support au travail collaboratif sur les modèles. En effet, il est rare qu'un méta-modèle soit issu du travail d'une seule personne. Il est nécessaire de favoriser la discussion autour de l'activité de modélisation et de capitaliser les apports et les connaissances de chacun.
- de guides pratiques pour la conception, mais aussi pour l'évaluation de langages. Même si des règles de bonnes pratiques sont préconisées lors de la conception d'un méta-modèle (comme c'est le cas pour UsiXML), elles sont souvent trop abstraites pour être facilement suivies. Le problème est encore plus saillant dans le cas de l'évaluation pour laquelle les connaissances sont en cours d'élaboration. Des supports méthodologiques doivent venir assister les concepteurs de langages dans ces tâches difficiles.
- de techniques de visualisation à coupler avec celles de modélisation pour obtenir des modèles plus lisibles. Ces techniques ayant montré leur utilité en particulier pour les grandes masses de données, leur utilisation semble une piste intéressante pour favoriser l'auto-explication d'un système.

4.5 Principaux résultats et encadrements

Principales publications

- Alfonso García-Frey, Eric Céret, Sophie Dupuy-Chessa et Gaëlle Calvary. Quimera : a quality metamodel to improve design rationale.

Dans Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing System (EICS'2011), pages 265-270. ACM, 2011.

– Mario Cortes-Cornax. Service choreographies through a graphical notation based on abstraction layers and viewpoints. Dans Proceedings of the 5th IEEE International Conference on Research Challenges in Information Science (RCIS'2011). IEEE, 2011 *Meilleur article de doctorant*.

Encadrements

– Alfonso García Frey, Auto-explication d'IHM plastiques, thèse co-encadrée avec Gaëlle Calvary, début 2009

– Mario Cortes-Cornax, Auto-explication de chorégraphies de services, M2R et thèse co-encadrés avec Dominique Rieu, M2R en 2010, thèse depuis oct. 2010

Projets

– projet européen ITEA UsiXML (User interface eXtensible Mark-up Language), porteur Thalès France, 2009-2012

5 Pratiques et outils pour la gestion des modèles, des problèmes et des solutions communs

Les schémas du mathématiciens, comme ceux du peintre ou du poète, doivent être beaux ; les idées, comme les couleurs ou les mots, doivent s'assembler de façon harmonieuse. La beauté est le premier test : il n'y a pas de place durable dans le monde pour les mathématiques laides.

Hardy

Sommaire

Mon dernier axe de contributions porte sur la gestion de modèles. Il s'agit de pratiques et d'outils ayant pour but de faciliter le travail des

concepteurs : dans la première contribution, il s'agit de guider la conception d'environnements de modélisation ; le deuxième axe concerne la qualité des modèles et des langages et leur évaluation. Dans les deux cas, des solutions communes aux domaines de l'IHM et des SI sont proposées. La route sera donc identique pour les deux domaines afin de fournir des outils de support à la gestion des modèles.

5.1 Construction d'environnements de modélisation

5.1.1 Contexte

Comme nous avons pu le constater au travers de nos travaux en IHM et SI, les besoins des concepteurs en termes de gestion de processus et produits sont divers. Les concepteurs travaillent au sein d'équipes dans lesquelles participent des personnes de différents domaines avec des compétences diverses. Les outils actuels (outils IDM et Ateliers de Génie Logiciel) ne permettent pas la gestion coordonnée et coopérative de modèles requise dans de telles conditions. Ces concepteurs s'appuient sur des outils de modélisation hétérogènes incomplets, limités en fonctionnalités et souvent réduits à supporter quelques méta-modèles et modèles.

L'environnement universel n'existant pas, nous avons envisagé de faciliter la construction d'environnements de modélisation adaptés aux besoins spécifiques des concepteurs de modèles. Toutefois nous ne tentons pas de résoudre le problème technique de l'intégration d'outils (Wicks et Dewar 2007), nous cherchons plutôt à identifier un environnement de gestion de modèles en fonction des acteurs et des processus de conception.

Dans le cadre de la thèse de Jorge-Luis Pérez-Medina (2006-2010, coencadrement avec D. Rieu), nous avons spécifié une plateforme permettant de trouver, de coupler et d'utiliser des briques fonctionnelles répondant aux besoins spécifiques de chaque concepteur. C'est ce travail, qui a été décrit en détails dans (Pérez-Medina et al. 2010a), que nous présentons dans cette section.

5.1.2 Approche

Nous avons choisi d'appuyer nos travaux sur la réutilisation de processus existants, de modèles et d'outils pour trouver une solution aux besoins de concepteurs de modèles et des chefs de projets. Pour atteindre cet objectif de réutilisation, le concept de services est intéressant puisqu'il est possible de sélectionner, voire de découvrir, les services répondant à certaines caractéristiques, puis de les composer pour offrir un service plus complet. Les services servent donc de cadre unificateur à notre approche qui est à la convergence de quatre axes de recherche : l'ingénierie des méthodes situationnelles, l'ingénierie des besoins, la réutilisation et les outils de gestion de modèles.

L'ingénierie de méthodes situationnelles promeut la notion de composants de méthodes réutilisables, de processus de sélection et d'assemblage de ces composants selon la situation particulière de chaque projet (Brinkkemper et al. 1998, Ralyté et Rolland 2001). Elle offre des concepts et des techniques pour construire des processus spécifiques aux projets par l'assemblage/composition de fragments de méthodes (Rolland 2005) . Parmi la littérature étudiée, nous distinguons plus particulièrement l'approche SO2M (Guzelian et Cauvet 2007), plus représentative de nos objectifs, qui utilise la notion de services pour construire des méthodes adaptées aux besoins des concepteurs/développeurs. Par contre, cette approche ne traite pas de l'implémentation de services "méthode » considérée par l'approche MaaS (Rolland 2008). Ces deux approches n'offrent aucun mécanisme de support pour construire des environnements de modélisation pour les

concepteurs/développeurs qui utilisent les services méthodes. Aussi, nous avons plutôt choisi une approche proche de celles de (Mirbel et Crescenzo 2010) proposée pour la recherche de services web pour la communauté des neuroscientifiques. Dans ce travail, la recherche d'un service est basée sur la représentation des processus métier des neurosciences et par celle des intentions des utilisateurs. De façon similaire, le choix d'un environnement de modélisation est issu des buts des concepteurs et de leur processus de modélisation.

Les buts que nous considérons sont inspirés de ceux de l'ingénierie des besoins. Ce domaine étudie comment identifier et représenter les objectifs de l'entreprise dans lequel le système sera utilisé. Donc, le contexte le plus abstrait de fonctionnement du système est centré autour des concepts de scénarios et de buts. Cette approche nous aide à définir et organiser les besoins des concepteurs de modèles.

La réutilisation permet de construire un système nouveau à partir d'éléments (objets, composants, etc.) existants, éprouvés et réutilisables. Nous utilisons cette approche pour réutiliser des produits (modèles) et processus (méthodes) afin de faciliter la construction des environnements adaptés aux besoins des concepteurs.

Enfin, il est nécessaire de disposer d'outils existants de gestion de modèles à aligner avec les besoins intentionnels et organisationnels permettant d'assister les méthodes.

Ces quatre axes sont utilisés pour concevoir les services qui structurent l'ensemble des connaissances nécessaires pour décrire les besoins des concepteurs de modèles, les méthodes adaptées à ces besoins et les outils supports de ces méthodes.

5.1.3 Trois niveaux de services pour la construction d'environnements de modélisation

Présentation générale

Notre structuration s'appuie sur trois niveaux de modélisation (Fig. 5.1) dont les fournisseurs, les clients et les services sont différents.

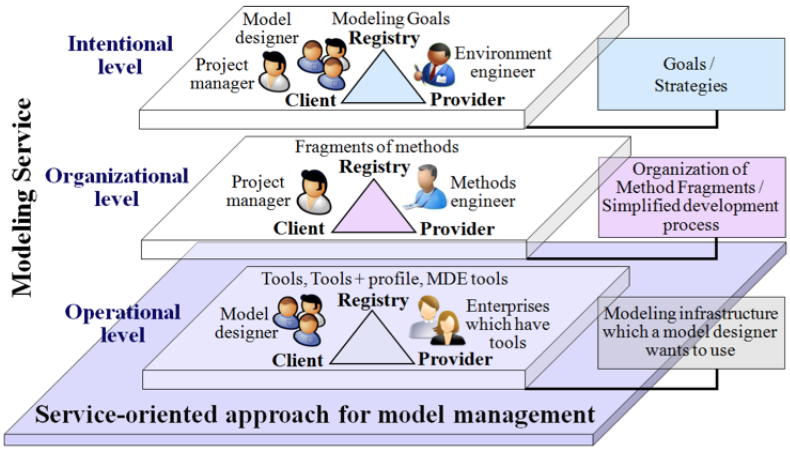


FIGURE 5.1 – *Trois niveaux de services issu de (Pérez-Medina et al. 2010a)*

Le premier niveau correspond à la couche opérationnelle. Cette couche

permet de définir l'infrastructure de modélisation d'un concepteur de modèles lambda ("models designers »). Il peut, par exemple, s'agir d'un concepteur en IHM qui cherche un outil d'édition pour les arbres de tâches. Le client est donc un concepteur de modèles qui désire gérer des modèles de manière individuelle ou collaborative (avec d'autres concepteurs). Il a besoin de définir un environnement de modélisation offrant des outils adaptés à ses préférences en termes de gestion de modèles. Les services opérationnels correspondent aux outils de modélisation (ateliers de génie logiciel ou outils d'IDM).

La couche organisationnelle permet la description de fragments de processus. Par exemple, on peut considérer comme un service organisationnel le processus de "Spécification Interactionnelle des besoins" de Symphony Etendue. La couche organisationnelle est inspirée des travaux de Ralyté (Ralyté 1999) qui a adopté les idées de l'ingénierie des méthodes situationnelles. Dans notre travail, nous utilisons la notion de service pour soutenir la construction de processus de conception de systèmes en assemblant des fragments de méthodes. Il s'agit d'offrir un support à base de services à des groupes de projet. Dans cette couche, les rôles et les activités sont exprimés sous la forme de processus de développement simplifiés. L'objectif est de capitaliser des fragments de méthodes dans le but de fournir à chaque concepteur, donc chaque rôle du groupe projet, l'environnement de modélisation qui lui convient. Un fragment de méthode est modélisé par un ensemble d'activités réalisées par des rôles. Les activités sont décrites en termes d'actions sur des modèles.

La couche organisationnelle permet de réutiliser de façon coordonnée les services opérationnels. Les clients sont, dans ce cas, les chefs de projet ("projets manager ») cherchant à définir et à administrer des rôles et des activités sous la forme de processus de développement. Les chefs de projet vont choisir des services organisationnels (parties de processus de conception) qui nécessitent la mise en place de services opérationnels de gestion de modèles. Ainsi ils définissent la création des environnements de gestion de modèles pour les concepteurs impliqués dans leurs processus de développement.

La couche intentionnelle permet la modélisation des buts. Il s'agit de conceptualiser des besoins stratégiques de modélisation requis par un sujet individuel, un groupe d'individus, une unité de travail ou toute organisation qui participe au processus de développement de systèmes. Cette couche permet de réutiliser les services organisationnels. Le fournisseur correspond à l'ingénieur de l'environnement ("environment engineer »). Il s'agit d'un nouveau métier qui s'occupe de l'administration et la gestion de la plateforme. Les clients sont ceux des couches organisationnelle et opérationnelle, c'est-à-dire les concepteurs de modèles ("model designer ») et les chefs de projets ("project manager »). Pour ces clients, les services sont les buts proposés par l'ingénieur de l'environnement qui peuvent être vus comme les buts à atteindre pour la gestion de modèles, par exemple : " spécifier un Système Interactif ».

Utilisation des ontologies

Nous faisons l'hypothèse d'une mixité entre ceux qui conçoivent les environnements et ceux qui les utilisent. Ce haut niveau de maturité entre client et fournisseur peut néanmoins être réduit par l'alignement des vocabulaires c'est-à-dire par l'utilisation d'un vocabulaire contraint et commun spécifié sous forme d'ontologies. Elles définissent une terminologie pour partager un vocabulaire commun au sein d'un domaine. Nous pouvons noter en particulier, OWL-S (Coalition 2004) qui permet une description des services enrichie notamment par un modèle de processus exprimant comment le service peut être utilisé. Si cette solution est intéressante, nous la voyons comme une solution technique qui pourrait contraindre notre solution conceptuelle (nous tenons à avoir 3 niveaux conceptuellement distincts). Aussi nous retiendrons de ces travaux l'intérêt d'utiliser des ontologies pour décrire les services.

Nous avons choisi d'intégrer les ontologies dans nos modèles de services de manière à offrir une vision intégrée de chaque niveau d'abstraction. Les ontologies généralement intégrées sous la forme d'une adaptation du patron Item-Description auquel nous avons rajouté deux classes énumérées. Nous appellerons ce patron : Patron "Concept-Term" (fig. 5.2). Nous avons utilisé ce patron Concept-Term pour les ontologies de verbes d'intention, des produits de modélisation, d'acteurs, de processus et de facteurs de qualité logicielle.

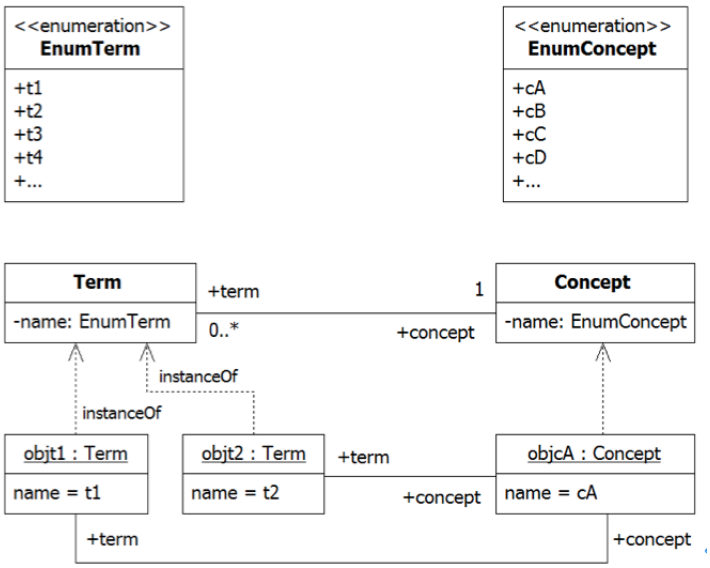


FIGURE 5.2 – *Patron "Concept-Term" issu de (Pérez-Medina et al. 2010a)*

Par exemple, notre modélisation des verbes utilise les ontologies de buts (Guzelian et Cauvet 2007) qui décrivent les intentions spécifiques pour la gestion de modèles. D'un côté, nous souhaitons identifier des catégories de verbes (instances de la classe ConceptVerb), telles que "Acquisition of Knoweldge" ou "Configuration", selon ses propriétés communes et de l'autre, des verbes, comme "Define" ou "Study", qui seront utilisés pour la modélisation des intentions (instances de la classe TermVerb). Les

verbes correspondant au même type d'activité de développement appartiennent à la même catégorie de verbes.

Nous verrons par la suite comment cette ontologie et celles des produits de modélisation, des acteurs, des processus et des facteurs de qualité logicielle sont utilisées dans nos modèles de services.

Couche intentionnelle

Un service intentionnel est un service orienté métier qui représente les buts ou intentions dans le cadre du développement logiciel (par exemple : spécifier un système interactif, identifier les cas d'utilisation, élaborer un scénario, personnaliser le processus de spécification de besoins). Il peut être composé d'autres services intentionnels plus élémentaires (Fig. 5.3). Ainsi "Spécifier un système interactif" est un but composé de plusieurs sous-buts dont "Spécifier les aspects Interactionnels" qui est lui-même décomposé en sous-buts. L'un de ses sous-buts est "Spécifier les tâches de l'utilisateur" qui ne peut pas être décomposé et que nous considérons donc comme un but élémentaire.

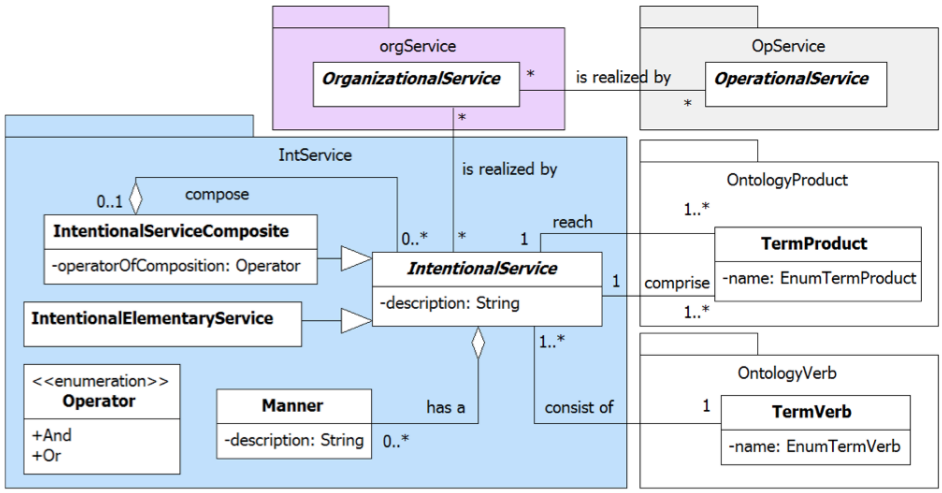


FIGURE 5.3 – *Modèle de services intentionnels issu de (Pérez-Medina et al. 2010a)*

Un service intentionnel est défini par un verbe. Un service intentionnel est également associé aux produits utiles à la réalisation du service (association "comprise ») et aux produits générés par le service (association "reach"). De plus, un service intentionnel est lié à une façon de faire, une manière, pour réaliser l'intention.

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

Le produit est l'élément qui complète le verbe pour décrire l'intention, cet élément est représenté par la classe "TermProduct" de l'ontologie de produits "OntologyProduct" qui regroupe les différents objets qui

peuvent être élaborés, modifiés ou utilisés au cours de la gestion de modèles comme scénario ou processus métier. Ce sont les objets sur lesquels portent les actions. Ils sont instance de "TermProduct" et sont rattachés à leur catégorie (instance de "ConceptProduct") : par exemple, modèle, méta-modèle, documentation.

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

Les services intentionnels sont représentés sous la forme de graphes "et/ou" sans contrainte temporelle. Par exemple, un service "Spécifier un Système Interactif" peut être décomposé par les services : "Spécifier les aspects fonctionnels", "Spécifier les aspects Interactionnels" et "Spécifier l'architecture logicielle". Tous les services peuvent être liés à des services organisationnels pour proposer une solution (en termes de processus) aux buts correspondants.

Couche organisationnelle

Un service organisationnel consiste en une composition de fragments de méthodes qui peuvent être réutilisés par des concepteurs de modèles, pour répondre aux buts des services intentionnels. Il est évidement possible de lier les buts intentionnels à plusieurs processus existants. Par exemple, des processus répondant au but "Spécifier les aspects Interactionnels" sont le processus proposé par (Sousa et al. 2007) ou une partie de la phase "Spécification organisationnelle et interactionnelle des besoins" de la méthode Symhony étendue (Fig. 5.6). Dans notre approche, ces activités correspondent à deux services organisationnels complexes.

Un service organisationnel complexe est composé d'autres services organisationnels. Par exemple, le fragment correspondant à la phase de "Spécification organisationnelle et interactionnelle des besoins" de Symphony étendue constitue un service organisationnel composé de sous-services (un par activité). La composition d'un service organisationnel est représentée dans le paquetage "orgService ». Comme dans les méthodes situationnelles, il est difficile de composer des fragments de processus. Aussi la pertinence d'une composition de service est laissée à l'appréciation de l'ingénieur de méthodes. Néanmoins la composition est facilitée par l'utilisation de modèles de processus identiques et par un vocabulaire commun défini dans nos ontologies.

Dans le modèle de la Fig. 5.4, un fragment de méthode est représenté par un service élémentaire organisationnel défini en termes de manipulations de modèles. La manipulation des produits consiste en la description de chaque service élémentaire sous la forme d'actions sur les produits utilisés ou produits : par exemple, un processus de spécification interactionnelle des besoins, produit un modèle des tâches de l'utilisateur qui peut être utilisé par d'autres fragments de méthode. Nous utilisons ici l'ontologie des produits ("OntologyProduct").

Un service organisationnel est effectué par un ou plusieurs rôles. Les

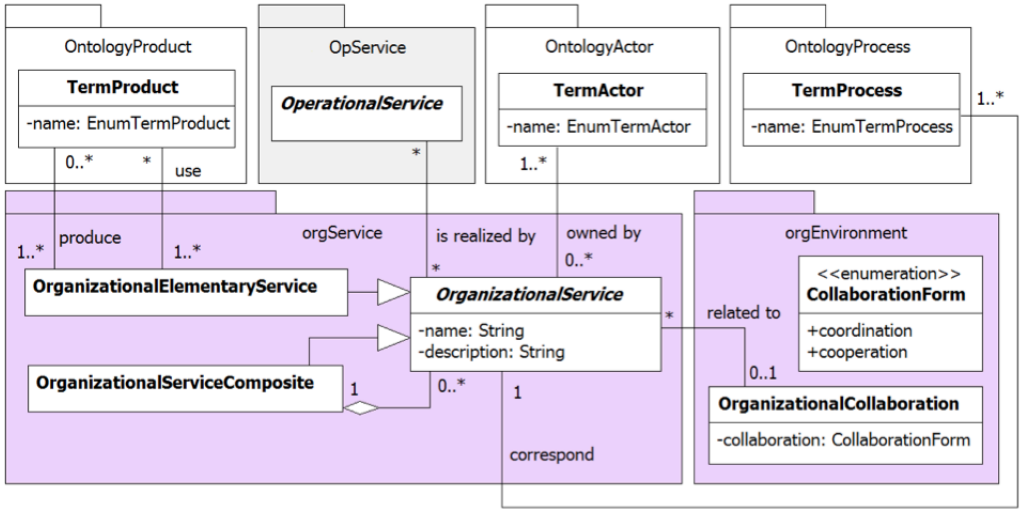


FIGURE 5.4 – *Modèle de services organisationnels issu de (Pérez-Medina et al. 2010a)*

acteurs correspondent aux rôles responsables de la réalisation d'un service organisationnel. Pour ce faire, nous utilisons l'ontologie des acteurs "OntologyActor" qui définit un ensemble de rôles (concepteur, développeur, architecte, etc.) pour les acteurs qui ont en charge les problèmes d'ingénierie (Guzelian et Cauvet 2007).

Un concepteur de modèles peut définir et réutiliser plusieurs services organisationnels. A ce niveau, le terme de collaboration (Grebici et al. 2005) est utilisé pour représenter les tâches de coordination et de coopération entre les concepteurs. La collaboration prend en compte la cohérence et la synchronisation des activités entre les différents spécialistes intervenant dans un processus de conception. Les activités de coordination reposent sur une décomposition du travail en activités de buts similaires, alors que les activités de coopération sont basées sur un objectif de modélisation commun. Chaque spécialiste apporte ses modèles et la coopération permet de produire des modèles consensuels ou communs. Ainsi l'activité de "Spécification externe de l'interaction" est caractérisée par un service qui a la particularité de nécessiter une coopération entre le spécialiste IHM et l'ergonome.

Un autre aspect considéré par notre modèle organisationnel est le fait que les services organisationnels utilisent les services opérationnels pour supporter la gestion des activités de modélisation. Par exemple, un service organisationnel qui comprend la description du modèle de tâches peut être lié aux services opérationnels qui offrent toutes les fonctionnalités pour faciliter l'édition des arbres de tâches.

Couche opérationnelle

Un service opérationnel (voir le paquetage "opService" dans la Fig. 5.5) correspond à une application exécutable composée d'autres services de gestion de modèles. Un service de gestion de modèles est un outil de gestion de modèles offert par un fournisseur. Ces services peuvent être fournis par des outils de gestion de modèles (par exemple : un AGL, un outil de modélisation ou un outil de transformation de modèles, etc.).

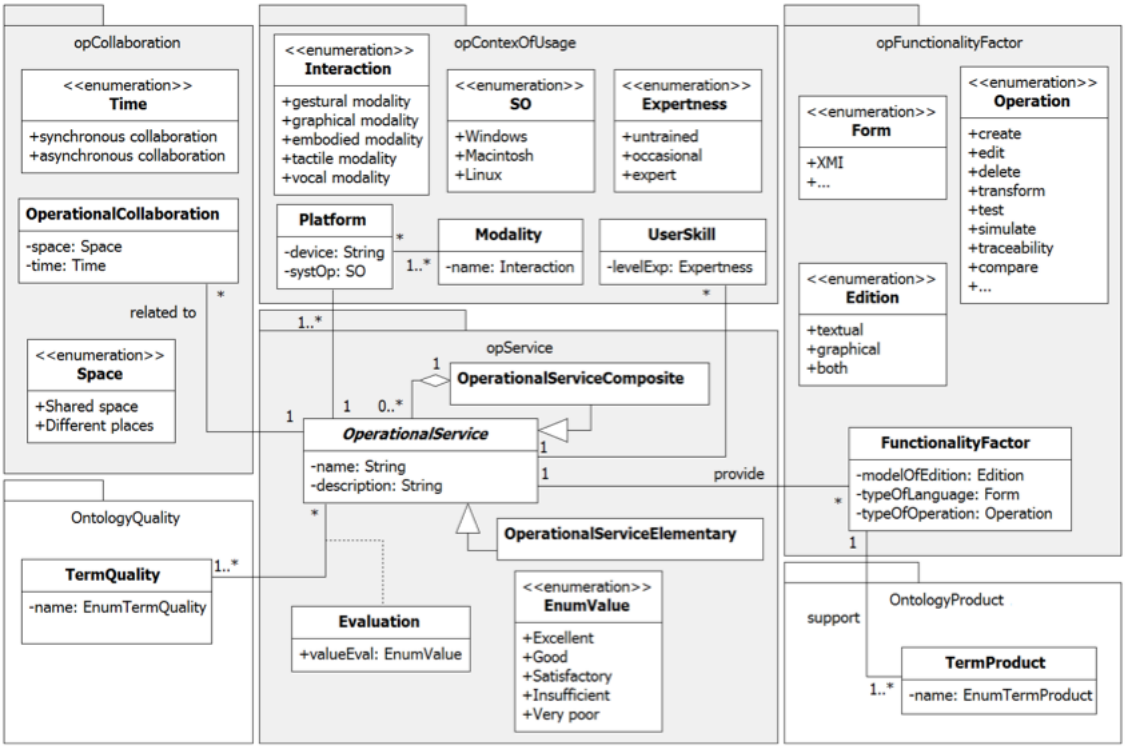


FIGURE 5.5 – *Modèle de services opérationnels issu de (Pérez-Medina et al. 2010a)*

Les services opérationnels peuvent porter sur un ou plusieurs modèles (par exemple : le modèle des arbres de tâches, le modèle de classes, le modèle de cas d'utilisation, etc.). De la même façon que dans les couches intentionnelle et organisationnelle, nous avons utilisé ici l'ontologie des produits "OntologyProduct".

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

Pour supporter les besoins de création d'environnements collaboratifs, un service opérationnel est lié à plusieurs formes de collaboration. Une forme de collaboration est caractérisée suivant le modèle de Denver de Salvador (Salvador et al. 1996) et sur la classification Espace-Temps proposée par Ellis (Ellis et al. 1991). Le modèle de Denver établit que les utilisateurs peuvent être proches ou éloignés et interagir de façon synchrone (temps réel) ou asynchrone. La classification Espace-Temps proposée par Ellis repose sur deux caractéristiques, à savoir où et quand une action est exécutée par un des utilisateurs par rapport aux autres utilisateurs. Donc, dans le contexte de notre travail, une collaboration est définie en fonction de deux axes : l'espace ("space"), et le temps ("time").

En ce qui concerne le profil de l'utilisateur et le contexte d'utilisation d'un service opérationnel, un service opérationnel peut être exécuté dans différents contextes d'utilisation par des spécialistes qui ont divers niveaux

d'expertise. Le contexte d'utilisation est modélisé par le profil de l'utilisateur, la plateforme et la modalité d'interaction. Un service opérationnel est caractérisé par le niveau d'expertise nécessaire pour l'utilisateur. Nous considérons, dans nos travaux, les niveaux d'expertise suivants : expert, occasionnel et novice. Le contexte d'utilisation est un espace d'information structuré qui inclut les plateformes logicielle et matérielle exigées par le service. Il est relié à une variété de modalités d'interaction. Par exemple, nous considérons ici la possibilité d'avoir des services de gestion de modèles en mode graphique, gestuelle, tactile ou vocale.

Pour garantir la qualité dans l'utilisation d'un outil de gestion de modèles, nous avons inclus dans notre approche la notion de qualité logicielle. La qualité logicielle est une appréciation globale d'un logiciel basée sur de nombreux indicateurs (par exemple : l'interopérabilité, la maintenance, ...). Le modèle opérationnel intègre l'ontologie des facteurs de qualité "OntologyQuality" modélisée par imitation du patron Concept-Term. Dans la figure 5.5, seule la classe TermQuality est représentée. Cette ontologie a été établie à partir du standard IEEE et de l'ISO sur la qualité logicielle.

Comme nous l'avons souligné précédemment un service organisationnel peut être réalisé par plusieurs services opérationnels. Pour l'action "description de modèles de tâches", est représentée sous forme de service organisationnel dans notre approche, est liée à différents services opérationnels, correspondant à des outils pour aider la réalisation de la modélisation des tâches (Fig. 5.6). Ces outils sont des éditeurs de modèles de tâches tels que CTTE (Paterno et al. 1997), TERESA (Berti et al. 2004), K-MADe (Baron et al. 2006).

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

5.1.4 Plateforme de support envisagée

Nous avons étudié la possibilité de réaliser un outil supportant nos trois couches de services pour la construction d'environnements de modélisation. Diverses solutions techniques étaient envisageables. Nous aurions pu, en particulier, nous baser sur les technologies web dont OWL-S. Mais nous avons voulu conserver trois niveaux de services distincts et ne pas nous limiter aux environnements web. Aussi notre solution a été basée sur une plateforme de gestion de services "générique". Nos services pourront ensuite évoquer des services exécutables dans différentes technologies dont OWL-S. Le lien entre les services de modélisation et les services exécutables était envisagé via les paramètres d'entrée et de sortie des services exécutables.

Le prototype doit garantir l'enregistrement, la consultation, la re-

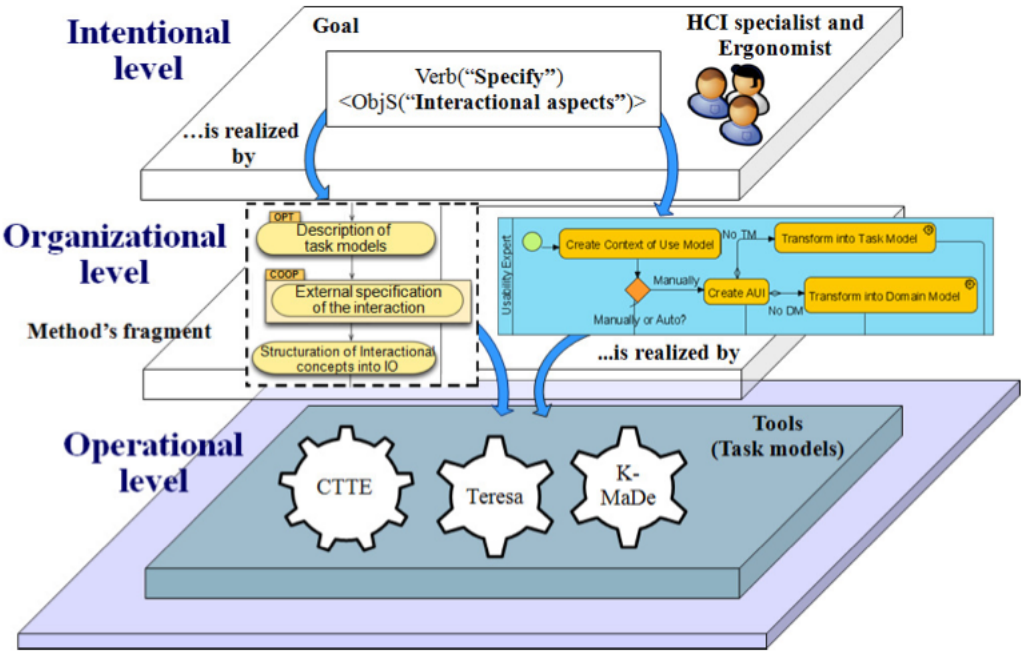


FIGURE 5.6 – *Exemple de lien entre les couches issu de (Pérez-Medina et al. 2010b)*

cherche et la conception de nos trois niveaux de services. A ce jour, le prototype est constitué de deux blocs indépendants mais complémentaires.

Le premier bloc (Fig. 5.7) considère la mise en œuvre d'un registre de services avec l'atelier de gestion de composition de services "ChiSpace" (Yu et al. 2008). L'objectif de cet environnement est de simplifier le travail des développeurs lors de la réalisation d'applications par composition de services. ChiSpace est composé d'un ensemble d'éditeurs basés sur une perspective personnalisée d'Eclipse. Le registre de services correspond à la base de données où seraient stockées les descriptions des trois niveaux de services. Ces descriptions seraient basées sur les trois modèles de services présentés dans les sections précédentes. Le deuxième bloc permettrait d'ajouter, d'afficher, de sélectionner et de valider les services qui sont stockés dans le registre de services. Il est basé sur 'Eclipse Rich Platform' (Eclipse-RCP). Ainsi, Eclipse-RCP est utilisé pour développer les interfaces permettant d'utiliser les fonctionnalités de la plateforme pour les clients et les fournisseurs. La version développée du prototype permet d'ajouter et de rechercher des services intentionnels. La figure 5.7 (partie a) montre l'interface de recherche des services intentionnels.

5.1.5 Apports et Perspectives

Contributions

La proposition présentée dans cette section vise à favoriser la construction d'environnements de modélisation adaptés aux buts et au processus de conception. La caractérisation selon trois dimensions (intentionnelle, organisationnelle et opérationnelle) semble pertinente, même au delà de l'objectif visé. En effet, ce travail peut aussi être considéré comme une for-

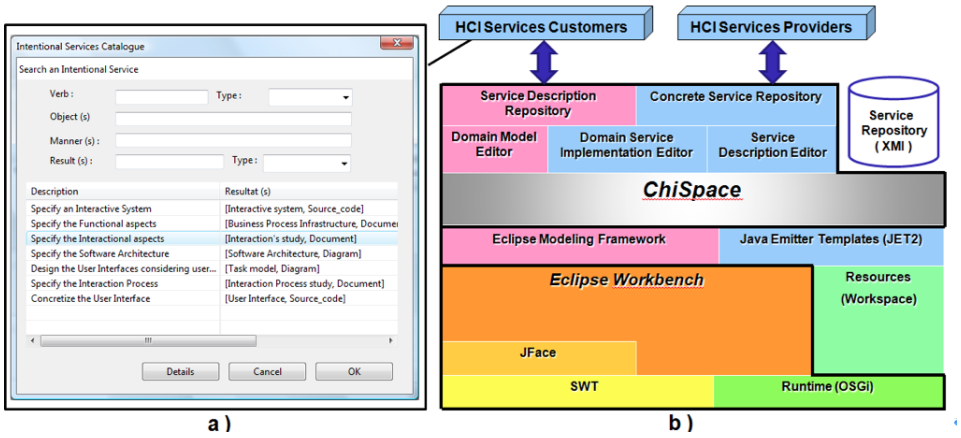


FIGURE 5.7 – *Plateforme envisagée issu de (Pérez-Medina et al. 2010a)*

malisation des connaissances sous-jacentes aux processus de conception, réalisant ainsi un pas vers la réutilisation de fragments de processus.

Perspectives

L'approche service apporte une forme de granularité adaptée à la réutilisation. Néanmoins elle n'est pas pleinement exploitée puisque la composition dynamique et la découverte de services n'ont pas été abordées. Il aurait fallu aller plus loin dans la réalisation du prototype pour voir les possibilités et les limites des services.

Ce travail mériterait d'être poursuivi pour valider ses principes sur des études de cas représentatives. Nous voudrions en particulier valider les trois couches ainsi que leurs liens. Il nous semble nécessaire d'étudier si la couche intentionnelle est effectivement la couche de haut niveau ou si elle est orthogonale aux deux autres. Nous devons aussi montrer l'intérêt de considérer les buts comme des services à part entière, et non pas comme une partie des descriptifs de services comme c'est généralement le cas.

5.1.6 Principaux résultats et encadrements

Principales publications

- J.-L. Perez-Medina, S. Dupuy-Chessa, D. Rieu, Approche orientée services pour la construction des environnements de modélisation, revue des Sciences et Technologies de l'Information, série Ingénierie des Systèmes d'Information, N° spécial "Ingénierie d'entreprise et des systèmes d'information", num. 4, vol. 15, pp. 113-137, 2010.
- J.-L. Perez-Medina, S. Dupuy-Chessa, D. Rieu, A service-oriented approach for interactive system design, 8th International Workshop on TAsk Models and DIAgrams (Tamodia'2009), LNCS 5963, Springer Verlag, September 2009, Belgium, pp 44-57, 2010.

Encadrements

- Jorge-Luis Pérez-Medina, Approche orientée services pour la réutilisation de processus et d'outils de modélisation, thèse co-encadrée avec Dominique Rieu, 2006-2010.

– Stéphanie Marsal-Layat, Spécification d'un outil pour la conception collaborative et réalisation d'un prototype, stage de M2Pro coencadré avec Jorge-Luis Pérez-Medina et Agnès Front, 2007.

5.2 Evaluation des langages et des modèles

5.2.1 Contexte

Comme mentionné en section 2.3.1, la qualité des langages et des modèles ainsi que leur évaluation consitituent un verrou important pour l'expert-user modelling. De nombreux travaux ont abordé ce problème. On peut reprocher à certains de ne pas être utilisables pour n'importe quel langage de modélisation (Lange et Chaudron 2005, Unhelkar 2005) ; à d'autres de ne présenter qu'une pratique isolée des autres pratiques pour l'évaluation (Rossi et Brinkkemper 1996, Aranda et al. 2007, Patig 2008) ; et aux derniers d'être trop abstraits pour être facilement applicables (Lindland et al. 1994, Mohagheghi et al. 2009, Krogstie 2003). Ces défauts, j'ai pu les constater lorsque, au cours de mes travaux, j'ai été amenée à réaliser des études empiriques pour évaluer des langages de modélisation (Céret et al. 2010, Dupuy-Chessa et al. 2011, Cortes-Cornax et al. 2011b). J'ai pu noter le besoin de disposer d'un guide méthodologique, qui présente de manière coordonnée des réponses à des problèmes concrets en matière d'évaluation.

Le problème est que les connaissances dans le domaine de l'évaluation des modèles et des langages ne sont pas toujours matures et vont évoluer avec l'avancée des recherches. La capitalisation de ces savoirs et de ces savoir-faire doit être une mise en commun des pratiques de chacun, qui permettra leur démocratisation.

5.2.2 Approche

La solution que j'envisage est un outil où les spécialistes d'un domaine peuvent collaborer pour décrire, partager et faire évoluer leurs connaissances de manière collaborative. Nous avons entrepris la réalisation d'un tel outil. Nous l'utilisons pour définir un catalogue de patrons pour l'évaluation des langages et des modèles, en collaboration avec Nadine Mandran, Dominique Rieu et Agnès Front, enseignants/chercheurs spécialistes en modélisation dans l'équipe SIGMA du LIG. Ce catalogue est envisagé comme l'une des actions communes dans le cadre de l'action spécifique "Expet-User Modelling" du GDR GPL, qui a identifié les évaluations comme point d'intérêt commun aux membres du groupe.

Les connaissances sur l'évaluation peuvent parfois se traduire en métriques dont le calcul permet d'avoir des éléments estimatifs de la qualité du langage ou du modèle. Aussi ces métriques seront contenues dans notre guide méthodologique au titre des techniques d'évaluation quantitatives possibles, mais elles doivent aussi s'intégrer dans les environnements de modélisation ou de méta-modélisation pour automatiser leur calcul et les rendre plus facilement utilisables. Ce travail sur les métriques est réalisé en collaboration avec Xavier Le Pallec du LIFL dans le cadre du projet

ANR MOANO où nous devons fournir un langage de modélisation pour des botanistes et l'évaluer.

5.2.3 Communauté pour l'évaluation de la qualité des langages et des modèles

Approche

J'envisage la capitalisation des connaissances en qualité des modèles et des langages de modélisation en proposant, non pas des règles généralistes mais des éléments de réponse à des problèmes ponctuels et récurrents lors de l'évaluation d'un langage. Cette vision de réponse à un problème récurrent est représentée par l'approche basée sur les patrons. Les patrons peuvent à la fois représenter des savoirs et des savoir-faire. Un exemple de savoir est que pour la définition d'un langage, tout élément d'une syntaxe concrète doit correspondre à un élément de la syntaxe abstraite (Moody 2009). Un exemple de savoir-faire est de proposer un protocole expérimental générique pour évaluer la facilité de compréhension de n'importe quel langage (Aranda et al. 2007, Patig 2008). Ces savoirs et ces savoir-faire peuvent être liés dans un cataogue de patrons qui forme ainsi un guide méthodologique. Notre guide méthodologique pour la capitalisation des connaissances en évaluation de la qualité des modèles et des langages est donc fourni sous la forme d'un catalogue de patrons (Dupuy-Chessa et al. 2010).

Longtemps les patrons ont été perçus comme des connaissances stabilisées. Ils expriment alors des solutions consensuelles dans un domaine d'expertise et sont formulés par des experts d'un domaine et utilisés par l'ensemble de la communauté. A présent, l'approche patron est également perçue comme un moyen structurant et efficace de partager des connaissances non-stabilisées pour des savoirs et des savoir-faire qui évoluent avec l'avancée des connaissances des spécialistes d'un domaine. Récemment le site Ontology Design Patterns [1] a promu le concept de patron comme une technique de partage et de mise en accord de connaissances communautaires. Ce site regroupe une communauté dont les membres créent des patrons en collaboration grâce à trois sortes de collaboration : la discussion sur le patron, qui peut inclure des suggestions de modification, l'écriture d'une critique et la modification directe du patron. Les patrons deviennent ainsi collaboratifs.

Les patrons collaboratifs

Dans mon travail, les collaborations sont gérées au sein de communautés d'intérêt qui regroupent des catalogues cohérents de patrons. Par exemple, il existe une communauté pour l'évaluation des langages de modélisation et des modèles. Chaque communauté contient des patrons qui sont gérés de manière collaborative. Les modalités de collaborations sont issues des travaux de Lécaille (Lécaille 2003), sur l'évolution d'un type d'objets pendant la conception suivant trois modalités d'action :

1. http ://ontologydesignpatterns.org/wiki/Main_Page

- un brouillon ou draft est un objet sur lequel nous appliquons les modalités de création et de validation des hypothèses ou des solutions d'un problème. Il est défini par le concepteur lui-même ou conjointement avec d'autres acteurs qui utilisent des graphiques représentés par des objets traceables, imprimables ou visualisables à l'écran.
- une pièce à conviction (en anglais Exhibit) est un objet sur lequel on applique la modalité de persuasion en accord avec ce qui est représenté soit pour convaincre de l'existence d'un problème, soit pour montrer une solution et permettre une construction commune et des échanges de points de vue.
- une trace habilitée (Enabled trace en anglais) est un objet traceable ou digigraphique sur lequel on peut appliquer une modalité de circulation sans contrainte. Le créateur de cette trace accepte de la diffuser à d'autres. La trace habilitée est un objet non-officiellement validé, mais suffisament convainquant pour être publié.

Nous nous sommes inspirés de ce travail pour fixer les modalités d'échange entre des spécialistes concernant un patron (Fig. 5.8) (Dupuy-Chessa et al. 2010). Les brouillons ("Draft") sont gardés dans l'espace de travail personnel (privé) de l'un de membres la communauté, que nous nommerons le créateur du patron. Par exemple, Nadine Mandran, notre spécialiste en évaluation a créé des patrons, tels que "Design an experimental protocol" ou "Organize an experimental study", pour décrire ses pratiques pour l'évaluation.

Le créateur peut confronter ses idées avec les points de vue d'autres membres de la communauté : il peut alors proposer le patron à son espace de travail de proximité c'est-à-dire à des personnes de son réseau professionnel en qui il a confiance. Le patron devient alors "Exhibit". Dans l'espace de travail de proximité, le créateur du patron l'expose aux critiques et au jugement des autres. Nadine Mandran a ainsi demandé à Dominique Rieu, Agnès Front et moi-même de contribuer à l'élaboration de ses patrons.

Quand le patron est considéré suffisament abouti, il est validé pour être transmis en dehors du réseau personnel du créateur, dans l'espace public (état "Enabled"). Dans l'espace public, un patron ne peut pas être modifié, mais il peut être utilisé (c'est-à-dire sélectionné et adapté à un problème donné). Il peut aussi être annoté avec des suggestions d'évolution. Ainsi nous comptons rendre nos patrons sur l'évaluation de la qualité des langages publiques et demander aux membres de l'action spécifique "Expert-User Modelling" de les annoter.

Ainsi à partir de ces suggestions, le créateur du patron ou un autre de ses propriétaires peut décider de l'étudier à nouveau. Le patron revient alors dans l'espace privé de son créateur et peut être modifié.

Les opérations que nous venons d'évoquer (création, modification, annotation etc) ne peuvent pas être réalisées par n'importe qui. Nous distinguons trois rôles pour un patron :

- les propriétaires du patron sont le créateur du patron, ainsi que toute autre personne que le créateur accepte comme propriétaire. Ils peuvent effectuer toutes les opérations sur leurs patrons. Dans le cas des patrons sur l'évaluation de la qualité des modèles et des langages, le propriétaire est Nadine Mandran.

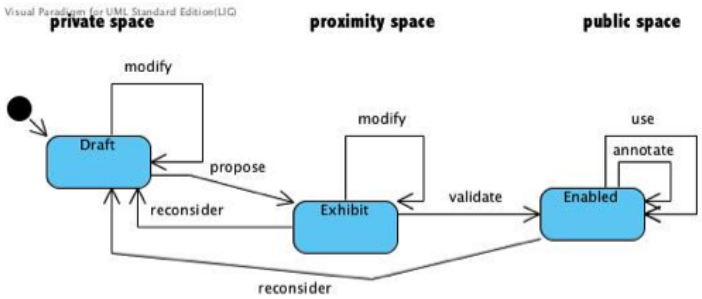


FIGURE 5.8 – *Cycle de vie d'un patron collaboratif issu de (Dupuy-Chessa et al. 2010)*

- les collaborateurs du patron sont les personnes dans l'espace de travail de proximité des propriétaires du patron. Ils peuvent voir et modifier le patron s'il est dans l'état "Exhibit". Dans notre cas, les collaborateurs sont Dominique Rieu, Agnès Front et moi-même. L'évolution d'un patron est donc une activité coopérative où les experts d'un point d'un domaine doivent travailler ensemble pour créer une solution valide.
- les lecteurs peuvent voir, utiliser et annoter le patron quand il est dans l'espace public. Les lecteurs ne sont pas des spécialistes du patron. Ils cherchent simplement des solutions à leurs problèmes. Les lecteurs seront les membres de l'actions specifiques "Expert-User Modelling".

Les patrons collaboratifs semblent être une solution intéressante pour construire, grouper et partager des connaissances d'un domaine. Pour être facilement utilisés, ils doivent être disponibles à travers un support informatique facilement accessible. Nous avons donc conçu un outil de gestion de patrons collaboratifs accessible via le Internet. Cet outil, nommé COpen pour COllaborative Pattern Environment, a été développé en 2010 dans le cadre des mémoires d'ingénieurs CNAM de Ludovic Bouteloup et Pascal Bourgeois, co-encadrés avec Dominique Rieu.

L'outil COpen

Notre objectif est d'obtenir un outil de gestion de patrons qui permette de gérer la création et la modification des patrons de manière collaborative et à distance. Pour éviter les problèmes d'installation et de comptabilité, nous avons voulu une solution accessible via Internet.

Nous avons opté pour l'utilisation d'un système de gestion de contenu qui apporte des outils de collaboration. Nous avons choisi le framework Alfresco Community [2], qui se caractérise par la facilité de création d'un site web avec un partage de documents (incluant forum, wiki, etc.), mais aussi par la complexité de développement de solutions non prévues. COpen est maintenant fonctionnel et peut être utilisé à l'adresse suivante : http ://copen.imag.fr/copen-1.0/page/site-index

Dans notre vision, une communauté, par exemple celle pour l'évaluation des langages, est un site web où les connaissances sont accessibles sur des pages web correspondant à des patrons. Le site web présente la com-

2. https ://www.alfresco.com/fr/

munauté avec ses acteurs, ses buts, ses événements etc. Il propose aussi la discussion avec des outils collaboratifs (wiki et forum) afin de permettre aux membres de la communauté de contribuer à sa vie.

En tant que visiteur, vous pouvez seulement voir ou chercher des patrons. Vous ne pouvez pas les annoter, ni en créer de nouveaux. Pour effectuer ces opérations, il faut être membre de la communauté du patron considéré. Les membres d'une communauté sont des personnes qui y ont adhéré. Ils peuvent utiliser et annoter les patrons. Ils peuvent aussi en créer, devenant ainsi des propriétaires de patrons. De plus s'ils sont collaborateurs pour un patron, ils peuvent le modifier. Enfin un des membres de la communauté a des droits particuliers : il s'agit du modérateur qui gère les membres, accepte ou refuse la publication d'un nouveau patron dans l'espace public. Un diagramme simplifié des fonctionnalités de COpen est donné en Fig. 5.9.

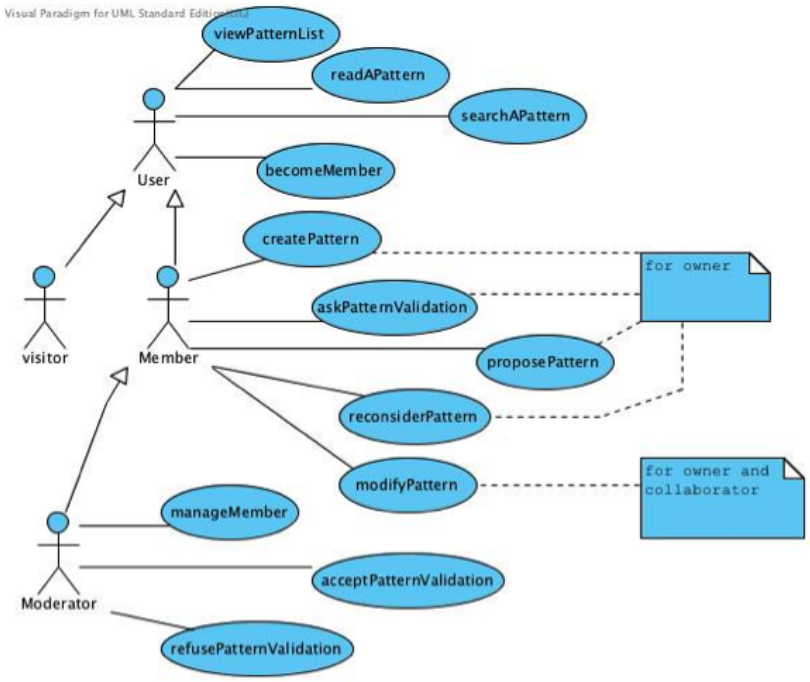


FIGURE 5.9 – *Fonctionnalités de COpen issu de (Dupuy-Chessa et al. 2010)*

La Fig. 5.10 présente l'interface utilisateur de COpen telle qu'elle a été imaginée initialement. Au centre se trouve le patron en lui-même, avec dans un espace en haut à gauche une description résumant le patron. En dessous de cette description, nous envisagions de présenter un historique de l'évolution du patron. Le patron peut aussi être annoté grâce à un espace en bas à droite.

Enfin l'interface présente aussi des informations sur les communautés avec en haut à droite (en symétrique de la description du patron), le descriptif de la communauté du patron et en bas à gauche la liste de autres commuanutés pour faciliter la navigation.

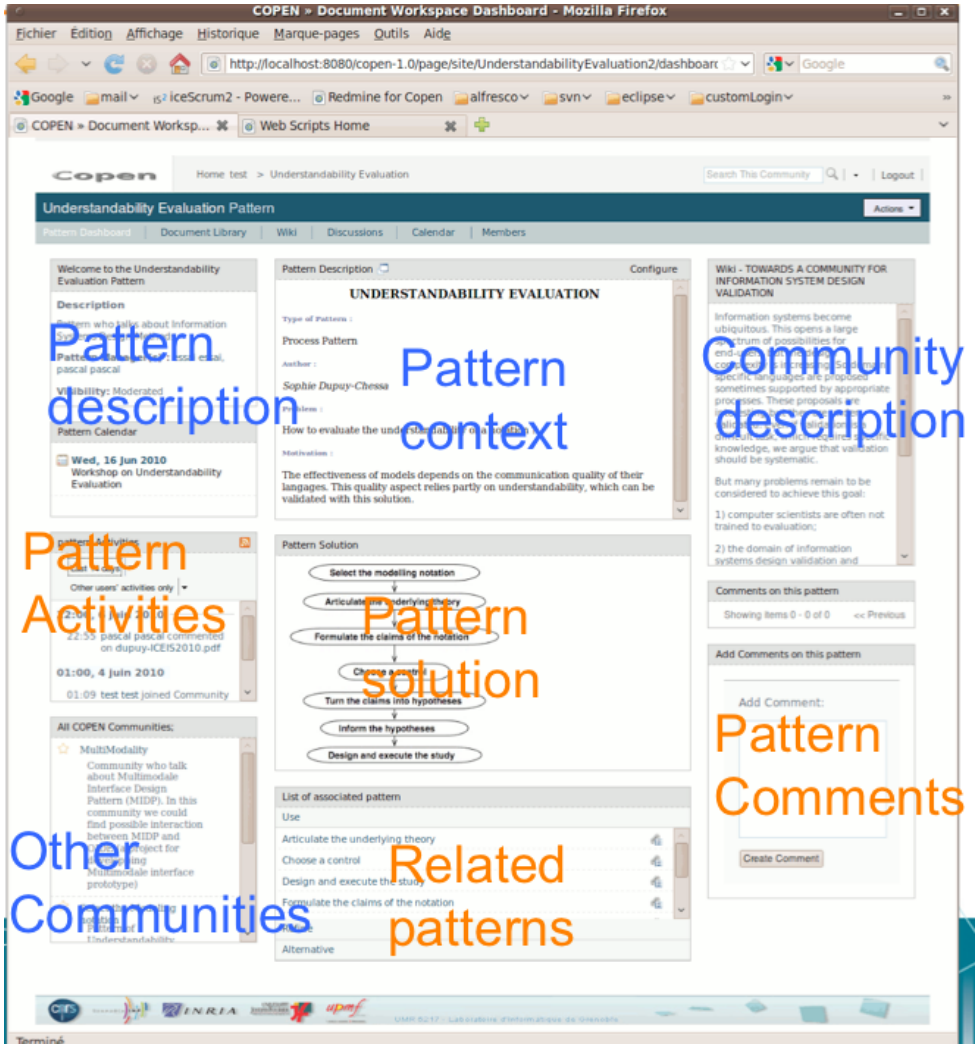


FIGURE 5.10 – *Interface de COpen*

Bilan

D'un point de vue théorique, nous avons introduit le concept de patron collaboratif pour décrire une capitalisation commune de connaissances, basée sur des échanges entre spécialistes d'un domaine. Nous avons débuté le travail de collecte des connaissances pour l'évaluation des modèles et des langages de modélisation en concevant de patrons collaboratifs sous COpen.

Si COpen n'a pour l'instant été utilisé que dans le domaine de l'évaluation, il a été conçu pour permettre à n'importe quelle communauté de partager, maintenir des savoirs et savoir-faire sous la forme de patrons collaboratifs. C'est un outil générique qui permet la collaboration à distance pour la création de patrons, mais aussi pour la prise en compte des retours d'expérience.

COpen est d'ores et déjà disponible sur Internet même s'il est toujours en cours de consolidation. Il s'agit principalement d'améliorer la mise en place du cycle de vie d'un patron collaboratif, mais aussi d'évoluer vers une version plus récente d'Alfresco. Une des limites actuelles de l'outil est

sa gestion des modèles. Les modèles sont considérés comme des images et doivent donc être réalisés dans des éditeurs spécifiques avant d'être inclus dans des patrons au sein desquels ils ne sont plus ni modifiables, ni imitables. Si la gestion des modèles est un point que nous avons identifié comme problématique, d'autres le sont aussi certainement et une évaluation de l'utilisabilité de COpen est prévue pour identifier ses évolutions futures.

5.2.4 Définition de métriques pour l'évaluation de la qualité des langages et des modèles

Approche

Le guide méthodologique proposé dans COpen a pour avantage de regrouper les connaissances pour la qualité des modèles et des langages indépendemment de tout environnement de modélisation. Néanmoins les pratiques d'évaluation qui correspondent à des métriques doivent être automatisées pour faciliter le travail des concepteurs de langages et de modèles. Ainsi de façon similaire aux outils comme JDepend [3], qui calculent des métriques sur le code à la volée pendant le développement, je veux permettre aux concepteurs de calculer des métriques sur la qualité de leur langage ou de leur modèle, leur permettant ainsi d'éviter des erreurs. Les métriques sont donc incluses dans l'environnement de (méta-) modélisation (Pallec et Dupuy-Chessa 2011).

Dans l'environnement, un ensemble de métriques est proposé par défaut aux concepteurs de langages et de modèles, qui peuvent en définir d'autres par eux-mêmes. Notre approche est en ce sens similaire à celle proposée dans (Kersulec et al. 2009). Ainsi nous proposons de gérer la qualité des langages et celle des modèles de manière similaire :

- Au niveau des modèles, les concepteurs de langages déterminent quelles sont les bonnes métriques pour les modèles, instances de leurs méta-modèles.
- Au niveau des langages, les métriques, servant à garder des méta-modèles et leur syntaxe concrète associée compréhensibles, sont définies par des experts en qualité. Ces métriques demeurent modifiables par les concepteurs de langages s'ils les jugent inadéquates.

Notre objectif n'est pas de lister toutes les métriques possibles, mais de montrer l'intérêt de leur intégration à un environnement de (méta-) modélisation.

Inclusion de métriques dans ModX

En collaboration avec Xavier Le Pallec du LIFL, nous avons implémenté quelques métriques dans son générateur d'éditeurs de modèles, ModX (Xavier Le Pallec et Moura 2005). ModX est un outil de modélisation et méta-modélisation créé à Lille en 2004 et basé sur la norme MOF (Meta-Object Facility (OMG 2007b;a)). Cet éditeur a pour but de manipuler graphiquement tout type de modèle dans le domaine de l'Ingénierie Logicielle. Aussi les concepteurs de langage ont la possibilité de définir

3. http ://sourceforge.net/projects/jdepends/

la syntaxe abstraite de leur langage avec un méta-modèle et une syntaxe concrète graphique qui est liée au méta-modèle.

Dans le cadre de nos travaux, nous utilisons ModX pour montrer la faisabilité et l'intérêt d'inclure des métriques de qualité pour les modèles et les langages (Pallec et Dupuy-Chessa 2011). Les métriques pour la qualité des langages se basent sur la définition du méta-modèle, de la notation graphique et de leurs liens. Celles pour la qualité des modèles se calculent à partir des instances du méta-modèle et des éléments graphiques dans le modèle considéré. Ainsi ModX a été étendu pour offrir les fonctionnalités suivantes :

- Calcul de métriques pré-définies sur les modèles et les langages (syntaxes abstraite et concrète). ModX propose des métriques sur les modèles telles que la densité informationnelle d'un modèle qui se mesure en fonction du nombre d'éléments présents dans un diagramme. Ce nombre ne doit pas dépasser le nombre magique de Georges A. Miller (7 ± 2). Au delà de 9 éléments, on peut dire que la densité est trop importante. La fig. 5.11 montre le résultat d'un calcul de densité informationnelle pour un diagramme de cas d'utilisation. La densité est évlauée à 5 c'est-à-dire qu'elle est très bonne pour ce diagramme comme le nombre d'éléments est inférieur à 9. Le mode de calcul de la densité n'a pas été validé ; il est présenté à titre d'illustration de l'outil que nous souhaitons réaliser.
- Ajout de métriques sur les modèles et les et les langages (syntaxes abstraite et concrète). Dans ce cas, les concepteurs de méta-modèles ou les experts en qualité peuvent modifier les fonctions relatives aux métriques pour personnaliser les métriques proposées ou en définir de nouvelles.

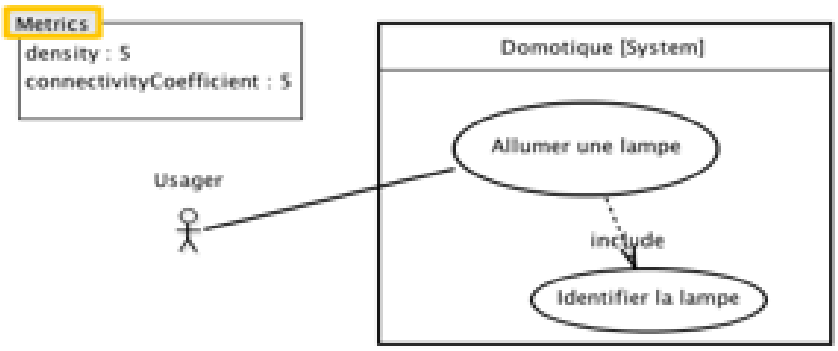


FIGURE 5.11 – *Exemple de calcul de métriques issu de (Pallec et Dupuy-Chessa 2011)*

Bilan

Nous avons présenté comment des métriques pouvaient être intégrées dans un environnement de (méta-)modélisation. L'originalité provient du fait que les concepteurs de langages peuvent définir eux-même les métriques qu'ils souhaitent voir vérifier sur les instances de leurs langages ou que les experts en qualité peuvent tester de nouvelles métriques sur les langages.

Maintenant que la faisabilité de notre approche a été montrée, il convient de vérifier sa pertinence en réalisant des expérimentations auprès de concepteurs de langages. Ces expérimentations serviraient à valider l'intérêt de disposer de métriques sur les langages et de permettre aux concepteurs de définir leurs propres métriques.

Un autre point important pour rendre ModX plus performant est d'enrichir l'ensemble des métriques proposées par défaut dans l'outil. Il nous semble particulièrement important d'en proposer certaines relatives à la qualité des notations visuelles en nous basant des travaux sur la physique des notations (Moody 2009). Ces résultats pourraient ensuite être validées dans le cadre du projet ANR MOANO qui vise notamment à fournir des modèles appropriés pour des botanistes. Nous espérons ainsi faire un pas vers des modèles dédiés aux experts d'un domaine et non plus aux informaticiens.

5.2.5 Apports et Perspectives

Contributions

Mes contributions en matière de qualité des modèles etdes langages de modélisation ont principalement consisté à mettre en place des outils de support à la capitalisation des connaissances d'évaluation et au calcul de métriques de qualité. Un premier pas vers avec une une communauté pour l'évaluation des modèles et des langages a aussi été effectué.

Perspectives

La faisabilité technique ayant été montrée, il est maintenant nécessaire d'étendre l'ensemble des connaissances pour l'évaluation des modèles et des langages. Nous pensons, en particulier, étudier la qualité de la syntaxe concrète graphique des langages, qui est aspect souvent oublié des concepteurs de langages. Cette perspective recoupe celle de l'auto-explication sur l'utilité de coupler les techniques de modélisation avec celles de visualisation pour obtenir des modèles plus utilisables. Ce point nous semble primordial pour aboutir à des modèles manipulables par des experts d'un domaine.

Enfin si les outils que nous proposons semblent répondre à nos besoins, il reste à évaluer leur utilité et leur utilisabilité. Cela peut constituer une réelle limite surtout pour COpen, qui conjugue les difficultés de favoriser un travail collaboratif avec celles liées à des utilisateurs de communautés (donc de cultures) différentes.

Outre l'outil, il faut aussi mesurer la pertinence et l'utilité du concept de patrons collaboratifs. Ayant trouver un concept similaire dans le domaine de la gestion de production, nous envisageons de collaborer sur ce point avec des membres du laboratoire G-Scop de Grenoble. Nous espérons que ce travail montrera l'intérêt de la capitalisation collaborative de connaissances au delà des frontières disciplinaires.

5.2.6 Principaux résultats et encadrements

Principales publications

- S. Dupuy-Chessa, Quality in Ubiquitous Information System Design, 3rd Int. Conf. On research Challenge in Information Science (RCIS'2009), pp 343-352, 2009.
- S. Dupuy-Chessa, D. Rieu, N. Mandran, Towards a community for Information System design Validation, Proc. of the 12th International Conference on Enterprise Information System (ICEIS'2010), pp. 362-367, 2010.

Outils

- COpen : http ://copen.imag.fr/copen-1.0/page/site-index

Encadrements

- Ludovic Bouteloup, Mise en œuvre d'un processus Agile : développement d'un outil de gestion de modèles de conception, stage de mémoire CNAM co-encadré avec Dominique Rieu, 2010
- Pascal Bourgeois, stage de mémoire CNAM co-encadré avec Dominique Rieu, 2010 (mémoire non soutenu).
- Khaoula Sayeb, Réalisation d'un outil de gestion de design patterns collaboratifs, projet de fin d'études d'ingénieur co-encadré avec Dominique Rieu, 2011
- Javier Orozco, Qualité des langages de modélisation, M2R co-encadré avec Nadine Mandran, 2011

Projets

- projet ANR CONTINT MOANO (Modèles et Outils pour Applications NOmades de découverte de territoire), porteur LIUPPA, 2010-2013.
- projet local MSTIC UJF K-IHM (capitalisation de connaissances en IHM), 2009-2010.

Conclusion

6

Autosatisfaction : évaluation erronée.

Ambrose Bierce

Sommaire

Un piste commune à la conception des SI et des interfaces homme-machine a été frayée. Elle doit maintenant se poursuivre pour ouvrir de nouvelles voies pour la suite de mes travaux de recherche.

6.1 Vers un chemin commun ?

Il existe un lien fort entre la modélisation de l'IHM et celle des SI. Ce lien ne se limite pas à l'utilisation de techniques d'ingénierie des modèles. L'IHM et les SI ont des points de vue complémentaires sur les systèmes : l'IHM est centrée sur l'utilisateur alors que le SI est centré sur l'organisation. Ce sont deux éléments indispensables pour réaliser des systèmes adaptés aux utilisateurs, mais aussi à leur contexte de travail.

Mes travaux tirent parti de ces visions complémentaires pour contribuer aussi bien au domaine des SI qu'à celui de l'IHM :

- Symphony étendue et Sonata enrichissent tout autant les pratiques de conception et d'implémentation de l'IHM que celles du noyau fonctionnel, en couplant des points de vue complémentaires.
- QUIMERA, le méta-modèle de qualité, permet de justifier les choix de conception. S'il a été conçu pour l'auto-explication des IHM, il peut aussi être utilisé dans d'autres domaines.
- La modélisation des chorégraphies de services est bien spécifique aux SI. Toutefois les techniques de méta-modélisation pourraient être utilisées à bon escient pour la définition de langages en IHM tels que UsiXML.
- Les pratiques et outils pour la gestion de modèles se veulent génériques et contribuent autant au domaine de l'IHM qu'à celui des SI.

Des pratiques différentes ont été développées et permettent aujourd'hui une fertilisation croisée des connaissances en IHM et en SI. En particulier, l'IHM apporte au SI ses pratiques en matière d'évaluation alors que les SI disposent de connaissances en matière d'ingénierie des processus intéressantes pour l'IHM. C'est pourquoi il me semble intéressant de poursuivre cette route commune en ingénierie des méthodes de conception.

6.2 Perspectives

6.2.1 Gestion de la séparation des préoccupations

En terme de gestion de la séparation des préoccupations de l'IHM et des SI, la méthode Symphony étendue est un premier pas intéressant. Pourtant, comme nous l'avons noté en 3.6.2, la méthode ne couvre pas les IHM adaptables au contexte d'usage. Ce manque sera comblé dans le cadre de la thèse d'Eric Céret qui est co-encadré avec Gaëlle Calvary.

Le travail d'Eric Céret se situe dans le cadre du projet européen UsiXML, qui propose une approche dirigée par les modèles pour la génération d'interfaces. Il utilisera les métamodèles proposés par UsiXML comme modèles de référence, c'est-à-dire comme modèles auxquels il convient de se ramener quel que soit le point de départ souhaité par le concepteur. Par exemple, dans UsiXML, le modèle le plus abstrait est un modèle de tâches. Ce type de modèle est courant en IHM mais peu répandu en dehors de cette communauté. Nous proposerons au concepteur, s'il le souhaite, d'utiliser le workflow auquel il peut être habitué, en mettant à sa disposition des outils de transformation pour ramener ce modèle

vers un modèle de tâches. Tout en apportant un peu de flexibilité au processus, nous maintenons ainsi un principe important de la gestion des préoccupations qui est de conserver les habitudes de travail des concepteurs.

De plus, la colonne vertébrale formée par les modèles UsiXML fournissent les points d'ancrage nécessaires à la mise en correspondance avec les modèles de l'espace métier. Le point dur est de trouver **des mécanismes de mise en correspondance** adaptés à n'importe quel type d'interfaces homme-machine. Comme nous avons pu le constater avec l'utilisation des Objets Symphony, la mise en correspondance entre les espaces métier et interactionnel ne se limite pas en la mise en commun d'éléments de deux modèles (Finkelstein et al. 1992, Anwar et al. 2010). Il ne s'agit pas d'identifier les éléments communs à deux modèles et de les regrouper. La mise en correspondance dont nous avons besoin est basée sur le comportement des différents objets. L'approche par aspects apporte actuellement une solution élégante à ce problème. Toutefois cette solution est-elle toujours envisageable dans le cas d'objets variables qui s'adapteraient au cas d'usage ou dans celui d'une mise en correspondance qui elle-même devrait s'adapter au contexte ? La complexité engendrée risque de rendre une telle solution inexploitable. D'autres pistes devront alors éventuellement être envisagées. L'une d'elles se base sur l'utilisation de services : les Objets Symphony et les Objets Translation donneraient lieu à des services et la découverte automatique de services apporterait la dynamicité nécessaire à l'adaptation. Mais les questions de la complexité et de l'adaptation de la mise en correspondance sont là encore des difficultés à étudier.

6.2.2 Ingénierie des processus

Mes propositions en matière de méthodes de conception me mènent à définir des processus. Comme les miens, de nombreuses descriptions de méthode comportent des modèles de processus. Mais à ma connaissance, peu d'études ont été menées sur la caractérisation des modèles de processus. Pourtant c'est un pas important pour l'ingénierie des processus qui permettrait de mieux cerner les différentes propositions et de pouvoir les comparer. Certains travaux portent sur des synthèses comparatives de méthodes de conception. S'ils se focalisent souvent sur l'approche sous-jacente à la méthode ou sur les bénéfices attendus, ces études incluent tout de même une comparaison des processus. Elles peuvent être divisées en 2 types : d'une part, celles qui incluent une description informelle de quelques méthodes permettant ainsi une comparaison la plupart du temps basée sur les concepts fondamentaux proposés (Sommerville 1996, Hug et al. 2008, Ahmed Seffah 2005) ; d'autre part, celles qui se focalisent sur des critères pour proposer une catégorisation plus formelle (Alexander et Davis 1991, Pérez et al. 1995, Sharon et al. 2010). Nous souhaitons aller plus loin en proposant une taxonomie (Céret 2011) : dans le cadre de la thèse d'Eric Céret, nous avons identifié les similarités principales et les différences entre de nombreux modèles de processus, puis nous avons abstrait des concepts pour définir des catégories d'entités, catégories que nous nommons axes. Chaque axe est lui-même divisé en sous-axes. Ainsi l'axe le plus important est celui du cycle de vie qui a 7 sous-axes : incré-

ment, itération, parallélisme, gestion des retours arrière, durée du cycle, approche (descendante, ascendante, mixte ...), et la focale (activités, produits, décision, contexte et but). Un site internet [1] permet de saisir les caractéristiques d'un modèle de processus pour le comparer de manière graphique ou textuellement avec les autres modèles de processus existants.

Ce travail qui nécessite encore d'être évalué et finalisé m'a permis d'identifier que mes contributions en matière de modèles de processus se situent sur les axes "Collaboration" (i.e. les recommandations des processus pour gérer l'implication des différents acteurs du projet) pour Symphony étendue et "Flexibilité" (c'est-à-dire la capacité d'un processus à s'adapter au contexte du projet et aux besoins) pour le travail sur la réutilisation de processus et d'environnements de modélisation.

L'axe "Flexibilité" est l'un des défis en ingénierie des méthodes. Ainsi de nombreuses solutions à ce problème ont été proposées dont en particulier, l'ingénierie des méthodes situationnelles. C'est avec cette vision que nous avons dirigé nos travaux sur la réutilisation de processus et d'environnements de modélisation. Toutefois, nous avons pu constater que si cette approche est conceptuellement intéressante, elle se heurte à la pratique à cause d'une mise en œuvre souvent complexe et difficile à outiller.

Aussi j'envisage de trouver une approche qui offre un niveau moindre de flexibilité, mais qui soit plus facilement utilisable. Par exemple, (Becker et al. 2007) propose un langage de modélisation de processus spécifique à la gestion des administrations au sein duquel des fragments de processus sont à composer pour obtenir le modèle de processus final. Je pense qu'apporter de la flexibilité au sein de processus spécifique pourrait répondre à mes besoins car se baser sur le processus d'un domaine permet de conserver les bonnes pratiques et les spécificités de ce domaine tout en pouvant adapter le processus au contexte de projet. C'est cette piste que nous allons explorer lors de la thèse d'Eric Céret. Le processus spécifique permettra la conception d'IHM adaptables au contexte avec trois niveaux de flexibilité :

1. Souplesse : le processus se basera sur des modèles de référence servant de colonne vertébrale au processus. Si un concepteur souhaite utiliser d'autres modèles, il pourra le faire mais devra transformer son modèle dans l'un des modèles de la colonne vertébrale.
2. Incomplétude : Tous les modèles ne seront pas nécessaires. Il sera ainsi possible de fournir des points d'entrée à différents niveaux du processus. De plus, des modèles optionnels pourront être remplacés par des modèles par défaut.
3. Orientation but : comme dans nos travaux précédents, la modélisation du processus comprendra un niveau intentionnel lié à un niveau organisationnel afin de permettre d'envisager plusieurs chemins de conception répondant à un objectif identique.

Enfin, des enquêtes montrent que les processus ne sont jamais appliqués comme ils le devraient (Russo et al. 1995, Fitzgerald 1997). Pour mieux cerner les raisons et les impacts de tels écarts, il est nécessaire

1. http ://www.design-methods.net

d'évaluer les résultats d'un processus et dans quelle mesure ce processus a été suivi. Par exemple, (Cook et Wolf 1999) présente une technique pour identifier et mesurer les écarts entre un modèle de processus et ses exécutions. De telles techniques doivent être généralisées à tout modèle de processus quelque soit leur cycle de vie et leur flexibilité. Pour cela, il est nécessaire de pouvoir tracer automatiquement les activités effectuées ou leurs résultats et de les comparer avec le modèle de processus. C'est l'un des points qui devrait être abordé lors de la thèse de Nicolas Hili qui débute en collaboration avec le CEA Leti.

6.2.3 Expert-User Modelling

Dans ce que j'ai envisagé jusqu'à présent en termes de perspectives, ce sont des informaticiens qui suivent les processus afin d'aboutir à des modèles "métier". Ces travaux sont des avancées nécessaires afin de maîtriser les ingénieries des modèles et des processus qu'il faudra mettre en place pour permettre à des experts d'un domaine d'application de comprendre, faire évoluer ou créer leurs propres applications. En effet, dans ce cadre, les informaticiens devront mettre en place des méthodes dédiées à ces experts : les processus devront être adaptés au domaine, mais aussi flexibles pour s'adapter aux compétences des différents experts et aux spécificités des projets ; les modèles utilisés devront pouvoir représenter différents points de vue du système (IHM, organisation etc) ; ces préoccupations devront pouvoir être modifiées et composées afin d'obtenir une solution en adéquation avec les besoins. Je laisse de côté les outils de simulation, de tests etc qu'il faudra mettre en place pour compléter la boîte à outils de ces experts. Pour ma part, je souhaite me concentrer sur l'implication de tels changements pour les concepteurs, qui seront dans un premier temps des informaticiens avant de devenir des experts d'un domaine.

Le premier verrou que j'avais identifié, la dichotomie entre les modèles et le code, n'est pas spécifique au fait que l'on veut que les modèles soient gérés par des experts du domaine d'application. Toutefois, supprimer cette dichotomie est une réelle nécessité pour envisager des modèles modifiables à l'exécution. C'est pourquoi nous avons débuté le travail sur l'auto-explication. Il nécessite de mettre en place des modèles vivants à l'exécution. C'est ce point que nous traitons en priorité dans les thèses d'Alfonso Garcia-Frey et de Mario Cortès-Cornax. Nous envisageons dans ce cadre une approche orientée services dans laquelle les services embarqueraient leurs modèles dans leur description.

Une fois ce problème abordé, il faudra envisager la représentation des modèles afin de fournir aux utilisateurs de modèles compréhensibles. Ce point soulève deux questions principales :

- la première est la qualité des modèles puisque les informations contenues dans le modèle doivent correspondre aux besoins des utilisateurs. Nous avons débuté un travail de capitalisation et d'automatisation des connaissances qui doit être poursuivi pour acquérir des bases solides sur l'évaluation des modèles et des langages. Il me semble en particulier nécessaire d'aborder la question de la syntaxe concrète graphique d'un langage ou "stylistique" : c'est l'un des éléments importants de manipulation des modèles bien qu'il soit

souvent laissé de côté (Fondement et Baar 2005). Si l'utilisation de "boîtes" et de "flèches" peut être enseignée à des informaticiens, elle ne saurait porter du sens pour des experts d'un domaine.

– la deuxième question est la visualisation du modèle qui ne doit pas se limiter à la syntaxe concrète graphique du langage pour rendre les modèles compréhensibles par des utilisateurs non informaticiens. Ainsi de nombreuses techniques d'interaction ou de visualisation pourraient être mises à profit pour faciliter la compréhension et la manipulation des modèles. Par exemple, des techniques de rendu en 3D sont déjà utilisées pour visualiser des liens entre modèles [2].

Ces éléments sont des pas importants pour permettre dans un premier temps la compréhension des modèles. Un chantier plus difficile encore est à envisager pour permettre la modification ou la création de modèles par des non-informaticiens. Aussi j'essaierai d'y contribuer modestement...

2. http ://wiki.eclipse.org/GEF3D$_{S}ample_{A}pplications$

Bibliographie

H. Tardieu A. Rochfeld, R. Colletti. *La méthode Merise, tome 2 : Démarche et pratiques*. Editions d'Organisation, 1985. (Cité pages 4 et 26.)

Michel C. Desmarais Ahmed Seffah, Jan Gulliksen. *Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle*. Springer, 2005. ISBN 978-1-4020-4027-6. (Cité page 100.)

Patrick Albert, Mireille Blay-Fornarino, Philippe Collet, Benoit Combemale, Sophie Dupuy-Chessa, Agnès Front, Anthony Grost, Philippe Lahire, Xavier Le Pallec, Lionel Ledrich, Thierry Nodenot, Anne-Marie Pinna-Dery, et Stèphane Rusinek. End-user modelling. Dans *Actes des deuxièmes journées nationales du Groupement De Recherche CNRS du Génie de la Programmation et du Logiciel*, pages 285–288, 2010. (Cité page 28.)

L. Alexander et A. Davis. Criteria for selecting software process models. Dans *International Computer Software and Applications Conference*, 1991. (Cité page 100.)

Freddy Allilaire et Tarik Idrissi. ADT : Eclipse development tools for ATL. Dans *Proceedings of the Second European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations (EWMDA-2)*, pages 171–178. Computing Laboratory, University of Kent, Canterbury, Kent CT2 7NF, UK, 2004. (Cité pages 18 et 19.)

Adil Anwar, Sophie Ebersold, Bernard Coulette, Mahmoud Nassar, et Abdelaziz Kriouile. A rule-driven approach for composing viewpoint-oriented models. *Journal of Object Technology*, 9(2) :89–114, Mars 2010. ISSN 1660-1769. URL `http://www.jot.fm/contents/issue_2010_03/article1.html`. (Cité page 100.)

Jorge Aranda, Neil Ernst, Jennifer Horkoff, et Steve Easterbrook. A framework for empirical evaluation of model comprehensibility. Dans *Proceedings of the International Workshop on Modeling in Software Engineering*, MISE '07, pages 7–, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2953-4. (Cité pages 15, 86 et 87.)

Robert Balzer. Tolerating inconsistency. Dans *ICSE*, pages 158–165, 1991. URL `http://dblp.uni-trier.de/db/conf/icse/icse91.html#Balzer91`. (Cité page 20.)

Franck Barbier. Enterprise model-driven development with blu age, 2007 ? http ://www.omg.org/mda/mda_files/White_paper_Netfective_technology.pdf. (Cité page 28.)

Mickaël Baron, Vincent Lucquiaud, D. Autard, et Dominique L. Scapin. K-made : un environnement pour le noyau du modèle de description de l'activité. Dans *Proceedings of the 18th International Conference of the Association Francophone d'Interaction Homme-Machine, IHM'2006*, volume 133 de *ACM International Conference Proceeding Series*, pages 287–288. ACM, 2006. (Cité page 83.)

M. F. Barthet et J. C. Tarby. The Diane+ method. Dans J. Vanderdonckt, éditeur, *Computer-aided design of user interfaces*, pages 95–120, Namur, Belgium, 1996. Presses Universitaires de Namur. (Cité page 35.)

J.M. Christian Bastien et Dominique L. Scapin. Ergonomic criteria for the evaluation of human-computer interfaces. Rapport Technique RT-0156, INRIA, Juin 1993. URL `http://hal.inria.fr/inria-00070012/en/`. (Cité pages 61 et 62.)

Michel Beaudouin-Lafon. Designing interaction, not interfaces. Dans *Proceedings of the working conference on Advanced visual interfaces, AVI 2004*, pages 15–22. ACM Press, 2004. (Cité page 21.)

Kent Beck. Embracing change with extreme programming. *Computer*, 32 : 70–77, October 1999. ISSN 0018-9162. (Cité page 4.)

Jorg Becker, Daniel Pfeiffer, et Michael Rackers. Domain specific process modelling in public administrations - the PICTURE-Approach. Dans *Electronic Government*, volume 4656 de *Lecture Notes in Computer Science*, pages 68–79. Springer Berlin / Heidelberg, 2007. (Cité page 101.)

Silvia Berti, Giulio Mori, Fabio Paternò, et Carmen Santoro. A transformation-based environment for designing multi-device interactive applications. Dans *Proceedings of the 9th international conference on Intelligent user interfaces*, IUI '04, pages 352–353, New York, NY, USA, 2004. ACM. (Cité page 83.)

Jean Bézivin et Olivier Gerbé. Towards a precise definition of the OMG/MDA framework. Dans *Proceedings of the 16th IEEE International Conference on Automated Software Engineering*, pages 273–280. IEEE Computer Society, 2001. (Cité page 9.)

Xavier Blanc, Isabelle Mounier, Alix Mougenot, et Tom Mens. Detecting model inconsistency through operation-based model construction. Dans Wilhelm Schafer, Matthew B. Dwyer, et Volker Gruhn, éditeurs, *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008*, pages 511–520. ACM, 2008. ISBN 978-1-60558-079-1. (Cité page 20.)

Sjaak Brinkkemper, Motoshi Saeki, et Frank Harmsen. Assembly techniques for method engineering. Dans *Proceedings of the 10th International Conference on Advanced Information Systems Engineering*, pages 381–400, London, UK, 1998. Springer-Verlag. ISBN 3-540-64556-X. (Cité page 75.)

Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, et Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3) : 289–308, 2003. (Cité pages 20, 21, 22 et 23.)

Stuart K. Card, Allen Newell, et Thomas P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2000. ISBN 0898598591. (Cité page 15.)

Alfonse Carlier. *Management de la qualitÃ© pour la maÃ®trise du S*. Hermès-Lavoisier, 2006. (Cité page 61.)

J. M. Carroll. Scenario-based design. Dans M. Helander, T. K. Landauer, et P. Prabhu, éditeurs, *Handbook of Human-Computer Interaction*, pages 383–406. Elsevier Science B.V., 1997. (Cité page 41.)

P. Caspi, D. Pilaud, N. Halbwachs, et J. A. Plaice. Lustre : a declarative language for real-time programming. Dans *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '87, pages 178–188, New York, NY, USA, 1987. ACM. ISBN 0-89791-215-2. (Cité page 14.)

Eric Céret. Toward a flexible design method sustaining uis plasticity. Dans *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '11, pages 307–310, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0670-6. (Cité page 100.)

Eric Céret, Sophie Dupuy-Chessa, et Guillaume Godet-Bar. Using software metrics in the evaluation of a conceptual component model. Dans *Proceedings of the Fourth IEEE International Conference on Research Challenges in Information Science, RCIS 2010*, pages 507–514. IEEE, 2010. (Cité pages 52, 62 et 86.)

René Chalon et Bertrand T. David. Modélisation de l'interaction collaborative dans les systèmes de réalité mixte. Dans *IHM*, pages 37–44. ACM, 2004. (Cité pages 21 et 35.)

M. Clauss. Generic Modeling using UML extensions for variability. Dans *In Proceedings of OOPSLA Workshop on Domain-specific Visual Languages*, pages 11–18, Tampa, FL, USA, 2001. (Cité page 26.)

OWL-S Coalition. Owl-s specification, 2004. http ://www.daml.org/services/owl-s/1.1/. (Cité page 78.)

Larry L. Constantine, Robert Biddle, et James Noble. Usage-centered design and software engineering : Models for integration. Dans *Proceedings of ICSE 2003 Workshop on Bridging the Gaps Between Software Engineering and Human-Computer Interaction, ICSE Workshop on SE-HCI*, pages 106–113, 2003. (Cité pages 22, 35 et 41.)

Jonathan E. Cook et Alexander L. Wolf. Software process validation : quantitatively measuring the correspondence of a process to a model. *ACM Trans. Softw. Eng. Methodol.*, 8 :147–176, April 1999. ISSN 1049-331X. (Cité page 102.)

Daniel Corlett. Design : innovating with ovid. *interactions*, 7 :19–26, July 2000. ISSN 1072-5520. (Cité page 35.)

Mario Cortes-Cornax. *Des chorégraphies de services vers les processus métier*. Rapport de M2R, Grenoble Institut National Polytechnique, 2010. (Cité page 68.)

Mario Cortes-Cornax. Service choreographies through a graphical notation based on abstraction layers and viewpoints. Dans *Proceedings of the 5th IEEE International Conference on Research Challenges in Information Science, RCIS 2011*. IEEE, 2011. (Cité pages 59, 68 et 69.)

Mario Cortes-Cornax, Sophie Dupuy-Chessa, et Dominique Rieu. Bridging the gap between business processes and service composition through service choreographies. Dans *IFIP WG8.1 Working conference on Method Engineering (ME'2011)*, 2011a. (Cité pages 66, 67 et 68.)

Mario Cortes-Cornax, Sophie Dupuy-Chessa, Dominique Rieu, et Marlon Dumas. Evaluating choreographies in bpmn 2.0 using an extended quality framework. Dans *Proceedings o the 3rd International Workshop on the Business Process Model and Notation, BPMN 2011*, LNBIP. Springer-Verlag, 2011b. (Cité pages 69 et 86.)

Joëlle Coutaz. PAC, on object oriented model for dialog design. Dans *Interact'87*, 1987. 6 pages. (Cité pages 35 et 44.)

Joëlle Coutaz. User interface plasticity : model driven engineering to the limit ! Dans *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing System, EICS 2010*, pages 1–8. ACM, 2010. (Cité pages 20, 21 et 28.)

Céline Coutrix et Laurence Nigay. Mixed reality : a model of mixed interaction. Dans *Proceedings of the working conference on Advanced visual interfaces, AVI 2006*, pages 43–50. ACM Press, 2006. (Cité pages 21 et 35.)

Gyorgy Csertan, Gabor Huszerl, Istvan Majzik, Zsigmond Pap, Andras Pataricza, et Daniel Varro. VIATRA – Visual Automated Transformations for Formal Verification and Validation of UML Models. Dans *Proceedings of the 17th IEEE International Conference on Automated Software Engineering*, pages 267–270. IEEE Computer Society, 2002. (Cité page 19.)

Krzysztof Czarnecki et Simon Helsen. Classification of model transformation approaches. Dans *OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture*, 2003. (Cité page 18.)

Andy Dearden et Janet Finlay. Pattern languages in hci : A critical review. *Human-Computer Interaction*, 21 :49–102, 2006. (Cité page 27.)

R. Geoff Dromey. Cornering the chimera. *IEEE Softw.*, 13 :33–43, January 1996. ISSN 0740-7459. (Cité page 61.)

Emmanuel Dubois. *Conception, Implémentation et Evaluation de Systèmes Interactifs Mixtes : une Approche basée Modèles et Centrée sur l'Interaction*. Habilitation à diriger des recherches, Université de Toulouse, 2009. (Cité pages 20, 21 et 35.)

Emmanuel Dubois, Laurence Nigay, J. Troccaz, O. Chavanon, et L. Carrat. Classification space for augmented surgery, an augmented reality case study. Dans A. Sasse et C. Johnson, éditeurs, *Proceedings of INTERACT'99*, pages 353–359, Edinburgh, UK, 1999. (Cité page 35.)

Sophie Dupuy, Yves Ledru, et Monique Chabre-Peccoud. Vers une intégration utile de notations semi-formelles et formelles : une expérience en uml et z. *L'OBJET*, 6(1), 2000. (Cité page 14.)

Sophie Dupuy-Chessa. Quality in ubiquitous information systems design. Dans *Proceedings of the Third IEEE International Conference on Research Challenges in Information Science, RCIS'2009*, pages 343–352. IEEE, 2009. (Cité page 51.)

Sophie Dupuy-Chessa, Guillaume Godet-Bar, David Juras, et Dominique Rieu. Principes pour une méthode de conception de systèmes mixtes. Dans *Proceedings of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine*, IHM '07, pages 75–82, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-791-9. (Cité page 36.)

Sophie Dupuy-Chessa, Guillaume Godet-Bar, Jorge-Luis Pérez-Medina, Dominique Rieu, et David Juras. A software engineering method for the design of mixed reality systems. Dans *The Engineering of Mixed Reality Systems, chapter 15*. Springer, 2009. eds E. Dubois, P. Gray and L. Nigay. (Cité pages 37, 38 et 39.)

Sophie Dupuy-Chessa, Nadine Mandran, Guillaume Godet-Bar, et Dominique Rieu. A case Study for Improving a Collaborative Design Process. Dans *IFIP WG8.1 Working confÃ©rence on Method Engineering (ME'2011)*, 2011. (Cité pages 39, 51, 52 et 86.)

Sophie Dupuy-Chessa, Dominique Rieu, et Nadine Mandran. Towards a community for information system design validation. Dans *Proceedings of the 12th International Conference on Enterprise Information Systems, ICEIS'2010, Volume 3*, pages 362–367, 2010. (Cité pages 87, 88, 89 et 90.)

R. Eckstein. Java SE application design with MVC, 2007. http ://java.sun.com/developer/technicalArticles/javase/mvc/. (Cité page 53.)

Clarence A. Ellis, Simon J. Gibbs, et Gail Rein. Groupware : some issues and experiences. *Commun. ACM*, 34 :39–58, January 1991. ISSN 0001-0782. (Cité page 82.)

Marcos Didonet Del Fabro et Frédéric Jouault. Model Transformation and Weaving in the AMMA Platform. Dans *Proceedings of the International Summer School in Generative and Transformational Techniques in Software Engineering (GTTSE'05)*, volume 4143 de *LNCS*, pages 71–77. Springer Verlag, 2007. (Cité page 19.)

Jean-Marie Favre. Towards a basic theory to model model driven engineering. Dans *In Workshop on Software Model Engineering, WISME 2004, joint event with UML2004*, 2004. (Cité page 9.)

Jean-Marie Favre, Jacky Estublier, et Mireille Blay-Fornarino. *L'Ingénierie Dirigée par les Modèles, au-delà du MDA*. Hermès-Lavoisier, February 2006. (Cité pages 10, 13, 28 et 29.)

Anthony Finkelstein, Jeff Kramer, Bashar Nuseibeh, L. Finkelstein, et Michael Goedicke. Viewpoints : A framework for integrating multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering*, 2(1) :31–57, 1992. (Cité pages 19 et 100.)

Brian Fitzgerald. The use of systems development methodologies in practice : a field study. *Inf. Syst. J.*, 7(3) :201–212, 1997. (Cité page 101.)

Frédéric Fondement et Thomas Baar. Making metamodels aware of concrete syntax. Dans *Model Driven Architecture - Foundations and Applications, First European Conference (ECMDA-FA 2005)*, pages 190–204, 2005. (Cité pages 14 et 103.)

David Fox, Jonathan Sillito, et Frank Maurer. Agile methods and user-centered design : How these two methodologies are being successfully integrated in industry. Dans *Proceedings of the Agile 2008*, pages 63–72, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3321-6. (Cité page 35.)

Martin D. Fraser, Kuldeep Kumar, et Vijay K. Vaishnavi. Strategies for incorporating formal specifications in software development. *Commun. ACM*, 37 :74–86, October 1994. ISSN 0001-0782. (Cité page 14.)

Alfonso García Frey, Gaëlle Calvary, et Sophie Dupuy-Chessa. Xplain : an editor for building self-explanatory user interfaces by model-driven engineering. Dans *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing System, EICS 2010*, pages 41–46. ACM, 2010. (Cité page 59.)

Alfonso García Frey, Eric Ceret, Sophie Dupuy-Chessa, et Gaëlle Calvary. QUIMERA - toward an unifying quality metamodel. position paper, 2011a. (Cité page 62.)

Alfonso García Frey, Eric Ceret, Sophie Dupuy-Chessa, et Gaëlle Calvary. Quimera : a quality metamodel to improve design rationale. Dans *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing System, EICS 2011*, pages 265–270. ACM, 2011b. (Cité pages 61 et 63.)

Agnès Front, Dominique Rieu, et Jean-Pierre Giraudin. Une vision sur les problématiques actuelles de la recherche en systèmes d'information, 2009. http ://sigma.imag.fr/enjeux-des-nouveaux-systemes-d-information/blog. (Cité pages 3 et 25.)

Elizabeth Furtado, Vasco Furtado, Kênia Soares Sousa, Jean Vanderdonckt, et Quentin Limbourg. KnowiXML : a knowledge-based system generating multiple abstract user interfaces in USIXML. Dans *Proceedings of the 3rd annual conference on Task models and diagrams*, TAMODIA '04, pages 121–128. ACM, 2004. ISBN 1-59593-000-0. (Cité page 20.)

Erich Gamma, Richard Helm, Ralph E. Johnson, et John Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1995. (Cité page 27.)

Guillaume Godet-Bar. *Spécification et outillage d'une méthode de conception des systèmes de réalité mixte*. Doctorat en informatique, Grenoble Institut National Polytechnique, 2009. (Cité pages 37, 40, 41, 42 et 52.)

Guillaume Godet-Bar, Sophie Dupuy-Chessa, et Dominique Rieu. When interaction choices trigger business evolutions. Dans *Advanced Information Systems Engineering, 20th International Conference, CAiSE 2008*, pages 144–147, 2008. (Cité page 39.)

Guillaume Godet-Bar, Sophie Dupuy-Chessa, et Dominique Rieu. Sonata : Flexible connections between interaction and business spaces. *System and Software*, indéfini(indéfini) :en cours de révision, soumis. (Cité pages 43, 44, 45, 46, 47, 49 et 50.)

Guillaume Godet-Bar, David Juras, Sophie Dupuy-Chessa, et Dominique Rieu. Vers une méthode de conception de systèmes mixtes. principes et mise en œuvre. *Ingénierie des Systèmes d'Information*, 12(6) :39–66, 2007a. (Cité pages 36 et 39.)

Guillaume Godet-Bar, Dominique Rieu, et Sophie Dupuy-Chessa. HCI and business practices in a collaborative method for augmented reality systems. *Information & Software Technology*, 52(5) :492–505, 2010. (Cité pages 36 et 39.)

Guillaume Godet-Bar, Dominique Rieu, Sophie Dupuy-Chessa, et David Juras. Interactional Objects : HCI Concerns in the Analysis Phase of the Symphony Method. Dans *Proceedings of the Ninth International Conference on Enterprise Information Systems, Volume HCI, ICEIS'2007 (5)*, pages 37–44, 2007b. (Cité page 42.)

K. Grebici, E. Blanco, et Dominique Rieu. Toward non mature information management in collaborative design processes. Dans *Proceedings of the International Conference on Engineering Design ICED'05*, Melbourne, Australia, /Aug 2005. (Cité pages 38 et 81.)

J. Gulliksen, B. Goransson, I. Boivie, J. Persson, S. Blomkvist, et A. Cajander. Key principles for user-centred systems design. Dans *Human-Centered Software Engineering â Integrating Usability in the Software Development Lifecycle*. Springer Publishing Company, Incorporated, 2005. (Cité page 35.)

Gwladys Guzelian et Corine Cauvet. SO2M : Towards a service-oriented approach for method engineering. Dans *the 2007 World Congress in Computer Science, Computer Engineering and Applied Computing, in the proceedings of International Conference Information and Knowledge Engineering IKE'07*, Las Vegas, Nevada, USA, 2007. (Cité pages 75, 78 et 81.)

Michael Hammer et James Campy. *Le Reengineering*. Dunod, 1993. (Cité page 64.)

Ibtissem Hassine. *Spécification et formalisation des démarches de développement à base de composants métier : la démarche Symphony*. Doctorat en informatique, Grenoble Institut National Polytechnique, 2005. (Cité pages 24, 25 et 26.)

Ibtissem Hassine, Dominique Rieu, Fethi Bounaas, et Omar Seghrouchni. Symphony : un modèle conceptuel de composants métier. *Ingénierie des Systèmes d'Information*, 7(4) :33–59, 2002. (Cité page 42.)

Jim Highsmith et Martin Fowler. The agile manifesto. *Software Development Magazine*, 9(8) :29–30, 2001. (Cité page 4.)

Charlotte Hug, Agnès Front, et Dominique Rieu. A process engineering method based on a process domain model and patterns. Dans *Proceedings of the International Workshop on Model Driven Information Systems Engineering : Enterprise, User and System Models (MoDISE-EUS'o8) held in conjunction with the CAiSE'o8 Conference*, pages 126–137, 2008. (Cité page 100.)

ISO. ISO/IEC 13407 : 1999 (E) Human-Centred Design Processes for Interactive Systems. 1999. (Cité page 22.)

ISO. ISO/IEC 9126 : Software engineering â Product quality - Part 1 : Quality model. 2001. (Cité page 61.)

Ivar Jacobson, Magnus Christerson, Patrik Jonsson, et Gunnar Overgaard. *Object Oriented Software Engineering : A Use Case Driven Approach*. Addison-Wesley, 1992. (Cité page 4.)

Gwenaël Kersulec, Samira Si-Said Cherfi, Isabelle Comyn-Wattiau, et Jacky Akoka. Un environnement pour lévaluation et lamélioration de la qualité des modèles de systèmes dinformation. Dans *Actes du XX-VIIème Congrès INFORSID, INFORSID'2009*, pages 329–344, 2009. (Cité page 92.)

Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean marc Loingtier, et John Irwin. Aspect-oriented programming. Dans *European Conference on Object-Oriented Programming, ECOOP'97*. SpringerVerlag, 1997. (Cité page 45.)

Soon-Kyeong Kim et David Carrington. Formalizing the uml class diagram using object-z. Dans *Proceedings of the 2nd international conference on The unified modeling language : beyond the standard*, UML'99, pages 83–98, Berlin, Heidelberg, 1999. Springer-Verlag. ISBN 3-540-66712-1. (Cité page 14.)

A. Kleppe, J. Warmer, et W. Bast. *MDA Explained : The Model Driven Architecture - Practice and Promise*. Addison-Wesley, 2003. (Cité page 9.)

A. G. Kleppe. A language description is more than a metamodel. Dans *Fourth International Workshop on Software Language Engineering, Nashville, USA*, Grenoble, France, October 2007. megaplanet.org. ISBN not assigned. (Cité pages 13 et 14.)

J. Koehler et B. Srivastava. Web service composition : Current solutions and open problems. Dans *ICAPS 2003 Workshop on Planning for Web Services*, pages 28–35, 2003. URL `http://www.informatik.uni-freiburg.de/~koehler/bpia/icaps-ws.pdf`. (Cité page 64.)

G. Krasner et S. Pope. A description of the Model-View-Controller user interface paradigm in the smalltalk-80 system. *Journal of Object Oriented Programming*, 1(3) :26–49, 1988. (Cité page 35.)

John Krogstie. Integrating the understanding of quality in requirements specification and conceptual modeling. *SIGSOFT Softw. Eng. Notes*, 23 : 86–91, January 1998. ISSN 0163-5948. (Cité page 12.)

John Krogstie. Evaluating uml using a generic quality framework. Dans *UML and the unified process*, pages 1–22. IGI Publishing, Hershey, PA, USA, 2003. ISBN 1-931777-44-6. (Cité pages 12, 15, 16, 17 et 86.)

John Krogstie et Håvard D. Jørgensen. Quality of interactive models. Dans *ER (Workshops)*, volume 2503 de *Lecture Notes in Computer Science*, pages 351–363. Springer, 2002. (Cité page 28.)

Philippe Kruchten. *The Rational Unified Process : An Introduction*. Addison-Wesley Longman Publishing Co., Inc., 3 édition, 2003. ISBN 0321197704. (Cité page 26.)

Régine Laleau et Amel Mammar. From UML diagrams to B specifications. Dans Marc Frappier et Henri Habrias, éditeurs, *Software Specification Methods : an Overview Using a Case Study*, pages 60–79. Hermes Science Publishing, ISTE London, 2006. ISBN 1-905209-34-7. (Cité page 14.)

Christian F. J. Lange et Michel R. V. Chaudron. Managing model quality in uml-based software development. Dans *Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice*, pages 7–16, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2639-X. (Cité pages 10, 11 et 86.)

P. Lécaille. *La trace-habilitée, une éthnographie des espaces de conception dans un bureau d'études mécanique : l'échange et l'équipement des objets graphonumériques entre outils et acteurs de la conception*. Doctorat, Institut National Polytechnique de Grenoble, 2003. (Cité page 87.)

Odd Ivar Lindland, Guttorm Sindre, et Arne Sølvberg. Understanding quality in conceptual modeling. *IEEE Softw.*, 11 :42–49, March 1994. ISSN 0740-7459. (Cité pages 11 et 86.)

Allan MacLean, Richard M. Young, Victoria M. E. Bellotti, et Thomas P. Moran. Questions, options, and criteria : elements of design space analysis. *Hum.-Comput. Interact.*, 6 :201–250, September 1991. ISSN 0737-0024. (Cité page 60.)

Adel Mahfoudhi, Wided Bouchelligua, Mourad Abed, et Mohamed Abid. Towards a new approach of model-based hci conception. Dans *Proceedings of the 6th WSEAS International Conference on Multimedia, Internet &*

Video Technologies, pages 117–125, Stevens Point, Wisconsin, USA, 2006. World Scientific and Engineering Academy and Society (WSEAS). ISBN 960-8457-53-X. (Cité page 20.)

David. Marca et Clement L. McGowan. *SADT : structured analysis and design technique / David A. Marca, Clement L. McGowan ; with a foreword by Douglas T. Ross*. McGraw-Hill, New York, 1988. ISBN 0070402353. (Cité page 4.)

J McCall. *Factors in Software Quality : Preliminary Handbook on Software Quality for an Acquisiton Manager*, volume 1-3. General Electric, 1977. (Cité page 61.)

Kashif Mehmood, Samira Si-Said Cherfi, et Isabelle Comyn-Wattiau. Data quality through conceptual model quality - reconciling researchers and practitioners through a customizable quality model. Dans *Proceedings of the 14th International Conference on Information Quality, ICIQ 2009*, pages 61–74, 2009. (Cité page 61.)

Nikola Milanovic et Miroslaw Malek. Current solutions for web service composition. *IEEE Internet Computing*, 8 :51–59, November 2004. ISSN 1089-7801. (Cité page 64.)

M. L. Minsky. Matter, mind and models. Dans *International Federation of Information Processing Congress, New-York, USA*, volume 1, pages 45–49, 1965. (Cité page 9.)

Isabelle Mirbel et Pierre Crescenzo. Des besoins des utilisateurs à la recherche de services web. une approche sémantique guidée par les intentions. *Ingénierie des Systèmes d'Information*, 15(4) :89–112, 2010. (Cité page 76.)

Parastoo Mohagheghi et Jan Aagedal. Evaluating quality in model-driven engineering. Dans *Proceedings of the International Workshop on Modeling in Software Engineering*, MISE '07, pages 6–, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2953-4. (Cité pages 15 et 61.)

Parastoo Mohagheghi, Vegard Dehlen, et Tor Neple. Definitions and approaches to model quality in model-based software development - a review of literature. *Inf. Softw. Technol.*, 51 :1646–1669, December 2009. ISSN 0950-5849. (Cité pages 11, 15 et 86.)

Daniel L. Moody. Theoretical and practical issues in evaluating the quality of conceptual models : current state and future directions. *Data Knowl. Eng.*, 55(3) :243–276, 2005. (Cité page 10.)

Daniel L. Moody. The "physics" of notations : Toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Software Eng.*, 35(6) :756–779, 2009. (Cité pages 15, 67, 87 et 94.)

Daniel L. Moody, Guttorm Sindre, Terje Brasethvik, et Arne Sølvberg. Evaluating the quality of information models : empirical testing of a conceptual model quality framework. Dans *Proceedings of the 25th International*

Conference on Software Engineering, ICSE '03, pages 295–305, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1877-X. (Cité page 12.)

Sebastien Mosser, Alexandre Bergel, et Mireille Blay-Fornarino. Visualizing and assessing a compositional approach of business process design. Dans *Proceedings of the 9th international conference on Software composition*, SC'10, pages 90–105, Berlin, Heidelberg, 2010. Springer-Verlag. (Cité page 69.)

Brad Myers, Scott E. Hudson, et Randy Pausch. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact.*, 7 : 3–28, March 2000. ISSN 1073-0516. (Cité page 59.)

Donald A. Norman et Stephen W. Draper. *User Centered System Design ; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1986. ISBN 0898597811. (Cité page 22.)

Nuno Jardim Nunes et Joao Falccao e Cunha. Wisdom - whitewater interactive system development with object models. Dans *Object modeling and user interface design*, pages 197–243. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. ISBN 0-201-65789-9. (Cité page 35.)

OASIS. Business process execution language for web services (BPEL4WS), version 1.1, 2003. URL `http://www.ibm.com/developerworks/library/specification/ws-bpel/`. (Cité page 64.)

OMG. Business process management notation (BPMN 2.0), 2011. URL `http://www.omg.org/spec/BPMN/2.0/`. (Cité pages 26 et 66.)

Object Management Group OMG. MOF QVT Final Adopted Specification, 2005. http ://www.omg.org/spec/QVT/1.0/. (Cité page 18.)

Object Management Group OMG. Unified modeling language 2.1.2 super-structure specification, November 2007a. http ://www.omg.org/docs/formal/07-11-04.pdf. (Cité pages 13 et 92.)

Object Management Group OMG. Unified modeling language (uml) 2.1.2 infrastructure, November 2007b. http ://www.omg.org/docs/formal/07-11-04.pdf. (Cité pages 13 et 92.)

Object Management Group OMG. OCL 2.2 Specification, 2010. http ://www.omg.org/spec/OCL/2.2. (Cité page 19.)

Oracle. ORACLE BPEL Process Manager, dernière consultation 07/2011. URL `http://www.oracle.com/technology/products/ias/bpel/index.html`. (Cité page 64.)

Xavier Le Pallec et Sophie Dupuy-Chessa. Intégration de métriques de qualité des modèles et des méta-modèles dans lâoutil modx. position paper, 2011. (Cité pages 92 et 93.)

Stephen R. Palmer et John M. Felsing. *A Practical Guide to Feature-Driven Development (The Coad Series)*. Prentice Hall PTR, Février 2002. ISBN 0130676152. (Cité page 26.)

Fabio Paterno. *Model-Based Design and Evaluation of Interactive Applications*. Springer-Verlag, London, UK, 1st édition, 1999. ISBN 1852331550. (Cité page 20.)

Fabio Paterno. Concurtasktrees : An engineered notation for task models. Dans *The Handbook of Task Analysis for HumanComputer Interaction*, pages 483–503. Lawrence Erlbaum Associates, 2003. (Cité page 21.)

Fabio Paterno, C. Mancini, et S. Meniconi. Concurtasktree : a diagrammatic notation for specifying task models. Dans *Proceedings of INTERACT'97*, pages 362–369, 1997. (Cité pages 20 et 83.)

Susanne Patig. A practical guide to testing the understandability of notations. Dans *Proceedings of the fifth Asia-Pacific conference on Conceptual Modelling - Volume 79*, APCCM '08, pages 49–58, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc. ISBN 978-1-920682-60-6. (Cité pages 15, 86 et 87.)

Graciela Pérez, Khaled El Emam, et Nazim H. Madhavji. Customising software process models. Dans *Proceedings of the 4th European Workshop on Software Process Technology*, EWSPT '95, pages 70–78, London, UK, 1995. Springer-Verlag. ISBN 3-540-59205-9. (Cité page 100.)

Jorge-Luis Perez-Medina. *Approche orientée services pour la réutilisation de processus et d'outils de modélisation*. Doctorat en informatique, Université Joseph Fourier, 2010. (Cité pages 10, 13 et 18.)

Jorge Luis Pérez-Medina, Sophie Dupuy-Chessa, et Agnès Front. A survey of model driven engineering tools for user interface design. Dans *Task Models and Diagrams for User Interface Design, 6th International Workshop, TAMODIA 2007*, volume 4849 de *Lecture Notes in Computer Science*, pages 84–97. Springer, 2007. (Cité page 20.)

Jorge Luis Pérez-Medina, Sophie Dupuy-Chessa, et Dominique Rieu. Approche orientée services pour la construction des environnements de modélisation. *Ingénierie des Systèmes d'Information*, 15(4) :113–137, 2010a. (Cité pages 75, 76, 78, 79, 81, 82 et 85.)

Jorge Luis Pérez-Medina, Sophie Dupuy-Chessa, et Dominique Rieu. A service-oriented approach for interactive system design. Dans *Task Models and Diagrams for User Interface Design, 8th International Workshop, TAMODIA 2009, Revised Selected Papers*, volume 5963 de *Lecture Notes in Computer Science*, pages 44–57. Springer, 2010b. (Cité page 84.)

Jolita Ralyté. Reusing scenario based approaches in requirement engineering methods : Crews method base. Dans *Proceedings of the 10th International Workshop on Database & Expert Systems Applications*, DEXA '99, pages 305–, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0281-4. (Cité page 77.)

Jolita Ralyté et Colette Rolland. An assembly process model for method engineering. Dans *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, CAiSE '01, pages 267–283, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42215-3. (Cité page 75.)

David Raneburger. Interactive model driven graphical user interface generation. Dans *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '10, pages 321–324. ACM, 2010. ISBN 978-1-4503-0083-4. (Cité page 20.)

Philippe Renevier. *Systèmes Mixtes Collaboratifs sur Supports Mobiles : Conception et Réalisation*. Thèse de doctorat, Université de Grenoble, Grenoble, 2004. (Cité page 37.)

Colette Rolland. L'ingénierie des méthodes : une visite guidée. *E-revue En Technologies De l'Information (e-TI)*, 2005. URL `http://www.revue-eti.net/document.php?id=726`. (Cité pages 24, 26 et 75.)

Colette Rolland. Method engineering : towards methods as services. Dans *Proceedings of the Software process, 2008 international conference on Making globally distributed software development a success story*, ICSP'08, pages 10–11, Berlin, Heidelberg, 2008. Springer-Verlag. (Cité page 75.)

Colette Rolland, Naveen Prakash, et A. Benjamen. A multi-model view of process modelling. *Requirements Engineering*, 4(4) :169–187, 1999. URL `http://dblp.uni-trier.de/db/journals/re/re4.html#RollandPB99`. (Cité page 25.)

Matti Rossi et Sjaak Brinkkemper. Complexity metrics for systems development methods and techniques. *Information Systems*, 21(2) :209–227, 1996. (Cité pages 15 et 86.)

James E. Rumbaugh, Michael R. Blaha, William J. Premerlani, Frederick Eddy, et William E. Lorensen. *Object-Oriented Modeling and Design*. Pentice Hall International, 1991. (Cité page 4.)

Nancy L. Russo, Judy L. Wynekoop, et Diane B. Walz. The use and adaptation of system development methodologies. Dans *International Resources Management Association, International Conference*, 1995. http ://www.andrews.edu/ vyhmeisr/papers/sdm.html. (Cité page 101.)

Rajaa Saidi, Nicolas Arnaud, Dominique Rieu, et Mounia Fredj. Multi-view variability modelling for business component reuse. Dans *Second IEEE International Conference on Digital Information Management, ICDIM'07*, pages 603–608, 2007. (Cité page 26.)

Rajaa Saidi, Mounia Fredj, Agnès Front, et Salma Mouline. Variabilité dans les composants métiers multivues. *Ingénierie des Systèmes d'Information*, 14(2) :61–86, 2009. (Cité page 54.)

Tony Salvador, Jean Scholtz, et James Larson. The denver model for groupware design. *SIGCHI Bull.*, 28 :52–58, January 1996. ISSN 0736-6906. (Cité page 82.)

Ken Schwaber et Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st édition, 2001. ISBN 0130676349. (Cité page 4.)

Ed Seidewitz. What models mean. *IEEE Software*, 20(5) :26–32, 2003. (Cité page 9.)

P. Seligmann, G. Wijers, et H. Sol. Analyzing the structure of I.S. methodologies, an alternative approach. Dans R. Maes, éditeur, *Proceedings of the First Dutch Conference on Information Systems*, 1989. (Cité page 4.)

Itamar Sharon, Michel dos Santos Soares, Joseph Barjis, Jan van den Berg, et Jos L. M. Vrancken. A decision framework for selecting a suitable software development process. Dans *ICEIS 2010 - Proceedings of the 12th International Conference on Enterprise Information Systems, Volume 3*, pages 34–43, 2010. (Cité page 100.)

Keng Siau et Yuhong Tian. The complexity of unified modeling language : A goms analysis. Dans *Proceedings of the International Conference on Information Systems (ICIS 2001)*, pages 443–448, 2001. (Cité page 15.)

Ian Sommerville. Software process models. *ACM Comput. Surv.*, 28(1) : 269–271, 1996. (Cité page 100.)

Jean-Sebastien Sottet, Gaëlle Calvary, Joëlle Coutaz, et Jean-Marie Favre. A model-driven engineering approach for the usability of plastic user interfaces. Dans *Engineering Interactive Systems - EIS 2007 Joint Working Conferences, EHCI 2007, DSV-IS 2007, HCSE 2007*, volume 4940 de *Lecture Notes in Computer Science*, pages 140–157. Springer, 2008. (Cité page 24.)

Kênia Soares Sousa et Elizabeth Furtado. An approach to integrate hci and se in requirements engineering. Dans M. Borup Harning et Jean Vanderdonckt, éditeurs, *Proceedings of the IFIP TC13 workshop on Closing the gaps : Software engineering and Human-Computer Interaction*, 2003. (Cité page 35.)

Kênia Soares Sousa, Hildeberto Mendonca, et Jean Vanderdonckt. Towards method engineering of model-driven user interface development. Dans *Proceedings of the 6th international conference on Task models and diagrams for user interface design*, TAMODIA'07, pages 112–125. Springer-Verlag, 2007. ISBN 3-540-77221-9, 978-3-540-77221-7. (Cité pages 22 et 80.)

J. M. Spivey. *The Z notation : a reference manual*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1992. ISBN 0-13-978529-9. (Cité page 14.)

Allistair Sutcliffe. Convergence or competition between software engineering and human computer interaction. Dans Ahmed Seffah, Jan Gulliksen, et Michel C. Desmarais, éditeurs, *Human-Centered Software Engineering â Integrating Usability in the Software Development Lifecycle*, volume 8 de *Human-Computer Interaction Series*, pages 71–84. Springer Netherlands, 2005. ISBN 978-1-4020-4113-6. (Cité page 41.)

Bhuvan Unhelkar. *Verification and Validation for Quality of UML 2.0 Models*. Wiley-Interscience, 2005. ISBN 0471727830. (Cité pages 11 et 86.)

Axel van Lamsweerde. Goal-oriented requirements engineering : A guided tour. Dans *5th IEEE International Symposium on Requirements Engineering, RE 2001*, page 249. IEEE Computer Society, 2001. (Cité page 25.)

Lode Vanacken, Erwin Cuppens, Tim Clerckx, et Karin Coninx. Extending a dialog model with contextual knowledge. Dans *Proceedings of the 6th international conference on Task models and diagrams for user interface design*, TAMODIA'07, pages 28–41. Springer-Verlag, 2007. ISBN 3-540-77221-9, 978-3-540-77221-7. (Cité pages 20 et 21.)

Jean Vanderdonckt. Model-driven engineering of user interfaces : Promises, successes, failures and challenges. Dans *Proceedings of the 5th Annual Romanian Conference on Human-Computer Interaction, ROCHI'2008*, pages 1–10, 2008. (Cité pages 21 et 28.)

W3C. Web services choreography description language (WS-CDL), version 1.0, 2005. URL `http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/`. (Cité page 64.)

M. N. Wicks et R. G. Dewar. Controversy corner : A new research agenda for tool integration. *Journal of System and Software*, 80 :1569–1585, September 2007. ISSN 0164-1212. (Cité page 75.)

E. Renaux Xavier Le Pallec et C.O. Moura. ModX - a graphical tool for MOF metamodels. Dans *European Conference on Model Driven Architecture - Foundations and Applications, ECMDA-FA'2005*, pages –, 2005. Tools Exhibition. (Cité page 92.)

Jianqi Yu, Philippe Lalanda, et Stéphanie Chollet. Development tool for service-oriented applications in smart homes. Dans *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2*, pages 239–246, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3283-7-02. (Cité page 84.)

Tewfik Ziadi et Jean-Marc Jézéquel. Manipulation de lignes de produits logiciels : Une approche dirigée par les modèles. Dans Sébastien Gérard, Jean-Marie Favre, Pierre-Alain Muller, et Xavier Blanc, éditeurs, *1ere journées sur l'Ingénierie Dirigée par les Modèles - IDM'05*, juin 2005. (Cité page 26.)

==========

Titre **Modélisation en Interaction Homme-Machine et Système d'Information : à la croisée des chemins**

Résumé Mes recherches visent à inventer des méthodes de conception ou des composants de méthodes (c'est-à-dire des modèles de produits, de processus, et des outils) de nature à soutenir le développement de Systèmes d'Information (SI) innovants, bénéficiant des avancées technologiques pour assurer à l'utilisateur une qualité en tout contexte d'usage. Elles se situent ainsi à la croisée de trois communautés, introduisant chacune une préoccupation : l'Interaction Homme-Machine (IHM) pour la prise en compte de l'utilisateur final ; les Systèmes d'Information (SI) pour la considération du contexte organisationnel ; et le Génie Logiciel (GL) pour l'étude et l'invention des techniques et outils supports. Mes contributions s'articulent autour de trois axes : 1) la mise en commun des pratiques des domaines de l'IHM et des SI pour favoriser la prise en compte des spécificités de ces deux domaines ; 2) l'étude conjointe de l'auto-explication pour avancer, avec les spécificités de chacun, vers le défi commun des modèles interactifs ; 3) les pratiques et outils de gestion de modèles au delà des domaines de l'IHM et des SI.

Mots-clés interaction homme-machine, système d'information, méthode, modèle, processus

---

Title **At the crossroads of Human Computer Interaction and Information System Modelling**

Abstract The aim of my work is to invent design methods or method components (i. e. product and process models, and tools), which can buttress the development of innovating information systems with technology advances, in order to make sure of the quality for end-users in any context of use. Then it is set at the intersection of three communities, each of them introducing a specific concern : Human-Computer Interaction to take into account end-users, Information Systems to consider the organizational context ; and software engineering for the study and the invention of techniques and tool supports. My contributions are centred around three axes : 1) the pooling of practices from the HCI and IS domains to favour the consideration of the specificities of each domain ; 2) the joint study of the self-explanation problem to advance, with each one's specificities, towards the interactive model challenge ; 3) practices and tools for model management, that are necessary beyond the HCI and SI domains.

Keywords human-computer interaction, information system, method, model, process