Spécification dans une vision IDM des procédés de développement logiciel

Samba Diaw[1] et Rédouane Lbath[1] et Bernard Coulette[1]

1 : Université de Toulouse, Laboratoire IRIT-UTM, 5, allées A. Machado 31058 TOULOUSE CEDEX 9

{diaw, lbath, coulette}@univ-tlse2.fr

RESUME. L'ingénierie dirigée par les modèles (IDM) a pour principal objectif d'accroitre la productivité et d'améliorer la qualité des logiciels à travers la manipulation des modèles dans tout le cycle de vie du développement logiciel. La finalité étant de pouvoir utiliser non seulement les modèles à des fins de documentation et de description mais aussi de production. Avec cette nouvelle approche, nous assistons à une émergence de nouveaux procédés logiciels appelés procédés IDM. Le but de ces procédés est de guider les développeurs dans l'élaboration et la transformation des modèles. Cependant, un langage de modélisation de procédés bien outillé pour décrire, réutiliser, mettre en œuvre et faire évoluer ces procédés fait défaut. Dans ce contexte, nous présentons une nouvelle approche basée sur les transformations de modèles pour spécifier les procédés IDM.

MOTS-CLES : Ingénierie dirigée par les modèles (IDM), Architecture dirigée par les modèles (MDA), Transformation de modèle, Langage de Modélisation de Procédés(LMP), Développement guidé par les modèles, procédé logiciel IDM

ABSTRACT. The main objective of MDE (Model-Driven Engineering) is to increase software productivity and to improve software quality by manipulating models in all the software development lifecycle. The finality is not only to use models for documentation and description, but also for production. With this new approach, new software development processes called MDE processes have started to emerge. The goal of these processes is to guide developers in elaboration and transformations of models. However, a tool-supported Process Modeling Language (PML) for describing, reusing, enacting and evolving MDE processes is still lacking. In this context, this paper presents a new approach based on model transformations for MDE processes specification.

KEYWORDS: Model-Driven Engineering (MDE), Model-Driven Architecture (MDA), Model Transformation, Process Modeling Language (PML), MDD (Model-Driven Development), MDE Software Process

1. Introduction

L'Ingénierie Dirigée par les Modèles (IDM) connue sous le terme MDE en Anglais est une discipline récente du génie logiciel qui promeut les modèles en entités de première importance dans le développement logiciel [2]. Persuadés que le modèle est devenu le paradigme majeur par lequel l'industrie du logiciel pourra lever le verrou de l'automatisation du développement, des organismes tels que l'OMG (Object Management Group) et les chercheurs en génie logiciel, cherchent à enrichir les métamodèles qui sont utilisés dans la conception d'applications ou à en définir de nouveaux, à faciliter la création de nouveaux espaces technologiques plus adaptés aux besoins des utilisateurs, et à faciliter les différentes étapes de modélisation nécessaires à l'élaboration d'un produit fini. Ainsi, pour obtenir un produit fini qui répond aux attentes des utilisateurs, il est nécessaire de pouvoir transformer des modèles d'un niveau d'abstraction à un autre ou d'un espace technologique à un autre. Le processus de développement des systèmes appelé procédé IDM (MDE Process en Anglais) [3, 4, 5, 6, 10] peut alors être vu comme une séquence de transformations de modèles partiellement ordonnée, chaque transformation prenant un ou des modèles en entrée et produisant un ou des modèles en sortie, jusqu'à l'obtention du code exécutable.

L'émergence de l'IDM a fait naitre une nouvelle démarche de développement logiciel basée sur les modèles et les opérations qu'on peut leurs appliquer (transformation, fusion, composition, raffinement, synthèse, etc.). Cette nouvelle démarche appelée procédé IDM (processus en Y de MDA [9], procédé UWE (UML-Based Web Engineering) [4,5], etc.) a pour objectif de guider un développement logiciel dirigé par les modèles. Les procédés IDM permettent de décrire les activités d'un développement logiciel orienté modèles par le biais des transformations de modèles mais ils souffrent d'une absence de langages outillés pour les décrire, les réutiliser, les mettre en œuvre et les faire évoluer. Des LMPs tels que le standard de l'OMG SPEM 2.0 (Software Process Engineering Metamodel) [9], UML4SPM [1], etc., ne prennent pas en compte dans leur métamodèle les notions de base de l'IDM (modèles, métamodèles et transformations de modèles) sur lesquelles reposent la description des procédés IDM.

Dans ce contexte, nous proposons une nouvelle approche pour la spécification des processus de développement qui prend en compte les notions de base de l'IDM. Cette approche sera basée sur un langage générique et flexible dénommé SPEM4MDE (SPEM for MDE). Le métamodéle associé à notre langage étend certains concepts de SPEM 2.0 et d'UML 2.2 [8]. Il est dédié à la modélisation et à la mise en œuvre des procédés de développement logiciel dirigés par les modèles.

Cet article est organisé comme suit : la section 2 introduit la problématique et les objectifs de nos travaux. La section 3 présente le métamodéle SPEM4MDE de notre

approche. Dans la section 4, nous présentons un extrait du procédé UWE (UML-Based Web Engineering) pour illustrer notre approche. Nous concluons par la section 5 qui résume les points abordés dans cet article et les perspectives. Par manque de place, seul l'aspect statique des procédés IDM sera traité dans cet article.

2. Problématique et Objectifs

L'IDM a radicalement changé notre façon de concevoir les applications en décrivant à la fois le problème et sa solution en utilisant des modèles et en établissant une méthodologie qui montre le passage d'un problème à sa solution par le biais des transformations. Avec cette nouvelle donne, la communauté du génie logiciel ne cesse de voir fleurir de nouveaux procédés appelés procédés IDM. Ces nouveaux procédés, bien que s'inscrivant dans une démarche MDA, ont besoin de langages adéquats et d'outils support pour leur description, leur réutilisation, leur mise en œuvre et leur évolution. Pour combler ce vide, des LMPs (Langages de Modélisation de Procédés) tels que le standard SPEM 2.0, UML4SPM, etc. ont tenté de proposer un formalisme pour représenter ces procédés. Malheureusement, les concepts proposés par ces LMPs sont trop génériques pour représenter ces procédés qui utilisent spécifiquement les notions de base de l'IDM (modèles, métamodèle, transformations de modèles, etc.).

Notre objectif est donc de proposer une nouvelle approche pour spécifier les procédés IDM afin de palier ce manque qui n'a pas pu être résolu par ces LMPs. L'objectif visé est de décrire ces procédés et de guider leur mise en œuvre. Nous rappelons que par manque de place, nous présentons seulement les concepts relatifs à la description statique des procédés IDM.

3. Notre approche pour la spécification des procédés IDM

Dans cette section, nous présentons les concepts qui définissent notre langage de modélisation de procédés IDM nommé SPEM4MDE. Pour cela nous utilisons la technique de métamodélisation, une activité du génie logiciel qui consiste à définir un langage de modélisation par le biais d'un modèle appelé métamodèle. Le métamodéle de SPEM4MDE est composé de deux paquetages : *MDE Process Structure* qui permet de décrire statiquement les procédés de développement IDM et *MDE Process Behavior* qui permet de mettre en œuvre ces procédés IDM (exécution du procédé par des acteurs humains). Pour des raisons de place, seul le paquetage MDE Process Structure sera présenté. La figure 1 ci-aprés montre les concepts du paquetage MDE Process Structure. Nous donnons par la suite, la description de chaque concept de MDE Process Structure.

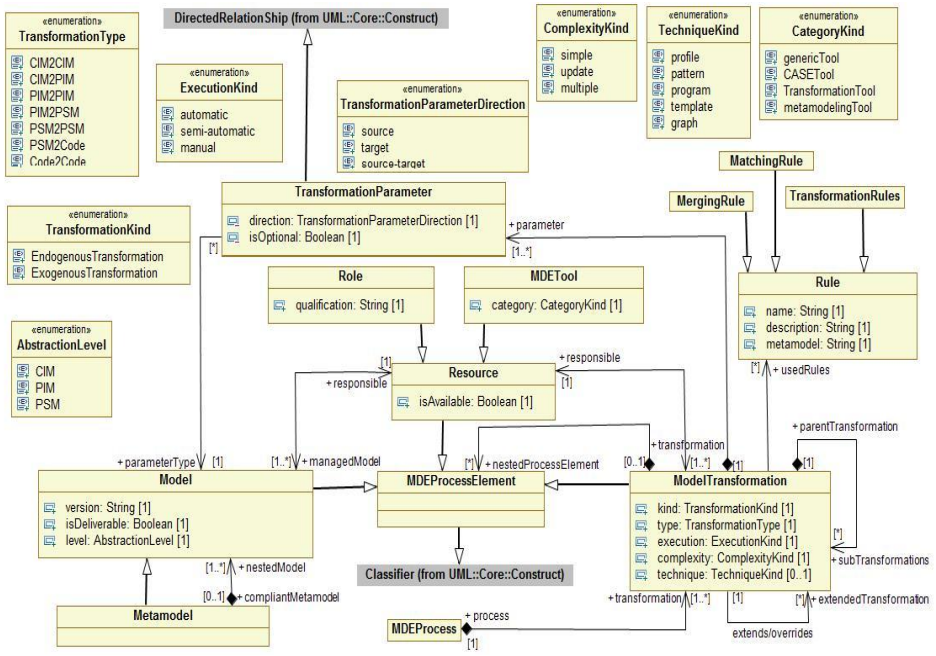

Figure 1. *Présentation du paquetage MDE Process Structure*

– *MDEProcess* : c'est un concept qui décrit un procédé IDM. Un procédé IDM est un procédé logiciel basé sur une séquence de transformations de modèles partiellement ordonnée.

– *MDEProcessElement* : c'est un concept abstrait pour représenter une généralisation des éléments d'un procédé IDM (transformations de modèles, ressources utilisées (rôles et outils IDM), modèles, métamodèles).

– *Model* : ce concept permet de représenter n'importe quel modèle dans un développement logiciel, qu'il soit conforme à un métamodéle ou non. Un modèle est une vue abstraite d'un ou d'une partie d'un système réel : on dit que le modèle représente le système (exemple de modèles : diagramme de classes UML, diagramme de cas d'utilisation, etc.). Ce concept a des propriétés : *version* pour connaitre la version courante du modèle manipulé ; *isDeliverable* pour connaître si le modèle est un livrable ou un produit intermédiaire ; *level* qui permet de préciser le niveau d'abstraction du modèle selon l'approche MDA. Les niveaux d'abstraction sont : CIM (Computational Independent Model), PIM (Platform Independent Model) et PSM (Platform Specific Model). Un modèle est dit indépendant d'une plate-forme si aucun de ses éléments n'utilise les fonctionnalités offertes par la plate-forme

– *Metamodel* : ce concept permet de représenter un métamodéle dans un développement logiciel. Un métamodéle modélise ou représente le langage de modélisation. C'est un modèle qui décrit les concepts d'un langage et leurs relations. Exemple : *le métamodéle d'UML* qui décrit les concepts qui permettent de représenter l'ensemble des modèles UML.

– *ModelTransformation* : c'est un concept qui permet de représenter l'activité de transformations de modèles dans un développement logiciel dirigé par les modèles. Une transformation de modèles est une activité qui consiste à générer un ou plusieurs modèles cibles conformes à un métamodéle à partir d'un ou de plusieurs modèles sources conforme au même métamodéle (transformation endogène) ou à un autre métamodéle (transformation exogène).Une transformation de modèles peut être de plusieurs types : CIM2CIM (ex. , restructuration d'un diagramme de cas d'utilisation), CIM2PIM (ex., passage d'un diagramme de cas d'utilisation vers un diagramme de classes), PIM2PIM (ex., restructuration d'un diagramme de classes ou transformation de ce diagramme vers un modèle entité-relation), PIM2PSM (ex., passage d'un diagramme de classes vers un modèle de composants EJB), PSM2PSM (ex., restructuration d'un modèle de composants EJB ou migration d'une plate-forme COBOL vers une plate-forme J2EE), PSM2Code (ex., génération de code java à partir du modèle de composants EJB) et enfin Code2Code (enrichir un code Java généré pour obtenir le code total de l'application).

Une transformation peut aussi être : automatique (si elle ne requiert aucune décision de la part des utilisateurs du système), semi-automatique (si l'utilisateur choisit les éléments du modèle source à transformer) ou manuelle (si l'utilisateur produit le modèle de sortie). Ces définitions sont tirées de [5].

Une transformation peut avoir plusieurs types de complexité : simple (un seul modèle en entrée vers un seul modèle de sortie), de mise à jour (update) (c'est le cas lorsqu'un modèle est enrichi, modifié ou remplacé), multiple (plusieurs modèles en entrée vers plusieurs modèle de sortie).

Une transformation (*parentTransformation*) peut être décomposée en plusieurs transformations (*subTransformations*). Une transformation peut étendre (*extend*) ou remplacer (*override*) une autre transformation.

Pour spécifier une transformation plusieurs techniques peuvent être utilisées : les profils, un patron de conception (Pattern), un programme, un template (modèle paramétré), les graphes dirigés et étiquetés. Une transformation peut aussi être spécifiée en utilisant des règles déclaratives ou impératives (Rule)

Les paramètres d'une transformation sont des modèles ou des métamodèles. Ces paramètres ont une *direction* (source, cible ou source-cible).

– *Rule* : c'est un concept qui permet de spécifier les règles de transformation. Nous distinguons les types de règles suivants : *MergingRule* (règles pour fusionner des

modèles), *MatchingRule* (règles de correspondance pour composer des modèles) et *TransformationRule* (règles pour transformer un modèle d'entrée vers un modèle de sortie). Une règle a un *nom*, une *description* et est spécifiée dans un langage associé à un *métamodéle* (QVT, ATL, etc.).

– *Resource* : C'est un concept abstrait pour représenter une généralisation des ressources utilisées dans le développement logiciel (*Role*, *MDETool*). Une ressource à une *responsabilité* sur une transformation de modèles. Les rôles représentent les qualifications et compétences nécessaire pour réaliser la transformation. Les outils IDM (concept *MDETool*) sont des outils utilisés dans la cadre de l'IDM pour faire de la transformation de modèles ou de la vérification sémantique des modèles. Nous distinguons quatre catégories d'outils IDM représentés par la propriété *category*: les outils génériques (*genericTool*) (outils de transformations de graphes, outils du monde XML, etc.), les outils intégrés aux AGL (*CASETool*) (par exemple Objecteering MDA Modeler ou Rational Software Modeler), les outils de transformations (*transformationTool*) (ADT, QVTEclipse, ...) et les outils de métamodélisation (*metamodelingTool*) (EMF/Ecore, Kermeta, TOPCASED, OpenEmbedded, etc.).

4. **Exemple : Procédé UWE**

Dans cette section nous présentons d'abord un extrait du procédé UWE (UML-Based Web Engineering), puis décrivons une partie de cet extrait avec les concepts de notre langage SPEM4MDE défini dans la section précédente. Pour représenter une partie de l'extrait du procédé UWE, nous avons traduit le métamodéle du langage SPEM4MDE en un profil UML. Ce choix s'explique par le fait que nous ne disposons pas pour le moment d'un environnement qui implémente notre langage et qu'en plus les profils UML ont l'avantage de réutiliser l'outillage existant d'UML. Par manque de place, le profil UML de SPEM4MDE n'est pas présenté dans cet article.

UWE est utilisé pour le développement d'applications web basé sur les transformations de modèles. La partie gauche de la figure 2 donne une vue d'ensemble d'un procédé UWE tiré de l'article [5]. Le procédé démarre avec les modèles de type CIM *RequirementsModels* qui définissent les exigences de l'application web et le processus métier. Les modèles fonctionnels *Content Model* et *NavigationModel*, qui sont de type PIM, représentent les différentes préoccupations de l'application web. Ils sont dérivés des modèles *RequirementsModels* par le biais de la transformation *Req2Functional*. Les modèles fonctionnels incluent un modèle métier (*ContentModel*) pour représenter le contenu de l'application et un modèle de navigation (*NavigationModel*) qui représente les nœuds de l'application web (indexes, menus, etc.) et les liens existant entre ces nœuds. Les modèles fonctionnels sont par la suite composés et fusionnés pour donner un autre modèle (*BigPicture Model*) qui permet de

valider et d'intégrer les modèles fonctionnels. Par la suite, les modèles PSM (modèle de code J2EE, modèle de code. Net.) sont dérivés à partir du modèle *BigPicture*. Finalement, le code de la plate-forme ciblée sera généré à partir de ces modèles PSM. L'intérêt d'un tel procédé est de séparer, comme préconisé par l'approche MDA, la spécification fonctionnelle d'une application web des détails de son implémentation sur une plate-forme cible.

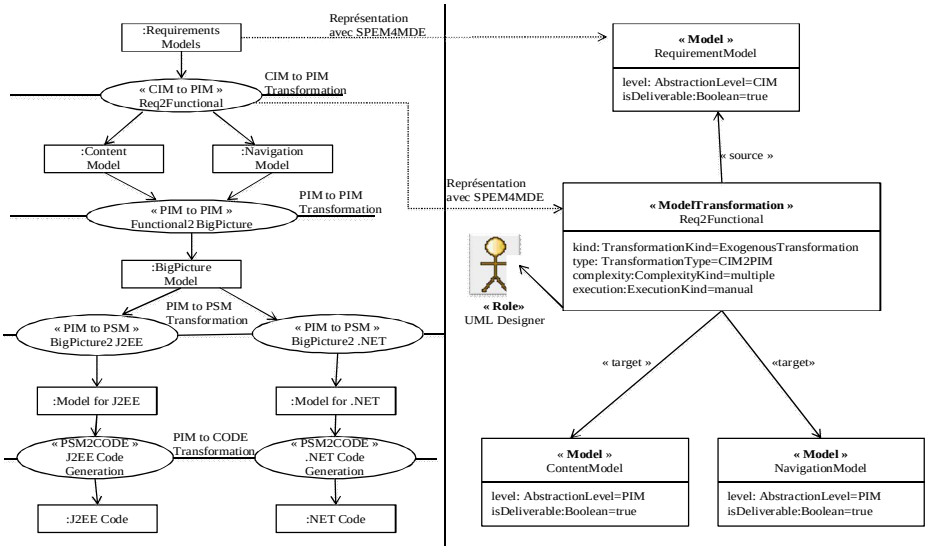

Figure 2. *Extrait du procédé UWE (à gauche) et représentation d'une partie de cet extrait avec les concepts de SPEM4MDE (à droite)*

Par manque d'espace, nous ne présentons que la première transformation du procédé UWE avec les concepts du langage SPEM4MDE (partie droite de la figure 2). La transformation Req2Functional est exogène, multiple, manuelle et de type CIM2PIM. Elle a pour source le modèle des exigences *RequirementModel* et pour cible les modèles *ContentMode*l et NavigationModel de l'application web. Le modèle des exigences est un livrable qui se situe au niveau CIM. Les modèles fonctionnels sont aussi des livrables mais ils se situent au niveau inférieur PIM. Le rôle *UML Designer* est le responsable de cette transformation. Ce rôle doit avoir des compétences en UML et en ingénierie web pour réaliser la transformation.

5. Conclusion

L'objectif de nos travaux est de proposer à la communauté du génie logiciel un nouveau langage de modélisation de procédés pour spécifier les procédés IDM et assister la mise en œuvre. Dans cet article, nous avons présenté une nouvelle approche pour formaliser les procédés IDM. Comme nous l'avons dit dans l'introduction, nous avons présenté seulement les concepts qui permettent de décrire statiquement un procédé IDM. Un extrait du procédé UWE nous a permis d'illustrer notre approche. Le développement logiciel dirigé par les modèles a pour principal objectif de concevoir des applications en séparant les préoccupations et en automatisant les tâches répétitives du développement logiciel par le biais des transformations. Ces transformations peuvent s'avérer parfois complexes. Un langage de modélisation de procédés de développement logiciel bien outillé qui permet de guider les développeurs dans l'élaboration et la génération des modèles devient plus qu'une nécessité. C'est dans cette perspective que nous travaillons actuellement à l'élaboration un environnement centré-procédés IDM qui implémente le métamodéle SPEM4MDE

6. Bibliographie

[1]Bendraou R., Gervais M., Blanc X., Jézéquel J. M. «Vers l'Exécutabilité des Modèles de Procédés Logiciels » *LMO*, Montréal, Quebec : Canada, 2008

[2]Bézivin J., « Sur les principes de base de l'ingénierie des modèles » *RTSI-L'Objet*, 10 / 2004 : Page 145-157, 2004

[3]Diaw S.., Lbath R., Coulette B. «SPEM4MDE: un métamodèle basé sur SPEM 2 pour la spécification des procédés MDE » Dans *MajecSTIC*, Avignon, France, 16 au 18 Novembre 2009

[4]Koch. Nora. "Transformations Techniques in the Model-Driven Development Process of UWE". In *6th ICWE*, California, USA, 2006

[5]Koch. N., Zhang G., Escalona M. J. "Model Transformations from Requirements to Web System Design". *6th ICWE*, California, USA, 2006.

[6]Maciel R., Magalhães B., Rosa N. "An approach to model-driven development process specification" In Proc., *ICEIS*, Milan, Italy, 2009.

[7]OMG, MDA, http://www.omg.org/mda/executive_overview.htm, 2006.

[8]OMG, UML 2.2 Infrastructure, http://www.omg.org/spec/UML/2.2

[9]OMG, SPEM 2, http://www.omg.org/spec/SPEM/2.0, Avril 2008

[10]Porres I., Valiente M. C. "Process Definition and Project Tracking in Model Driven Engineering" In: Proc of *7th PFOFES*, LNCS 4034; pp 127-141, Amsterdam, The Netherlands, June 12-14, 2006.