

MoDELS 2006 : Synthèse

Le Goaer Olivier

olivier.le-goer@univ-nantes.fr

Equipe MODAL – Université de Nantes

1 Introduction

En cette année 2006, la conférence MoDELS (anciennement conférence UML) s'est tenue à Gênes, en Italie. Elle a été l'occasion de rassembler une nouvelle fois des universitaires et des professionnels autour de l'ingénierie des modèles (IDM). Nous sommes trois étudiants à avoir bénéficié d'une bourse pour pouvoir assister à ces 3 journées de conférences. Le présent document constitue une synthèse des interventions et démonstrations auxquelles j'ai assisté.

2 Keynote – Irun Cohen

L'ouverture de MoDELS'06 s'est faite par l'intervention du Professeur Irun Cohen, éminent immunologiste au *Weizmann Institute of Science*, en Israël. Il débute sur une analogie entre le système immunitaire des êtres vivants et la notion de modèles, afin d'ouvrir singulièrement cette conférence sur l'ingénierie des modèles. En effet, ses contributions à l'immunologie incluent la théorie de *l'homunculus immunologique*, une supposée image de soi utilisée par le système immunitaire pour gouverner ses interventions. Le système immunitaire aurait donc un modèle de référence ? Théorie séduisante lorsque l'on sait que ce modèle fixe explicitement des limites, permettant ainsi au corps de détecter les éléments étrangers, puis de tenter de les éliminer pour rétablir son intégrité. Il n'en faut pas plus à l'assemblée présente pour s'accorder sur le fait qu'un système, quelqu'il soit (immunitaire, logiciel, ...), doit idéalement posséder une image de soi – son modèle, pour pouvoir (a) se protéger et (b) se maintenir. Par ailleurs, une loi ressort de la présentation du Pr. Cohen : plus un système est complexe, plus il est instable dans le temps. A titre d'exemple, il compare la stabilité d'une bactérie à travers les âges au terrible déclin des dinosaures. En conséquence, les systèmes logiciels de plus en plus complexes ont besoin de moyens pour se protéger et de se maintenir dans le temps et la conscience réflexive par les modèles est une solution possible. Mais tout n'est pas sans faille. Ainsi, le système immunitaire commet parfois des erreurs. Cela signifie-t-il que son modèle contient des erreurs ? Si c'est le cas, il serait nécessaire de disposer d'un méta-modèle immunitaire garantissant que le modèle est sain. Mais comment faire si ce méta-modèle immunitaire contient à son tour des erreurs ? ...Selon Cohen, c'est une problématique immunologique du XXI^e siècle. Pour terminer, notons qu'en réunissant et en accordant la médecine et l'informatique sur une même problématique, la modélisation démontre une nouvelle fois son incroyable pouvoir d'unification.

3 L'IDM dans le développement logiciel

Adopting Model Driven Software Development in Industry - A Case Study at Two Companies (Staron)

Cette session cherche à étudier le fossé qui sépare les académiques et les industriels sur les mises en oeuvre de l'ingénierie des modèles. En particulier elle s'interroge sur l'adéquation entre les développements dirigés par les modèles (DDM) et les réalités d'entreprises. En guise d'illustration, une étude de l'intégration du DDM des logiciels chez dans les entreprises *Ericsson* et *ABB* est présentée. Il en ressort essentiellement que les entreprises utilisent déjà UML, mais qu'elles rechignent à passer entièrement au DDM car (a) les bénéfices ne sont pas assurés et (b) qu'il manque les outils autour des DSLs, idée qui néanmoins les intéresse (bien moins onéreux que le développement de profils). En effet, si il y a bien un point sur lequel s'accorde les industriels et les académiques, c'est l'émergence des Langages spécifiques au domaine pour traiter chaque préoccupation. Mais l'adoption du tout DDM est jugé trop lourd et trop coûteux par les entreprises. La conclusion est donc sans appel, le DDM séduit mais son manque de maturité retarde son adoption par les entreprises.

Use Case Driven Iterative Development : Hurdles and Solutions (Ceria, Cukier)

Dans les développements logiciels itératifs, les diagrammes de cas d'utilisation (ou *Use Cases*) sont revisités à chaque itération et il est intéressant de capturer et de gérer leurs variations (appelés *Deltas*). Les auteurs proposent donc un format de spécification (une table) des Deltas de UC. Dès lors que l'on peut faire varier les Deltas eux-mêmes, nous sommes en présence de Deltas d'ordre supérieur – ou “Deltas de Deltas”. Les auteurs expliquent comment entreposer des UCs et des Deltas dans un *Use Case Repository*, ainsi que les différentes stratégies de fusion entre UCs et Deltas. A titre d'exemple, ils préconisent une fusion périodiquement, après un nombre fixé d'itérations. En somme, cette approche différencie les UCs “projet” des UCs “itérés” et assure à la fois traçabilité et réutilisation.

Model Driven Development with SDL - Process, Tools and Experiences (Kuhn, Gotzhein, Webel)

Cette session présente SDL-MDD, un processus de développement guidé par les modèles basé sur le langage de conception SDL (*Specific and Description Language*). SDL-MDD est processus de développement itératif destiné aux systèmes distribués et ubiquitaires, fournissant une méthodologie spécifique pour définir des PIMs (indépendance à la plate-forme) et des PSMs (détails spécifiques à la plate-forme). La suite d'outils SDL-MDD a été étendue dans deux directions :

1. *ns+SDL*, un outil pour la simulation de performance des modèles SDL
2. *SEnF* (SDL Environment Framework), une librairie de routines d'interfaçage pour éviter le codage manuel durant l'implantation de systèmes SDL ouverts.

Les auteurs ont illustré leur approche avec un extrait de spécification de PIM et de PSM pour un logiciel de *Training* pour vélo (*Assisted Bicycle Trainer*) embarqué dans un PDA.

4 UML pour l'interaction et la coordination

Compositional MDA (van Gool, Punter, Hamilton, van Engelen)

Les exigences importantes des système lithographiques (*waferscanners*) sont la haute performance et une synchronisation précise. Les auteurs présente un langage qui permet de modéliser un aspect important des waferscanners de la société ASML, appelé *coordination*. Le langage est un sous-ensemble compositionnel de diagrammes d'activités (AD) d'UML 2.0. Cela signifie que le langage est construit à partir de quelques simples patterns d'éléments d'ADs, et où les patterns peuvent avoir des sous-parties qui sont elles-mêmes construites avec ces patterns. Les auteurs définissent ensuite une transformation de modèles exprimés dans ce langage (les PIMs) en modèles propriétaires d'ASML qui implémentent une coordination (les PSMs).

CUP 2.0 : High-Level Modeling of Context-Sensitive Interactive Applications (Van den Bergh, Coninx)

CUP 2.0 est un profil UML pour une modélisation haut-niveau d'applications interactives sensibles au contexte (*context-sensitive*). Ce profil a été créé pour faciliter la communication entre des spécialistes de l'IHM et des ingénieurs logiciels, au sujet de la conception de ces applications. Le profil CUP 2.0 permet ainsi de spécifier cinq modèles : *System interaction*, *Context*, *Application*, *Abstract User Interface*, *User Interface Deployment*. Le profil autorise une description détaillée à la fois de la structure et du comportement de l'interface utilisateur d'une application sensible au contexte, supporté par tous les outils de modélisation UML qui acceptent l'extension de leur méta-modèle via les profils.

5 Sécurité

A MOF/QVT-based Domain Architecture for Model Driven Security (Hafner, Alam, Breu)

Les intervenants de cette session détaillent leurs méta-modèles de sécurité (basés sur le MOF) définissant un langage spécifique au domaine (DSL) de la conception de *workflows* inter-organisationnels. Le langage supporte diverses catégories de *pattern* de sécurité qui sont appliqués aux modèles de *workflows*, pour y intégrer l'aspect sécurité. Ensuite, les intervenants spécifient des transformations de modèle à modèle, basées sur le standard du MDA MOF-QVT. Ces transformations traduisent les PIMs en PSMs, ces derniers étant destinés à une architecture de référence. Notons pour terminer que tout cela est regroupé au sein d'un framework nommé SECTET.

A Model Transformation Semantics and Analysis Methodology for SecureUML (Brucker, Doser, Wolff)

SecureUML est un langage de modélisation de la sécurité pour formaliser les exigences de contrôles d'accès, de façon déclarative. SecureUML prend la forme d'un profil UML qui vient étendre les notations et la sémantique standard d'UML. Par ce biais, l'idée est d'inclure le contrôle d'accès dans le modèles de données supporté par un diagramme de classes UML. A partir de ce mixage, un modèle du système sécurisé est généré (UML+OCL), qui peut (a) être analysé sous l'angle de la sécurité, (b) puis transformé en code.

MDA-based Re-Engineering with Object-Z (Süß, McComb, Kim, Wildman, Watson)

SIFA (*Security Information Flow Analyser*) est un outil existant d'analyse – avec interface graphique – utilisé pour prédire l'impact des fautes de sécurité dans les circuits électroniques. SIFA évolue régulièrement pour satisfaire de nouvelles exigences, mais n'a pas de méta-modèle. Par conséquent, tester la validité de SIFA est difficile. Pour y remédier, les auteurs développent un modèle EMF du domaine à partir de l'implantation de SIFA (rétro-ingénierie). Par une transformation de modèle, ils passent du modèle EMF à un modèle Object-Z. Pour terminer, ils complètent le modèle Object-Z en spécifiant le comportement du système. Au final, ils disposent à la fois d'un méta-modèle précis du domaine (sécurité dans les circuits électroniques) et d'une base formelle solide pour tester les implémentations de SIFA.

6 Transformation de modèles : outils et implémentations

Experimenting with model transformation through a plugin-based language (Sánchez Cuadrado, García Molina)

RubyTL est un langage hybride (c-a-d impératif/déclaratif) de transformation définit en tant que DSL interne à Ruby (le langage de programmation), et est conçu comme un langage extensible : un mécanisme de plugin permet d'ajouter de nouvelles fonctionnalités aux fonctionnalités du noyau. Cette session met principalement l'accent sur le mécanisme d'extension. Dans l'approche proposée, les nouvelles fonctionnalités sont ajoutées en créant des plugins (codés en Ruby) qui implantent des *points d'extensions* prédéfinis. Ces plugins permettent principalement d'adapter le langage de transformation à une famille de problèmes, sans pour autant avoir à modifier le noyau de fonctionnalités de transformations.

Incremental Model Transformation for the Evolution of Model-Driven Systems (Hearnden, Lawley, Raymond)

Les auteurs présentent une stratégie pour la maintenance incrémentale des exécutions de transformations déclaratives à base règle. Les mises à jour incrémentales d'un modèle de transformation peut être mise en oeuvre efficacement par le biais de transformation *live*. Les transformations *live* impliquent de considérer les transformations en tant que calculs continus. Les dépendances entre l'exécution de la transformation et ses entrées peuvent être capturées par un arbre de résolution étiqueté. Ces dépendances peuvent alors être utilisées pour propager les changements sources aux changements cibles. Cette approche a été largement plébiscité par l'assistance. Un membre fait remarquer que les industriels misent eux aussi sur ce type de solution.

SiTra : Simple Transformations in Java (Akehurst, Bordbar, Evans, Howells, McDonald-Maier)

Cette présentation part du constat que de nombreux frameworks de transformations de modèles sont régulièrement proposés, chacun possédant ses propres spécificités et obligeant à chaque fois l'utilisateur à apprendre un nouveau langage (même si ils sont basés sur le standard QVT). Dans ces conditions, la moindre transformation requière un temps d'apprentissage important. A l'opposé, les intervenants présente SiTra, une

bibliothèque minimale en Java 5.0 supportant l'implantation de nombreuses transformations. Le mot d'ordre de leur approche est "simplicité". Les transformations écrites en Java bénéficient de la familiarité des utilisateurs avec ce langage. Cette approche est audacieuse dans sa simplicité en prenant les autres propositions à contre-pied, mais aussi moins puissante (pas de méta-modèle par exemple), "mais c'est voulu", dicit l'intervenant. Enfin, un membre de l'assistance fait remarquer que SiTra est donc un DSL Java pour la transformation.

7 Spécification de transformations

Model Transformation By Example (Varró)

Dans cette approche, l'interconnection d'un modèle source et d'un modèle cible génère (semi-)automatiquement un prototype de transformation. Cette transformation peut ensuite être raffinée à mesure que d'autres paires de modèles sources-cibles sont fournies. Ainsi, plus l'on fournit d'exemples, plus la transformation générée est précise. Le principal avantage de cette approche est que les concepteurs de transformation n'ont pas besoin d'apprendre un nouveau langage de transformation de modèle, car au lieu de cela ils mettent simplement en relation leurs modèles cibles et leurs modèles sources. Maintenant, il reste à encore à démontrer l'applicabilité de cette approche à de large échelles et avec des problématiques de transformations industrielles.

Graphical Definition of Rule-based Transformation in the Eclipse Modeling Framework (Biermann, Ehrig, Kuhns, Taentzer, Weiss)

Les auteurs présente une approche pour la définition graphique de transformations de modèles "sur place" (*in place transformations*), c'est à dire des transformations endogènes. Ils proposent une notation visuelle des règles de transformations pour la plate-forme Eclipse. Les règles de transformations contiennent une partie gauche et une partie droite, qui sont des structures d'objets. De plus, des patterns objets "négatifs" peuvent être définis pour restreindre l'application des règles. Il est possible de migrer les règles vers AGG, un outil de transformation de graphes, tout comme il est possible de générer le code Java correspondant aux règles. Enfin, notons que leur approche de transformation ne respecte pas le standard QVT.

Model Transformations ? Transformation Models ! (Bezivin, Buettner, Gogolla, Jouault, Kurtev, Lindow)

La plupart des travaux sur la transformation de modèles sont exécutables par nature, ce qui est nécessaire d'un point de vue implantation. Mais d'un point de vue conceptuel, les transformations peuvent également être vues comme des modèles descriptif listant uniquement les propriétés que les transformations doivent combler, en omettant les détails d'exécution. Cette session discute de l'abstraction des transformations de modèles en tant que modèles de transformations. Un modèle de transformation est un diagramme de classe, qui contient la syntaxe et la sémantique du domaine décrit. Cette session a permis de se convaincre que les transformations sont également des modèles à part entière et qu'à ce titre, elles peuvent elles-mêmes être transformées : ce sont les transformations d'ordre supérieur.

8 Keynote – Hassan Gomaa

Cette présentation a débuté sur une large perspective sur la modélisation logicielle, décrivant comment les méthodes de modélisation et de conception des systèmes temps réels ont progressé, des tout premiers systèmes centralisés aux systèmes temps réels modernes et aux lignes de produits. Ces derniers sont d'ailleurs tous les deux distribués, orientés objet, et conçus en UML. La présentation a ensuite décrit les éléments clés des méthodes de conception pour les lignes de produits à base de composants logiciels, qui promettent réutilisation, gestion de la variabilité, et évolution. Les outils de l'ingénierie des lignes de produits sont également abordés. Enfin, quelques défis majeurs sont mis en avant, en particulier la conception d'architectures logicielles évolutives et configurables dynamiquement.

9 Ponts entre modèles

Building Abstractions in Class Models : Model Transformations coupled with Formal Concept Analysis (Arévalo, Falleri, Huchard, Nebut)

Les auteurs proposent une approche pour détecter et construire automatiquement des abstractions dans un diagramme de classe UML. Leur méthode est fondée sur l'analyse relationnelle de concepts, une extension de l'analyse formelle de concepts. Cela fonctionne par des transformations successives de modèles, basées sur différents méta-modèles et implémentées en langage Kermeta. L'approche introduit des abstractions pour les classes (avec des liens de spécialisation), les attributs, les méthodes, etc., dans un diagramme de classe.

Lifting Metamodels to Ontologies - A Step to the Semantic Integration of Modeling Languages (Kappel, Kapsammer, Wimmer, Kramler, Reiter, Retschitzegger)

Cette session présente *the lifting procedure*, qui permet de construire des ontologies à partir de méta-modèles représentant des langages de modélisation. L'application de patterns de retro-ingénierie sur ces ontologies permet d'explicitier des concepts initialement enfouis et ainsi d'améliorer le support automatisé des tâches d'intégration sémantique. Cette intégration est en effet devenue une tâche essentielle au regard de l'utilisation des différents langages de modélisation dans le développement logiciel.

Incremental Model Synchronization with Triple Graph Grammars (Giese, Wagner)

Pour supporter la synchronisation de modèles, les intervenants présentent une approche de transformation incrémentale de modèles. Ils emploient une technique de transformation bidirectionnelle, formelle et visuelle, basée sur les grammaires triple graphes (source-correspondance-cible). Avec ce formalisme de spécification déclarative, ils se focalisent sur l'exécution des règles de transformation. Leur solution a été intégrée dans la suite logicielle *Fujaba*. L'outil permet de spécifier visuellement les règles de grammaire triple graphes, l'obtention automatique du graphe des règles de ré-écritures, et l'exécution incrémentale de ces règles. Notons pour finir que les transformations basées sur les grammaires triple graphes ne respectent pas le standard QVT.

10 Conclusion

La conférence MoDELS 2006 a été pour moi l'occasion de confronter ma vision des architectures à base de composants (mon domaine de recherche) avec celle des spécialistes de l'ingénierie des modèles. Ce sont deux espaces d'ingénierie qui ont des problématiques communes tout en ayant leur propres spécificités. Ce qui est certain, c'est que ces deux espaces peuvent s'enrichir mutuellement de leurs travaux.

Les organisateurs ont statué que la prochaine conférence MoDELS en 2007 se tiendra à Nashville, aux Etats-Unis.