Vers une approche centrée humain pour la définition de langages de modélisation graphiques

Sophie Dupuy-Chessa[1], Benoît Combemale[2], Marie-Pierre Gervais[3], Thierry Nodenot[4], Xavier Le Pallec[5], Laurent Wouters[6]

1. Université de Grenoble Alpes, Laboratoire d'informatique de Grenoble
F-38000 Grenoble, France
Sophie.Dupuy@imag.fr
2. Inria et Université de Rennes 1, IRISA Campus de Beaulieu, 35042 Rennes Cedex benoit.combemale@irisa.fr
3. Laboratoire d'Informatique de Paris 6, Université Paris Ouest Nanterre La Défense,
4 Place Jussieu 75252 Paris Cedex 05marie-pierre.gervais@lip6.fr
4. Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour,
IUT de Bayonne, 2 allée du Parc Montaury, 64600 Anglet
Thierry.Nodenot@iutbayonne.univ-pau.fr
5. Laboratoire d'Informatique Fondamentale de Lille
Bâtiment M3, Cité Scientifique, 59655 Villeneuve d'Ascq Cedex
xavier.le-pallec@univ-lille1.fr
6. CEA LIST
CEA Saclay Nano-INNOV 91191 Gif sur Yvettes
laurent.wouters@cea.fr

RESUME. *Avec la complexification des systèmes d'information (systèmes ubiquitaires, entreprises ouvertes, etc.), de nombreux nouveaux langages de modélisation sont proposés. Face à ce développement de langages spécifiques, leur qualité devient un enjeu important pour la modélisation des systèmes d'information. Cet article traite de ce problème en suggérant une approche centrée utilisateur pour la création de langages. Nos propositions s'appuient sur nos travaux antérieurs issus de différents projets, mais constituent un tout cohérent qui nous permet de montrer l'intérêt et la faisabilité de l'approche.*

ABSTRACT. *The increasing complexity of information systems (ubiquitous systems, open enterprises, etc.) calls for the introduction of always new modeling languages. However, the development of new domain-specific languages makes the question about their quality an important issue for the modeling of information systems. This article deals with this issue by suggesting a user-centred approach for language creation. Our proposals rely on our past work from different projects, but constitute a consistent whole, which allows us to show the approach interest and feasibility.*

MOTS-CLES : qualité, ingénierie des langages, définition centrée utilisateur

KEYWORDS: quality, modeling langage, user centred definition

1. Introduction

Les modèles, qui servent à comprendre et à représenter les systèmes, prennent une importance accrue face à la complexification des systèmes d'information (SI). Pour les gérer, l'ingénierie dirigée par les modèles (IDM) développe des techniques et des outils. Ainsi, il est maintenant possible de créer de nouveaux langages, spécifiques à un domaine (DSL). Ces nouveaux langages, s'ils ne sont pas définis et évalués avec le plus grand soin, risquent d'amener à la production de modèles inutiles ou inacceptables par les utilisateurs et donc à la remise en cause complète de ces langages de modélisation. Pour limiter les risques, il est nécessaire de s'intéresser à la manière dont ils sont créés. Dans cet article, nous nous intéressons à une approche centrée humain pour la définition des langages de modélisation en nous focalisant sur les langages graphiques. Nous basons notre réflexion sur l'étude des travaux existants en matière de qualité des langages et nous montrons comment, à travers différents projets, nous avons abordé la création de langages du point de vue de leurs concepteurs et de leurs futurs utilisateurs.

La section 2 présente un survol de la littérature existante sur la qualité des langages graphiques de modélisation. La section 3 décrit nos contributions pour la création de langages graphiques. Pour conclure, nous synthétisons en section 4 nos contributions et identifions quelques perspectives de ce travail.

2. Travaux existants

Dans cet article, nous utilisons une définition en intention d'un langage (description des propriétés communes aux instances possibles). Ainsi un langage est défini par une syntaxe abstraite, une syntaxe concrète et une sémantique. La syntaxe abstraite capture le vocabulaire et la taxonomie (i.e. les concepts) du langage (Fondement et Baar 2005) alors que la syntaxe concrète décrit la notation, c'est-à-dire la représentation des éléments du langage. En IDM, la syntaxe abstraite se décrit par un métamodèle ; la syntaxe concrète peut être graphique ou textuelle. Une séparation claire entre syntaxe abstraite et syntaxe concrète est une technique pour gérer la complexité de la définition d'un langage de modélisation car elle permet de définir les éléments d'un langage indépendamment de leur représentation. La description du langage est complétée par une sémantique. Nous n'aborderons pas ici la qualité de la sémantique, largement traitée par la thématique des langages formels et pour laquelle 4 formes de définition sont recommandées (Kleppe 2007) : dénotationnelle, opérationnelle, translationnelle et pragmatique.

Les travaux existants abordent souvent la qualité des langages de modélisation de manière globale. Dans ce cas, les expériences ou les cadres théoriques s'intéressent au langage indépendamment de ses éléments constitutifs (syntaxes abstraite, concrète et sémantique). Une autre approche consiste à étudier la qualité des composants du langage en se basant sur l'hypothèse que la qualité globale du langage est influencée par celle de ces composants. Ces deux approches sont présentées dans le survol de la bibliographie qui suit.

2.1. Cadre global de qualité

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

De manière complémentaire à ces travaux, des référentiels étudient les langages dans leur globalité (syntaxes et sémantique). Ils s'inscrivent dans l'approche sémiotique pour la qualité des modèles. Ainsi (Krogstie 2003) identifie 5 caractéristiques pour un langage de modélisation : 1) l'adéquation au domaine ; 2) l'adéquation aux connaissances des acteurs (i.e. les concepteurs de modèles) ; 3) l'adéquation à la capacité d'externaliser les connaissances : il ne doit pas y avoir d'assertions dans les connaissances explicites des participants qui ne peuvent être exprimées dans le langage. ; 4) l'adéquation à la capacité de compréhension des parties prenantes; et 5) l'adéquation à l'interprétation par des acteurs techniques (les outils de modélisation, de simulation, de preuve etc).

Les travaux que nous venons de citer apportent une caractérisation des langages. Pour autant, la qualité peut s'aborder sous un angle différent, qui est celui des différents composants des langages, restreints dans cet article et comme mentionné dans les sections précédentes à la syntaxe abstraite et la syntaxe concrète graphique. Etudier la qualité selon ces deux aspects permet de cibler au mieux les questions de recherche ouvertes.

2.2. Qualité de la syntaxe abstraite

La qualité de la syntaxe abstraite est relative au métamodèle et se base sur son évaluation. Ainsi les travaux décrits ci-dessous ont cherché à établir un lien entre la qualité du métamodèle et celle du langage.

Dans (Mohagheghi et Aagedal 2007), l'hypothèse est qu'un métamodèle complexe conduit à un pouvoir d'expression plus grand et donc à des modèles plus petits. Suivant la même approche, (Rossi et Brinkkemper 1996) propose une méthode de calcul de la complexité conceptuelle théorique. Dans ce cas, l'hypothèse (non démontrée) est qu'il existe une dépendance intrinsèque entre les métamodèles et la facilité d'apprentissage du langage : « the more complex a metamodel, the harder the method is to learn ». Ce travail a permis de comparer plusieurs langages de modélisation orientés objet à partir de leur métamodèle et conclut qu'ils

deviennent plus complexes au fil du temps. En utilisant les mêmes règles de calcul, Siau et Cao cité dans (Krogstie 2003), aboutissent à des résultats similaires : UML est de 2 à 11 fois plus complexe que n'importe quel autre langage de modélisation orienté-objet.

2.3. Qualité de la syntaxe concrète graphique

Comme introduit précédemment, la syntaxe concrète d'un langage de modélisation peut être textuelle ou graphique. Dans le contexte de cet article, nous nous concentrerons uniquement sur les syntaxes graphiques qui sont utilisées par les concepteurs de modèles conceptuels.

Les travaux étudiant la qualité de la syntaxe concrète s'intègrent dans l'approche de la sémiologie graphique. Ils considèrent que l'objectif d'un modèle est de transcrire graphiquement une information. Dans ce cadre, nous pouvons citer les travaux de Bertin (Bertin 1967) qui propose d'exploiter les capacités du système visuel humain à percevoir la profondeur des objets que nous voyons. Par exemple, la taille a un fort pouvoir d'ordonnancement (profondeur donc) : dans un diagramme de classes, l'augmentation significative de la taille de la police de caractères utilisée pour l'affichage du nom des paquetages, permettrait de mettre en avant les paquetages présents.

Par la suite, le principal travail pour appréhender la qualité d'une notation graphique est la Théorie de la Physique des notations (Moody 2009) qui donne 9 principes pour concevoir des notations visuelles cognitivement efficaces. Nous citerons à titre d'exemples la transparence sémantique, qui traite en partie du lien entre syntaxe concrète et sémantique, et la discriminabilité perceptive qui est entièrement focalisée sur la notation :

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

Si on se réfère à la théorie de compréhension des graphes de Pinker (Pinker 1990), les critères d'évaluation des notations visuelles (en Ingénierie Logicielle) de Moody interviennent soit au niveau perceptif (p. ex., discriminabilité perceptuelle), soit au niveau cognitif (p. ex., transparence sémantique, clarté sémiotique) soit aux deux (p. ex., gestion de la complexité). L'intérêt de cette théorie, qui synthétise de nombreux travaux en visualisation, est d'indiquer que des mécanismes liés à ces critères sont «innés» et d'autres sont fonction de l'histoire du lecteur.

Comme expliqué dans (Moody 2009), les travaux sur la physique des notations sont complémentaires des travaux plus anciens de T. Green qui portent non pas sur une évaluation détaillée des seules notations visuelles mais plutôt sur des critères d'analyse de l'utilisabilité des environnements visuels exploitant de telles notations (Green & Petre 1996), (Green & Blackwell 2003), (Green et all 2006). Ainsi, les travaux de Green proposent quatorze dimensions d'analyse que des non spécialistes peuvent percevoir et discuter lorsqu'ils utilisent les environnements visuels mis à leur disposition. Certains critères de D. Moody et T. Green se recoupent cependant (la dimension « Abstraction » de T. Green est un élément important du principe de « Gestion de la complexité » de Moody ; la dimension « Dépendance cachée » de T. Green est à rapprocher du principe de « Transparence sémantique » de D. Moody, etc.). Mais le recouvrement n'est que partiel car pour T. Green, la notation visuelle n'est que l'un des éléments d'un puzzle visant à produire des instruments adaptés à l'activité et aux spécificités des usagers (Fig. 1) : notation et environnement doivent être intégrés comme le montre la partie droite de la figure et non juxtaposés.

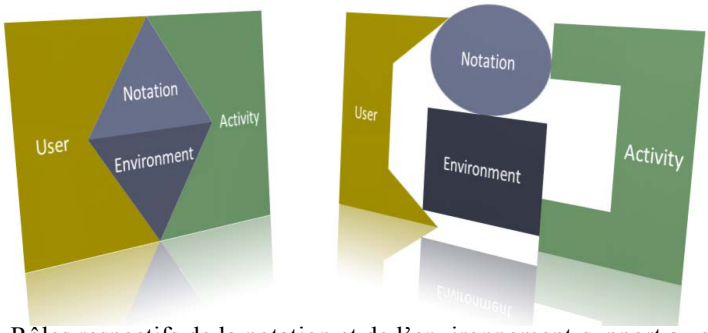

Figure 1. Rôles respectifs de la notation et de l'environnement-support au service de l'activité de modélisation.

Aussi, par la suite, nous nous appuierons également sur trois dimensions des travaux de T. Green qui nous paraissent influer fortement sur la capacité qu'aura un utilisateur à explorer ou rechercher, encoder, modifier des éléments dans un modèle via un environnement visuel :

- La visibilité : la capacité de la notation et de l'environnement à rendre compte des différentes facettes des modèles.
- L'évaluation progressive : la capacité de la notation et de l'environnement à supporter une évaluation progressive des modèles produits.

- L'engagement prématuré : capacité à éviter au concepteur de devoir prendre des décisions de modélisation prématurées compte tenu de son rôle et de l'objectif de son activité.

2.4. Synthèse

Au travers de notre revue de la littérature sur la qualité des langages (syntaxe abstraite et syntaxe concrète graphique), nous pouvons observer que le concepteur d'un langage de modélisation a de nombreuses contraintes à prendre en compte : l'adéquation au domaine, à la connaissance des acteurs, s'adapter aux limitations de l'éditeur potentiel... En plus de ces contraintes générales, il ou elle doit aussi prendre garde à ne pas rendre trop complexe la syntaxe abstraite au fil du temps et être au fait de l'importance de la syntaxe concrète. Malheureusement, le point de vue du concepteur désireux de créer un langage de modélisation de qualité n'est pas pris en compte par les précédents travaux. Ainsi la question de la disponibilité de techniques et outils permettant aux concepteurs de langages de modélisation d'être performants et efficaces reste ouverte. C'est un point que nous aborderons dans la section suivante.

3. Vers une approche centrée humain pour la création de langages

Nous abordons la qualité des langages de modélisation en considérant les deux facettes, syntaxes abstraite (partie 3.1) et concrète (partie 3.2), ainsi que les liens qu'elles peuvent avoir (partie 3.3). Cette section présente des travaux issus de différents projets, non coordonnés, auxquels les auteurs ont pu participer. Ils ont néanmoins tous en commun d'avoir suivi une approche centrée sur le point de vue de l'humain pour la définition de nouveaux langages.

3.1. Définition de la syntaxe abstraite

La qualité de la syntaxe abstraite est relative au métamodèle. Pour le définir, les experts qui souhaitent capturer précisément le périmètre de leur domaine doivent maîtriser deux formalismes différents: un langage orienté-objet aligné sur MOF pour modéliser la structure du domaine; et un langage supportant la logique du premier ordre aligné sur OCL pour ajouter les contraintes nécessaires précisant la structure des modèles attendus. L'utilisation conjointe des deux langages de métamodélisation représente un défi majeur qui n'est à ce jour pas supporté par des méthodologies ou des bonnes pratiques. Ce dernier point est particulièrement difficile dans le cas de l'évolution de l'une ou l'autre des vues. Un cas notable de l'évolution de la norme UML à l'OMG est la classe *AssociationEnd* qui a disparu du métamodèle après la version 1.4 en 2003. Pourtant, jusqu'à la version 2.2, sorti en 2009, il a subsisté des expressions OCL se référant à cette classe (Selic 2008). De la même manière, la spécification d'OCL 2.2 dépend de MOF 2.0, mais une section particulière de la spécification définissant la liaison entre MOF et OCL (OCL2, p.169, 2010) fait usage de la classe *ModelElement* qui n'a existé que jusqu'à la version 1.4 de MOF.

Pour comprendre et essayer d'améliorer l'utilisation conjointe de ces deux formalismes, nous avons mené une étude empirique sur 1262 contraintes issues de 33 métamodèles (Cadavid et al, 2012). Cette étude nous a amené à définir formellement de nouvelles métriques (p.ex., taille du métamodèle, nombre d'invariant (total et par contexte), complexité des invariants par rapport au métamodèle, complexité des invariants par rapport au langage OCL, etc.) dédiées à l'évaluation de la qualité d'un métamodèle décrit en utilisant conjointement MOF et OCL, et permettent ainsi de révéler différents aspects du couplage et de la dispersion. L'ensemble des métriques a été intégré dans un outil permettant l'analyse automatique d'un métamodèle fourni en entrée. Nous observons sur les données prises en compte dans l'étude empirique que les experts tendent vers une utilisation dans leurs contraintes d'un petit ensemble des concepts de leur domaine (i.e., la majorité des contraintes utilise moins de 5 concepts du domaine). Malgré ce faible couplage, nous observons également que l'utilisation conjointe de MOF et OCL soulève des problèmes de consistance. 422 contraintes parmi les 1262 n'ont pas pu être analysées à cause d'un problème de lien avec la structure du métamodèle décrit à l'aide de MOF. Bien que OCL soit devenu un standard de facto pour la définition de contraintes sur une structure orientée-objet décrite à l'aide de MOF, ce n'est pas son objectif initial. En conséquence, nous observons également qu'une partie très significative du langage n'est jamais utilisée dans des contraintes d'un métamodèle : 10 sur les 22 concepts d'OCL ne sont jamais utilisés dans notre ensemble de données. Cette étude confirme la difficulté pour un expert d'un domaine à utiliser conjointement MOF et OCL.

Capturer précisément un domaine au sein d'un langage de modélisation assure la pertinence des modèles construits à partir de ce langage vis-à-vis du domaine considéré. Néanmoins nous avons pu constater la difficulté pour un expert d'un domaine à définir une syntaxe abstraite avec les techniques de l'IDM (MOF et OCL). Une perspective immédiate est donc d'offrir une base de recommandations et un outillage aidant les experts à utiliser conjointement MOF et OCL pour capturer précisément leur domaine dans un métamodèle.

3.2. Définition de la syntaxe concrète graphique

La syntaxe concrète graphique s'intéresse à définir des représentations adéquates pour les concepteurs. Nous avons d'une part, cherché à mesurer certains critères de qualité lors de la création de la syntaxe concrète graphique ; d'autre part, nous étudions le caractère social d'une notation c'est-à-dire la capacité d'une notation à permettre à des acteurs de la modélisation de travailler ensemble.

Pour mesurer la qualité de la syntaxe concrète graphique, nous cherchons à identifier des métriques qui aideront les concepteurs de langages à penser et à évaluer leur syntaxe. ModX[1], un méta-éditeur existant développé depuis plusieurs années au LIFL (et présenté dans la partie 3.3) a été étendu pour ajouter des

[1] Site web de ModX, http://www.lifl.fr/modx

métriques calculables sur toute syntaxe concrète graphique définie dans ModX (Le Pallec et Dupuy-Chessa 2011) (le Pallec et Dupuy-Chessa 2012). Les principales propriétés «visuelles» identifiables dans la physique des notations ont été inventoriées (Le Pallec et Dupuy-Chessa 2012). Ce travail nous montre qu'il est très difficile, voire impossible pour certains critères, de définir des mesures automatiques de qualité (p.ex., transparence sémantique, adaptation cognitive, gestion de la complexité). Les métriques ne peuvent donc être envisagées que comme des indicateurs partiels et doivent forcément être couplées avec des évaluations expérimentales. Elles peuvent néanmoins aider les concepteurs de langages à proposer une syntaxe concrète de meilleure qualité.

La syntaxe concrète d'un langage de modélisation peut aussi s'analyser dans ce qu'elle entretient avec les acteurs qui la manipulent au service d'une tâche donnée, dans les ponts qu'elle permet d'établir entre des acteurs différents tant dans les objectifs que dans les compétences en modélisation. Nous avons acquis l'expérience suivante lors de la définition et de la mise en œuvre de deux langages graphiques de modélisation :

- le langage offert par l'environnement CPM (Laforcade, Nodenot et al. 2005), (Nodenot, Laforcade et al. 2007), (Nodenot 2008)
- et le langage offert par l'environnement WindMash (Luong et al. 2011), (Luong et al. 2012)

Ces langages nous ont ainsi conduit à considérer que la notation visuelle et plus généralement que l'environnement support à l'exploitation d'une notation visuelle sont des éléments clés pour articuler les perspectives de modélisation de différents acteurs appartenant à des mondes sociaux différents mais travaillant sur un projet commun.

Dans ces travaux, il s'agissait d'amener des enseignants et des informaticiens à se coordonner malgré leurs points de vue différents, de leur permettre de construire des compréhensions communes sans perdre la diversité de leurs points de vue. Lors de nos expérimentations, nous avons pu constater qu'un effort tout particulier est nécessaire pour définir les notations mises à disposition de ces acteurs afin de leur permettre de décrire/manipuler/interroger des objets frontières de modélisation que S.L. Star définit comme : 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), (Star 2010). Supportée par la notation, la flexibilité d'interprétation d'un objet-frontière peut alors devenir le support à des traductions hétérogènes, comme dispositif d'intégration des savoirs, comme médiation dans le processus de coordination d'experts et de non experts (Trompette et Vinck 2009).

Les retours d'expériences nous ont amenés aux conclusions suivantes sur les capacités que doivent avoir les notations et l'environnement de modélisation exploitant ces notations complémentaires des objets-frontières.

Concernant la capacité à rendre compte des différentes facettes des modèles (cf. Visibilité), l'environnement-support doit faciliter la juxtaposition des représentations complémentaires d'un objet-frontière. D'une part, les relations qu'entretiennent les

diverses notations d'un même concept doivent être explicitées lors de la définition des syntaxes concrètes mais l'environnement de modélisation doit en plus concrétiser matériellement ces relations pour les utilisateurs lorsque les deux représentations sont visibles et juxtaposées. Nous avons ainsi dû intégrer dans l'environnement WindMash des mécanismes de notification spécifiques que chaque éditeur de modèles devait exploiter pour concrétiser les liens entre objets frontières : contenus à valoriser rendus disponibles au niveau de la conception des interfaces-usager via des opérations de drag'n drop, composants d'interface devenant des lignes de vie lors de la spécification des interactions possibles entre un utilisateur et ces composants d'interface (voir *http://youtu.be/3uxR8euHPwM?hd=1*)**.**

La syntaxe concrète doit supporter une évaluation progressive des modèles produits (cf. Evaluation Progressive). Les objets-frontières représentant des enjeux bien particuliers dans le processus de modélisation, il est essentiel que la syntaxe concrète des langages facilite l'évaluation des modèles par les concepteurs au fur et à mesure de l'avancement de l'activité. Dans le cas de l'environnement WindMash et des Live Sequence Charts, des éléments de la notation vont porter les résultats de l'exécution partielle d'un modèle, facilitant ainsi l'évaluation de la pertinence du travail de conception d'un objet-frontière donné. Si l'exécutabilité des modèles est bien sûr un prérequis des environnements-support correspondants, il est important de noter que l'environnement exécutant ces modèles doit pouvoir modifier à la volée/valoriser/visualiser les éléments dont l'état a changé, la dynamique du cycle modélisation/exécution des modèles étant donc portée par la notation.

Enfin il est important d'éviter au concepteur de devoir prendre des décisions de modélisation prématurées compte tenu de son rôle et de l'objectif de son activité (cf. Engagement Prématuré). Pour pouvoir exécuter des modèles, un environnement-support doit s'appuyer sur des modèles totalement définis, condition rarement satisfaite surtout lorsque les concepteurs ne sont pas des spécialistes en modélisation mais des spécialistes de leur domaine (par exemple des enseignants amenés à coopérer avec des informaticiens pour concevoir des applications éducatives). Plutôt que de forcer les concepteurs à prendre des décisions dont ils ne perçoivent ni les conséquences ni l'intérêt, les travaux que nous venons de décrire ont montré qu'il était efficace d'affecter des valeurs par défaut aux éléments de modèles non précisés explicitement par le concepteur à condition que ces éléments portant des valeurs par défaut soient modifiables via le système de notation proposé au concepteur. C'est donc au cours de l'exécution de modèles que les acteurs pourront découvrir, par l'expérience, ces éléments de modélisation complémentaire : la notation devient alors le support à un processus d'ajustement entre le comportement perçu du modèle et le comportement à atteindre construit par un processus de résolution de problème.

Qu'il s'agisse de la définition des caractéristiques perceptives ou des caractéristiques cognitives d'une notation graphique, nous retenons le rôle des environnements-supports (notamment des méta-éditeurs) pour mesurer la qualité des notations et procéder aux nécessaires ajustements requis pour satisfaire aux différents critères de qualité énoncés. Ces ajustements concernent tout autant la notation proprement dite que l'articulation entre notations comme moyen d'établir des ponts entre les préoccupations de différents acteurs.

3.3. Influence entre syntaxes concrète et abstraite

La définition d'un langage de modélisation dédié passe par l'élaboration de sa syntaxe abstraite et de sa syntaxe concrète. Quelles sont les dépendances entre ces deux types de syntaxes, quelles sont leurs mises en relation et quelle est la répercussion de l'une sur l'autre ? Cette section étudie l'impact de cette mise en relation sur une approche orientée humain de la construction de langages de modélisation et des éditeurs les supportant.

Les artefacts intervenants dans cette production sont présentés sur la figure 2. Un éditeur est généralement construit à partir i) du modèle du domaine métier (c'est-à-dire le métamodèle représentant la syntaxe abstraite), ii) d'un modèle de la syntaxe concrète (décrivant par exemple des représentations textuelles ou graphiques) et iii) d'un modèle de mapping permettant de configurer le lien entre les deux modèles précédents.

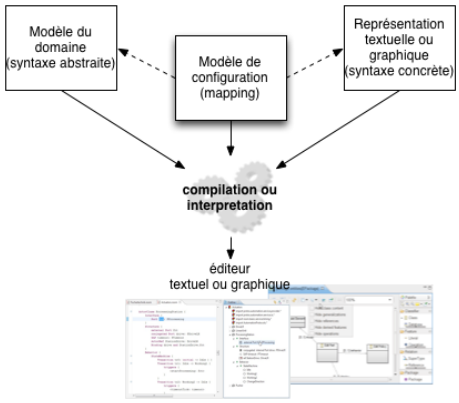

Figure 2. Définition d'un éditeur dédié à un domaine.

La méthodologie traditionnellement (Ráth et al. 2010) utilisée dans la communauté IDM pour produire un tel éditeur, que nous appellerons par la suite « approche métamodèle », est la suivante :

1. Eliciter les concepts du domaine et construire la syntaxe abstraite du langage (le métamodèle). Cette étape peut être réalisée par des entretiens avec les experts du domaine pour mettre en lumière les concepts du domaine et leurs relations ;
2. Produire la syntaxe concrète du langage par l'association de symboles aux éléments de la syntaxe abstraite. Ces symboles peuvent être visuels dans le cas d'un langage de modélisation graphique, ou textuel si un langage textuel est recherché.

Cette méthodologie peut être simplifiée pour le concepteur du langage par l'utilisation d'outils dédiés à cela, et généralement inclus dans des outils appelés « meta-tools » (Ráth et al. 2010), (Grundy 2008).

La caractéristique de cette méthodologie est qu'elle met l'accent sur la syntaxe abstraite du langage, ce qui peut être un avantage ou un inconvénient suivant les cas. Ce focus sur la syntaxe abstraite permet de répondre au mieux aux attentes en terme de manipulation des données qui seront décrites dans ce langage (requêtes, transformations, etc.). En revanche, la syntaxe concrète est le parent pauvre de cette approche et peut être limitée par les choix faits lors de la production de la syntaxe abstraite. Evans et al. (Evans et al., 2009) montrent que les concepteurs sont peu enclins à apporter des modifications au métamodèle même lorsque la construction de la syntaxe concrète amène à sa remise en cause. C'est alors la syntaxe concrète qui est adaptée pour correspondre au métamodèle, parfois au détriment des concepteurs. Or ce sont eux qui manipulent la syntaxe concrète. Une autre limite de cette approche est le couplage fort entre les deux syntaxes, qui ne permet qu'une simple mise en correspondance entre les deux syntaxes, comme illustré dans (Ráth et al. 2010).

Aussi Wouters et al. explorent une approche inverse, appelée « approche notationnelle » et mettant l'accent sur la syntaxe concrète dans le cas de langages graphiques (Wouters et al., 2013) :

1. Produire la syntaxe concrète en essayant de respecter au mieux les notations existantes dans le domaine visé ;
2. Déduire la syntaxe abstraite de la syntaxe concrète et l'adapter au besoin.

Cette approche notationnelle a pour objectif de produire un langage le plus proche possible des attentes des experts en termes de notation. Elle est particulièrement applicable lorsqu'une notation préexiste dans le domaine et qu'il est important de reprendre le plus possible celle-ci afin de maximiser l'adoption du langage par les experts.

La comparaison de ces deux approches méthodologiques a fait l'objet d'une étude empirique avec des étudiants de Master 2 (Wouters et al., 2013). Les résultats obtenus par comparaison sur un cas commun montrent que l'approche notationnelle procure une meilleure proximité avec les attentes des experts du domaine. De même, les syntaxes abstraites obtenues par l'approche notationnelle sont de meilleure qualité. En outre, cette étude montre qu'il existe une forte corrélation entre la proximité de la syntaxe concrète et la qualité de la syntaxe abstraite avec l'approche notationnelle. Ceci s'explique par le fait que la syntaxe abstraite est déduite de la syntaxe concrète. Cette corrélation est une propriété importante de l'approche notationnelle, car elle montre que la construction de syntaxes concrètes de meilleure qualité conduit à des syntaxes abstraites également de meilleure qualité.

L'approche notationnelle démontre l'importance de la prise en compte de la syntaxe concrète dans la création d'éditeurs de modèles. Cette prise de conscience est similaire à celle, il y a de nombreuses années, concernant l'importance de l'interaction Homme-Machine lors de la conception d'applications informatiques. Profiter des bonnes pratiques en ingénierie logicielle pour la création d'éditeurs de modèles a été notre objectif lors de la création du «meta-outil ModX. Nous sommes donc partis d'une réflexion très générale sur la conception d'application, pour

déterminer quelles caractéristiques devait présenter notre outil afin de supporter toute démarche efficace de création d'éditeurs de modèles.

Concevoir un éditeur de modèles est aussi concevoir une application informatique avec une partie fonctionnelle (la syntaxe abstraite), une partie non-fonctionnelle (essentiellement la persistance) et une partie interactive (la syntaxe concrète et l'outillage associé). Ainsi avec l'outil ModX, nous avons voulu supporter, d'un point de vue logiciel, toute démarche liée à la conception d'éditeurs de modèles où le prototypage serait une nécessité. Pour cela, le cahier des charges de ModX était de permettre à un concepteur (de métamodèles) de pouvoir changer à tout moment d'aspect, c'est-à-dire définition de la syntaxe abstraite, définition de la syntaxe concrète et de l'outillage, et édition de modèles. Tous s'exécutent en parallèle et ModX assure continuellement la cohérence entre chacun de ces aspects.

Le premier avantage de notre solution est de permettre de tester rapidement les capacités d'expression d'un métamodèle nouvellement créé car un premier éditeur est non seulement fourni instantanément mais est facilement personnalisable. Enfin, grâce aux propriétés réflexives de ModX, chacun des 3 aspects pilotant un éditeur de modèles peut être modifié à tout moment, et les répercutions seront automatiques sur les modèles en cours d'édition. C'est ce que montre la figure 3 : si le ou les modèles créés ne conviennent pas, la possibilité de modifier un ou des aspects du langage de modélisation et de voir les répercussions dynamiquement permet de déterminer plus vite les évolutions nécessaires à apporter.

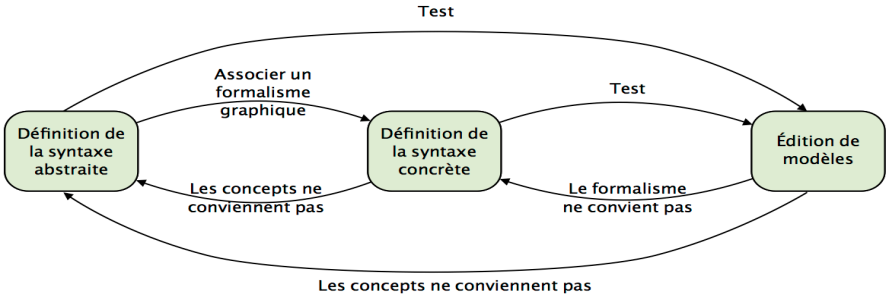

Figure 3. Cycle de vie classique de ModX

Suivant la même approche notationnelle, (Mandran 2013) définit un autre processus de création de langage où l'évaluation est centrale (Fig. 4). La définition de ce processus est le résultat d'une demi-douzaine d'expérimentations faites au Laboratoire d'Informatique de Grenoble lors de la création de langages. Les deux principes qui y sont défendus sont que d'une part l'évaluation d'un langage doit être réalisée tout au long du processus de développement d'un langage (i.e. lors de ses phases d'analyse, de conception et d'opérationnalisation) et que d'autre part l'évaluation doit être réalisée par des expérimentations centrées utilisateur. Il s'agit d'impliquer les futurs utilisateurs d'un langage dès l'étape d'analyse du futur langage. Des expérimentations impliquant les futurs utilisateurs du langage doivent être menées tout au long du processus de développement, certes pour valider le

langage, mais également pour l'explorer et le co-construire (cycles exploration/co-construction/validation sur le bas de la figure).

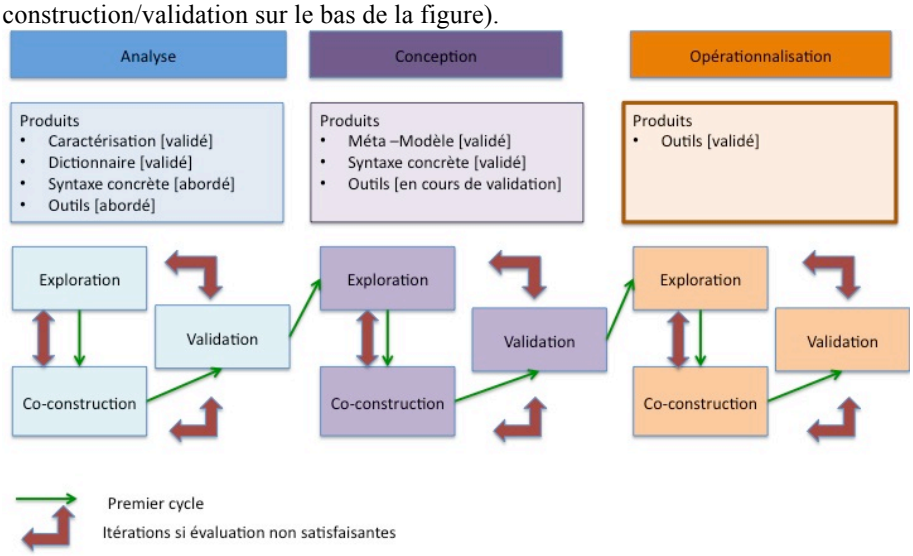

Figure 4. Cycle d'évaluation intégré au cycle de développement d'un langage

Les approches notationnelle et itérative remettent en question la primauté et la prédominance de la syntaxe abstraite lors de la conception d'un éditeur de modèles. Syntaxes abstraite et concrète sont deux facettes d'un même système, et la partie abstraite n'a pas à être favorisée. L'approche notationnelle va même jusqu'à privilégier la syntaxe concrète. Si on considère que l'aspect interactif est une préoccupation majeure de cette syntaxe, privilégier celle-ci rappelle les démarches de conception centrées sur l'humain où l'utilisateur/l'utilisation est centrale. La conférence Models a d'ailleurs récompensé les travaux de (de Lara 2012) qui préconise une approche ascendante où la définition d'un métamodèle est principalement faite au travers d'exemples d'utilisation.

4. Conclusion et perspectives

Cet article nous a permis de montrer l'intérêt d'une approche centrée sur l'humain pour la définition de langages de modélisation dédiés. Nos travaux sont le fruit d'une analyse conjointe a posteriori à partir des données d'usage que chacun des auteurs a pu recueillir à l'occasion de projets de longue durée mettant des utilisateurs en situation de modéliser à l'aide d'un langage dédié. La principale leçon qui ressort de cette mise en commun des expériences porte sur la nécessaire intégration des utilisateurs d'un langage durant sa conception que ce soit pour la définition et l'évaluation des éléments du langage. En particulier, l'outillage doit permettre de définir de manière aisée la syntaxe abstraite, en prenant en compte les difficultés liées à la manipulation du MOF et d'OCL ; il doit aider à mesurer la qualité des notations, tout en permettant de nécessaires ajustements pour mieux prendre en compte les utilisateurs ; et il doit apporter de la flexibilité dans la création

du langage pour supporter une approche centrée sur les concepteurs et les utilisateurs des langages.

Les perspectives de ce travail sont donc de proposer une ingénierie des langages centrée sur l'humain. On peut ainsi envisager des cycles de conception flexibles de langage où les futurs concepteurs, utilisateurs du langage, seraient parties prenantes dans la définition du langage et dans son évaluation ; des outils de métamodélisation qui ne restreindraient pas les pratiques des concepteurs de langages et leur permettraient de facilement le faire évaluer en fonction des besoins des concepteurs de modèles ; des outils de définition de métriques flexibles qui permettraient à chacun d'adapter ou de définir ses propres métriques sur les langages etc. Ainsi nous devons adapter les solutions proposées en ingénierie de la conception de système au domaine de la conception de langage de modélisation graphique. Les travaux que nous avons présentés dans cet article sont des premiers pas vers cette ingénierie, qui prend le parti d'une approche centrée sur l'humain, qui, nous l'espérons, contribuera à l'obtention de modèles et de langages de meilleure qualité car tout autant expressifs et mieux acceptés.

Remerciements

Les auteurs remercient l'association Inforsid pour son soutien financier pour l'organisation de réunions de travail. S. Dupuy-Chessa, X. Le Pallec et T. Nodenot sont aussi reconnaissants envers l'ANR pour son soutien sur ce sujet dans le cadre du projet MOANO.

Bibliographie

J. Aranda, N. Ernst, J. Horkoff, et Steve Easterbrook (2007) A framework for empirical evaluation of model comprehensibility. Dans Proceedings of the International Workshop on Modeling in Software Engineering, MISE '07, pages 7–, Washington, DC, USA, IEEE Computer Society.

J. Bertin (1967). Sémiologie graphique: les diagrammes, les réseaux, les cartes. The Hague, Paris: Mouton and Gauthiers-Villars.

A. Blackwell, and T. Green. (2003) Notational systems - the Cognitive Dimensions of Notations framework. Dans J.M. Carroll (Ed.) HCI Models, Theories and Frameworks: Toward a multidisciplinary science. San Francisco: Morgan Kaufmann, pages 103-134

J. Cadavid, B. Combemale, and B. Baudry (2012) Ten years of Meta-Object Facility: an Analysis of Metamodeling Practices, INRIA, Rapport de recherche RR-7882.

S. K. Card, A. Newell, and T. P. Moran (2000) The Psychology of Human-Computer Interaction. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

J. de Lara, J. Sánchez Cuadrado and E. Guerra. (2012) Bottom-up meta-modelling: An interactive approach. Lecture Notes in Computer Science 7590, Springer. pp.: 3-19. Presented at MODELS'12: ACM/IEEE 15th International Conference on Model Driven Engineering Languages and Systems. Springer best paper award.

A. Evans, M. Fernandez, and P. Mohagheghi (2009) Experiences of Developing a Network Modeling Tool Using the Eclipse Environment. In Model Driven Architecture -

Foundations and Applications, volume 5562 of Lecture Notes in Computer Science, pages 301-312. Springer Berlin / Heidelberg.

F. Fondement and T. Baar. (2005) Making metamodels aware of concrete syntax. Dans Model Driven Architecture - Foundations and Ap- plications, First European Conference (ECMDA-FA 2005), pages 190–204.

T. Green and M. Petre. (1996). Usability analysis of visual programming environments: a cognitive dimensions framework. Journal of Visual Languages and Visual Computing, 7, pages 131-174.

T. Green, A. Blandford, L. Church, C.R. Roast, et S. Clarke (2006). Cognitive dimensions: Achievements, new directions, and open questions. Journal of Visual Languages & Computing , pages 328-365.

J.C. Grundy, J . G. Hosking, J. Huh, K. Na-Liu Li. Marama: an Eclipse Metatoolset for generating multi-view environments. In proceedings of the 30th International Conference on Software Engineering. ACM, 2008.

K. Henricksen, J. Indulska (2004) Modelling and Using Imperfect Context Information, Proc the CoMeRea workshop at Percom'2004, Orlando, USA.

A. Kleppe (2007) A language description is more than a metamodel. Dans Fourth International Workshop on Software Language Engineering, Nashville, USA,.

J. Krogstie (2003) Evaluating UML using a generic quality framework. Dans UML and the unified process, pages 1–22. IGI Publishing, Hershey, PA, USA.

P. Laforcade, Thierry Nodenot et Christian Sallaberry (2005). "Résultats et perspectives d'un travail exploratoire mené en modélisation et méta-modélisation UML pour la conception de situations d'apprentissage." Numéro spécial "Conceptions et usages des plates-formes de formation" de la revue STICEF (Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation), volume 12

X. Le Pallec, S. Dupuy-Chessa (2011) Intégration de métriques de qualité des modèles et des métamodèles dans l'outil ModX, Inforsid 2011, papier court

X. Le Pallec, S. Dupuy-Chessa (2012) Intégration de métriques de qualité des diagrammes et des langages dans l'outil ModX, In Conférence en IngénieriE du Logiciel (CIEL), Renne.

N. Mandran, S. Dupuy-Chessa, A. Front, D. Rieu, Démarche centrée utilisateur pour une ingénierie de langages de modélisation de qualité , revue des Sciences et Technologies de l'Information, série Ingénierie des Systèmes d'Information, numéro spécial Evaluation des Systèmes d'Information, volume 18, numéro 3, p. 65 -94, août 2013

S. L. Star. (1989) The structure of ill-structured solutions: boundary objects and heterogeneous distributed problem solving. Dans Distributed Artificial Intelligence (Vol. 2), Morgan Kaufmann Publishers Inc. San Francisco, CA, US, pages 37 – 54

S. L. Star (2010) Ceci n'est pas un objet-frontière! Réflexions sur l'origine du concept. Dans Revue d'anthropologie des connaissances, Vol 4, n° 1, pages 18-35

O. Lindland, G. Sindre and A. Solvberg (1994). Understanding Quality in Conceptual Modeling. IEEE Softw. 11, 2, March 1994, 42-49.

D. L. Moody (2009) The "physics" of notations : Toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Software Eng., 35(6):756–779.

T. Nhan Luong, S. Laborie, T. Nodenot (2011). A Framework with Tools for Designing Web-based Geographic Applications. Dans the Eleventh ACM Symposium on Document Engineering (DocEng 2011). Googleplex, Mountain View, USA Pages 33-42

The Nhan Luong, Patrick Etcheverry, Christophe Marquesuzaà, Thierry Nodenot (2012). A visual programming language for designing interactions embedded in web-based geographic applications Dans the 17th ACM International Conference on Intelligent User Interfaces (IUI 2012). Lisbon, Portugal, Pages 207-216

T. Nodenot, P. Laforcade and X. Le Pallec (2007). Visual Design of coherent Technology-Enhanced Learning Systems: a few lessons learnt from CPM language. Handbook of Visual Languages in Instructional Design; Theories and Practices. L. Botturi and T. Stubbs, Hershey, PA: IDEA Group: 254-280.

T. Nodenot (2008). Scénarisation pédagogique et modèles conceptuels d'un EIAH : Que peuvent apporter les langages visuels ?, Revue Internationale des Technologies en Pédagogie Universitaire (RITPU) / International Journal of Technologies in Higher Education (IJTHE). Special Issue "Scénariser l'apprentissage, une activité de modélisation" 7(4): 85-103.

P. Mohagheghi et J. Aagedal (2007) Evaluating quality in model-driven engineering. Dans Proceedings of the International Workshop on Modeling in Software Engineering, MISE '07, pages 6–, Washington, DC, USA, IEEE Computer Society.

OCL2, OMG object constraint language, v2.2 (2010)

S. Patig (2008) A practical guide to testing the understandability of notations. Dans Proceedings of the fifth Asia-Pacific conference on Conceptual Modelling - Volume 79, APCCM '08, pages 49–58, Darlinghurst, Australia, Australian Computer Society, Inc.

S. Pinker (1990) A Theory of Graph Comprehension,"Artificial Intelligence and the Future of Testing, R. Freedle, ed., Lawrence Erlbaum and Assoc., pp. 73-126.

I. Ráth, A. Ökrös, D. Varró (2010) Synchronization of Abstract and Concrete Syntax in Domain-Specific Modeling Languages. Software and System Modeling 9:453-471

M. Rossi et S. Brinkkemper (1996) Complexity metrics for systems development methods and techniques. Information Systems, 21(2) :p. 209–227.

B. Selic (2008) UML 2 specification issue 6462, http://www.omg.org/issues/issue6462.txt, 2003, updates dating until 2008.

K. Siau et Y. Tian (2001) The complexity of unified modeling language: A goms analysis. Dans Proceedings of the International Conference on Information Systems (ICIS 2001), pages 443–448.

P. Trompette et D. Vinck. (2009) Retour sur la notion d'objet-frontière. Dans Revue d'anthropologie des connaissances, Vol. 3, n° 1, pp. 5-27

L.Wouters, M.-P. Gervais (2013) Notation-Driven vs Metamodel-Driven Development of Domain-Specific Modeling Languages: an Empirical Study. Symposium on Applied Computing, SAC 2013. ACM.