End-User Modelling

Albert Patrick *(IBM, Paris)*, Blay-Fornarino Mireille *(I3S, Sophia-Antipolis)*, Collet Philippe *(I3S, Sophia-Antipolis)*, Combemale Benoit *(IRISA, Rennes)*, Dupuy-Chessa Sophie *(LIG, Grenoble)*, Front Agnès *(LIG, Grenoble)*, Grost Anthony *(ATOS, Lille)*, Lahire Philippe *(I3S, Sophia-Antipolis)*, Le Pallec Xavier *(LIFL, Lille)*, Ledrich Lionel *(ALTEN Nord, Lille)*, Nodenot Thierry *(LIUPPA, Pau)* et Pinna-Dery Anne-Marie *(I3S, Sophia-Antipolis)* et Rusinek Stéphane *(Psitec, Lille)*

1 Contexte et domaines applicatifs

En 2006, le *Standish Group* indiquait que l'amélioration du taux de réussite des projets informatiques (passant de 16 à 32%) provenait de différents facteurs, dont le plus important était l'implication grandissante des utilisateurs finaux. Le mouvement syndicaliste scandinave (début 70) nommé “conception participative” avait déjà influencé l'ingénierie logicielle dans les années 80 dans ce sens: l'adhésion des employés pour un nouvel outil peut s'obtenir par leur participation à sa conception. Ce sont les démarches agiles, officialisées en 2001, qui ont popularisé ce principe d'implication des utilisateurs dans la conception logicielle. Parallèlement, l'augmentation de la production logicielle et de la taille des applications informatiques a accentué la préoccupation de réutilisation logicielle. Une réponse à celle-ci a été d'accorder plus d'importance à la modélisation dans la conception logicielle : un modèle fournit un meilleur support pour la discussion, la compréhension d'une architecture ou d'un composant et demeure moins impacté par les évolutions technologiques.

L’utilisation de plus en plus intensive des « wiki » et la mise à disposition d’outils google témoignent de l’intérêt de donner aux utilisateurs les moyens de « programmer » leurs propres outils. Cette thématique se retrouve au sein d’éditeurs d'applications *mash up* comme Yahoo Pipes, MS Pop Fly. Ces éditeurs, très en vogue actuellement, sont souvent basés sur des éditeurs de modèles graphiques très accessibles et en principe au moins lisible par un grand nombre d'utilisateurs “avertis”. De manière plus industrielle, la construction d’usines logicielles et les environnements de modélisation dédiés à des domaines spécifiques entrent également dans ce cadre, en donnant à l’utilisateur des langages de modélisation adaptés à son métier. Le défi proposé ici est donc : *comment donner aux experts d’un domaine les moyens de construire leur propre système informatique en utilisant des techniques de modélisation ?*

Les enjeux sont une plus grande productivité et une meilleure adéquation des produits aux problèmes. En particulier en permettant aux utilisateurs d'adapter le logiciel, une plus grande réactivité/agilité est obtenue. Ces enjeux ne seront atteints que si les artefacts de modélisation donnés sont adéquats et si les produits résultants sont fiables et opérationnels.

Les risques sont la définition de méga-métamodèles non exploitables, l’obtention de produits inutiles, non performants, non réutilisables.

Nous proposons d’aborder ce défi par la construction d’environnements de modélisation dédiés à des domaines spécifiques. Des systèmes tels que CENTAUR [1] dans les années 80-90 avaient des objectifs similaires mais étaient orientés langages de programmation. Le défi que nous proposons en s’appuyant sur ces acquis envisage une appréhension de la “programmation” dirigée non pas par la syntaxe mais par les concepts, les usages et les contraintes du métier. Le défi est de faciliter la construction de ces environnements en utilisant des techniques de modélisation, en particulier la modélisation sous la forme de diagrammes.

2 Verrous

Les verrous identifiés se situent au niveau (i) des représentations à donner aux multiples profiles d'utilisateurs qui doivent être en adéquation avec leur usage, (ii) de l’interprétation de ces notations par le système pour assurer l’aide, la validation et la collaboration et l’exécution éventuelle des systèmes produits, (iii) de l’impact sur les démarches usuelles de développement.

1. Représentation métier :
   - comment éviter la surcharge cognitive au niveau des syntaxes concrètes dédiées aux utilisateurs ?
   - comment supporter la montée en compétences (connaissances du système) d'un utilisateur ?
   - comment supporter les multi-représentations lors de travaux collaboratifs ?
2. Interprétation des modélisations métier :
   - comment supporter plusieurs représentations d'un même système (multi-utilisateurs, multi-niveaux) ?
   - comment rendre opérationnelles ces représentations en limitant les coûts de développements et en assurant la cohérence des systèmes construits ? (intégration de l'existant, incrément, agilité des développements ?
   - comment permettre à l'utilisateur final d'interagir avec le système de manière à en adapter le comportement (e.g., systèmes adaptables) ?
   - Un des verrous à lever est le juste appariement entre des approches dîtes "formelles" indispensables à ancrer le procédé dans une logique de fiabilité, et des approches de modélisation plus flexibles qui supportent une expression plus "naturelle" des domaines métiers.
3. Démarches de développement :
   - comment supporter l'agilité de la construction, la robustesse des environnements construits tout en acceptant les incohérences nécessaires au développement collaboratif, …

3 Fondements

L'IDM apporte un ensemble de fondements permettant d'aborder la problématique soulevée par ce défi [2]. En effet, la montée en abstraction supportée par l'IDM pour la construction de systèmes complexes réduit l'écart entre la spécification et la conception/développement d'un système. En premier lieu, la prise en compte des différents profils d'utilisateurs finaux nécessitent de définir des représentations adéquates pour chacun d'entre eux. Cette adéquation nécessite une étude des propriétés cognitives des formalismes utilisés. Pour cela, nous pourrons nous baser sur les synthèses de Bernard Morand [16] sur les propriétés cognitives des diagrammes (du point de vue de la psychologie cognitive, de la linguistique…) comme sur les travaux de Gerald Lohse [17] qui sont plus spécifiques aux diagrammes représentant des systèmes informatiques. D'autres travaux ont porté sur la compréhensibilité globale d'une notation et ont permis d'aboutir à des protocoles expérimentaux réutilisables et utilisées dans le monde industriel [18]. Dans tous ces cas, l'approche utilisée est celle des expériences utilisateurs.

Mais une autre approche, plus automatisable, est possible pour évaluer la qualité d'un modèle ou d'un langage : elle se base sur l'utilisation de métriques qui peuvent être définies sur les modèles [19] ou les métamodèles [20]. Il existent donc de nombreuses possibilités pour appréhender ce qui a été définie dans les frameworks de qualité [21][22] comme la qualité pragmatique, c'est-à-dire la qualité dépendant de l'interprétation des utilisateurs et des concepteurs.

La définition d'un langage dédié (*Domain Specific Language*, ou DSL), et plus récemment de langage de modélisation dédié (*Domain Specific Modeling Language*, ou DSML) a été étudié au travers de techniques telles que les profils UML (consistant à spécialiser UML pour des besoins particuliers), ou la métamodélisation offrant toute l'ingénierie pour la définition d'un nouveau langage [9]. L'outillage de celui-ci peut par ailleurs être facilité à l'aide d'approches génératives ou génériques, telle que par exemple GMF[1] qui permet de générer automatiquement des outils de modélisation métier graphique pour un langage dédié donné. Le traitement automatique, à base de modèles, de langue naturelle ou de règles métiers est également un axe de recherche intéressant pour la résolution de ce défi. La publication récente par l'OMG de la norme *Semantics for Business Vocabulary and Rules* (SBVR)[2] permet par exemple d'interpréter formellement sous la forme de modèle un anglais structuré [23]

Les travaux sur les approches par points de vues devront également être pris en considération de manière à bien préciser l'intention de chaque type d'utilisateurs [15], de même que les travaux sur les usines logicielles [8]. D'autres travaux sur l'extension des métamodèles, tant syntaxique que comportementale (e.g., la modélisation orientée aspect [3,4]) offre également des moyens d'adapter ou de préciser un langage existant pour des besoins spécifiques.

---

[1] Cf. http://www.eclipse.org/gmf/

[2] *Semantics of Business Vocabulary and Business Rules* 1.0 specification (2008), http://www.omg.org/spec/SBVR/1.0/

D'autre part, des techniques de composition [5,6,10] et de transformation de modèles [7] permettent de prendre en compte les modèles issus de différentes préoccupations de manière à obtenir un modèle unique du système complet. Ces techniques permettent ainsi de rendre productif les modèles métiers et de les considérer comme des artefacts terminaux du processus de développement. Le couplage transparent des représentations métiers avec les méthodes formelles [13] devra également être pris en considération afin d'assurer la fiabilité des systèmes construits.
L'utilisation de ces techniques doit toutefois s'intégrer dans une démarche agile favorisant un cycle court de manière à permettre à l'utilisateur de voir très rapidement le résultat de son travail. Des adaptations dynamiques de l'environnement doivent également être envisagées [24,25]. La nécessité d'intégrer un processus itératif est donc indispensable et doit s'appuyer pour cela sur la traçabilité entre les modèles métiers et le système final obtenu. Cette traçabilité peut par exemple être utilisée pour indiquer les impacts sur le système final de chaque modification d'un modèle métier.

Enfin, l'assistance à la modélisation peut s'appuyer d'une part sur les modèles paramétrés pour permettre une modélisation par la réutilisation de briques génériques [14], et d'autre part sur le typage de modèle [11] pour permettre la généricité de certain traitement [12].

4 Usages et impacts sociétaux

- La programmation des applications est donnée aux experts des domaines, la construction des environnements est donnée aux experts logiciels. Mais les environnements eux-mêmes doivent pouvoir évoluer par des cycles courts.
- Les services informatiques représentent une activité économique importante (> 700Md $ en 2008). C'est pourtant un secteur dont la majorité des projets sont des échecs. L'implication des utilisateurs est le facteur de réussite préconisé mais sa mise en œuvre reste difficile. Un objectif majeur du défi est de diminuer voir supprimer cette difficulté.

5 Jalons, démarche

- Circonscrire la complexité par des études de cas :
  - Expérimentations sur des classes d'applications pour lesquelles les langages/formalismes sont maîtrisables par les "personnes métier" (ex. : Business Rules, modélisation de workflows en analyse d'images médicales, modélisation de systèmes de diffusions d'informations en milieu scolaire, domotique),
- Définition des artefacts de modélisation (formalismes) cognitivement adaptés aux différents rôles (ou personnes) impliquées
  - Processus de construction attendu, validations, tests : les artefacts se définissent relativement à leur usage.
  - Les experts d'un domaine peuvent adapter ou faire évoluer l'application uniquement sur la partie métier.
- Corrélation entre modélisation métier et mise en œuvre au niveau des plates-formes :
  - Dans un premier temps, des experts en programmation définissent les méthodologies, processus, outils de tests et de validation qui permettent de construire, modifier, valider des "modèles métiers" et d'assurer la cohérence des systèmes construits : "*Controlled Agility*".
  - Dans un second temps, des patrons (transformations, mécanismes de profiling) sont utilisés pour, sur la base de la définition de métamodèles métiers, obtenir les outils associés (validation, tests, …). Partiellement atteint dans le cadre des langages de programmation, cet objectif devrait bénéficier des avancées en matière de métamodélisation.
  - En obtenant une certaine automatisation de la production des outils, les utilisateurs finaux devraient pouvoir faire évoluer leur propre environnement de modélisation dans la limite des contraintes nécessaires pour assurer la cohérence des environnements et des systèmes construits.

References

1. P. Borras, D. Clément, Th. Despeyroux, J. Incerpi, G. Kahn, B. Lang, and V. Pascual, *Centaur: the system* , Proceedings of SIGSOFT'88, Third Annual Symposium on Software Development Environments (SDE3), Boston, USA, 1988.
2. Jean-Marie Favre, Jacky Establier, and Mireille Blay-Fornarino, editors. L'ingénierie dirigée par les modèles : au-delà du MDA. Hermes-Lavoisier, Cachan, France, February 2006.
3. Jacques Klein, Franck Fleurey, and Jean Marc Jézéquel. Weaving multiple aspects in sequence diagrams. Transactions on Aspect-Oriented Software Development (TAOSD), LNCS 4620 :167–199, 2007.
4. Jean-Marc Jézéquel. – Model driven design and aspect weaving. – Journal of Software and Systems Modeling (SoSyM), 7(2):209–218, may 2008.
5. Robert France, Franck Fleurey, Raghu Reddy, Benoit Baudry, and Sudipto Ghosh. Providing support for model composition in metamodels. In EDOC'07 (Entreprise Distributed Object Computing Conference), Annapolis, MD, USA, 2007.
6. Olivier Barais, Jacques Klein, Benoit Baudry, Andrew Jackson, Siobhan Clarke, *Composing Multi-View Aspect Models*, 7th IEEE International Conference on Composition-Based Software Systems (ICCBSS) (2008)
7. Jean Bézivin, Salim Bouzitouna, Marcos Didonet Del Fabro, Marie-Pierre Gervais, Frédéric Jouault, Dimitrios S. Kolovos, Ivan Kurtev, and Richard F. Paige. A canonical scheme for model composition. In Arend Rensink and Jos Warmer, editors, ECMDA-FA, volume 4066 of Lecture Notes in Computer Science, pages 346–360. Springer, 2006.
8. Carlos Parra , Rafael Leano , Xavier Blanc, Laurence Duchien, Nicolas Pessemier, Chantal Taconet and Zakia Kaziaoul, *Dynamic Software Product Lines for Context-Aware Web Services*., in Enabling Context-Aware Web Services: Methods, Architectures, and Technologies. Chapman and all, July 2009
9. F. Barbier, *UML 2 et MDE - Ingénierie des modèles avec études de cas*, Dunod, 2005
10. Mireille Blay-Fornarino, *"Interprétations de la composition d'activités"*, HDR Thesis University of Nice-Sophia Antipolis, 1-201 pages, Sophia Antipolis, France, apr 2009
11. Jim Steel and Jean-Marc Jézéquel. – On model typing. – Journal of Software and Systems Modeling (SoSyM), 6(4):401–414, December 2007.
12. Naouel Moha, Vincent Mahé, Olivier Barais, and Jean-Marc Jézéquel. – Generic Model Refactorings. – In ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems (MODELS'09), Denver, Colorado, USA, Oct 2009.
13. B. Combemale, X. Crégut, P.-L. Garoche and X. Thirioux – Essay on Semantics Definition in MDE. An Instrumented Approach for Model Verification – Journal of Software (JSW), 4(6), December 2009
14. Alexis Muller, Olivier Caron, Bernard Carré, Gilles Vanwormhoudt, *On Some Properties of Parameterized Model Application* in Proceedings of ECMDA-FA'2005: First European Conference on Model Driven Architecture - Foundations and Applications, Nuremberg, November 2005, LNCS 3748, A. Hartman and D. Kreische Eds.
15. Adil Anwar, Sophie Ebersold, Bernard Coulette, Mahmoud Nassar, Abdelaziz Kriouile. A Rule-Driven Approach for composing Viewpoint-oriented Models. Dans : Journal of Object Technology, ETH Swiss Federal Institute of Technology, p. 1-26, 2010
16. B. Morand, Logique de la Conception, Figures de sémiotique générale d'après Charles S. Peirce, L'Harmattan, Collection L'Ouverture Philosophique, 294 p.
17. G. L. Lohse, D. Minb and J. R. Olsonc, Cognitive evaluation of system representation diagrams, Information & Management, Volume 29, Issue 2, August 1995, Pages 79-94
18. (Patig 2008) : S. Patig, A practical guide to testing the understandability of notations, Conferences in Research and Practice in Information Technology Series; Vol. 325, Proceedings of the fifth on Asia-Pacific conference on conceptual modelling - Volume 7, 2008, pp 49-58
19. (Lange 2005) : C. Lange, M. Chaudron, Managing Model Quality in UML-Based Sotware Delevopment, Proc. of the 13th workshop on software Tachnology and Engineering Practice (STEP'05), 2005, pp. 7-16.
20. (Rossi 1996) M. Rossi et S. Brinkkemper, Complexity metrics for systems development methods and techniques, Information Systems, Vol. 21, num 2, 1996, pp. 209-227.
21. (Linbdland 1994) O. Lindland, G. Sindre, A. Solvberg, Understanding Quality in Conceptual Modeling, IEEE Software, Vol. 11, 1994, pp. 42-49.
22. (Krogstie 1998) J. Krogstie, Integrating the Understanding of Quality in Requirements Specification and Concptual Modeling, Software Engineering Notes, ACM SIGSOFT, Vol 23, num 1, 1996, pp. 86-91
23. Mathias Kleiner, Patrick Albert and Jean Bézivin, Parsing SBVR-based controlled languages, In ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems (MODELS'09), empirical track, Denver, Colorado, USA, Oct 2009, pages 122-136.
24. Reza Razavi, Noury Bouraqadi, Joseph W. Yoder, Jean-François Perrot, Ralph E. Johnson: *Language support for adaptive object-models using metaclasses*. Computer Languages, Systems & Structures 31(3-4): 199-218 (2005)
25. Razavi, Reza, Kirill Mechitov, Gul Agha, Jean-Francois Perrot. "Ambiance: A Mobile Agent Platform for End-User Programmable Ambient Systems," J.C. Augusto and D. Shapiro (eds.), Advances in Ambient Intelligence, Frontiers in Artificial Intelligence and Applications (FAIA), vol. 164, IOS Press, 2007