

Un compte rendu de la conférence Models 2008 (Toulouse)

Mathieu Acher¹ et Vincent Aranega²

¹ Université de Nice Sophia-Antipolis - Equipe Rainbow, Laboratoire I3S
acher@i3s.unice.fr <http://www.i3s.unice.fr/~acher/>

² Laboratoire d'Informatique Fondamentale de Lille (LIFL)
vincent.aranega@lifl.fr

Résumé Dans le cadre des activités de l'Action IDM, nous avons pu assister à la conférence internationale Models 2008. A cette occasion, de nombreux travaux de recherche ont été présentés et ce document a pour objectif de mettre en forme nos notes personnelles. Ce compte-rendu décrit le déroulement du workshop Models@run.time et résume le contenu scientifique de plusieurs présentations.

1 Introduction

La conférence Models 2008 a eu lieu du 1^{er} au 3 octobre 2008 en France à Toulouse. En complément des présentations classiques et des sessions, plusieurs interventions orales sous forme de *keynotes* et quelques tables rondes (*panels*) ont été tenues. Associés à la rencontre, plusieurs groupes de travail (*workshop*) et tutoriaux ont été organisés les jours précédents.

Ce document s'adresse aussi bien aux participants de la conférence Models qu'aux personnes intéressées par la conférence mais n'ayant pu y assister. L'organisation du document suit l'ordre chronologique des événements de la conférence, en débutant par un compte rendu du workshop Models@run.time puis en décrivant les trois jours de la conférence à Toulouse.

2 Workshop : Models@run.time

Le workshop Models@run.time 2008³ s'est déroulé le 30 Septembre 2008, en tant qu'évènement satellite précédant la conférence Models proprement dite. La troisième édition du workshop Models@run.time a été organisée par Nelly Bencomo, Gordon Blair, Robert France, Freddy Munoz et Cedric Jeanneret. L'objectif du workshop est de discuter des modèles à l'exécution. Ce concept est *a priori* nécessaire pour de nombreuses classes d'applications qui évoluent dans un environnement fortement hétérogène, distribué et/ou sujet à de nombreux changements et qui, dans ce contexte, requièrent des mécanismes adaptés à leurs besoins pour pouvoir e.g. se reconfigurer, auto-réagir ou simplement

3. <http://www.comp.lancs.ac.uk/~bencomo/MRT/>

être surveillés à l'exécution. L'idée est alors de développer des mécanismes, des techniques, des formalismes et des outils pour répondre à ces besoins tout en gardant les bénéfices d'une approche dirigée par les modèles. Le concept de Models@run.time mérite cependant d'être discuté (e.g. à quoi ressemble un modèle à l'exécution?) et illustré tant il peut s'appliquer à des applications et à des objectifs divers. Ainsi, six papiers longs et six papiers courts ont été acceptés pour 20 soumissions.

Les présentations du matin n'ont pas dévié de l'objectif initial du workshop, en développant des approches assez différentes les unes des autres et dont les domaines d'application variaient fortement. Par exemple, deux présentations se sont intéressées à intégrer des mécanismes dans les langages orientés objet^{4 5}; une autre présentation décrivait une technique pour tracer le comportement des modèles à l'exécution⁶ tandis que deux présentations proposaient une démarche^{7 8} pour construire des applications capables de supporter des scénarios d'adaptation. Ces contributions, parmi d'autres, montrent la diversité des objectifs pouvant être atteints ainsi que des mécanismes de modèle pouvant être mis en place. Aussi, chaque présentation fut l'occasion, pour les participants, de défendre leurs visions d'un "Model@run.time", adapté à la problématique rencontrée.

L'après-midi a débuté par une démonstration (réalisée par Brice Morin) de l'outil Kermeta permettant un tissage dynamique d'aspects dans les modèles. Le workshop s'est poursuivi et a consisté d'une part à discuter de la maturité du concept de modèle à l'exécution (i.e. le concept est-il capable d'avoir un impact industriel et/ou sur la communauté modèle?) et d'autre part à dégager des consensus sur la notion de modèle à l'exécution⁹. Pour répondre à ces deux interrogations, les participants ont formé trois groupes et ont délibéré pendant une heure. Ce fut notamment l'occasion de revenir sur les présentations du matin en comparant les démarches et les expériences de chacun, ou simplement d'exposer les limites actuelles du domaine.

Le workshop s'est achevé en réunissant un responsable par groupe de travail, rejoignant les trois invités de la table ronde : Bran Selic, Jean-Marc Jézéquel, Øystein Haugen. Parmi les conclusions du workshop, notons les remarques suivantes :

- le rôle des modèles à l'exécution mérite encore d'être précisé et documenté tout comme la définition de ce qu'est un modèle à l'exécution ;

4. **Embedding State Machine Models in Object-Oriented Source Code**, Michael, Striwe, Moritz Balz and Michael Goedicke

5. **FAME—A Polyglot Library for Metamodeling at Runtime**, Adrian Kuhn and Toon Verwaest

6. **Model-Based Traces**, Shahar Maoz

7. **Runtime Models for Self-Adaptation in the Ambient Assisted Living Domain**, Daniel Schneider and Martin Becker

8. **Modeling and Validating Dynamic Adaptation**, Franck Fleurey, Vegard Dehlen, Nelly Bencomo, Brice Morin, Jean-Marc Jézéquel

9. à noter la difficulté de la traduction de "Model@run.time" : est-ce un modèle à l'exécution ? un modèle exécuté ? un modèle d'exécution ?

- les domaines d’application des `models@run.time` peuvent se classer en trois catégories : les systèmes adaptatifs, la simulation à l’exécution et la compréhension des programmes. Cette classification mérite également d’être affiné et ne se prétend pas exhaustive ;
- le besoin de vrais cas d’études et d’applications industrielles pour pouvoir démontrer les bénéfices des `models@run.time`. Certains participants ont pensé que les travaux relatifs aux `models@run.time` n’étaient pas encore assez matures, tandis que d’autres se sont montrés beaucoup plus nuancés et optimistes ;
- la non prise en compte dans les exposés du workshop des besoins extra-fonctionnels (qualité de service) ;
- la volonté de faire évoluer et réagir les modèles, d’avoir des modèles interprétés avec qui on peut “jouer” et ainsi simuler une abstraction décente de la réalité ;
- une question récurrente : en quoi les modèles@run.time sont-ils différents des modèles@design time ? (e.g. mêmes rôles ? mêmes technologies ?).

Un appel à contributions pour “IEEE Computer : Special issue on Models@run.time”¹⁰ a été lancé : ce sera certainement l’occasion de développer les points mentionnés précédemment et de présenter des avancées dans le domaine. Enfin, le *proceedings* de l’évènement est en ligne sur le site du workshop.

3 Models : conférence principale (Mercredi 1^{er} octobre 2008)

3.1 Keynote : Abstraction and modelling a complementary partnership, *Jeff Kramer*

Jeff Kramer a présenté pendant un peu plus d’une heure un aspect souvent négligé dans le processus de modélisation et pourtant fondamental : l’abstraction. Pour Kramer, l’abstraction est l’action de retirer ou de supprimer des détails non importants à la compréhension d’un problème (i.e une simplification de la réalité) de manière à pouvoir se focaliser sur les propriétés importantes et faciliter ainsi le raisonnement. L’identification des propriétés communes ou plus générales d’un système fait également partie de ce processus d’abstraction. A travers divers exemples (notamment artistiques), il a montré l’importance du concept. La manière de représenter la carte du métro à Londres en est un : est-il nécessaire d’avoir les noms des rues sur la carte ? à qui s’adresse cette carte ? quel est l’objectif de la carte ? La simplification de la carte des métros de Londres au cours de l’histoire montre ce besoin d’abstraction. Une idée proche est qu’un modèle est là pour répondre à des questions par rapport au système qu’il représente. Aussi, la modélisation et l’abstraction sont deux partenaires complémentaires : nous avons besoin d’abstraction pour concevoir des modèles, et la modélisation est une technique fondamentale pour l’amélioration et l’expression de notre pouvoir d’abstraction. Le manque d’abstraction est souvent le maux de nombreux

10. <http://www.comp.lancs.ac.uk/~bencomo/MRT/>

problèmes d'ingénierie (e.g la construction de logiciels). Selon l'auteur, la raison pour laquelle certains étudiants ou ingénieurs logiciels sont capables de produire des programmes propres, concis et élégants, alors que d'autres n'y parviennent pas, résulte directement des capacités d'abstraction. Aussi, Kramer s'interroge sur les manières de mesurer les aptitudes à s'abstraire des problèmes et soutient fortement que l'enseignement de l'abstraction est un enjeu majeur pour la formation des futurs étudiants. Il conclut en rappelant le lien fort entre modélisation et abstraction – lien qui pourrait d'ailleurs être utilisé pour parvenir à son objectif.

Nous invitons les lecteurs intéressés à étudier le papier : Jeff Kramer, Is abstraction the key to computing? *Commun. ACM*, 50(4) :36–42, 2007.

3.2 Session 1 : Model Transformation Foundations

Algebraic Models for Bidirectional Model Synchronization, *Zinovy Diskin*

Cette première présentation ouvrait le thème des transformations de modèles. Il fut présenté un framework algébrique à partir duquel les sémantiques des différents types de synchronisation bi-directionnelle de modèle et des transformations peuvent être spécifiées et analysées.

L'auteur commença par présenter de manière assez formelle les différents types de système de synchronisation qui sont au nombre de trois :

- le système diagonal,
- le système trigonal,
- le système “*lenses*”.

Associé à ses trois systèmes de synchronisation bi-directionnel de modèles, quatre groupes de lois algébriques représentant quatre concepts de synchronisation sont ensuite définis. Ces concepts sont :

- *correctness*,
- “*hippocraticness*”,
- *history-ignorance and undoability*,
- *invertibility*.

Parmi les trois systèmes proposés, c'est le système “*lenses*” qui fut développé plus particulièrement, les autres systèmes étant formellement décrits dans l'article.

En conclusion, il est possible via ce framework algébrique de définir formellement les sémantiques des différents système de synchronisation de modèle bi-directionnel ainsi que leurs transformations (que cela soit en *forward* ou en *backward*).

An invariant-based Method for the Analysis of Declarative Model to Model Transformation, *Jordi Cabot, Robert Clarisó, Esther Guerra et Juan de Lara*

Dès que l'on parle de transformation de modèle à modèle, on pense aux langages de transformation. Il existe deux approches clefs des langages de transformation : une approche opérationnelle et une approche déclarative. Dans l'ap-

proche déclarative, plusieurs types de *pattern* (graphiques ou textuels) sont utilisés pour représenter les règles de la transformation. Malgré le nombre de notations proposé pour spécifier les transformations modèle vers modèle, l’approche déclarative manque cruellement de méthode pour vérifier la correction et la bonne formation des règles. A travers l’utilisation des règles utilisée par les TGG¹¹ et des morphismes de graphes, les auteurs nous proposent une solution par extraction d’invariant OCL.

Tout d’abord, les expérimentations ont été effectuées sur des règles classiques utilisées par les TGG, puis sur leur formes déclaratives. Par l’analyse de ces règles, un invariant OCL est donc extrait pour permettre ensuite son analyse. Finalement, ils ont étendu leur réflexion aux règles déclaratives spécifiées en QVT¹².

Une fois l’invariant OCL extrait, les auteurs l’analysent pour permettre la validation de la transformation de modèle ainsi que sa vérification. La validation permet de répondre à la question “la transformation est-elle correctement spécifiée?”, et la vérification à la question “la transformation est-elle correcte?”. Pour effectuer ces deux analyses, l’invariant est vérifié en utilisant un *solver* : UML-toCSP qui va se charger d’effectuer les divers calculs automatiquement.

A travers cet article, les auteurs nous proposent une façon de prouver la correction et la bonne formation de règles déclaratives spécifiées sur des TGG ou via QVT par extraction d’un invariant OCL qui va servir à l’analyse des règles.

Precise Semantics of EMF Model Transformations by Graph Transformation, *Enrico Biermann, Claudia Ermel et Gabriele Taentzer*

Actuellement, EMF¹³ est très largement utilisé dès lors que l’on parle d’ingénierie dirigée par les modèles. Cependant, lors de transformations ou de manipulations de modèles, rien n’assure que le modèle produit soit consistant. L’idée générale serait donc de pouvoir vérifier que les règles de transformations proposées par l’utilisateur engendreraient un modèle consistant.

Pour résoudre ce problème, les auteurs proposent de passer par des graphes. Un modèle EMF peut donc être vu comme un graphe possédant une certaine sémantique. Par conséquent, un ensemble de règles de transformations EMF peut être vu comme un ensemble consistant de règles de transformation de graphe. C’est à travers l’exemple d’un *refactoring* de machines à état que l’auteur a ensuite décidé de nous présenter les différentes règles de consistance. Elles sont au nombre de quatre :

- la création,
- la suppression,
- la création d’une composition,
- la suppression d’une composition.

Le point le plus délicat est notamment la création de cycles qui peut intervenir lors de l’ajout ou la suppression de compositions.

Une fois les diverses règles traduites sous forme de règle de transformation de

11. Triple Graph Grammar

12. Query/View/Transformation

13. Eclipse Modeling Framework

graphes, il reste donc à vérifier, en utilisant la sémantique proposée, si ces diverses règles sont violées et si un “cycle de compositions” intervient ou non. Les auteurs proposent donc la validation des transformations EMF par l’utilisation des transformations de graphe et l’ajout d’une sémantique pour vérifier la consistance des règles écrites.

3.3 Session 2 : Requirement Modelling

A Formal Metamodel for Problem Frames, *Denis Hatebur, Maritta Heisel et Holger Schmidt*

Les *Problem Frames* sont une méthode et une notation servant à classifier des classes de problèmes rencontrés lors du développement de logiciels ; ils permettent aussi de structurer l’analyse des exigences d’un logiciel. Actuellement, il manque une notation et une sémantique rigoureuse pour les Problem Frames. L’approche défendu ici est l’élaboration un méta-modèle formel pour les Problem Frames. Les auteurs ont ainsi utilisé un modèle de classe UML combiné à des spécifications OCL. Le modèle de classe permet de représenter chaque élément syntaxique d’un Problem Frame, ainsi que les relations entre les différents concepts (Domain, Interface, Requirement, etc.). Les conditions d’intégrité, exprimés en OCL sous la forme d’invariant, permettent d’assurer une cohérence syntaxique et une validation sémantique pour l’instanciation de nouveaux Problem Frame en conformité avec le méta-modèle. Un outil développé sous la plate-forme Eclipse permet à l’utilisateur de générer graphiquement des nouveaux Problem Frame, ce qui a permis par exemple aux auteurs d’instancier certains Problem Frames décrits dans la littérature. L’outil a été généré grâce au méta-modèle et parvient à détecter des erreurs introduites délibérément par les auteurs dans des instances de Problem Frame.

Les bénéfices de l’approche sont nombreux (compréhension non ambiguë, clarification de la syntaxe, génération quasi automatique de l’outil, etc.) et posent les fondations pour une utilisation future de l’approche Problem Frame.

3.4 Session 3 : Domain-Specific Modeling

The Future of Train Signaling, *Andreas Svendsen, Oystein Haugen, Goran K. Olsen, Erik Carlson, Jan Endresen, Thomas Moen, Kjell-Joar Alme*

L’objectif d’un système de contrôle entrecroisé (interlocking system) pour les chemins de fer est d’empêcher les conflits et les dangers des mouvements de trains. Le système de contrôle de la signalisation contrôle les commutations et les signaux dans une gare en vue d’établir un parcours sûr et fiable pour les trains. Ces systèmes sont maintenant informatisés. Malheureusement, malgré leur haute importance et leur caractère critique, la conception de tels systèmes est potentiellement source de nombreuses erreurs et de nombreuses phases (génération de codes, intégration, tests, etc.) requièrent une intervention manuelle.

Pour faire face à ces limites, Andreas Svendsen présente un développement dirigé par les modèles, qui s’appuie sur un langage spécifique au domaine (DSL),

Train Control Language (TCL). Une partie du méta-modèle TCL est décrite : le méta-modèle est construit en utilisant EMF ; la représentation graphique des concepts de TCL est rendue possible en utilisant le framework GMF. La sémantique de TCL est définie par des transformations (utilisation de MOFScript) vers d'autres représentations (équations logiques, code source, génération des tables entrecroisés, etc.). Les représentations générées sont consistantes entre elles grâce à l'utilisation d'un modèle commun. Les contraintes sur les éditeurs (générés automatiquement) assurent la complétude du système généré. Un autre bénéfice important de l'approche est que de nombreuses phases du développement sont maintenant automatisées (notamment les tests). Les auteurs ont également présenté quelques leçons de leurs travaux (e.g. importance de transférer la connaissance du domaine et d'échanger avec les experts, non altération du cycle de développement en V, etc.). En conclusion, les auteurs soutiennent que l'utilisation de mécanismes de variabilité dans les modèles peut améliorer le processus de développement (e.g. création de nouvelles stations à partir de stations déjà existantes en faisant varier les valeurs et/ou la structure). Définir et résoudre la variabilité en utilisant un langage dédié à l'expression de la variabilité est un axe de recherche intéressant qui pourrait s'intégrer dans les travaux présentés, sans augmenter la complexité du DSL.

NAOMI - An Experimental Test Bed for Multi Modeling, *Trip Denton, Edward Jones, Srini Srinivasan, Kenneth Owens, Richard Buskens*

La multi-modélisation a pour objectif de modéliser les divers aspects (e.g. sous-systèmes) d'un système et d'intégrer différentes expertises (ou vues) dans sa conception. Ainsi, dans des projets complexes, plusieurs DSLs sont généralement nécessaires pour faire face aux diverses préoccupations. Cependant, l'utilisation de multiples modèles n'est pas sans poser de problèmes et sont autant de challenges pour la recherche. Il s'agit *i*) de capturer les interdépendances entre les modèles (e.g. identification des données échangées) ; *ii*) d'assurer une consistance entre les modèles (e.g. assurer que les changements sur un modèle soient propagées aux autres) ; *iii*) de fournir une sémantique suffisamment précise pour décrire les échanges entre les modèles. Les auteurs ont ensuite détaillé les différents besoins pour atteindre de tels objectifs (l'aspect sémantique sera étudiée dans un travail futur). La réponse apportée à la problématique de la multi-modélisation et aux besoins qui ont été identifiés est NAOMI, une plate-forme expérimentale. C'est une tentative pour examiner la faisabilité de l'intégration de multiples modèles. Il est notamment possible : de spécifier des interfaces pour les modèles (un ensemble d'attributs représentant les entrées et les sorties et décrit dans un langage spécifique), de modéliser les contraintes des modèles (en entrée et en sortie), d'établir des relations entre les modèles et d'assurer la propagation des modifications d'un modèle pour assurer la consistance entre les modèles, etc. Edward Jones a ainsi réalisé une démonstration des possibilités de la plate-forme, en utilisant l'application développée (un client riche Netbeans) dans le projet. Le travail présenté ouvre de nombreuses perspectives de recherche vis-à-vis du développement du concept de multi-modélisation.

Conférencier invité : Patrick Rauhut “3D Parametric models for aeroplanes - from idea to design”

3.5 Awards

Au cour de la conférence, quelques prix ont été remis dans différentes catégories :

ACM Most distinguished paper award :

- Empirical Analysis of the Relation between Level of Detail in UML Models and Defect Density, *Ariadi Nugroho, Bas Flaton, Michel Chaudron*
- Semantically Configurable Code Generation, *Adam Prout, Joanne M. Atlee, Nancy A. Day, Pourya Shaker*

Best Paper Award Offered by Springer :

- Empirical Analysis of the Relation between Level of Detail in UML Models and Defect Density, *Ariadi Nugroho, Bas Flaton, Michel Chaudron*

Ten Year Most Influential Paper Award :

- The UML as a Formal Modeling Notation, *Andy Evans, Robert B. France, Kevin Lano, Bernhard Rumpe*

Best Doctoral Paper Award :

- Georgia Kapitsaki et Elina Kalnina

3.6 Session 4 : Model Transformation : Techniques

Model Transformation as an Optimization Problem, *Marouane Kessentini, Houari Sahraoui et Mounir Boukadoum*

Actuellement, lors d’une transformation de modèle, il est souvent difficile d’écrire les règles à cause, notamment, du nombre d’approches de transformations possibles. La finalité recherchée par les auteurs est de pouvoir trouver et exprimer au mieux les règles de transformations en s’appuyant sur un petit ensemble d’exemples.

Pour effectuer un tel travail, les exemples utilisés par leur moteur MOTOE¹⁴ sont représentés sous la forme d’un triplé comprenant une description du modèle source, du modèle cible ainsi qu’un ensemble de blocs de mapping. Grâce à cette représentation, il leur est possible d’évaluer une “qualité” de transformation. La recherche de la meilleure transformation revient donc à exhiber celle qui produit la meilleure “qualité”. L’espace de recherche étant très important, il est nécessaire d’utiliser une heuristique pour réduire l’espace de recherche. Les auteurs se sont appuyés sur l’algorithme PSO¹⁵ pour évaluer au mieux les exemples et rechercher le plus rapidement et efficacement possible la meilleure transformation.

Il est proposé ici une nouvelle approche pour automatiser les transformations de modèles en utilisant l’algorithme PSO.

14. Model Transformation as Optimization by Example

15. Particle Swarm Optimization

Detecting Patters of Poor Design Solutions Using Constraint Propagation, *Ghizlane El-Boussaidi et Hafedh Mili*

Appliquer un design pattern (patron de conception) à un modèle en entrée implique plusieurs étapes. Il faut comprendre le pattern, identifier les situations dans lesquelles il est justifié (i.e quel est le problème de conception du modèle ?), et enfin appliquer la solution recommandée à de telles situations (i.e quelle est la solution de conception ?). L'implémentation des patrons nécessite également une mise en correspondance entre les éléments de conception du patron et chaque élément du modèle. L'objectif des auteurs est de parvenir à reconnaître automatiquement à quelle instance d'un problème de conception les modèles d'entrée correspondent. Typiquement, la mauvaise conception d'un modèle sera identifiée et corrigée par l'application d'un design pattern. Pour atteindre cet objectif, l'approche défendu consiste à représenter les designs patterns par un triplé (MP, MS, T) : MP est un modèle représentant le problème de conception, MS est un modèle de la solution proposée par le pattern, et T la transformation qui transforme un modèle en entrée, possiblement instance de MP, en une instance de la solution associée, MS. Ainsi, le modèle en entrée subit une phase de "marquage" : étant donné un modèle en entrée, il s'agit de reconnaître les instances des modèles de problème (MP). Chaque élément du modèle MP est apparié avec le modèle en entrée. Comme les modèles peuvent être vus comme des graphes, les auteurs soutiennent que ce processus peut être considéré comme un problème de correspondance de motifs entre graphe¹⁶, autrement dit comme un problème d'homomorphisme de graphe. Ce dernier peut être reformulé et résoud comme un problème de satisfaction de problème (CSP). Aussi, les auteurs ont proposé la construction des CSPs à partir des modèles de problème et d'un modèle en entrée. Une solution au CSP existe si le modèle en entrée est une instance d'un modèle de problème (MP). Une implémentation complète a été réalisée en utilisant notamment ILOG JSolver. Au final, le framework proposé assiste l'architecte logiciel dans toutes les étapes nécessaires à l'utilisation d'un design pattern.

3.7 Session 6 : Model Comprehension

Constructing Models with the Human-Usable Textual Notation, *Louis M. Rose, Richard F. Paige, Dimitrios S. Kolovos et Fiona A.C. Polack*

Pour pouvoir échanger, comprendre et manipuler des instances de méta-modèles, il est nécessaire de proposer une syntaxe concrète (i.e notation textuelle ou graphique) aux utilisateurs. Le développement de langages spécifiques à un domaine (DSL) est une approche possible. Les auteurs se sont focalisés sur l'utilisation d'une syntaxe générique, qui présente l'avantage de ne pas exiger une transformation entre les sorties du parser et les concepts abstraits du méta-modèle. Ainsi, le développement incrémental d'un méta-modèle n'oblige pas à réviser la syntaxe concrète associée, et favorise le prototypage rapide de

16. graph pattern matching problem

méta-modèle. Plus précisément, les auteurs se sont tournés vers HUTN (Human-Usable Textual Notation), une syntaxe concrète, générique et textuelle, proposée par l'OMG, et dont l'objectif est de manipuler des méta-modèles MOF, quel-qu'ils soient. Le fonctionnement de HUTN a été expliqué sur un exemple simple (un méta-modèle représentant une famille), mettant en valeur le caractère intuitif de la notation (qui est concise et lisible par un humain). Les bénéfices de HUTN ont été mis en valeur avec l'écriture rapide de jeux de tests générant des instances du métamodèle (i.e des modèles de famille). Cependant, les auteurs se sont aperçus qu'il n'existait aucune implémentation de référence de HUTN. L'implémentation proposée est construite à partir de ANTLR et de la plate-forme Epsilon. L'implémentation de HUTN utilise les techniques de l'ingénierie des modèles : un modèle arbre syntaxique abstrait (AST) est transformé en un modèle intermédiaire, qui est lui-même transformé en un modèle cible. L'utilisation du modèle intermédiaire s'explique par la difficulté à transformer le modèle AST directement dans le modèle cible. De plus, le modèle intermédiaire permet d'effectuer une phase de validation. L'originalité de leur implémentation (outre que c'est la seule à l'heure actuelle) repose sur la possibilité *i)* de refactoriser des informations inutiles en HUTN ; *ii)* de modifier le méta-modèle tout en régénérant automatiquement l'information HUTN précédemment associée ; *iii)* d'inférer un méta-modèle à partir de spécifications écrites en HUTN.

4 Models : conférence principale (Jeudi 2 octobre 2008)

Conférencier invité : Don Batory "The Objects and Arrows of Computational Design"

4.1 Session 7 : Model Management

Metamodel Matching for automatic Model Transformation Generation, *Jean-Rémy Falleri, Marianne Huchard, Mathieu Lafourcade et Clémentine Nebut*

Dans cet article, les auteurs proposent une approche pour développer des transformations de modèles automatiques entre deux méta-modèles. La génération de la transformation s'effectue en plusieurs étapes :

1. la génération de graphes étiquetés associés à chaque méta-modèle,
2. l'utilisation de l'algorithme *Similarity Flooding* sur les graphes produits pour faciliter la recherche d'une correspondance,
3. la génération d'un alignement entre les deux méta-modèles.

Lors de la première étape, selon la configuration employée, le graphe produit est différent, ce qui influe énormément sur les résultats de l'algorithme *Similarity Flooding*. Les auteurs ont présenté une approche automatique de détermination de correspondance entre deux méta-modèles à travers l'utilisation d'un algorithme de recherche. La principale partie du travail s'est effectuée sur la détermination du meilleur passage du méta-modèle vers un graphe "étiquette". Ils ont déterminé en tout six configurations différentes donnant des résultats plus ou moins probants pour les correspondances.

4.2 Session 9 : Metamodeling and Modularity

Formal Definition of MOF 2.0 Metamodel Components and Composition, *Ingo Weisemöller and Andy Schürr*

Le MOF¹⁷ est le langage le plus souvent utilisé pour la définition de DSLs¹⁸. Cependant, le manque de modularité rend parfois les modèles difficile à maintenir. Les auteurs ont ici défini une approche formelle de meta-modélisation orienté composant ainsi que la définition d’une opération de composition de méta-modèles approprié. C’est à travers un exemple alliant composition et modularité à base de composant qu’ils ont présenté leurs travaux.

Interfaces and Metainterfaces for Models and Metamodels, *Anders Hesselund and Andrzej Wasowski*

Actuellement, le problème de l’évolution et de l’adaptation des systèmes à base de composant est aussi bien connu dans le monde académique qu’industriel. Cependant, les solutions apportées divisent les scientifiques. D’un cotés, un composant est aperçue comme une boîte noire avec un interface et un seul point d’accès, de l’autre, celui-ci ne posséderait ni encapsulation, ni distinction entre les interfaces.

Les auteurs ont présenté ici une approche compositionnelle automatique pour une dépendance suivi à base d’utilisation d’interface. En partant de cette base, les auteurs ont présenter, entre autre, un langage d’interface pour la spécification d’interfaces et de méta-interfaces et une notion de composant d’interface par composition d’interfaces de modèles.

Le travail effectué fût présenté à travers deux exemples industriels :

- un système de planning de ressources d’entreprise,
- un système d’information de soins de santé.

4.3 Panel : "Past & Future of MDD"

Modérateur : Robert B. France

Participants à la table ronde :

Richard M. Soley, Chairman and CEO, Object Management Group, USA

Jeff Kramer, Professor of Distributed Computing, Dean of the Faculty of Engineering, Imperial College, UK

Bran Selic, Retired IBM Distinguished Engineer, Canada

Grady Booch, IBM Fellow and Chief Scientist, IBM Rational, USA

Patrick Farail, Airbus, France

Lors de la table ronde “Past & Future of MDD”, plusieurs interlocuteurs se sont succédés pour présenter leurs visions de l’ingénierie dirigée par les modèles (MDD). A noter que l’intervention de Graddy Booch en direct de 2nd Life a été écourtée à cause d’un problème réseau. Au fil des présentations, la question

17. Meta Object Facilities

18. Domain Specific Language

de l'exécutabilité des modèles s'est imposée. Selon Bran Selic, au départ était le code, lui succéda les modèles comme spécification et le code comme implémentation, puis l'interaction très forte entre modèles et code généré (période dans laquelle nous nous trouvons actuellement). Par conséquent, l'exécutabilité des modèles fût présenté comme une suite logique et le futur du MDD.

Le débat s'orienta aussi vers l'utilisation actuelle et passée du MDD chez Airbus avec l'exposé de Patrick Farail. Une fois celle-ci présentée, il justifia les motivations d'une approche par modèles et nous fit entrevoir le futur de la modélisation chez Airbus. Il conclut son intervention avec quelques interrogations ouvertes sur les méta-modèles et profils UML.

Lors de ce débat, la discussion se concentra sur les difficultés passées du MDD (difficultés à modéliser avec UML, compréhension des modèles, etc.), les anciennes approches ainsi que l'approche actuelle visant la génération de code et une synchronisation entre code et modèles. Les problèmes de consistance furent notamment mis en avant. Le futur pour le MDD entrevu par beaucoup serait de pouvoir se reposer sur l'exécution des modèles, mais bon nombre de questions restent en suspens avant de pouvoir faire tomber la barrière entre idée et réalité.

5 Models : conférence principale (Vendredi 3 octobre 2008)

5.1 Session 11 : Model Analysis

Integrating Performances Analysis in the Model Driven Development of Software Product Lines, *Rasha Tawhid, Dorina C. Petriu*

L'application d'un développement dirigé par les modèles dans la conception de lignes de produits logiciels (SPL) est envisagé dans cette présentation. En particulier, l'objectif est d'intégrer la dimension de performance (e.g. temps de réponse, débit, monopolisation, etc.) pour un produit donné de la SPL. Plus précisément, le profil MARTE (Modeling and Analysis of Real-time and Embedded systems) est utilisé pour annoter la SPL avec des paramètres de performance, qui deviennent ainsi partie intégrante des éléments¹⁹ réutilisables de la ligne.

La difficulté de ce travail de recherche est de parvenir à réduire la distance sémantique entre un modèle SPL et un modèle de performance. Dans l'approche présentée par Dorina Petriu, les feature modèles n'ont pas été utilisés au profit d'un profile UML capable d'exprimer les concepts de commonalité et de variabilité tels qu'on les trouve dans l'approche SPL (e.g option, alternative, groupes, etc). Ceci permet d'envisager la transformation suivante : à partir d'un modèle UML de la SPL et des annotations de performance associées, on veut pouvoir dériver un modèle UML + MARTE d'un produit particulier de la ligne. Ainsi, après analyse (i.e résolution) de la variabilité, le produit aura des propriétés de performance concrètes. Des détails sur l'implémentation et un exemple d'application ont été donnés pour illustrer l'approche.

19. assets

En conclusion, quelques perspectives de recherche sont énoncées et discutées : comment automatiser et faciliter la dérivation d'un modèle de produit via la SPL (e.g utilisation des techniques de la modélisation orientée aspect) ? est-ce que le profile MARTE a toutes les qualités requises pour permettre la réutilisation des annotations de performance ?

5.2 Session 13 : Adaptive and Autonomic Systems

Autonomic anagment Policy Specification : from UML to DSML, *Benoit Combemale, Laurent Broto, Xavier Crégut, Michel Daydé et Daniel Hagimont*

La conception de systèmes autonomiques est une voie prometteuse pour maîtriser l'augmentation (e.g en coût et en temps) des tâches de gestion. Le cas d'utilisation présenté concerne une application J2EE, qui nécessite d'éditer et de configurer de nombreux fichiers manuellement. Dans ce contexte, il s'agit par exemple de redémarrer automatiquement un serveur tomcat suite à une erreur ou bien de répliquer des serveurs suite à une charge trop importante. Chaque serveur à gérer est encapsulé dans un composant : l'environnement logiciel est une architecture à composants. L'objectif des auteurs est de faciliter l'implémentation d'encapsulateurs (wrappers) de programmes existants (e.g pour surveiller le programme), de décrire le déploiement des programmes et enfin de permettre leurs reconfigurations autonomiques.

La première approche a consisté à utiliser le modèle de composants Fractal. Cependant, plusieurs inconvénients ont été rapportés : apprentissage d'un nouveau framework (encore un!), écriture des wrappers demandant beaucoup de travail et de temps – processus d'ailleurs sujet à des erreurs – et difficulté d'exprimer des politiques de reconfiguration. Aussi, les auteurs proposent une adaptation (i.e spécialisation) de la sémantique d'UML à même de fournir des formalismes de haut niveau pour les besoins de leur application. Ensuite, plusieurs DSLs ont été associés pour gérer l'encapsulation, le déploiement et la reconfiguration des composants. La syntaxe concrète et les outils des DSLs ont également été développés. Il est maintenant possible pour un utilisateur final de spécifier des politiques de reconfiguration en utilisant le DSL de reconfiguration des composants.

Xavier Cregut conclue en détaillant les bénéfices de l'approche (e.g renforcement de la cohérence des modèles, vues de haut niveau, etc.). Les travaux présentés ouvrent de nouvelles perspectives quant à la gestion dirigée par les modèles des systèmes informatiques.

5.3 Panel : "Addressing the Challenges of Multi-Modeling for Domain-Specific Modeling Languages"

Modérateur : Doug Schmidt

Participants à la table ronde :

Trip Denton, Lockheed Martin Advanced Technology Labs, USA

Edward A. Lee, University of California at Berkeley, USA
Jose Meseguer, University of Illinois at Urbana-Champaign, USA
Douglas C. Schmidt, Vanderbilt University, USA

5.4 Session 15 : Evolution and Reverse Engineering

Heterogeneous Coupled Evolution of Software Languages, *Sander Vermolen, Eelco Visser*

Les modèles doivent évoluer lorsque le métamodèle dont ils sont des instances est modifié : c'est le processus de co-évolution. Lorsqu'un métamodèle est adapté, les modèles sont inconsistants avec le nouveau métamodèle, et il s'agit de migrer l'ensemble des modèles. On parle également d'évolution couplée.

Sander Vermolen l'illustre à partir de l'exemple d'un modèle de données d'un wiki. En prenant en compte les changements de besoins et les contraintes de maintenance, le modèle de données est modifié au cours du temps (e.g. ajout du concept de groupe pour gérer les utilisateurs). Or, le wiki a déjà stocké des pages, des utilisateurs, etc. Les données doivent donc être modifiées (i.e. transformées) en fonction de l'évolution du modèle de données.

Plutôt que de résoudre le problème de l'évolution couplée dans un domaine spécifique et dans un espace technique homogène (e.g. grammaire, schéma de bases de données, DTD et XML, etc.), les auteurs présentent et soutiennent le concept d'évolution couplée *hétérogène* qui s'applique à l'évolution de n'importe quel type de langage. L'architecture proposée permet d'automatiser le processus d'évolution couplée en dérivant automatiquement un langage de transformation spécifique à un domaine tout en associant un interpréteur pour exécuter les transformations. Il est ainsi possible de migrer automatiquement les données suite à une évolution du métamodèle en utilisant des concepts abstraits de haut niveau et des transformations génériques. L'approche est de plus complètement outillée et a donné lieu à des expérimentations concluantes.

5.5 Session 16 : Modeling Language Semantics

A Lightweight Approach of a Modeling Language Formal Semantics of a Modeling Language, *Pierre Kelsen, Qin Ma*

La définition d'une sémantique formelle d'un langage de modélisation est importante pour raisonner à partir du langage et ainsi fournir un support aux outils de modélisation. Les approches actuelles pour formaliser les sémantiques sont souvent difficiles à mettre en oeuvre, ce qui pourrait expliquer la profusion de langages de modélisation manquant de description d'une sémantique formelle.

Les auteurs proposent ici une nouvelle approche pour la définition d'une syntaxe abstraite, de la sémantique statique et dynamique basé sur les techniques de métamodélisation et le langage Alloy. Les deux principaux avantages de l'approche basés sur Alloy est la simplicité de la notation ainsi que la facilité d'analyse qui est plus facilement automatisable. La présentation s'est articulée autour de la présentation d'un exemple concret et d'une comparaison vis-à-vis des autres techniques usuellement utilisées.

6 Bilan

Nous remercions tout particulièrement l'action IDM de nous avoir permis d'assister à la conférence. Bien entendu, tout commentaire sur ce document est le bienvenu.

Enfin, l'édition 2009 de Models aura lieu à Denver (Colorado, Etats-Unis) du 4 au 9 octobre.