Université Lille 1
Laboratoire d'Informatique Fondamentale de Lille

Habilitation à diriger des recherches

spécialité « Informatique »

Centrer l'Ingénierie Dirigée par les Modèles sur l'humain

par

Xavier Le Pallec

Habilitation soutenue le 05 Décembre 2014 devant le jury composé de :

M.	Jean-Marc Geib	Université Lille 1	(Garant)
M.	Jean-Marc Labat	Université Pierre et Marie Curie	(Rapporteur)
M.	Philippe Palanque	Université Toulouse III	(Rapporteur)
M.	Michel Riveill	Université Nice Sophia Antipolis	(Rapporteur)
M.	Patrick Heymans	Université de Namur	(Examinateur)

Je dédicace cette HDR à toutes les personnes avec qui j'ai travaillé toutes ces années et sans qui, rien de ce qui est présenté dans ce rapport n'aurait pu être fait.

Remerciements

Depuis le début de ma carrière scientifique, de nombreuses personnes m'ont servi et me servent encore aujourd'hui de modèle. Je profite de ce manuscrit pour les remercier de tout mon coeur

Alain Derycke pour son appétit de connaissance

Bénédicte Fiévet pour rester zen malgré tous les problèmes que je lui apporte quotidiennement

Bernard Carré pour son intégrité

Jean-Claude Tarby pour son sens de l'amitié, son enthousiasme, son ouverture et sa droiture

Jean-Marc Geib pour sa sincérité gentille et son sens du partage

José Rouillard pour son sens de la communication

Laurence Duchien pour son dynamisme et sa créativité

Lionel Seinturier, dont la modestie n'a d'égale que l'excellence de son parcours

Mona Laroussi pour sa capacité à gérer douze mille tâches à la fois

Nicolas Genon pour son perfectionnisme et sa méticulosité

Patrick Heymans pour sa clairvoyance et sa rigueur scientifiques

Philippe Roose pour son ultra réactivité

Raphaël Marvie pour ses conseils avisés et ciblés

Sophie Dupuy-Chessa pour son sens du collectif et son abnégation

Sophie Tison pour son équité et sa disponibilité

Thierry Nodenot pour sa perspicacité

Et Nina dont les actes quotidiens me montrent chaque jour comment intégrer ses différentes qualités.

Merci aussi à mes rapporteurs (Jean-Marc, Michel et Philippe) d'avoir lu jusqu'au bout le manuscrit et de m'avoir fait croire qu'il était de bonne qualité.

Merci à Pierre-André, Rim, Nadia, Daniel et Michel pour ces heures de discussion passionnante et ces moments de joie que nous avons partagés.

Villeneuve d'Ascq, le 1er décembre 2014.

Table des matières

Liste des figures

Introduction

Quand nous réfléchissons à un système à construire, quand nous analysons une situation ou quand nous tentons d'expliquer un phénomène, nous raisonnons généralement à partir d'une simplification du système à concevoir, de la situation ou du phénomène. Ceci est dû en partie aux limites de notre système perceptif et de nos capacités cognitives. Nous pouvons remplacer le mot *simplification* par modèle - un modèle est créé à partir d'une entité originale, entité dont il reflète suffisamment de propriétés afin de pouvoir être utilisé à la place de celle-ci dans un ou plusieurs buts donnés (Stachowiak 1973) - et dire que nous raisonnons à partir de modèles. Souvent en science, la notion de modèle est associée à celle de formalisation, mais la simplification guidée par un objectif demeure le principe fondamental du modèle et fait que modéliser est une activité récurrente chez l'être humain. Il n'est donc pas étonnant de retrouver la modélisation en Ingénierie Logicielle (IL) où le code des systèmes à concevoir ou à maintenir est suffisamment complexe pour que les informaticiens aient recours à des modèles. Dans les années 80 et 90, les modèles ont été au centre des **outils CASE** (Computer-Aided Software Engineering). L'idée de ce nouveau type d'outils était d'élever le niveau d'abstraction, comme l'avaient fait les langages de 2ème et 3ème génération. Des modèles textuels ou graphiques constituaient ce nouveau niveau d'abstraction. Dans les types de modèles utilisés, on trouvait des modèles entités-relations, des diagrammes de flux, des automates à états...

Ces outils fournissaient aussi des artefacts graphiques représentant des fonctionnalités évoluées. Leur utilisation dans un diagramme signifiait la génération automatique du code associé. S'il y a eu beaucoup d'intérêts scientifiques autour de ces outils CASE, **leur adoption dans l'industrie logicielle n'a pas réellement eu lieu**. Schmidt (Schmidt 2006) indique **quatre problèmes à cela**.

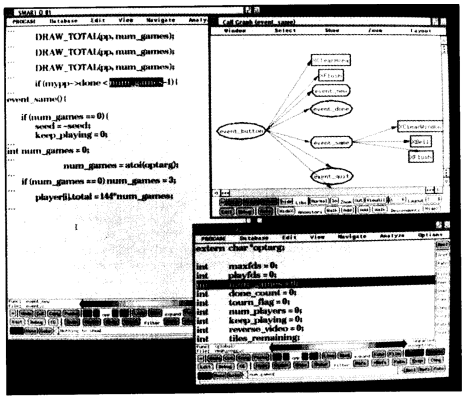


Figure 1 – *Smart System - 1990 (Oman et al. 1990)*

Développement fastidieux Simplifier la programmation, c'est aussi se reposer sur une plateforme proposant un grand nombre de fonctionnalités et de services (concurrence d'accès, distribution...). À l'époque, ce type de plateforme étant très rare, les concepteurs d'outils CASE ont du développer leur propre plateforme afin de proposer les artefacts précédemment cités. Le développement, le

débuggage et la maintenance a alourdi fortement la tâche des concepteurs d'outils CASE.

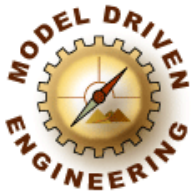
La montée en charge non prise en compte L'utilisation de ce type d'outils était plutôt limitée : un seul programmeur pour développer l'application ou plusieurs mais un seul à la fois (c'est-à-dire une utilisation en série et non pas concurrente des fichiers de conception). Ceci est difficilement compatible avec bon nombre de contextes de développement dans l'industrie logicielle.

Plateforme propriétaire Les outils CASE généraient du code pour un seul environnement d'exécution, généralement propriétaire vu le problème précédent de maturité des OS/plateformes. Ceci rendait difficile l'intégration de ce code avec d'autres plateformes de développement.

Langage Généraliste Les langages de modélisation étaient généralistes et s'accordaient mal à des préoccupations particulières.

Les concepteurs des outils CASE n'ont pas assez pris en compte le contexte d'usage (montée en charge, langage généraliste) et économique (plateforme propriétaire) de leur clients. Changer de paradigme de développement provoquait une rupture importante et il aurait été très judicieux de rester le plus proche possible des concepteurs et de leurs besoins. Une approche plus pragmatique a été adoptée par d'autres acteurs du logiciel. Elle consistait à améliorer les outils existants en répondant aux demandes des développeurs. De plus en plus de plateformes industrielles de développement sont ainsi apparues : les middlewares, les IDE [2] centrés sur le code (comme Eclipse ou Netbeans). Non seulement elles fournissaient un nombre croissant de fonctionnalités (ex : librairies graphiques, fichiers) et de services (ex : application répartie, annuaire, base de données) mais le niveau d'abstraction augmentait (objet, interface distante, composant). Cette évolution a grandement facilité le travail des développeurs et peut-être enterrer définitivement les outils CASE.

Malheureusement ces plateformes ont rapidement atteint un degré de complexité tel qu'il est difficile pour les développeurs de les maîtriser totalement. De ce fait, les informaticiens se spécialisent souvent pour un aspect donné et parfois sur une version donnée. Par ailleurs, ces plateformes ont permis de construire de grandes applications distribuées, mais dont la difficulté de développement est elle aussi très élevée. Cette difficulté est exacerbée par l'évolution des plateformes et la mise à jour manuelle nécessaire mais souvent périlleuse des applications.



L'utilisation de modèles est donc revenue d'actualité. Chercheurs et industriels impliqués ont veillé à ne pas commettre les mêmes erreurs commises avec les outils CASE. C'est ainsi que s'est dessinée petit à petit une nouvelle approche que l'on nomme depuis le début des années 2000, Ingénierie Dirigée par les Modèles (Model Driven Engineering). Comme nous le verrons dans le chapitre 1, deux idées prédominent dans cette démarche où les modèles sont les artefacts logiciels les plus importants. Tout d'abord, pour modéliser un système il est préférable d'employer plusieurs langages de modélisation où chacun est spécifique à une préoccupation particulière (Interaction, logique

2. Integrated Development Environment

métier, déploiement...). Un point fort de l'IDM sera donc d'apporter des solutions pour la mise en correspondance des différents sous-modèles. Ensuite l'indépendance technologique est aussi un principe fondamental : la conception d'un système doit être réalisée au travers de modèles les plus indépendants possibles des plateformes technologiques. Cela permet ensuite de projeter/transformer ces modèles en d'autres modèles, eux technologiques, et donc de produire ensuite du code pour la technologie de son choix. Pour cette raison de nombreux travaux liés à l'IDM concernent la transformation de modèles et dans une moindre mesure la génération de code.

Après plus de dix ans d'évangélisation, l'IDM n'est pas encore une réalité dans l'industrie. Sur les cinquante informaticiens interrogés par Marion dans (Petre 2013), un seul avait testé une telle approche. En Décembre 2010, nous avions réalisé une journée de rencontre régionale entre chercheurs et industriels autour de l'IDM. Six acteurs industriels importants dans la région avait présenté leur pratique de l'IDM. Cela peut sembler significatif et un bon signe, mais leur vision de l'IDM était plus proche des outils CASE qu'une véritable démarche IDM. De mon côté, j'ai interviewé en 2012, une vingtaine de professionnels pour voir quel était leur opinion vis-à-vis de la modélisation en général, et je suis retombé sur les mêmes résultats que ceux de l'étude de Marion Petre. Les seules pratiques proches de l'IDM se limitaient à décrire des MCD [3] pour générer la base de données ou de spécifier des modèles BPMN [4] pour générer des aiguilleurs de flux dans un contexte multi-systèmes. Encore une fois, ce sont des approches ressemblant aux anciens outils CASE et, pour les MCD, qui existaient avant l'IDM.

Est-ce un signe que l'IDM n'est pas efficace, c'est-à-dire qu'elle ne permet pas aux entreprises de construire plus vite, avec moins d'erreur et de manière plus confortable qu'auparavant ? Le temps d'adoption, généralement long, est un facteur à prendre en compte dans cette réflexion. Toutefois, les enquêtes de Marion Petre ou celle de (Grossman et al. 2005) montrent qu'il y a aussi des facteurs humains qui empêchent l'adoption massive de la modélisation logicielle (et UML en particulier) : modèles manquant de contexte, un découpage vertical d'UML [5] inadapté à la réalité du terrain, UML n'est clairement pas adapté à certains types de collaborateurs (même les cas d'utilisations)... Daniel L. Moody ou ses collaborateurs détaillent d'ailleurs plus précisément les problèmes notationnels d'UML dans (Moody et van Hillegersberg 2008), et aussi ceux de i* et BPMN dans (Moody et al. 2010) et (Genon et al. 2011). S'ils sont très peu pris en compte dans la notation, les facteurs humains ne le sont guère plus dans les outils liés à ces langages de modélisation. La lourdeur que nous avons mentionnée précédemment renvoie à des IHM (Interactions Humain-Machine) peu efficaces : lenteur au démarrage, ralentissement pour des diagrammes "un peu" volumineux, nom des opérations proposées dans les menus peu clair, pas d'optimisation (raccourcis) pour les actions les plus usuelles, ni pour la personnalisation des diagrammes (titre, couleur...), etc.

3. Modèles Conceptuels de Données

4. Business Process Model and Notation. `http://www.bpmn.org`

5. UML ne permet d'avoir les cas d'utilisation, les structures des entités, leur comportement... dans un seul et même diagramme.

Card et al. ont démontré depuis longtemps l'importance des IHM dans tout système informatique dans le livre à l'origine de celles-ci (Card et al. 1983) : à l'aide de nombreux résultats issus d'expérimentations sur les capacités sensorielles, mémorielles, cognitifs et moteurs des humains, ils montrent la limitation de ces capacités et prouvent que la façon d'interagir avec un système doit être adapté à celles-ci sinon en résulteront une utilisation limitée du système, des erreurs fréquentes de manipulation ou un rejet total. Cette adaptation ne se limite pas à une "jolie" interface. Cela implique de connaître les préoccupations des utilisateurs, leurs attentes, les activités dans lesquelles s'insère le système informatique, les tâches qui lui seront demandées à ces moments-là, le modèle mental des utilisateurs dans ces activités... Appréhender ces différents aspects peut être réaliser en *poursuivant la conception dans l'usage* comme l'indiquent Rabardel et al. dans (Rabardel et Pastré 2005). C'est d'ailleurs ce que prônent les approches agiles (scrum, lean startup), itératives (RUP) ou de conception centrée utilisateur. Si une majorité de chercheurs en Génie Logiciel prônent à leurs étudiants ce type d'approches et si l'industrie du logiciel les adopte de plus en plus, il semble que la communauté IDM ne suit pas une telle approche. En effet, la majorité des travaux en IDM concerne des mécanismes ou des opérateurs de typage, de composition, de transformation, de réutilisation de modèles ou de construction de langages de modélisation spécifiques, mais très peu sont liés à des retours d'expérience ou à des études d'usage de ces mécanismes/opérateurs. Ce type d'études ou retours nous permettraient d'affiner les objectifs de l'IDM et d'améliorer les mécanismes existants et sûrement d'en créer de nouveaux qui seraient adaptés aux contextes des concepteurs.

C'est ce qui a été mon leitmotiv et ma problématique de recherche pendant 11 ans : ***poursuivre la conception de l'IDM dans l'usage***. L'IDM est une approche que je trouve ingénieuse et originale et j'ai souhaité - et je souhaite toujours - y contribuer en me focalisant sur les usages afin, en autres, que l'IDM ne reproduise pas certaines erreurs qu'a commises "l'approche CASE".

Pour adopter une telle démarche, j'ai choisi 2 domaines d'applications : le e-learning (ou, en français les EIAH : Environnements Informatiques pour l'Apprentissage Humain) et les interactions multimodales (Bolt 1980, Nigay 2001). J'ai aussi développé un outil, ModX, pour rapidement prototyper des "chaines IDM". Pour l'étude des interactions multimodales, j'ai implémenté un bus à message dédié, WSE, afin de concevoir/générer rapidement des applications multimodales. Mon étude des usages s'est d'abord focalisé sur le mapping vertical de modèles avec la thèse de Pierre-André Caron. Focus que j'ai continué avec les thèses de Rim Drira et Nadia Elouali. Le travail avec Pierre-André Caron et l'utilisation de ModX en général m'ont aussi permis de comprendre qu'une étude des usages de la modélisation logicielle était nécessaire. Cela a été au coeur d'un travail avec Raphaël Marvie et Jean-Claude Tarby (et le début d'une longue collaboration) et des thèses de Daniel Liabeuf et Michel Dirix. Plus récemment, j'ai porté une attention particulière à un des aspects de l'activité de modélisation : les notations visuelles. Cela a donné lieu à plusieurs collaborations, dont une avec l'équipe SIGMA du LIG (Grenoble) et une avec le laboratoire PReCISE de Namur.

L'objectif de cette HDR est de décrire ces différents travaux et de montrer leurs apports à la thématique principale. Ce document se structure en 5 chapitres.

Le premier chapitre décrit d'abord les différentes composants de l'IDM. Grâce à cette description, je peux introduire et détailler les trois problématiques qui ont structuré ma recherche pendant 11 ans : le mapping vertical, l'activité de modélisation et la notation visuelle. Pour chacune d'elle, je montre les travaux scientifiques qui y ont trait et indique le positionnement de ma recherche par rapport à eux.

Le second chapitre commence par un aperçu des deux domaines métiers sur lesquels j'ai appliqué l'IDM : le e-learning et les IHM, et en particulier les interactions multimodales. Ceci permettra au lecteur de mieux comprendre les contributions qui y sont attachées. La seconde moitié du chapitre présente deux outils que j'ai développés : ModX et WSE. WSE a, au final, peu servi dans les contributions présentées dans cette HDR mais a permis de prototyper de nombreuses idées de réalisation autour des interactions multimodales et nous sert actuellement beaucoup dans l'équipe Carbon pour faire communiquer des éditeurs de modèles Web et l'environnement Eclipse. Carbon est une équipe en cours de création et ses membres sont Cédric Dumoulin, Jean-Claude Tarby et moi.

Le mapping vertical de modèles est au coeur du **troisième chapitre**. La thèse de Pierre-André Caron qui au départ devait montrer l'intérêt du MDA pour le e-learning. Le MDA, défini par l'OMG, peut être vue comme un sous-ensemble de l'IDM, où le mapping vertical est important. La thèse a finalement montré des problèmes de traçabilité de ce mécanisme. Ces problèmes ont eu comme résultat une sensation de perte de contrôle pour les enseignants et donc un rejet de la solution MDA de leur part. Se focalisant sur les "objets frontières", la thèse de Rim Drira visait à supprimer cette rupture lors du mapping de modèles. Les contributions principales ont été 1) de mettre en lumière les bonnes pratiques de projection comme un intérêt majeur du mapping de modèles 2) de proposer un mapping plus humain et moins automatisé. La thèse de Nadia Elouali termine ce chapitre et montre que, pour des besoins liés à la conception, la projection technologique peut prendre une forme différente de la traditionnelle transformation de modèles.

Le quatrième chapitre se concentre sur l'activité de modélisation et les préoccupations liées à l'ergonomie. Ce chapitre aborde mes travaux liés aux Processus de Modélisation Incrémentale (PMI) qui visaient à associer des méthodologies à des langages de modélisation afin de guider les concepteurs. La dualité diagramme/modèle est ensuite traitée au travers de la thèse de Daniel Liabeuf. Sa problématique de réutilisation de bouts de diagrammes par le copier/coller permet de souligner, comme les PMI, l'importance de la prise en compte des facteurs humains. Mais elle met aussi en lumière le manque de sémantique dans les spécifications classiques de méta-modèles et l'intérêt de définir rigoureusement les interactions possibles autour de celui-ci comme un moyen efficace de pallier à ce manque. La thèse de Michel Dirix clôture ce chapitre en montrant les éléments importants à communiquer entre collaborateurs pour faciliter leur collaboration lors de l'édition de modèles.

La notation visuelle est le sujet du **cinquième et dernier chapitre**. Si

elle était une préoccupation récurrente dans le développement et l'utilisation de ModX, elle est devenue ces dernières années une part importante de mon activité de recherche. Jusqu'ici, ma principale contribution a été de fournir un support logiciel pour aider à la création de métriques de qualité des notations visuelles. Ce chapitre donnera un rapide aperçu de mes travaux actuels et des collaborations associées.

Ce document se termine par une conclusion qui me permet d'évoquer la suite logique de ces travaux : la création de l'équipe Carbon, où l'IDM est la thématique centrale et où les facteurs humains et les IHM sont les pierres angulaires de notre démarche.

Afin de donner des points de repère au lecteur, voici une chronologie des thèses que j'ai encadrées (ou à celle que j'ai participé).

FIGURE 2 – *Thèses encadrées (et participation)*

1 Ingénierie Dirigée par les Modèles et problématiques d'usage

Sommaire

Ce chapitre présente l'Ingénierie Dirigée par les Modèles (IDM) et décrit ses différents composants, qui seront évoqués tout au long de ce document. Cette description permet de fixer le vocabulaire qui sera utilisé dans ce document mais aussi de pointer ensuite sur les 3 aspects de l'IDM que j'ai étudiés d*u point de vue de l'humain* : le mapping vertical (Kent 2002), l'activité de modélisation et la notation visuelle. Après avoir mentionné les travaux scientifiques existants autour de ces problématiques, je détaillerai précisément les points qui ont été peu ou pas abordés non traités et sur lesquels j'ai travaillé.

1.1 Naissance de l'IDM : MDA, MDE, DSML, PIM, PSM...

L'initiative Model Driven Architecture (MDA) adoptée en 2001 par l'Object Management Group (OMG) peut être considérée comme l'impulsion à l'origine de la naissance de l'IDM. Dans (Miller et Mukerji 2003), il est indiqué que l'OMG définit le MDA comme une approche logicielle où les spécifications des fonctionnalités d'un système sont séparées des spécifications de leur implémentation sur une plateforme technologique. Le point commun de ces deux types de spécifications est qu'elles sont matérialisées par des modèles. Ceci donne donc lieu à deux types de modèles : les modèles indépendants des plateformes (PIM : Platform Independent Models) et les modèles spécifiques à une plateforme (PSM : Platform Specific Models).

Un PIM est une spécification formelle des aspects structurels et fonctionnels d'un système qui s'abstrait des détails techniques. L'adjectif *formel* renvoie à une syntaxe bien formée et à une sémantique définie plus ou moins formellement. Un PSM est une spécification d'un système pour une plateforme donnée. Ce type de modèle est en général issu d'un PIM, au travers d'une ou plusieurs transformations.

Les principaux avantages de cette approche avancée par l'OMG sont :

L'Interopérabilité Il est plus facile de spécifier des liens entre systèmes (dans des contextes d'intégration ou d'interopérabilité) à l'aide de termes indépendants des plateformes utilisées, les aspects technologiques pouvant être traités dans un second temps.

La Validation Il est plus facile de vérifier l'exactitude et la correction à l'aide d'un modèle (en particulier un PIM) plutôt qu'avec du code

Le Multi-plateforme Les PIM permettent de produire des implémentations sur différentes plateformes, avec peu ou pas de différences fonctionnelles entre elles car basées sur le même modèle de structure et de comportement.

Si (Miller et Mukerji 2003) mentionnent qu'il peut y avoir plusieurs PIM pour un même système, associés à différents points de vue ou perspectives, plus ou moins abstraits, le guide du MDA ne s'aventure pas à décrire cet espace de modélisation, c'est-à-dire les types possibles de PIM, leurs relations, et les opérations possibles pour passer de l'un à l'autre. Au final, le MDA vise essentiellement à exploiter les modèles comme des outils pour l'abstraction, pour leur qualité synthétique et le fait d'offrir différentes perspectives.

Parce qu'il juge la dichotomie PIM/PSM simpliste, Kent (Kent 2002) souhaite explorer le précédent espace de modélisation. Il estime : que cet espace est multidimensionnel (il n'y a pas que la dimension abstrait-concret) ; qu'une perspective (liée à une préoccupation) est un ensemble de points d'intersection entre les axes des précédentes dimensions ; et qu'un modèle est lié à une perspective. En se référant à (Tarr et al. 1999), il part du principe que le nombre de dimensions n'est pas figé et qu'il dépend du projet. Toutefois, il liste quatre dimensions qui, selon lui, sont récurrentes à tout projet informatique : le domaine traité, l'aspect (contrôle des flux,

informationnelle), l'abstraction (en ne se limitant pas aux deux seules positions définies par le MDA : abstrait et concret) et les parties prenantes. Enfin, il note qu'une nouvelle perspective ne signifie pas un nouveau langage de modélisation. Une seule des dimensions impliquées dans la perspective fournira en général le langage, les autres dimensions n'ayant un impact que sur le contenu des modèles spécifiés. Il nommera cette approche, plus générale que le MDA, le Model Driven Engineering (MDE) ou IDM en français. Schmidt (Schmidt 2006) indique d'ailleurs que ces modèles spécifiques à une perspective sont un des points forts de l'IDM par rapport aux outils CASE dont les langages étaient trop généraux. Pour construire ce type de modèles, il évoque les Domain Specific Modeling Langages [1].

Au final, c'est dans (Jezequel 2008) que J.M. Jézéquel résume parfaitement ce qu'est l'IDM : représenter les différents aspects d'un (futur) système logiciel à travers différents modèles ... *This requires that models are no longer informal, and that the weaving process is itself described as a program manipulating these models to produce a detailed design that can ultimately be transformed to code or at least test suites.* ***This is really what Model Driven Design is all about.***

1.2 Les différents ingrédients de l'IDM

Trois ingrédients se dégagent de la description précédente de l'IDM :

- les langages de modélisation
- le tissage de modèles
- la génération de code

1.2.1 Les langages de modélisation

Modéliser consiste à créer des modèles. Inévitablement, nous devons indiquer ce que nous entendons par "modèle" dans ce manuscrit. (Muller et al. 2012) regroupent une dizaine de définitions de ce qu'est un **modèle** : *Un modèle est une description ou une spécification partielle, une simplification, une abstraction d'un système, un ensemble de déclarations... d'un système, d'un espace de problèmes ou de quelque chose... Un modèle est créé par quelqu'un, dans un but ou une intention précise et pour une personne*. Ils citent aussi Ludewig (Ludewig 2004) pour rappeler que chercher une définition universelle de ce qu'est un modèle est peine perdue, car la communauté (IDM) ne partagent pas une compréhension commune de la notion de modèles. Dans ce manuscrit, nous optons pour la définition "simple" de (Kleppe et al. 2003) - *Un modèle est une description partielle ou non d'un système dans un langage bien*

1. Nous ferons ici la différence entre les DSML et les DSL (Domain-Specific Languages). Ces derniers sont généralement textuels et, dans les faits, plutôt orientés pour la programmation et non la modélisation. Même cette distinction n'est pas de l'avis de (Clark et al. 2004)

formé [2]. Celle-ci a comme intérêt de mettre en avant la notion de langage bien formé, c'est-à-dire composé d'une syntaxe et d'une sémantique.

La **syntaxe** est souvent divisée en deux parties (Atkinson et Kuhne 2003) :

- la syntaxe abstraite : les concepts et associations à partir desquels les modèles sont construits. (Wile 1997) parle d'essence structurelle du langage ;
- la syntaxe concrète : le rendu concret de ces concepts, c'est-à-dire le formalisme visuel au travers duquel on accède à un modèle (Kelly et Tolvanen 2007).

La **sémantique** d'un langage décrit la signification des concepts de celui-ci, c'est-à-dire la corrélation ou la correspondance des concepts avec les pensées et les expériences des concepts du monde qui nous entoure (Clark et al. 2004). (Harel et Rumpe 2004) va plus loin en précisant qu'il faut dans un premier temps décrire la sémantique du domaine traité de manière large, en intégrant par exemple les notions ou règles qui sous-tendent les décisions que l'on peut prendre sur ce qu'exprimera le langage de modélisation. Dans un deuxième temps, il faut décrire dans le détail la correspondance entre cette sémantique et tout élément appartenant à la syntaxe du langage.

Pour illustrer les syntaxes abstraite / concrète et la sémantique, nous pouvons prendre l'exemple d'un langage de modélisation simple dédié à la représentation d'activités et des flux entre elles. Voici d'abord un modèle issu de ce langage, décrivant la rédaction d'un article scientifique par un doctorant. Le modèle est ici représenté par un diagramme.

2. Les aspects intentionnels et humains soulignés par Steenmüller (Steinmüller 1993) (pour qui, par qui et pour quoi) mis en avant par (Muller et al. 2012) seront vus plus en détail dans les sections suivantes.

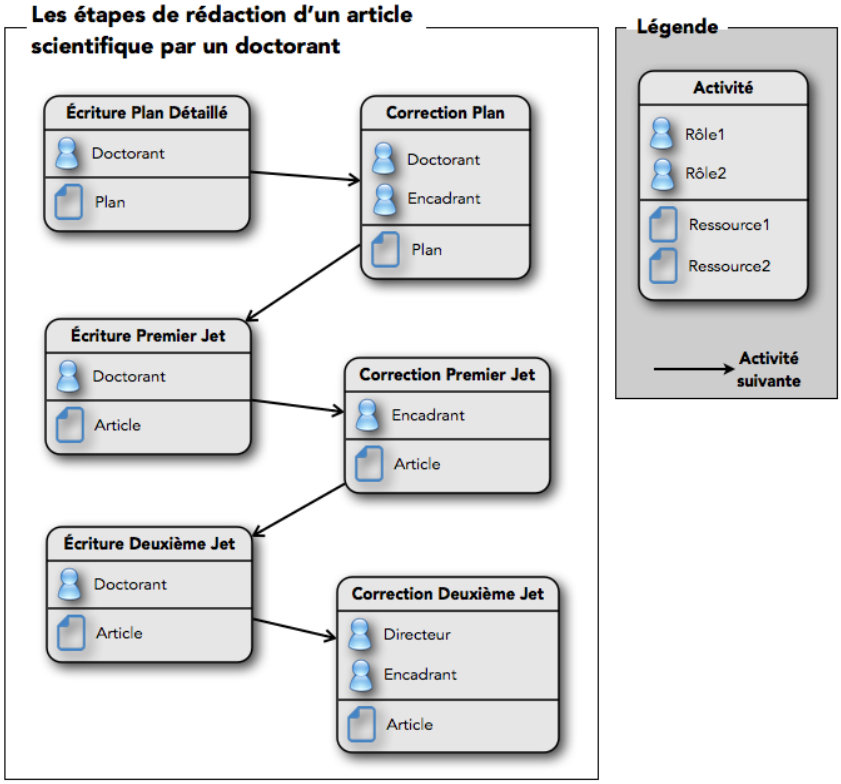


FIGURE 1.1 – *Diagramme/Modèle : rédaction d'un article scientifique par un doctorant*

La sémantique des concepts et des associations sous-jacents à ce modèle peut être décrite de la manière suivante.

Sémantique (informelle) des Flux d'Activités

Le langage de modélisation a pour but de définir les activités humaines que supporteront nos futures applications. Une **Activité** est caractérisée par un **nom** qui l'identifie (il doit unique) et dont le sens apporte un ou des indices sémantiques sur l'activité. Les activités sont associées entre elles au travers d'un **Flux d'Activité** : une activité peut être **précédée** par une activité et **suivie** par une autre. Un ensemble de **Rôles** est défini pour ce flux d'activités, et un rôle peut intervenir dans une ou plusieurs activités. Un rôle est caractérisé par un **nom** (lui aussi, unique et apportant du sens). Dans la réalisation d'un flux d'activité, un agent (humain ou informatique) pourra jouer un ou plusieurs rôles. Le support de travail de chaque activité est constitué par une ou plusieurs **ressources**. Une ressource est caractérisée par un **nom** (unique + sémantique) et une **url**. Les droits d'accès (lecture et écriture) auront à être définis au travers de la ressource réelle pointée par l'url (par exemple, un document google permettant l'écriture ou non).

FIGURE 1.2 – *Exemple de Sémantique d'un langage de modélisation*

À l'aide du méta-méta-modèle MOF[3], on peut définir la syntaxe abstraite de la manière suivante :

3. Meta-Object Facility : norme définie par l'OMG

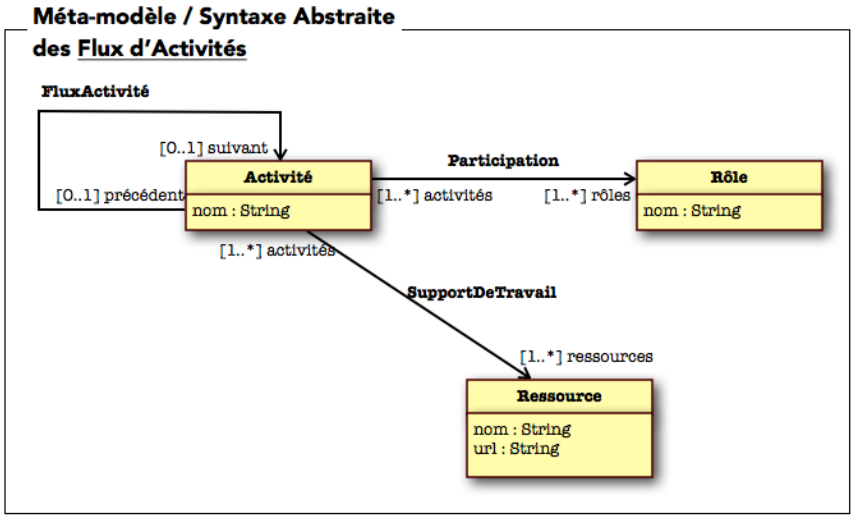


FIGURE 1.3 – *Exemple de syntaxe abstraite*

En accord avec la définition sémantique, on y retrouve les concepts d'Activité, de Rôle, de Ressource et des associations entre eux.

Enfin, la représentation graphique utilisée dans le diagramme précédent est la suivante :

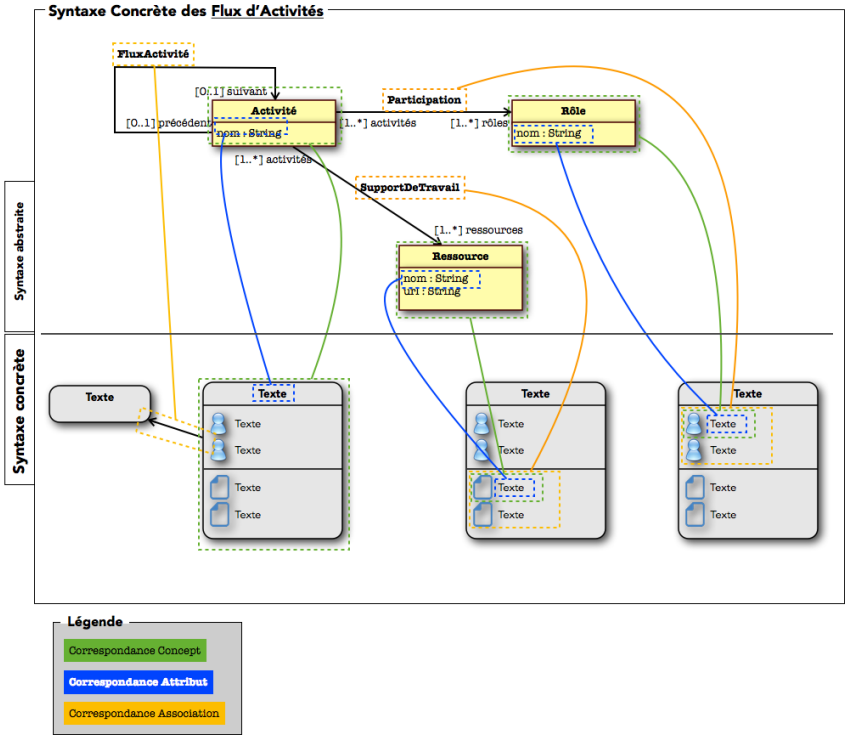


FIGURE 1.4 – *Exemple de syntaxe concrète*

Ce n'est pas une définition à proprement parler. Il s'agit plutôt ici de montrer les principes de correspondances. Généralement, les définitions se font à travers des formulaires (Obeo Designer (Obeo 2013), MetaEdit+ (MetaCase 2013) ou ModX(Le Pallec et al. 2006a)). Mais des outils comme

Eugenia (Foundation 2013) ou xOWL (Wouters 2012) proposent de le faire via des langages textuelles dédiés. La récente norme Diagram Definition de l'OMG (Elaasar et Labiche 2011) propose de le faire en deux temps : d'abord la définition des différents éléments graphiques utilisables, puis un ensemble de règles QVT (OMG 2014) permettant la construction d'un diagramme à partir d'un ensemble d'éléments de modèles.

1.2.2 Le tissage de modèles

Le tissage des différents modèles - d'un système - entre eux est la pierre angulaire de l'IDM. (Kent 2002) propose de faire la distinction entre mapping/tissage de modèles de même langage (model translation) et modèle de langages différents (language translation). (Clark et al. 2004) fait aussi une distinction entre le mapping horizontal et vertical.

Mapping Vertical

Le mapping vertical est très similaire à la dimension abstraction de Kent, la différence étant que Clark & al. détaille un peu plus les raisons d'un tel mapping : passer à un modèle plus "concret" peut être guidé par un besoin d'optimisation, de changement de paradigme (pour coller à celui de la future plateforme d'implémentation), de changement d'architecture (bus à événement ou objet, architecture orientée service ou ressource, etc.) . . .. Si on peut bien sûr utiliser un langage comme UML pour spécifier des modèles de plus en plus concret, de nombreux travaux scientifiques s'intéressent à ce type de mapping quand il s'agit de projeter ou traduire un modèle dans un autre langage.

Sur notre précédent exemple de flux d'activités, nous pouvons imaginer que nous souhaitons concrétiser les flux sur la plateforme Android. Le modèle "Rédaction d'un article scientifique" pourrait se traduire ainsi

FIGURE 1.5 – *Modèle Android de la rédaction d'un article scientifique*

Ce nouveau modèle serait conforme à ce méta-modèle [4] :

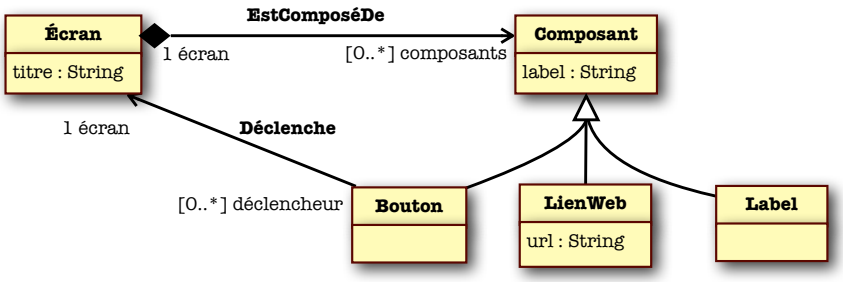


FIGURE 1.6 – *Méta-modèle Android sous-jacent*

Les règles de transformations *Activités vers Android* pourraient, avec le langage ATL [5], commencer comme cela :

```
rule RoleVersEcran {
  from
    role : FluxActivites!Role
  to
    bouton : Android!Bouton (
        label <- role.nom,
        ... // lien vers le premier écran
    ),
    ecran : distinct Android!Ecran foreach(activite in role.activites)(
        titre <- activite.nom,
        ... // création d'un bouton pointant l'écran suivant
    )
}
```

Mapping Horizontal - Même langage

Le mapping horizontal est plus intéressant que le précédent mapping car il permet la modélisation multi-perspectives, intérêt majeur de l'IDM. Malheureusement, ce mapping est aussi plus complexe. Une illustration sur notre précédent exemple peut nous aider à détailler cette complexité. Nous allons adopter pour cela la modélisation orientée aspect à l'aide de template (Baniassad et Clarke 2004, Stein et al. 2002, Al Abed et Kienzle 2011, Vanwormhoudt et al. 2013).

4. À noter que les préoccupations liées à l'agencement ne sont ici pas prises en compte
5. ATL : ATL Transformation Language `http://www.eclipse.org/atl/`

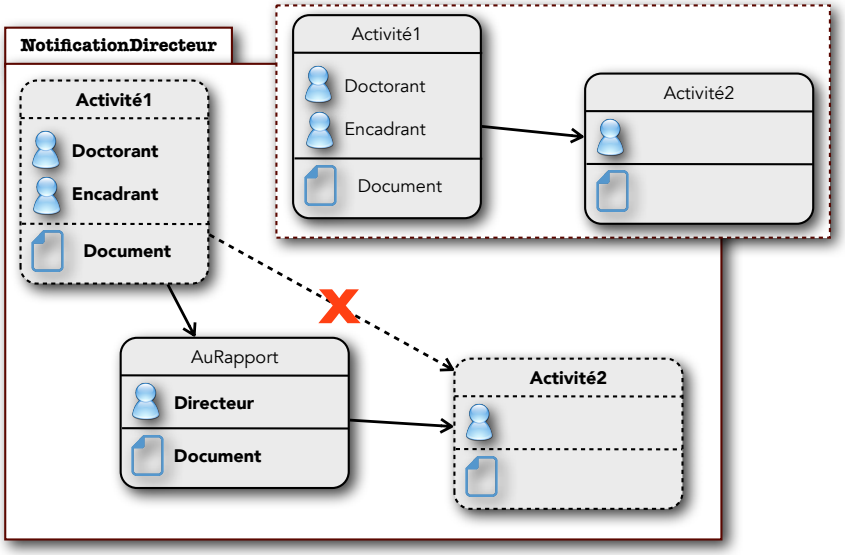


FIGURE 1.7 – *Mapping horizontal à l'aide de templates*

Dans cet exemple, on définit le point de coupe (*pointcut*) suivant : tout couple d'activités 1) qui se suivent 2) où la première dispose d'un document et 3) implique le doctorant et l'encadrant. Le greffon (*advice*) a associer à ce point de coupe est une activité intermédiaire nommée "Au Rapport" qui implique le directeur et le document. Il y a donc l'ajout d'1 élément (l'activité "Au rapport") et de 4 liens (2 flux d'activités et les 2 liens pour associer directeur et document à la nouvelle activité). La syntaxe qui est ici utilisée est celle de (Vanwormhoudt et al. 2013) où lorsque les paramètres apparaissent dans le coeur du template, ils sont en pointillés et/ou la police de caractère est en gras. Il y a ici une liberté - pas rapport aux travaux cités - qui est prise avec la suppression d'un lien (croix rouge).

Une autre approche (Jezequel 2008) consiste à simplement afficher le point de coupe et son greffon dans deux compartiments d'un même bloc qui constitue l'aspect.

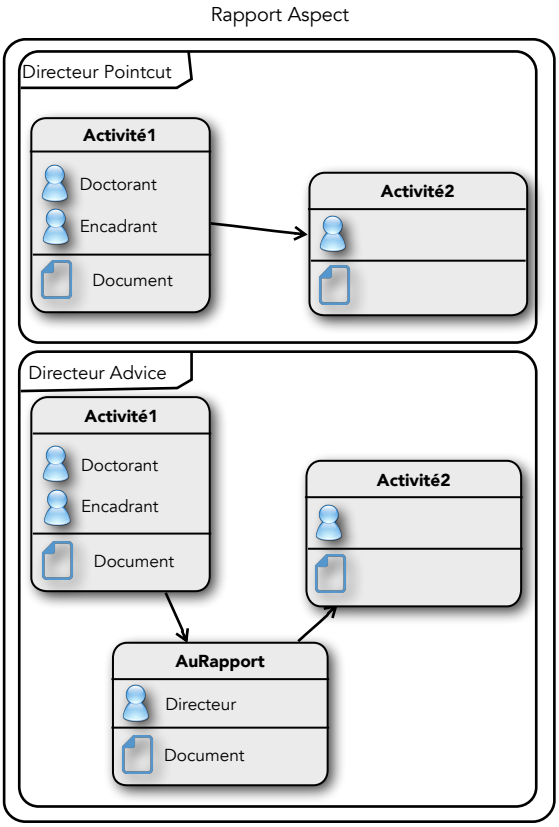


FIGURE 1.8 – *Mapping horizontal selon (Jezequel 2008)*

Sur ces deux exemples, on peut voir le principal problème de ce genre de modélisation : s'il y avait un deuxième aspect avec le même point de coupe, *dans quel ordre appliquer les deux aspects*, sachant qu'en appliquant l'aspect "Rapport" le point de coupe n'existera plus (le lien entre l'activité 1 et 2 étant supprimé) ? En plus de cet aspect chronologique, il faut aussi prendre en compte les possibles conflits : si un deuxième aspect ajoute un lien sur l'activité 1, il y aura un problème car une activité ne peut avoir qu'une seule activité suivante. Et dans ce cas quel aspect choisir ou comment régler ce conflit ? Ces points sont abordés dans (Jezequel 2008) ou (Vanwormhoudt et al. 2013) mais rentrer dans le détail de ces moteurs de fusion serait hors de propos (car trop éloigné de mes travaux).

Mapping Horizontal - Langages différents

La situation se complique un peu plus si les modèles à fusionner sont issus de langages différents. (Clark et al. 2004) parle de deux possibilités. La première (*langage views*) est un système où on utilise un seul langage mais avec plusieurs syntaxes concrètes, chacune représentant une perspective à traiter. C'est le cas d'UML. Le mapping entre les différentes vues restent un point sensible mais, dans le cas d'UML, il est techniquement simplifié car des "associations ou des types de jonctions" sont déjà présents. (Schauerhuber et al. 2006) indique d'ailleurs qu'une grande majorité des travaux sur la modélisation orientée aspect utilise UML comme support. Ce n'est pas le cas de la deuxième possibilité (*systems views*) où

chaque perspective à son propre langage. Dans ce type de situations, parfois nommé "composition de domaines" (Estublier et al. 2005a;b), la principale originalité est d'établir des liens de correspondance entre les concepts des différents méta-modèles. (Clark et al. 2004) proposent une approche dynamique avec le langage Xsync afin de synchroniser des modèles entre eux, qu'ils soient ou non d'un même méta-modèle.

En ce qui concerne les liens entre modèles, l'utilisation de point de jonction est toujours possible, mais la correspondance par *name matching* semble peut-être plus aisée.

1.2.3 La génération de code

La génération s'apparente en général à des fichiers à trous lesquels vont être remplis par des éléments de modèles [6]. Toutefois, les langages de génération de code tel que MTL (OMG 2008) ont souvent un paradigme très différent des langages de programmes classiques. Ils proposent généralement des directives adaptées à un raisonnement ensemblistes (pour tout élément de type T, faire ...). Le langage OCL [7] d'ailleurs est intégré à MTL afin de renforcer les capacités de ciblage d'éléments.

Voici un exemple de génération de code Android sur notre exemple précédent :

```
[comment encoding = UTF8 /]
[module generate('http://www.lifl.fr/~lepallex/android')]
[template public generateActivite(uneActivite : Activity)]
[comment @main/]
[file (uneActivite.nom, false, 'MacRoman')]
public class [name/] extends Activity {
[for (aView : View | views)]
    [let aButton : Button = aView]
        public Button [name/];
    [/let]
    [let aLabel : Label = aView]
        public TextView [name/];
    [/let]
  ...
[/for]
}
[/file]
[/template]
```

6. C'est une des raisons pour lesquelles, par exemple, le module acceleo d'Eclipse utilise le module de génération des JSP.

7. Object Constraint Language `http://www.omg.org/spec/OCL/`

1.3 Problématique 1 : Mapping vertical du point de vue de l'humain

1.3.1 Un expert peut-il facilement vérifier la cohérence entre un modèle source et un modèle généré ?

Dans une perspective IDM, les modèles d'implémentation sont construits automatiquement à partir de modèles plus abstraits. Les Documents de Spécifications Fonctionnels (DSF) constituent une pratique - très répandue - où des modèles abstraits (dans les DSF Générales) sont traduits ensuite en modèles d'implémentation (DSF Détaillées). C'est d'ailleurs dans ce type de situation que l'IDM a un intérêt car la traduction y est majoritairement manuelle. Dans le cas des DSF, les modèles abstraits sont généralement écrits en concertation avec le client (ou un représentant de celui-ci) : réunion, conversations téléphoniques, échanges de mails... Dans le cas où ces modèles sont transformés automatiquement en modèles d'implémentation, des informaticiens vont récupérer ces modèles générés pour les compléter car ces derniers sont bien sûr incomplets et nécessitent l'expertise de spécialistes de la plateforme d'implémentation visée.

De manière générale, les règles de transformation de modèles sont "normalement" écrites par un groupe de personnes dont l'expertise couvre le langage de modélisation source et celui de destination. Ces règles ne sont pas parfaites. Et même si elles l'étaient, elles le sont pour une période donnée : rien ne dit qu'une nouvelle situation dans un nouveau projet ne soit pas inadaptée à ces règles. Il est donc important qu'une personne vérifie la correction de la transformation. Pour cela, la traçabilité est primordiale.

C'est sur ce point que les travaux de Pierre-André Caron ont porté (Caron 2007) : **est-ce que cette opération de transformation automatique de modèles est viable du point de vue de l'humain ?** En d'autres termes :

1. est-ce que l'opérateur humain trace facilement le passage entre la source et le résultat ?

2. est-ce qu'il fait le lien facilement entre le contexte de départ (modèle fonctionnel) et le contexte actuel (modèle d'implémentation) afin d'effectuer les ajouts nécessaires ?

3. sera-t-il facile de vérifier si les modifications apportées au modèle d'implémentation restent en accord avec le modèle fonctionnel ?

Les travaux sur la transformation de modèles n'abordent que très peu cette problématique de la perception et de l'utilisabilité[8] à du modèle généré. La conférence ICMT[9] centrée sur les transformations de modèles compte à ce jour 6 éditions dont chacune consiste entre 15 et 20 publications de travaux relatifs à cette thématique. Parmi eux, nous n'avons

8. L'utilisabilité, dans le sens défini par la norme ISO 9241-11 : *le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié.*

9. International Conference on Model Transformation
`http://www.model-transformation.org/`

trouvé que les travaux Amstel et al. (2012), Vallecillo et Gogolla (2012) qui s'intéressaient à ce problème. (Amstel et al. 2012) ajoute le problème - classique en GL - de la traçabilité : *'Traceability plays an essential role in a number of typical model development scenarios such as debugging and change impact analysis. When debugging models, it is important to understand whether the erroneous part of the target model results from the source model or from the transformation itself, and to pinpoint the corresponding parts of the source model and/or transformation functions. Moreover, when source models are about to change, one should determine the effect of proposed changes on the target model."*. Mais ces travaux consistaient plus à avoir un monitoring des transformations effectuées (voir la figure suivante) que d'accompagner le concepteur dans son processus de compréhension des éléments générés.

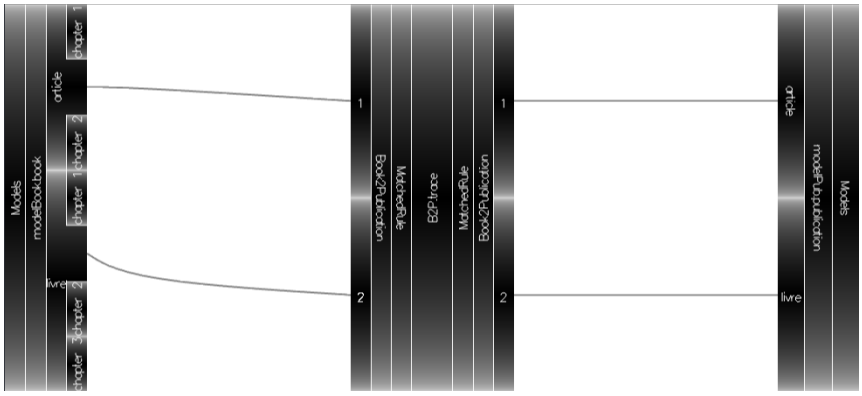


FIGURE 1.9 – *Traçabilité lors d'une transformation de modèles avec (Amstel et al. 2012)*

Un article plus ancien (rémy Falleri et al. 2006) portait aussi la traçabilité mais n'abordait pas l'apport de celle-ci vis-à-vis de la perception de l'agent humain.

Cette préoccupation se retrouve dans les travaux de (Laforcade 2010) qui traite de l'utilisation de l'IDM pour la construction d'environnements pédagogiques. Les enseignants spécifient leurs environnements au travers d'un langage de modélisation et des informaticiens vont prendre le relais pour compléter *technologiquement* cette spécification. Dans un souci de coordination entre enseignants et informaticiens, les auteurs ont porté une attention particulière à la transition entre les deux points de vue en proposant des objets frontières de modélisation. Ils en rappellent la définition qui nous semble tout à fait appropriée ici : *des objets abstraits ou concrets dont la structure est suffisamment commune à plusieurs mondes sociaux pour qu'elle assure un minimum d'identité au niveau de l'intersection tout en étant suffisamment souple pour s'adapter aux besoins et contraintes spécifiques de ces mondes (Star 1989)*.

> Dans le chapitre 3, nous démontrerons au travers d'une expérimentation que lors d'une transformation de modèles, la traçabilité est réellement problématique et nous proposerons un mécanisme différent de la transformation classique pour répondre à ce problème qui reprend l'idée des objets frontières.

1.3.2 L'intégration d'éléments technologiques

On se réfère à la dimension "abstrait/concret" lorsqu'on souhaite indiquer dans quelle proportion on prend en compte les détails d'implémentation : plus un modèle comporte des éléments ou des caractéristiques proches d'une plateforme d'implémentation, plus il est concret. Cette qualification d'abstrait (ou concret) est relative (Kent 2002) : un modèle UML est abstrait par rapport à un modèle Java mais concret par rapport à un modèle i* (Yu et John 1994). Elle ne touche pas qu'à une différence de langage de modélisation utilisée : une description à un grain plus fin peut rendre un modèle plus concret qu'un autre.

L'une des directives fortes du MDA est de préférer les modèles au code car la valse des plateformes d'implémentation rend le code vite obsolète :

> *"When these systems in their turn need modifying and integrating with next year's hot new technology (and they will) the result is the kind of maintenance nightmare all computer users fear"* ;
> *"[MDA] allows the same model specifying system functionality to be realized on multiple platforms"* ;
> *"supporting system evolution as platform technologies come and go"*.

Pourtant, dans ce même MDA, il est question de modèles spécifiques à des plateformes (PSM). Ces modèles vont donc être eux aussi rapidement obsolètes, de manière plus ou moins forte. De plus cette vision, comme le dit S. Kent, est grossière car dans la dimension abstrait/concret, il n'y a pas de séparation nette mais plutôt une progression. L'argument de non-sensibilité aux évolutions technologiques des modèles est donc à modérer : il faut surtout analyser dans quelle mesure les éléments ou les préoccupations technologies, auxquelles un langage de modélisation fait référence, sont sensibles aux évolutions technologiques. Si un langage de modélisation est lié fortement à une plateforme, le risque d'obsolescence sera important. S'il fait référence à un ensemble de plateformes, et est donc suffisamment abstrait pour éviter une obsolescence complète, il sera toutefois sensible aux évolutions logicielles ou matérielles générales, aux bonnes pratiques d'implémentation. . ..

De ces observations, on peut logiquement déduire que l'intégration d'éléments/préoccupations technologiques est une nécessité fréquente. Nous avons dû faire face à cette problématique lorsque Nadia Elouali [10] a développé un éditeur de modèles où les interactions multimodales étaient au coeur de la perspective de modélisation associée et où les supports visés étaient les smartphones. Les capteurs embarqués dans ces derniers (accéléromètre, gyroscope, GPS) évoluent assez fréquemment : de nouveaux capteurs apparaissent (ex : pression atmosphérique, lecteur d'empreintes digitales) et de nouveaux usages émergent (ex : l'utilisation du vibreur, le magnétomètre, le gyroscope et de l'accéléromètre pour faire tourner un smartphone - application *cycloramic*). Il n'est pas possible d'intégrer conceptuellement les capteurs et les usages associés car cela engendrerait un problème de versioning trop important. L'alternative est bien sûr l'utilisation de composants. Mais sous quelle forme doivent être spécifiés ces composants : modèle de composant de type Fractal (Bruneton et al. 2012)

10. Titre de la thèse : "Approche à base de modèles pour la construction d ?applications mobiles multimodales". Soutenance prévue en Octobre 2014.

ou CCM (OMG 2006), modèles paramètres de type UML templates (Caron et al. 2004) . . . ? Quelle forme est la plus adaptée pour l'auteur de modèles (utilisateur des composants), pour le lecteur et pour la génération de code ou la transformation de modèles qui est associée à l'éditeur ?

> Dans la dernière partie du chapitre 3, nous verrons comment le contexte technologique et d'usage nous ont guidés dans la conception d'un DSML. De plus, notre démarche s'inscrit dans la vision de Kent d'une perspective de modélisation : le domaine traité (les interactions multimodales) est la base de notre langage, et la dimension abstraction n'influe que le contenu des modèles, influence renforcée ici par un mécanisme de templates.

1.4 Problématique 2 : Ergonomie liée à l'activité de modélisation

Le terme ergonomie référence ici l'ergonomie cognitive (Hoc 1991), c'est-à-dire les processus mentaux - perception, mémoire ou raisonnement - et les réponses moteurs et non pas l'ergonomie organisationnelle (structure socio-technique) ni celle physique (souffrance, fatigue).

Lorsqu'on développe des outils de conception en adoptant une approche IDM, c'est-à-dire avec différentes perspectives de modélisation et des outils de mapping et de génération de code, on se rend compte assez vite que l'activité même de modélisation n'est pas simple et qu'il est difficile d'implémenter des éditeurs de modèles efficaces (rapides, limitant les erreurs et confortables). Nous avons déjà mentionné dans l'introduction les problèmes d'utilisabilité des langages de modélisation logicielle. Si on regarde les études majeures sur l'utilisation d'UML en industrie (Forward et al. 2010, Petre 2013, Chaudron et al. 2012, Dzidek et al. 2008), on se rend compte que deux des trois problèmes majeures d'UML concernent l'ergonomie : l'inadaptation conceptuelle de la notation et les outils jugés trop lourds. Malgré cela, l'ergonomie liée à la modélisation (et même à la conception logicielle en général) est un sujet qui est peu abordé en comparaison des autres préoccupations. Les *dimensions cognitives* sont un des rares et très pertinents frameworks sur ce sujet. Nous les détaillons ici pour mettre en lumière la complexité de cette préoccupation.

1.4.1 Les dimensions cognitives

Proposées originalement par Thomas R.G. Green en 1989 (Green 1989), puis exposées et détaillées par M. Petre en 1996 (Green et Petre 1996), les *Cognitive Dimensions* (CD) sont un framework dont le but est d'analyser l'utilisabilité des artefacts informationnels. Si les systèmes logiciels sont le principal type d'artefacts visés, les CDs peuvent s'appliquer aussi à des produits "classiques" (*made out of plastic and paper*). Ce framework est LA référence quant il s'agit de *design ergonomics* lié aux langages de programmation visuelle. Ceci n'est pas le cas en IHM, domaine aussi visée par les

CDs, car la profusion d'approches concurrentes dans ce domaine en a fait un framework parmi d'autres.

Sept types d'activités

Un des constats principaux des CDs est que l'efficacité d'un système (notation + environnement) est fonction de l'usage qu'on en fait. Afin de mieux évaluer chacune des dimensions (cognitives) d'un système, les auteurs définissent 7 types d'activité que l'on peut mener lors de la conception :

Recherche Rechercher une information. Ex : Quel est le point de départ d'un diagramme d'activités.

Incrémentation Ajouter des informations. Ex : Ajouter une activité ou un paramètre d'entrée (pin) à une déjà existante.

Modifier la structure Ex : Ajouter une partition à un diagramme d'activités et y placer des activités déjà présentes.

Transcription Passer d'un support à un autre. Ex : retranscrire un diagramme "papier" dans un format électronique.

Conception exploratoire Incrémenter l'idée, l'intuition qu'on a en tête. Ex : Se focaliser sur un flux d'activités pour un scénario "bien clair".

Compréhension exploratoire Essayer de comprendre l'objectif du système (*what something's for*) plutôt que les détails de son fonctionnement (*how it does it*).

Comparaison Comparer un élément avec un autre. Ex : Est-ce que cette activité est la même que celle-ci (cad deux représentations graphiques du même élément de modèle) ou est-ce deux éléments différents ?

Les treize dimensions

Ces activités vont avoir un impact sur l'importance de chacune des dimensions cognitives. Avant de parler de leur impact, voici la liste des dimensions cognitives [11].

Abstraction Cette dimension concerne les mécanismes d'abstraction disponibles dans un système. De ce point de vue, les auteurs définissent 3 types de systèmes : abstraction-hating, abstraction-tolerant, abstraction-hungry. Si le premier type de système ne permet pas de traiter un groupe d'éléments comme une seule entité, son apprentissage est plus rapide. Le dernier type de système nécessite continuellement d'effectuer des abstractions et est donc plus difficile au premier abord, mais il gère beaucoup mieux de hauts niveaux de complexité.

11. Plutôt que de parler de langage de modélisation, les auteurs des CD parle de *système* qui renvoie au langage de modélisation (syntaxe et sémantique) et au support logiciel associé (souvent un éditeur). Le terme *notation* est souvent utilisée. S'il correspond généralement à la syntaxe concrète du langage, il est ici à considérer comme le langage de modélisation où la syntaxe concrète est primordial.

Proximité du mapping Il s'agit de la distance entre les concepts pour décrire un problème réel et les concepts de la notation. Si une notation est dédiée à un type de problèmes particuliers, la distance conceptuelle devrait être idéalement nulle. Un des objectifs des DSML est d'avoir une distance la plus faible possible (Frank 2011).

Cohérence Quand une personne connaît quelques structures de conception proposées par le système, jusqu'où peut-il deviner (avec succès) les structures restantes ? Plus une notation a une cohérence élevée, plus son temps d'apprentissage est court.

Dispersion/concision Combien de lexemes faut-il utiliser pour décrire un problème ? Normalement la concision est de mise car elle permet de gagner du temps. Mais elle peut être parfois problématique car elle rend plus difficile la perception de la différence entre deux descriptions. Le besoin de concision se retrouvera plus tard dans le contexte MDE/MDA aussi bien pour des modèles UML (Lange et Chaudron 2005) que pour des modèles spécifiques (Frank 2011).

Risque d'erreurs Dans quelle mesure un système peut aider le concepteur à éviter les faux-pas, les erreurs d'inattention ou tout autre type d'erreurs qui sont dues à une mauvaise utilisation de la notation. La définition de types de valeur, la complétion de code, le soulignage en rouge d'instructions erronées sont des mécanismes classiques d'*error-proneness* pour les langages de programmation. La vérification de modèle est un mécanisme lié à cette dimension (Campos et Harrison 2001).

Opérations mentales difficiles Manipuler des "objets" est plus difficile lorsque des conditions ou des négations s'appliquent sur eux. C'est encore plus dur si on combine des objets conditionnés/inversés. La notation doit être élaborée afin de diminuer la charge cognitive liée au raisonnement. En d'autres termes, la notation doit permettre de retranscrire visuellement et directement toutes déductions ou syllogismes possibles sur les objets afin d'éviter au lecteur de faire lui-même les inférences et donc de mobiliser significativement ses capacités cognitives (Recanati 2007).

Dépendances cachées Les dépendances non visibles (comme les références entre cellules d'une feuille de calcul) sont sources d'erreurs. Il faut veiller à les minimiser. Généralement, pour les langages de modélisation graphiques, ce type de problème vient des propriétés qui ne sont pas affichées par la syntaxe concrète mais seulement via une fenêtre de propriétés de l'éditeur associé. Une fonctionnalité de recherche rapide est une solution possible pour retrouver ces dépendances.

Engagement prématuré En programmation, il est fréquent de devoir écrire du code en y laissant des "blancs" qui seront remplis plus tard (ex : traiter du gros grain avant le grain fin) ou travailler sur papier avant de spécifier en détail avec l'éditeur de code. Ce type d'engagement prématuré est bien sûr source d'erreurs ultérieures

car on peut oublier de remplir ces trous, on peut passer à côté de détails importants quand on travaille à gros grain... ces allers-retours constituent une activité sensible. Pour les notations visuelles, on va trouver des problèmes de réagencement des éléments graphiques (afin de mieux retranscrire visuellement la logique globale finale ou afin d'éviter des liens qui se croisent) ou des mauvais choix sur la structure syntaxique à utiliser.

Évaluation progressive Il s'agit de la capacité d'un système à permettre à un concepteur d'évaluer son travail à tout moment, même si le modèle est incomplet. Cette propriété va plus loin que le besoin de simulation souvent cité pour l'IDM (Rivera et al. 2009), et elle est très rarement proposée par les éditeurs de modèles qui ont cette fonction de simulation (car tout doit être connecté, spécifié...).

Expressivité du rôle Cette dimension est liée au rôle, à l'objectif ou à l'utilité de certaines constructions. En programmation, on utilise des noms de variables très significatifs, on met des commentaires, on indente le code, on regroupe les fonctions qui sont proches... Si l'utilisation de la notation secondaire (voir plus bas) est possible avec les notations visuelles, l'utilisation de balises ou de commentaires pour exprimer le rôle d'un ensemble d'éléments n'est pas toujours proposé. Pourtant, si un groupe d'éléments nécessite des opérations mentales difficiles, exprimer le rôle du groupe est salutaire.

Notation secondaire et s'émanciper du formalisme L'indentation, le choix des noms, les commentaires, le choix des instructions, la structuration d'un programme... sont libres - c'est-à-dire non structurés par les règles de construction syntaxique - et apportent eux aussi beaucoup d'informations, de sémantique. Dans les CD, c'est ce qui est appelé la notation secondaire. Pour les langages de modélisation graphique, l'agencement est un des moyens les plus importants en ce qui concerne la notation secondaire. Les auteurs des CD évoquent aussi l'utilisation de tags ou de commentaires.

Viscosité La viscosité concerne essentiellement les éditeurs associés aux langages de modélisation : quel est le temps nécessaire pour effectuer une modification ? Par exemple, si un élément de type A ne peut être lié qu'à un seul élément de type B et que le premier est déjà lié à un élément : si on veut le lier à un autre élément de type B, faudra-t-il supprimer le premier lien et en créer un deuxième ou la création du deuxième va-t-elle supprimer automatiquement le premier lien ? On peut aussi penser aux nombres de clics nécessaires pour modifier une propriété, créer un élément et l'associer à un autre... Le déplacement d'un élément est aussi problématique car ses liaisons avec d'autres éléments vont peut-être nécessiter des ajustements (si les liens sont segmentés).

Visibilité et Juxtaposition La visibilité renvoie à l'accessibilité : si l'obligation de placer le code d'une classe dans un fichier dédié en Java permet de gérer la complexité de "grands" programmes... elle rend

l'accessibilité générale plus difficile. Là encore, l'éditeur est important car il doit proposer des fonctions simples de navigation. Ce problème d'accessibilité intervient aussi avec les notations visuelles où, par exemple, certaines constructions sont encapsulées dans un seul élément (ex : implémentation d'une méthode de classe UML spécifiée via une machine à état, activité structurée dans les diagrammes d'activités UML...). La juxtaposition est un mécanisme pertinent pour ce problème d'accessibilité : il s'agit de la possibilité d'afficher deux bouts de code ou modèle l'un à côté de l'autre (où l'un des bouts peut être le contenu d'un élément affiché dans l'autre).

Impact des dimensions sur les activités liées à la conception

Voici les impacts de ces dimensions sur les précédentes activités :

	Transcription	**Incrémentation**	**Modification**	**Exploration**
Viscosité	Acceptable	Acceptable	Pénalisant	Pénalisant
Dépendances cachées	Acceptable	Acceptable	Pénalisant	Acceptable pour des petites tâches
Engagement prématuré	Pénalisant	Pénalisant	Pénalisant	Pénalisant
Difficulté à abstraire	Pénalisant	Pénalisant	Pénalisant	Pénalisant
Obligation d'abstraire	Utile	Utile (?)	Utile	Pénalisant
Notation secondaire	Utile	-	Très utile	Très pénalisant
Visibilité / juxtaposition	Non vital	Non vital	Important	Important

Aperçu rapide des impacts dimensions/activités Church et Green (2008)

C'est l'incrémentation qui a été au centre de mes travaux sur l'activité de modélisation.

1.4.2 L'incrémentation

Pour l'incrémentation, la résistance au changement ou les dépendances cachées ne sont pas trop problématiques, une accessibilité faible non plus. Par contre, l'engagement prématuré est plus problématique, tout comme le fait de pouvoir difficilement abstraire, encapsuler des éléments dans des entités de plus haut niveau.

Engagement prématuré et Guidage

À notre connaissance, l'engagement prématuré est un aspect peu traité. Si (France et Rumpe 2007) soulignent bien l'importance de fournir des critères ou des recommendations (*Good modeling methods should provide modelers with criteria and guidelines for developing quality models*), ils ne proposent pas aux concepteurs un moyen d'avancer mais plutôt un moyen de vérifier

si les modèles produits sont de bonnes qualités. Quand (Kelly et Pohjonen 2009) évoquent, pour l'utilisation d'un langage, les problèmes d'ignorance du processus d'utilisation (*Ignoring the use process*) ou du non-besoin d'apprentissage (*No training*), ils parlent des problèmes de réutilisation et de la difficulté à maîtriser les concepts aussi bien que son ou ses créateurs. Mais, en aucun cas, ils n'évoquent ni les problèmes de prédiction et de mémorisation liés à chaque incrémentation, ni les problèmes de guidage : "ce n'est pas parce qu'on a les ingrédients que l'on connait la recette !". De plus, le guidage renvoie aux bonnes pratiques, ce que même le concepteur du langage n'a pas au départ. Et les bonnes pratiques renvoient à l'évolution finale d'un modèle (est-ce que le modèle vérifie telle règle ?) et non à la démarche pour y arriver.

> Cela a été l'objet de recherche autour des processus de modélisation incrémentale ou comment guider pas-à-pas un concepteur dans la création d'un modèle. Ces travaux sont décrits dans la section 1 du chapitre 4.

Réutilisation

La réutilisation est une pratique courante en Ingénierie Logicielle (Ghezzi et al. 1991, Visser 1995) depuis la création de cette branche de l'informatique lors de la séparation code - machine dans les années 50. Aujourd'hui on trouve de nombreuses formes de réutilisation, comme l'héritage, l'importation, les composants, les design patterns... Comme l'indique (Kelly et Pohjonen 2009), un langage de modélisation n'est pas fait pour créer un seul modèle mais plutôt une multitude. Il est important de prévoir des mécanismes de réutilisation pour éviter aux concepteurs de re-saisir les mêmes fragments de modèles ou de faire de nombreux copier-coller. Pour cela, on peut employer des techniques de réutilisation de modèles comme les aspects et le template-binding vus précédemment mais aussi l'héritage, la composition (Lau et Wang 2007) ou l'importation (Cuellar et al. 2003, Laguna et Marqués 2009).

Toutefois ces techniques sont des techniques avancées de réutilisation. Le copier/coller est une technique plus accessible et de ce fait utilisé très fréquemment en programmation dans un "premier temps" avant une refactorisation où prendra place les précédentes techniques. D'ailleurs, les travaux de (Rosson et Carroll 1993) et (Burkhardt et al. 2002) montrent l'importance du copier- coller dans le processus de compréhension de fragments de code, préalable à l'utilisation de techniques plus évoluées. Les étapes d'abstraction, de sélection, de spécialisation et d'intégration (Krueger 1992) du copier-coller permettent effectivement de saisir en détail l'intérêt, l'utilité et la cohérence d'un fragment de modèle. Donc le copier-coller peut être utilisé comme une étape intermédiaire à l'utilisation de techniques avancées de réutilisation. Nous pouvons donc dire que le copier/coller est une fonction qui améliore l'ergonomie de ces techniques avancées.

Si on omet cet aspect, le copier-coller est aussi une technique simple qui permet de créer beaucoup d'éléments rapidement. (Forward et al.

2010) mentionne que le clavier et la souris ne sont pas les meilleurs périphériques d'interaction pour éditer des diagrammes. L'exemple suivant l'illustre parfaitement.

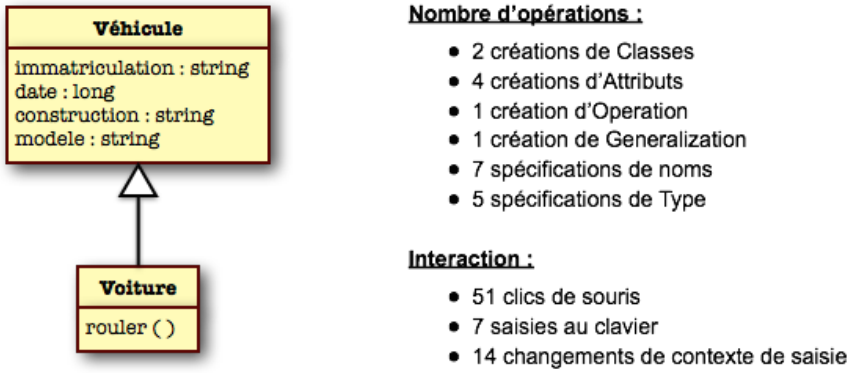


FIGURE 1.10 – *Clavier-Souris pour la création de deux classes UML*

Le nombre de clics et de changements de contexte pour la création de deux classes UML met en lumière un des problèmes liés à la lourdeur des éditeurs UML actuels. Le copier-coller va permettre d'éviter un certain nombre de ces interactions.

Faciliter l'utilisation de techniques avancées de réutilisation, éviter de nombreuses interactions, deux avantages qui montrent l'importance du copier-coller dans l'édition de modèles d'un point de vue ergonomique. Toutefois les règles de fonctionnement du copier-coller ne sont pas si simples et cela a pour conséquence un résultat qui semble peu satisfaisant pour les utilisateurs des éditeurs de modèles [12]. Des travaux ont déjà été faits sur le copier/coller de manière générale, et le "coller spécial" dans les éditeurs de texte ou dans les tableurs est une émanation de ces travaux. Mais le problème du copier-coller avec les éditeurs de diagrammes ne vient pas du collage mais de la sélection de départ. Si on sélectionne la classe Voiture sur le diagramme suivant, et qu'on copie-colle dans un autre projet/modèle UML, quel est va être le résultat ?

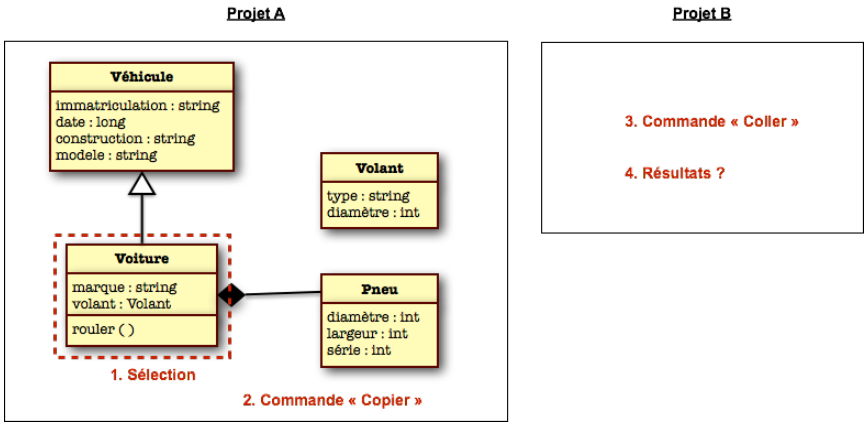


FIGURE 1.11 – *Problème de la sélection externe et implicite lors du copier-coller*

12. Nous reparlerons de cette affirmation dans le chapitre 4

Doit-on garder la classe Véhicule ? L'association de composition avec la classe Pneu ? La classe Volant ?

La thèse de Daniel Liabeuf (2011-2014) devait au départ traiter de la détection de bonnes pratiques à partir des copier/coller réalisés par une communauté de concepteurs. Mais rapidement, nous nous sommes rendus compte que les précédentes questions n'avaient jamais fait l'objet de travaux scientifiques ni d'un consensus parmi les outils existants.

> La section 2 du chapitre 4 porte sur une étude approfondie du copier/coller du point de vue de l'usager : quelles sont les attentes des concepteurs concernant le résultat d'un copier/coller d'éléments de diagramme ? Nous verrons aussi que ceci renvoie à des questions liées à la perception d'éléments de diagrammes, en particulier ce qui constitue leur identité.

1.4.3 La modélisation collaborative

Dans le monde industriel, la modélisation est une activité qui est rarement solitaire. Pour les petites équipes (< 6), il est possible qu'il n'y ait qu'une personne (le "fonctionnel") qui édite les modèles. Mais 1) ses collaborateurs viendront lire ses modèles (la lecture fait aussi partie de l'activité de modélisation), 2) le développement implique souvent des évolutions au niveau des modèles qu'effectueront les développeurs. Pour les plus grandes équipes, les modèles seront construits par plusieurs fonctionnels au travers de réunions, de sessions collaboratives de modélisation (synchrone) ou au tour par tour (asynchrone). La modélisation collaborative est une nécessité qui est malheureusement peu traitée dans la littérature (Rittgen 2012b).

Le développement de supports logiciels à des activités collaboratives est loin d'être un problème trivial et a fait l'objet de milliers de publications depuis une trentaine d'années. Concernant la modélisation, les premiers problèmes à résoudre - pour avoir un éditeur de modèles collaboratif qui fonctionne - sont d'ordre technique, et concerne la communication (Thum et al. 2009, Farwick et al. 2010a). Dans les différentes opérations proposées à l'utilisateur par l'éditeur, il faut déterminer celles qu'il faudra transmettre aux autres utilisateurs à chaque fois qu'elles seront réalisées. Ensuite, il faut déterminer le protocole de communication (quelles sont les informations à transmettre : principalement l'opération effectuée et ses arguments). Enfin, les temps de transmission n'étant pas nuls, il faut aussi gérer les conflits possibles en cas d'ordres contradictoires.

La complexité s'accroit lorsque l'on s'intéresse aux aspects interactifs et ergonomiques liés à la collaboration : quelles opérations doit proposer l'éditeur afin que les utilisateurs puissent collaborer efficacement ? Quelles actions effectuées par une personne doivent s'afficher sur les éditeurs de ses collaborateurs ? Prenons le cas simple où un concepteur crée différents éléments dans un diagramme de classes UML le Lundi, et qu'il revient sur celui-ci le Mercredi, sachant qu'un autre concepteur y a effectué des modifications le Mardi. L'éditeur doit-il afficher les modifications qui ont eu lieu depuis sa dernière connection (le Lundi) ? Si oui, comment doit-il affi-

cher la suppression d'un attribut ou d'une classe, un changement de type d'attribut, un déplacement de classe ? Si l'éditeur doit afficher la création d'une classe, doit-il en indiquer l'heure, l'auteur, son rôle, la raison de la suppression ... Sur le principe, toute information est intéressante, mais 1) l'espace de conception graphique est limité et 2) il faut aussi veiller à éviter la surcharge cognitive. Donc il va falloir faire un tri pour déterminer les informations à afficher (point 2) et trouver un moyen visuel efficace pour leur affichage (point 1). Le tri va être guidé par la tâche à effectuer par le concepteur mais aussi par le contexte et le processus général de conception/modélisation dans lequel la précédente tâche se trouve.

Seule l'équipe Chico (Gallardo et al. 2011, Gallego et al. 2011) de l'université de Zaragoya a proposé des mécanismes visuels pour aider à la collaboration dans le cadre de la modélisation. Toutefois, les auteurs de ces travaux n'ont pas au préalable étudié en détail le contexte d'utilisation et les processus possibles liés à la modélisation comme a commencé à le faire Rittgen (Rittgen 2010; 2012b;a).

Une étude sur cet aspect de la modélisation a démarré en Mars 2013 avec la thèse de Michel Dirix (sous la co-direction de Jean Marc Geib). C'est une thèse financée par la société Axellience, qui est une start-up INRIA, dirigée par Alexis Muller. Axellience a créé et mis en ligne depuis Décembre 2013, un outil Web de modélisation UML collaboratif nommé GenMyModel. Les aspects techniques précédemment évoqués ne sont pas la principale cible de la thèse de Michel. Il s'agit plutôt des mécanismes visuels qui permettent aux utilisateurs d'avoir "conscience" des autres et de leur travail, et donc de collaborer efficacement. Cette notion d'awareness (Dourish et Bellotti 1992) est donc au coeur des travaux de cette thèse.

> Dans la section 3 du chapitre 4, nous détaillons la première étude que Michel Dirix a menée en observant une centaine de projets collaboratifs en cours sur la plateforme GenMyModel. Cette étude a visé à déterminer le degré d'importance des éléments d'awareness pour les utilisateurs et l'influence des facteurs extérieurs sur ce degré.

1.5 Problématique 3 : La notation visuelle

L'utilisation de diagrammes est récurrente en conception logicielle. Dans les études sur UML citées précédemment (Forward et al. 2010, Petre 2013, Chaudron et al. 2012, Dzidek et al. 2008), il en ressort que la majorité des praticiens du logiciel vont à un moment ou un autre, dessiner sur une feuille, sur un tableau ou utiliser un éditeur pour spécifier la structure de "leur" application, le processus d'utilisation, les flux entre composants... Mais, ces mêmes études montrent qu'il ne suffit pas d'utiliser un formalisme graphique quelconque pour profiter des aspects visuels du dessin : encore faut-il qu'il soit efficace.

Limite des dimensions cognitives

Les dimensions cognitives traitent des concepts des langages graphiques et de l'environnement associés mais les principes fondamentaux liés aux aspects visuels ne constituent pas directement un sujet d'étude à eux seuls (mis à part les travaux de Engelhardt (von Engelhardt 2002, Blackwell et Engelhardt 2002)). Il est souvent question d'utiliser "convenablement" l'espace de conception graphique. C'est souvent le cas de travaux présentés dans les conférences Diagrams [13] ou VL/HCC [14]. Kosslyn, dans (Kosslyn 1985), cite des cadres plus ou moins rigoureux / formalisés sur la description des mécanismes liés à la conception graphique : la sémiologie graphique de J. Bertin (Bertin 1983), les méthodes graphiques de Chambers et al. (J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey 1983), le mapping informationnel de Fisher (Fisher 1983)... Mais ces cadres ne sont pas directement applicables au génie logiciel car ils demeurent très généraux et les graphes (ou "réseaux" selon Bertin) y sont rarement traités.

La physique des notations

Il a fallu attendre les travaux de Daniel L. Moody (Moody 2009) pour disposer d'un framework focalisé sur l'efficacité perceptuelle et cognitive des notations visuelles, telles que celles que l'on trouve en génie logiciel. Après avoir indiqué que le système de perception visuelle représente 25% du cerveau, qu'il est massivement parallèle et après avoir démontré qu'une non-exploitation de ce système de calcul évolué est préjudiciable, Moody définit ce qu'il appelle la "physique des notations" : 9 principes, basés sur la littérature, qui permettent de concevoir une notation "efficace".

Clarté sémiotique Il doit y avoir une correspondance 1-1 entre les constructions sémantiques d'une notation et les symboles graphiques, soit un concept par symbole et un symbole par concept.

Discriminabilité perceptuelle Il s'agit du niveau de discrimination visuelle entre deux éléments concrets différents. Plus celle-ci est élevée, plus la perception des diagrammes sera rapide et plus le traitement cognitif qui suivra en sera donc facilité. La distance visuelle est la principale fonction de ce critère. Elle renvoie à plusieurs variables visuelles (définie par J. Bertin) : la position (x,y), la taille, la valeur (clair vers foncé), le grain (ex : hachure, texture), la couleur, l'orientation et la forme. La distance visuelle entre deux éléments est en rapport avec le nombre de variables visuelles sur lesquelles les éléments diffèrent.

Transparence sémantique A la lecture d'un diagramme, l'intelligence perceptive du lecteur va attacher du sens à chaque élément visuel. Pour les expert(e)s d'une syntaxe concrète, le sens sera celui de la construction sémantique associée. Pour les autres, il est important

13. Diagrams : the Theory and Application of Diagrams
http://diagrams-conference.org/

14. VL/HCC : Visual Languages and Human-Centric Computing
https://sites.google.com/site/vlhcc2014/

que le sens que chacun d'eux associe à la représentation ne soit pas trop éloigné de l'élément abstrait associé. La transparence sémantique renvoie à cette distance sémantique.

Gestion de la complexité La représentation d'un système complexe est une thématique de recherche transversale à de nombreuses disciplines. C'est aussi une problématique lorsqu'il s'agit de représenter des modèles complexes c'est-à-dire constitués de nombreux éléments et/ou nombreuses connexions. Gérer cette complexité est donc une condition *sine qua none* et implique des mécanismes dédiés dans les syntaxes concrètes. L'encapsulation graphique, la fragmentation en différents diagrammes sont des exemples de mécanismes possibles.

Intégration cognitive Lorsque le lecteur navigue dans les différents diagrammes via les mécanismes de gestion de complexité associés, il lui faut des éléments visuels lui permettant de se rappeler dans quel contexte se situe le diagramme qu'il/elle a sous les yeux. En d'autres termes, il faut pouvoir intégrer la partie du modèle étudié dans la représentation mentale du modèle global. Par exemple, pour une page web, il est classique d'afficher le chemin de la page actuelle au sein de la hiérarchique du site.

Expressivité visuelle Utiliser les variables visuelles pour organiser les éléments d'un diagramme. Par exemple, faire varier la couleur permet de créer des groupes, faire varier la taille permet d'ordonner les éléments...

Double codage Enrichir les diagrammes avec des descriptions textuelles (commentaires, tags).

Économie graphique Il convient de ne pas avoir un vocabulaire visuel trop important, c'est-à-dire d'utiliser un trop grand nombre de formes/liens différents. Pour une notation trop riche, l'activité mémorielle dédiée à l'association représentation-sens devient chez les lecteurs non-experts trop importante et gênera la lecture des diagrammes produits. Ce critère implique par exemple de partitionner les modèles en diagrammes de types différents. La clarté sémiotique est ici en défaut, car l'économie graphique peut impliquer, dans le cas de langages sémantiquement riches, un déficit de symboles dû au partitionnement.

Adaptation cognitive Les capacités de dessin sont en rapport avec le support utilisé pour le dessin des diagrammes. Par exemple, l'utilisation d'images complexes est peu viable lorsque les diagrammes se feront au stylo sur une feuille de papier. Parallèlement, le niveau d'expérience du lecteur concernant l'écriture de diagrammes de type ingénierie logicielle est aussi à prendre en compte. En effet, les différents mécanismes cités plus haut (comme la fragmentation en plusieurs diagrammes) demandent moins d'effort dans leur utilisation chez une personne expérimentée que chez un débutant.

Un framework à affiner

Mis à part la clarté sémiotique, les principes de la physique des notations ne sont pas assez quantifiés et ne permettent pas à un concepteur d'avoir des *guidelines* précises à suivre étape par étape. Par exemple, le principe de discriminabilité perceptuelle indique d'avoir au moins deux variables visuelles qui diffèrent entre chaque symbole : ces variations ont-elles le même poids si elles s'appliquent sur le bord, le texte ou le fond du symbole ? À partir de quel moment peut-on considérer que deux valeurs (pour une même variable visuelle) sont perceptuellement différentes ? Y a-t-il des variables plus perceptibles que d'autres ? Pour l'expressivité visuelle, les questions sont encore plus nombreuses.

> Le chapitre 5 décrit les récents travaux que j'ai effectués et qui visent à contribuer à affiner la physique des notations : l'identification des variables informationnelles liées à celle-ci et leur intégration dans ModX afin de faciliter la création de métriques (de qualité de la notation visuelle)

.

Conclusion du chapitre

L'utilisabilité des mécanismes fondamentaux de l'IDM - le mapping de modèles et les perspectives de modélisation - est la problématique générale qui a guidé mes travaux de recherche pendant 11 ans. Dans ce chapitre, j'ai montré les différents aspects et verrous qui en dérivaient et sur lesquels j'ai travaillé : la traçabilité de la transformation de modèles, les méthodologies pour guider l'activité de modélisation, la notation visuelle... Avant de montrer les résultats de ces travaux dans les chapitres 3, 4 et 5, je vais présenter rapidement dans le chapitre suivant les domaines métiers qui m'ont servi de base de travail et les outils que j'ai du développer pour accompagner ce travail.

2 Domaine d'études et outils développés

Sommaire

Dans ce chapitre, nous présentons les domaines métiers et les outils implémentés qui ont servi de support à nos travaux.

Au démarrage de mes travaux sur l'IDM, j'ai voulu savoir dans quelle mesure le chainage MDA était viable. Au départ d'un telle chaîne, on trouve les modèles indépendants des plateformes (PIM) qui, en plus d'être "abstraits", ont pour vocation d'être spécifiques à un domaine métier. Le laboratoire et l'équipe où je me trouvais à l'époque sont intervenus dans le choix du domaine métier que j'ai du faire : ce fut le e-learning. Ceci a aussi eu un impact sur les plateformes technologiques en fin de chaîne. En effet, il ne s'agissait pas d'étudier les plateformes technologiques classiques mises en avant par l'OMG (J2E, .Net...) mais de s'orienter vers les plateformes d'implémentation classiques en e-learning : généralement des dérivés de CMS (Ganesha, Moodle, Claroline ou même des Wikis). Après un changement de laboratoire et aussi d'équipe, j'ai changé de domaine métier en m'intéressant aux interactions multimodales (IMM), sous domaine des IHM. L'avantage de ce domaine était de pouvoir travailler sur des plateformes technologiques récentes (Android, iOS) qui sont bien moins éloignées des plateformes usuelles en IDM que les plateformes pédagogiques.

Le e-learning a été la matière première de l'étude sur la traçabilité des transformations de modèles, et les IMM celle pour le support aux évolutions technologiques. Les deux domaines ont été aussi la matière pour

étudier les processus incrémentaux de modélisation. Dans la première partie de ce chapitre, nous détaillons en quoi consistent la modélisation en e-learning et celle des IMM.

Les travaux que je présente dans ce document ont débuté en 2003. L'outillage IDM était à l'époque balbutiant : les outils pour créer des éditeurs de modèles spécifiques n'étaient pas très performants et les moteurs de transformations étaient si peu évoluées que les langages de programmation classiques (Java, XSL) leur étaient souvent préférés. J'ai implémenté un outil de méta-modélisation, ModX, afin de pouvoir créer rapidement des DSML et les éditeurs associés. L'intégration d'un langage de script m'a ensuite permis de facilement écrire des procédures de transformation de modèles. La seconde partie de ce chapitre commencera par la description de l'outil ModX. Elle se poursuivra avec la plateforme Miny [1] liée aux interactions multimodales. Les modèles d'IMM que j'ai abordés concernaient dans un premier temps les maisons intelligentes et dans un second temps, les applications mobiles. Il a fallu créer une plateforme dédiée afin d'expérimenter une chaine IDM destinée à paramétrer un environnement type maisons intelligentes. Cet environnement s'appelle Miny.

1. Multimodality Is Nice for You

2.1 Les modèles métiers

2.1.1 Scénarios et Dispositifs Pédagogiques

Mes activités de recherche en tant que Maitre de Conférence ont démarré en Septembre 2003 dans l'équipe NOCE du laboratoire Trigone (Université Lille 1). Ce dernier avait pour particularité de réunir des chercheurs en Sciences de l'Éducation et en Informatique. L'équipe NOCE (Nouveaux Outils pour la Coopération et l'Éducation) était composée essentiellement d'informaticiens mais leurs travaux avaient, à de rares exceptions près, pour thème les Environnements Informatiques pour l'Apprentissage Humain (EIAH), plus généralement appelé e-Learning. Alain Derycke, le responsable de l'équipe NOCE et co-directeur du laboratoire m'avait donné pour mission d'appliquer mes travaux sur l'IDM aux EIAH. Mes travaux se sont rapidement tournés vers l'Ingénierie des EIAH qui a pour objet, selon Tchounikine (Tchounikine 2006), *"d'étudier les principes de construction des EIAH et de produire un ensemble de méthodes, de techniques et d'outils visant à encadrer et systématiser leur processus de conception"*. Concernant les outils, je me suis focalisé sur les plateformes pédagogiques de formation accessibles par le Web.

J'ai eu la chance de débuter ce travail avec Pierre-André Caron qui, après un stage de recherche (Master 2) début 2004, a poursuivi en thèse (2004-2007) [2] avec moi sous la direction d'Alain Derycke.

Plateformes de formation

Qu'est-ce qu'une plateforme de formation ? Selon l'Office de la Langue Française, c'est *un système informatique destiné à automatiser les diverses fonctions relatives à l'organisation des cours, à la gestion de leur contenu, au suivi des progrès des participants et à la supervision des personnes responsables des différentes formations*. De telles plateformes peuvent être vues comme *des système qui permettent de gérer et de donner accès à un ensemble d'activités et de ressources pédagogiques* (George et Derycke 2005). Enfin Landon dans (Landon 2002) indique les principales fonctions que peut fournir une plateforme de formation :

1. Communiquer : forum, mail, blog...
2. Augmenter la productivité des apprenants : signet, recherche, calendrier...
3. Mener une évaluation dans le groupe ou individuel : groupe de travail, portfolio, autotest...
4. Administrer les personnes : authentification, inscription au cours...
5. Dispenser des cours : Suivi des élèves, Gestion des parcours, gestion des cours, test
6. Respecter de la charte de la plateforme : accessibilité, design des cours, respect des normes.

2. Titre de la thèse : "Ingénierie dirigée par les modèles pour la construction de dispositifs pédagogiques sur des plateformes de formation"

Les plateformes pédagogiques sont généralement appelées des LMS (Learning Managment Systems). En Septembre 2005, le site Thot recensait 240 plateformes de formation (Thot 2014). Un grand nombre de ces plateformes ont aujourd'hui disparu. Des plateformes comme Moodle ou Claroline sont "sortis du lot" et ont eu raison de nombreuses plateformes *maisons*.

Les plateformes sur lesquelles l'équipe NOCE travaillait n'étaient pas forcément des LMS, c'est-à-dire conçues pour supporter des activités pédagogiques. Comme l'explique Pierre-André Caron dans sa thèse, c'est l'intentionnalité de l'utilisateur qui nous importe : si il ou elle utilise une plateforme Web à des fins pédagogiques c'est un EIAH [3]. Les plateformes étudiées pouvaient être des CMS (Content Management Systems) ou de simples Blogs ou Wikis.

Quelque soit le système, il y a toujours 3 rôles principaux (Écoutin Eric & Guidon Jacques 2000) et (Rasseneur 2004) : l'apprenant, le formateur et l'administrateur.

Il est important de noter que deux approches s'opposent. Une approche industrielle où il est possible de concevoir une plateforme pour une communauté enseignante (Paquette 2002) et une approche artisanale centrée sur les usages d'enseignants encadrant artisanalement des formations légères et où il faut adapter l'EIAH (déjà existant) aux intentions pédagogiques des enseignants. C'est la deuxième approche qui m'a intéressé dans mes travaux. Pour cette raison, nous resterons sur la définition de plateforme de formation que donne Pierre-André dans sa thèse : *Une plateforme de formation est un environnement informatique collectif permettant à une communauté d'enseignants de produire les conditions qui favorisent l'apprentissage et le suivi des apprenants.*

Scénarios pédagogiques

Pendant longtemps, les plateformes de formation ont essentiellement servi à l'auto-formation. N'étant pas encore vu comme un espace communautaire, le Web avait comme principale fonction la mise à disposition de ressources. Dans ce contexte, les concepteurs de plateformes de formations adoptaient généralement une approche documentaire, c'est-à-dire proposer la description, le référencement, la mise à disposition et le séquencement de documents. LOM (Learning Object Metadata) ou SCORM (Shareable Content Object Reference Model) sont les principales normes pour ce type d'approche. D'un point de vue pédagogique, ces plateformes peuvent s'inscrire dans une démarche :

Béhavioristes L'idée principale de ce modèle est que l'on apprend par les conséquences de ses actes (Crinon et Legros 2002). Il y a donc apprentissage si, lorsqu'une tâche est effectuée par l'apprenant, celle-ci est évaluée très rapidement et que l'apprenant est informé de la réussite ou de l'échec de cette tâche. Un programme de remédiation individuelle peut alors être mis en oeuvre. La remédiation individuelle et le délai de correction sont des facteurs importants du renforcement.

3. À la différence de P. Tchounikine qui exclut du champ des EIAH les plateformes ou outils généraux détournés à des fins pédagogiques.

Cognitivistes L'apprentissage est perçue non plus comme une activité de transmission mais comme un processus de construction des connaissances (Legendre 1988). Les pré-requis, les pré et post-test, et l'ordonnancement des documents à soumettre aux élèves sont primordiaux.

La révolution communautaire du Web 2.0 a permis aux concepteurs d'adopter une démarche socio-constructiviste, c'est-à-dire prendre comme principe fort que l'on apprend en se confrontant aux pairs et que l'apprentissage tire sa source de l'interaction sociale. Cela implique pour un enseignant de proposer des activités permettant la confrontation (mise en scène du conflit socio-cognitif). Si l'accessibilité des documents, leur référencement et des moyens de stimulations étaient importants pour les précédentes démarches, ici il faut porter attention aux outils :

d'administration permettant de créer les entités et les liens composant le dispositif pédagogique (cette partie sera détaillée plus loin dans le chapitre).

de rétroaction de groupe synchrone et asynchrone,

de production de groupe

de résolution de conflit

de workflow permettant de favoriser le conflit, de le mettre en scène.

Cette liste montre l'importance du travail de groupe et rappelle des fonctionnalités récurrentes des collecticiels et particulièrement les systèmes de workflow : la gestion des activités de groupes. L'émergence d'une approche par activité a été accélérée par l'apparition du standard *Learning Design* du consortium IMS [4], standard généralement appelé IMS-LD. IMS-LD permet la description de processus pédagogiques, supporte tout types de pédagogies, suffisamment formel pour être interprétés par un ordinateur et n'est pas attaché à une plateforme particulière (Koper et van Es 2004). L'objectif d'IMS-LD est de la *"performance of individual and group learning activities designed to attain learning objectives and, in the process, making use of learning objects"* (Koper 2004). La notion de processus évoquée par IMS-LD a vite laissé place à la notion de scénario, dont on trouve différentes définitions :

1. une séquence orchestrée de phases dans lesquelles les apprenants ont des tâches à effectuer et des rôles spécifiques à jouer.(Schneider et al. 2003).

2. concrétisation du savoir didactique des enseignants lors de la réalisation d'un acte pédagogique, un cours, une séquence, l'ensemble d'un parcours (Koper et Tattersall 2005). La métaphore de la pièce de théâtre (reprise dans IMS-LD) indique les différents éléments d'un scénario : la méthode, la pièce, les actes, les rôles, les activités, l'environnement, les propriétés, les conditions, les ressources qui lui sont liées...

4. IMS signifiait au départ Instructional Management Systems. Mais son orientation vers l'enseignement supérieur a rendu caduque cette signification. IMS n'est plus un sigle.

3. par le design de scénarios pédagogiques, "le concepteur établit les liens entre les sources d'informations et les différents acteurs" (Paquette et al. 2002).

La figure suivante montre un exemple de scénario pédagogique au format IMS-LD.

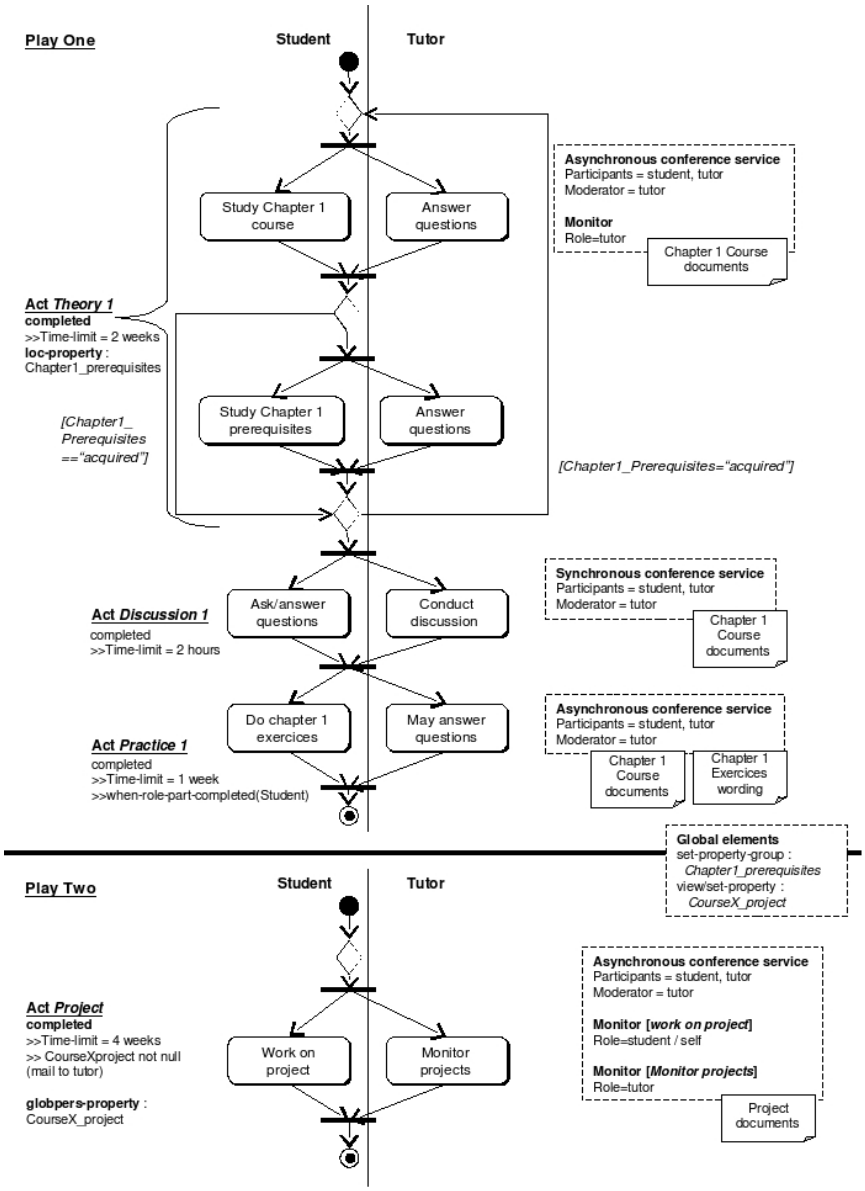


FIGURE 2.1 – *Exemple d'une séquence de cours classique modélisée en IMS-LD (Peter et al. 2007)*

Dispositifs pédagogiques

L'approche par activités ou scénaristique est particulièrement appropriée à une vision industrielle de l'enseignement à distance. C. Tattersal, R. Koper et G. Paquette sont dans des universités où le nombre d'étudiants à distance se compte par milliers voir par dizaines de milliers. Comme nous l'avons dit plus haut, à l'approche industrielle s'oppose une vision plus

artisanale avec un nombre d'étudiants limité (quelques dizaines) et une organisation où l'improvisation/l'adaptation a une place importante.

Pour ce type de contexte, l'approche par dispositif semble plus adaptée. Un dispositif *consiste en une organisation de moyens au service d'une stratégie, d'une action finalisée, planifiée visant à l'obtention d'un résultat* (Peraya 2000). Si on est tenté de produire un scénario pour indiquer comment utiliser ces moyens, il ne faut pas oublier que la façon dont son organiser ces moyens reflète l'intentionnalité - *intentionnalité flottante et transversale* (Berten 1999) - du ou des enseignants. En d'autres termes, le scénario peut se déduire (en partie) de l'organisation : *Le dispositif est une oeuvre ouverte qui invite l'enseignant, l'apprenant à l'interpréter dans son usage* (Paquelin 2005). Cette interprétation fait la force d'un dispositif car cela laisse suffisamment de liberté à l'improvisation et fait *simplement exister un espace particulier préalable dans lequel ce "quelque chose" peut se produire* (Peeters et Charlier 1999). Cette capacité d'improvisation et donc d'adaptation est fondamentale pour l'activité d'enseignement. (Huberman 1983) indique ainsi qu'une fois en classe, la préoccupation majeure de l'enseignant doit être de réagir aux élèves et non de suivre un scénario et qu'au final, *L'activité effective de l'enseignant et des élèves n'est jamais l'interprétation d'une pièce écrite par le maître* (Perrenoud 1983). Il est d'ailleurs intéressant de noter qu'aujourd'hui des millions de personnes utilisent Moodle ou Claroline qui sont principalement des plateformes orientées dispositifs (la création de workflow n'est possible qu'à l'aide de certains plug-ins).

C'est dans ce contexte - plateforme de formation, scénario et dispositif pédagogiques - que se sont déroulés les travaux autour de la viabilité du mapping vertical (voir chapitre 1 - Problématique n°1) du point de vue de l'humain.

2.1.2 Les interactions multimodales

L'équipe NOCE a été intégrée au LIFL en Juin 2007. En Juin 2008, j'ai souhaité accentuer mon travail sur l'IDM et m'éloigner des préoccupations liées aux EIAH. Pour cela, j'ai intégré l'équipe COCOA du LIFL, codirigée par Jean-Marc Geib et Bernard Carré. Ceci a eu pour conséquence de concentrer les travaux de thèse de Rim Drira [5] sur les modèles paramétrés, technique au coeur de l'équipe COCOA. Toutefois, j'ai eu l'occasion de travailler à partir de 2009 avec José Rouillard et Jean-Claude Tarby (tous les deux membre de l'équipe NOCE) sur les interactions multimodales (les IMM) et l'utilisation de l'IDM pour accélérer le développement incluant ce type d'interaction. Si cette collaboration était à la base prévue pour être anecdotique, elle s'est révélée longue du fait du projet ANR MOANO (2010-2014) [6] dont une des préoccupations était la production d'applications interactives multimodales à l'aide de l'IDM. J'ai été le référent lillois de ce projet où 3 équipes du LIFL participaient (ADAM, COCOA, NOCE).

5. Thèse commencée avant mon intégration dans l'équipe COCOA

6. Modèles et Outils pour Applications NOmades de découverte de territoire `http://moano.liuppa.univ-pau.fr/`

Que sont les interactions multimodales ?

Avant d'expliquer ce que sont les interactions multimodales, rappelons quelques définitions. Un système interactif est *un système dont le fonctionnement dépend d'informations fournies par un environnement externe qu'il ne contrôle pas* (Wegner 1997). Un ordinateur, un smartphone ou une maison intelligente est un système interactif et les informations fournies par leur environnement externe peuvent être des actions de l'utilisateur, la position de celui-ci, la température... Un capteur est *un organe qui élabore, à partir d'une grandeur physique, une autre grandeur physique, souvent de nature électrique, utilisable à des fins de mesure ou de commande* (dictionnaire Larousse). Les capteurs de contact des touches d'un clavier, le capteur CMOS d'une webcam ou de la caméra embarquée dans un smartphone, le gyroscope ou l'accéléromètre sont des exemples de capteurs utilisés dans les systèmes interactifs pour collecter des informations de l'environnement externe. Un périphérique (d'interaction) est un objet composé d'un ou plusieurs capteurs, agencés d'une certaine manière, qui communique avec un système interactif. Un clavier, une caméra, une kinect sont des exemples de tels périphériques. L'accéléromètre aussi est un périphérique d'interaction car on désigne généralement le capteur et le couple capteur/connexion au système par le même nom ("accéléromètre"). Une modalité d'interaction m est l'utilisation particulière d'un périphérique d'interaction p. Par utilisation particulière, nous entendons un sous-ensemble des possibilités d'interaction, sous-ensemble spécifié par un langage représentationnel r. La définition stricte est $m =< p, r >$ ou $m =< m, r >$ (Nigay 2001).

La plupart des systèmes proposent différentes modalités. Un ordinateur portable actuel propose en entrée 4 modalités : le clavier, le touchpad/trackpad, le micro et une caméra. Il en propose 2 en sortie : écran et haut-parleur. Une application est dite multi-modale si certaines interactions font intervenir plusieurs modalités[7]. Par exemple, un navigateur Web est une application multimodale car il est possible de cliquer sur un lien tout en appuyant sur la touche Shift (ouvrir la page référencée dans un autre onglet).

Les propriétés CARE caractérisent les relations possibles entre les modalités (Coutaz et al. 1995). Le cadre théorique dans lequel sont définies ces propriétés se base sur les concepts d'état (et de transition), de but et de relation temporelle (simultanée ou séquence dans un temps donné). CARE est l'acronyme de Complementarity, Assignment, Redundancy and Equivalence :

Complémentarité Si plusieurs modalités doivent être utilisées dans un intervalle donné pour passer d'un état à un autre. Le précédent exemple du click sur un lien Web avec la touche Shift enfoncée est une complémentarité.

Affectation Une seule modalité permet de passer d'un état X à un activité Y. Elle est *affectée* à cette transition. Par exemple, le bouton On/Off d'un smartphone est (généralement) le seul moyen de mettre en veille celui-ci directement.

7. On peut intégrer dans ces deux ensembles les adapteurs wifi, ethernet, bluetooth... notamment le wifi ou le bluetooth qui peuvent être utilisés pour savoir si un autre ordinateur ou smartphone est prés ou non

Équivalence Pour une transition donnée, deux modalités sont possibles. Par exemple, ouvrir un nouvel onglet dans un navigateur Web peut se faire en cliquant sur le + de la barre d'onglet ou appuyer sur les touches CTRL+N (ou Cmd+N pour Mac).

Redondance Pour une transition donnée, deux modalités sont équivalentes et peuvent être utilisées dans un intervalle de temps donné. Par exemple, une sonnerie et une vibration peuvent indiquer un appel entrant pour un smartphone alors que la sonnerie seule suffit.

Domotique

Les premiers tests que nous avons effectués concernaient des applications de domotique. Nous cherchions à l'époque un contexte permettant de rapidement mettre en oeuvre des interactions multimodales, et la domotique s'y prêtait bien. De plus, il était possible de travailler avec du matériel bon marché : notre collaboration démarrait, et nous ne pouvions investir massivement sans avoir au préalable des premiers résultats concrets.

La figure suivante montre un exemple de configuration sur lequel nous avons travaillé.

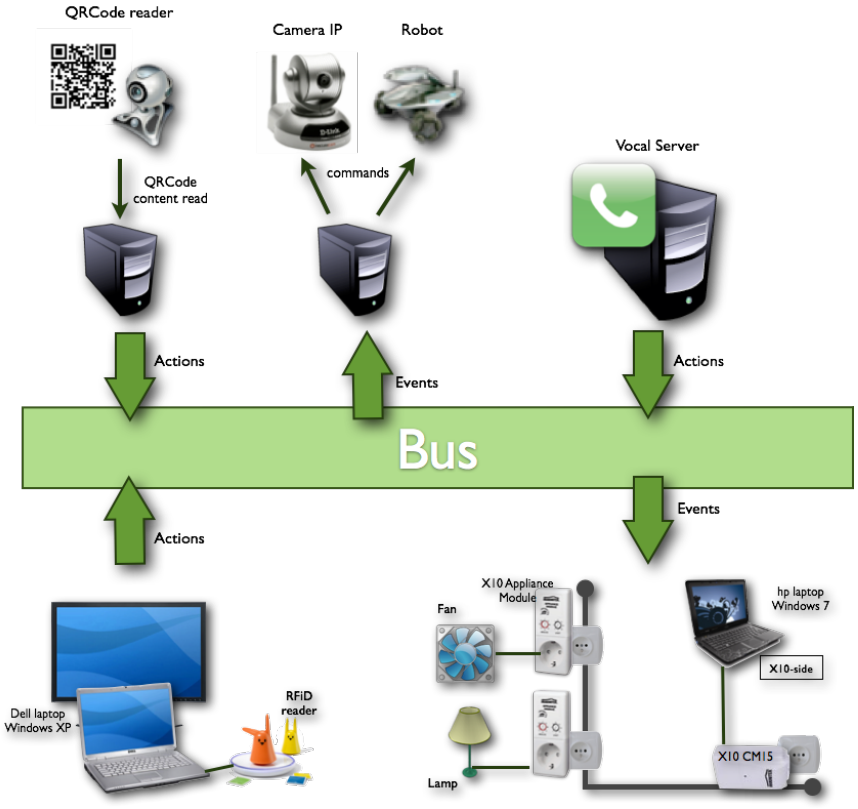


FIGURE 2.2 – *Exemple de système de domotique multimodal*

Dans cet environnement, "l'habitant" peut allumer ou éteindre un ventilateur et une lampe, déplacer la caméra de gauche à droite et de bas en haut et enfin déplacer une voiture robot (qui dispose aussi d'une caméra).

Pour faire ces différentes actions, l'habitant peut le faire vocalement (serveur vocal), via des tags RFiD ou à l'aide de QRCode. On a donc 3 modalités d'entrées différentes plus le clavier et la souris/trackpad des différents ordinateurs présents. Les modalités de sortie sont le serveur vocal (qui peut confirmer une action) ou les différentes écrans et haut-parleurs présents. Mais on peut aussi considérer les précédents équipements (lampe...) comme des modalités de sorties. Les combinaisons de modalités sont majoritairement des utilisations des précédentes modalités d'entrées et de la touche Shift ou Control en même temps. L'exemple est inspiré d'un environnement présenté dans les articles (Rouillard et al. 2010) et (Rouillard et al. 2011).

Exemple sur les applications mobiles

Une opportunité de collaboration avec le LIUPPA, le LIG et l'IRIT (le projet ANR MOANO) a déplacé notre terrain d'étude aux applications mobiles pour smartphones avec un focus sur les activités réalisées en plein air (ex : relevés naturalistes, topologiques...). Nous avions déjà utilisé les smartphones dans le contexte de la domotique, et le nombre de capteurs intégrés (accéléromètre, capteur de proximité, de lumière...) nous avaient tout de suite interpellés quant au fait que le smartphone était un bon support d'étude pour les interactions multimodales : on peut effectuer des commandes tactiles, les dicter oralement, effectuer des gestes avec le téléphone (secouer, pencher à gauche/droite) ou devant la caméra frontale, et le téléphone peut afficher des informations, les diffuser de manière sonore, vibrer. . ..

L'exemple suivant, venant de l'article (Elouali et al. 2013), montre un exemple d'application multimodale.

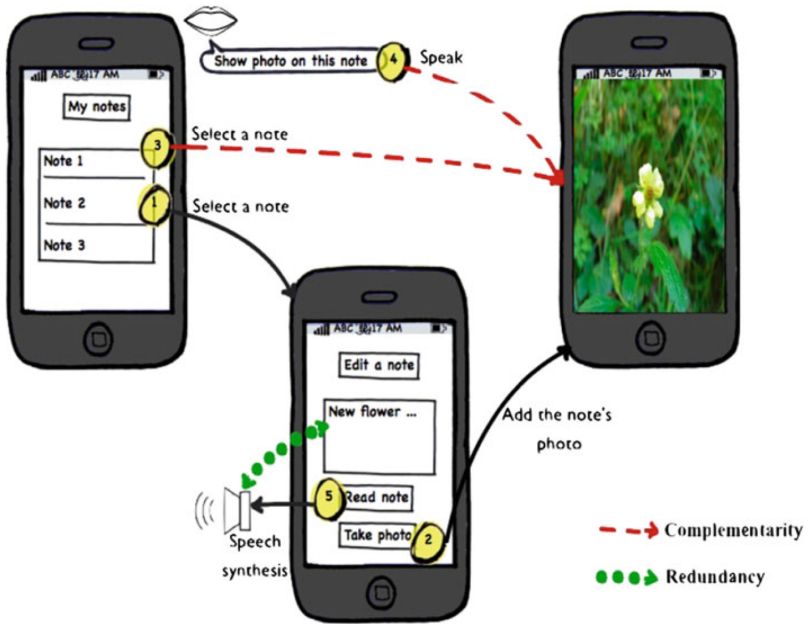


FIGURE 2.3 – *Exemple d'application mobile multimodale*

Dans cette application de prise de notes, l'utilisateur peut appuyer/toucher sur une des notes affichées dans la liste de l'écran de départ et, sur le nouvel écran qui apparaît - dédié à la note sélectionnée - appuyer sur "prendre une photo" pour prendre une photo et l'associer à la note. Il peut arriver au même résultat en sélectionnant une note au départ et, tout en appuyant sur la note, dicter la commande "prendre une photo" (reconnaissance vocale). Cette combinaison est une complémentarité du tactile et de la voix. Enfin, pour la redondance, le contenu d'une note est affiché dans l'écran qui lui est dédiée mais peut être aussi dictée par le smartphone (synthèse vocale).

2.2 Les outils développés

Cette section décrit deux outils que j'ai développés qui ont été des supports importants pour différents travaux de recherche.

ModX[8] est un outil de méta-modélisation qui a été le support à la thèse de Pierre-André Caron (partie 3.1.3), au Processus de Modélisation Incrémentale (partie 4.1), au début de la thèse de Nadia Elouali (partie 3.2), et au travail sur les environnements intelligents.

Miny[9] est un bus "Web" à événement qui a été utilisé pour les environnements intelligents, dans de nombreux projets étudiants (notamment ceux de Jean-Claude Tarby sur le BCI - Brain Computer Interaction) et qui est, au moment de l'écriture de ce rapport, au centre de l'outillage développé par l'équipe Carbon.

2.2.1 ModX, un outil de méta-modélisation

En décrivant l'Ingénierie Dirigée par les Modèles, le chapitre 1 a rappelé l'importance de la modélisation en Ingénierie Logicielle et la nécessité de pouvoir créer plusieurs formalismes de modélisation afin d'adresser chaque préoccupation au travers d'un langage dédié et donc, si possible, optimisé. En 2003, le MDA avait été adopté depuis 2 ans, EMF[10] - la référence en IDM aujourd'hui - faisait de timides débuts, la première version de GMF[11] allait sortir dans trois ans et les rares outils de méta-modélisation graphique, c'est-à-dire permettant de créer des éditeurs de modèles graphiques, n'étaient pas compatibles avec le standard MOF de l'OMG[12]. Pour débuter des travaux sur l'IDM et notamment son application à l'Ingénierie des EIAH, il nous fallait un outil permettant de spécifier facilement des méta-modèles et de créer aussi aisément les éditeurs de modèles associés. Ces éditeurs devraient être "simples"... dans le sens "accessibles à des non-informaticiens", par exemple des enseignants. Les éditeurs de modèles devaient au minimum être graphiques au sens de

10. Eclipse Modeling Framework
http://www.eclipse.org/modeling/emf/
11. Graphical Modeling Framework
http://www.eclipse.org/modeling/gmp/
12. MetaEdit+ (http://www.metacase.com/fr/mep/) est basé sur GPPR, ATOM3 sur le formalisme ER...

Moody. L'impulsion de départ a été donnée via le réseau d'excellence Kaleidoscope [13], où un work package consistait à étudier l'IDM dans le cadre des EIAH. La première version de ModX a été un des livrables de ce WP.

Principales fonctionnalités

La première fonctionnalité de ModX est de permettre de spécifier un méta-modèle au format MOF 1.4 au travers de la syntaxe graphique UML et d'un navigateur de propriétés de type formulaire.

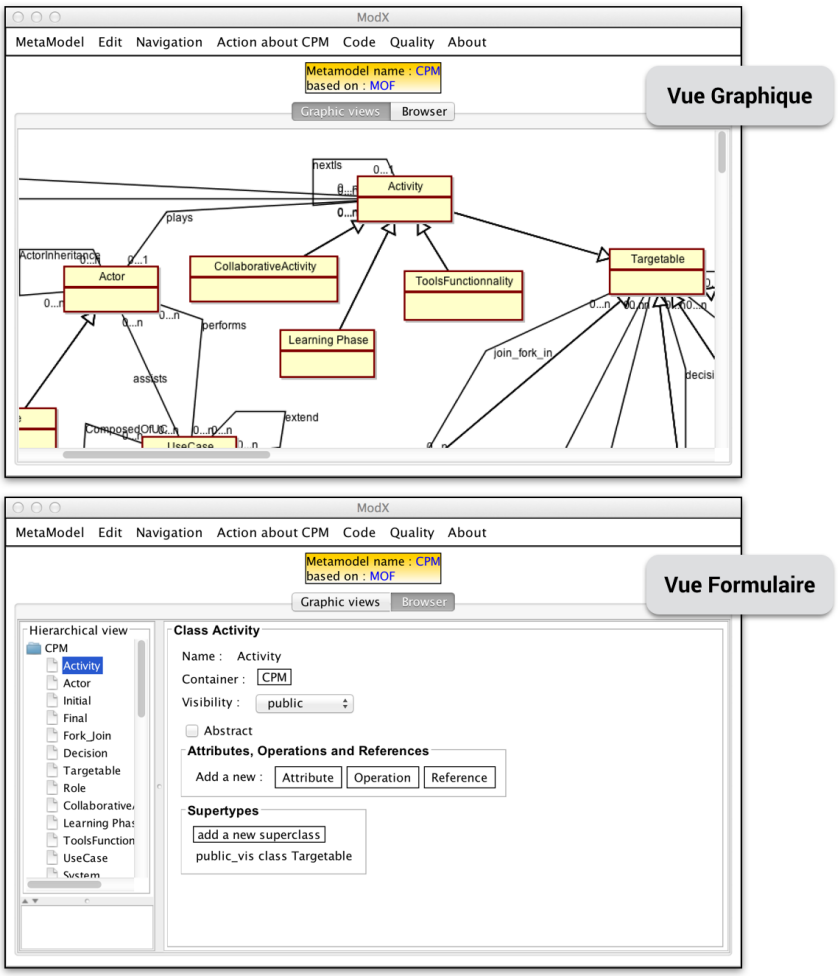


FIGURE 2.4 – *Les deux types de vue d'un méta-modèle dans ModX*

La deuxième fonctionnalité est de pouvoir définir et associer une ou plusieurs syntaxes concrètes (appelées dans ModX des *view types*) à un méta-modèle.

13. http://www.noe-kaleidoscope.org/telearc/

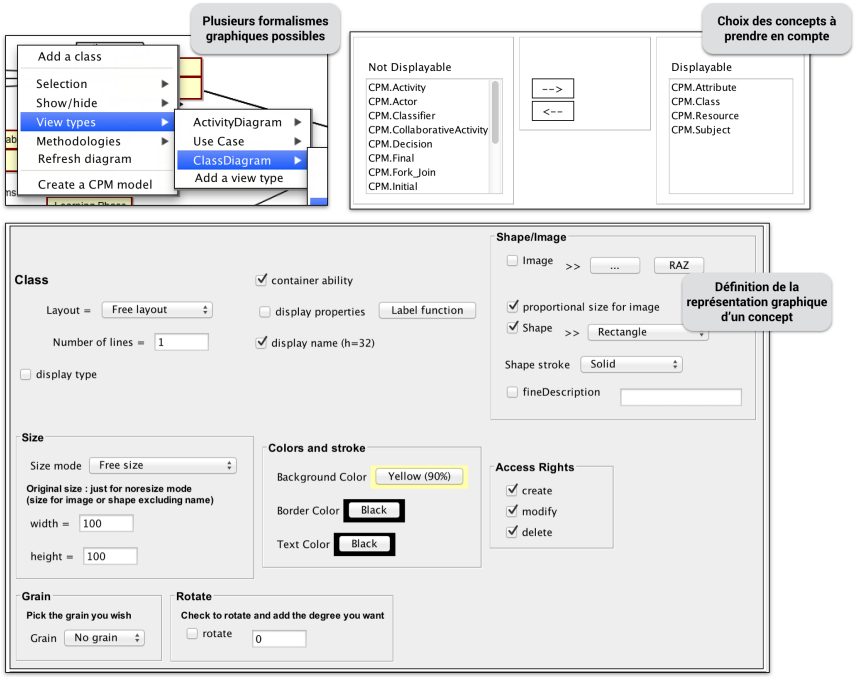


FIGURE 2.5 – *Définition d'une syntaxe concrète / formalisme graphique*

Enfin la dernière fonctionnalité est de pouvoir "instancier" un méta-modèle et de pouvoir l'éditer au travers des syntaxes concrètes associées.

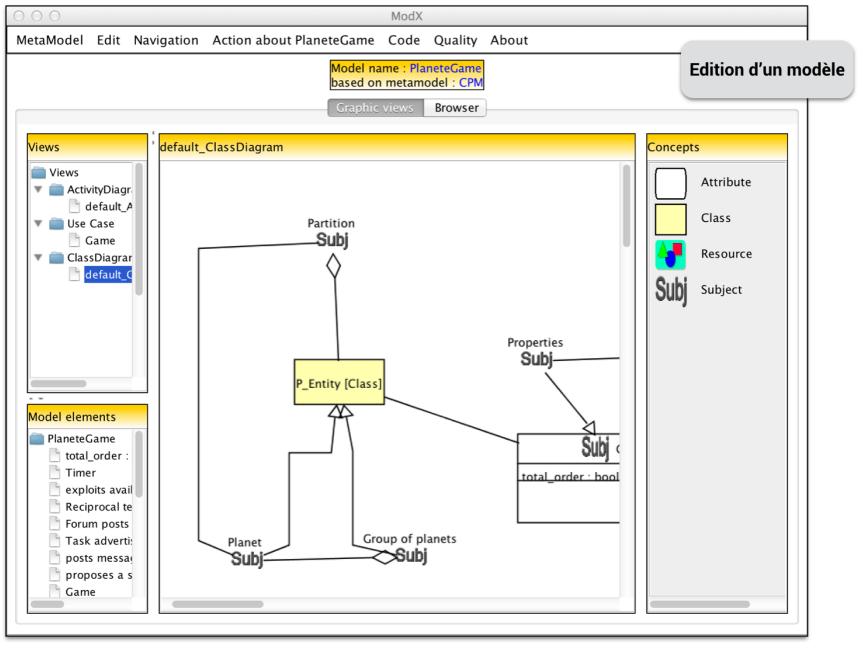


FIGURE 2.6 – *Édition d'un modèle*

La persistance des méta-modèles et modèles est conforme à la norme XMI de l'OMG [14]. Des fonctions d'importation/exportation avec le format XMI de EMF sont présentes (implémentées par Simon Urli).

14. XML Metadata Interchange (XMI)
http://www.omg.org/spec/XMI/

Aspects dynamiques

Quand un ou plusieurs modèles conformes à un méta-modèle sont chargés dans ModX, il est toujours possible de modifier le méta-modèle et/ou une de ses syntaxes concrètes à tout moment. La répercussion sur le ou les modèles se fait instantanément. Pour la modification d'un méta-modèle, les règles de répercussion sont "simples" (mais pas leur implémentation) :

1. l'ajout d'une classe dans le méta-modèle permet de créer des instances de celle-ci dans tout modèle ouvert
2. la suppression d'une classe supprime toutes ses instances dans les modèles chargés
3. la modification d'une classe suit à peu près le même principe pour l'ajout ou la suppression d'un attribut ou d'une référence. Par contre, la modification de la cardinalité de ces derniers supprime les valeurs existantes de propriétés/instances. Enfin la modification de l'héritage ajoute/supprime les attributs/références en conséquence.

La représentation graphique des modèles étant recalculée à chaque modification du modèle ou de son méta-modèle (syntaxe abstraite ou concrète), la modification de la syntaxe concrète est moins problématique conceptuellement (mais encore une fois, l'implémentation reste complexe).

Enfin, les éléments des syntaxes abstraites, concrètes et des modèles (données et représentations graphiques) sont accessibles et manipulables dynamiquement au travers d'un interpréteur Javascript. J'ai utilisé pour cela la librairie Rhino de Mozilla [15] et fait le mapping entre les objets de ModX et Rhino. Cet accès Javascript est utile pour faire des transformations de modèles, de la génération de code (une API de génération de code Java a aussi été implémentée mais il est possible de générer du 'texte' et donc tout type de code) ou pour faire tout type de traitement sur des modèles (affichage conditionnel, recopie conditionnelle...).

Conclusion

Le tableau ci-dessous montre que ModX a été un des premiers outils de méta-modélisation et de modélisation graphique basé sur le standard MOF. Avec MetaEdit+ (non standard et coûteux), ModX est encore le seul outil à proposer la création de méta-modèles, d'y associer plusieurs notations visuelles, de créer des instances/modèles correspondants, et de pouvoir changer le méta-modèle et les notations, le tout de manière dynamique.

15. `https://developer.mozilla.org/fr/docs/Rhino`

	MetaEdit+ (Tolvanen et al. 2007)	OpenTool (Lions et al. 2002)	Atom3 (De Lara et al. 2004)	EMF (Boldt et PATER-NOSTRO 2006)	ModX	GMF (Herrmannsdoerfer et al. 2010)	Eugenia (Eugenia Foundation 2013)	xOWL (Wouters 2012)
Année de création	1995	2001	2002	2002	2003	2006	2009	2010
Origine	MetaCase	TNI-Valiosys	McGill University	IBM	Trigone	Open Source (plusieurs pays)	University of York	EADS/LIP6
Langage d ?implémentation	Smalltalk	OTScript / profile UML	Python	Java	Java	Java	Java	Java
Format natif	GOPRR	OTScript	Entité-Relation	XMI-EMF	XMI-MOF	XML	XMI-EMF	OWL
Formats supportés		?		XMI-EMF	XMI-EMF	XMI-EMF	XMI-EMF	OWL
Payant / Open Source	Payant	Open source	Open source	Open source	Open source	Open source	Open source	?
Support pour définir la syntaxe abstraite	Graphique	Graphique	Graphique	Graphique / Arbre	Graphique	via EMF	via EMF	Graphique
Support pour définir la syntaxe concrète	Formulaire	?	Grammaire dédié	Non	Formulaire	XML	Emfatic	Grammaire textuelle dédiée
Support pour créer des modèles	Oui	Graphique	Formulaire + Graphique	Arbre	Formulaire + Graphique		via GMF	Graphique
Intercession	Oui	Possible	Non (compilation nécessaire)	Oui (mais rarement utilisé)	Oui	"Non via XML Oui via Java"	Non (compilation nécessaire)	Non (compilation nécessaire)
Manipulation de modèles	Langage de script dédié	OTScript	Python	Java + langages Epsilon	Javascript	Java + langages Epsilon	Java + langages Epsilon	

TABLE 2.1 – *ModX et les autres outils de méta-modélisation*

Des outils basés sur EMF - comme GMF, Eugenia ou Obeo Designer (cité dans le chapitre 5) - qui sont apparus entre 3 et 6 ans après sont certes plus stables mais sont malheureusement plus complexes à utiliser : ces outils sont intégrés dans Eclipse et EMF, et il est nécessaire de connaître le fonctionnement précis d'EMF (nécessité d'une classe Racine, génération obligatoire de l'API Java associé au méta-modèle, interface obscure...). Ceci rend ce type d'outils inaccessibles aux non-informaticiens et les problèmes de compatibilité de versions entre plug-ins peuvent repousser les plus courageux.

Les caractéristiques de ModX nous ont permis de l'utiliser dans de nombreux contextes et projets :

1. Processus de Fabrication Alimentaire : projet étudiant GIS polytech 2006-2007 en collaboration avec l'IAAL [16]
2. Composants de services (la norme SCA - Service Component Architecture) : projet étudiants e-services
3. Application ubiquitaires : le projet CPER Mosaique
4. e-Business XML : TP du module "solutions technologiques" du master e-services
5. e-Learning :
   (a) la norme IMS-LD (Le Pallec et al. 2006b, Peter et al. 2007)

16. www.polytech-lille.fr/genie-biologique-et-alimentaire-p121.html

   (b) le DSML CPM du LIUPPA (Nodenot et al. 2007)
   (c) les plateformes Moodle (Caron et al. 2007d;c), Ganesha (Caron et al. 2005), Claroline (Peter et al. 2007), WikiniMST (Caron et al. 2007a)
   (d) les EAPC (Caron et al. 2007b) - cf. partie 3.1.3

6. les IHM :
   (a) le design pattern PAC (aka MVC2) (Tarby et al. 2006) - cf. partie 4.1.3
   (b) les interactions multimodales (Rouillard et al. 2011, Elouali et al. 2012)

7. Mesure de la qualité des notations

Comme le montre cette liste, j'ai souvent utilisé ModX pour créer rapidement des chaines IDM. Ce n'est que récemment que j'ai décidé d'essayer d'autres outils pour ma recherche, notamment ceux basés sur EMF comme Obeo Designer (pour la thèse de Nadia Elouali et celle de Daniel Liabeuf). Toutefois, les difficultés dues aux problèmes de compatibilité entre plug-ins, la lenteur d'Eclipse (notamment pour des modèles volumineux) et le peu d'ergonomie des outils (de nombreux clics sont nécessaires pour définir une vue, associer un symbole à un concept...) ne m'ont pas convaincu d'abandonner ModX. C'est la raison pour laquelle, mes récents travaux sur la qualité des notations visuelles (cf. chapitre 5) se sont faits à l'aide de ModX. C'est aussi pour cela que je réutilise ModX - depuis 1 an - pour mes cours d'IDM en master 2 : il permet de se focaliser sur les notions essentielles en évitant de trop nombreuses considérations techniques/ergonomiques.
Pour le futur, je souhaite faire évoluer ModX techniquement et conceptuellement. Techniquement en ré-implémentant cet outil dans un environnement pur Web afin de profiter des capacités collaboratives du Web et surtout l'absence d'installation pour les utilisateurs. De plus, la technologie SVG permet non seulement un rendu de bien meilleure qualité mais aussi des fonctions d'affichage plus évoluées que Java. Conceptuellement, car l'utilisation de la méta-modélisation classique (définition des concepts PUIS modélisation) semble montrer ses limites dans certains retours d'expérience de professionnels : ils semblent préférer l'utilisation d'outils de présentation comme MS PowerPoint car ils sont plus laxistes et donc offrent plus de liberté. Des travaux comme (Guychard et al. 2013) autour d'OpenFlexo [17] ou ceux de (Sanchez-Cuadrado et al. 2012) autour de MetaDepth [18] montre une autre approche où la modélisation commence non pas par définir un méta-modèle mais par l'écriture d'un diagramme et l'identification ensuite des concepts communs aux éléments du modèle. S'il n'y a pas encore de preuves scientifiques pour prouver que cette approche ascendante semble mieux acceptée par les concepteurs, les discussions avec ces derniers - à la suite de la manipulation de tels outils - montrent une préférence pour celle-ci. C'est donc vers cette approche a

17. https://openflexo.org/
18. http://astreo.ii.uam.es/~jlara/metaDepth/

priori mieux adaptée aux besoins des concepteurs, et donc plus ergonomique, que mes prochains développements se dirigeront.

2.2.2 Miny, un support aux applications Web multimodales

Comme nous l'avons expliqué précédemment pour nos travaux sur la domotique, nous nous sommes intéressés aux périphériques d'interaction peu onéreux (quelques centaines d'Euros maximum) et, point important, avec des pilotes stables et dont l'installation est simple. Sont candidats des périphériques comme la Kinect, le casque cérébral Epoc, le lecteur RFiD mir :ror, le Leap Motion, la WiiMote... Ce type de périphériques fleurit depuis quelques années, et les blogs de veille technologique leur font la part belle. L'intérêt pour nous réside dans l'API qui est souvent fourni avec le pilote et qui permet à toute personne ayant des notions de programmation de tester le périphérique dans d'autres applications que celles fournies et, par la même occasion, de créer de nouvelles actions possibles, voir de nouvelles modalités d'interaction. Après plusieurs maquettes et prototypes réalisées, nous nous sommes rendus compte qu'il serait très avantageux de pouvoir construire des applications multimodales qui soient des applications Web. En effet, placer la logique d'interactions dans une application Web permet de bénéficier des points forts de celle-ci. Ainsi l'application :

1. fonctionnera sur un grand nombre de terminaux et systèmes d'exploitation. La maturité prochaine d'HTML 5 ne fera que consolider cet état de fait en devenant un concurrent sérieux aux applications "natives".

2. ne nécessitera pas d'installation (et donc de droits particuliers).

3. pourra être facilement modifiée à l'exécution - grâce aux propriétés réflexives du langage Javascript - sans perte de performance. Implémenter des capacités d'adaptation au contexte est plus simple avec ce langage qu'avec des langages (pré-)compilés comme Java ou C#.

Dans une utilisation classique, les périphériques d'interaction peuvent se trouver soit sur la machine où est exécutée l'application Web soit sur une machine distante (ayant accès au périphérique). Donc pour gérer les différentes interactions possibles au sein de la page Web, il faut un bus de communication qui permette à celle-ci de communiquer avec ces "processus extérieurs". Un tel bus va devoir répondre à 5 contraintes dues à ce contexte particulier.

Les 5 contraintes pour des applications Web multimodales

Tout d'abord, le support doit permettre une programmation répartie sur un réseau hétérogène (Wifi, 3G) et peuplé de pare-feux (Contrainte n°1). Les pilotes des périphériques d'interaction ne sont pas tous accessibles au travers du même langage : cela peut être C# (Kinect), C (Epoc), Java (Android)... Ceci implique l'accès au support logiciel à partir des langages généralement utilisés (C2). Afin de faciliter l'utilisation de périphériques, le support doit être livré avec des bibliothèques associées à des types de périphériques existants (Smartphone, Kinect...) (C3). Le support doit être

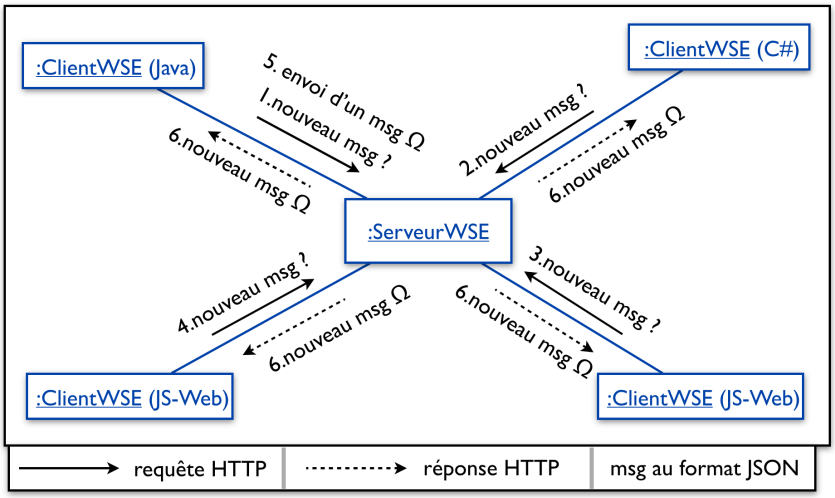
accessible au sein d'une page Web en Javascript (C4). Enfin, la persistance des messages est très pratique dans un contexte réparti pour rejouer un ensemble de messages à des fins de stockage ou de "undo" (C5).

La non-adéquation des bus actuels

Nous avons étudié trois types de supports logiciels. Le premier d'entre eux concerne les environnements dédiés aux interactions multimodales comme OpenInterface (Interface 2008) ou Squidy (König et al. 2009) Ils fournissent des plateformes puissantes permettant la conception de nouvelles modalités et la combinaison de composants interactifs. Toutefois ces plateformes ne permettent pas un accès via des pages web (C4), leur installation est fastidieuse et l'ajout de périphériques distants n'est pas facilité. Les intergiciels constituent le second type de supports étudié. Nous avons testé Corba-RMI, JMS (Oracle 2014) et IVY(CENA 2014). Le principal inconvénient revient à l'utilisation de sockets qui vient en contradiction avec C1. Des plug-ins sont possibles pour une utilisation sur HTTP mais complexifie l'installation. Enfin, les plateformes à composants (comme Frascati (OW2 2013)) proposent également des supports intéressants avec de nombreux protocoles de communication (socket, HTTP). Si la traçabilité (C5) et l'accès au sein de page Web (C4) sont parfois présents, l'installation de ces plateformes et leur utilisation lors du développement sont plus complexes que les autres. Enfin seul le premier type de supports proposent des composants interactifs (C3). Nous nous sommes enfin intéressés aux web sockets. Malheureusement, même si elles semblent être une technologie adaptée, nous la jugeons peu stable et sans API standard pour l'utiliser à partir d'autres langages (C2).

WSE : un bus à événements Web

N'ayant pas donc de bus correctement adapté à nos objectifs, j'ai décidé d'en développer un conforme à nos attentes. Nommé WSE (Web Server Event), il a été conçu pour être orienté événement/message et basé sur le protocole HTTP.



Fonctionnement de WSE

Comme on le voit sur le schéma ci-dessus, les éléments constituants le bus sont

- Des librairies Java, Javascript et C# qui permettent à des applications Java, Web ou .Net d'émettre et de recevoir des messages (contraintes 2 et 4 respectées).
- un ensemble de script PHP - sur un serveur Web - qui reçoit les demandes d'émission de messages et qui les diffuse à tous les "clients" connectés.

Le principe de fonctionnement "bas-niveau" est illustré sur le schéma : chaque client envoie une requête HTTP pour demander s'il y a de nouveaux messages (1-2-3-4). Si un client envoi une requête HTTP pour émettre un nouveau message M1 (5), alors les précédentes requêtes HTTP auront une réponse (6) contenant le message M1. S'il n'y a pas de nouveaux messages pendant un certain laps de temps (fixé par défaut à 30s), une réponse similaire à la requête 6 est envoyé mais cette fois, elle indique qu'il n'y a pas de nouveau message. Dans un cas ou dans l'autre, à la réception de la réponse, les librairies émettent automatiquement une requête HTTP pour savoir s'il y a un nouveau message. On a donc un fonctionnement de type COMET(Wikipedia 2014a) avec une approche long-polling. Cette approche est conseillée pour les cas où il y a rarement un nombre important de clients connectés simultanément au serveur, et c'est justement le contexte que nous visons.
L'installation est similaire à celle d'un CMS (des fichiers à placer sur le serveur Web, un installeur qui vérifie les droits d'écriture) : notre contrainte sur la simplicité d'installation est ainsi respectée. De plus, depuis Janvier 2014, WSE intègre un mécanisme CORS[19]. Il est donc possible d'écrire une page Web qui utilise WSE et de l'exécuter directement sur sa propre machine sans la mettre sur un serveur. Ceci facilite le déploiement d'une application Web utilisant WSE.
À un niveau plus applicatif, les librairies WSE Java, Javascript et C# présentent toutes une méthode pour envoyer des messages et un message d'observeur/observable pour être notifié des nouveaux messages. Les communications entre entités logicielles sont filtrées par un mécanisme de session : chaque message envoyé sur WSE a sa portée limitée aux participants de la session associée. Les messages échangés sont des objets JSON (format utilisé au sein des architectures RESTful).
Le code suivant montre une utilisation de la librairie Javascript

```
// joindre une session
wse.joinSession("HDR_XLP");

// envoyer un message
wse.sendMessage(
  {objet : "Un rapport sympa", qualite : "très bonne"}
);

// écouter les messages
```

19. Cross-Origin Resource Sharing (Wikipedia 2014b)

```
auditeur = {};
auditeur.newMessageReceive = function (message) {
  alert("J'ai reçu un message sur WSE : "+message);
};
wse.addListener(auditeur);
```

Chaque message est numéroté et cette numérotation suit un ordre croissant. La connexion à une session consiste en partie à demander quel est le numéro du dernier message afin de demander les messages qui ont un numéro plus grand. Si d'ordinaire, il ne faut pas y prêter attention, ce mécanisme a pour avantage de permettre de rejouer des messages précédents en fixant volontairement le numéro du dernier message à un rang inférieur. Ceci nous permet de respecter la contrainte 5.

Miny : vers une plateforme dédiée à la multimodalité

L'utilisation d'un bus à message peut vite devenir fastidieuse dans des contextes où les liens entre entités logicielles communicantes sont nombreux : il faut implémenter dès lors un aiguillage des messages au sein de chaque entité et des fonctions de "marshalling/ unmarshalling". Pour ce type de contexte (que nous avons rencontré souvent par la suite), j'ai implémenté un générateur de code qui permet d'utiliser WSE avec une approche objet, comme le bus CORBA. Le concept d'événement étant fondamental en IHM, j'ai associé le principe d'événement à la notion d'objet, en plus des méthodes : une entité logicielle peut invoquer une méthode sur un objet distant, mais aussi s'abonner aux événements qu'il peut diffuser.
Ce mécanisme nous a permis d'encapsuler différents périphériques d'interaction et donc simplifier leur utilisation sur le bus. Ce générateur peut également être utilisé pour des périphériques que nous n'avons pas encore traités, mais aussi rendre facilement accessible toute entité logicielle (objet, module, fonction) distante. Dans les deux cas, le développeur aura à définir l'interface publique correspondante au périphérique/entité logicielle. Par exemple, s'il souhaite définir un type d'objet pour les smartphones (tout OS confondu) avec la méthode *vibrer* et l'événement *accelerometre*, l'interface sera déclarée ainsi :

```
{
   name : "Smartphone",
   ...
   methods : {
     vibrer : {
       duree : Integer
     }
   },
   events : {
     accelerometer :
     {
       x : Integer, y : Integer, z : Integer
     }
   }
```

```
}
```

À l'utilisation, c'est-à-dire une fois le code généré, du côté du téléphone, la création d'un objet de ce type s'effectuera simplement comme une instanciation. Il faudra néanmoins associer un identifiant sous la forme d'un couple de valeurs "location" et "locationParams". Ainsi, les entités distantes souhaitant se connecter à cet objet devront indiquer les mêmes valeurs pour ces deux paramètres. Quand une entité se connecte à un objet de type smartphone sans indiquer de valeurs pour ces deux paramètres, elle sera connectée à tous les objets distants de type "smartphone". Ce principe d'identification est moins automatique que l'utilisation d'un IOR sous CORBA et RMI, mais le fait de pouvoir être connecté à plusieurs objets distants de même type via un seul objet local permet des scénarios plus riches en matière d'interaction.
Le code ci-dessous illustre l'utilisation de cette sur-couche pour le cas du smartphone présenté plus haut.

```
// récupérer l'adaptateur pour le téléphone enregistré
// à location = ergo & locationParams=jc
var jeanAndrophone=manager.getSmartphone("ergo","jc");

// demande d'action de vibration sur ce
//téléphone (100ms)
jeanAndrophone.vibrate(100);

// enregistrer une fonction de "réaction" à tout
// événement venant de l'accéléromètre
jeanAndrophone.accelerometer = function (z, y, x)
  { // réaction à l'événement accéléromètre}
```

WSE/Miny ont été présentés à UIST 2010 (Le Pallec et al. 2010) et Ergo'IHM 2012 (Le Pallec et al. 2012) [20].

Conclusion du chapitre

L'étude approfondie du e-Learning et des interactions multimodales [21] et la collaboration avec des experts de ces domaines, n'ayant pas de connaissance en IDM, ont été fondamentales dans mon activité de recherche car elles m'ont permis de mieux percevoir les attentes des utilisateurs d'outils IDM en terme d'utilisabilité. Certains de ces experts n'hésitaient d'ailleurs pas à remettre complètement en question certains principes de l'IDM, comme modéliser de manière abstraite un système ou utiliser des diagrammes de type UML ("des boites et des flèches") : l'exemple le plus amusant est celui où Pierre-André a demandé un jour à Gilles Leclerc

20. parmi les 3 finalistes pour le meilleur papier

21. Jean-Claude Tarby et moi avons eu la chance d'être invités au journées Flupa UX-Day en Mai 2012 à Paris pour présenter les différentes modalités proposées par les smartphones actuels http://fr.slideshare.net/flupa/flupa-uxday-2012-atelier-usage-des-modalits-dinteraction-sur-smartphones-par-jc-tarby-x

(Professeur en Sciences de l'Éducation) de montrer comment il s'y prendrait pour schématiser son dispositif pédagogique, celui-ci, à la manière de St Exupéry, lui a dessiné une boite et lui a dit que son dispositif était dedans. Et il était malheureusement assez sérieux. Tout ceci a nourri le cahier des charges de ModX tout au long de son développement, et m'a permis d'avoir une certaine distance par rapport à l'IDM.
Comme je l'ai indiqué dans ce chapitre, ModX s'est révélé au fil des différentes expériences (utilisation par "mes" étudiants, "mes" collègues et moi-même) ergonomique, rapide et proposant un ensemble de fonctionnalités que seul MetaEdit+ (généralement considéré comme le leader du marché) offre. Sa principale faiblesse est son instabilité : pour des activités de recherche (et donc de prototypage), cela n'est pas un aspect rédhibitoire.
Miny et le bus WSE vont continuer à être utilisé dans mes prochaines activités de recherche : pour la simple et bonne raison qu'il n'y a pas encore d'équivalent sur le "marché". Certains aspects manquent encore de robustesse (notamment le démarrage de session ou l'envoi simultané de plusieurs messages par le même périphérique), mais les prochains doctorants participeront à la maintenance de WSE et corrigeront sûrement ces faiblesses.

Mapping vertical du point de vue de l'humain

3

Sommaire

Ce chapitre relate les travaux que nous avons menés sur le mapping vertical de modèles dans le cadre du e-learning et des interactions multimodales. Après un rappel de l'intérêt de l'IDM pour le e-learning (partie 3.1.1) et la description de l'outillage nécessaire qu'il a fallu développer pour les plateformes associés (partie 3.1.2), nous relatons les travaux de Pierre-André Caron (partie 3.1.3). Ils ont porté sur la compréhension qu'avaient les enseignants des modèles issus d'une transformation de modèles - modèles qu'ils avaient eux-mêmes spécifiés. L'objectif de la transformation de modèles étaient la projection technologique. Le changement conceptuel et, en particulier, l'ajout de préoccupations technologiques a constitué un obstacle majeur à la compréhension des modèles générés : certains éléments générés étaient trop techniques et ont empêché les enseignants de retrouver certains éléments spécifiés dans leur modèle source/abstrait. La solution proposée par Pierre-André a été de mixer les deux perspectives de modélisation en une seule. Si les enseignants ont beaucoup mieux accepté et intégré cette solution, elle faisait faire un bond

en arrière (pour ressembler aux outils CASE), et supprimait deux avantages importants de l'IDM : la séparation des préoccupations et la projection multi-plateformes. Afin de permettre aux enseignants de mieux maîtriser cette projection technologique, Rim Drira a, pendant, sa thèse élaboré une approche où le mapping vertical se fait manuellement / humainement afin de pallier au problème de traçabilité (partie 3.1.4). Se faisant, elle a du outiller de façon explicite l'intégration de bonnes pratiques, qui se sont révélées être ici le point fondamental du mapping vertical de type MDA.
Dans une approche plus horizontale [1], cette projection technologique devient encore plus problématique lorsque celle-ci prend une part importante des préoccupations d'une perspective de modélisation. Si on souhaite créer suffisamment de concepts pour permettre une spécification suffisante et si la technologie visée évolue rapidement, le risque est grand d'avoir un problème de versionning à l'usage avec le méta-modèle résultant. L'intégration d'une librairie d'éléments réutilisables semble être une solution plus efficace, mais accentue le problème de la compréhension des concepts : un mauvais choix de concept peut avoir un impact important sur l'utilisabilité d'une telle librairie. Et, quelques soient les choix conceptuels, quel(s) mécanisme(s) utiliser pour "instancier" ces éléments de librairie ? La partie 3.2 relate notre expérience relative à ces problèmes dans le cadre des interactions multimodales.

1. mais où le mapping vertical reste une phase de la construction : par exemple, la génération de code

3.1 Vérification par l'humain de la cohérence entre modèle source et modèle généré

3.1.1 Intérêts de l'IDM / plateformes pédagogiques

Pourquoi une approche IDM peut ou plutôt pouvait se révéler pertinente au milieu des années 2000 pour le e-learning ? Et plus particulièrement pour le déploiement de dispositifs pédagogiques sur des plateformes pédagogiques...

La réutilisation. Construire un dispositif sur une plateforme implique une série de tâches vite rébarbatives si elles doivent être répétées chaque année. Ces tâches consistent par exemple à définir des espaces de documentation, de partages et de dépôts de travaux, des conférences asynchrones dédiées à des sujets particuliers, indiquer la chronologie, fournir un descriptif détaillé du cours associé en explicitant par exemple les objectifs, les modalités d'évaluation, les modes d'apprentissages... L'espacement entre deux promotions ne permet pas à l'enseignant de s'habituer à la plateforme et d'acquérir de manière pérenne la maîtrise de la plateforme. Une fonction de sauvegarde du dispositif est bénéfique ici [2] afin d'éviter la reconstruction du dispositif pour chaque nouvelle promotion. Toutefois une telle fonctionnalité doit alors prendre en charge plusieurs caractéristiques sophistiquées parmi lesquelles on dénombre par exemple :

- la sélectivité des éléments du dispositif à sauvegarder, certains messages sur un forum pouvant par exemple faire partie des informations à reconstruire,
- l'abstraction de certains éléments construits comme par exemple la vue d'ensemble d'un dispositif, celle-ci étant construite sur la base des éléments existants ...

Le partage. Actuellement, partager un dispositif pédagogique entre deux enseignants nécessite pour eux l'utilisation d'une même plateforme de formation. En effet, faute d'un langage de spécification de dispositifs indépendant des plateformes de formation, il est difficile pour un enseignant d'exprimer un dispositif indépendamment des choix technologiques qui le mettent en oeuvre. Ceci implique pour l'enseignant, auteur du dispositif, de devoir extraire les intentions pédagogiques de son dispositif technologique. Cette extraction est rendue difficile par divers facteurs, comme l'oubli des intentions qui ont permis la création du dispositif, le caractère semi-improvisé des activités menées par un enseignant ou l'évolution naturelle que subit un dispositif pédagogique dès lors qu'il implique chez les apprenants des activités de co-construction. De fait, le modèle sous-jacent à un dispositif pédagogique est en constante évolution et une grand partie n'est pas explicitée.

L'alternance technologique. Une formation ou un enseignant peut être amené à changer de plateforme afin de s'adapter à des changements de son contexte (changement de politique universitaire, de direction) ou plus

2. Des plateformes comme Moodle propose depuis quelques années la sauvegarde, mais à l'époque de nos travaux, une telle fonctionnalité était quasi inexistante ou ultra-basique

généralement de profiter d'une plateforme plus efficace ou plus adaptée au type d'enseignement appliqué. La difficulté pour un enseignant à exprimer les concepts sous-jacents à un dispositif pédagogique entraîne alors pour lui une difficulté à les transférer conceptuellement d'une plateforme à une autre. Si on peut penser qu'aujourd'hui Moodle fait autorité dans le milieu universitaire français, il peut être utile de se rappeler qu'avant que celui se répande, des plateformes maisons avaient été créées (à Lille, au Mans, à Chamberry) et que l'utilisation de celles-ci perdurent. De plus, force est de constater que Moodle ne fait pas tant autorité qu'il n'y paraît et que nombreux sont les enseignants utilisant des plateformes plus basiques (ex : wikis, blogs, Google Drive). Nous avions d'ailleurs déjà vu la préférence des enseignants à utiliser ce type de plateformes - plus simples - lors du projet PCDAI (décrit plus loin et au coeur de la première expérimentation).

3.1.2 Déployeur générique : lien entre modèles et plateformes

La réutilisation, le partage et l'alternance technologique sont trois préoccupations qui nous ont naturellement conduits à adopter une démarche IDM de type MDA (verticale), démarche assez "expérimentée" à l'époque. Nous avons outillé cette démarche comme l'illustre la figure ci-dessous, outillage que nous avons testé sur les plateformes Claroline (Peter et al. 2007), Ganesha(Caron et al. 2005), Moodle (Caron et al. 2007d;c, Drira et al. 2012) et WikiMST(Caron et al. 2007a). La chaîne complète est illustrée ci-dessous.

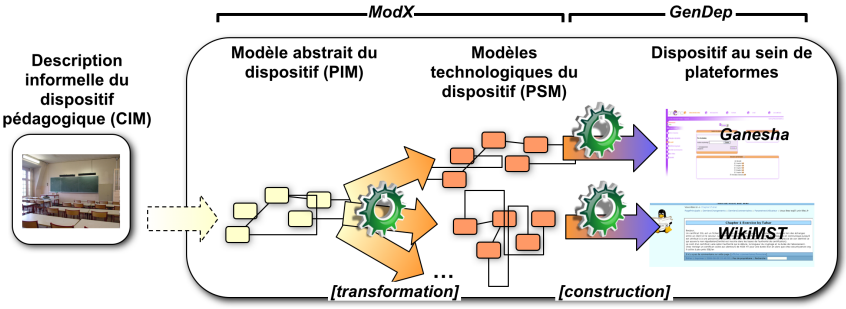


FIGURE 3.1 – *Démarche adoptée et son outillage*

La modélisation abstraite et concrète est réalisée grâce à ModX. Par contre, le déploiement d'un dispositif pédagogique sur une plateforme se fait via GenDep, un déployeur générique qu'a développé Pierre-André pendant sa thèse. Son utilisation est assez singulière et propre aux plateformes pédagogiques. En effet, lorsqu'il faut déployer un dispositif pédagogique, il y a deux différences importantes avec la génération de code, qui est l'opération équivalente et habituelle du processus MDA :

1. L'enseignant ne veut pas générer un ou des fichiers, il veut que son dispositif modélisé (PSM) soit construit directement sur la plateforme qu'il veut utiliser pour son cours. Donc au lieu de générer du texte, la chaine IDM doit communiquer avec cette plateforme pour construire dynamiquement le dispositif sur celle-ci. Par exemple,

quand il y a un espace de documents dans le modèle, l'outillage va demander à la plateforme de créer un espace de documents avec le même nom, et les mêmes documents. D'ailleurs, il y a tout un contenu multimédia à gérer dans cette construction.

2. On ne construit pas un dispositif n'importe où. Il faut le contextualiser : quel est le login/mot de passe à utiliser pour se connecter et construire le dispositif. Y a-t-il un endroit en particulier où mettre le dispositif (année universitaire par exemple) ? Ensuite il faut lier les différents rôles et participants abstraits avec des utilisateurs de la plateforme. **C'est la principale particularité de cette démarche : le déployeur doit se renseigner auprès de la plateforme pour connaitre l'environnement et demander à l'enseignant de faire le mapping entre son dispositif et l'environnement qu'il cible.**

Ces différences entre construction dynamique et génération d'un fichier peuvent sembler anodines voire purement techniques mais elles sont pourtant fondamentales d'un point de vue de l'utilisabilité et, comme on va le voir, elles vont avoir un impact sur l'utilisabilité du mapping vertical.

La figure ci-dessous montre les nécessaires informations de connection pour le déploiement d'un dispositif avec GenDep.

FIGURE 3.2 – *Déploiement : informations de connection nécessaires*

La figure du dessous montre cette fois que la "création d'un étudiant" sur la plateforme consiste à indiquer les utilisateurs déjà inscrits sur la plateforme qui vont jouer le rôle d'étudiants dans le dispositif en cours de construction. Il est possible de créer ses utilisateurs (ou autres éléments de tout autre type) à partir d'un fichier CSV contenant toutes les informations nécessaires. Dans le cas des étudiants, GenDep créera les utilisateurs ET les affectera comme étudiants dans le dispositif.

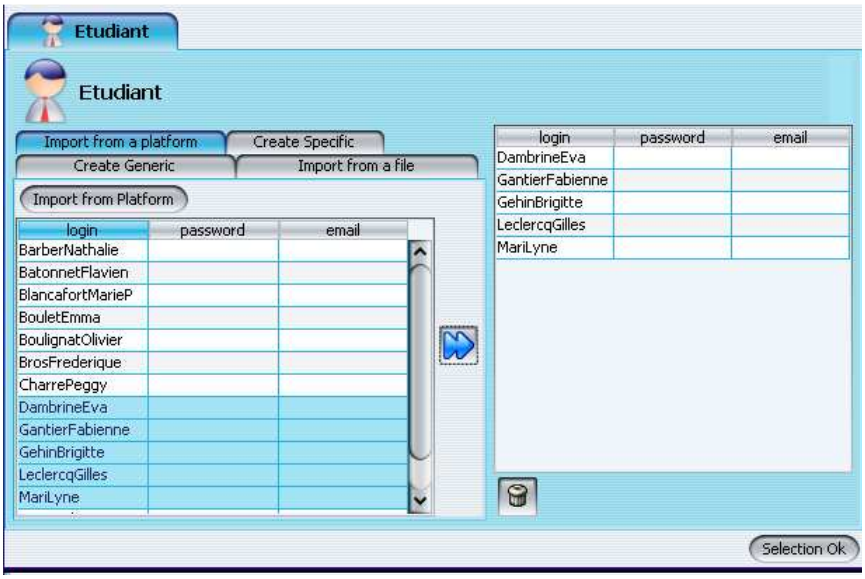


FIGURE 3.3 – *Déploiement : mapping entre le dispositif et l'existant*

Pour le déploiement, GenDep lit un modèle (au format de ModX) et construit les formulaires d'interaction à partir de celui-ci. Pour cela, il prend d'abord en paramètre le méta-modèle (aussi au format ModX) sous-jacent. Ce méta-modèle correspond au modèle de fonctionnement de la plateforme (PDM : Platform Definition Model)
Les communications avec la plateforme se font via des services Web. Comme pour un middleware comme Corba, GenDep pré-mâche le travail : à partir d'un PDM, il génère automatiquement la surcouche SOAP (client en Java pour GenDep, serveur en PHP pour les plateformes). Le travail restant consiste principalement à incorporer la partie SOAP PHP dans la plateforme en implémentant le comportement pour chaque opération du service Web généré. Le principe de mapping méta-modèle -> SOAP est simple : un méta-modèle engendre un service Web, et chaque concept/association une série d'opérations CRUD.

3.1.3 Une expérience autour du mapping vertical : le projet PCDAI

Le projet *Pratique Collective Distribuée d'Apprentissage sur Internet* - PCDAI [3] nous a permis de disposer d'un contexte idéal pour tester une démarche de type MDA. Un des objectifs de ce projet était de définir un dispositif pédagogique pour que les étudiants en stage puissent - à distance - être accompagnés et s'entraider, notamment pour la rédaction de leur mémoire professionnel. Cet objectif avait pour but d'expérimenter/favoriser des formes d'apprentissage plus actives sur Internet. Pour cela, un ensemble d'enseignants et de chercheurs en Sciences de l'Éducation se sont concertés au travers d'un certain nombre de sessions de travail. Le dispositif qui en a résulté s'est appelé *Explorateur d'Actions Personnelles et Collectives* (EAPC).

3. PCDAI, S., "Symposium Environnements numériques en formation professionnelle, 8 articles n° 314 à 318 et 372 à 376" congrés AREF 2007, http ://www.congresintaref.org/, (Strasbourg, 2007).

La chaine IDM mise en place

L'EAPC est basé sur les concepts de membres, de groupes et de balises : la collaboration se déroule dans un espace de travail balisé. Les balises permettent par exemple d'expliciter des pratiques. Comme les informations relatives au stage doivent être associées à une ou plusieurs balises, l'espace informationnel a ainsi plus de relief et permet à un groupe de s'y retrouver facilement dans l'espace de chacun de ses membres. Le méta-modèle suivant spécifie ces concepts.

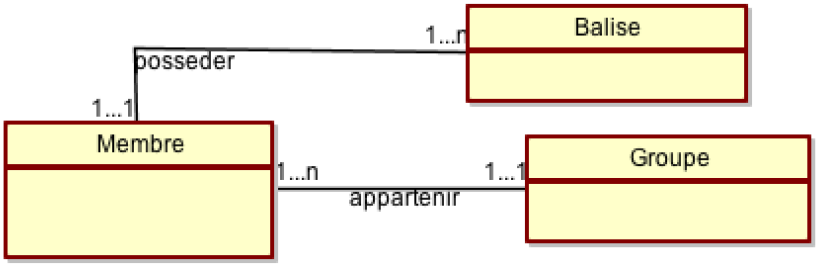


FIGURE 3.4 – *Métamodèle abstrait : le dispositif pédagogique EAPC*

Le modèle suivant montre le dispositif EAPC appliqué à l'accompagnement des stagiaires (où les étudiants coopèrent). Les balises sont représentées par des drapeaux, les membres par des rectangles arrondis à bord bleu et les groupes par des rectangles verts.

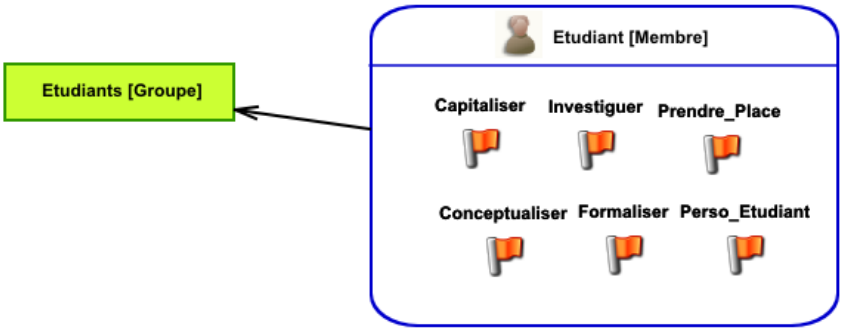


FIGURE 3.5 – *Le dispositif EAPC pour les étudiants en stage*

Comme on peut le constater, ce dispositif est abstrait : il ne contient aucune considération technologique. Il n'est donc pas attaché à une plateforme particulière et peut être projeté vers n'importe laquelle (dans les limites d'une certaine faisabilité).
Le groupe d'enseignants et de chercheurs ont choisi comme première plateforme de projection l'application Web WikiniMST [4]. Ils le trouvaient assez proches conceptuellement de l'EAPC et ne présentait pas une trop grande complexité (pour la compréhension). Toutefois, si on regarde le métamodèle lié au fonctionnement de cette plateforme, la proximité semble toute relative.

4. `http://recitmst.qc.ca/wikinimst/WikiNiMST`

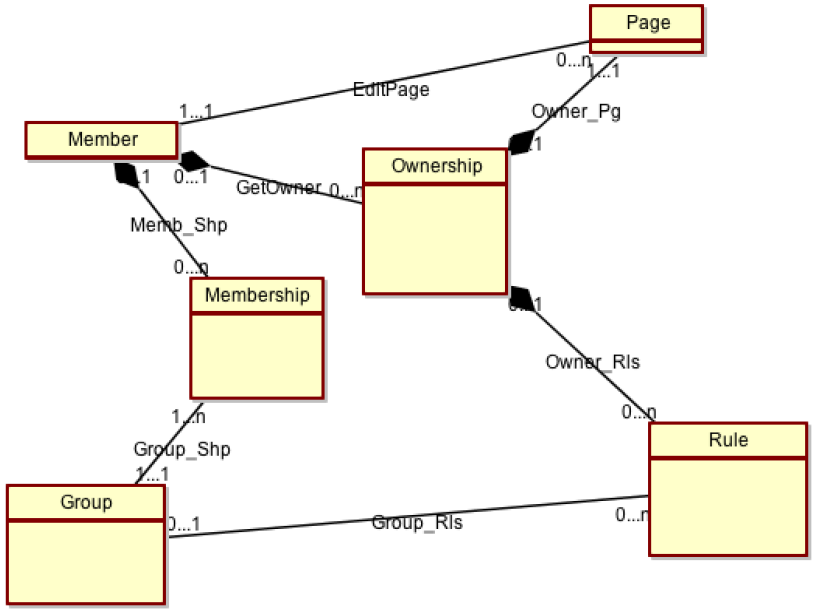


FIGURE 3.6 – *Concepts sous-jacents à la plateforme WikiniMST*

On y retrouve bien les concepts de Membre et de Groupe mais les balises y sont remplacées par des pages et l'association Groupe-Membre est surtout plus complexe. L'association Membre-Page est même pilotable par des règles d'accès assez personnalisables. Ces différences sont toutefois pertinentes car elles vont permettre d'adapter des pages Wiki à des EAPC. En effet, un membre a ses propres balises pour y *mettre ses informations* et si tous les membres de son groupe peuvent *enrichir* ces informations, ils ne peuvent en ajouter de nouvelles (les non-membres ne peuvent que consulter). *Mettre des informations* correspond à écrire dans la page et *enrichir* correspond à commenter une page.
Des règles de transformation de modèles EPAC vers WikiniMST ont été écrites dans ModX pour traduire un dispositif EPAC en sa réalisation sur le wiki. La surcouche SOAP pour la plateforme a été générée par GenDep et Pierre-André l'a intégrée dans la plateforme en implémentant les différentes opérations.
Les règles de transformation sont illustrées au travers du modèle "généralisé" suivant où *bE* renvoie à une balise du modèle source, *mE* un membre et *gE* un groupe.

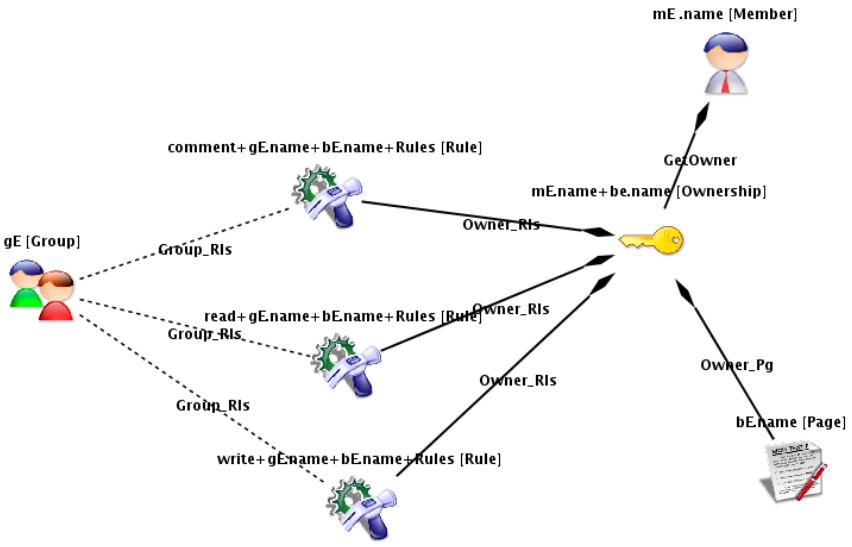


FIGURE 3.7 – *Modèle généralisé d'un dispositif EAPC sur le WikiniMST*

Les retours

L'équipe pédagogique a donc utilisé la chaine IDM précédemment décrite. Si le modèle EAPC pour les stages a été défini de manière collaborative, les déploiements ont été réalisés par plus petits groupes ou individuellement avec la présence de Pierre-André.
Le premier constat est que le processus de projection technologique et de déploiement sur un Wiki "vierge" s'est passé sans ombrage. Les enseignants ont pu accéder à des pages wiki correspondant au dispositif qu'ils avaient précédemment spécifié pour une liste d'étudiants qu'ils avaient passée sous forme de fichier CSV à GenDep lors du déploiement.
Le deuxième constat est que ce processus était beaucoup plus problématique pour une projection vers un Wiki où le(s) enseignant(s) et surtout où des étudiants s'étaient déjà inscrit et disposaient déjà de leurs propres pages. Dans cette situation, les enseignants ont souhaité vérifier et donc piloter le déploiement au travers de GenDep. Le problème est que GenDep présentait les pages existantes du Wiki et indiquait les règles d'accès selon la terminologie Wiki : les enseignants n'arrivaient pas à appréhender finement les propositions de déploiement de GenDep et trouvaient cette situation complètement inconfortable [5].

5. Le confort est une des 3 propriétés fondamentales de l'utilisabilité (les 2 autres étant la rapidité et le risque d'erreurs)

Au final, l'essai ne fut pas concluant et la séparation abstrait-concret a montré une limite (qui semble évidente) : les éléments additionnels - résultants d'une projection - présentent un risque de confusion car, de notre avis, leurs concepts ne sont pas forcément connus par le lecteur / déployeur et leur liens avec leur "origine abstraite" n'est pas immédiate. Quand bien même les concepts et les liens avec le modèle abstrait seraient connus, la manipulation du modèle concret nécessite un ensemble d'opérations mentales important.

Évolution de la chaine IDM

Pour régler le problème de décalage conceptuel, nous avons adopté - en accord avec l'équipe pédagogique - un langage de modélisation unique résultant de la fusion des deux précédents : au méta-modèle EAPC sont ajoutées les règles d'accès propres à WikiniMST. Cette fusion a été opérée manuellement par Pierre-André lors d'une session de travail avec l'équipe pédagogique. Il en ressort deux choses :

1. Quand les deux méta-modèles ont un concept équivalent, c'est le nom de celui du WikiniMST qui est choisi

2. La présence de concepts propres au Wiki ne concerne que les droits d'accès aux pages. Les autres concepts ou propriétés ne sont pas gardés car ils n'interviennent pas dans le déploiement. C'est particulièrement pregnant pour le concept *Ownership* du méta-modèle WikiniMST dont la puissance d'expression n'est pas utile pour la problématique de l'EAPC et peut donc être réduite à une simple série de propriétés dans les concept et association *Spaces* et *OwnerShip*.

Le méta-modèle final est donc la traduction conceptuelle EAPC vers la plateforme WikiniMST. Le diagramme suivant montre le méta-modèle résultant.

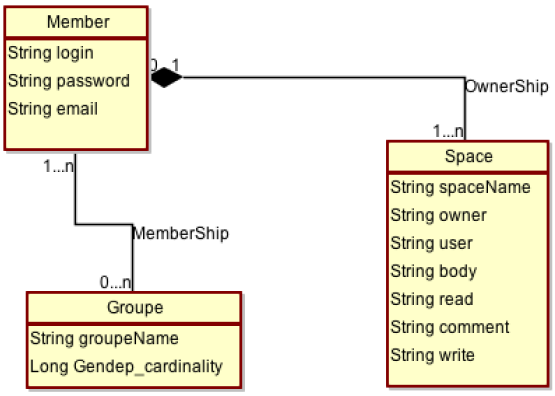


FIGURE 3.8 – *Méta-modèle issue de la fusion EAPC et WikiniMST*

Avec cette traduction, les enseignants peuvent travailler dans un espace de solution proche de leur espace de problème avec les ajouts conceptuels

technologiques juste utiles pour appréhender les différentes subtilités du déploiement vers WikiniMST. C'est l'objectif de cette proposition : fournir une solution au problème lié à la complexité cognitive de la précédente chaine IDM (compréhension, mémorisation et traduction mentale).
Une fois les modifications expliquées aux enseignants, ceux-ci ont trouvé cette solution plus confortable. Lors de la phase de modélisation, ils ont été plus dans le détail pour la spécification de leur dispositif. Et lors du déploiement, ils ont pu vérifier aisément l'application des règles d'accès aux pages existantes.
Le modèle lié au suivi de stage a été spécifié (en groupe avec la présence de Pierre-André) comme suit dans le nouveau méta-modèle.

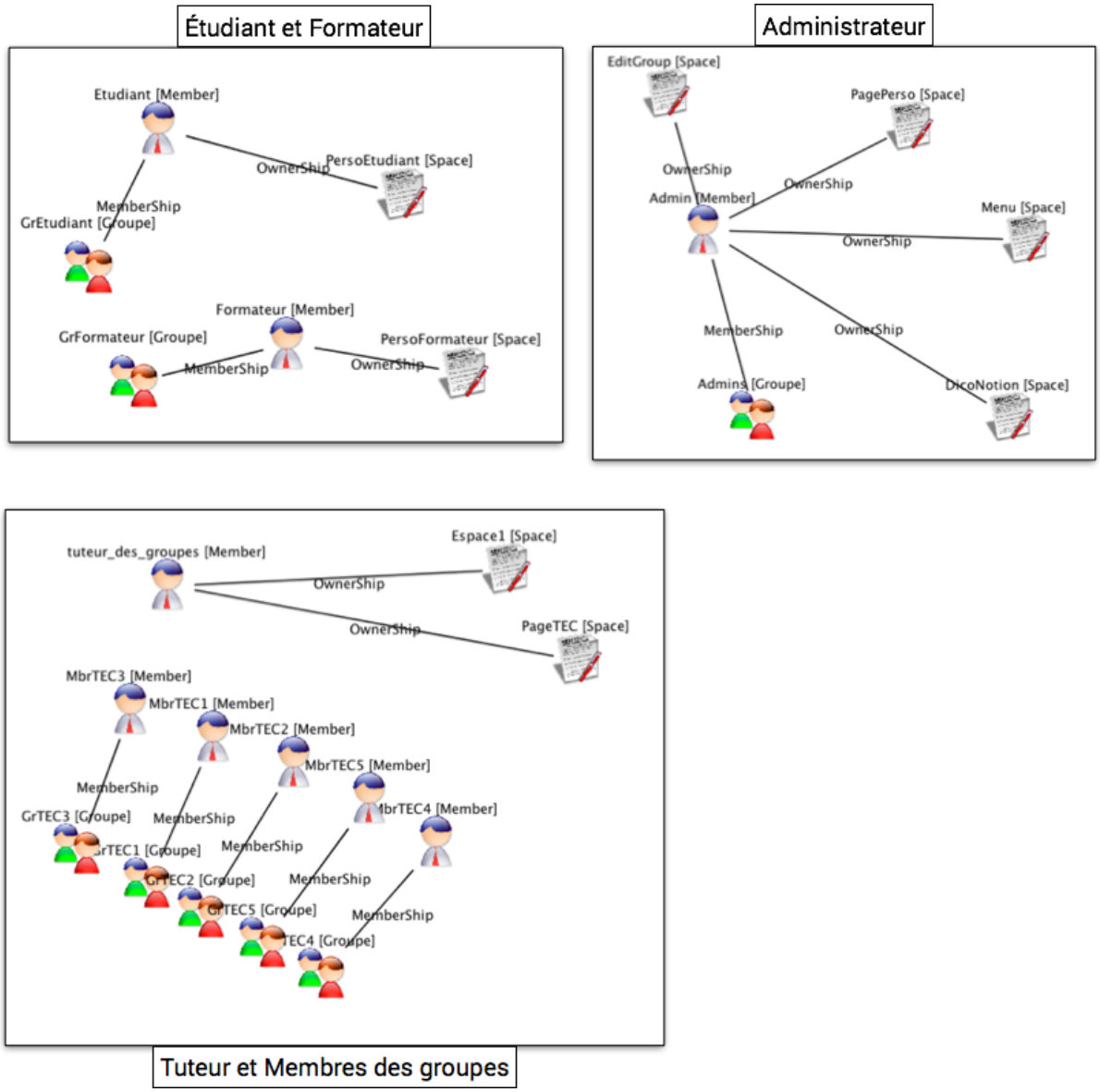


FIGURE 3.9 – *Le modèle complet du dispositif selon les concepts du méta-modèle fusionné*

Malheureusement, ce type de solution est à contre-sens de l'IDM car elle n'est pas faite pour séparer les préoccupations (ici : métier & technique)

et va se limiter à une seule plateforme, et donc ne plus atteindre l'objectif du multi-plateforme de l'IDM.

> La contribution forte de ce travail réside dans la mise en avant de la difficulté pour l'humain à rester maître des opérations qui s'effectuent lors d'un processus MDA : en particulier le mapping entre deux méta-modèles.
> Pour cette raison, nous avons adopté une autre démarche où le mapping automatique va laisser place à un mapping plus manuel afin que celui-ci devienne plus appréhendable et **traçable**.

3.1.4 Les bonnes pratiques : point fondamental du mapping vertical

En observant l'utilisation des plateformes pédagogiques, nous avons remarqué que les enseignants avaient des connaissances sur la plateforme qu'ils utilisent et que la traduction de leur dispositif abstrait vers celle-ci pouvait se faire "à la main". L'avantage par rapport à notre démarche type MDA est que le dispositif technique généré serait beaucoup mieux maîtrisé par l'enseignant. Le premier inconvénient est, par contre, que le déploiement serait beaucoup plus long (beaucoup de clics, de chargements de page, de choses répétitives). Le second, qui est moins frappant à première vue, serait que la traduction ne profiterait pas des bonnes pratiques d'utilisation de la plateforme ou de méthodes pédagogiques répandues et/ou expérimentées par des collègues, comme par exemple celles qu'on pouvait trouver dans les règles EAPC -> WikiniMST concernant les règles d'accès aux pages Wiki. Les bonnes pratiques de codage (liées ou non à l'entreprise) étaient d'ailleurs un point qui avait été énoncé par des praticiens du logiciel lors de la journée IDM organisée en Décembre 2010 entre universitaires et industriels [6].
Les travaux de thèse de Rim Drira vont partir de ces observations et réflexions et proposer une démarche 1) où la traduction sera suffisament manuelle pour éviter le manque de traçabilité vue précédemment 2) tout en guidant l'enseignant par des bonnes pratiques (comme évoquées plus haut) : il ou elle pourra ainsi maîtriser de bout en bout la construction de son dispositif concret tout en bénéficiant des subtilités techniques et pédagogiques acquises par des experts ou ses collègues.

Contextualisation du dispositif à son environnement

La thèse de Rim Drira [7] s'inscrit dans le projet SAME2 ("e-Services, Approches Multi-Echelle, et E-formation" 2007-2010) en collaboration avec les laboratoires RIADI de Tunis et CRI de Paris I. Rim était encadrée par Mona Laroussi (RIADI) et moi, et dirigée par Henda Ben Ghezala (RIADI) et Alain Derycke. La problématique du Multi-Échelle du projet SAME 2 fait ici référence aux différents niveaux de préoccupations s'appliquant sur un dispositif pédagogique : le dispositif, le cours dans lequel il inter-

6. http ://idm2010.lifl.fr

7. Titre de la thèse : "Assistance à la modélisation et à la contextualisation de dispositifs pédagogiques complexes"

vient, la formation proposant le cours, l'institut proposant la formation, la politique générale imposée par l'université sur chacun de ces instituts... Chaque niveau a sa terminologie, ces concepts mais pourtant, toutes les "règles" doivent être traduites et intégrées dans chaque dispositif pédagogique construit par les enseignants. Les bonnes pratiques que nous avons évoquées dans le paragraphe précédent renvoient aussi aux règles issues de niveaux plus macroscopiques, c'est-à-dire décidées par exemple par le responsable de la formation ou le conseil d'un institut (ou UFR)... car ces règles viennent en partie de bonnes pratiques observées au fil des années. Un des problèmes du MDA est que, selon la norme, les bonnes pratiques de projection sont intégrées dans les règles de transformation : nous avons vu que le lecteur peut alors perdre la trace de ces éléments abstraits lors de l'analyse du dispositif concret généré. Avec une approche par aspect comme proposée par (Jezequel 2008), la contextualisation, c'est-à-dire ici un tissage de bonnes pratiques, est plus manuelle car elle peut être faite à l'aide de modèles paramétrés : la traçabilité est bien meilleure. Mais là encore, certaines contextualisations - en particulier celles technologiques - vont rester intégrées à des règles de génération et vont donc demeurer problématiques d'un point de vue de la traçabilité.
L'approche proposée par Rim va essayer de répondre à ce problème de contextualisation et de traçabilité. Les deux points principaux de son approche sont les suivants :

1. la projection technologique (mapping vertical PIM -> PSM) est définie par l'enseignant à partir de son modèle abstrait (et non pas au niveau du méta-modèle abstrait).

2. les choix de mapping sont comparés à une liste de bonnes pratiques propres au contexte de l'enseignant pour son dispositif.

Nous allons voir comme cette approche a été concrétisée.

ACoMoD : Gen-IC + Gen-Com

Rim a mis en place une chaine IDM appelée ACoMoD : Assistance for Contextualized Modeling of learning systems. Le principe général de fonctionnement est le suivant :

1. le dispositif pédagogique doit d'abord être décrit dans un modèle UML (diagramme de classes)

2. le modèle UML est templatisé (les principales propriétés deviennent des paramètres du template résultant)

3. l'enseignant choisit la plateforme sur laquelle il souhaite déployer son cours, puis indique pour chaque précédent paramètre l'élément de la plateforme qui y correspond - mapping réalisé avec Gen-COM

4. l'enseignant aura au préalable choisit les bonnes pratiques (BP) qu'il veut suivre, et tout au long de la définition du mapping, Gen-COM indiquera si ces BP sont respectées ou non.

5. Le référentiel de BP est géré par l'outil Gen-IC

Le schéma suivant montre l'architecture générale de l'outillage d'ACoMoD.

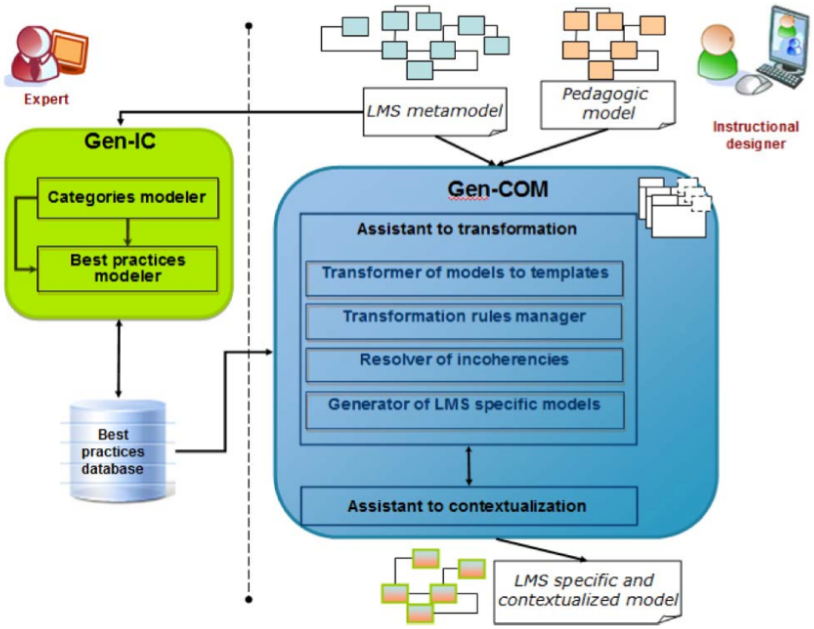


FIGURE 3.10 – *Architecture générale de la chaîne ACoMoD*

Le premier outil d'ACoMoD est **Gen-IC**. C'est un référentiel de bonnes pratiques, où chacune d'elle est associé à une suite de mots-clés ou une situation (voir l'exemple dans le tableau plus bas) qui permettra ensuite à un enseignant de trouver les BP liées à son contexte. Chaque BP est, normalement, spécifiée par un ou des experts institutionnels (enseignants ou non). Elle est d'abord spécifiée de manière générique, puis ensuite dérivée en plusieurs versions, chacune d'elles étant spécifique à une plateforme pédagogique.
La figure suivante montre un exemple de BP générique et une de ses versions spécifiques.

Examples of GBP		Examples of SBP	
GBP	**Parameters**	**Effective Values**	**SBP**
The use of a *tool* is prohibited	Tool Situation	Tool: Chat Situation: when students meet regularly	The use of chat is prohibited
The maximum number of a *tool* is *valmax*	Tool Valmax Situation	Tool: forum Valmax: 3 Situation: the use of more than three forums could create the problem of having some messages in each one or unused forums	Maximum number of forums is 3
		Tool: Wiki Valmax: 1 Situation:…	Maximum number of wikis is 1

FIGURE 3.11 – *Bonne Pratique Générique (GBP) et Spécifique (SPB)*

Le second outil est **Gen-COM**. Il permet de faire le mapping entre le modèle pédagogique et la plateforme pédagogique choisie. D'un point de vue interactif, comme le montre l'écran ci-dessous, l'outil propose d'un côté tous les éléments du modèle pédagogique et de l'autre tous les concepts de la plateforme.

FIGURE 3.12 – *Mapping vertical selon Gen-COM*

À la différence de l'approche de Pierre-André, le modèle pédagogique est spécifié sous forme de diagramme de classe UML. Le mapping est techniquement réalisé à l'aide de templates UML. En effet, lorsqu'un modèle de ce type va être sélectionné par l'enseignant avec Gen-COM, ce dernier va le transformer en un template UML où les paramètres de celui-ci seront tous les éléments (et les liens) du modèle et leurs propriétés. La zone *a* de la figure suivante montre cette templatisation sur l'exemple d'un modèle relatif à un cours sur Java (appelé *Temp_Projet*).

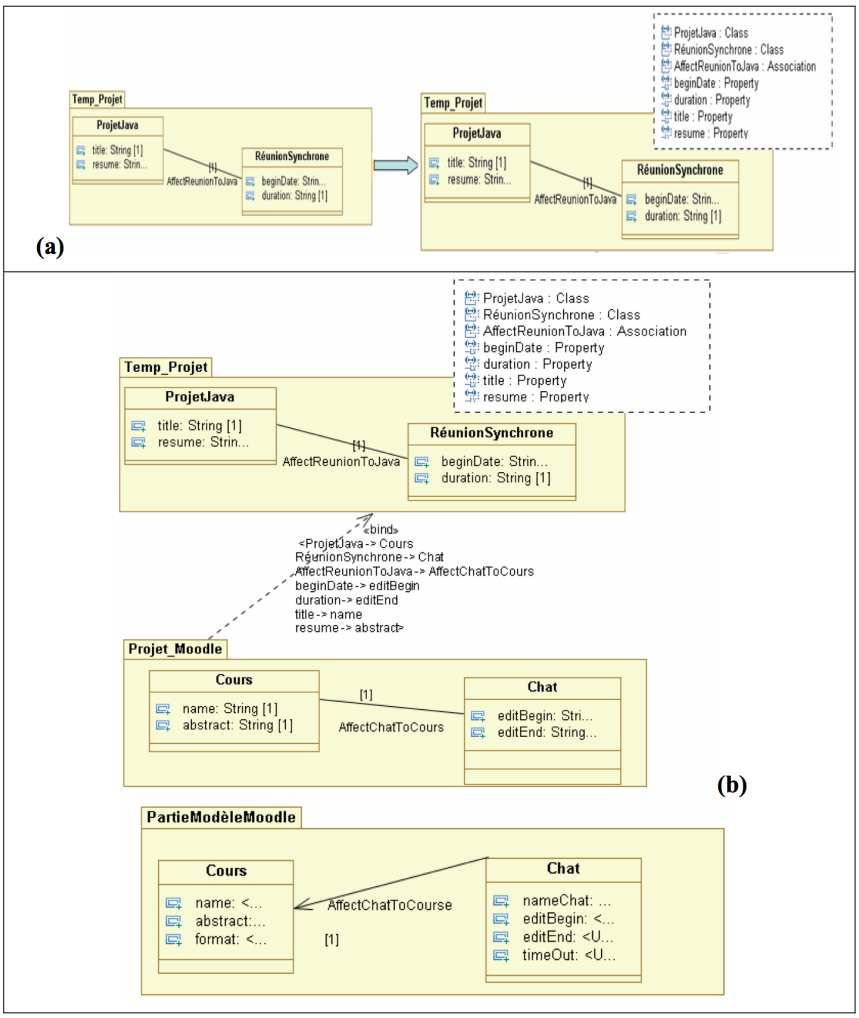


FIGURE 3.13 – *Utilisation de template UML pour le mapping*

On peut voir dans la zone *b* de cette figure que les concepts du méta-modèle de la plateforme (ici Moodle, sur la figure *PartieModèleModlle*) sont utilisés pour "binder" une construction (*Projet_Moodle*) au précédent template. Cette construction est produite par Gen-COM et sera la modèle concret final.

> *La principale limitation de l'outillage d'ACoMoD est de ne pas utiliser un langage de modélisation spécifique au métier de l'enseignant : même s'ils sont simples, les diagrammes de classes sont terminologiquement éloignés des préoccupations de l'enseignant. Toutefois, un intérêt important lié à l'utilisation de templates était de pouvoir définir des dispositifs pédagogiques abstraits "génériques" et de pouvoir les "instancier" ensuite au contexte pédagogique de l'enseignant : par exemple un dispositif EAPC générique pouvait être bindé (partiellement ou complètement) afin de donner un dispositif EAPC pour le suivi de stage. Il faut toutefois fournir un outil simplifiant la manipulation de templates et du binding pour les rendre accessible aux enseignants. Rim avait commencé à concevoir cet outil (appelé Gen-PTE) mais le développement n'a pas complètement abouti. Les possi-*

bilités d'expression et de réutilisation qu'offraient les templates n'ont donc pas pu être expérimentées.

Lors de la définition du mapping, Gen-COM va vérifier que les choix effectués sont cohérents d'un point de vue structurel (essentiellement sur les types de valeurs et les cardinalités). Avant de spécifier le mapping, l'enseignant indique son contexte à Gen-COM (suite de mots + choix de la plateforme). Des bonnes pratiques seront présentées et pourront être sélectionnées par l'enseignant. Gen-COM vérifie ensuite si ces bonnes pratiques sont respectées. Si ce n'est pas le cas, il indique les éléments qui sont en conflit avec la bonne pratique (dans l'onglet *Choices Summary* en bas à gauche). Il affiche aussi le message d'assistance lié à la BP pour que l'enseignant comprennent le principe de celle-ci (onglet *Contextualization sheet*).

Expérimentation avec le projet Mepulco et la plateforme Moodle

Afin de vérifier la pertinence de l'approche ACoMoD et d'évaluer son utilité et utilisabilité, nous avons choisi un cas similaire au projet PCDAI : la méthode Mepulco. Celle-ci a été définie à l'IUT de Calais et a aussi été utilisée à Polytech'Lille. Elle est destinée à superviser des projets étudiants réalisés en groupe. Mepulco est un scénario d'apprentissage actif et a deux principaux objectifs. Le premier est d'aider des groupes d'étudiants à réussir à créer un produit commun en respectant une date d'échéance et de développer différentes compétences individuelles telle que l'analyse, la synthèse et l'argumentation. Le second est d'aider les tuteurs à superviser les différentes étapes d'un projet et de fournir une évaluation justifiée du produit final.
Mepulco repose sur des réunions régulières qui permettent de vérifier la bonne progression et la viabilité du projet quand à la date d'échéance. Tous les principes de la méthode Mepulco sont rédigés dans deux documents : un destiné aux tuteurs et un pour les étudiants. On y trouve l'organisation des différentes réunions, leur ordre du jour, l'affectation des rôles et les lignes directrices pour l'écriture du rapport final et la présentation finale. Il y a aussi des documents type à utiliser pour l'écriture de spécifications ou de rapports de progression. Chaque projet basé sur Mepulco fonctionne au travers d'activités distantes supportées par un site web, un blog ou un LMS. Le modèle Mepulco et le méta-modèle de Moodle sont décrits dans (Drira et al. 2012).
Le but de l'expérimentation - pour les enseignants/tuteurs - était de les aider à déployer un dispositif Mepulco pour chaque projet supervisé sur la plateforme Moodle. Pour nous, comme nous l'avons dit, l'objectif était de vérifier la pertinence d'ACoMoD. Trois critères d'évaluation ont été définis pour cela :

1. l'utilité de l'assistance pendant la projection technologique d'un dispositif pédagogique
2. l'utilité des recommendations issues des BP
3. l'utilisabilité de Gen-COM.

44 concepteurs (enseignants ou ingénieurs pédagogiques) ont participé à l'expérimentation. Ces concepteurs étaient impliqués dans des formations de l'IUT de Calais (encadré par B. Warin), de Polytech'Lille (F. Hoogstoel) ou dans le master e-Services International (M. Laroussi). La moitié avait de bonnes connaissances en UML et l'autre non, comme le montre le tableau ci-dessous.

Number of Participants	UML Modeling	UML Templates	Model Transformation
20	No	No	No
6	Excellent	Excellent	Excellent
4	Excellent	Excellent	Good
2	Good	Good	Average
2	Good	Good	Good
10	Average	No	No

FIGURE 3.14 – *Compétences des participants liées à UML et la transformation de modèles*

Six questions, relatives aux 3 précédents critères, ont été posées aux participants après que ceux-ci aient utilisé ACoMoD pour l'adaptation de leur dispositif Mepulco à Moodle et son déploiement dans le cadre de leur suivi de projet. La figure ci-dessous donne un aperçu générale des réponses.

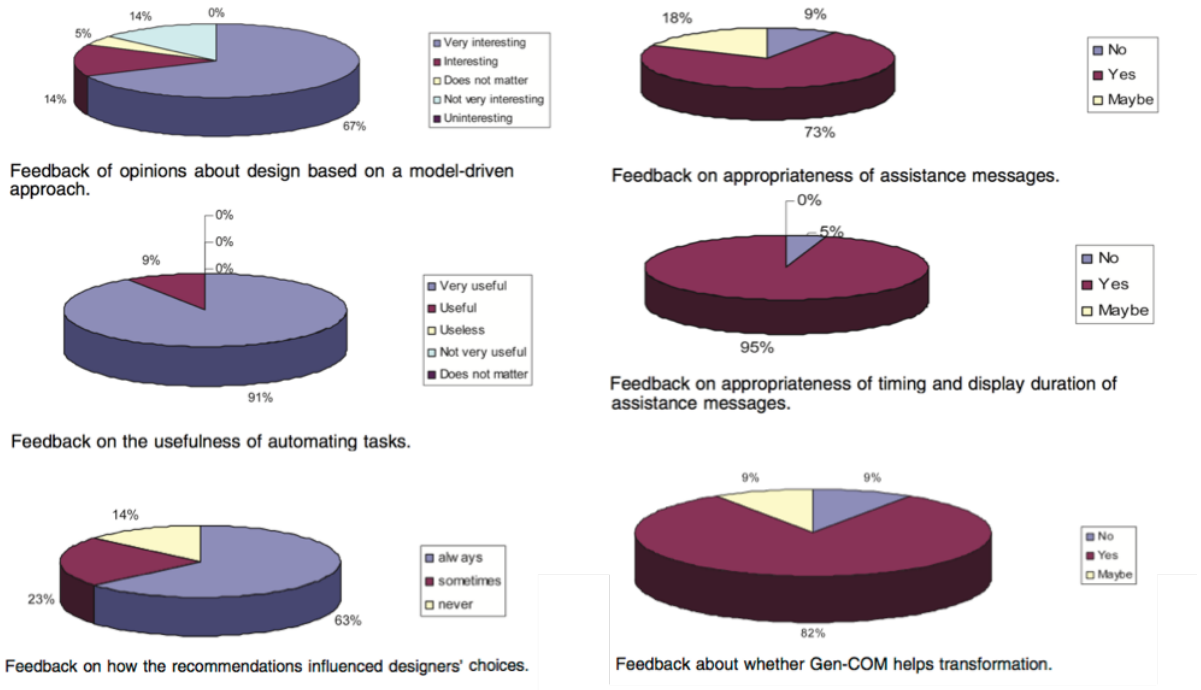


FIGURE 3.15 – *Retours des concepteurs sur l'utilisation d'ACoMoD*

Comme on peut le voir, les retours sont bons, notamment sur le fait qu'une approche type IDM est très intéressante, que les recommendations (es-

sentiellement les BP) ont majoritairement influencé la conception et que Gen-COM a été très utile pendant la transformation.
Si les retours sur l'expérimentation de Pierre-André ne sont pas aussi quantifiés, et qu'aucune expérimentation n'a été faite pour comparer les deux approches, les retours sur l'approche ACoMoD ont toutefois montré que l'approche (et l'outillage associé) était accessible et à aucun moment les participants n'ont indiqué une perte de traçabilité (ce qui est logique vu la démarche). En plus de conserver une indépendance vis-à-vis de la plateforme (à l'opposé de la deuxième version de l'approche de Pierre-André), les possibilités de tissage de préoccupations/BP offertes par les templates rapprochent ACoMoD des approches présentées dans le chapitre 1 où l'IDM est exploitée au maximum.

La conclusion générale que nous pouvons faire ici sur le mapping vertical du point de vue de l'humain est que :

1. la traçabilité entre modèle source et modèle final est fondamentale et son absence peut être rédhibitoire. Une possibilité est de demander à "l'utilisateur" de définir lui-même les liens de mapping.
2. les bonnes pratiques constituent un élément clé de l'IDM et l'intérêt du mapping vertical ne vient peut-être pas de leur automatisation mais, à notre avis, de sa capacité à les intégrer et à les indiquer/transmettre aux utilisateurs.

3.2 Intégration d'éléments technologiques

Quelle que soit la perspective de modélisation, l'intégration d'éléments technologiques peut être problématique. En effet, la technologie évolue et, c'est ce que nous souhaitons, elle évolue vite. Si les concepts et associations d'une perspective de modélisation intègre des éléments liés à une technologie, les évolutions de celle-ci vont confronter la précédente perspective à un problème de versionning. Si la perspective impactée est en bout de chaine, on reste dans le monde idéal du MDA : la technologie subit de nombreux changements, spécifiez votre application à un niveau plus abstrait et vous ne serez pas touché(e) par ces évolutions. Si par contre, la perspective impactée est plus haut dans la chaine, les changements risquent d'avoir des répercussions sur toute la chaine (les règles de transformation ou de génération). Le ROI (Retour Sur Investissement) d'une telle chaine par rapport à du "codage" classique risque d'être revu à la baisse.
Cette intégration d'une dimension technologique est au coeur des travaux de Nadia Elouali. Nous allons voir que le choix des mécanismes qu'elle a mis en place a été essentiellement guidé par l'aspect humain, c'est-à-dire pour que l'utilisabilité de ces mécanismes soit élevée.

La thèse de Nadia s'est déroulée d'Octobre 2011 à Octobre 2014[8]. La thèse a été encadrée par Jean-Claude Tarby et dirigée par José Rouillard. Nadia ayant grandement participé au projet MOANO, j'ai eu souvent l'occasion de collaborer avec elle, notamment sur les aspects IDM. Le travail présentée ici est le fruit d'une de ces collaborations.

3.2.1 Contraintes liées aux interactions multimodales au sein de la conception d'applications mobiles

La problématique qui nous intéresse ici est la conception d'interactions multimodales pour les applications mobiles. Le contexte technologique associé a déjà été abordé dans le chapitre 2. Voici les contraintes qu'exerce ce contexte sur l'activité de conception :

Hétérogénéité de plateformes Il y a 3 plateformes importantes concernant les applications mobiles : iOS, Android et HTML5. Mais on peut aussi évoquer Windows Phone, Blackberry OS ou Firefox OS. Cette diversité est une réalité couteuse pour les commanditaires d'applications mobiles car il faut souvent financer le développement d'une application sur les 3 premières plateformes citées (la 3ème étant généralement destinée aux autres OS).

Fragmentation Pour Android, le nombre de versions de l'OS utilisées par les utilisateurs, les nombreuses tailles d'écran existantes, les différentes sensibilités des capteurs constituent ce qui est généralement appelé "fragmentation". Et implémenter une application sur Android implique de gérer ces différences[9]. Cette fragmentation est bien moindre sur iOS, mais, aux dires des professionnels, elles commencent à apparaître.

Évolution technologique La liste des capteurs embarquées dans les téléphones évoluent depuis le début (2007 - iPhone 1). Entre la version 2.1 d'Android et la version 4.0, 5 nouveaux capteurs ont été intégrés au SDK (gravité, accélération linéaire, rotation vectorielle, température, humidité). Les périphériques connectés (montres, bracelets, lunettes, accessoires sportifs) ne vont pas diminuer cette tendance.

Évolution des pratiques Les types d'interaction permis par les capteurs embarqués ne sont pas stabilisés comme pour la souris. Non seulement de nouvelles apparaissent (comme le mouvement avec le téléphone imitant 2 coups d'accélérateur pour le moto X pour lancer l'appareil photo) mais la signification d'une interaction n'est pas commune (secouer veut dire *Défaire* sur iOS 5, *ajouter un commentaire* sur Google Maps ou *envoyer un rapport de bug* avec Facebook).

Événements à deux niveaux Les utilisateurs voient les types d'interaction à un niveau plus abstrait que ce que proposent les SDK. Par exemple, "secouer" a longtemps été absent des SDKs : le développeur devait

8. Titre de la thèse : "Approche à base de modèles pour la construction d'applications mobiles multimodales"

9. Nous avons discuté avec des "agences mobiles" où le coût du développement demandé au commanditaire variait selon la couverture des versions OS supportées.

implémenter une analyse des changements d'accélération (x, y et z) pour détecter la secousse. Il en est de même pour un balayage de la main devant le capteur de proximité ou de lumière (dont les valeurs envoyées lors d'un événement sont respectivement la distance ou la luminosité en lumens).

L'intérêt d'une chaine IDM provient de l'hétérogénéité des plateformes mais aussi de la complexité grandissante des applications mobiles et donc de l'intérêt de la séparation des préoccupations : une perspective de modélisation pour les interactions multimodales nous semble donc très pertinente.
Voyons comment les précédentes contraintes vont guider les choix pour la définition du méta-modèle pour cette perspective.

3.2.2 Choix conceptuels

Niveau d'abstraction. La perspective de modélisation doit permettre au concepteur de se focaliser sur les interactions multimodales proposées aux utilisateurs. Il nous semble donc ici logique de proposer des interactions de haut niveau telles que "secouer", "scanner un QRCode" plutôt "changement d'accélération avec les valeurs x=3, y = 3 et z =0" : si le concepteur doit effectuer de nombreuses opérations mentales pour pouvoir déduire les événements proposés aux utilisateurs ("opérations mentales difficiles" des dimensions cognitives), les inférences nécessaires impactent négativement l'ergonomie du méta-modèle.
Toutefois, qu'en est-il d'avoir une abstraction à deux niveaux, comme le proposent les outils SMUIML (Dumas et al. 2014) et DynaMo (Avouac et al. 2011), c'est-à-dire avoir des concepts qui permettent de définir des événements de haut niveau qui peuvent ainsi être utilisés pour spécifier les interactions d'une application ? Étant donné le problème de fragmentation, la modélisation "réaliste" d'un événement de haut niveau peut devenir très complexe : il s'agira d'un ensemble d'algorithmes où chacun sera propre à un groupe de versions/téléphones ou alors un algorithme général avec différentes déclinaisons selon la version/téléphone. Les modèles et surtout les diagrammes sont surtout pertinents pour des groupes d'éléments interconnectés (Larkin et Simon 1987), ce qui n'est pas le cas ici. Ensuite cette modélisation va surtout être du code modélisé c'est-à-dire que le méta-modèle fournira peu d'abstraction. Il n'y a donc pas d'intérêt et il vaut mieux rester sur du code car l'interface (celle proposée par les IDE : multiples raccourcis claviers pour le copier/coller, la navigation, l'auto-complétion) est bien meilleure que les interfaces des éditeurs de modèles.
Intégration des aspects technologiques
Nous avons donc fait le choix d'avoir des concepts relatifs à un haut niveau d'abstraction. En d'autres termes, le concept d'*Événement* ne comportera pas de propriété liée à des valeurs de bas-niveau mais plutôt des propriétés pour indiquer le type de modalité (pas forcément le capteur), et un simple label pour désigner l'événement. Cette affirmation sur le label est loin d'être anodine ou évidente. Dans l'approche MARIA (Manca et al. 2013), les événements de sortie (les widgets) font partie intégrale du méta-modèle : chaque widget est un concept qui hérite indirectement

ou non du concept d'*Interaction*. Les événements d'entrée (comme ceux liés à une zone d'édition) sont directement associés à leur widget, et donc "codés en dur" dans le méta-modèle. Les précédentes contraintes liées à l'évolution ne permettent pas d'adopter une telle démarche : son manque de souplesse est incompatible avec les changements fréquents actuels des smartphones et de leurs capteurs.
Une autre solution consiste à spécifier de manière abstraite les interactions sans indiquer précisément le geste à effectuer, ou le widget à "toucher", le raffinement technologique étant ensuite effectué via un langage de modélisation dédié. Cette solution, proposée dans l'approche UsiXML [10], ne nous semble pas réaliste. Tout d'abord, la question de l'intégration se pose aussi pour le méta-modèle dit "technologique". Ensuite raisonner sur des choses non tangibles n'est pas une pratique que nous avons observé. En effet, depuis Avril 2014, Jean-Claude Tarby et moi effectuons une enquête auprès de professionnels du développement mobile. A l'heure de l'écriture de ce manuscrit, nous avons interviewé plus d'une quarantaine de professionnels (France, USA, Belgique, Chine). Et à chaque fois, le récit sur la manière dont il conçoive les interfaces et interactions indique que les discussions se font sur du "concret" : il est important de savoir ce que propose une plateforme pour connaitre les possibilités d'interaction. L'exemple de Square Inc [11], basée à San Francisco, en est un parfait exemple (les dessins d'interface de départ sont accrochés à l'accueil) : sans les possibilités d'interactions offertes par les smartphones, l'application de paiement Square n'aurait jamais vu le jour.

Au final on retiendra que le langage de modélisation doit proposer toutes les possibilités d'interaction afin d'exploiter pleinement la ou les plateformes de développement mobile mais le niveau d'abstraction doit être suffisamment élevé pour permettre au concepteur de se focaliser uniquement sur les interactions et non sur les soucis de fragmentation ou d'hétérogénéité.

3.2.3 Modèles paramétrables comme niveau intermédiaire

L'idée principale de notre approche est d'utiliser une bibliothèque d'événements d'entrée/sortie et d'actions (ex : changement d'image pour le widget image, redimensionnement...). Nous allons voir ici comment se fait l'intégration de cette bibliothèque dans le langage de modélisation - de haut niveau d'abstraction - et comment se fait la génération de code correspondante.

Intégration conceptuelle

Le méta-modèle pour le langage de modélisation M4L défini par Nadia - pour les interactions multimodales - est le suivant :

10. USer Interface eXtensible Markup Language
http://www.usixml.org/
11. https://squareup.com/

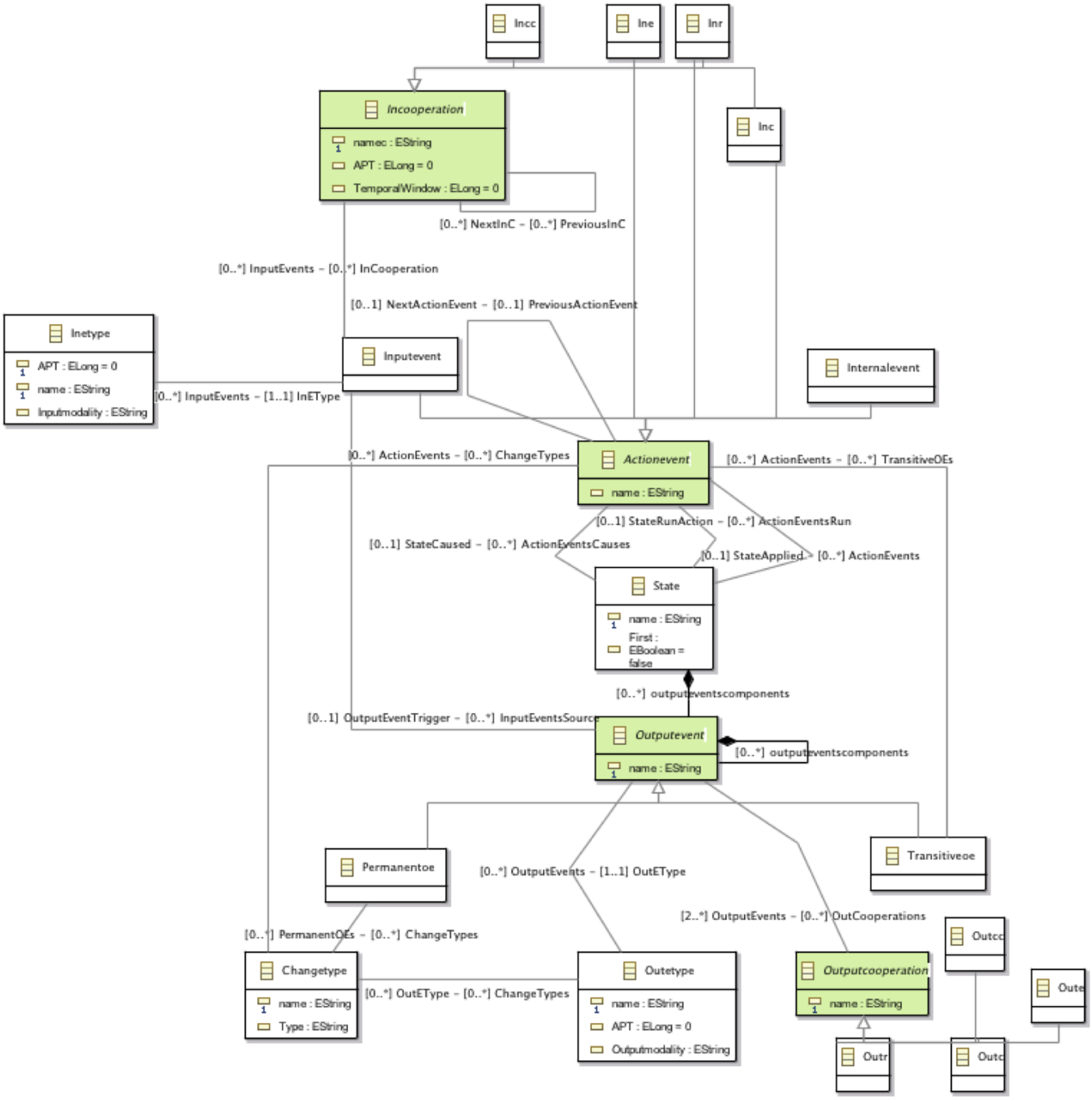


FIGURE 3.16 – *Le méta-modèle M4L*

Ce méta-modèle est dérivé de SMUIML de manière à s'adapter au mieux aux contraintes vues précédemment. Ce qui nous intéresse ici sont les concepts *Inputevent*, *Inetype*, *Outputevent*, *Outetype*. Les concepts **event* permettent d'indiquer dans un modèle les événements que gérera l'application mobile. Le nom est la seule propriété et permet d'apporter un peu de sens à l'événement (ex : "touch final" au lieu de "touch"). Les concepts

**type* indique la modalité utilisée (ex : tactile), et le nom de l'interaction (ex : touch).
Afin de proposer une bibliothèque prêt à l'emploi, nous avons adopté le principe des templates UML où le "bind" a laissé la place à un opérateur de copie mais où il faut toujours paramétrer le template. La figure suivante montre ce type de template pour l'événement "touch".

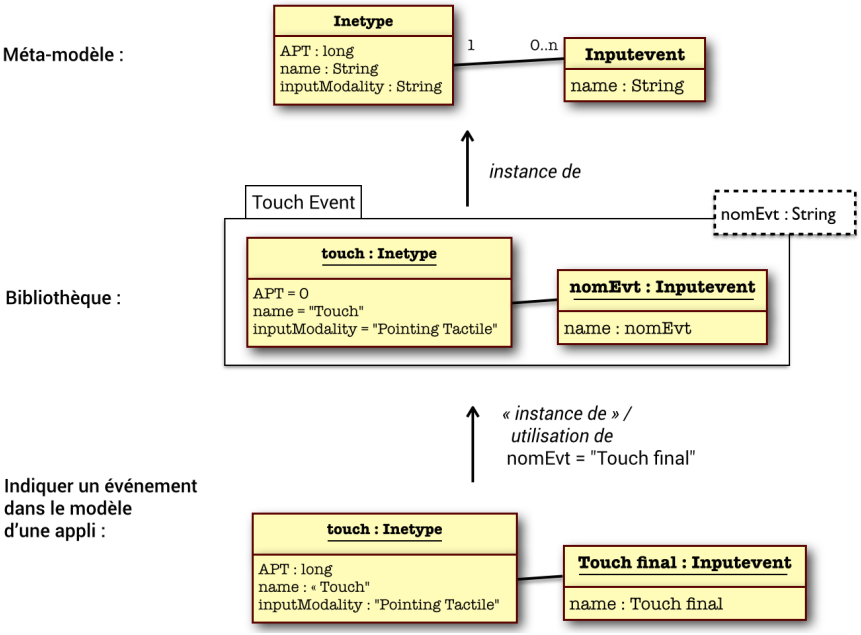


FIGURE 3.17 – *Mécanisme pour la bibliothèque*

La notion de template peut sembler ici artificielle (ce n'est qu'une fonction de clônage où certaines propriétés sont vides) mais elle est transverse à deux autres aspects : la représentation graphique et la génération de code :

Représentation graphique Dans l'éditeur de modèles [12], les événements sont affichés mais pas leur type. Toutefois, quand l'éditeur détecte l'utilisation du template *Touch Event*, l'événement est affiché selon l'icône qui est associé au template. Les différents événements disponibles sont proposés dans la palette d'outils de l'éditeur. Un *drag-n-drop* "instancie" le template et demande un nom pour le paramétrer.

Génération de code Une bonne partie des règles de génération fonctionne par détection de templates. Quand un template est détecté, du code spécifique va être généré. Afin d'alléger les règles de génération, le code généré utilise une librairie de programmation [13].

Le schéma suivant montre les aspects associés à l'événement Touch.

12. implémenté par Nadia avec Obeo Designer `http://www.obeodesigner.com/`
13. implémentée par Nadia et utilisant d'autres librairies

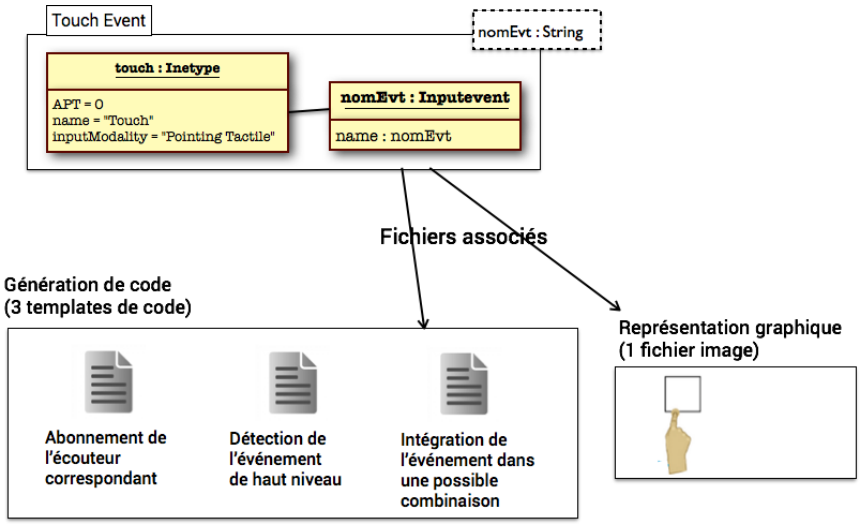


FIGURE 3.18 – *Fichiers de génération de code et d'image associés à un remplace (Ex : le touch)*

Comme on le voit sur cette figure, le principe de template est transversal et important. On pourrait voir le principe de la bibliothèque se réduire à une suite de types sur lesquelles les événements doivent pointés. Mais l'utilisation explicite de template permet d'exprimer le besoin de détection de patterns d'événement, c'est-à-dire d'une référence vers un type particulier. Pour l'instant, ce principe de template n'est pas codé de manière générique. Le principe a surtout été testé de manière réaliste : une bibliothèque de plus de 80 événements (in/out) et plus d'une vingtaine de modèles d'application de tests. Une prochaine étape est de proposer un mécanisme où un expert pourra créer un type d'événement / template, lui associer un icône, et des fichiers/templates de génération. Ces templates seront chargés au démarrage de l'éditeur afin d'y être intégrés et affichés.

Éditeur et Expérimentation

Cette partie sur l'éditeur de modèles et l'expérimentation de son utilisabilité n'est pas en lien direct avec les mécanismes sous-jacents à la bibliothèque que nous venons de présenter. Toutefois, elle permet de voir si le langage de modélisation et sa bibliothèque d'événements prêts à l'emploi sont performants. Enfin, elle souligne une fois encore l'intérêt de l'utilisabilité des outils IDM.
Une idée importante de la thèse de Nadia était d'adopter une approche pragmatique et de vérifier à tout moment que chaque ajout ou modification était viable sur une série d'applications tests. Le site web de la chaîne IDM de Nadia, appelée MIMIC [14], montre cette impressionnante batterie de test (là où les autres chaînes du même type se limite à 1 ou 2 exemples) : `http://www.lifl.fr/~eloualin/tool.html`. Dans cette optique, l'éditeur a été implémenté pour être facilement utilisable et accessible.

14. MobIle MultImodality Creator

Sans rentrer dans les détails de l'outil [15], l'environnement se compose de 3 zones :

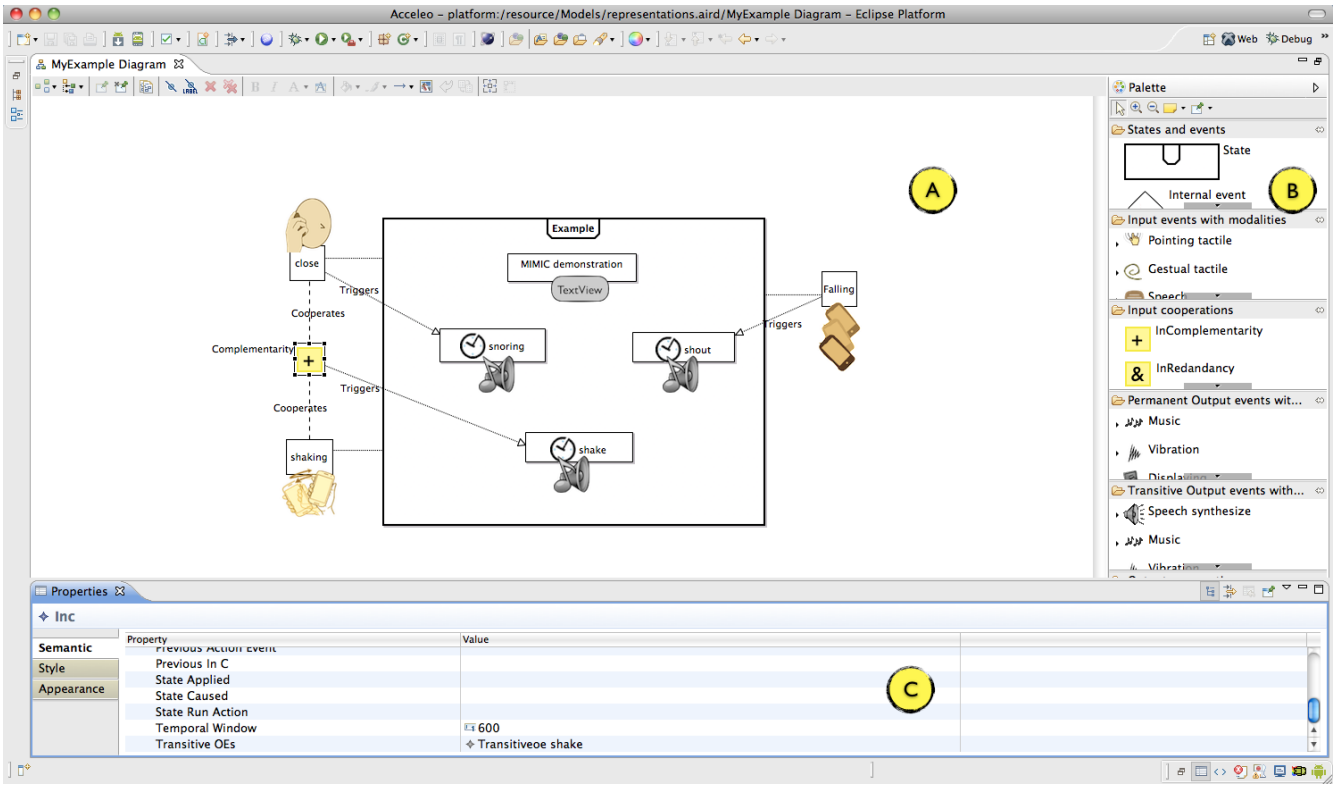


FIGURE 3.19 – *Éditeur de modèle de MIMIC*

La zone A est la feuille de création de diagrammes. La zone B correspond à la palette d'outils intégrant la bibliothèque d'événements et d'actions. La zone C permet d'éditer finement les propriétés. L'éditeur de modèles est basé sur Obeo Designer (`http://www.obeo.fr/pages/obeo-designer/fr`).

L'expérience Pour évaluer l'utilisabilité de MIMIC, une expérience a été menée auprès de 20 étudiants du master e-services. Ces étudiants étaient formés à Android et ils avaient eu un cours/TP de 4H sur l'utilisation des capteurs et leur programmation sous Android. Ils ont eu aussi 2H pour effectuer un tutorial sur MIMIC. Le principe de l'expérience est le suivant :

Concevoir un jeu multimodal Chaque étudiant va devoir coder en 2H une application (un jeu) utilisant plusieurs modalités d'interaction dont 2 combinaisons (une équivalence et une complémentarité). Une bonne partie de l'application est déjà fournie, les étudiants devant se focaliser essentiellement sur la partie "interaction".

Le sujet est progressif Un ordre est donné pour les fonctionnalités à implémenter. Mais il n'est pas obligatoire de le suivre.

15. ils se trouvent sur le site web très détaillé

Avec et sans MIMIC Les étudiants vont être divisés en 2 groupes de 10 personnes : 1 groupe va implémenter les interactions sans utiliser MIMIC et un 1 autre groupe avec MIMIC

Évaluation L'évaluation de l'utilisabilité portera sur la comparaison entre le nombre d'interactions implémentées avec ou sans MIMIC.

Les résultats sont intéressants en terme d'utilisabilité : utiliser MIMIC est plus efficace en terme de temps de développement, par contre en terme de confort, cela semble être l'inverse.
Voici les résultats en terme d'interactions et combinaisons d'interactions implémentées pour chacun des groupes.

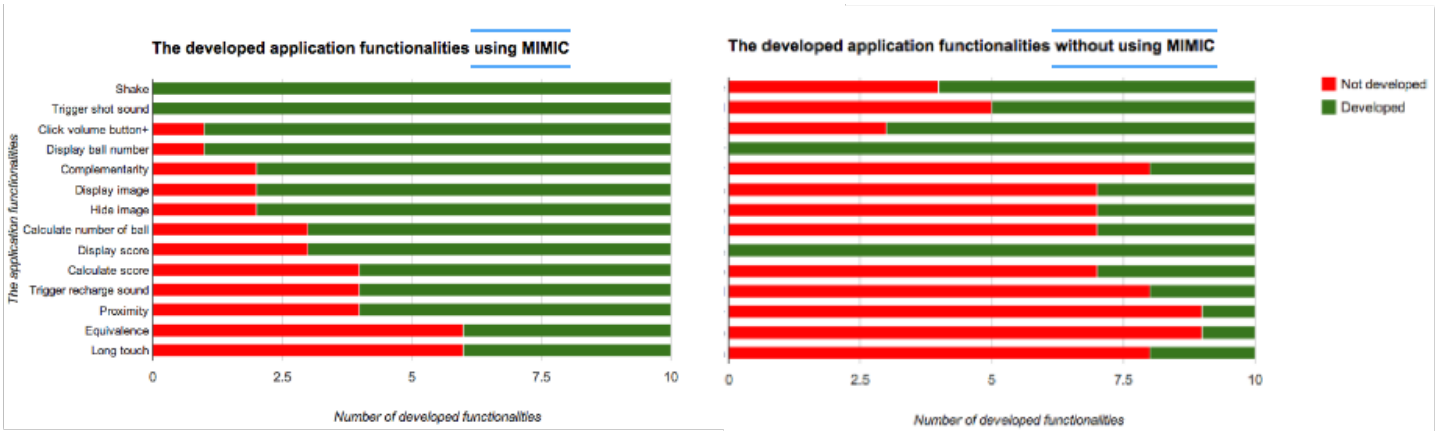


FIGURE 3.20 – *Comparaison avec et sans MIMIC*

On voit clairement que, pour chaque fonctionnalité, le nombre de d'étudiants qui arrivent à la développer est bien plus important pour le groupe utilisant MIMIC. Tout d'abord, les différences sont importantes pour les événements propres au capteurs. Ensuite, les étudiants utilisant MIMIC suivent plus facilement l'ordre conseillé pour implémenter les fonctionnalités et vont donc plus loin. Enfin, cela ne se voit pas forcément sur les 2 graphiques, mais deux étudiants de ce groupe ont réussi à implémenter totalement l'application alors qu'aucun étudiant n'a réussi cela dans le groupe n'utilisant pas MIMIC. L'approche MIMIC permet d'être plus performant pour implémenter cette application mobile multimodale.
Sur les graphiques suivants, on peut avoir un retour des utilisateurs sur MIMIC.

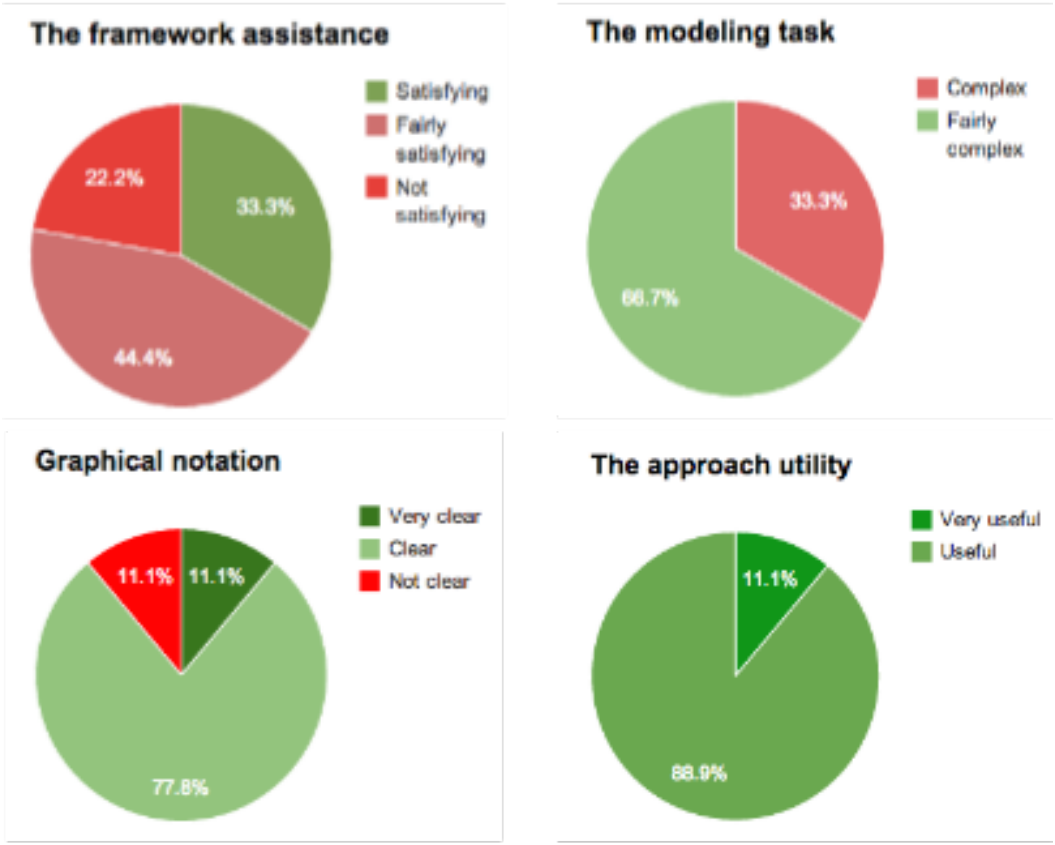


FIGURE 3.21 – *Retour utilisabilité MIMIC*

Si l'utilité de MIMIC semble ne pas avoir de détracteur et si sa notation visuelle est à près de 90% jugé (très) clair, il n'en est pas de même pour l'activité de modélisation et l'assistance proposée par MIMIC. En effet, pour une première expérience, un bon tiers des participants ont trouvé la modélisation selon MIMIC complexe. On peut penser qu'avec un peu de pratique cela changera, mais il reste qu'une "courbe d'apprentissage" est bien présente. Est-ce dû à l'ergonomie de l'outil ou aux concepts utilisés ? Les questions n'ont pas été posées dans ce sens et nous n'avons pas de réponses à cela. Enfin, l'assistance proposée par l'outil ne semble pas suffisante pour tous les participants. L'assistance consiste en un ensemble de vérifications et une documentation. De par nos observations, la documentation n'a pratiquement jamais été consultée par les participants. Comme les vérifications ne couvrent pas tous les aspects, on peut comprendre que 1) l'assistance n'ai pas été suffisante 2) une documentation ne semble pas être ici une bonne méthode pour guider la modélisation. Un contexte professionnel (et non d'expérimentation) aurait peut-être donné l'impulsion nécessaire pour la consulter.

Conclusion

Les travaux de Nadia sont réellement pertinents pour l'étude générale de l'utilisabilité des approches de type IDM. Tout d'abord, Nadia a créé un langage de modélisation en respectant les contraintes d'un contexte tech-

nologique et économique complexe (hétérogénéité des plateformes et fragmentation du marché), ce qui n'avait pas été le cas des travaux précédents / actuels sur une perspective de modélisation dédiée aux IMM. Ensuite, elle a pris en compte le point de vue du concepteur en veillant à ne pas fournir un langage qui puisse tout faire et qui aurait été trop complexe. Au lieu de cela, elle a conçu le langage pour qu'il se focalise sur l'essentiel, c'est-à-dire gérer les événements de haut niveau. C'est, pour le concepteur, l'intérêt principal de cette perspective de modélisation. Toutefois, ce type d'événements impliquent l'intégration de considérations technologiques. L'OMG proposait, avec le MDA, la séparation PIM-PSM pour maîtriser les changements constants des technologies d'implémentation logicielle. Malheureusement, ce mapping vertical ne pouvait s'appliquer ici au vu des besoins du concepteur. Pour cette intégration technologique, Nadia a inséré un mécanisme de template à son langage qui complète ainsi pleinement sa réponse globale aux problèmes de départ.
Elle apporte un élément de réflexion sur la gestion de la dimension abstraction en montrant que dans ce contexte, elle semble être mieux gérée par un mapping horizontal comme (Jezequel 2008) que par un mapping vertical.

Conclusion du chapitre

Ce chapitre était consacré au mapping vertical en IDM avec une focalisation sur la perception de l'humain de ce mapping. Les travaux de Pierre-André ont montré que, dans le cas du e-learning, la projection technologique (le cas classique de mapping vertical) posait des problèmes de traçabilité, notamment lorsqu'il y avait déjà un existant. C'est d'ailleurs une préoccupation que nous avons souvent entendue lors de nos discussions avec des professionnels : la possible modification automatique de code produit par un développeur donne l'impression à ce dernier de ne plus maîtriser son développement.
Des travaux de Rim Drira, on retiendra comme contribution majeure le fait de préférer, pour le mapping vertical, le guidage par bonnes pratiques plutôt que des règles de transformation "automatiques". La traçabilité est ainsi améliorée et, en GL, on peut imaginer que ce type de solution serait bien mieux perçu et n'engendrerait pas un sentiment de dépossession. Il permettrait aussi de transmettre aux "juniors" l'expérience acquise par l'entreprise en terme de développement.
Enfin, dans les cas où la projection technologique ne peut pas être faite en bout de chaîne, nous avons vu au travers des travaux de Nadia, que l'utilisation de modèles paramétrables est une solution aux problèmes de versionning dus aux évolutions fréquentes des plateformes visées. Là encore, la perception humaine est fondamentale : la définition de concepts doit *d'abord* être guidée par celle-ci (afin d'abstraire les détails technologiques non pertinents et ralentissant le raisonnement via un modèle) et *ensuite* respecter les considérations logicielles ou techniques.

Ergonomie de la modélisation logicielle

4

Sommaire

Un langage de modélisation est destiné à définir des modèles. En IDM, pour que les modèles soient productifs, cette action est souvent faite en IDM à travers un support logiciel. Ceci implique une interaction Humain - Système avec ce support. Cette interactivité est guidée par la sémantique du langage, en d'autres termes, le langage répond à la question "quelles sont les actions possibles pour éditer un modèle ?". Mais il doit répondre aussi à d'autres formes d'interaction que les simples opérations d'édition :

- Par quoi commencer ? Quelles sont les étapes dans l'activité de modélisation avec ce langage ? Y a-t-il une méthode associée au langage ?

- Quand le concepteur va sélectionner un bout de diagrammes pour faire un copier/coller, opération très fréquente en informatique, quel va être le résultat de cette opération ? Le clone va-t-il être le reflet de l'original ?
- La modélisation intervient souvent dans un projet où plusieurs personnes sont impliquées. Il est donc important que l'outil d'édition reflète cette collaboration pour que chaque membre de l'équipe en ait conscience quand il modélise, afin d'articuler efficacement son travail avec le reste de l'équipe. Quelles sont les informations importantes à afficher sur cette collaboration ? Quelle sont les actions effectuées sur le modèle qui sont susceptibles d'avoir le plus d'impact sur le travail des "autres" ?

Dans l'absolu, si toute la sémantique d'un langage de modélisation était exprimée dans sa syntaxe abstraite, il serait possible de répondre à ces questions, tout comme il serait possible de déduire la syntaxe concrète la plus appropriée à un contexte particulier. Comment exprimer la sémantique pour les parties ? Peut-être comme on le fait avec la syntaxe concrète : avec un langage dédié à cela. Les trois questions précédentes ont été traitées par des travaux auxquels j'ai participé (1 collaboration et 2 encadrements de thèse). L'objectif de ces travaux est de cerner les besoins ergonomiques de la modélisation et de définir le plus formellement possible les concepts sous-jacents dans ce qu'on pourrait appeler une syntaxe d'interactions. Même si cette explicitation est déjà un apport important à l'IDM, une telle syntaxe serait extrêmement utile pour les méta-éditeurs comme Obeo Designer, MetaEdit+ ou Modx afin d'améliorer l'expérience utilisateur des éditeurs qu'ils génèrent.
Dans ce chapitre, nous allons voir les réponses que nous avons tenté d'apporter aux questions concernant l'assistance à la modélisation au travers de méthodologies associées, la construction de modèles grâce au copier/coller et la modélisation collaboration. Les deux derniers travaux sont des thèses en cours et il n'y a pas encore de "solution apportée". Mais il y a néanmoins une contribution à chaque fois : une meilleure cartographie du problème au travers d'une étude bibliographique et d'une étude de terrain rigoureuse.

4.1 Assister l'expert dans la création de modèles

Comme nous l'avons dit en 1.4.2, l'assistance à la modélisation est une nécessité mais peu de travaux scientifiques s'y intéressent. Les aspects ergonomiques semblent encore une fois être secondaires en IDM. Une solution d'assistance peut être de proposer une "simple" documentation. Toutefois, nous avons vu dans l'expérimentation de Nadia autour de MIMIC que les étudiants n'allaient pas naturellement consulter la documentation fournie. D'ailleurs de nombreux éditeurs de logiciel proposent, en plus de la documentation, un tutorial intégré à leur logiciel qui permet non seulement de faire un tour d'horizon des fonctionnalités mais aussi d'indiquer les principes de base de l'utilisation de celles-ci. Par exemple, le trombone du logiciel Word pouvait guider l'utilisateur lors des premières étapes de création d'un document.
Pour la modélisation, il est bien sûr important pour le concepteur de comprendre les concepts, les associations et la notation visuelle d'un langage de modélisation qu'il souhaite utiliser. Mais il est aussi utile pour lui/elle de connaître une ou plusieurs méthodes associées pour l'utiliser "convenablement" afin de spécifier un modèle. UML n'est pas associée, de manière standard, à une ou plusieurs méthodes. C'est une force car il peut ainsi convenir à une audience très large. C'est une faiblesse car 1) l'utilisation d'UML exige de s'acheter, au minimum, un livre et d'adopter la démarche définie/adoptée par les auteurs 2) la définition sémantique d'UML en pâtit, ce qui alimente ces discussions interminables sur la nature des concepts UML.
On imagine aisément le gain pour un débutant d'avoir dans son éditeur UML, un assistant (débrayable) qui lui demande quelle méthode il souhaite adopter et qui lui indique étape par étape ce qu'il y a à faire, tout en simplifiant l'interface pour ne laisser que ce qui est utile à l'étape en cours. C'est ce type d'approche que nous avons adopté, formalisé et implémenté dans ModX. Nous avons appelé cette approche les *Processus Incrémental de Modélisation* (PIM). Pour expérimenter cette approche, nous avons défini un PIM pour le langage IMS-LD (e-Learning) et pour un langage de modélisation dédié au design pattern PAC (IHM).

4.1.1 Les processus de modélisation incrémentale

Nous définissons un PIM comme la décomposition du processus de modélisation en une suite d'étapes séquentielles. Le concepteur suit les étapes les unes à la suite des autres, mais peut à tout moment revenir en arrière. La figure suivante montre la structure d'un PIM. Ce dernier est appelé parfois une méthodologie : dans ModX, ce terme nous a semblé plus compréhensible que "PIM".

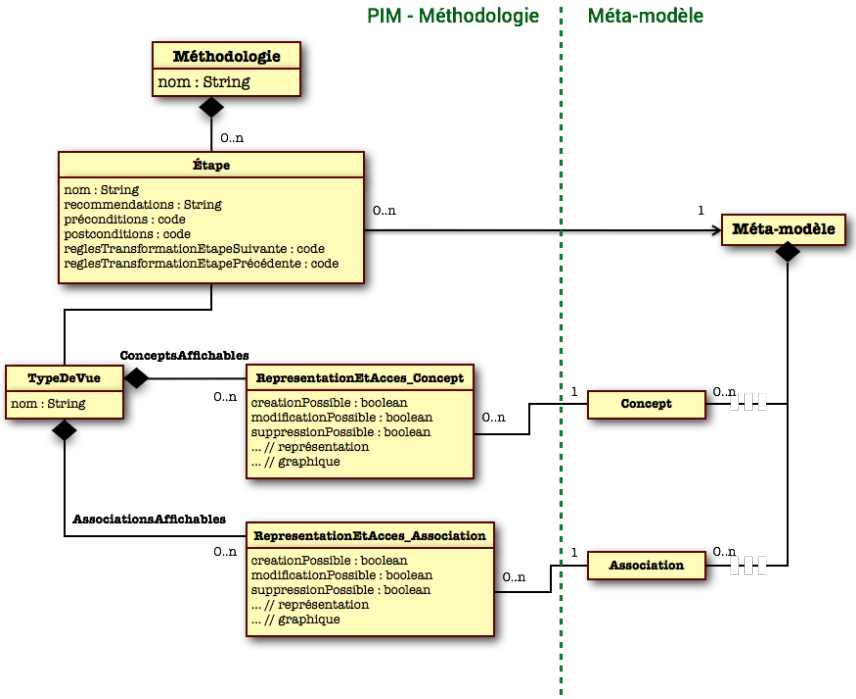


FIGURE 4.1 – *Structure d'une méthodologie*

Une méthodologie est composée d'étapes. Chaque étape est d'abord définie par un sous-ensemble du méta-modèle et une syntaxe graphique associée à ce sous-ensemble : en réduisant ainsi les concepts et associations utilisables, on permet au concepteur de se focaliser sur un aspect du langage de modélisation et donc sur un aspect de son modèle. Cette réduction "visuelle et interactive" se fait au travers d'un type de vue. Grâce à ce mécanisme, la focalisation est de deux formes :

1. soit le type de vue ne concerne qu'un sous-ensemble du méta-modèle
2. soit le type de vue concerne tout le méta-modèle mais seuls les éléments issus d'un sous-ensemble de celui-ci sont modifiables (ou "créables").

La figure ci-dessous illustre ces deux types de d'approches. Dans la méthodologie 1, le concepteur ne peut créer, modifier et voir que des instances de A et B lors de la première étape. Dans la deuxième étape, le concept B laisse sa place au concept C. De l'étape 1, le concepteur ne voit que les instances de A et il va rajouter 3 instances de C. À l'étape 3, tous les concepts sont visibles, le concepteur voit apparaître la totalité de son modèle.
Dans la méthodologie 2, on a presque la même chose, sauf à l'étape 2. En effet, celle-ci permet de continuer à voir les instances de B, mais elles ne sont plus modifiables. Et le concepteur ne peut plus créer des instances de B. De plus, pour indiquer que B est moins important dans cette étape, la taille de ces instances a été modifiée (modification de la syntaxe concrète).

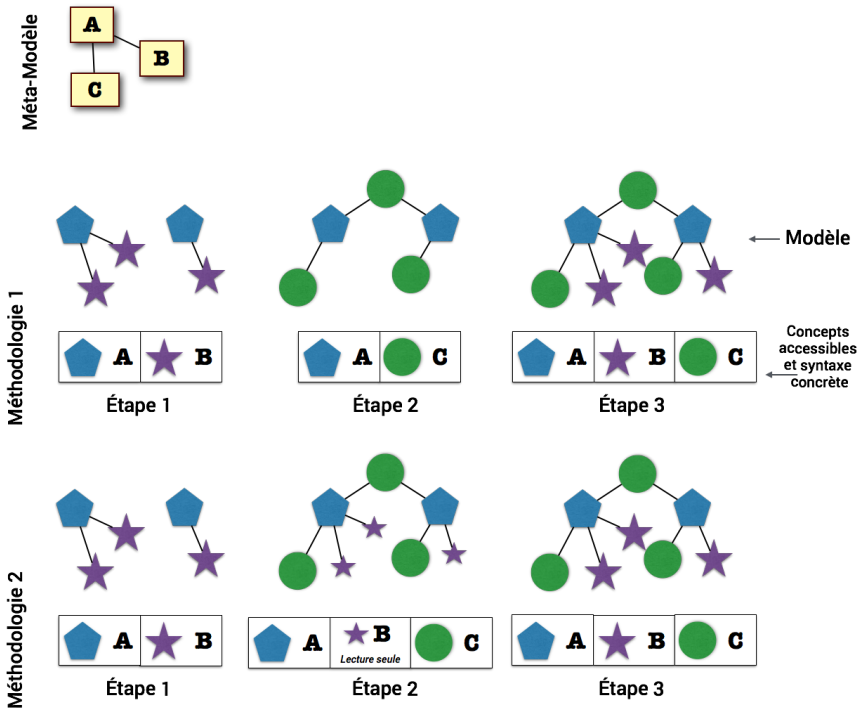


FIGURE 4.2 – *Exemples de méthodologies*

Les pré et post conditions d'une étape permettent de vérifier des conditions plus fines que la présence ou non d'instances de concepts ou d'associations. Sur l'exemple, on pourrait imaginer une pré-condition pour l'étape 1, afin qu'il y ait au moins une instance de A quand cette étape est terminée (ou quand l'étape 2 commence, soit en post-condition). Pour l'étape 3, une condition pourrait être que toutes les instances de A soit reliées entre elles par des instances de C (comme pour un workflow et des documents échangés entre chaque activité).
Une étape peut aussi posséder un ensemble de règles de transformation (de modèle) dont la source serait le modèle à l'issue de l'étape et dont le receveur ou consommateur serait l'étape suivante. Une étape peut aussi être associée à des règles de transformation qui vont être exécutées lorsque le concepteur était sur l'étape suivante et qu'il décide de revenir en arrière. Sur la figure suivante, on voit un ensemble de règles d'avancement, associé à l'étape n, qui produit un chemin entre les instances de A via des instances de C (voir précédent exemple) selon un ordre aléatoire. L'ensemble de règles de recul est ici à l'opposé car elle vise à effacer tout les liens "C" entre instances de A.

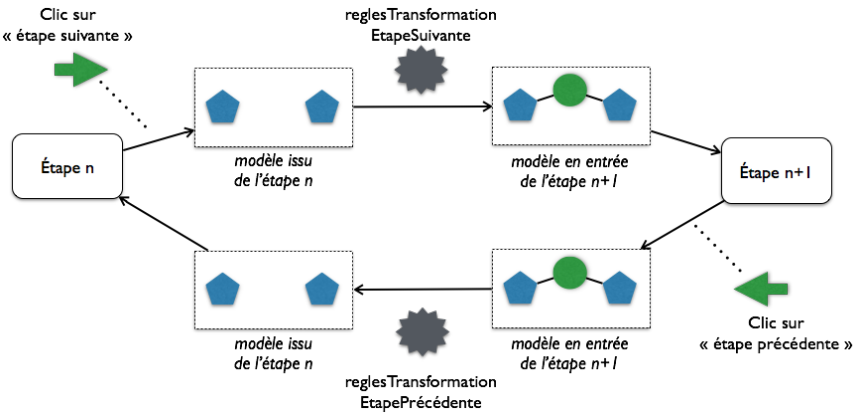


FIGURE 4.3 – *Règles de transformation entre étapes*

Ce travail commencé en 2005 a été le fruit de ma première collaboration avec Raphaël Marvie et Jean-Claude Tarby. Nous avons longtemps hésité à utiliser un système de workflow avec la possibilité de brancher une étape à plusieurs autres possibles et avec, bien sûr, des branchements qui pourraient être conditionnés. Mais nous avons décidé que l'apport d'une telle complexité serait trop faible par rapport aux efforts supplémentaires pour spécifier les méthodologies de ce type : en effet, notre expérience en modélisation (GL, IHM, e-learning) nous confortait dans l'idée que 1) une méthodologie linéaire sera généralement largement suffisante 2) et bien plus simple à définir qu'un processus de type workflow.
Nous avons appliqué les PIM sur IMS-LD et sur le patron de conception PAC (Présentation - Abstraction - Contrôleur (Coutaz 1987)).

4.1.2 En e-learning

L'adéquation d'IMS-LD aux PIM était double :

1. IMS-LD est un langage très complexe
2. Les spécifications d'IMS-LD proposent déjà une séquence linéaire d'étapes à suivre pour concevoir un scénario pédagogique.

Comme on le voit ci-dessous, le méta-modèle d'IMS-LD est complexe.

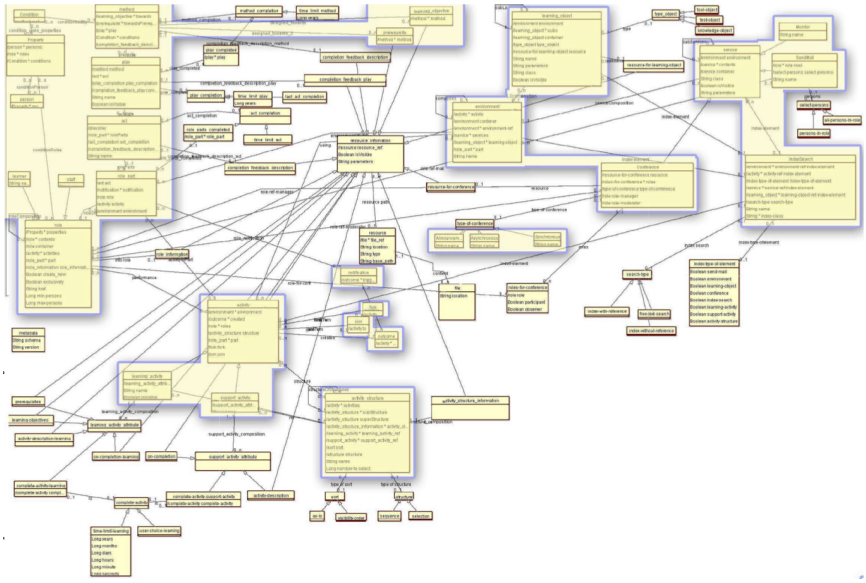


FIGURE 4.4 – *Aperçu de la complexité du méta-modèle IMS-LD*

Nous ne rentrons pas dans le détail du méta-modèle ici. La figure montre surtout le grand nombre de concepts d'IMS-LS : il en résultat les concepts utilisés se limitent à ceux bleutés sur la figure. La métaphore théâtrale à la base d'LD, son faible niveau d'adoption, les nombreuses polémiques autour du nombre important de concepts nous permettent de qualifier IMS-LD de complexe.
En revanche, IMS-LD fournit une *bonne pratique* de modélisation au travers de cette suite d'étapes.

1. *Identifying activities*
2. *Identifying roles and binding each one to activities* it plays.
3. *Defining the scheduling of activities*: sequentiality, parallelism, alternatives. The next steps consist in the translation from scheduling into activity structures (step 4), acts (step 5) and finally plays (step 6).
4. *Definition/deduction of activity structures.* An activity structure is a set of activities that a role will play (this is notified through a rolePart associated to the structure). The set composed with these activities comes from a semantic relation: in fact, these activities constitute components of an activity which is more coarse-grained. They may be played sequentially or in any order.
5. *Definition/deduction of acts.* An act is a set of activities (generally activity struc-tures) that are played by different roles. To play its next activity, each role has to wait for other actors to complete their activity. On the UML diagram, each synchronization point between different roles correpond to an act.
6. *Definition/deduction of plays*: A play is a succession of acts. These acts are exe-cuted sequentially. However, all plays are played together (at the same time). On the diagram, if one sees parts who have less semantic relation and who may be played in parallel, one may deduce plays.

FIGURE 4.5 – *Bonne pratique de modélisation IMS-LD*

La première étape consiste à identifier/définir les différentes activités. La deuxième vise quant à elle à identifier/définir les rôles et à les associer aux activités précédentes. La troisième étape a pour objectif de spécifier l'organisation temporelle entre les activités : parallélisme, séquencement ou alternatives. Les étapes 4, 5 et 6 sont destinées à structurer les activités à une granularité de plus en plus grande :

- (4) en structure d'activités : un groupe d'activités qui peuvent être vues comme une activité plus générale et qu'on appelle "structure d'activités". Ces activités sont jouées par une personne (via le rôle associé à la structure).
- (5) en actes : un acte consiste en une liste d'associations rôles - structures d'activités.
- (6) en pièces : une pièce est une série séquentielle d'actes. Si les actes sont joués séquentiellement, toutes les pièces sont jouées en parallèle.

Cette bonne pratique a souvent été citée par les concepteurs d'LD car elle permet d'y voir plus clair sur l'ensemble des concepts. Cela rejoint notre réflexion sur le besoin d'associer un processus de modélisation à un langage pour lui donner du sens.
Nous avons créé la méthodologie correspondante à cette bonne pratique dans ModX (Marvie et al. 2005, Le Pallec et al. 2006a). La capture d'écran montre la définition de l'étape 2 (ajout des rôles).

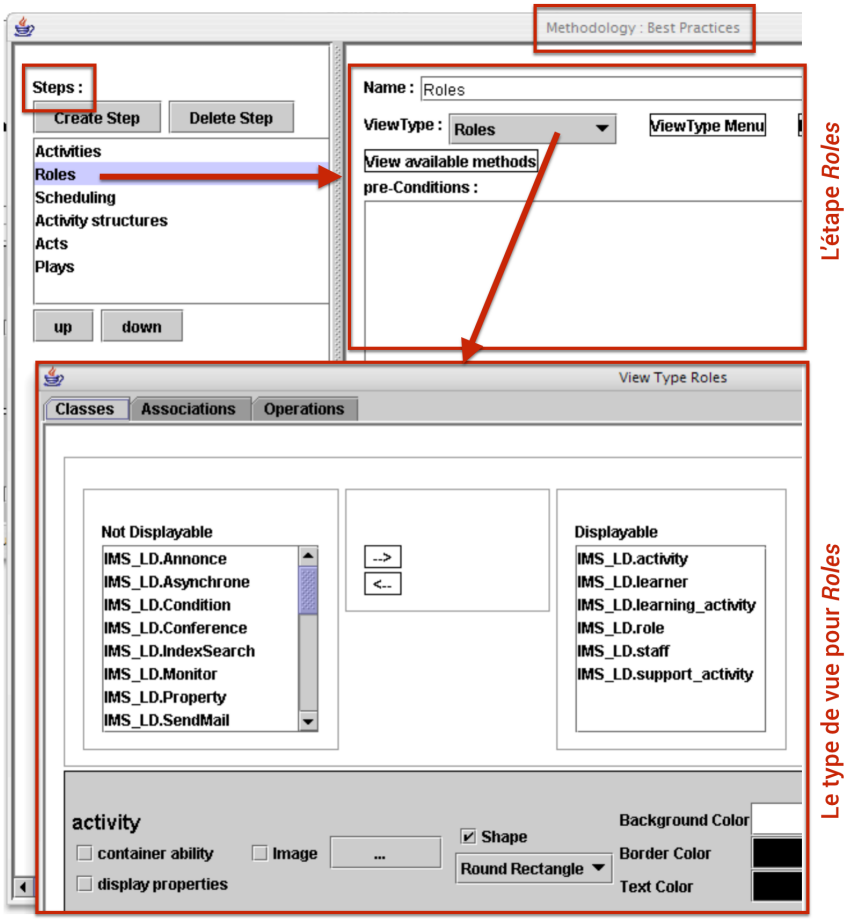


FIGURE 4.6 – *Définition de la méthodologie / bonnes pratiques IMS-LD dans ModX (étape 2)*

On peut voir cette étape en action dans la capture suivante.

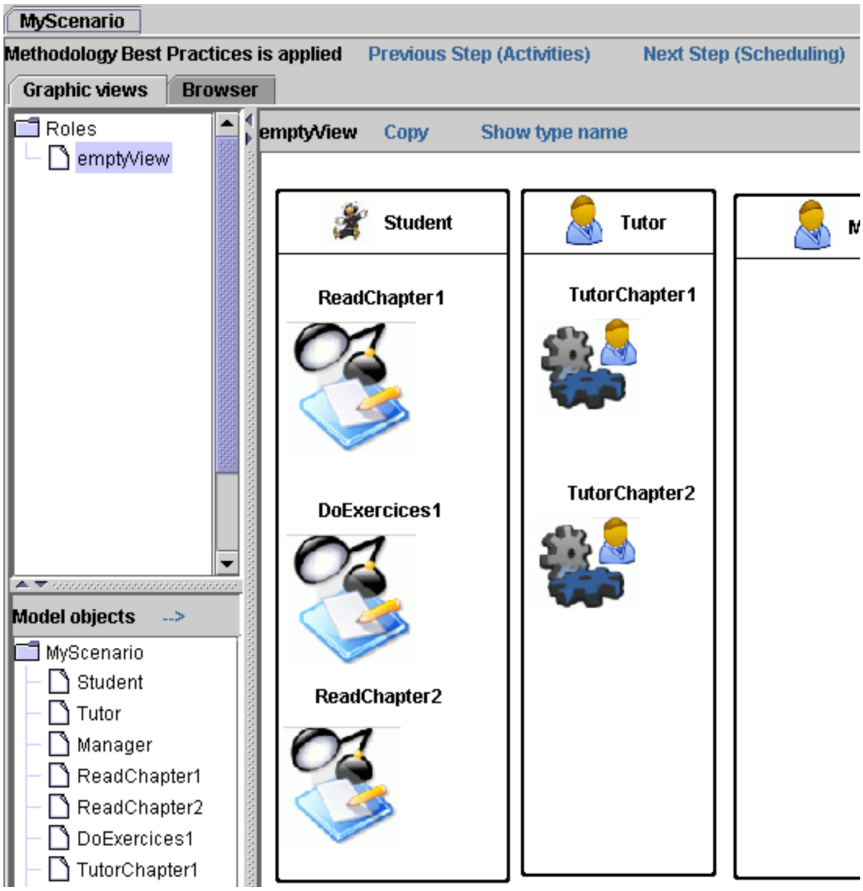


FIGURE 4.7 – *Spécification des rôles d'un modèle IMS-LD dans l'étape 2*

Ceci a constitué notre première validation : l'existence de processus de modélisation linéaire (IMS-LD) et la possibilité d'implémenter intégralement celui-ci via les PIM/ModX.

4.1.3 En IHM

Pour appliquer les PIM en IHM, nous nous sommes focalisés sur le patron de conception PAC (Tarby et al. 2006). Jean-Claude Tarby enseignait les patrons de conception IHM dans le DESS MICE / master e-services depuis plusieurs années et il avait constaté que les étudiants avaient du mal à appliquer MVC ou plus particulièrement PAC : si les concepts étaient compris, leur mise en application ne se déroulait pas correctement.
Avec Jean-Claude et Raphaël, nous avons donc défini une méthodologie pour ce patron de conception et nous l'avons implémentée dans ModX. Elle est récapitulée dans la figure suivante.

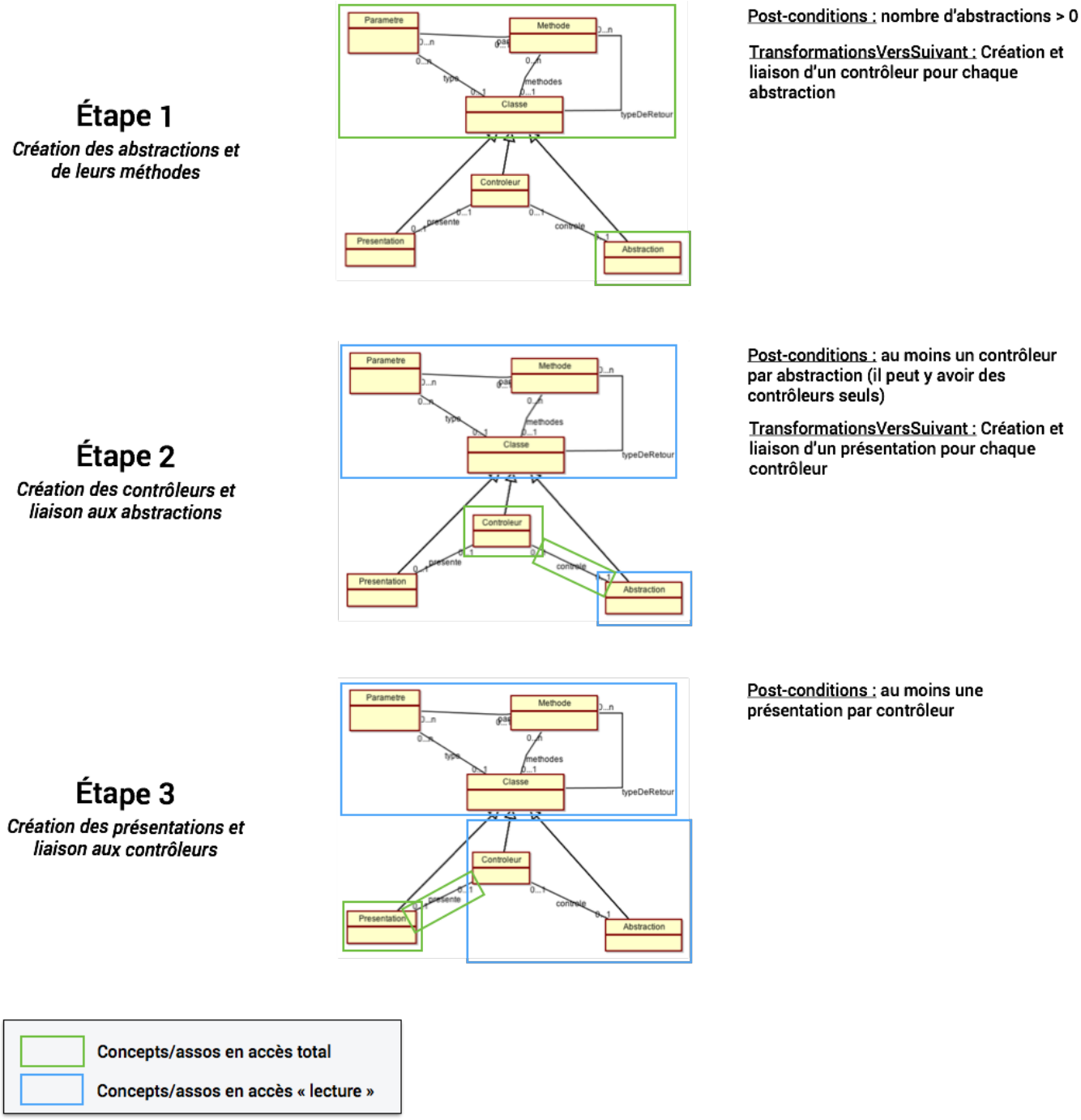


FIGURE 4.8 – *Une méthodologie pour PAC*

Jean-Claude s'est servi de cette méthodologie et de ModX pour ces TP en MICE pendant deux ans. Puis la partie sur les patrons de conception n'a plus fait partie des TP de son cours. Nous n'avons pas fait d'observations qualitatives ou quantitatives de manière scientifique. L'apport de l'outil aux étudiants était indéniable mais ces derniers regrettaient l'aspect "prototype" de la solution. Pour cela, un projet étudiant avait été mis en place pour rendre plus robuste ModX dans une version dédiée à PAC. Mais cela

n'a pas abouti. Par contre, il a été intéressant de voir que la méthodologie s'est affinée de différentes manières au travers de l'usage qu'en ont fait les étudiants : il est logique qu'une méthodologie "académique" évolue pour terminer en une "bonne pratique" issue de l'expérience. Il aurait fallu une pratique sur plusieurs années pour synthétiser les différentes versions et en faire une bonne pratique issue du terrain.

4.1.4 Perspectives

Ce travail sur les PIM est inachevé car il lui manque une expérimentation en bonne et due forme. À l'époque chacun d'entre nous avait d'autres projets et l'énergie nécessaire n'a pas toujours été trouvée. Mais une raison plus "sourde" a été le rejet d'un article présentant notre travail à ECMDA-FA 2005[1]. La principale raison du rejet est qu'un reviewer a perçu notre travail comme un simple mapping de modèles. Avec le recul, je pense qu'à l'époque la communauté IDM se souciait peu des aspects ergonomiques : il y avait beaucoup à faire en terme d'opérateurs (transformations, génération, composition...) et ces aspects "humain" n'étaient pas prioritaires. Il aurait fallu que nous insistions plus sur ces aspects mais des travaux de références manquaient. Par exemple, les dimensions cognitives n'étaient pas aussi reconnues qu'aujourd'hui, Daniel Moody n'avait pas encore défini sa physique des notations et Natalia Juristo et Ana Moreno n'avaient pas encore mis au clair les bonnes pratiques pour concevoir des expérimentations en Software Engineering. Je pense que le contexte est plus favorable maintenant pour ce type de contributions. De plus, la création de l'équipe Carbon (autour de la thématique IHM pour l'IDM) va me permettre de retourner "librement" à ce précédent travail. Enfin, les notations visuelles ayant un regain d'attention actuellement, il serait intéressant de mentionner les directives "graphiques" que j'avais dû implémenter à l'époque et qui étaient accessibles au sein des règles de transformations : par exemple, créer une nouvelle vue vide, rester sur la vue précédente, changer de vue mais la garder de manière permanente... Chacune de ces directives avait été utile pour IMS-LD et PAC et il serait intéressant de déterminer la signification précise de leur besoin.

4.2 Réutiliser les éléments d'un modèle

4.2.1 Origine et problématique générale

Nous l'avons dit dans le chapitre 1 (partie 1.4.2), la réutilisation est importante dans l'industrie du logiciel. Si différentes techniques de réutilisation existent (l'héritage, la composition), il en est une qui permet de tester rapidement l'intérêt (ou non) de réutiliser un artefact logiciel et ensuite de choisir éventuellement un mécanisme de réutilisation plus adapté : le copier/coller (CC). Ce dernier est encore plus intéressant dans le contexte de l'édition graphique de modèles où les éditeurs sont généralement considérés comme lourds : nous avons vu un exemple dans le chapitre 1 où le nombre de clics et de changements de modalités d'interaction était important pour seulement deux éléments de diagramme.

1. L'article soumis a été transformé ensuite en rapport technique (Marvie et al. 2005)

La thèse de Daniel Liabeuf se place dans ce contexte de réutilisation d'éléments de diagramme [2]. C'est une thèse commencée en Octobre 2011, encadrée par moi et dirigée par José Rouillard. Elle est financée par le projet MOANO et s'inscrit dans la problématique générale de faciliter la création de modèles logiciels pour les utilisateurs finaux.
La thèse visait au départ à utiliser les traces des CC (et leur raffinement) produites par un ensemble de concepteurs/utilisateurs afin de détecter des bonnes pratiques de modélisation, ou tout du moins des composants élémentaires de modélisation. Pour chacun de ces composants, il serait aussi déduit les parties fixe et variable. Avec cet ensemble de bonnes pratiques / composants élémentaires de modélisation, nous visions à spécifier un nouveau type d'assistant de modélisation : lorsqu'un concepteur crée des éléments dans un modèle, l'assistant détecte des similarités avec un ou plusieurs composants existants et soit les propose en remplacement soit indique des éléments additionnels possibles (provenant de la différence entre le composant et les éléments créés par le concepteur). Nous avons alors observé "nos" étudiants en TP pour voir comment ils utilisaient le CC et si leur pratique était compatible avec notre projet. Assez vite, nous nous sommes rendus compte qu'ils ne l'utilisaient pratiquement pas. Nous en avons discuté avec d'autres personnes enseignant la modélisation logicielle et ils nous ont confirmé nos observations : eux-mêmes ne l'utilisaient pas car le résultat produit par cette action n'était pas adapté à leur besoin. Enfin, nous en avons finalement parlé avec des concepteurs d'outils UML (OpenFlexo, Papyrus, GenMyModel) et nous avons compris que le CC était difficile à implémenter pour de nombreuses raisons. Par rapport à notre démarche concernant l'assistant de modélisation, nous avons trouvé judicieux et pertinent d'étudier en profondeur la perception du CC par les concepteurs. De plus, cela renvoie à une question beaucoup plus fondamentale : quelle signification donnons-nous à chaque élément d'un diagramme (de modèle) que nous lisons ou manipulons. Dans cette problématique générale, l'étude du CC se focalise ici sur le principe d'identité des éléments de diagramme ou de modèle : pour un élément donné, quels sont les éléments liés qui participent à son identité ?
Pour cette étude sur le CC d'éléments de diagramme (Liabeuf et al. 2014), Daniel s'est focalisé sur les diagrammes de classes UML, car ces derniers sont extrêmement utilisés (Forward et al. 2010) et sont très répandus en terme de support logiciel. Enfin, Daniel s'est concentré sur le concept de classe car celui-ci dispose de plusieurs types d'associations sémantiquement et visuellement assez différents (héritage, association, lien avec le paquetage...). Ceci était suffisant pour étudier l'influence des dimensions syntaxique, sémantique et visuelle.

Différence avec le coller spécial

Le problème du CC d'un ou plusieurs éléments de diagramme est de savoir s'il ne faut pas prendre d'autres éléments (non-sélectionnés) afin que la copie ne soit pas trop dénaturée. Il est possible de penser que ce problème renvoie aux solutions de type "coller spécial". Mais ce n'est pas

2. Pour cette partie, le terme **diagramme** renvoie à un diagramme représentant *graphiquement* des éléments d'un modèle

le cas. Le coller spécial correspond à un raffinement interne à la sélection (copiée). En d'autres termes, "que garde-t-on des éléments sélectionnés ?". Dans notre contexte, le raffinement doit plutôt être externe, autrement dit "que doit-on prendre d'autre (que la sélection) pour que le résultat du coller reste fidèle à l'original ?". Pour cette raison, nous parlerons plutôt d'un copier spécial où la sélection peut se voir être étendue.
La figure ci-dessous montre la différence entre le raffinement interne (coller spécial) et externe (copier spécial) sur le cas d'une classe associée à 3 autres classes selon 3 types de liaison différents (héritage, composition, attribut). Un point d'interrogation indique une décision quant à la prise en compte ou non de l'élément associé dans le clonage.

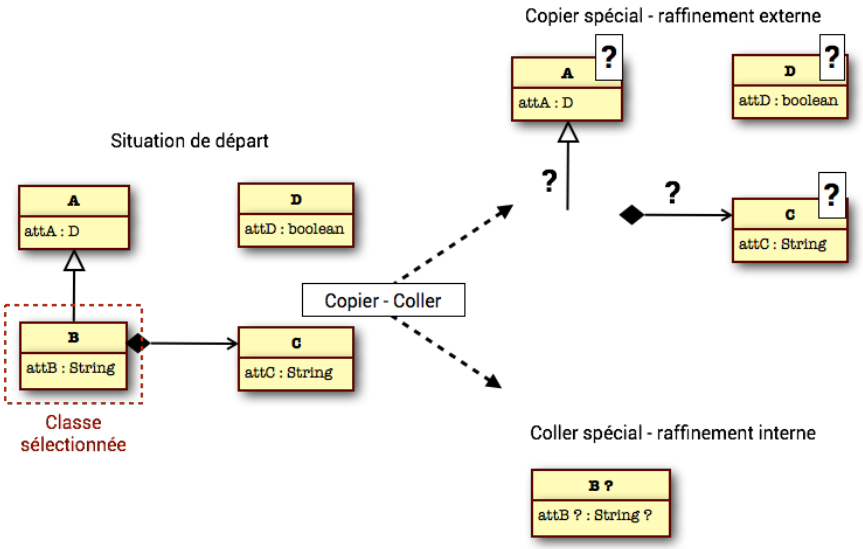


FIGURE 4.9 – *Différence entre le "copier spécial" et le "coller spécial"*

4.2.2 Étudier les facteurs influençant les attentes du CC

L'expression "fidèle à l'original" renvoie à ce que le concepteur et utilisateur du CC perçoit de l'original. Cette perception va dépendre de la tâche en cours. Mais est-ce totalement dépendant ? Si c'est le cas et si on neutralise cette aspect, alors quelque soit la personne qu'on interroge sur le résultat d'un CC en modélisation, nous devrions avoir toujours la même réponse. Et les outils de modélisation graphique n'étant pas influencés par les tâches en cours, tous devraient proposer un comportement identique pour le CC. Le travail de Daniel Liabeuf a donc commencé par la vérification de cette dernière hypothèse. Comme cela était prévisible, les éditeurs réagissent de manière assez différente et aucun consensus n'existe pour le CC de classe UML dans un diagramme. Il y a donc d'autres facteurs que le contexte (i.e. tâche à effectuer) qui influencent les attentes (des concepteurs) d'un CC. Il est coutume de mentionner la syntaxe, la sémantique et la notation visuelle comme dimensions sous-jacentes à un diagramme. C'est ce que nous avons d'ailleurs fait dans le chapitre 1. Nous sommes donc partis du principe que *ces 3 dimensions constituent les précédents facteurs*. L'étude allait consister à vérifier cette hypothèse et dans quelle mesure ces facteurs influençaient les attentes d'un CC.

Classe UML et tests associés

Que ce soit pour comparer le comportement des éditeurs UML ou pour analyser les attentes des concepteurs vis-à-vis du CC, Daniel a utilisé la même batterie de tests. Celle-ci se concentre sur les cinq types de liens possibles suivants pour une classe :

1. l'héritage
2. l'appartenance à un paquetage
3. une association
4. une association de composition
5. un attribut

Au moins deux tests sont associés à chaque association afin d'étudier le CC du point de vue des deux extrémités de celle-ci. Par exemple, pour l'héritage il y a les deux tests suivants :

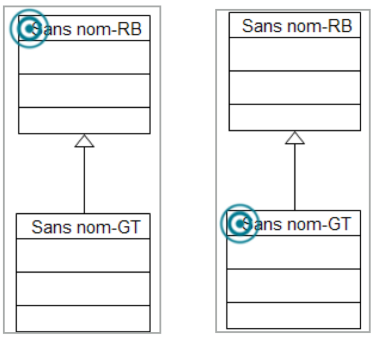


FIGURE 4.10 – *Les 2 tests pour l'héritage*

Comme on le voit sur cette figure, un test est défini par un ensemble d'éléments de diagramme (ici les classes *Sans nom-RB* et *Sans nom-GT*) et un élément sélectionné (indiqué par les trois ronds). Le résultat du test est égal au résultat d'un CC dans cette configuration. Le résultat du test va donc varier selon l'opérateur qui effectue le CC. Dans certains cas, le nom des classes est suffisamment significatif pour indiquer une certaine sémantique à une liaison (exemple utilisé : une voiture est composée de roues). Une différence de réponse en cas de noms significatifs permet de tester les connaissances d'UML du participant.
La batterie de tests va permettre d'étudier en profondeur l'implantation du CC dans les éditeurs et ce qu'en attendent les concepteurs.

4.2.3 Un manque de consensus pour les éditeurs de classes UML

Daniel a testé 23 éditeurs UML dont la version datait pour la plupart de moins de 2 ans. Seuls 15 éditeurs supportent le CC dans un même projet et parmi eux seulement 7 le proposent entre projets différents, soit moins d'un tiers ! Concernant leur comportement, il est résumé dans le tableau suivant (excepté pour les liens avec un paquetage).

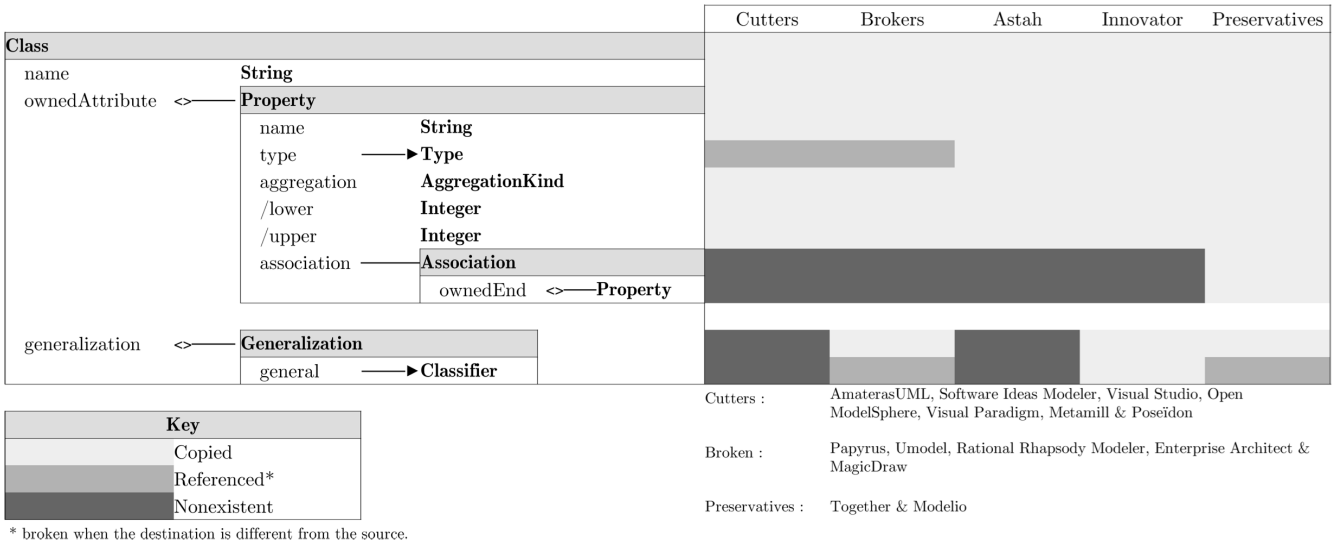


FIGURE 4.11 – *Comportement des éditeurs pour le CC* [3]

Sans rentrer dans les détails, on voit que le comportement varie de manière assez sensible. De ce fait, Daniel a pu invalider l'hypothèse que le CC ne dépend que de la tâche dans lequel il se déroule. D'autres facteurs influencent l'humain (et donc les développeurs d'outils UML) et nous allons maintenant voir l'étude de cette influence.

4.2.4 Influence des dimensions syntaxique, sémantique et visuelle

Dans leur livre (Kelly et Tolvanen 2007), Kelly et Tolvanen indique que le CC par valeur d'éléments de modèles - quelque soit le langage de modélisation - est régie par une certaine profondeur à choisir : *copy by value thus always includes the idea that the copy is to a certain depth*. Ainsi, si on fixe cette profondeur à 1 pour un CC particulier, tous les éléments à distance 1 du ou des éléments sélectionnés seront dupliqués. Le principal problème de cette définition est qu'elle n'indique pas sur quelle base est calculée cette distance ou profondeur, ce qui implique différentes possibilités d'interprétation.

La syntaxe (abstraite)

Nous avons vu dans la partie sur les éditeurs UML, la distance syntaxique : si on considère un modèle comme un graphe orienté, le chemin syntaxique entre deux éléments de modèle est le chemin le plus court entre eux, et la distance correspondante est le nombre d'arcs. Dans le tableau montrant le comportement des éditeurs, la partie gauche était d'ailleurs inventée de manière à refléter la distance syntaxique.
Prendre en compte **la dimension syntaxique** est évident car elle constitue le coeur du langage de modélisation - la syntaxe abstraite - : elle traduit en partie la sémantique du langage et dirige la représentation visuelle (qui

3. Astah et Innovator sont des éditeurs et non des groupes d'éditeurs

est basée dessus). C'est ce que nous avons mentionné dans la partie 1.2.1. Qu'en est-il des deux autres dimensions du langage : la sémantique et la notation visuelle ? Influencent-elles le concepteur dans ses attentes du CC ?

L'enquête

Pour cette question, Daniel a interrogé plus de 70 personnes, en majorité des étudiants, avec différents niveaux de connaissance UML afin d'être représentatifs de la population "praticiens du logiciel". Il y a des étudiants non informaticiens (donc sans connaissance d'UML), des étudiants qui ont vu UML pendant une année, d'autres pendant deux ans et enfin un certain nombre l'ayant vu pendant trois ans et que nous pouvons considérer comme ayant des connaissances solides en UML. A chaque participant était distribué un questionnaire où se trouvait toute la batterie de tests précédemment évoquée. Pour chaque test, un espace blanc était associé et il était demandé au participant de dessiner le résultat du CC avec la classe sélectionnée dans le bout de diagramme correspondant. Afin de ne pas influencer les participants par une tâche particulière, les directives étaient de *pronostiquer le résultat d'un CC effectué par un éditeur UML*. Ceci permettait d'être le plus neutre possible. Au début du questionnaire, étaient posées différentes questions, comme la fréquence d'utilisation d'un éditeur UML et l'utilisation ou non de l'opération de CC.
Afin d'évaluer l'influence de la sémantique et de la notation au travers des réponses fournies, Daniel a d'abord défini le moyen de calculer une distance pour chacune de ces dimensions. Ensuite, il a regardé s'il y avait une corrélation entre la variation de distance et les pronostics des participants. La validation des influences terminée, il a effectué une analyse générale pour voir quelle était la dimension qui semble avoir le plus d'influence.

Notation visuelle

Il y a deux parties de la notation visuelle qui sont pertinentes pour le CC : une concernant les regroupements et une concernant la transparence sémantique. Cette dernière renvoie, d'après la physique des notations, au sens d'un symbole qui peut être inféré ou déduit de son apparence. Nous verrons l'influence de cette transparence sémantique dans la section suivante afin de faire le lien entre notation visuelle et sémantique.
En UML, les seules variables visuelles utilisées sont la position et la forme. Si nous excluons la forme ici (pour l'étudier plus tard), il nous reste la position (x et y). Dans le cas du CC, la position a un rôle considérable car l'être humain a une tendance naturelle à regrouper des éléments visuels entre eux pour constituer des éléments plus gros. Cette tendance a été décrite par les lois de la Gestalt (Wertheimer 1923). Daniel s'est servi de ces lois pour ordonner les distances visuelles dans le cadre du CC. La distance la plus courte est affectée à un élément inclus dans un autre (ex : un attribut dans une classe). La distance suivante (plus longue donc) l'est elle pour les éléments liés. La distance la plus longue l'est pour les éléments non liés (ex : une classe qui est le type de l'attribut d'une autre classe).
Deux tests étaient dédiés à évaluer l'influence de la dimension visuelle : un test avec un attribut et un avec une composition. Sémantiquement et

syntaxiquement, ces deux liens sont, selon UML, identiques. Donc, seul l'aspect visuel change.
Pour vérifier qu'il y a bien une influence de la dimension visuelle, un test de Fisher a été effectué pour invalider l'hypothèse nulle, c'est-à-dire "il n'y a pas de corrélation entre la variation des réponses des participants pour ces 2 tests et la variation visuelle associée". Le test s'est révélé non significatif, et donc a montré l'existence d'une corrélation entre variation visuelle et variation de la réponse.

Notation visuelle 2 : la transparence sémantique

Après les facteurs de regroupement de la notation visuelle, l'étude s'est portée sur la signification des symboles qui peuvent correspondre plus (clairs) ou moins (pervers) à la sémantique à laquelle ils sont associés. Les deux tests qui ont été utilisés pour évaluer l'impact des symboles et leur signification sont ceux de la composition : un où la classe composante est sélectionnée et un où la classe composite est sélectionnée.
Les significations du losange et de la flèche sont diamétralement opposés, d'où un possible impact sur les réponses des participants. Un test statistique équivalent à celui pour les regroupements a été effectué et le résultat indique une corrélation entre variation de la sémantique visuelle et variation des réponses.

Sémantique UML

Pour tester l'importance de la sémantique dans le choix des réponses, le couple de tests était centré sur la composition. Les classes de premier test avaient des noms non significatifs (*Sans nom-RB* et *Sans nom-GT*) et le nom des classes de second test indiquait par contre une composition (*Voiture* et *Roue*). Aucun dimension ne variait sauf la sémantique : elle variait selon le degré de connaissance d'UML du participant. Là encore, un test statistique a montré la corrélation entre cette variation et celle des réponses. x

Influence générale

La limite des trois précédentes observations vient du fait que chacune d'elle est focalisée sur un seul "cas". Cela nous permet certes de montrer l'impact d'une dimension mais pas d'avoir une estimation de la force de cette impact sur les différents réponses observées. Pour cette raison, nous avons effectué une série de tests plus généraux : ils consistent à montrer le taux de compatibilité des réponses des participants par rapport à chaque dimension.
Les histogrammes suivants montrent la répartition des participants par rapport à leur compatibilité à chaque dimension.

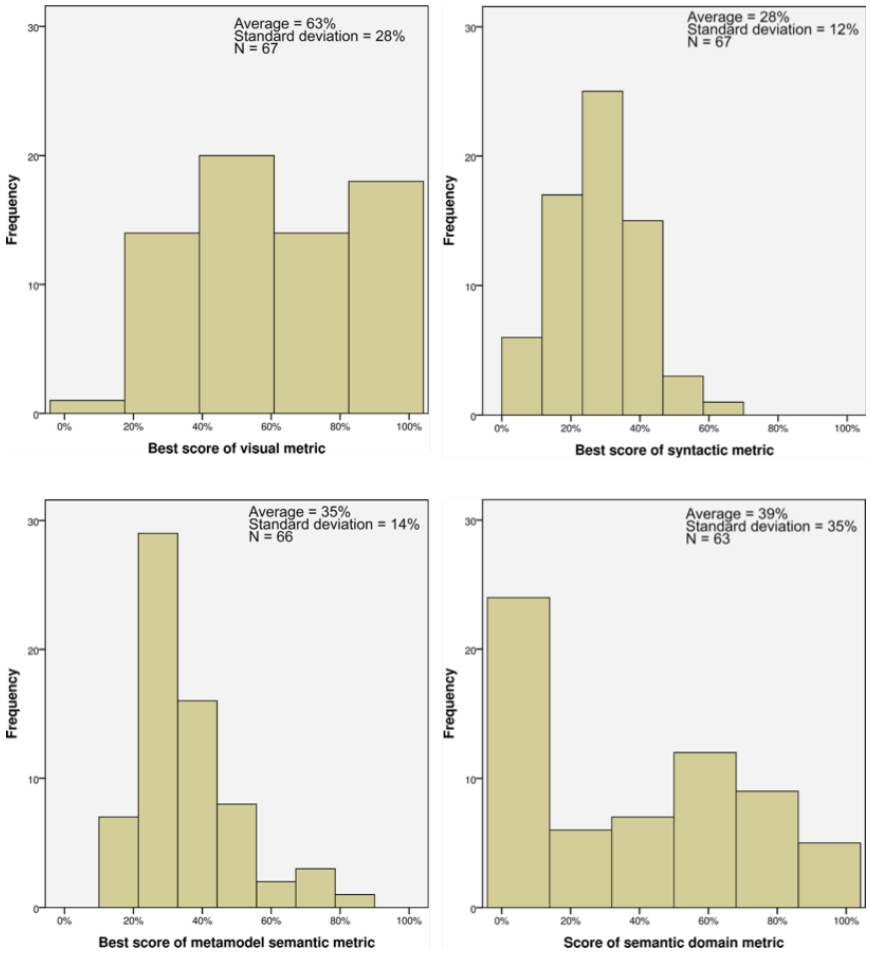


FIGURE 4.12 – *Répartition des participants par rapport aux dimensions (Liabeuf et al. 2014)*

Sur l'histogramme correspondant à la dimension visuelle, on voit par exemple qu'une catégorie de personnes (2 participants) ont des réponses qui sont compatibles entre 0 et 18% avec cette dimension. On trouve 4 autres catégories où les compatibilités varient entre 18% et 40%, entre 40% et 60%, ...

On remarque tout de suite que pour la dimension visuelle, la répartition est assez homogène. Ceci est plutôt mauvais signe : il y a forcément une différence entre le groupe "expert" et le groupe "débutant" ou des différences au sein des mêmes groupes. Dans le premier cas, cela signifie que la notation visuelle n'est pas en phase avec la dimension sémantique d'UML. Dans le second cas, que la sémantique n'est pas suffisamment claire pour qu'on puisse déterminer le raffinement extérieur. Ou les 2 ! En analysant le groupe des experts, on voit qu'il y a différents types de comportement, ce qui renforce l'idée d'une sémantique UML qui présente des faiblesses pour délimiter la portée et l'identité d'une classe... d'où le fait d'être influencé par la notation visuelle pour la décision finale. Cette hypothèse semble réaliste et raisonnable mais elle reste à vérifier de manière rigoureuse.

4.2.5 Perspectives et conclusion sur le travail de Daniel Liabeuf

Ce premier travail réalisé par Daniel permet d'avoir une première ébauche du cahier des charges du copier/coller d'éléments de diagramme. Il montre aussi que le cahier des charges d'une interaction sur un diagramme logiciel se base sur la définition sémantique, syntaxique et visuelle du langage de modélisation correspondant... ce qui est assez logique. Ceci implique que, si la sémantique n'est pas rigoureusement et pleinement définie, le cahier des charges d'une interaction risque d'être difficilement formulable. Ce sera aussi le cas s'il y a un manque de cohérence entre sémantique et notation visuelle.

Nous avons vu ici que la notation visuelle avait une forte influence sur la perception des éléments d'un modèle et qu'elle ajoutait parfois du sens à la syntaxe (*attribut vs composition*). Est-ce que cela veut dire que la syntaxe concrète complète la syntaxe abstraite quand un concepteur souhaite exprimer la sémantique de son langage ? Il faudrait effectuer une étude de terrain pour cela... mais il semble à mon avis que ce soit le cas, ou plutôt que cela devrait l'être. La syntaxe concrète est un excellent moyen de donner du sens à un langage. D'ailleurs c'est dans ce sens, que vont des articles comme (Störrle 2012; 2013, Caire et al. 2013, Genon et al. 2010) publiés dans les conférences MODELS, SLE ou RE. Néanmoins, ces travaux qui montrent l'importance des aspects ergonomiques en IDM se limitent à la notation visuelle. Serait-il possible d'avoir suffisamment d'informations à partir des syntaxes abstraite et concrète pour déduire directement le cahier des charges d'une interaction comme le CC ? Comme pour les aspects visuels et la syntaxe concrète, je pense qu'une syntaxe d'interactions serait plus adapté pour définir les interactions avec un langage de modélisation et qu'elle serait elle aussi un très bon support pour compléter la sémantique du langage. Cette syntaxe d'interactions pourraient déjà inclure la définition de méthodologies et de copier/coller.

L'étude présentée ici constitue la première partie de la thèse de Daniel Liabeuf. Une réponse au problème du CC constitue la seconde partie de sa thèse. Une proposition de solution semble se dessiner autour d'un CC proposant un raffinement externe le plus probable mais avec une possibilité de l'étendre ou le diminuer comme pour un *copier spécial*. Il n'est pas encore décidé si la proposition concernera UML ou tout type de méta-modèle. Si c'est le dernier cas, alors des informations seront à rajouter dans la définition du méta-modèle afin d'indiquer le comportement du CC (ou du moins, ajouter de la sémantique nécessaire pour cette opération).

4.3 Modélisation collaborative et Awareness

La start-up Axellience a été créée fin 2011 avec comme objectif de proposer un éditeur UML en ligne. L'idée est de répondre aux problèmes récurrents des éditeurs UML :

- Installer l'éditeur. Trouver le site web correspondant, avoir la licence, télécharger la bonne version (pour être "compatible" avec celle du reste de l'équipe), paramétrer l'éditeur... La compatibilité implique que tous les membres d'une équipe soient d'accord pour mettre à jour ou non l'éditeur utilisé lorsqu'une nouvelle version apparaît.
- Trouver les fichiers sur lesquels travailler. Il semble que les référentiels de modèles ne soient pas généralisées dans les équipes de développement logiciel et que l'endroit où se trouvent les fichiers "modèle" soit souvent un mystère.
- Lourdeur des éditeurs. De nombreux éditeurs sont lents à lancer et peu réactifs.

L'éditeur proposé par Axellience, appelé GenMyModel, apporte des réponses à ces problèmes. Tout d'abord c'est un éditeur léger car il ne se focalise que sur les opérations de modélisation récurrentes (à la manière d'Apple) : son chargement est rapide et son interface claire. Ensuite, il n'y a pas besoin d'installation ni de mise à jour. Enfin, comme les modèles sont dans le "cloud" et attachés à chaque compte ou équipe, il n'y a plus de problème lié à l'emplacement physiques des modèles.
Une première version bêta a été mise en ligne et l'audience réduit à quelques centaines d'utilisateurs. Axellience a mis en place un système permettant aux beta-testeurs de faire facilement des retours aussi bien concernant les bugs que des demandes d'évolution. Concernant ces dernières, la principale a été rapidement la collaboration. S'il était prévu d'implémenter des mécanismes de collaboration, il n'était pas prévu de devoir les développer si tôt. En observant ce qui existait dans les éditeurs existants, Alexis Muller, le PDG d'Axellience, s'est très vite rendu compte d'une grande faiblesse de ces outils concernant le support à la collaboration. Mais il a aussi vite compris la difficulté inhérente à ce dernier. Une thèse en collaboration avec le LIFL sur ce sujet a donc démarré en Mars 2013 afin d'étudier l'état actuel des supports logiciels à la modélisation collaborative dans l'industrie et dans le monde académique et voir comment les résultats concernant le CSCW [4] pouvaient inspirer une avancée significative dans notre problématique. Cette thèse est effectuée par Michel Dirix, dirigée par Jean-Marc Geib et encadrée par moi-même.
Une fois démarrée, la thèse s'est assez vite focalisée sur l'awareness. Ce dernier, par définition, *provides important information about activities of collaborators and interactions with the shared workspace in order to place the user in the best context*. C'est une des dimensions clés du CSCW avec l'articulation du travail et le protocole de communication logicielle. Vue la complexité structurelle d'un modèle UML et les interactions possibles sur celui-ci, on peut se douter que le choix des informations à transmettre concernant les actions effectuées par les autres et leur représentation au sein d'un modèle peut s'avérer très délicat.
Nous présentons ici la première année de la thèse de Michel Dirix, dont les résultats sont synthétisés dans (Dirix et al. 2014). Le postulat de départ de ses travaux est assez simple. L'awareness consiste essentiellement à af-

4. Computer Supported Collaborative Work : support informatique au travail collaboratif

ficher des informations sur le travail des autres (passé, présent ou futur). Il n'est pas possible de tout afficher sous peine de surcharge cognitive et les informations doivent être affichées judicieusement afin de réduire au minimum les inférences mentales nécessaires pour que le concepteur puisse faire le lien entre le modèle et l'information affichée. Il faut donc faire un choix dans les informations à afficher et ce choix va être guidé par 2 aspects : l'importance de l'information selon le contexte et sa facilité à être affichée judicieusement. La première année s'est focalisée sur l'importance de l'information. L'awareness étant centrée sur l'utilisateur, Michel a adopté une approche centrée utilisateur : analyse - prototypage - utilisation - discussion en vue d'amélioration et nouvelle itération. L'analyse a débuté par l'étude des différentes types d'informations d'awareness dans le contexte de la modélisation collaborative. Comme il y avait déjà un existant, l'analyse a consisté à analyser qui est implémenté dans les outils actuels. Ceci a permis de faire un premier prototype et donc d'intégrer ces mécanismes d'awareness dans GenMyModel, Quatre mois après la mise en production (et donc 4 mois d'utilisation), Michel a enquêté auprès des utilisateurs pour évaluer quelles sont ou quelles seraient les informations d'awareness les plus importantes à afficher. Il en a aussi profité pour voir quelle était l'influence des facteurs extérieurs (taille de l'équipe, distribution géographique...) sur leurs réponses.

4.3.1 Les dimensions de l'Awareness et leur présence dans les outils

Le tableau suivant (Steinmacher et al. 2013) montre les différentes dimensions de l'awareness et pour chacune d'elles les éléments la constituant. Chaque élément répond à une question que peut se poser une personne lors d'une activité collaborative.

Awareness dimension	**Element**	**Question**
Workspace	Identity	Who is that?
	Authorship	Who is doing that?
	Action	What are they doing?
	Action history	How did that operation happen?
	Intention	What goal is that action part of?
	Intention history	What goal was that action part of?
	Artifact	What object are they working on?
	Artifact history	How did this artifact come to be in this state?
	Location	Where are they working?
	Location history	Where has a person been?
	Gaze	Where are they looking?
	View	Where can they see?
	Reach	Where can they reach?
	Event history	When did that event happen?
Informal	Opinion	What is their opinion?
	Presence	Is anyone in the workspace?
	Presence history	Who was here and when?
Social	Interest level	How interested are they?
	Emotional feelings	How are they feeling?
	Availability	Are they available?
Group-structural	Roles and responsibilities	What are their roles/positions?

FIGURE 4.13 – *Les questions liées à la collaboration (Steinmacher et al. 2013)*

La dimension espace de travail (workspace) concerne la connaissance des actions et interactions qui se passent *actuellement* dans l'espace de travail. La dimension informelle est plus générale et concerne des aspects plus macroscopiques. La dimension sociale renvoie à des questions liées

à l'opinion ou au ressenti des collaborateurs. Enfin une dimension est dédiée aux rôles et responsabilités associées, la structure du groupe.

Implémentations existantes

Le support à l'awareness au sein des outils commerciaux est assez limité. Tout d'abord, les éditeurs de modèles ne proposent pas de collaboration temps-réel et les mécanismes d'awareness sont très limités d'un point de vue ergonomique (descriptif textuelle des modifications). Ce sont les éditeurs de diagrammes (non basés sur un méta-modèle) qui proposent le plus de fonctionnalités. Toutefois, une étude approfondie des outils Creately (`http://creately.com/`), LucidChart (`https://www.lucidchart.com/`) et Visio (`http://office.microsoft.com/en-001/visio`) nous a montré que le support à l'awareness est souvent réduit à la liste des personnes connectées (*Identité* et *Présence*), les éléments sur lesquels ils travaillent (*Artefact* et *Paternité* partiels) et parfois un historique partiel des modifications (*Historique des événements*).
Concernant les prototypes de recherche, seuls deux éditeurs de modèles collaboratifs existent. Ce sont des outils en ligne : GEMSjax (Farwick et al. 2010b) et SLIM (Thum et al. 2009). GEMSjax est un outil de (méta-)modélisation qui propose un chat et indique les éléments manipulés par les autres collaborateurs. L'awareness se limite donc ici à la *Paternité* et *Artefact*, mais de manière partielle car il faut travailler sur le même diagramme. SLIM propose les mêmes mécanismes mais ajoute la liste des collaborateurs connectés, ce qui apporte les informations liées à l'*Identité* et la *Présence*.

4.3.2 Étude de terrain

Les éditeurs de diagrammes (UML ou non) et même les prototypes de recherche existants proposent/affichent peu d'informations pour permettre à un utilisateur d'être conscient du travail de ces collaborateurs. Nous aurions pu nous dire qu'il suffisait d'implémenter des fonctionnalités pour afficher l'ensemble des informations liées à l'awareness. Il ne resterai plus, ensuite, qu'à raffiner l'affichage de ces informations. Mais comme nous l'avons indiqué, l'espace visuel est limité et il faut faire des choix. De plus, afficher trop d'informations sature rapidement l'utilisateur. En adoptant une démanche centrée utilisateur, Michel a entrepris d'interroger les utilisateurs de GenMyModel sur ce point. Mais, afin de mener une enquête plus aboutie, l'idée a été de passer directement à la phase de discussion / feedback des utilisateurs. Michel a implémenté les fonctions de collaboration que l'on trouve dans SLIM ou LucidChart, les a rendu disponible sur GenMyModel et a attendu un peu plus de 2 mois d'utilisation pour : 1) voir quelles étaient les articulations de travail qui émergeaient ; 2) interroger des utilisateurs ayant fait l'expérience de la collaboration quant aux informations d'awareness qu'ils jugent importantes et qu'ils souhaiteraient avoir.

3 articulations de travail

Deux mois et demi après le déploiement des fonctionnalités de collaboration, 508 projets collaboratifs réels[5] étaient gérés par GenMyModel. Michel les a d'abord analysés afin de trouver les articulations de travail qui en ressortaient. Il y a d'abord 2 grandes catégories :

- Les projets où une seule personne travaille et les autres ne sont que spectateurs. Cela peut être le cas d'une équipe de développement où seul le fonctionnel spécifie le modèle (un unique contributeur (UC))
- Les projets où tout le monde contribue au modèle (plusieurs contributeurs (PC))

Dans le deuxième grand type d'articulations, nous trouvons deux articulations qui cohabitent : un fonctionnement tour par tour (TPT) et des phases de collaboration temps réel (TR).

Enquête Web : le questionnaire

Michel a contacté les personnes impliquées dans ces projets pour leur soumettre un questionnaire électronique. Comme il y avait deux types de projets du point de vue collaboratif, deux questionnaires ont été produits. Un tri a été fait pour que seules les questions pertinentes (liées à l'awareness) pour un type de projet demeurent dans le questionnaire correspondant. Dans chacun des 2 questionnaires, des informations étaient demandées sur la taille de l'équipe, la répartition géographique, le type de projet (étudiant, académique, industriel) et le type d'application modélisée (mobile, web ou desktop). Ceci permettait d'évaluer l'impact du contexte sur le besoin d'informations liées à la collaboration. Quelque soit le questionnaire, la personne interrogée devait noter l'utilité des éléments d'awareness qui lui étaient présentés (accordés au type d'articulation). Par exemple pour *Historique de la présence*, il était demandé de noter l'utilité de 1(peu utile) à 4 (très utile) l'information suivante *Qui était connecté quand vous n'étiez pas en ligne*.
Le questionnaire sur le projet à plusieurs contributeurs se terminait par demander de mettre dans l'ordre d'importance les 5 éléments d'awareness les plus utiles. Le classement permet de connaître les informations liées à l'awareness jugées les plus importantes. Les notes sur leur utilité permettent de voir comment ce classement peut évoluer en fonction du contexte du projet.
17 personnes sur 308 contactées ont répondu pour les projets UC. 50 personnes ont répondu pour l'autre type de projet.

4.3.3 Classement et influence du contexte

Classement des éléments d'awareness pour les projets PC

Pour le tour par tour, ce sont les *actions effectuées*, sur quoi et par qui, qui sont en haut du classement. On retrouve d'ailleurs en partie cela dans

5. dont le modèle n'était pas issu d'un clônage du modèle test du tutorial de GenMyModel, et qui impliquait un nombre suffisamment grand d'actions effectuées

les systèmes de versionning (SVN, Git) ou dans des systèmes collaboratifs comme Google Docs. Ensuite, c'est l'*intention* (et l'historique associé) qui est l'élément jugé le plus utile. Ce qui est intéressant est de voir que ces caractéristiques ne sont pas implémentées dans les outils vus précédemment. Nous avons dit que certains éditeurs classiques (i.e. non Web) implantaient des systèmes de versionning, mais ces systèmes ne sont pas adaptés aux diagrammes (compte-rendu des différences sous forme de texte) et leur utilisation est trop fastidieuse pour qu'on puisse affirmer qu'il y ait une véritable implémentation.

Pour le temps réel, les éléments d'awareness en haut du classement sont ceux que l'on retrouve dans LucidChart ou SLIM (*Présence*, *Identité*, *Paternité*). Il y a donc une assez bonne adéquation de ces outils avec la demande pour les phases de Temps Réel. Toutefois, comme pour le TPT l'*intention* est assez demandée (quatrième du classement) et elle n'est implémentée dans aucun outil. Enfin, l'*historique* (de la paternité ou des intentions) est ici moins importante. Toutefois, cet historique reste important au démarrage d'une session (en TPT ou en TR). Michel avait relevé qu'il n'y avait aucune action la première minute d'une session : une question avait été intégrée dans le questionnaire pour savoir ce que chacun faisait pendant cette minute. Majoritairement, les participants ont répondu qu'ils regardaient les changements qui avaient été faits depuis la dernière fois (*historique*).

Influence du contexte

Dans l'ensemble, les notes correspondent aux classements précédents. L'intérêt principal de ces notes est de pouvoir évaluer une influence du contexte sur l'importance de chaque élément d'awareness. Pour identifier s'il y avait une corrélation entre la note des éléments d'awareness et des facteurs extérieurs (ex : taille du projet), des tests de corrélation de Pearson ont été effectués.

Si aucune corrélation n'a été détecté pour les projets UC, ce n'est pas le cas pour les projets à plusieurs contributeurs.

Pour les phases de TPT, la seule corrélation pour laquelle nous avons une hypothèse plausible concerne celle entre les projets étudiants et *l'opinion + historique*. Nous pensons que les étudiants ne sont pas des experts et ont donc plus besoin de l'opinion des autres, et de voir comment ils ont procédés, afin de construire leur modèle. Les autres corrélations (ex : taille du projet et l'utilité de l'opinion, le type d'applications modélisées sur certains éléments d'awareness) restent énigmatiques et des discussions avec des concepteurs nous aideront à mieux comprendre ces liens.

Pour les phases de TR, il y a encore plus de corrélations.La taille du projet et celle de l'équipe semblent avoir un impact sur l'utilité perçue de nombreux éléments d'awareness. Cela semble logique pour de nombreuses corrélations :

- Plus un projet est grand, plus les collaborateurs ont besoin de s'organiser et donc de connaître les intentions de tous et les change-

ments qu'ils font. Il n'est peut-être pas nécessaire que ces informations soient gérées par le support logiciel pour de petits projets, mais pour des grands projets, vu le nombre d'information, ce n'est sûrement plus le cas.

- Plus l'équipe est grande, plus les collaborateurs vont se poser des questions sur l'endroit où se trouvent les autres et comment les joindre. Une hypothèse est qu'il est important d'humaniser ses collaborateurs et quand l'équipe est grande et qu'on ne connait pas tout le monde, et donc on souhaite avoir plus de détails sur eux. Comme pour le TPT, on retrouve aussi une utilité croissance de l'opinion quand le nombre de collaborateurs augmente.
- ...

Dans les projets à plusieurs contributeurs, l'importance des informations liées à l'awareness semble donc impactée par le contexte de différentes manières. Nous comptons sur l'intégration dans GenMyModel de ces différentes informations pour évaluer si l'impact est suffisamment important pour implémenter un affichage contextualisé.

4.3.4 Conclusion et perspectives de travail

Le travail effectué par Michel permet déjà de contribuer à la problématique de la modélisation collaborative en indiquant quels sont les besoins actuels des concepteurs et surtout leurs relations avec les articulations de travail et le contexte des équipes. Il reste à implémenter des mécanismes pour afficher les informations actuellement absentes et pour les adapter au contexte (pas seulement du projet). Cette adaptation renvoie à des questions d'ergonomie cognitive (ici : quelle information à quel moment). Parmi ces questions, il y a aussi celle liée à la granularité. Par exemple, doit-on simplement notifier visuellement qu'une classe a été modifiée (en la mettant dans une couleur particulière) ou doit-on indiquer (toujours visuellement) l'attribut ou la propriété à l'origine de la modification ? Doit-on aussi indiquer le déplacement d'une classe ? Si oui, comment : tout le parcours effectué ou faire une compression des déplacements et ne montrer que les différences ? Il est bien sûr prévu que chaque ajout dans GenMyModel lié à l'aspect collaboratif fasse l'objet de tests utilisateurs et de discussion avec ces derniers. Ceci permettra aussi 1) de voir si les corrélations observées étaient la trace d'une réelle influence ou juste un lien fortuit 2) de garder ou non certaines informations.
Il est étonnant de voir que l'aspect collaboratif de l'activité de modélisation ait été au centre de si peu de travaux et que, 20 ans après le création d'UML, un support logiciel efficace à la collaboration n'existe pas. Pourtant, comme pour l'assistance à la modélisation et le copier/coller, il y a un intérêt à spécifier finement ce type d'interactivité :

la cartographie qu'a réalisée Michel et les futurs tests utilisateur vont permettre d'indiquer quels sont les éléments d'UML qui doivent participer à l'affichage de chaque élément d'awareness. Ce type de specification va apporter du sens à UML et nous conforte dans l'idée de l'intérêt d'une syntaxe d'interactions.

Conclusion du chapitre

Derrière chacun des travaux présentés dans ce chapitre, il y avait la même motivation : améliorer le support logiciel à l'activité de modélisation, et par la même faciliter celle-ci. Le premier travail a porté sur l'apprentissage d'un langage de modélisation et la présence d'un guidage logiciel. La réutilisation était au coeur du deuxième travail. Il n'était pas question d'une réutilisation logiciel comme l'utilisation de composant ou d'héritage mais des aspects interactifs de la réutilisation qui commence souvent par un copier/coller. Enfin, un intérêt des diagrammes logiciels est qu'il facilite la communication entre collaborateurs. Nous avons donc commencé un travail pour faciliter (logiciellement) cette collaboration.
Ces aspects interactifs/ergonomiques sont réellement fondamentaux dans ma perception de l'IDM. Comme je l'ai dit au départ, un langage de modélisation ne se réduit pas à une syntaxe abstraite et une syntaxe concrète : une autre *préoccupation* clé de ce système est l'interaction. L'objectif des travaux de Daniel Liabeuf et Michel Dirix est, comme pour les PIM, de déboucher sur une spécification précise du comportement du support logiciel pour le copier/coller et la modélisation collaborative, et de montrer que ces spécifications apportent du sens au langage de modélisation.

Les notations visuelles

5

Sommaire

Durant ma thèse, soutenue en 2002, j'avais développé un outil de méta-modélisation appelé RAM3 (Le Pallec et Bourguin 2001). Ma motivation était d'explorer les possibilités qu'offrait la méta-modélisation en Ingénierie Logicielle. RAM3 permettait de créer des méta-modèles ou des modèles, soit de façon textuelle soit via des formulaires. L'utilisation de diagrammes n'était pas implémentée. Comme je l'ai indiqué dans le chapitre 2, un tel outil restait important pour les travaux que j'allais entreprendre mais il lui fallait devenir plus simple : c'est là qu'est né ModX. Sa différence avec RAM3 était essentiellement sa capacité à proposer des syntaxes concrètes diagrammatiques. En implémentant ces mécanismes graphiques, mon premier constat a été de voir que les possibilités étaient réellement nombreuses et qu'aucun travaux n'avait défini de manière rigoureuse l'espace de conception graphique pour les diagrammes logiciels. Il était donc difficile de savoir si les *besoins utilisateurs* (les utilisateurs étant ici les méta-concepteurs) étaient satisfaits. Comme cela ne semblait pas une préoccupation scientifique à l'époque, il semblait risqué d'y investir du temps. Un deuxième constat est apparu au fur et à mesure des différents langages de modélisation qui ont été créés avec ModX : la difficulté à évaluer leur qualité cognitive. En effet, certains langages ne me paraissaient pas optimaux d'un point de vue cognitif mais j'avais alors du

mal à expliquer pourquoi à leur concepteur/commanditaire. Et l'inverse était aussi vrai : certaines personnes avaient du mal à me dire les raisons pour lesquelles ils trouvaient peu utilisable un langage que j'avais conçu. De nombreuses fois, j'ai souhaité orienter mes travaux de recherche vers l'efficacité cognitive des diagrammes mais une *croyance* existait dans la communauté IDM : "toute cette problématique a déjà du être traitée en psychologie, et il n'y a donc rien à faire". Mais en 2009, Daniel L. Moody a publié un article (Moody 2009) dans *IEEE Transactions of Software Engineering* où il définit la "Physique des notations" (PdN) dans lequel il faisait le point sur cette problématique. Effectivement des travaux en psychologie, en sociologie, en cartographie, etc. avaient été menés sur les représentations graphiques mais ils constituent uniquement de la matière première qu'il fallait travailler pour parvenir à des directives et des règles quant à l'écriture de diagrammes logiciels. Moody a synthétisé l'ensemble de ces travaux et les a adaptés, transformés, reformulés afin de fournir des lignes directrices pour le concepteur de langage de modélisation logicielle. Dans le chapitre 1, nous avons décrit ces 9 critères. Nous avons aussi indiqué qu'ils manquaient d'éléments quantitatifs et que tels quels, ils ne permettaient pas à un concepteur de langage de modélisation d'être guidé précisément : difficile de suivre *rigoureusement* des principes s'il ne sont pas définis de manière *rigoureuse*. Par conséquence, il m'était difficile d'intégrer ces critères dans l'outil ModX. Et comme ce sujet me passionnait, j'ai donc commencé à travailler sur cette problématique.

Mon travail autour des notations visuelles constitue une activité scientifique naissante. Pour cette raison, ce chapitre va montrer des travaux moins aboutis que ceux présentés tout au long de ce document. Il constitue un aperçu de mes prochaines années de recherche. La première section évoque d'abord la nécessité de définir des métriques pour les notations visuelles. La seconde section se focalise essentiellement sur la base - qui me semble nécessaire - à tout travail concernant la spécification rigoureuse d'un critère sur la qualité cognitive : définir l'espace informationnel associé à la qualité des notations visuelles, c'est-à-dire identifier les variables informationnelles qui sous-tendent chaque métrique possible. Nous avons intégré ces variables dans ModX afin de permettre à tout méta-concepteur de pouvoir facilement et rapidement spécifier de telles métriques et les appliquer sur tout langage manipulé dans ModX. Enfin, la dernière section montre une étude pilote que j'ai menée avec l'équipe IIHM de Grenoble et l'équipe de Patrick Heymans de l'Université de Namur sur le critère de *Discrimination Perceptuelle*.

5.1 Pourquoi définir des métriques

Si les 9 critères de la PdN éclairent le concepteur de langages sur les bonnes pratiques à suivre pour la conception de notation visuelle, ils ne sont pas suffisamment explicites pour le guider précisément, ou pour qu'ils puissent être utilisés pour mesurer la qualité d'une notation visuelle. Pour montrer cet état de fait, nous allons prendre le cas de la clarté sémiotique. Celle-ci est souvent vue comme le critère le plus simple : en effet, il faut que chaque concept de la syntaxe abstraite soit représenté par un seul symbole de la syntaxe concrète et que celui-ci ne soit pas utilisé pour un autre concept. Cette règle vaut aussi pour les associations.
A première vue, on peut se dire qu'à partir du moment où tous les symboles choisis sont différents, il ne reste qu'à vérifier que la relation *1 concept - 1 symbole* est respectée. Mais dans cette affirmation, "différent" est un opérateur qui n'est pas si simple à définir de manière rigoureuse. La figure suivante reprend la syntaxe abstraite utilisée dans le chapitre 1 et la syntaxe concrète qui lui était associée (ici, indiquée comme la numéro 1). La figure affiche aussi une syntaxe concrète alternative (la numéro 2).

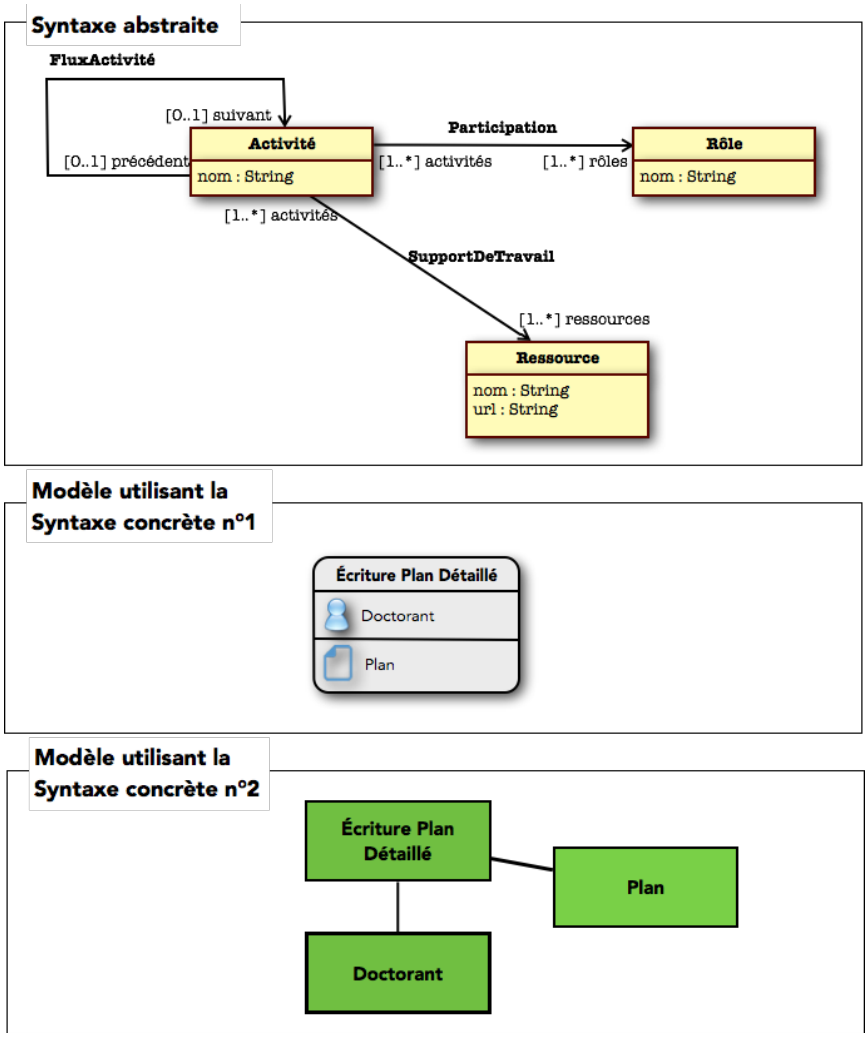


Figure 5.1 – *Différences perceptuelles - syntaxe concrète*

Cette syntaxe alternative est définie comme suit :

- le concept Tâche
  - × un rectangle
  - × couleur de fond RVB (112,191,65)
  - × bordure largeur = 2pt, couleur = RBV (0,0,0)
- le concept Doctorant
  - × un rectangle
  - × couleur de fond RVB (112,191,65)
  - × bordure largeur = 3pt, couleur = RBV (0,0,0)
- le concept Ressource
  - × un rectangle
  - × couleur de fond RVB (121,208,71)
  - × bordure largeur = 2pt, couleur = RBV (0,0,0)
- l'association Participant
  - × trait de largeur 2pt
- l'association SupportDeTravail
  - × trait de largeur 3pt

À la vue de ces données, on peut conclure que le critère de clarté sémiotique est respecté (couleurs différentes ou largeurs de traits/bordures différents). Pourtant, en regardant le diagramme qui utilise cette notation (dans la figure précédente), les différences sont très faibles, et de nombreuses personnes ne vont voir que des rectangles verts avec des traits/bordures identiques. La question qui découle de cette situation est : à partir de quand peut-on considérer que des symboles sont différents ? Cette question renvoie au critère appelé discrimination perceptuelle. Dans ce dernier, le premier élément pour dire si deux symboles sont différents est la distance visuelle. Cette mesure est fonction du nombre de variables visuelles sur lesquelles les deux symboles diffèrent et l'importance de ces différences. Moody indique qu'il y a certes une infinité de valeurs pour chacune de ces variables, mais que des études ont montré qu'il y a des niveaux ou paliers perceptibles qui ont été identifiés et qui peuvent aider le concepteur à faire des choix : si celui-ci souhaite prendre des nuances différentes de vert, les précédentes études montreraient "quels verts" choisir. Il cite 3 travaux présentant ces études. La sémiologie graphique de J. Bertin (Bertin 1983) en fait partie, et a d'ailleurs une place importante dans le travail de Moody. Mais l'auteur de la sémiologie graphique relate une réalité plus complexe que ce qu'avance Moody. Voici trois observations qui montrent, sur ce seul critère, la nécessité d'approfondir la réflexion autour de la PdN pour rendre celle-ci opérationnelle, c'est-à-dire formuler de manière rigoureuse ses neuf critères.

Interrelation des variables Tout d'abord, les niveaux perceptibles d'une variable visuelle dépendent des autres variables utilisées. Par

exemple, la différence de couleur (ou de luminosité) est perçue à partir d'une certaine taille de la surface colorée. Donc une différence de couleur pour la bordure ou le texte sera bien moins efficace que pour le fond. La variable rotation dépend de la forme utilisée : la rotation d'un cercle est inutile, la rotation de 90° d'un carré aussi. Ceci peut sembler simpliste, mais dans une perspective visant à formuler rigoureusement la distance visuelle, ce genre de considérations complexifie les règles de calcul.

Poids des variables Une autre subtilité concerne le poids des variables visuelles : elles n'ont pas toutes le même impact dans le calcul de la distance visuelle. Bertin conseille d'utiliser plutôt une variation de luminosité que de couleur car cette dernière nécessite de nombreux mouvements oculaires (les cellules détectant la lumière sont localisées dans une petite zone de la rétine appelée le fovéa). Si l'excitation sensorielle est plus importante avec la couleur (très forte densité des précédentes cellules dans le fovéa), cette variable est moins rapide et peut, en plus, poser des problèmes avec les daltoniens. La forme, si elle offre une palette importante de variations (par rapport à la couleur ou à la luminosité), est très mauvaise en terme de performance : un mouvement oculaire est pratiquement nécessaire pour analyser chaque forme car seul le fovéa a une densité - et donc une résolution - suffisante pour bien détecter les contours d'une forme. Cette différence de poids implique qu'il peut être plus efficace de faire varier une seule variable visuelle plutôt que deux (ex : couleur vs. forme+rotation).

Proximité de la sémantique visuelle Deux symboles dont la forme est bien différente graphiquement peuvent être perçus comme identique si leur sémantique visuelle est la même. Par exemple, si dans le précédent méta-modèle, nous avions le concept d'Utilisateur, et que nous choisissions  pour sa représentation, il pourrait y avoir une confusion entre Rôle et Utilisateur. Comme l'indique Bertin (Bertin 1983) (page 95), quand le *lecteur* parcourt un grand ensemble de symboles, "[il] ne s'arrête qu'aux formes figuratives et ignore pratiquement les formes géométriques. L'idée est seule créatrice d'intérêt et de possibilité de mémorisation, non la figure". Ce genre de considération est d'ailleurs repris par Moody au travers du critère de transparence sémantique, même si ce dernier est d'abord défini pour rappeler l'importance de l'alignement entre la sémantique d'un concept et la sémantique visuelle de sa représentation.

Ces observations sont des exemples de faits qui montrent la difficulté à opérationnaliser la PdN, c'est-à-dire de définir des règles ou des métriques issus des critères et de les intégrer dans des outils de méta-modélisation tel que Obeo Designer, Meta Edit+ ou ModX afin de guider le concepteur de langage de modélisation. Le succès actuel des outils de mesure de la qualité du code tel que SonarQube [1] ou Squale [2] montre l'intérêt d'avoir

1. http://www.sonarqube.org
2. http://www.squale.org

des outils qui vérifie le respect de règles issues de bonnes pratiques. La définition rigoureuse de *métriques* permet aussi à une communauté de se mettre d'accord de manière précise et rigoureuse sur des règles à suivre. Moody indique que "There is empirical evidence from the SE field for most of the principles : Semantic Transparency, Complexity Management, Cognitive Integration, Graphic Economy, and Cognitive Fit ; and from other fields for all principles except Semiotic Clarity". Ces évidences devraient permettre - en partie - de définir des métriques correspondant aux critères de la PdN. Des expérimentations pourraient compléter ces évidences, si celles-ci se révélaient insuffisantes.

C'est dans cette direction que j'ai récemment décidé d'orienter ma recherche : analyser les travaux référencés par la PdN pour affiner ses critères - réaliser des expérimentations si cela s'avère nécessaire - et intégrer les métriques en résultant dans un ou plusieurs outils de méta-modélisation afin d'améliorer ces métriques - voir de remettre en question certains critères de la PdN - au travers des retours utilisateurs.

J'ai commencé cette activité avec Sophie Dupuy-Chessa du Laboratoire d'Informatique de Grenoble. Elle est membre des équipes SIGMA (Système d'Information) et IIHM. Cette double casquette est le reflet de sa thématique de recherche : les Interactions Homme-Machine dans le contexte des Systèmes d'Information. Une des problématiques qu'elle aborde concerne les interactions lors du paramétrage de ces systèmes, en particulier à l'aide de langages visuels comme BPMN. Dans ce contexte, un objectif est d'améliorer l'efficacité cognitive de ce type de langage afin de les rendre plus accessibles aux utilisateurs de SI. C'est donc naturellement que nous avons commencé à collaborer d'abord par le dépôt d'un sujet lors des appels à défi du GDR-GPL en 2009 puis par la création d'une action spécifique de ce même GDR en 2010 et enfin dans le cadre du projet ANR MOANO (2011-2014). Par la suite, nous avons été rejoint par Nicolas Genon en thèse à Namur sous la direction de Patrick Heymans. Ce dernier avait entamé une collaboration avec Daniel Moody autour de la PdN et la thèse de Nicolas Genon portait principalement sur la transparence sémantique. Avec Nicolas Genon, je me suis principalement focalisé sur l'étude de la sémiologie graphique afin de consolider certains critères de la PdN. En effet, tous les deux, nous avions été frappés par l'ouvrage de Jacques Bertin qui est une mine d'information [3] : non seulement il peut apporter suffisamment de matière pour affiner certains critères de la PdN, mais la structuration - définie par Bertin - de l'espace de conception graphique peut aussi créer de nouveaux critères et mieux définir les liens entre eux.

Quand Sophie Dupuy-Chessa et moi avons commencé à essayer de définir des métriques liées à la PdN, nous nous sommes vite aperçu d'une des difficultés de ce travail : prendre en compte la grande variété de cas possibles dans l'espace de conception graphique, comme le montre l'exemple de la rotation dont l'effet est annulé pour certaines valeurs sur certaines formes. Nous avons donc commencer par travailler sur l'intégration de règles/métriques dans un outil de méta-modélisation afin de pouvoir adopter une approche itérative et pragmatique : utiliser un grand nombre

3. Le travail de Jacques Bertin est mondialement reconnu : il est une des rares personnes à avoir reçu la médaille d'or *Carl Mannerfelt* de l'Association Internationale de Cartographie (ICA) - *Bertin's ideas are presented as foreword of all good cartography book*

d'exemples pour évaluer chaque avancée de notre travail. Cette intégration technique est en fait un premier pas de formalisation car elle nécessite de définir quelles sont les variables ou composantes qui vont être utilisés par les règles et métriques, c'est-à-dire de délimiter l'espace informationnel. Comme nous l'avons indiqué plus haut, ce travail (Le Pallec et Dupuy-Chessa 2011; 2012; 2013) constitue la section 2 de ce chapitre. Concernant la formulation rigoureuse de métriques, nous avons pour l'instant travaillé principalement sur la distance visuelle. Comme nous le verrons dans la section 3, les résultats obtenus constituent une étude pilote et montrent la démarche que nous avons adoptée pour la suite. Ces résultats n'ont pas été publiés.

5.2 Espace informationnel pour les métriques liées à la PdN

Dans cette section, nous montrons d'abord comment nous avons identifié les variables informationnelles qui sont manipulées dans les critères de la PdN sur deux exemples. Nous indiquons ensuite que la palette d'expressivité proposée par la majorité des outils de méta-modélisation intègre un nombre limité des précédentes variables. Nous montront alors comment ces composantes sont intégrées dans ModX et quelle API nous avons mise en place pour faciliter la définition de métriques et leur évaluation.

5.2.1 Identification des variables

Nous décrivons l'identification des variables sur deux critères qui sont la clarté sémiotique et la discriminabilité perceptuelle, que nous avons déjà abordés dans la section 1. Le lecteur pourra trouver l'étude des autres critères dans (Le Pallec et Dupuy-Chessa 2013).

Note. Les mots clés suivants sont utilisés

stxAbs > syntaxe abstraite ;
elmAbs > élément de la syntaxe abstraite (concept ou association) ;
stxCon > syntaxe concrète ;
elmCon > élément de la syntaxe concrète (représentation graphique d'un concept ou d'une association).

Clarté sémiotique.
Sans chercher à optimiser l'algorithme lié à la clarté sémiotique, on peut définir celle-ci comme suit :

```
fonction ClartéSémiotique (MétaModèle M, notation N)
  renvoie booléen

  // Il y a une et une seule représentation par concept
  // ou association
  Boucle (E dans M)
    si longueur ( élémentConcrets ( E ) ) != 1
      alors renvoyer faux

  // Il y a un et un seul concept / association par
  // représentation
```

```
Boucle (E dans N)
  si longueur ( élémentAbstraits ( E ) ) != 1
    alors renvoyer faux

// Chaque représentation utilisée est suffisamment
// différente des autres pour ne pas prêter à
// confusion

Boucle (R dans N)
  Boucle (R' dans N-{R})
    si distance(R,R') < seuil
      alors renvoyer faux

Renvoyer vrai
```

Sont soulignés ce que nous considérons comme les variables informationnelles. Par exemple, `élémentConcrets` correspond à *quels sont les éléments concrets associés à un élément abstrait donné* et c'est une information primordiale pour la PdN et ses critères. `Distance` est une fonction qui intervient dans le second critère que nous allons étudier maintenant.

Discrimination perceptive.
La discrimination perceptive renvoie à la facilité et la précision avec lesquelles les symboles graphiques peuvent être différenciés des uns des autres. Selon la PdN, il y a 5 aspects à prendre en compte pour cela :

la distance visuelle Plus le nombre de variables visuelles sur lesquelles diffèrent deux symboles est élevé, plus ils sont différenciables... en prenant en compte, bien sûr, les subtilités telles que que nous en avons auparavant énoncées.

la primauté de la forme Par son nombre infinie de valeurs distinguables et par la sémantique que peut véhiculer une forme, cette variable visuelle est à utiliser en priorité pour augmenter la distance visuelle entre les éléments.

le codage redondant Si deux symboles ne se différencient que sur une seule variable visuelle, la distance est faible. Une deuxième variable *de différence* est bien plus efficace.

la mise en avant perceptuelle Si tous les symboles ont la même valeur pour une variable visuelle donnée sauf un symbole, ce dernier sortira du lot et pourra être mis en avant si le lecteur chercher cette valeur en particulier (lecture pré-attentive).

la différentiation textuelle Utiliser des labels textuels pour différencier des symboles est inefficace (ex : préfixe *interface* en UML pour le différencier d'une classe). Il en est de même pour des variations typographiques du texte (ex : souligner le texte pour indiquer une instance de classe). Ces variations ne sont pas à prendre en compte dans le calcul de la distance visuelle.

La distance visuelle est l'aspect le plus important de ce critère : la primauté de la forme et la différenciation textuelle donnent des indices sur comment pourrait être définie la fonction *distance visuelle* et le codage redondant peut être vu comme un type de seuil que la précédente fonction doit dépasser pour indiquer une certaine efficacité. Seule la mise en avant perceptuelle est séparée de la distance visuelle, en ce sens qu'elle est une autre fonction qui se base sur les variables visuelles.
On peut donc affirmer que la discrimination perceptive est constituée de deux fonctions `distanceVisuelle` et `miseEnAvant`. Comme on l'a vu dans la section 1, la première de ces deux fonctions est complexe à calculer mais on peut avoir une vision ultra-simpliste de celle-ci, simplement pour pouvoir identifier les composantes manipulées.

```
function distanceVisuelle ( symbole1, symbole2 )
  renvoie entier

  distance : entier = 0

  // Forme
  si symbole1.forme != symbole2.forme
    alors distance = distance + 1
  // idem pour formeDeLaBordure

  // Couleur
  si symbole1.couleurDeFond != symbole2.couleurDeFond
    alors distance = distance + 1
    // idem pour couleurDeLaBordure et couleurDuTexte

  // Luminuosité
  si symbole1.luminuositéDuFond!=symbole2.luminuositéDuFond
    alors distance = distance + 1
  // idem pour luminuositéDeLaBordure et luminuositéDuTexte

  // Rotation
  si symbole1.rotationForme!=symbole2.rotationForme
    alors distance = distance + 1
  // idem pour rotationTexte

  // Grain : grainDuFond, grainDeLaBordure
  // Taille : largeur, hauteur

  Renvoyer distance
```

Comme nous l'avons dit, cette formule est simpliste : il reste à prendre en compte la primauté de la forme, la différenciation textuelle et surtout à étudier de manière approfondie la sémiologie graphique pour quantifier les relations entre variables visuelles. Mais ceci ne rajoute pas de composantes informationnelles. La fonction `miseEnAvant` est elle aussi basée sur les composants précédentes soulignées : il n'est pas nécessaire de décrire ici (une version simpliste de) l'algorithme de cette fonction.

L'ensemble des composantes informationnelles.
Nous avons analysé les autres critères de la PdN dans (Le Pallec et Dupuy-Chessa 2011; 2012; 2013). Le résultat de cette analyse est synthétisé dans le tableau suivant : à gauche, se trouvent les composantes identifiées, en haut les critères de la PdN, et les croix correspondent à la présence des composantes au sein des critères.

	Clarté sémiotique	Discrimination perceptive	Transparence sémantique	Gestion de la complexité	Intégration cognitive	Expressivité visuelle	Double codage	Économie graphique	Adaptation cognitive
élémentsConcrets	X							X	
élémentsAbstraits	X								
position		X				X		X	
taille		X				X		X	
luminuosité (fond, bordure, texte)		X				X		X	
texture (fond, bordure, texte)		X				X		X	
couleur (fond, bordure, texte)		X				X		X	
orientation (fond, texte)		X				X		X	
forme (fond, bordure)		X				X		X	
mécanismeComplexité				X					
représentationContexte					X				
annotationTextuelle							X		
supportDiagrammePrivilégié									X

TABLE 5.1 – *Correspondances entre critères de la PdN et fonctions élémentaires*

5.2.2 Plateforme de prototypage et d'expérimentation

Afin de parvenir à définir des métriques et à les évaluer sur de nombreuses notations visuelles, il faut un outil 1) qui nous permettent de pouvoir facilement définir et redéfinir (approche itérative) des métriques qui accèdent aux valeurs des précédentes composantes pour toute notation à évaluer 2) qui intègre le plus possible des composantes citées précédemment.

API pour accéder aux composants d'une syntaxe concrète

Les éditeurs de méta-modèles (EMM) n'intègrent pas de facilités liées aux métriques pour les syntaxes concrètes. Notre idée n'est pas simplement de tester le résultat de nos réflexions sur les métriques, mais d'avoir un outil qui pourraient permettre aux chercheurs intéressés par cette problématique d'avoir un outil où la création et l'évaluation de métriques seraient simples... suffisamment pour catalyser les recherches sur ce thème. Au moment de ce travail, et concernant les EMM, la quantité de travail pour implémenter (et évaluer) des métriques variait selon les outils. Tout d'abord, beaucoup d'EMM se focalisent seulement sur les notations textuelles ou à base de formulaires comme MPS (JetBrains 2013), Spoofax (Visser 2013), Whole platform(Solmi 2012) ou Rascal(Klint 2010). Nous pouvons donc dire que ces EMM ne sont pas de bons candidats pour le prototypage de métriques sur les syntaxes concrètes. Ensuite, écrire du code ou des règles

qui calculent automatiquement des métriques implique une API pour accéder aux spécifications de la notation visuelle. Une telle API est rarement proposée directement. Les outils basés sur Eclipse comme Obeo Designer (Obeo 2013) ou Eugenia (Foundation 2013) n'en proposent pas. Toutefois, comme la spécification de leurs syntaxes concrètes sont des modèles EMF, il est possible d'utiliser des plug-ins Eclipse comme EOL (Foundation 2012) ou QVTo (Foundation 2008) pour naviguer de manière programmatique au sein de ces modèles. Cela nécessite donc d'installer des plug-ins supplémentaires et de les connecter convenablement (sans oublier les problèmes de versions des plug-ins et d'Eclipse). De plus, des fonctions comme `representationVisuelle` ou `annotationTextuelle` ne seront pas présentes et il faudra les implémenter. Des EMM comme MetaEdit+ (MetaCase 2013), ModX ou xOWL (Wouters 2012) fournissent eux un accès direct aux spécifications des syntaxes concrètes. Enfin, il est préférable de proposer une intégration complète de la notion de métrique : un endroit pour écrire du script ou des règles pour définir des métriques et aussi un endroit où le résultat de ce script ou de ces règles sera affiché, un endroit bien connu et facilement accessible, afin de guider la conception de syntaxes concrètes. À notre connaissance, il n'y a pas de tel EMM.
Comme les outils existants ne proposaient rien de plus que ModX, et que nous avions une meilleure connaissance de celui-ci, nous sommes partis sur celui-ci.
Il restait à voir si les types de valeur des composantes informationnelles étaient "compatibles" avec le calcul de métriques.

Types de valeur des composantes informationnelles

Les composantes informationnelles *élémentsConcrets*, *élémentsAbstraits* renvoient aux éléments constituant les syntaxes abstraites et concrètes, et leur type dépend du format utilisé (EMF, MOF, OWL...).
Les composantes *mécanismeComplexité*, *représentationContexte* et *annotationTextuelle* renvoient aux mécanismes de gestion de complexité, de représentation du contexte et de labélisation proposés par les outils. Il s'agit donc à chaque fois d'une liste de valeurs ordinales, où chaque valeur réfère un mécanisme proposé. La seule comparaison possible entre les outils est que plus la liste est longue, meilleure est la capacité d'expression proposée au concepteur.
La composante *supportDiagrammePrivilégié* est elle aussi une liste de valeurs ordinales dont les valeurs sont, d'après la PdN, écran, feuille de papier et tableau.
C'est sur les variables visuelles que les types de valeurs sont plus problématiques. Le tableau suivant montre celles qui sont proposées dans les EMM et leur type de valeur. La colonne la plus à droite montre ce que nous avons implémenté dans ModX après notre analyse.

Part	variables	Eugenia	MetaEdit+	Obeo Designer	xOWL	ModX
Main Shape	Color	RGB def	RGB def	Predefined list	RGB def	primary colors
	Brightness	No	HSV def	No	No	percentage
	Size	l x h (px)	l x h (px)	l x h (px)	l x h (px)	l x h (px)
	Shape	Rectangle, Ellipse, Polygon, svg	Any shape	Rectangle, Lozenge, Triangle, Ellipse	Rectangle, Ellipse, Polygon	Rectangle, Ellipse, Triangle...
	Grain	No	No	No	No	No
	Orientation	No	No	No	No	Yes
Border	Color	RGB def	Yes	Predefined list	RGB def	primary colors
	Brightness	No	HSV def	No	No	percentage
	Size	h (px)	l (1 to 10)	h (px)	h (px)	h (px)
	Shape	Solid, Dash, Dot	7 combinations	No	No	Dash, Dot, Double Line, Zigzag, Handwritte
	Grain	No	No	No	No	Yes
	orientation	No	No	No	No	No
Text	Color	No	RGB def	Predefined list	RGB def	primary colors
	Brightness	No	HSV def	No	No	percentage
	Size	No	h (pt)	h (pt)	h (pt)	No
	Grain	No	No	No	No	No
	Orientation	No	No	No	No	No

TABLE 5.2 – *Support des variables visuelles dans les outils de méta-modélisation*

Pour la couleur (fond, bordure et texte), les outils proposent soit un triplet Rouge-Vert-Bleu, soit, pour Obeo Designer, une liste de couleurs prédéfinis (qui ne sont pas les couleurs primaires). Les types possibles sont donc un triplet d'entiers de 0 à 255 ou une liste de valeurs ordinales.
Pour la luminosité (fond, bordure et texte), le seul type de valeur possible est ici Teinte-Saturation-Valeur où la valeur est la luminosité. La valeur HSV est d'ailleurs une autre façon de définir une couleur, on choisit la teinte, la saturation et la luminosité. Ces 3 variables sont des valeurs entières comprises entre 0 et 255.
La taille est exprimée en pixel ou en point.
La forme est généralement une liste de formes prédéfinies, soit une liste de valeurs ordinales. Seul Meta-Edit+ propose un éditeur de forme : le type de valeur est alors un tableau de symboles géométriques choisis (forme, couleur, luminosité...).
Le grain n'est pas proposé, tout comme l'orientation.

Dans l'ensemble, mis à part MetaEdit+, les outils proposent peu de variations visuelles et sont donc peu performants pour permettre au concepteur d'avoir un bon "score" pour la discrimination perceptuelle. Bien sûr, il est toujours possible d'utiliser des images (tous le proposent). Mais, dans ce

cas, il ne sera pas possible de connaître la valeur pour chaque variable visuelle.
Ensuite, les types de valeur pour la couleur et la luminosité ne sont pas adaptés si l'on se réfère à la sémiologie graphique : pour Bertin, on choisit parmi la liste des couleurs primaires et on fixe une luminosité. Ceci permet de savoir si deux symboles, ayant des couleurs différentes sont bien sur la même luminosité ou non.
Pour ModX, nous avons essayé de respecter au mieux les principes de la sémiologie graphique. Le choix d'une couleur se fait parmi la liste des couleurs primaires. La luminosité est un pourcentage de 10 à 90% mais le concepteur ne peut choisir que des dizaines : car le nombre maximal de paliers visibles est de 9. La rotation est disponible et la valeur est l'angle choisi. Pour la forme, de nombreuses figures géométriques sont permises ainsi que de nombreux types de traits. Seul le grain, qui renvoie au zoom effectué sur un motif choisi, n'est pas proposé par ModX.

5.2.3 Implémentation dans ModX

Cadre général et point d'accès

Nous avons implémenté un accès programmatique à ces composantes dans ModX afin de disposer d'un outil permettant l'étude de métriques sur les syntaxes concrètes. Depuis plusieurs années, ModX proposait une API Javascript qui permet d'accéder aux différents éléments qu'il manipule (cf. chapitre 2). Je me suis basé sur cet API pour intégrer le calcul de métriques. Le point d'entrée pour l'ajout de métriques consiste en deux fonctions : une fonction de calcul de la qualité des méta-modèles et une pour la qualité des modèles. Ces fonctions peuvent être invoquées quand on édite respectivement un méta-modèle ou un modèle. La figure ci-dessous montre l'affichage du résultat pour les deux métriques qui sont prédéfinies et intégrées dans ces fonctions sur l'exemple de la notation pour les cas d'utilisations UML.

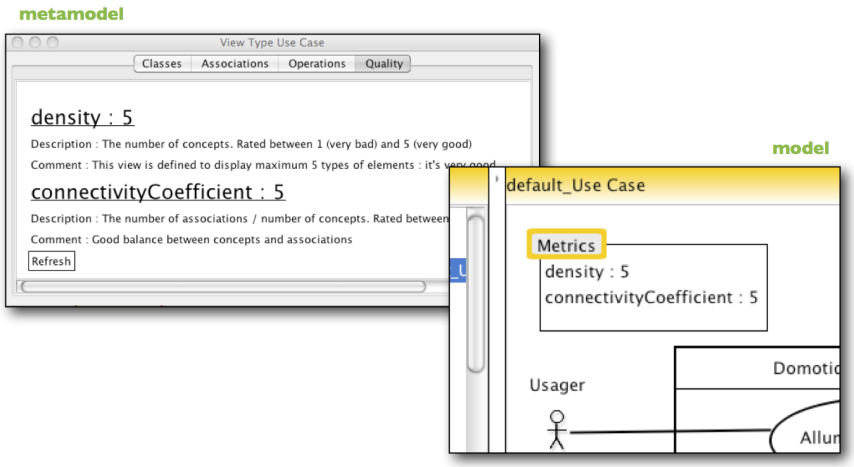


FIGURE 5.2 – *Affichage des métriques dans ModX*

Si on souhaite ajouter une métrique, il faut modifier le code d'une des deux précédentes fonctions[4] afin d'y intégrer le code de la métrique. La modification de ces fonctions se fait via un éditeur intégré à ModX. Le script suivant montre le calcul (simpliste) de la distance visuelle dans ModX et son utilisation pour la discrimination perceptuelle.

```
1  visualDistance = {
2    between2classes : function ( element1, element2) {
3      var difference = 0;
4      // les deux utilisent une forme geometrique
5      if (element1.formMode==element2.formMode &&
6          element1.formMode=concreteSyntax.SHAPE_MODE) {
7        difference+=(element1.shape!=element2.shape ? 1 : 0 );
8        difference+=(!element1.borderColor.equals(element2.borderColor) ? 1 : 0 );
9        difference+=(element1.borderType!=element2.borderType ? 1 : 0 );
10       if (element1.resizeMode==element2.resizeMode &&
11         element1.resizeMode=concreteSyntax.NORESIZE)
12         difference+=(element1.width!=element2.width || element1.height!=element2
               .height ? 1 : 0 );
13     } else difference = 2;
14     return difference;
15   },
16   between2associations : function (element1, element2) {
17     var difference=0;
18     // les deux utilisent des traits graphiques
19     if (element1.style==element2.style && element1.style=concreteSyntax.
           LINK_MODE) {
20       difference+=(element1.leftEndStyle!=element2.leftEndStyle ? 1 : 0 );
21       difference+=(element1.rightEndStyle!=element2.rightEndStyle ? 1 : 0 );
22       difference+=(element1.stroke!=element2.stroke ? 1 : 0 );
23     } else difference = 2;
24     return difference;
25   },
26   computeForAll : function (concreteSyntax) {
27     this.score = 0;
28     this.comments = "";
29     var stxCon_classes=concreteSyntax.__classes;
30     var stxCon_associations=concreteSyntax.__associations;
31
32     for (var left=0;left<stxCon_classes.length-1;left++) {
33       for (var right=left+1;right<stxCon_classes.length;right++) {
34         var delta=differenceVisualVariables(left,right);
35         if (delta<2) {
36           this.comments+=" ("+left.source.__name+" - "+right.source.__name+")";
37           this.score++;
38         }
39       }
40     }
41     //Idem pour les associations
42   }
43 }
44 visualDistance.computeForAll(concreteSyntax);
45 addMetric("Discrimination perceptive", this.score, this.comments, "Les couples
       indiquees sont trop proches visuellement");
```

FIGURE 5.3 – *Script pour la distance visuelle dans ModX*

Les deux premières lignes permettent d'avoir deux variables qui référencent la liste des éléments concrets pour les concepts et celle pour les associations. Pour chacune des paires, la fonction `computeForAll` (lignes 26-42) va calculer une distance visuelle. Si elle est supérieure à un 1 - dans l'idée qu'il y a plus d'une variable visuelle où les valeurs diffèrent - alors la distance est considérée comme suffisante. Sinon (lignes 35-38), le couple visuellement trop proche sera affiché. Pour les classes (lignes 2-15), la distance ne se calcule que si les deux éléments concrets utilisent des formes géométriques (lignes 5-6). Si au moins l'un des deux utilise une image, alors la distance vaut 2 (ligne 13) donc une distance suffisamment grande. Dans le cas contraire, on regarde si la forme géométrique choisie (ligne 7), la couleur de remplissage (ligne 8), la bordure (trame, ligne 9) ou la taille

4. selon que l'on vise une métrique pour la qualité des modèles ou des méta-modèles

(si fixée, lignes 10-12) diffèrent. Pour les associations, la distance ne se calcule qu'entre éléments concrets ayant choisi des liens/traits graphiques (ligne 19). Dans ce cas, on regarde les différences entre les formes utilisées pour chacune des extrémités (lignes 20-21) et la trame du trait (ligne 22). La dernière ligne est importante car elle montre comment *ajouter* une métrique dans l'affichage. C'est la fonction `addMetric` intégrée à ModX qui s'en charge. Elle prend en paramètre le nom à afficher, la valeur, un commentaire associé à la valeur et un descriptif général de la métrique.

5.2.4 Conclusion

Dans ce travail, nous avons montré l'intérêt à opérationnaliser la PdN mais aussi la difficulté sous-jacente à cette tâche. Nous avons délimité l'espace informationnel des critères et règles possibles issus de la PdN en identifiant les variables utilisées dans les critères. Cette identification a permis de montrer que les outils de méta-modélisation actuels supportaient peu ces variables ou de façon incorrecte et donc limitaient le travail sur les métriques. Nous avons intégré la majorité de ces variables dans ModX. Dans ce dernier, nous avons aussi défini un emplacement où le concepteur peut définir (en Javascript) des scripts calculant des métriques grâce notamment à un accès aux précédentes variables pour chaque élément d'un méta-modèle/modèle.

5.3 Études empiriques

Sophie Dupuy-Chessa, Nicolas Genon et moi avons souhaité effectuer une étude pilote sur l'utilisation des variables visuelles dans les diagrammes logiciels afin de commencer notre travail par la discrimination perceptuelle. La sémiologie graphique fournit déjà un grand nombre d'éléments de structuration, ce qui nous donne suffisamment d'informations pour avoir une ébauche de la "formule" concernant la distance visuelle (au coeur de la discrimination perceptuelle). Notre travail consiste donc ici à trouver la valeur des facteurs de cette formule. Toutefois, même si on se limite aux représentations généralement utilisées en génie logiciel, le nombre de variations pour chaque variable visuelle et leur combinaison implique un ensemble des possibles suffisamment grand pour rendre notre tâche ardue. C'est pour cette raison que nous avons voulu effectuer une expérimentation test pour identifier de manière pragmatique les difficultés de ce travail : la charge de travail nécessaire, la facilité/difficulté à trouver des participants, détecter des possibles variables parasites pour les calculs statistiques... Nadine Mandran, qui est une ergonome travaillant dans l'équipe IIHM, nous a accompagnés dans notre démarche.

5.3.1 Objectif de l'étude pilote

Notre étude s'est focalisée sur "comment les variations introduites dans une notation visuelle influencent la clarté des diagrammes qui peuvent en être instanciés". Nous choisissons une notation visuelle - dont les choix graphiques sont classiques pour le génie logiciel - et nous faisons évoluer une par une les variables visuelles. Nous demandons à des partici-

pants d'évaluer la clarté pour chacune des variations. Nous pouvons ainsi mesurer, au regard des résultats, le facteur d'importance de chaque variable visuelle. Comme nous ne souhaitions pas cloisonner nos résultats à une tâche particulière (chercher les éléments d'un certain type dans un diagramme, retrouver un élément en particulier...), nous avons utilisé le concept de clarté : est-ce qu'à première vue, le participant trouve le diagramme "clair" ? Nous restons donc à un niveau très perceptif sans déclencher un travail cognitif chez le participant.

5.3.2 Matériel utilisé

Nous avons utilisé un langage de modélisation avec une seule notation visuelle. Le langage était spécifique au e-learning et comprenait les concepts suivants : Étudiant, Enseignant, Cours, Séance, Document, Page Web, Wiki et Forum. Sur la figure ci-dessous, on peut aussi voir les associations correspondantes.

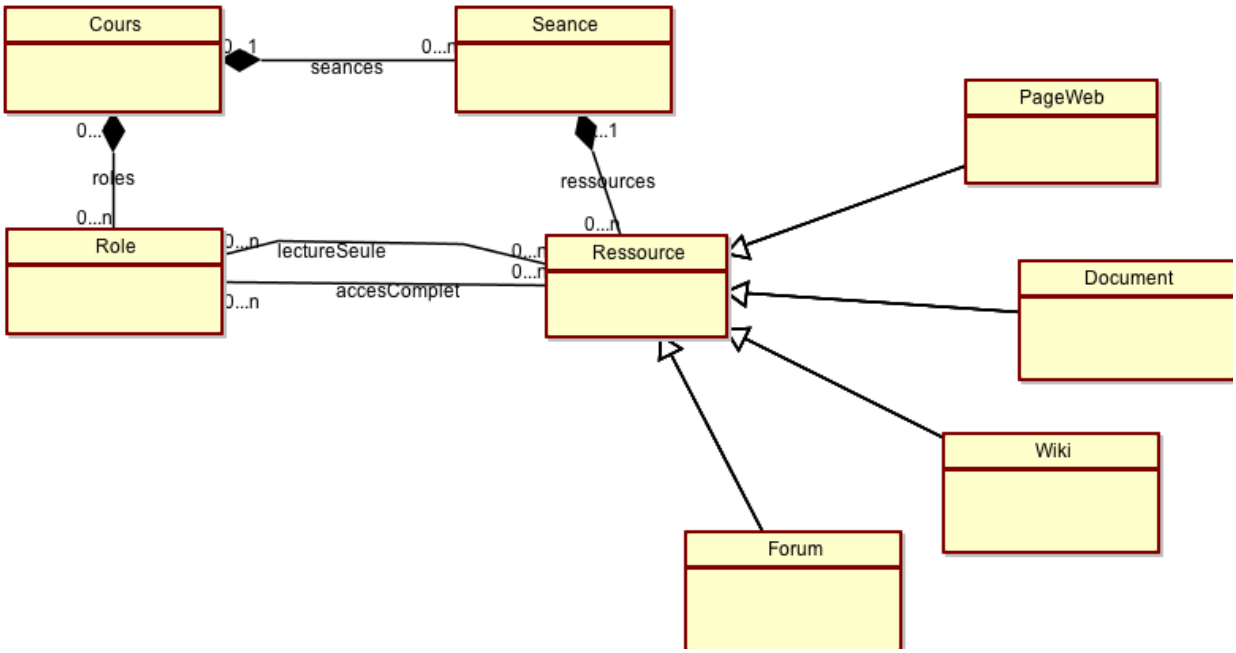


FIGURE 5.4 – *Méta-Modèle du langage utilisé pour l'expérimentation*

Nous avons instancié et utilisé un seul modèle/diagramme. Il s'agit d'un cours d'économie générale.

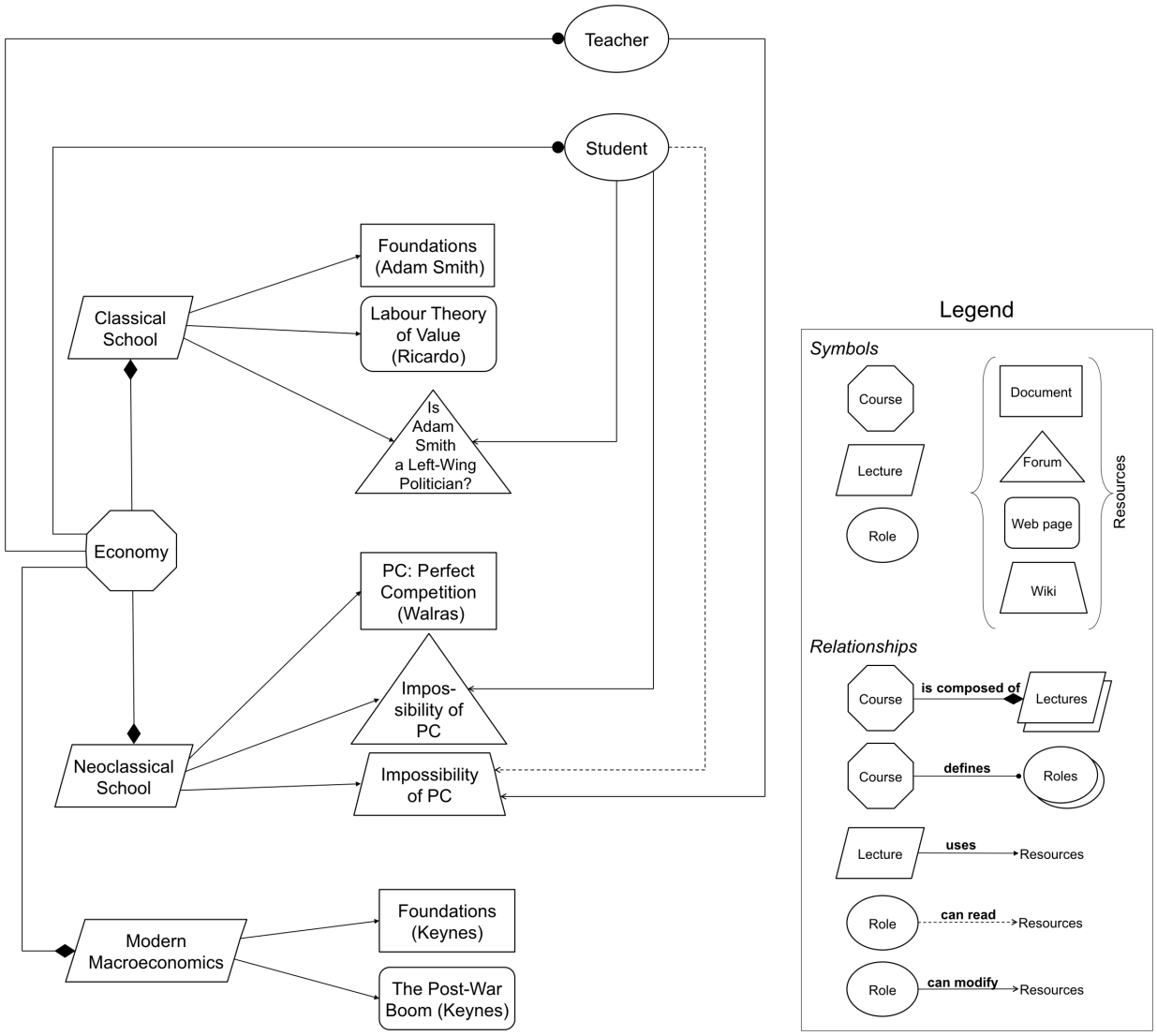


FIGURE 5.5 – *Diagramme utilisé*

Les choix concernant la notation et le diagramme proviennent du besoin de neutraliser les critères de la PdN autre que la discriminabilité perceptuell, c'est-à-dire qu'il ne devait pas y avoir d'autres facteurs de variation/d'impact que les variables visuelles. Par exemple, pour la clarté sémiotique, il ne fallait pas que celle-ci varie. Pour l'instant, ce critère a comme valeur possible : correct ou incorrect, sachant qu'il y a plusieurs niveaux d'incorrection (ex : un concept a deux symboles graphiques, un symbole est utilisé par deux ou trois concepts...). Pour éviter une variation d'incorrection, nous avons donc affecter un symbole et un seul à chaque concept et tous les symboles étaient réellement différents. Ainsi quelque soit le diagramme, la clarté sémiotique était toujours à correct. Pour éviter des variations sur la transparence sémantique, nous avons veillé à ce qu'aucun symbole ne représente visuellement une quelconque sémantique en choisissant des formes géométriques simples. Nous avons aussi éviter d'associer le rectangle à un concept de classification qui est un classique en

modélisation logicielle (UML, MCD). Ainsi le score de la transparence sémantique, quelque soit le diagramme, était toujours "opaque". Nous avons adopté la même approche pour les six autres critères.
Concernant les variables visuelles, nous n'avons fait varier que la forme, la couleur du fond, la luminosité du fond et la forme de la bordure : il s'agit des variations usuelles en modélisation logicielle, et nous avons souhaité rester dans l'espace de conception graphique classique de ce domaine. Pour la forme, nous avons choisi encore une fois les valeurs classiques du domaine : Ellipse (Cas d'Utilisation UML), Rectangle (Attribut d'ER), Rectangle arrondi (Action/Activité d'UML), Triangle (Entité d'ER), Parallélogramme (Entrée/Sortie des Flowchart) et Trapèze (Opération manuelle de Dataflow), Octogone (inspiré de PréCondition de Flowchart). Pour les associations, nous avons choisi les flèches et losanges habituelles (UML, ER...). Seule l'extrémité avec un cercle plein est inhabituelle, mais elle a l'avantage d'être fort différente du reste. Comme les Roles et les Ressources sont connectables par deux associations différentes, nous avons souhaité garder une proximité visuelle entre leur représentation visuelle (la forme de l'extrémité) et les faire varier sur autre chose (la forme du trait). Les couleurs ont été choisies pour avoir toute la même luminosité : ainsi lorsque la couleur variera, il n'y aura pas d'autre variation. De manière analogue, une seule couleur a été choisie pour la variation de luminosité : le noir (ou le blanc). Enfin, pour la variation de la forme de la bordure, nous avons du mixer les valeurs usuelles car elles sont peu nombreuses : pointillé (Template d'UML), ligne épaisse (Appel d'Opération de BPMN), double lignes (Entité faible d'ER) et ligne simple. Nous n'avons pas considéré les associations comme ayant une bordure.
Les différentes variations sont montrées dans la figure suivante.

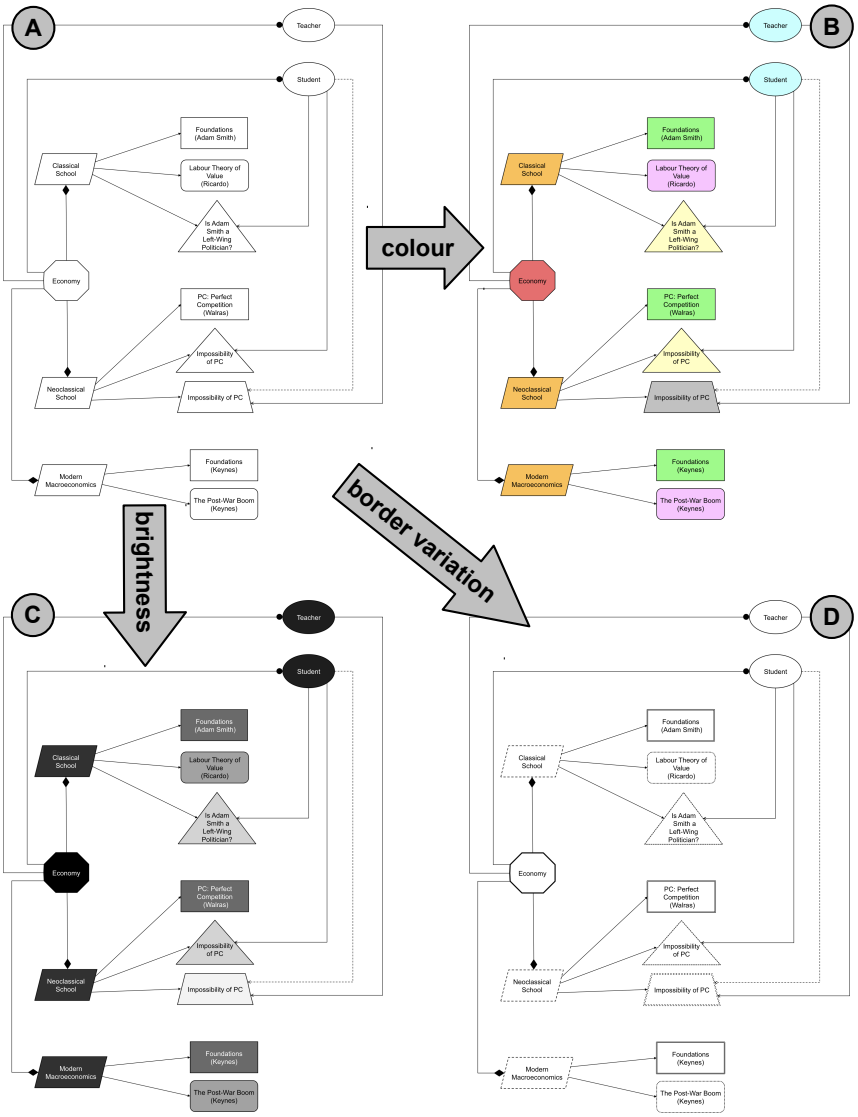


FIGURE 5.6 – *Variations visuelles sur le diagramme*

5.3.3 Protocole d'expérimentation et résultats

Protocole

Il y a deux façons d'évaluer comment les variables visuelles impactent l'efficacité perceptuelle des diagrammes. La première est issue de la psychologie où les concepts sont étudiés séparément. La deuxième repose sur un point de vue systémique global où les études sociologiques sont utilisées. La première approche est à grain fin mais le nombre nécessaire de cas d'études est exponentiel. La deuxième approche est à gros grain et itérative : elles permet d'obtenir rapidement des résultats qui peuvent être raffinés. Nous avons adopté la seconde approche pour son rapport bénéfices/coûts, avec un choix particulier pour une méthode utilisée pour les études de marché (Dutka 1995), méthode qui mesure la satisfaction produite par un produit ou un service. Un des objectifs de ce type de sondage est de déterminer quelle caractéristique influence le plus la satisfaction de

l'utilisateur/client. À cette fin, les participants notent leur niveau de satisfaction du produit ou service présenté et notent ensuite chacune de ses caractéristiques. Par exemple, si l'étude concerne un service d'enquête, la satisfaction sera globalement évaluée puis ensuite ses caractéristiques : la rapidité de réponse, la pertinence des informations, la gentillesse des participants...

Pour notre expérimentation test, nous avons montré le diagramme *Cours d'Économie* avec différentes variations (équivalentes des "différents produits") et nous avons demandé aux participants leur satisfaction générale, ici la clarté du diagramme, puis une note de satisfaction pour chacune de ses caractéristiques. Cette dernière notation consistait à demander s'ils percevaient une variation de couleurs, formes, bordure, etc. entre les éléments de diagrammes. La note de satisfaction générale permettait d'évaluer si une variation de couleur ou de bordure améliorait la clarté (ou non) et dans quelle mesure. La note pour chacune des caractéristiques permet de vérifier si la variation effectuée était bien perçue et si les participants ne percevaient que celle-là (exceptée la forme qui était commune à tous les diagrammes).

Nous avons conçu le questionnaire en suivant ce principe : quatre diagrammes, pour chacun une note de clarté et une évaluation de la perception des variations possibles. Nous avons demandé à la fin, de classer les diagrammes en ordre de préférence, quels étaient le niveau d'expertise en modélisation logiciel et le sexe du participant.

Résultats

Perception des variations. Nous avons vérifié si les différences introduites entre le diagramme basique et les trois variations ont été identifiées par les participants. Le tableau montre la moyenne des différences perçues pour la forme, la couleur, la luminosité et la bordure (colonnes) pour chaque version du diagramme (lignes). On remarque que la variable visuelle utilisée comme vecteur de variation est identifiée comme la caractéristique la plus discriminante, excepté pour la luminosité (où la forme "l'emporte").

		Perception des variations visuelles					Note Clarté	
		Forme	Couleur	Taille	Bordure	Lumino.	Moy.	ET
Diagram	Basique	4.6	0.5	2	2.1	1.3	4.64	1.684
	Couleur	5	8.2	2.1	2.2	4.2	6.69	1.544
	Luminosité	4.5	2.6	1.8	2	3.5	4.39	1.676
	Bordure	4.6	0.9	1.9	6.5	1.6	4.74	1.482

TABLE 5.3 – *Perception des variations visuelles et clarté*

Clarté de chaque variante du diagramme. Le précédent tableau montre aussi les scores de clarté obtenus pour le diagramme de base et ses variantes. La clarté du premier est de moyenne 4,64. Les moyennes pour la luminosité et la bordure sont très proches de cette valeur (respectivement 4,34 et 4,74). Il semble même que la luminosité baisse le score de clarté (4,34 contre 4,64). Par contre, la variation de couleur augmente de 44% le score de clarté. Un test de Wilcox avec correction de Bonferroni montre que seule la couleur est significativement différente des autres. Ceci signifie que, dans notre contexte, seule la couleur a un impact sur la clarté de diagrammes utilisant déjà la variation de formes : les autres variations ne

semblent rien apporter. Pourtant, la luminosité est une variable visuelle phare de la sémiologie graphique pour les raisons expliquées précédemment. L'avons-nous mal utilisé : trop de paliers ? Est-elle plus intéressante pour l'expressivité visuelle (hiérarchisation) que pour la discriminabilité perceptuelle ? Des questions intéressantes et à approfondir par la suite.

Niveaux d'expertise. Le niveau d'expertise renseigné par chaque participant avait quatre valeurs possibles : faible, moyen, élevé, très élevé. Nous avons regardé, avec des tests de corrélation de Spearman, s'il y a des corrélations entre niveau d'expertise du participant et sa perception des différences visuelles ou ses notes de clarté. Le résultat est que le niveau d'expertise ne semble pas augmenter ou diminuer les capacités à percevoir les différences visuelles appliquées à notre diagramme. Pour la note de clarté, seule la luminosité semble présenter ce type de corrélation : plus le participant aurait une grande expertise en modélisation, plus il trouverait les différences de luminosité comme augmentant la clarté. Ce sont des observations intéressantes car elle montre une piste à approfondir : le processus cognitif dédié à la modélisation logicielle est plus évolué chez les experts, mais ces évolutions ou ces extensions ne semblent pas réagir par une augmentation de la discrimination visuelle. La raison est-elle que ces variations sont rares en modélisation logicielle et donc que les "experts" n'ont pas fait évoluer les capacités cognitives dans ce sens ? Ou est-ce qu'un diagramme "plus perceptible" n'apporte rien à la cognition comme semblent l'affirmer (Diemand-Yauman et al. 2011, Carpenter et al. 2013) où les mécanismes cognitifs, par un phénomène d'auto-régulation, ralentissent plus l'efficacité perceptuelle d'une représentation visuelle augmente ?

Impact de la perception des variations visuelles sur le clarté. Nous avons vu que le diagramme coloré était perçu comme plus "clair". Toutefois dans ce même diagramme, d'autres variations sont perçues même si elles le sont moins que la couleur : la forme et la luminosité. Donc une grande variation de couleur influence la clarté perçue, mais quelle est l'influence des autres variations visuelles qui sont elles aussi perçues mais dans une moindre mesure ? Pour cela, nous avons utilisé un modèle de régression linéaire, qui va consister à partir du principe que

$$\begin{aligned} clarté = constante + \\ \Delta couleur \times fa_{couleur} + \\ \Delta luminosité \times fa_{luminosité} + \\ \Delta forme \times fa_{forme} + \\ \Delta bordure \times fa_{bordure} \end{aligned}$$

où $\Delta variable$ est le différence perçue pour cette variable visuelle
et où fa_{valeur} est le possible facteur d'impact de cette différence sur la note de clarté.

Avec cette méthode, nous allons pouvoir vérifier si chaque delta rentre en jeu dans la clarté (facteur associé différent de zéro) et si c'est le cas, quelle est sa valeur, c'est-à-dire son impact (le facteur).
La variable dépendante est la note de clarté globale (sur une échelle de 10). Les variables indépendantes sont les différences perçues par les participants sur la forme, la couleur, la luminosité et la bordure (encore sur

une échelle de 10). Pour chaque diagramme, nous avons étudié l'évaluation de ces variables et leur corrélation. Le tableau ci-après résumé les résultats. La première colonne identifie la version du diagramme impliqué. La seconde colonne la significativité du modèle linéaire : il indique si, statistiquement et sur la base des données récoltées, il existe un modèle linéaire (clarté = constant + ...). Les autres colonnes montre les facteurs de la forme, couleur, luminosité et bordure.

Version Diagramme	Significativité	Constante	Effet Forme	Effet Couleur	Effet . Lumino.	Effet Bordure
Forme	F=33.7 ***	2.80 ***	0.39 ***			
Couleur (model 1)	F=20.1 ***	5.17 ***		0.3 ***		
Couleur (model 2)	F=17.88 ***	3.19 ***	0.24 ***	0.28 ***		
Lumino. (model 1)	F=58.23 ***	2.89 ***			0.43 ***	
Lumino. (model 2)	F=40.69 ***	2.18 ***	0.19 ***		0.38 ***	
Bordure	F=16.02 ***	3.52 ***	0.26 ***			

*** : significativité<1% - ** : < 5% - * : < 10%

TABLE 5.4 – *Modèle linéaire pour la clarté*

Variation de forme seule Pour le diagramme basique, la clarté est $2,8 + 0,39 \times \Delta forme$. Le facteur de forme étant positif, la constante indique une note minimale de 2,8. Ensuite le facteur associé à la perception des variations de forme est 0,39. Comme la perception est notée sur 10, sur le principe si le participant voit très fortement les différences de forme, la note de clarté peut monter à 6,7, soit un impact important sur celle-ci.

Variation de forme et de couleur Le modèle le plus significatif est $5,17 + 0,3 \times \Delta couleur$. Le facteur de couleur est important (0,3) mais surtout la constante est élevée (92% de plus que précédemment) : quelque soit la perception des différences de couleurs, les participants trouvent le diagramme assez clair. Le second modèle, moins significatif, laisse plus de place à la perception des variations de forme, ce qui semble être un modèle plus logique car la forme est un facteur discriminant important. La différence entre ces deux modèles mériterait d'être étudié plus en profondeur avec une expérimentation adaptée.

Variation de forme et de luminosité Comme pour le précédent diagramme, deux modèles : un plus significatif avec un seul impact sur la clarté provenant de la "nouvelle" variation, et un deuxième où la constante diminue et la variation de forme apparaît. Donc un impact possiblement important de la perception de la variation de luminosité mais avec, toutefois, une clarté en deçà des autres diagrammes.

Variation de forme et de bordure La clarté est ici de $3,52 + 0,26 \times \Delta forme$. C'est-à-dire que la note peut aller de 3,52 à 6,12. On reste dans les valeurs du diagramme basique, mais plus resserrées : on peut dire que les participants trouvent le diagramme plus clair en général mais qu'ils perçoivent moins les différences de forme.

5.3.4 Premières recommendations

Les recommendations que l'on pourrait extraire de cette première expérimentation sont les suivantes :

- Utiliser la couleur sur les éléments qui doivent être immédiatement perçus dans un diagramme. La couleur est la variable où l'efficacité perceptuelle est la plus haute.
- Ne pas utiliser la variation de bordure pour rendre les éléments discriminants. De nos résultats, il n'est pas clair si cette variation a un impact ou non sur la clarté.
- Renforcer la discrimination entre éléments en introduisant la variation de forme ou de luminosité. Ces deux variations ont un impact positif sur la clarté d'un diagramme, mais elles doivent être utilisées comme un élément additionnel.

Toutefois, ces recommendations renvoient à des affirmations que l'on peut d'ores et déjà trouvé en sémiologie graphique. Ce n'était de toute façon pas l'objectif ici. Comme nous l'avons dit précédemment, c'est une étude préliminaire pour un objectif à plus long terme qui est de quantifier ce type de recommendations, c'est-à-dire de rationnaliser l'espace de conception en modélisation logicielle afin de pouvoir répondre à des questions comme : la couleur est combien de fois plus impactante pour la clarté que les autres variables visuelles ? Dans quelle proportion, ce multiplicateur change-t-il en fonction des variations des autres variables visuelles ? Comment ces "facteurs" évoluent-ils en fonction du nombre d'éléments d'un diagramme et de leur emplacement et/ou proximité ? Ce type de questions implique une multiplication des études de cas (même en adoptant une approche systémique globale) importante et donc une grande complexité dans la spécification des protocoles d'expérimentation afin que ceux-ci soient réalisables (d'un point de vue ressources humaines). Mais l'apport des résultats possibles mérite de prendre le risque d'une telle entreprise.

5.4 Conclusion

Comme je l'ai indiqué au début de ce chapitre, ces travaux sur l'opérationnalisation de la physique des notations sont le résultat d'une activité naissante autour des notations visuelles. Un effort sera nécessaire pour que la suite de cette activité apporte des contributions importantes à cette problématique. Car celle-ci intéresse de nombreuses personnes au niveau international comme en témoignent les nombreux travaux cités tout au long de ce document, et au niveau national comme le montre les récents groupes de travaux *expert-user modelling* et VUExCoSSI[5] respectivement financés par le GDR GPL et Inforsid. Le premier, que j'ai piloté avec Sophie Dupuy-Chessa, était très orienté IDM, le second, piloté par David Bihanic (MCF en science du design à Valenciennes) et Thomas Polacsek (ONERA - Toulouse), avait une composante "Sciences Humaines" plus forte (psychologie cognitive, géomatique et design). Chaque action spécifique a d'ailleurs donné lieu à une publication commune : (Dupuy-Chessa et al. 2014) pour expert-user modelling et (Bihanic et al. 2013) pour VUExCoSSI[6]. Ce dynamisme national est motivant et me conforte dans l'idée de

5. Vision Utilisateur & Expression de la Complexité au sein des SI

6. Avec David Bihanic, Sophie Dupuy-Chessa et Thomas Polacsek, nous essayons actuellement de regrouper ces deux actions en une seule (nous avons soumis un défi com-

continuer dans cette direction. D'ailleurs, l'effort nécessaire évoqué plus haut pourrait venir de ces collaborations et d'une thèse sur cette thématique, en collaboration avec et financée par le CEA-Saclay, qui devrait débuter en Décembre 2014. Enfin, comme nous le verrons dans la conclusion qui suit, ce travail est fondamental dans Carbon, équipe que je viens de créer avec Cédric Dumoulin et Jean-Claude Tarby.

mun aux journées GDR-GPL 2014). Nous visons aussi bien sûr une ouverture de notre collaboration à l'international.

Conclusion générale

Au début du document, j'ai indiqué que le MDA avait été l'impulsion à la base de la création de l'IDM, une démarche de conception logicielle qui prône des chaînes de conception/développement où les modèles sont les principaux ingrédients. Toutefois, les travaux sur le rôle des modèles en Ingénierie Logicielle (IL) et leurs apports possibles avaient déjà bénéficié d'un regain d'intérêt au milieu des années 90 lors de la création d'UML et du MOF, tout deux standardisés par l'OMG en 1997. L'architecture MDA a en fait renforcé un pan de ces recherches qui visait à rendre les modèles plus productifs. Ce courant portait et porte encore essentiellement ses efforts sur les opérateurs de composition / transformation de modèles, de création de langages spécifiques et de génération de code. **Mes travaux** s'inscrivent dans ce courant mais se **sont concentrés sur l'utilisabilité** de ces opérateurs et sur l'activité de modélisation telle qu'elle a lieu **en IDM**. Ce focus s'est renforcé au fil des années de par la prise de conscience que cet aspect lié aux facteurs humains semblait être un des obstacles à une adoption massive de l'IDM.

J'ai étudié 3 aspects de l'IDM sous cet angle.

Le **premier d'entre eux est le mapping vertical** qui est généralement dédié au déplacement dans la dimension de modélisation *abstraction*, comme la projection technologique. La première contribution de mes travaux à cette problématique est l'un des apports majeurs de la thèse de Pierre-André Caron : l'implantation classique du mapping vertical pose un important problème de traçabilité et entraîne chez l'utilisateur une sensation de perte de maîtrise et donc possiblement un rejet de cette opération. Cette observation faite au travers d'une expérimentation a motivé Pierre-André à proposer une solution hybride qui supprimait ce problème mais allait à l'encontre de l'objectif *multi-plateforme* de l'IDM. Pour répondre à ce problème tout en restant dans le cadre du l'IDM, Rim Drira a défini un mapping vertical moins automatisé et plus humain. Une expérimentation a montré qu'il n'y avait plus de phénomène de rejet et de perte de maîtrise. De plus, ce que proposait Rim affichait explicitement aux utilisateurs les bonnes pratiques de tissage. Les retours des utilisateurs ont montré que cette explicitation était importante pour l'utilisabilité générale. La technique associée au travail de Rim était basée sur un mécanisme de templatisation, généralement utilisé pour des mappings dit horizontaux. Ce glissement vers un tissage plus horizontal pour gérer l'abstraction s'est accentué avec la thèse de Nadia Elouali. Dans son travail, Nadia a dû faire face à une projection technologique qui devait se faire plus tôt, c'est-à-dire dès la modélisation "métier". Nous avons montré qu'une utilisation judicieuse du mécanisme de template permettait d'intégrer des aspects technologiques dans une modélisation métier, tout en proposant une utilisabilité élevée, en respectant les contraintes techniques et économiques du

domaine métier et en évitant le problème de versionning pour le langage de modélisation.

Le **deuxième aspect** sur lequel j'ai contribué concerne **l'activité de modélisation**. J'ai d'abord proposé une démarche où des méthodologies pouvaient être spécifiées et associées à des langages de modélisation afin de faciliter la création de modèles. Avec le thèse Daniel Liabeuf, j'ai continué sur la simplification de l'activité de modélisation en visant à améliorer le copier/coller, opération très répandue pour la construction de tout type de contenu. Ce mécanisme est malheureusement sous-utilisé en modélisation. Daniel a montré qu'il n'y avait pas de consensus à l'heure actuelle sur le principe de fonctionnement de ce mécanisme pour la modélisation UML. Au travers d'une enquête, il a démontré, pour UML, la complexité de la relation de cette opération avec la syntaxe, la sémantique et la notation et les problèmes que cela engendre vis-à-vis de la détermination des attentes des utilisateurs. Enfin, la collaboration est aussi un moyen efficace voire une étape indispensable pour définir des modèles. Pour cette raison, j'encadre la thèse de Michel Dirix. Celle-ci a déjà comme résultats, l'ensemble des éléments d'awareness dont l'affichage est pertinent pour l'activité de modélisation. Ils vont permettre une spécification fine des besoins liés à la collaboration au sein de tout méta-modèle. Ces trois travaux ont aussi montré que la spécification d'interactions autour d'un langage de modélisation apportait du sens à ce dernier. Raison pour laquelle, l'existence d'une syntaxe d'interactions serait à mon avis utile.

La **notation visuelle est le dernier aspect** que j'ai abordé dans ma recherche. Cet aspect peut sembler inclus dans le précédent, mais je préfère en faire une problématique à part entière, comme l'a fait Moody. Non seulement, cet aspect renvoie à une grande complexité mais, de plus, il est transversal à l'IDM car on peut retrouver ce type de préoccupation dans la définition de règles de composition ou de transformation. Ma principale contribution a été d'identifier les variables élémentaires utilisées par la Physique des Notations et d'en délimiter ainsi l'espace informationnel, et d'offrir un support logiciel pour la création de métriques sur la qualité des notations visuelles. Comme nous l'avons indiqué dans le chapitre 5, ce résultat est la première étape d'une démarche que j'ai entamée il y a quelques années où je vise à terme de pouvoir guider très finement les concepteurs de langages de modélisation dans leur choix de syntaxe concrète. Une étude pilote a d'ailleurs été décrite dans ce même chapitre afin de donner plus de détails sur la démarche adoptée. Celle-ci s'inscrit dans une tendance actuelle de la communauté IDM, tendance à laquelle je participe activement.

Perspectives : l'équipe Carbon

Lors de mes travaux concernant les EIAH ou les IHM, j'ai pu constater que la modélisation, comme on la pratique en GL, n'était pas une activité répandue ni une préoccupation récurrente. En EIAH, j'avais choisi de travailler sur les dispositifs "artisanaux" et non sur les dispositifs industriels (à l'époque les MOOC n'en étaient qu'à leur balbutiement). J'ai pu observer que les enseignants ne trouvaient pas suffisants les bénéfices de réutilisation ou de déploiement automatique pour passer du temps à modéliser. Ce retour d'investissement trop faible entraînait une difficulté

à trouver des volontaires pour des expérimentations, et donc parfois une certaine démotivation de notre côté. En IHM, l'intérêt pour la modélisation est plus grand, mais cette communauté a déjà depuis longtemps expérimenté des approches par modèle et une certaine désillusion semble entourée cette pratique. De plus, de nos discussions avec des professionnels, il semble que la préoccupation principale en IHM n'est pas de coder la création de widgets graphiques et leur positionnement, ou les réactions à des événements. Les deux principaux problèmes sont 1) d'être le plus ergonomique possible (quelles informations afficher et à quel moment, à quel endroit) 2) de respecter les pratiques habituelles d'utilisation du périphérique en question et de suivre, si c'est le cas, la charte graphique imposée par un éventuel commanditaire. Des outils de sketching ou faire des maquettes d'interfaces au tableau leur semblent bien plus pertinent que des approches comme UsiXML. Enfin, les outils présentent le risque de parasiter "leur code". Même si, bien sûr, je ne partage pas leur avis, il reste que, comme en EIAH, il est difficile de trouver des volontaires "motivés". Pendant ces mêmes entretiens ou enquêtes auprès de professionnels, j'ai aussi remarqué que l'intérêt pour la modélisation était par contre évident en ce qui concerne les aspects logiciels : structure de l'application, comportement des services, déploiement dans une architecture orientée service... L'expérience de ces praticiens leur a aussi montré que pour la maintenance ou l'arrivée de nouveaux membres dans une équipe, des supports visuels sont d'une grande valeur. Et ils sont souvent assez conscient qu'il y a principalement un problème d'ergonomie des outils de modélisation et qu'avec une amélioration de celle-ci la modélisation aurait plus de place dans leur quotidien. Ces trois observations semblent donc indiquer que la modélisation a plus de "sens" en génie logiciel qu'en IHM ou en e-learning. Pour cette raison, j'ai réorienté depuis peu mon domaine d'application vers le génie logiciel.
Un intérêt supplémentaire à cette réorientation est pour moi que le terrain a ici déjà été étudié. En effet, les problèmes d'ergonomie de la modélisation logicielle en général sont connus pour ce domaine grâce à différentes enquêtes qui ont été menées sur ce sujet (Petre 2013, Forward et al. 2010, Chaudron et al. 2012, Grossman et al. 2005). On peut retenir trois faiblesses de la modélisation logicielle (et souvent d'UML en particulier) du point de vue ergonomique :

1. Notation trop complexe et manque de contexte : il y a trop de concepts/associations à connaître ; pour UML la séparation forcée entre cas d'utilisation, la structure et le comportement entraîne pour chaque diagramme un manque cruel de contextualisation ; les syntaxes concrètes sont généralement peu expressive (même si avec l'habitude, cela pose moins de problème)...

2. Désynchronisation entre le code et le modèle : souvent un modèle n'est rapidement plus à jour car des modifications conceptuelles sont faites pendant l'implémentation. Ceci réduit fortement son intérêt et donc celui de la modélisation en général.

3. IHM des outils de modélisation : dans (Forward et al. 2010), les participants à l'enquête place "la rapidité et la facilité de créer des mo-

> dèles" en 3ème position des attributs les plus importants de la modélisation logicielle, devant la génération de code ou le support à l'analyse. Et comme le disent Forward et al. "*Visual tools do not necessarily provide an intuitive interface to allow for easy development, and the inefficiencies of mouse actions versus keyboard strokes may limit the tools' abilities to be quick and efficient for the developer*".

Ces trois problèmes vont de pair : si la notation est efficace mais qu'à terme l'information contenu dans les modèles n'est plus pertinente, le modèle n'est plus intéressant ; s'il faut 51 clics et 7 saisies claviers pour créer un modèle à 2 éléments, une augmentation d'efficacité de la notation sera peu intéressante... *Améliorer l'ergonomie de la modélisation logicielle en adressant ces trois problèmes* est la thématique scientifique de l'équipe Carbon et la raison de sa création. Cette équipe est composée de Cédric Dumoulin, Jean-Claude Tarby et moi-même. Cédric Dumoulin est Maître de Conférence au LIFL dans l'équipe DART et aussi un des créateurs de l'outil de modélisation UML de référence d'Eclipse : Papyrus [7]. Ses derniers apports à cette outil concernent la génération de code à partir d'UML et surtout la rétro-ingénierie (générer un modèle UML à partir de code Java). Sa préoccupation en filigrane de cette activité est d'avoir une synchronisation entre le code et le modèle [8]. Jean-Claude Tarby, déjà mentionné dans le chapitre 4, est Maître de Conférence au LIFL dans l'équipe NOCE. Si son domaine scientifique est l'IHM, sa problématique de thèse concernait les modèles de tâches (Tarby 1993). La modélisation est restée une thématique récurrente dans son activité de recherche (Paris et al. 2001, Diaper et Stanton 2003, Le Pallec et al. 2006b, Lewandowski et al. 2007, Rouillard et al. 2010, Elouali et al. 2012). Son implication plus importante dans la thématique IDM, notamment au travers de l'encadrement actuel de la thèse de Nadia Elouali, l'a convaincu d'orienter son domaine d'application à l'IDM afin d'étudier comment augmenter l'efficacité des IHM des outils de modélisation. Il a donc été naturel que nous formions une équipe où l'intersection de nos compétences et thématiques correspondait parfaitement à la problématique précédente concernant l'ergonomie des outils de modélisation logicielle : notation complexe, synchronisation code-modèle, IHM.
Cette configuration va surtout nous permettre de nous nourrir mutuellement des avancées de chacun. Pour ma part, une synchronisation code-modèle me permettra d'avoir un raffinement des activités humaines liées à la modélisation. Un exemple de raffinement est le suivant : quand un développeur étudie le code d'une classe Java B qui hérite d'une classe A, les deux classes étant dans le paquetage P, quel type de représentation UML souhaite-t-il avoir ? Un diagramme de classes représentant A, B et P et les classes associées à B (via ses attributs) ? Un diagramme de classes représentant les classes impliquées dans la structure d'itération qu'il est, par exemple, en train d'analyser ? Ou le diagramme de séquence associé ? Par rapport à son choix, préférera-t-il avoir juste un diagramme avec les

7. `www.eclipse.org/papyrus`

8. et une des raisons pratiques est que l'implémentation de Papyrus est devenu tellement volumineuse que l'absence de modèles UML "à jour" est devenu problématique : les nouveaux arrivants dans le projet open source Papyrus ont bien du mal à "entrer dans le code" et ceux qui reviennent après une certaine période d'inactivité ont du mal à s'y remettre

éléments souhaités ou avoir un diagramme de classes plus conséquent qui fait partie de la documentation avec une série de décorations visuelles afin de mettre en avant les éléments souhaités ? Si oui, quelles seront les décorations qui fonctionneront le mieux pour cette situation ?
Concernant les IHM, la piste actuelle de recherche de Jean-Claude Tarby est d'étudier d'autres périphériques d'interaction (tablette, lunettes de réalité augmentée, table interactive) pour faciliter et accélérer la saisie de diagrammes ou leur manipulation (notamment la réorganisation). Toutefois, cela va soulever des problèmes d'adaptation à la taille de l'écran de visualisation du périphérique qui pourra varier de 25,4cm (tablette 10 pouces) à plus d'1 mètre de diagonale (table de 42 pouces). Sur un petit écran, pour un diagramme de classe, il pourrait être nécessaire de cacher les attributs et les paquetages. Le nombre d'attributs d'une classe (en lien avec la richesse informationnelle de celle-ci) pourrait être représenté visuellement (ex : augmentation de l'intensité lumineuse), ainsi que son appartenance à un paquetage particulier (choix de couleurs). Sur une grande surface d'affichage, le nombre d'éléments pourrait être suffisamment élevé pour qu'il soit nécessaire de gérer la complexité par l'utilisation de variables visuelles. Encore une fois, chaque solution serait guidé par le contexte et les besoins du lecteur/auteur. Cette influence de la tâche de l'utilisateur est fondamentale en ergonomie et ce travail de concert (dans l'équipe) permettra une meilleure identification des tâches des praticiens du logiciel. La matrice des dimensions cognitives de la partie 1.4.1 sera une de nos bases de travail pour la décomposition cognitive de ces tâches.
Améliorer l'ergonomie d'un outil passe obligatoirement par une étude détaillée des activités qui lui sont/seront associées. C'est pour cette raison qu'une étude approfondie des activités liées à l'instrument "modélisation logicielle" (et aux éditeurs correspondants) est au coeur de l'équipe Carbon. Trois nouvelles thèses (financées par le CEA [9]) vont commencer en Décembre 2014/Janvier 2015, chaque thèse portant sur une des 3 sous-problématiques de l'équipe Carbon. Les doctorants vont démarrer par une période d'observation des pratiques des professionnels afin de déterminer - plus finement que les enquêtes précédemment citées - quand, comment et pourquoi la modélisation est utilisée et, parmi les praticiens, dans quelles proportions. L'idée principale est de rapidement étudier les besoins d'un nombre restreint d'activités liées à la modélisation et d'évaluer le manque d'adaptation/optimisation des notations, outils et périphériques d'interactions utilisés. Ces doctorants proposeront des solutions génériques et les implémenteront dans un outil (ici Papyrus) afin d'expérimenter le tout auprès de professionnels pour valider leurs hypothèses liées au manque d'adaptation. Ce cycle observation/analyse/proposition/évaluation est la démarche que nous avons décidé d'adopter dans l'équipe Carbon pour l'ensemble de nos futurs travaux... travaux qui, nous l'espérons, couvriront le plus possible d'activités liées à la modélisation.
La figure suivante résume la démarche et l'objectif global de l'équipe Carbon : améliorer l'ergonomie de la modélisation logicielle par l'étude des activités du monde réel qui lui sont associées et en ciblant ses trois problèmes majeurs actuels - la synchronisation avec le code, les interactions

9. principal contributeur de l'outil Papyrus

et la notation visuelle. Cette évolution visée des diagrammes logiciels actuels devrait aboutir à ce que nous appelons dans l'équipe Carbon les *diagrammes logiciels 2.0*.

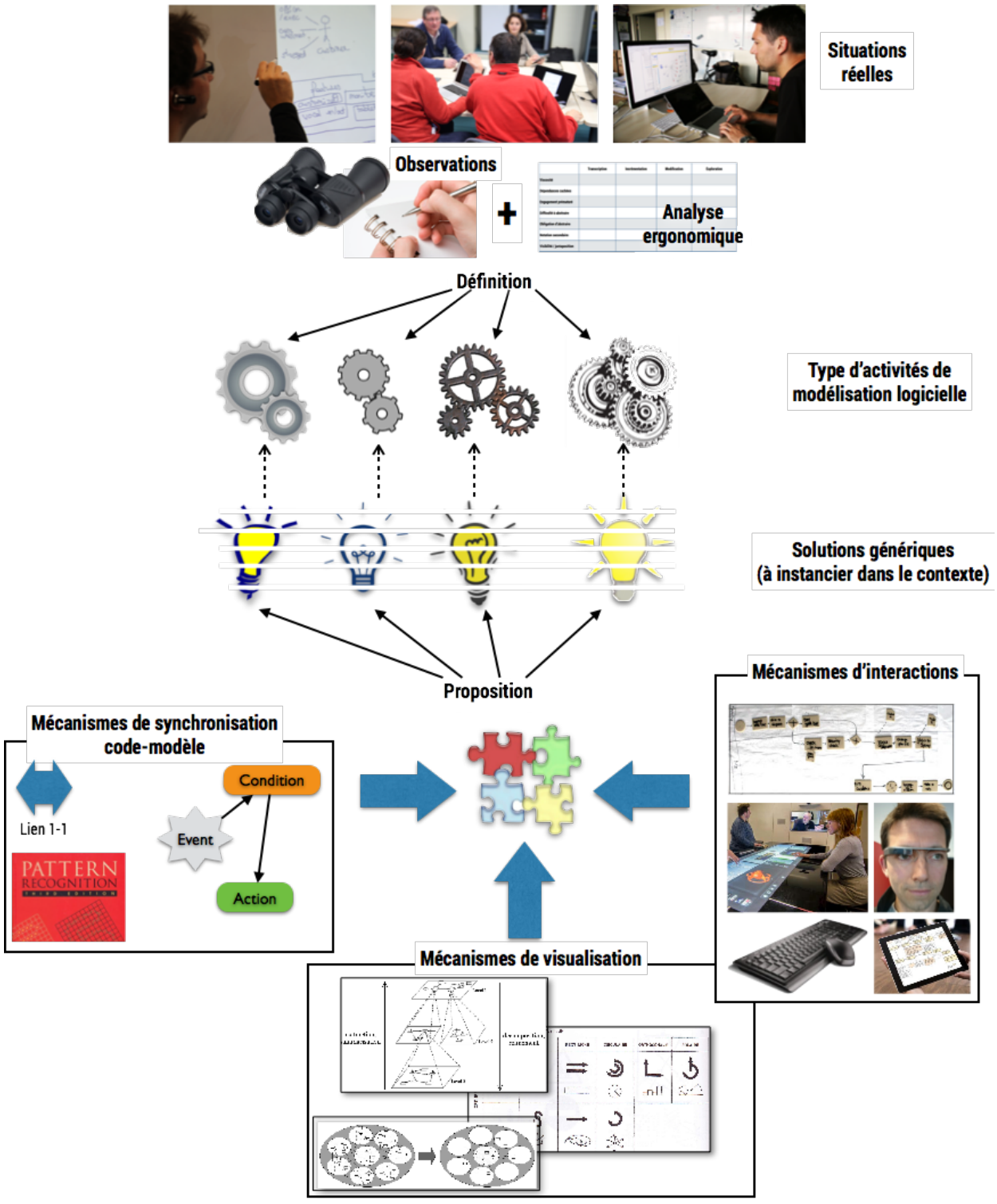


FIGURE 5.7 – *(Carbon) Diagramme Logiciel 1.0 + synchronisation code + périphériques d'interaction évolués + sémiologie graphique = Diagramme Logiciel 2.0*

Publications triées par thèse, collaboration ou thématique

Publications en lien avec la thèse de Pierre-André Caron

1. **"*Applying Model Driven Engineering Techniques and Tools to the Planets Game Learning Scenario*"**, Nodenot, Thierry ; Caron, Pierre-André ; Le Pallec, Xavier ; Laforcade, Pierre, *Journal of Interactive Media in Education* (2008-12-19) http ://jime.open.ac.uk/2008/23/

2. **"*La contextualisation de modèles, une étape indispensable à un développement dirigé par les modèles ?*"**, Caron, P.-A. ; Blay-Fornarino, M. ; Le Pallec, Xavier, *Revue RSTI - L'Objet, Numéro Spécial : Ingénierie Dirigée par les Modèles, 18 pages* (2007)

3. **"*Construire des dispositifs sur la plateforme Moodle application de l'ingénierie Bricoles*"**, Caron, P.-A. ; Hoogstoel, Frédéric ; Le Pallec, Xavier ; Warin, B., – Moodlemoot 2007 (2007)

4. **"*Scénarios et Dispositifs de formations spécialisés : Application de la Démarche d'Ingénierie BRICOLE pour une Instanciation sur MOODLE*"**, Caron, P.-A. ; Derycke, Alain ; Hoogstoel, Frédéric ; Le Pallec, Xavier ; Warin, B., – Colloque scénario 2007 Montreal, Canada France (2007)

5. **"*Le projet Metawep, Ingénierie dirigée par les modèles de dispositifs web support à l'apprentissage par projet*"**, Warin, B. ; Caron, P.-A. ; Le Pallec, Xavier ; Hoogstoel, Frédéric, – poster, EIAH 2007, Lausanne, Suisse (2007)

6. **"*Visual design of coherent technology enhanced learning systems : Lessons learned from the CPM language*"**, Nodenot, T. ; Laforcade, P. ; Le Pallec, Xavier, Handbook of Visual Languages for Instructional Design : Theories and Practices Chapitre no 2.8. Luca Botturi and Todd Stubbs editors, Hershey, PA : IDEA Group Inc. – (Décembre 2007)

7. **"*Pedagogical Scenario Modelling, Deployment, Execution and Evolution*"**, Peter, Yvan ; Le Pallec, Xavier ; Vantroys, Thomas, (Novembre 2007) Architecture Solutions for E-Learning Systems, Claus

Pahl Editor, Hershey, Architecture Solutions for E-Learning Systems, Claus Pahl Editor, Hershey, PA : IDEA Group Inc. 283–305 — (Novembre 2007)

Publications en lien avec la thèse de Rim Drira

1. **"*Contextualizing Learning Scenarios According to Different Learning Management Systems*"**, Drira, Rim ; Laroussi, Mona ; Le Pallec, Xavier ; Warin, Bruno, *IEEE Trans. Learn. Technol.* IEEE Computer Society Press **5**, **3**, 213–225 — (2012)

2. **"*ACoMoD : Assistance à la réutilisation et à la modélisation contextualisée de dispositifs pédagogiques dans le cadre de l'ingénierie dirigée par les modèles*"**, Rim Drira, Mona Laroussi, Xavier Le Pallec, Alain Derycke, *IEIAH 2009* — (2009)

3. **"*A Model Driven Approach to Adapt Instructional Strategies Modelling Language to Different Design Contexts*"**, Rim Drira , Mona Laroussi , Xavier Le Pallec , Alain Derycke , Henda Ben Ghezala, *IEEE eLearning and mLearning Conference (eL&mL)* Cancun, Mexico, 1-7 February, 2009

Publications en lien avec la collaboration avec Nadia Elouali

1. **"*Multimodal Interaction : a Survey from Model Driven Engineering and Mobile perspectives*"**, Elouali, Nadia ; Rouillard, José ; Le Pallec, Xavier ; Tarby, Jean-Claude, *Journal on Multimodal User Interfaces* — (2013) Springer

2. **"*Approche IDM pour le développement d'applications mobiles multimodales*"**, Elouali, Nadia ; Tarby, Jean-Claude ; Le Pallec, Xavier ; Rouillard, José, 9ème édition de la conférence MAnifestation des JEunes Chercheurs en Sciences et Technologies de l'Information et de la Communication - MajecSTIC 2012, Villeneuve d'Ascq, France — (2012-10-29)

Publications en lien avec les Environnements Intelligents et Applications Web Multimodales

1. **"*Challenges for the Design of Intelligent and Multimodal Cognitive Systems*"**, Rouillard, José ; Tarby, Jean-Claude ; Le Pallec, Xavier ; Marvie, Raphael, *ERCIM News - Journal of the European Research Consortium for Informatics and Mathematics* , ERCIM EEIG Intelligent and Cognitive Systems **84** — (2011)

2. **"*From Metamodeling to Automatic Generation of Multimodal Interfaces for Ambient Computing*"**, Rouillard, José ; Tarby, Jean-Claude ; Le Pallec, Xavier ; Marvie, Raphael, *International Journal On Advances in Software*, IARIA **3, 3&4** — (2011)

3. **"*Le projet Miny : des interactions multimodales pour les applications Web*"**, Le Pallec, Xavier ; Tarby, Jean-Claude ; Rouillard, José, Ergo'IHM 2012, Biarritz, France — (2012-10-16)

4. **"*Facilitating the Design of Multi-channel Interfaces for Ambient Computing*"**, Rouillard, José ; Le Pallec, Xavier ; Tarby, Jean-Claude ; Marvie, Raphael, 2010 Third International Conference on Advances in Computer-Human Interactions, Saint Maarten, Pays-Bas, IEEE — (2010-02-16)

5. **"*A support to multi-devices web application*"**, Le Pallec, Xavier ; Marvie, Raphael ; Rouillard, José ; Tarby, Jean-Claude, ACM, 391–392, Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology New York, NY, USA États-Unis — (2010-10-04)

Publications en lien avec les Processus de Modélisation Incrémentaux

1. **"*Supporting generic methodologies to assist IMS-LD modeling*"**, Le Pallec X., Moura C., Marvie R., Nebut M., Tarby J.C, *he 6th IEEE International Conference on Advanced Learning Technologies* July 5-7, 2006, Kerkrade, The Netherlands, IEEE **pp 923-927** — (2006)

2. **"*Processus de modéisation incrémental pour le développement d'applications interactives basées sur pac*"**, Raphael Marvie Jean-Claude Tarby, Xavier Le Pallec et Mirabelle Nebut, *Workshop IDM & IHM : Ingénierie Dirigée par les Modèles et Interaction Homme-Machine* dans 18e Conférence Francophone sur l ?Interaction Homme-Machine, Avril 2006 — (2006)

Publications en lien avec la thèse de Daniel Liabeuf

1. **"*Model-driven Evolution for multimodal mobile Geographic Information Systems*"**, Liabeuf, Daniel ; Le Pallec, Xavier ; Rouillard, José ; Tarby, Jean-Claude ; Elouali, Nadia, *ERCIM News - Journal of the European Research Consortium for Informatics and Mathematics* ERCIM EEIG Evolving Software **88** — (2012)

2. **"*An Empirical Study on the Anticipation of the Result of Copying and Pasting among UML Editors*"**, Daniel Liabeuf, Xavier Le Pallec and José Rouillard, *The 8th System Analysis and Modelling Conference* Septembre 2014, Valencia, Espagne Springer LNCS **18 pages**

Publications en lien avec la thèse de Michel Dirix

1. **"*Awareness Information For Modeling Tools*"**, Michel Dirix, Xavier Le Pallec, and Alexis Muller, *22nd International Conference on COOPERATIVE INFORMATION SYSTEMS (CoopIS 2014)* Octobre 2014, Amantea, Italie Springer LNCS **18 pages**

Publications en lien avec les notations visuelles

1. **"*Vers une approche centrée humain pour la définition de langages de modélisation graphiques*"**, Dupuy-Chessa, Sophie ; Combemale, Benoit ; Gervais, Marie-Pierre ; Nodenot, Thierry ; Le Pallec, Xavier ; Wouters, Laurent, 79-94, 32ème congrès Inforsid'2014, Lyon, France — (2014-05)

2. **"*Support for quality metrics in metamodelling*"**, Le Pallec, Xavier ; Dupuy-Chessa, Sophie, GMLD (Graphical Modeling Language Definition) Workshop @ ECFMA 2013, Montpellier, France — (2013-07-02)

3. **"*Modélisation graphique des SI : Du traitement visuel de modèles complexes*"**, Bihanic, David ; Chevalier, Max ; Dupuy-Chessa, Sophie ; Morineau, Thierry ; Polacsek, Thomas ; Le Pallec, Xavier, Inforsid 2013, Paris, France — (2013-05-29)

4. **"*Intégration de métriques de qualité des modèles et des langages dans l'outil ModX*"**, Le Pallec, Xavier ; Dupuy-Chessa, Sophie, 6p., CIEL 2012 - Conf erence en Ing enieriE du Logiciel, Rennes, France — (2012-06-19)

5. **"*Intégration de métriques de qualité des modèles et des métamodèles dans l'outil ModX*"**, Le Pallec, Xavier ; Dupuy-Chessa, Sophie, Inforsid 2011, Lille, France — (2011-05-24)

Bibliographie

Wisam Al Abed et Jörg Kienzle. Aspect-oriented modelling for distributed systems. Dans Jon Whittle, Tony Clark, et Thomas Kühne, éditeurs, *Model Driven Engineering Languages and Systems*, volume 6981 de *Lecture Notes in Computer Science*, pages 123–137. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-24484-1. URL http://dx.doi.org/10.1007/978-3-642-24485-8_10. (Cité page 15.)

MarcelF. Amstel, MarkG.J. Brand, et Alexander Serebrenik. Traceability visualization in model transformations with tracevis. Dans Zhenjiang Hu et Juan Lara, éditeurs, *Theory and Practice of Model Transformations*, volume 7307 de *Lecture Notes in Computer Science*, pages 152–159. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-30475-0. URL http://dx.doi.org/10.1007/978-3-642-30476-7_10. (Cité pages ix et 20.)

C. Atkinson et T. Kuhne. Model-driven development : a metamodeling foundation. *Software, IEEE*, 20(5) :36–41, Sept 2003. ISSN 0740-7459. (Cité page 11.)

Pierre-Alain Avouac, Philippe Lalanda, et Laurence Nigay. Service-oriented autonomic multimodal interaction in a pervasive environment. Dans *Conference Proceedings of ICMI'11, the 13th international conference on Multimodal interfaces, Spain, November 14-18, 2011, ACM Press*, pages 369–376, Alicante, Spain, 2011. ACM New York, NY, USA. (Cité page 78.)

E. Baniassad et S. Clarke. Theme : an approach for aspect-oriented analysis and design. Dans *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 158–167, May 2004. (Cité page 15.)

André Berten. Dispositif, médiation, créativité : petite généalogie. Dans *Hermès (Paris. 1988), 1999, 25, fascicule thématique" Le dispositif : entre usage et concept"–Dispositif et médiation des savoirs. Colloque international, Louvain-la-Neuve, BEL, 1998-04-24*. CNRS Editions, Paris (FRA), 1999. (Cité page 41.)

Jacques Bertin. *Semiology of Graphics. Diagrams, Networks and Maps*. University of Wisconsin Press, 1983. (Cité pages 31, 116 et 117.)

David Bihanic, Max Chevalier, Sophie Dupuy-Chessa, Thierry Morineau, Thomas Polacsek, et Xavier Le Pallec. Modélisation graphique des SI : Du traitement visuel de modèles complexes. Dans *Inforsid 2013*, Paris, France, Mai 2013. URL http://hal.inria.fr/hal-00823276. (Cité page 135.)

Alan Blackwell et Yuri Engelhardt. A meta-taxonomy for diagram research. Dans Michael Anderson, Bernd Meyer, et Patrick Olivier, éditeurs, *Diagrammatic Representation and Reasoning*, pages 47–64. Springer London, 2002. ISBN 978-1-85233-242-6. URL http://dx.doi.org/10.1007/978-1-4471-0109-3_3. (Cité page 31.)

Nick Boldt et MARCELO PATERNOSTRO. Introduction to the eclipse modeling framework. *IBM CASCON*, 2006. (Cité page 49.)

Richard A. Bolt. “put-that-there” : Voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, 14(3) :262–270, Juillet 1980. ISSN 0097-8930. URL http://doi.acm.org/10.1145/965105.807503. (Cité page 4.)

Eric Bruneton, Thierry Coupaye, et Jean-Bernard Stefani. The fractal project. dec 2012. URL http://fractal.ow2.org/. (Cité page 21.)

Jean-marie Burkhardt, Françoise Détienne, et Susan Wiedenbeck. Object-Oriented Program Comprehension : Effect of Expertise , Task and Phase. *Empirical Software Engineering*, 7(2) :115–156, 2002. (Cité page 27.)

Patrice Caire, Nicolas Genon, Patrick Heymans, et Daniel Laurence Moody. Visual notation design 2.0 : Towards user comprehensible requirements engineering notations. Dans *RE*, pages 115–124. IEEE, 2013. ISBN 978-1-4673-5765-4. (Cité page 105.)

José C. Campos et Michael D. Harrison. Model checking interactor specifications. *Automated Software Engg.*, 8(3-4) :275–310, Août 2001. ISSN 0928-8910. URL http://dx.doi.org/10.1023/A:1011265604021. (Cité page 24.)

Stuart K. Card, Thomas P. Moran, et Allen Newell. *The Psychology of Human-Computer Interaction*. Taylor & Francis, 1983. ISBN 9780898598599. URL http://books.google.fr/books?id=30UsZ8hy2ZsC. (Cité page 4.)

Olivier Caron, Bernard Carré, Alexis Muller, et Gilles Vanwormhoudt. An ocl formulation of uml2 template binding. Dans Thomas Baar, Alfred Strohmeier, Ana M. D. Moreira, et Stephen J. Mellor, éditeurs, *UML*, volume 3273 de *Lecture Notes in Computer Science*, pages 27–40. Springer, 2004. ISBN 3-540-23307-5. (Cité page 22.)

P.-A. Caron, M. Blay-Fornarino, et Xavier Le Pallec. La contextualisation de modèles, une étape indispensable à un développement dirigé par les modèles ? *Revue RSTI - L'Objet, Numéro Spécial : Ingénierie Dirigée par les Modèles, 18 pages*, pages –, 2007a. URL http://hal.archives-ouvertes.fr/hal-00731357. (Cité pages 50 et 60.)

P.-A. Caron, M. Blay-Fornarino, et Xavier Le Pallec. La contextualisation de modèles, une étape indispensable à un développement dirigé par les modèles ? *Revue RSTI - L'Objet, Numéro Spécial : Ingénierie Dirigée par les Modèles, 18 pages*, pages –, 2007b. URL http://hal.archives-ouvertes.fr/hal-00731357. (Cité page 50.)

P.-A. Caron, Alain Derycke, Frédéric Hoogstoel, Xavier Le Pallec, et B. Warin. Scénarios et Dispositifs de formations spécialisés : Application de la Démarche d'Ingénierie BRICOLE pour une Instanciation sur MOODLE. Dans *Colloque scénario 2007*, pages –, Montreal, Canada, France, 2007c. URL http://hal.archives-ouvertes.fr/hal-00731395. (Cité pages 50 et 60.)

P.-A. Caron, Frédéric Hoogstoel, Xavier Le Pallec, et B. Warin. Construire des dispositifs sur la plateforme Moodle application de l'ingénierie Bricoles. Dans *Moodlemoot 2007*, pages –, undef, France, 2007d. URL http://hal.archives-ouvertes.fr/hal-00731413. (Cité pages 50 et 60.)

Pierre-André Caron. *Ingénierie dirigée par les modèles pour la construction de dispositifs pédagogiques sur des plateformes de formation*. These, Université des Sciences et Technologie de Lille - Lille I, Juin 2007. URL http://tel.archives-ouvertes.fr/tel-00156376. (Cité page 19.)

Pierre-André Caron, Alain Derycke, et Xavier Le Pallec. The bricoles project : support socially informed design of learning environment. Dans Chee-Kit Looi, Gordon I. McCalla, Bert Bredeweg, et Joost Breuker, éditeurs, *AIED*, volume 125 de *Frontiers in Artificial Intelligence and Applications*, pages 759–761. IOS Press, 2005. ISBN 978-1-58603-530-3. (Cité pages 50 et 60.)

Shana K Carpenter, Miko M Wilford, Nate Kornell, et Kellie M Mullaney. Appearances can be deceiving : instructor fluency increases perceptions of learning without increasing actual learning. *Psychonomic bulletin & review*, 20(6) :1350–1356, 2013. (Cité page 133.)

CENA. Ivy home page, 2014. URL http://www.tls.cena.fr/products/ivy/. [Online ; accessed 22-May-2014]. (Cité page 52.)

Michel Chaudron, R.V., Werner Heijstek, et Ariadi Nugroho. How effective is uml modeling ? *Software & Systems Modeling*, 11(4) :571–580, 2012. ISSN 1619-1366. URL http://dx.doi.org/10.1007/s10270-012-0278-4. (Cité pages 22, 30 et 139.)

Luke Church et Thomas Green. Cognitive Dimensions - a short tutorial. Dans *Proceedings of PPIG08*, 2008. (Cité page 26.)

Tony Clark, Andy Evans, Paul Sammut, et James Willans. *Applied Metamodelling - A Foundation for Language Driven Development*. Second édition, 2004. URL http://citeseerx.ist.psu.edu/showciting?cid=3456058. (Cité pages 10, 11, 14, 17 et 18.)

Joelle Coutaz. Pac, an object oriented model for dialog design. *Proceedings Interact*, 87 :431–436, 1987. (Cité page 92.)

Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May, et Richard M Young. Four easy pieces for assessing the usability of multimodal interaction : the care properties. Dans *InterAct*, volume 95, pages 115–120, 1995. (Cité page 42.)

Jacques Crinon et Denis Legros. *Psychologie des apprentissages et multimédia*. Armand Colin, 2002. URL http://halshs.archives-ouvertes.fr/halshs-00750418. (Cité page 38.)

a. a. Cuellar, C. M. Lloyd, P. F. Nielsen, D. P. Bullivant, D. P. Nickerson, et P. J. Hunter. An Overview of CellML 1.1, a Biological Model Description Language. *Simulation*, 79(12) :740–747, Décembre 2003. ISSN 0037-5497. (Cité page 27.)

Juan De Lara, Hans Vangheluwe, et Manuel Alfonseca. Meta-modelling and graph grammars for multi-paradigm modelling in atom3. *Software and Systems Modeling*, 3(3) :194–209, 2004. (Cité page 49.)

D. Diaper et N. Stanton. *The Handbook of Task Analysis for Human-Computer Interaction*. Taylor & Francis, 2003. ISBN 9781410609403. URL http://books.google.fr/books?id=EuddOAMeI5sC. (Cité page 140.)

Connor Diemand-Yauman, Daniel M Oppenheimer, et Erikka B Vaughan. Fortune favors the () : Effects of disfluency on educational outcomes. *Cognition*, 118(1) :111–115, 2011. (Cité page 133.)

Michel Dirix, Xavier Le Pallec, et Alexis Muller. Awareness information for modeling tools. Dans *22nd International Conference on COOPERATIVE INFORMATION SYSTEMS (CoopIS 2014)*, volume 22 de *Lecture Notes in Computer Science*, page 18 pages, Amantea, Italy, oct 2014. Springer. (Cité page 106.)

Paul Dourish et Victoria Bellotti. Awareness and coordination in shared workspaces. Dans *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work*, CSCW '92, pages 107–114, New York, NY, USA, 1992. ACM. ISBN 0-89791-542-9. URL http://doi.acm.org/10.1145/143457.143468. (Cité page 30.)

R. Drira, M. Laroussi, X. Le Pallec, et B. Warin. Contextualizing learning scenarios according to different learning management systems. *Learning Technologies, IEEE Transactions on*, 5(3) :213–225, July 2012. ISSN 1939-1382. (Cité pages 60 et 74.)

Bruno Dumas, Beat Signer, et Denis Lalanne. A graphical editor for the {SMUIML} multimodal user interaction description language. *Science of Computer Programming*, 86(0) :30 – 42, 2014. ISSN 0167-6423. URL http://www.sciencedirect.com/science/article/pii/S0167642313001019. Special issue on Software Support for User Interface Description Languages (UIDL 2011). (Cité page 78.)

Sophie Dupuy-Chessa, Benoit Combemale, Marie-Pierre Gervais, Thierry Nodenot, Xavier Le Pallec, et Laurent Wouters. Vers une approche centrée humain pour la définition de langages de modélisation graphiques. Dans *32ème congrès Inforsid'2014*, pages 79–94, Lyon, France, Mai 2014. URL http://hal.archives-ouvertes.fr/hal-01002994. (Cité page 135.)

A.F. Dutka. *AMA Handbook for Customer Satisfaction*. NTC Business Books, 1995. (Cité page 131.)

W.J. Dzidek, E. Arisholm, et L.C. Briand. A realistic empirical evaluation of the costs and benefits of uml in software maintenance. *Software Engineering, IEEE Transactions on*, 34(3) :407–432, May 2008. ISSN 0098-5589. (Cité pages 22 et 30.)

Maged Elaasar et Yvan Labiche. Diagram definition : A case study with the uml class diagram. Dans Jon Whittle, Tony Clark, et Thomas KÃ$\frac{1}{4}$hne, éditeurs, *Model Driven Engineering Languages and Systems*, volume 6981 de *Lecture Notes in Computer Science*, pages 364–378. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-24484-1. URL http://dx.doi.org/10.1007/978-3-642-24485-8_26. (Cité page 14.)

Nadia Elouali, José Rouillard, Xavier Le Pallec, et Jean-Claude Tarby. Multimodal interaction : a survey from model driven engineering and mobile perspectives. *Journal on Multimodal User Interfaces*, 7(4) :351–370, 2013. ISSN 1783-7677. URL http://dx.doi.org/10.1007/s12193-013-0126-z. (Cité page 44.)

Nadia Elouali, Jean-Claude Tarby, Xavier Le Pallec, et José Rouillard. Approche IDM pour le développement d'applications mobiles multimodales. Dans Anne Etien, éditeur, *9ème édition de la conférence MAnifestation des JEunes Chercheurs en Sciences et Technologies de l'Information et de la Communication - MajecSTIC 2012 (2012)*, Villeneuve d'Ascq, France, Octobre 2012. Nicolas Gouvy. URL http://hal.inria.fr/hal-00780187. (Cité pages 50 et 140.)

Jacky Estublier, Anca Daniela Ionita, et German Vega. A domain composition approach. Dans *Software Engineering Research and Practice*, pages 389–395, 2005a. (Cité page 18.)

Jacky Estublier, German Vega, et Anca Daniela Ionita. Composing domain-specific languages for wide-scope software engineering applications. Dans *Model Driven Engineering Languages and Systems*, pages 69–83. Springer Berlin Heidelberg, 2005b. (Cité page 18.)

Matthias Farwick, Berthold Agreiter, Jules White, Simon Forster, Norbert Lanzanasto, et Ruth Breu. A web-based collaborative metamodeling environment with secure remote model access. Dans *Proceedings of the 10th International Conference on Web Engineering*, ICWE'10, pages 278–291, Berlin, Heidelberg, 2010a. Springer-Verlag. ISBN 3-642-13910-8, 978-3-642-13910-9. URL http://dl.acm.org/citation.cfm?id=1884110.1884133. (Cité page 29.)

Matthias Farwick, Berthold Agreiter, Jules White, Simon Forster, Norbert Lanzanasto, et Ruth Breu. A web-based collaborative metamodeling environment with secure remote model access. Dans Boualem Benatallah, Fabio Casati, Gerti Kappel, et Gustavo Rossi, éditeurs, *Web Engineering*, volume 6189 de *Lecture Notes in Computer Science*, pages 278–291. Springer Berlin Heidelberg, 2010b. ISBN 978-3-642-13910-9. URL http://dx.doi.org/10.1007/978-3-642-13911-6_19. (Cité page 108.)

H. T. Fisher. *Mapping Information*. Cambridge, MA : Abt Books, 1983. (Cité page 31.)

Andrew Forward, Timothy C. Lethbridge, et Omar Badreddin. Perceptions of Software Modeling : A Survey of Software Practitioners. Rapport technique, University of Ottawa, 2010. (Cité pages 22, 27, 30, 98 et 139.)

Eclipse Foundation. Qvto : Qvt operational. *http ://projects.eclipse.org/projects/modeling.mmt.qvt-oml*, 2008. URL `http://projects.eclipse.org/projects/modeling.mmt.qvt-oml`. (Cité page 123.)

Eclipse Foundation. Eol : Epsilon object language. *http ://www.eclipse.org/epsilon/doc/eol/*, 2012. URL `http://www.eclipse.org/epsilon/doc/eol/`. (Cité page 123.)

Eclipse Foundation. Eugenia : Gmf for mortals. *http ://www.eclipse.org/epsilon/doc/eugenia/*, 2013. URL `http://www.eclipse.org/epsilon/doc/eugenia/`. (Cité pages 14, 49 et 123.)

Robert France et Bernhard Rumpe. Model-driven development of complex software : A research roadmap. Dans *2007 Future of Software Engineering*, FOSE '07, pages 37–54, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2829-5. URL `http://dx.doi.org/10.1109/FOSE.2007.14`. (Cité page 26.)

Ulrich Frank. Some guidelines for the conception of domain-specific modelling languages. Dans Markus Nüttgens, Oliver Thomas, et Barbara Weber, éditeurs, *EMISA*, volume 190 de *LNI*, pages 93–106. GI, 2011. ISBN 978-3-88579-284-0. (Cité page 24.)

Jesus Gallardo, Ana I. Molina, Crescencio Bravo, Miguel A. Redondo, et Cesar A. Collazos. An ontological conceptualization approach for awareness in domain-independent collaborative modeling systems : Application to a model-driven development method. *Expert Systems with Applications*, 38(2) :1099 – 1118, 2011. ISSN 0957-4174. URL `http://www.sciencedirect.com/science/article/pii/S0957417410004161`. Intelligent Collaboration and Design. (Cité page 30.)

Fernando Gallego, Ana I. Molina, Jesús Gallardo, et Crescencio Bravo. A conceptual framework for modeling awareness mechanisms in collaborative systems. Dans Pedro Campos, T. C. Nicholas Graham, Joaquim A. Jorge, Nuno Jardim Nunes, Philippe A. Palanque, et Marco Winckler, éditeurs, *INTERACT (4)*, volume 6949 de *Lecture Notes in Computer Science*, pages 454–457. Springer, 2011. ISBN 978-3-642-23767-6. (Cité page 30.)

Nicolas Genon, Patrick Heymans, et Daniel Amyot. Analysing the cognitive effectiveness ofthebpmn2.ovisualnotation. Dans Brian A. Malloy, Steffen Staab, et Mark van den Brand, éditeurs, *SLE*, volume 6563 de *Lecture Notes in Computer Science*, pages 377–396. Springer, 2010. ISBN 978-3-642-19439-9. (Cité page 105.)

Nicolas Genon, Patrick Heymans, et Daniel Amyot. Analysing the cognitive effectiveness of the bpmn 2.0 visual notation. Dans Brian Malloy, Steffen Staab, et Mark van den Brand, éditeurs, *Software Language Engineering*, volume 6563 de *Lecture Notes in Computer Science*, pages 377–396. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-19439-9. URL http://dx.doi.org/10.1007/978-3-642-19440-5_25. (Cité page 3.)

Sébastien George et Alain Derycke. Éditorial du numéro spécial : Conceptions et usages des plates-formes de formation. *Revue Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation*, 12, 2005. (Cité page 37.)

Carlo Ghezzi, Mehdi Jazayeri, et Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991. ISBN 0-13-820432. (Cité page 27.)

T. R. G. Green. Cognitive dimensions of notations. Dans *People and Computers V*, pages 443–460. University Press, 1989. (Cité page 22.)

T. R. G. Green et M. Petre. Usability analysis of visual programming environments : a 'cognitive dimensions' framework. *JOURNAL OF VISUAL LANGUAGES AND COMPUTING*, 7 :131–174, 1996. (Cité page 22.)

Martin Grossman, Jay E. Aronson, et Richard V. McCarthy. Does uml make the grade ? insights from the software development community. *Inf. Softw. Technol.*, 47(6) :383–397, Avril 2005. ISSN 0950-5849. URL http://dx.doi.org/10.1016/j.infsof.2004.09.005. (Cité pages 3 et 139.)

Christophe Guychard, Sylvain Guerin, Ali Koudri, Antoine Beugnard, et Fabien Dagnat. Conceptual interoperability through Models Federation. Dans *Semantic Information Federation Community Workshop*, page ., Miami, États-Unis, Octobre 2013. URL http://hal.archives-ouvertes.fr/hal-00905036. 13939 13939. (Cité page 50.)

D. Harel et B. Rumpe. Meaningful modeling : what's the semantics of "semantics" ? *Computer*, 37(10) :64–72, Oct 2004. ISSN 0018-9162. (Cité page 11.)

Markus Herrmannsdoerfer, Daniel Ratiu, et Guido Wachsmuth. Language evolution in practice : The history of gmf. Dans *Software Language Engineering*, pages 3–22. Springer, 2010. (Cité page 49.)

Jean-Michel Hoc. *L'ergonomie cognitive*. PUF, 1991. (Cité page 22.)

Michael Huberman. Recipes for busy kitchens a situational analysis of routine knowledge use in schools. *Science Communication*, 4(4) :478–510, 1983. (Cité page 41.)

Open Interface. Open interace website, 2008. URL http://www.openinterface.org/home/index.html. [Online ; accessed 22-May-2014]. (Cité page 52.)

J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Chapman and Hall, New York, 1983. (Cité page 31.)

JetBrains. Meta programming system. *http ://www.jetbrains.com/mps/*, 2013. URL http://www.topcased.org/. (Cité page 122.)

Jean-Marc Jezequel. Model driven design and aspect weaving. *Software & Systems Modeling*, 7(2) :209–218, 2008. ISSN 1619-1366. URL http://dx.doi.org/10.1007/s10270-008-0080-5. (Cité pages ix, 10, 16, 17, 69 et 86.)

Steven Kelly et Risto Pohjonen. Worst practices for domain-specific modeling. *IEEE Softw.*, 26(4) :22–29, Juillet 2009. ISSN 0740-7459. URL http://dx.doi.org/10.1109/MS.2009.109. (Cité page 27.)

Steven Kelly et Juha-Pekka Tolvanen. *Domain-Specific Modeling : Enabling Full Code Generation*. John Wiley & Sons, Inc., Hoboken, NJ, USA, Février 2007. ISBN 9780470249260. URL http://doi.wiley.com/10.1002/9780470249260. (Cité pages 11 et 101.)

Stuart Kent. Model driven engineering. Dans *Proceedings of the Third International Conference on Integrated Formal Methods*, IFM '02, pages 286–298, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-43703-7. URL http://dl.acm.org/citation.cfm?id=647983.743552. (Cité pages 7, 9, 14 et 21.)

Anneke G. Kleppe, Jos Warmer, et Wim Bast. *MDA Explained : The Model Driven Architecture : Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. ISBN 032119442X. (Cité page 10.)

Paul Klint. Rascal : "rascal - meta programming language ". *http ://www.rascal-mpl.org/*, 2010. URL http://www.rascal-mpl.org/. (Cité page 122.)

Werner A. König, Roman Rädle, et Harald Reiterer. Squidy : A zoomable design environment for natural user interfaces. Dans *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 4561–4566, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4. URL http://doi.acm.org/10.1145/1520340.1520700. (Cité page 52.)

Colin Tattersall & Rob Koper. Eml and ims learning design : from lo to la. Rapport technique, Educational Technology Expertise Centre, The Open University of the Netherlands, 2004. (Cité page 39.)

Rob Koper et Colin Tattersall. Preface to learning design : A handbook on modelling and delivering networked education and training. *Journal of Interactive Media in Education*, 2005(1), 2005. ISSN 1365-893X. URL http://www-jime.open.ac.uk/jime/article/view/2005-18. (Cité page 39.)

Rob Koper et René van Es. Modeling units of learning from a pedagogical perspective. Dans *IN R. MCGREAL (ED.), ONLINE*, pages 43–58, 2004. (Cité page 39.)

Stephen M. Kosslyn. Graphics and Human Information Processing : A Review of Five Books. *Journal of the American Statistical Association*, 80 (391) :499–512, 1985. (Cité page 31.)

Charles W. Krueger. Software reuse. *ACM Comput. Surv.*, 24(2) :131–183, Juin 1992. ISSN 0360-0300. URL `http://doi.acm.org/10.1145/130844.130856`. (Cité page 27.)

Pierre Laforcade. A domain-specific modeling approach for supporting the specification of visual instructional design languages and the building of dedicated editors. *Journal of Visual Languages & Computing*, 21(6) : 347 – 358, 2010. ISSN 1045-926X. URL `http://www.sciencedirect.com/science/article/pii/S1045926X10000492`. Special Issue on Visual Instructional Design Languages. (Cité page 20.)

Miguel a. Laguna et José M. Marqués. Feature Diagrams and their Transformations : An Extensible Meta-model. *2009 35th Euromicro Conference on Software Engineering and Advanced Applications*, pages 97–104, 2009. (Cité page 27.)

Bruce Landon. Hard choices for individual situations : Selecting a course management system. Dans P. Barker & S. Rebelsky, éditeur, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 1073–1085, Denver, Colorado, USA, 2002. Chesapeake, VA : AACE. (Cité page 37.)

C.F.J. Lange et M. R V Chaudron. Managing model quality in uml-based software development. Dans *Software Technology and Engineering Practice, 2005. 13th IEEE International Workshop on*, pages 7–16, 2005. (Cité page 24.)

Jill H Larkin et Herbert A Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive science*, 11(1) :65–100, 1987. (Cité page 78.)

Kung-Kiu Lau et Zheng Wang. Software Component Models. *IEEE Transactions on Software Engineering*, 33(10) :709–724, Octobre 2007. ISSN 0098-5589. (Cité page 27.)

X. Le Pallec, C.Od.M. Filho, R. Marvie, M. Nebut, et J.-C. Tarby. Supporting generic methodologies to assist ims-ld modeling. Dans *Advanced Learning Technologies, 2006. Sixth International Conference on*, pages 923–927, July 2006a. (Cité pages 13 et 94.)

X. Le Pallec, C.Od.M. Filho, R. Marvie, M. Nebut, et J.-C. Tarby. Supporting generic methodologies to assist ims-ld modeling. Dans *Advanced Learning Technologies, 2006. Sixth International Conference on*, pages 923–927, July 2006b. (Cité pages 49 et 140.)

Xavier Le Pallec et Grégory Bourguin. Ram3 : un outil dynamique pour le meta-object facility. *LMO2001 : Langages et Modèles à Objets, numéro spécial de la revue L'Objet, Hermes*, 2001 :79–94, 2001. (Cité page 113.)

Xavier Le Pallec et Sophie Dupuy-Chessa. Intégration de métriques de qualité des modèles et des méta-modèles dans l'outil ModX. Dans *Inforsid 2011*, Lille, France, Mai 2011. URL http://hal.inria.fr/hal-00823076. (Cité pages 119 et 122.)

Xavier Le Pallec et Sophie Dupuy-Chessa. Intégration de métriques de qualité des modèles et des langages dans l'outil ModX. Dans P. Merle P. Collet, éditeur, *Actes de la Conférence en IngénieriE du Logiciel (CIEL)*, page 6p., Rennes, France, Juin 2012. URL http://hal.archives-ouvertes.fr/hal-00757420. Session Réingénierie : papier court - Publication prochaine des actes (vol. journ. ECEASST - ref. ISSN). (Cité pages 119 et 122.)

Xavier Le Pallec et Sophie Dupuy-Chessa. Support for quality metrics in metamodelling. Dans *Proceedings of the Second Workshop on Graphical Modeling Language Development*, GMLD '13, pages 23–31, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2044-3. URL http://doi.acm.org/10.1145/2489820.2489825. (Cité pages 119 et 122.)

Xavier Le Pallec, Raphaël Marvie, José Rouillard, et Jean-Claude Tarby. A support to multi-devices web application. Dans *Adjunct Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 391–392, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0462-7. URL http://doi.acm.org/10.1145/1866218.1866235. (Cité page 55.)

Xavier Le Pallec, Jean-Claude Tarby, et José Rouillard. Le projet Miny : des interactions multimodales pour les applications Web. Dans *Ergo'IHM 2012*, Biarritz, France, Décembre 2012. URL http://hal.inria.fr/hal-00823072. (Cité page 55.)

R. Legendre. *Dictionnaire actuel de l'éducation*. Larousse, 1988. ISBN 9782033003432. URL http://books.google.com/books?id=bJ00GwAACAAJ. (Cité page 39.)

A. Lewandowski, G. Bourguin, et Jean-Claude Tarby. Tasks models for component contextualization. Dans *Proceedings of ICEIS 2007, the 9th International Conference on Enterprise Information Systems*, pages – , Funchal, Madère, Portugal, France, 2007. URL http://hal.archives-ouvertes.fr/hal-00731383. (Cité page 140.)

Daniel Liabeuf, Xavier Le Pallec, et José Rouillard. An empirical study on the anticipation of the result of copying and pasting among uml editors. Dans *8th System Analysis and Modelling Conference (SAM 2014)*, volume 8 de *Springer Lecture Notes in Computer Science (LNCS)*, Valencia, Spain, sep 2014. Springer. (Cité pages x, 98 et 104.)

Jean Marie Lions, Didier Simoneau, Gilles Pitette, et Imed Moussa. Extending opentool/uml using metamodeling : An aspect-oriented programming case study. Dans *UML ?02 2 nd Workshop on Aspect-Oriented Modeling with UML*, 2002. (Cité page 49.)

Jochen Ludewig. Models in software engineering - an introduction. *Informatik Forschung und Entwicklung*, 18(3-4) :105–112, 2004. ISSN 0178-3564. URL http://dx.doi.org/10.1007/s00450-004-0155-7. (Cité page 10.)

Marco Manca, Fabio Paterno, Carmen Santoro, et LucioDavide Spano. Generation of multi-device adaptive multimodal web applications. Dans Florian Daniel, GeorgeA. Papadopoulos, et Philippe Thiran, éditeurs, *Mobile Web Information Systems*, volume 8093 de *Lecture Notes in Computer Science*, pages 218–232. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40275-3. URL http://dx.doi.org/10.1007/978-3-642-40276-0_17. (Cité page 78.)

Raphael Marvie, Xavier Le Pallec, Jean-Claude Tarby, et Mirabelle Nebut. Defining and controlling modelling processes. Research Report LIFL 08, LIFL, Université Lille 1, dec 2005. (Cité pages 94 et 97.)

MetaCase. Metaedit+ : Domain-specific modeling (dsm) environment. *http ://www.metacase.com/products.html*, 2013. URL http://www.metacase.com/products.html. (Cité pages 13 et 123.)

J. Miller et J. Mukerji. Mda guide version 1.0.1. Rapport technique, Object Management Group (OMG), 2003. (Cité page 9.)

Daniel L. Moody. The "Physics" of Notations : Towards a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35 :756–779, 2009. (Cité pages 31 et 114.)

Daniel L. Moody et Jos van Hillegersberg. Evaluating the visual syntax of uml : An analysis of the cognitive effectiveness of the umlfamily of diagrams. Dans Dragan Gasevic, Ralf Lämmel, et Eric Van Wyk, éditeurs, *SLE*, volume 5452 de *Lecture Notes in Computer Science*, pages 16–34. Springer, 2008. ISBN 978-3-642-00433-9. (Cité page 3.)

Daniel Laurence Moody, Patrick Heymans, et Raimundas Matulevicius. Visual syntax does matter : improving the cognitive effectiveness of the *i** visual notation. *Requir. Eng.*, 15(2) :141–175, 2010. (Cité page 3.)

Pierre-Alain Muller, Frédéric Fondement, Benoît Baudry, et Benoît Combemale. Modeling modeling modeling. *Software and Systems Modeling*, 11(3) :347–359, 2012. ISSN 1619-1366. URL http://dx.doi.org/10.1007/s10270-010-0172-x. (Cité pages 10 et 11.)

Laurence Nigay. *Modalité d'interaction et multimodalité*. Hdr, Université de la Méditerranée - Aix-Marseille II, Décembre 2001. URL http://tel.archives-ouvertes.fr/tel-00004696. theses/2001/Nigay.Laurence theses/2001/Nigay.Laurence. (Cité pages 4 et 42.)

Thierry Nodenot, Pierre Laforcade, Xavier Le Pallec, et al. Visual design of coherent technology enhanced learning systems : a few lessons learned from cpm language. *Handbook of Visual Languages for Instructional Design : Theories and Practices*, pages 254–280, 2007. (Cité page 50.)

Obeo. Obeo designer : Methods and tools for architects. *http ://www.obeo.fr/pages/obeo-designer/en*, 2013. URL http://www.obeo.fr/pages/obeo-designer/en. (Cité pages 13 et 123.)

P.W. Oman, A.J. Bowles, R. Mount, G. Karam, D. Kalinsky, M. Tervonen, V. Bundonis, H. Fischer, M. Fish, D. Longshore, et N. Akiha. Case : analysis and design tools. *Software, IEEE*, 7(3) :37–43, May 1990. ISSN 0740-7459. (Cité pages ix et 1.)

Object Management Group OMG. Corba component model (ccm), version 4.0. *Online specifications*, april 2006. URL http://www.omg.org/spec/CCM/. (Cité page 22.)

Object Management Group OMG. Mof model to text transformation language (mofm2t), 1.0. *Online specifications*, jan 2008. URL http://www.omg.org/spec/MOFM2T/1.0/. (Cité page 18.)

Object Management Group OMG. Meta object facility (mof) 2.0 query/view/transformation, v1.2 (beta). *Online specifications*, may 2014. URL http://www.omg.org/spec/QVT/1.2/Beta/. (Cité page 14.)

Oracle. Java message service (jms), 2014. URL http://www.oracle.com/technetwork/java/jms/index.html. [Online ; accessed 22-May-2014]. (Cité page 52.)

OW2. Frascati, 2013. URL http://wiki.ow2.org/frascati/Wiki.jsp?page=FraSCAti. [Online ; accessed 22-May-2014]. (Cité page 52.)

Didier Paquelin. Planification versus Potentialisation. De la structuration des contenus à la structuration de la contenance. France, 2005. URL http://edutice.archives-ouvertes.fr/edutice-00001388. http ://sif2005.mshparisnord.org/pdf/Paquelin.pdf. (Cité page 41.)

Gilbert Paquette. Designing virtual learning centers. Dans HeimoH. Adelsberger, Betty Collis, et JanM. Pawlowski, éditeurs, *Handbook on Information Technologies for Education and Training*, International Handbooks on Information Systems, pages 249–271. Springer Berlin Heidelberg, 2002. ISBN 978-3-662-07684-2. URL http://dx.doi.org/10.1007/978-3-662-07682-8_16. (Cité page 38.)

Gilbert Paquette, Ileana De la Teja, Karin Lundgren-Cayrol, Michel Léonard, et Diane Ruelland. La modélisation cognitive, un outil de conception des processus et des méthodes d'un campus virtuel. *Journal of Distance Education*, 17(3), 2002. (Cité page 40.)

Cécile Paris, Jean-Claude Tarby, et Keith Vander Linden. A flexible environment for building task models. Dans *Proceedings of Human Computer Interaction conference IHM-HCI 2001*, Lille, sep 2001. (Cité page 140.)

Hugues Peeters et Philippe Charlier. Contributions à une théorie du dispositif. *Hermès*, 25 :15–23, 1999. (Cité page 41.)

Daniel Peraya. Le cyberespace : un dispositif de communication et de formation médiatisée. 2000. (Cité page 41.)

Philippe Perrenoud. La pratique pédagogique entre l'improvisation réglée et le bricolage. *Éducation et recherche*, 2 :198–212, 1983. (Cité page 41.)

Yvan Peter, Xavier Le Pallec, et Thomas Vantroys. Pedagogical Scenario Modelling, Deployment, Execution and Evolution. Dans Hershey Claus Pahl Editor, éditeur, *Architecture Solutions for E-Learning Systems*, pages 283–305. PA : IDEA Group Inc., 2007. URL http://hal.archives-ouvertes.fr/hal-00731372. (Cité pages ix, 40, 49, 50 et 60.)

Marian Petre. Uml in practice. Dans *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 722–731, Piscataway, NJ, USA, 2013. IEEE Press. ISBN 978-1-4673-3076-3. URL http://dl.acm.org/citation.cfm?id=2486788.2486883. (Cité pages 3, 22, 30 et 139.)

Pierre Rabardel et Pierre Pastré, éditeurs. *Modèles du sujet pour la conception : dialectiques, activités, développement*. Collection Travail & activité humaine / dirigée par François Daniellou, Gilbert de Tressac & Yves Schwartz. Octarès, Toulouse, 2005. ISBN 2-915346-25-9. URL http://opac.inria.fr/record=b1124423. (Cité page 4.)

Dorothée Rasseneur. *Saafir : un environnement support à l'appropriation d'une formation à distance par l'apprenant*. PhD thesis, Université du Maine, Novembre 2004. URL http://lium3.univ-lemans.fr/lium_d5/sites/default/files/rasseneur.pdf. (Cité page 38.)

Catherine Recanati. Characteristics of diagrammatic reasoning. Dans Daniel Kayser edited by Stella Vosniadou et Athanassios Protopapas, éditeurs, *Proceedings of EuroCogScio7, the european cognitive science conference*, the second european cognitive science conference, pages pp 510–515, Delphi, Grèce, Mai 2007. Lawrence Erlbaum Associates. URL http://hal.archives-ouvertes.fr/hal-00153328. (Cité page 24.)

Peter Rittgen. Collaborative modeling : Roles, activities and team organization. *IJISMD*, 1(3) :1–19, 2010. (Cité page 30.)

Peter Rittgen. End-user involvement and team factors in business process modeling. Dans *HICSS*, pages 180–189. IEEE Computer Society, 2012a. ISBN 978-0-7695-4525-7. (Cité page 30.)

Peter Rittgen. The role of editor in collaborative modeling. Dans *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 1674–1679, New York, NY, USA, 2012b. ACM. ISBN 978-1-4503-0857-1. URL http://doi.acm.org/10.1145/2245276.2232046. (Cité pages 29 et 30.)

José E. Rivera, Francisco Durán, et Antonio Vallecillo. Formal specification and analysis of domain specific models using maude. *Simulation*, 85(11-12) :778–792, Novembre 2009. ISSN 0037-5497. URL http://dx.doi.org/10.1177/0037549709341635. (Cité page 25.)

Mary Beth Rosson et John M. Carroll. Active Programming Strategies in Reuse. Dans Oscar M. Nierstrasz, éditeur, *ECOOP93 Object-Oriented*

Programming, volume 707 de *Lecture Notes in Computer Science*, pages 4–20. Springer Berlin Heidelberg, Kaiserslautern, Germany, Août 1993. ISBN 978-3-540-47910-9. (Cité page 27.)

José Rouillard, Xavier Le Pallec, Jean-Claude Tarby, et Raphaël Marvie. Facilitating the design of multi-channel interfaces for ambient computing. Dans Ray Jarvis et Cosmin Dini, éditeurs, *ACHI*, pages 95–100. IEEE Computer Society, 2010. ISBN 978-0-7695-3957-7. (Cité pages 44 et 140.)

José Rouillard, Jean-Claude Tarby, Xavier Le Pallec, et Raphael Marvie. From Metamodeling to Automatic Generation of Multimodal Interfaces for Ambient Computing. *International Journal On Advances in Software*, 3 (3&4), 2011. URL `http://hal.inria.fr/hal-00808273`. 1&2 publisher = IARIA 1&2 publisher = IARIA. (Cité pages 44 et 50.)

Jean rémy Falleri, Marianne Huchard, et Clémentine Nebut. C. : Towards a traceability framework for model transformations in kermeta. Dans *In : ECMDA-TW Workshop*, 2006. (Cité page 20.)

Jesus Sanchez-Cuadrado, Juan de Lara, et Esther Guerra. Bottom-up metamodelling : An interactive approach. Dans *Model Driven Engineering Languages and Systems*, volume 7590 de *Lecture Notes in Computer Science*, pages 3–19. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33665-2. (Cité page 50.)

A. Schauerhuber, W. Schwinger, E. Kapsammer, W. Retschitzegger, et M. Wimmer. A survey on aspect-oriented modeling approaches. Rapport technique, 2006. (Cité page 17.)

D.C. Schmidt. Guest editor's introduction : Model-driven engineering. *Computer*, 39(2) :25–31, Feb 2006. ISSN 0018-9162. (Cité pages 1 et 10.)

Daniel K Schneider, P Synteta, C Frété, et S Girardin. Conception and implementation of rich pedagogical scenarios through collaborative portal sites : clear focus and fuzzy edges. Dans *Proceedings of the International Conference on Open & Online Learning, ICOOL*, 2003. (Cité page 39.)

Riccardo Solmi. Whole platform. *http ://whole.sourceforge.net/*, 2012. URL `http://whole.sourceforge.net/`. (Cité page 122.)

Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer-Verlag, Wien, New York, 1973. ISBN 978-3-211-81106-1. URL `http://books.google.de/books?id=DK-EAAAAIAAJ`. (Cité page 1.)

Susan Leigh Star. Distributed artificial intelligence (vol. 2). Chapitre The Structure of Ill-structured Solutions : Boundary Objects and Heterogeneous Distributed Problem Solving, pages 37–54. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989. ISBN 0-273-08810-6. URL `http://dl.acm.org/citation.cfm?id=94079.94081`. (Cité page 20.)

Dominik Stein, Stefan Hanenberg, et Rainer Unland. A uml-based aspect-oriented design notation for aspectj. Dans *Proceedings of the 1st International Conference on Aspect-oriented Software Development*, AOSD '02, pages

106–112, New York, NY, USA, 2002. ACM. ISBN 1-58113-469-X. URL http://doi.acm.org/10.1145/508386.508399. (Cité page 15.)

Igor Steinmacher, AnaPaula Chaves, et MarcoAurelio Gerosa. Awareness support in distributed software development : A systematic review and mapping of the literature. *Computer Supported Cooperative Work (CSCW)*, 22(2-3) :113–158, 2013. ISSN 0925-9724. URL http://dx.doi.org/10.1007/s10606-012-9164-4. (Cité pages x et 107.)

Wilhelm Steinmüller. *Informationstechnologie und Gesellschaft : Einführung in die angewandte Informatik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1993. (Cité page 11.)

Harald Störrle. On the impact of layout quality to understanding uml diagrams : Diagram type and expertise. Dans Martin Erwig, Gem Stapleton, et Gennaro Costagliola, éditeurs, *VL/HCC*, pages 49–56. IEEE, 2012. ISBN 978-1-4673-0852-6. (Cité page 105.)

Harald Störrle. Making sense to modelers - presenting uml class model differences in prose. Dans Slimane Hammoudi, Luís Ferreira Pires, Joaquim Filipe, et Rui César das Neves, éditeurs, *MODELSWARD*, pages 39–48. SciTePress, 2013. ISBN 978-989-8565-42-6. (Cité page 105.)

Jean-Claude Tarby. *Gestion automatique du dialogue homme-machine à partir de spécifications conceptuelles 1 microfiche*. PhD thesis, Toulouse 1, Grenoble, 1993. URL http://opac.inria.fr/record=b1061593. Th. : informatique. (Cité page 140.)

Jean-Claude Tarby, Xavier Le Pallec, Raphael Marvie, et Mirabelle Nebut. Processus de modéisation incrémental pour le développement d ?applications interactives basées sur pac. Dans *IDM & IHM : Ingénierie Dirigée par les Modèles et Interaction Homme-Machine*, 18e Conférence Francophone sur l'Interaction Homme-Machine, apr 2006. (Cité pages 50 et 95.)

Peri Tarr, Harold Ossher, William Harrison, et Stanley M. Sutton, Jr. N degrees of separation : Multi-dimensional separation of concerns. Dans *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pages 107–119, New York, NY, USA, 1999. ACM. ISBN 1-58113-074-0. URL http://doi.acm.org/10.1145/302405.302457. (Cité page 9.)

P. Tchounikine. Introduction à l'ingénierie des eiah. Dans Grandbastien M. et Labat J.M., éditeurs, *Environnements informatiques pour l'apprentissage humain*, pages 141–160. Hermes, 2006. ISBN 2-7462-1171-8. (Cité page 37.)

Thot. Thot cursus, 2014. URL http://thot.cursus.edu/. [Online ; accessed 22-May-2014]. (Cité page 38.)

Christian Thum, Michael Schwind, et Martin Schader. Slim a lightweight environment for synchronous collaborative modeling. Dans Andy Schurr et Bran Selic, éditeurs, *Model Driven Engineering Languages and Systems*, volume 5795 de *Lecture Notes in Computer Science*,

pages 137–151. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04424-3. URL http://dx.doi.org/10.1007/978-3-642-04425-0_11. (Cité pages 29 et 108.)

Juha-Pekka Tolvanen, Risto Pohjonen, et Steven Kelly. Advanced tooling for domain-specific modeling : Metaedit+. Dans *Sprinkle, J., Gray, J., Rossi, M., Tolvanen, JP (eds.) The 7th OOPSLA Workshop on Domain-Specific Modeling, Finland*, 2007. (Cité page 49.)

Antonio Vallecillo et Martin Gogolla. Typing model transformations using tracts. Dans Zhenjiang Hu et Juan Lara, éditeurs, *Theory and Practice of Model Transformations*, volume 7307 de *Lecture Notes in Computer Science*, pages 56–71. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-30475-0. URL http://dx.doi.org/10.1007/978-3-642-30476-7_4. (Cité page 20.)

Gilles Vanwormhoudt, Olivier Caron, et Bernard Carré. Aspectual Templates in UML. Rapport technique, Avril 2013. URL http://hal.archives-ouvertes.fr/hal-00846060. (Cité pages 15, 16 et 17.)

Eelco Visser. Spoofax : The spoofax language workbench. *http ://strategoxt.org/Spoofax*, 2013. URL http://strategoxt.org/Spoofax. (Cité page 122.)

Willemien Visser. Use of episodic knowledge and information in design problem solving. *Design Studies*, 16(2) :171 – 187, 1995. ISSN 0142-694X. URL http://www.sciencedirect.com/science/article/pii/0142694X94000082. Analysing Design Activity. (Cité page 27.)

Jorg von Engelhardt. *The Language of Graphics*. PhD thesis, Universiteit van Amsterdam, Institute for Logic, Language and Computation, September 2002. (Cité page 31.)

Peter Wegner. Why interaction is more powerful than algorithms. *Commun. ACM*, 40(5) :80–91, Mai 1997. ISSN 0001-0782. URL http://doi.acm.org/10.1145/253769.253801. (Cité page 42.)

Max Wertheimer. Untersuchungen zur Lehre von der Gestalt. II. *Psychologische Forschung*, 4(1) :301–350, 1923. ISSN 0340-0727. (Cité page 102.)

Wikipedia. Comet (programming) — wikipedia, the free encyclopedia, 2014a. URL http://en.wikipedia.org/w/index.php?title=Comet_(programming)&oldid=607406104. [Online ; accessed 22-May-2014]. (Cité page 53.)

Wikipedia. Cross-origin resource sharing — wikipedia, the free encyclopedia, 2014b. URL http://en.wikipedia.org/w/index.php?title=Cross-origin_resource_sharing&oldid=609873589. [Online ; accessed 12-July-2014]. (Cité page 53.)

D.S. Wile. Abstract syntax from concrete syntax. Dans *Software Engineering, 1997., Proceedings of the 1997 International Conference on*, pages 472–480, May 1997. (Cité page 11.)

Laurent Wouters. xowl : xowl infrastucture. *http ://xowl.codeplex.com/*, 2012. URL http://xowl.codeplex.com/. (Cité pages 14, 49 et 123.)

Eric S.K Yu et Mylopoulos John. From e-r to a-r modelling strategic actor relationships for business process reengineering. Dans *Entity-Relationship Approach at ER 94 Business Modelling and Re-Engineering*, volume 881 de *Lecture Notes in Computer Science*, pages 548–565. Springer Berlin Heidelberg, 1994. ISBN 978-3-540-58786-6. URL http://dx.doi.org/10.1007/3-540-58786-1_101. (Cité page 21.)

Écoutin Eric & Guidon Jacques. Comparatif technico-pédagogique de plates-formes pour la formation ouverte et à distance. Rapport technique, Algora, Novembre 2000. URL http://ressourcesv2.e-motive.com/virtual/30/Documents/pdf/pf2000.pdf. (Cité page 38.)

Ce document a été préparé à l'aide de l'éditeur de texte GNU Emacs et du logiciel de composition typographique LaTeX 2ε.

Titre Centrer l'Ingénierie Dirigée par les Modèles sur l'humain

Résumé L'ingénierie Dirigée par les Modèles est une démarche conception logicielle apparue au début des années 2000 qui prône des chaînes de conception/développement où les modèles sont les principaux ingrédients. La majorité des travaux scientifiques liée à l'IDM portent sur les opérateurs de composition / transformation de modèles, de création de langages spécifiques et de génération de code. J'ai, pour ma part, concentrer mes travaux sur l'utilisabilité de ces opérateurs et sur l'activité de modélisation telle qu'elle a lieu en IDM. Cette HDR présente les différentes contributions que j'ai pu apporter pour cette problématique. Trois aspects principaux ont été étudiés pendant mes onze années de recherche. Le premier concerne le mapping vertical, en particulier son problème de traçabilité mais aussi son inadaptation à certaines projections technologiques. Le second concerne l'activité de modélisation au travers des assistants de modélisation, de la réutilisation d'éléments de diagrammes et de la modélisation collaborative. Enfin la troisième se focalise sur les notations visuelles et leur efficacité cognitive.

Mots-clés Ingénierie Dirigée par les Modèles, Approche Centrée sur l'Humain, Modélisation Logicielle, Notations Visuelles

Title Orienting the Model Driven Engineering on Human

Abstract The Model Driven Engineering (MDE) is a software engineering method which is appeared in the early 2000s. It promotes the use of design/development chains where models are the fundamentals. Most of scientific works related to MDE concern model composition/transformation operators, creation of specific languages and code generation. For my part, I focused my work on the usability of these operators and the activity of modeling as it occurs in MDE. This HDR presents my different contributions to this topic. Three main aspects have been studied during the period described in this report. The first one concern the vertical mapping, and particularly its traceability problem but also the fact that it is totally unadapted to some technologic projections. The second one refers to the activity of modeling with a focus on related subproblems : modeling assistant, the reuse of diagram elements and collaborative modeling. Finally, the third aspect relates to visual notations and their cognitive effectiveness.

Keywords Model Driven Engineering, Human-Centred Approach, Software Modeling, Visual Notations