Maintien de la cohérence de modèles de conception hétérogènes

Mahmoud El Hamlaoui, Sophie Ebersold, Adil Anwar, Bernard Coulette, Mahmoud Nassar

Maintien de la cohérence de modèles de conception hétérogènes

Mahmoud El Hamlaoui[1,2], Sophie Ebersold[1], Bernard Coulette[1], Adil Anwar[3], Mahmoud Nassar[2]

1. Laboratoire IRIT, Équipe MACAO, Université Toulouse 2 – Jean Jaurès
5, allée Antonio Machado, 31058 Toulouse, France
{mahmoud.el-hamlaoui, sophie.ebersold, bernard.coulette}@irit.fr

2. Laboratoire SIME, Équipe IMS, Université Mohammed V de Rabat
BP 713 Agdal Rabat, Maroc
nassar@ensias.ma

3. Équipe Siweb, EMI, Université Mohammed V de Rabat BP 765
Agdal Rabat, Maroc
anwar@emi.ac.ma

RÉSUMÉ. *L'application de l'ingénierie dirigée par les modèles au développement de systèmes complexes introduit un ensemble de modèles hétérogènes, élaborés séparément, dans des langages variés. Afin de garantir la bonne exécution de l'ensemble et sa cohérence, il est nécessaire de procéder à la composition des modèles hétérogènes, puis à la mise en œuvre d'un ensemble de mécanismes permettant de l'exploiter. Cet article présente un processus semi-automatisé qui consiste d'une part à mettre en correspondance des modèles hétérogènes, et d'autre part, à maintenir la cohérence de l'ensemble par synchronisation des changements. Pour démontrer la faisabilité de l'approche, nous avons développé un outil support, en l'appliquant à l'exemple d'un système de gestions d'anomalies.*

ABSTRACT. *Complex systems development based on Model Driven Engineering introduces a set of heterogeneous models that are elaborated separately in different languages. To guarantee the soundness and the consistency of the whole, it is first necessary to compose heterogeneous models, and then to provide and run a set of mechanisms allowing to exploit it. This article presents a semi-automated process that consists in relating heterogeneous models by correspondences and then in a change processing, to insure the preservation of the coherence of the system. We illustrate this proposal by the example of a bug tracking system and run it on a tool that implements the approach.*

MOTS-CLÉS : *modèles hétérogènes, modèle de correspondances, gestion des changements.*

KEYWORDS: *heterogeneous models, correspondence model, change processing.*

1. Introduction

Face à la complexité grandissante des systèmes logiciels et à la nécessité de prendre en compte la multiplicité des concepteurs, nous avons proposé une première approche d'analyse/conception décentralisée par points de vue traduite par l'élaboration d'un profil UML, appelé VUML (Nassar, 2003) et par une démarche associée. Celle-ci passe par une étape de composition (de type fusion) des modèles partiels écrits en UML, afin de produire un modèle décrit dans le profil VUML. Pour automatiser (au moins partiellement) cette fusion des modèles partiels, nous avons appliqué l'approche IDM. Pour cela, nous avons traité la composition de diagrammes UML (plus précisément les diagrammes de classes (Anwar, 2010) et les diagrammes états-transitions (Ober, 2008)) comme des transformations exogènes prenant en entrée plusieurs modèles partiels conformes à UML et produisant en sortie un modèle conforme à VUML. Un prototype d'outil support a été réalisé en ATL, et a été validé sous l'environnement Eclipse.

La principale contrainte de cette approche fondée sur VUML est d'imposer un langage de modélisation unique, à savoir UML. Si cette contrainte peut être acceptable pour la conception de systèmes purement logiciels, il n'en est pas de même lorsque les systèmes sont plus complexes et intègrent des aspects matériels. Aussi avons-nous décidé de nous placer dans le cadre de modèles partiels décrits dans des langages adaptés aux différents domaines « métier » d'un système complexe. L'utilisation de langages dédiés correspond à la pratique industrielle et facilite la réalisation des modèles d'analyse/conception par les concepteurs. Cette problématique de multimodélisation est partagée par la communauté Génie Logiciel comme en témoigne notamment l'action spécifique Gemoc du GDR GPL, et le projet éponyme (Gemoc 2012).

Nous nous plaçons donc toujours dans le contexte d'une démarche guidée par les points de vue mais en considérant des modèles décrits dans divers DSML (*Domain Specific Modeling Language*), par des acteurs aux domaines de compétences variés. Ces modèles partiels (appelés aussi « source ») sont qualifiés d'« hétérogènes ». Cette approche pose bien entendu le problème de la gestion de la cohérence globale de l'ensemble des modèles partiels.

Une solution pourrait consister à transformer chaque modèle partiel en un modèle UML et à appliquer ensuite notre profil VUML. Cette approche, dont le principe est d'utiliser VUML comme langage pivot, a été écartée car jugée trop lourde à mettre en œuvre et donc ne passant pas à l'échelle En effet, il faut dans ce cas tout d'abord réaliser des transformations des modèles partiels hétérogènes vers des modèles UML, ce qui est loin d'être simple selon le degré d'hétérogénéité, mais il faut aussi réexécuter ces transformations à chaque évolution des modèles partiels.

Notre objectif est de proposer une approche permettant de connecter les modèles partiels − donc sans produire de modèle global − en constituant en quelque sorte un modèle « virtuel » sous forme d'un « réseau » de modèles partiels. Cette approche s'inscrit dans le cadre de la modélisation multi-DSL, mais il ne serait pas réaliste de traiter des modèles partiels appartenant à n'importe quel domaine, et donc

potentiellement très éloignés les uns des autres d'un point de vue sémantique. Aussi considérons-nous que le système complexe à modéliser appartient à un domaine d'application donné (ex : aéronautique, spatial, banque, automobile, logiciel embarqué, développement continu...) dont le vocabulaire est maîtrisable par un expert. Les concepteurs des modèles partiels exercent par contre des métiers différents et produisent des modèles hétérogènes dans leur domaine de compétences que nous appelons domaine « métier ».

Notre approche se présente sous la forme d'un processus composé de deux phases. La première consiste à mettre en correspondance les différents modèles partiels (source). La seconde phase traite la synchronisation des changements (et ainsi la gestion de l'évolution et le maintien de la cohérence) en s'appuyant sur les correspondances établies lors de la première phase. Les acteurs impliqués dans ce processus sont les concepteurs métier et l'expert du domaine d'application. Dans cet article, nous considérons pour simplifier que les concepteurs élaborent les modèles partiels et que l'expert du domaine réalise la mise en correspondance entre ceux-ci. Cependant, dans la réalité, le processus que nous décrivons est mis en œuvre de façon collaborative. La description de cet aspect collaboratif n'entre pas dans le périmètre de cet article et est abordé dans la discussion proposée en section 6. Une version préliminaire du processus est décrite dans (El Hamaloui *et al.,* 2013a).

Le processus proposé s'appuie sur un métamodèle de correspondances dont l'originalité est d'être générique et « spécialisable » selon les domaines. Il permet de créer des correspondances n-aires entre éléments de modèles, et de capturer les évolutions de modèles. Pour obtenir les relations de correspondance entre les modèles, nous identifions tout d'abord les correspondances entre métamodèles puis générons de façon semi-automatique les correspondances entre les modèles. Pour exprimer la sémantique des relations associées aux correspondances, nous avons défini un DSL dédié. L'identification des correspondances peut être fastidieuse. Pour aider l'expert du domaine, nous proposons une assistance en exploitant des techniques issues de l'ingénierie des connaissances. Ce travail en cours a été décrit partiellement dans (El Hamlaoui *et al.,* 2014).

L'identification des correspondances entre les modèles partiels est aussi, et ce n'est pas négligeable, un moyen de renforcer globalement la sémantique du réseau de modèles que nous cherchons à constituer, et permet de réduire les incohérences potentielles entre les modèles partiels. Nous avons développé comme preuve de concept un prototype opérationnel support au processus sous l'environnement Eclipse. Cet outil se présente comme un assistant de l'expert de domaine. Dans cet article, il est décrit essentiellement à travers son utilisation pour le cas d'étude choisi.

L'article est structuré de la manière suivante : dans la section 2, nous présentons une synthèse des travaux relatifs représentatifs et nous positionnons notre approche par rapport à l'existant. Notre proposition est illustrée à travers le cas d'étude d'un système de *bug tracking* que nous présentons dans la section 3. La phase de mise en correspondance est décrite dans la section 4. La phase de synchronisation des changements (et ainsi la gestion de l'évolution et le maintien de la cohérence) est

présentée dans la section 5. Le prototype HMS d'outil support à l'approche est présenté et illustré dans les sections 4 et 5. La section 6 est dédiée à une discussion sur les points pouvant poser question. Enfin, nous concluons cet article et présentons les principales perspectives à ce travail.

2. Travaux relatifs

Nous faisons ici une synthèse des principales approches de la littérature traitant de la problématique de multimodélisation de systèmes complexes.

D'un point de vue général, la coordination de modèles hétérogènes est au cœur des travaux fédérés par le projet international GEMOC[1]. Un bilan sur les défis de la coordination des langages de modélisation a été publié dans (Baudry *et al.,* 2011). Dans cet article, les auteurs exposent tout d'abord un bilan sur l'adoption de l'IDM et le constat de la multiplication des langages de modélisation dans les processus de développement industriels. Les auteurs présentent le concept de modèle de calcul. Ce modèle est utilisé pour formaliser la sémantique d'exécution et de synchronisation des entités. Le travail propose aussi une classification de l'hétérogénéité rencontrée en génie logiciel en trois niveaux : l'hétérogénéité des systèmes (comme l'orchestration des sous-systèmes), l'hétérogénéité dans les plates-formes d'exécution (c'est le cas par exemple des moteurs de simulation), et finalement l'hétérogénéité dans la modélisation, appelée aussi coordination de modèles spécifiques aux domaines. Les auteurs proposent ainsi quatre critères pour comparer les approches traitant l'hétérogénéité des modèles, à savoir : 1) la syntaxe des langages c'est-à-dire la définition des concepts d'un langage et ses relations, 2) les données permettant de spécifier la sémantique sous la forme d'attributs utilisés pour représenter l'état du système, 3) les règles d'évolution qui expliquent la manière dont les données sémantiques évoluent, 4) et finalement le modèle de calcul qui définit la façon dont les différentes règles d'évolution sont déclenchées.

En ce qui concerne la mise en correspondance de modèles, nous nous sommes plus particulièrement intéressés aux approches AMW, ECL, Kompose, Matchbox, EMFCompare et DSMDiff.

Dans AMW (Del Fabro *et al.,* 2005), les auteurs décrivent un langage qui permet d'utiliser les transformations M2M pour comparer des modèles. Cependant, AMW n'est utilisable que lorsque le modèle source et le modèle cible sont très similaires et les développeurs doivent ajouter des extensions du métamodèle, de manière à permettre la définition des relations, même pour les plus évidentes d'entre elles (ex : similarité). En outre, le langage ne fournit pas les concepts adaptés à la mise en correspondance de modèles. Pour optimiser la représentation d'un modèle composé, (Clasen *et al.,* 2011a) propose une technique de virtualisation de modèles. Cette technique peut être utile pour la mise en œuvre de la gestion des traces et le calcul des impacts en cas d'évolution des modèles source.

1. http://gemoc.org/ The GMOC initiative: On the Globalization of Modeling Languages.

ECL (Kolovos *et al.,* 2006b) est un langage de mise en correspondance difficile à utiliser car il nécessite des compétences élevées et de gros efforts d'écriture de règles (les correspondances sont produites en exécutant une séries de règles décrites en ECL). Il compare également les éléments de modèle un-à-un. En outre, pour exploiter ce résultat, le développeur doit ajouter des « sérialisateurs » afin de transformer les traces en un modèle de correspondances exploitable pour des opérations de type IDM.

L'approche Kompose (Drey *et al.,* 2009), propose un processus de mise en correspondance qui doit être paramétré en définissant au niveau métamodèle les signatures des opérateurs de mise en correspondance spécifiques. À notre connaissance, Kompose ne s'applique qu'à des modèles homogènes.

Dans l'approche MatchBox (Voigt *et al.,* 2010), les auteurs commencent par transformer les modèles d'entrée en un modèle arborescent : AMC (SAP Auto Mapping Core). Le processus se poursuit en appliquant un ensemble de stratégies de correspondance afin de produire le modèle de liens. L'inconvénient majeur de MatchBox est que c'est au développeur de définir les transformations vers le modèle AMC. Un autre inconvénient, soulevé par les auteurs, réside dans la perte d'informations lors des transformations.

EMFCompare permet de calculer les correspondances en se basant sur le principe de similarité (Brun et Pierantonio, 2008). Le moteur de correspondance se base sur des heuristiques et les éléments sont comparés suivant plusieurs métriques telles que la similarité de nom, de type de contenu et de relation. Dans le même contexte, DSMDiff (Lin *et al.,* 2007), extension du travail décrit dans (Xing et Stroulia, 2005), utilise une technique de calcul ayant l'avantage de supporter différents DSMLs spécifiés à l'aide d'un environnement générique de modélisation (GME : *Generic Modeling Environment*). L'inconvénient de ces deux approches est qu'elles n'utilisent que la similarité et n'exploitent pas d'autres relations.

Certaines approches proposent de réaliser une intégration ou une fusion des modèles hétérogènes. Ainsi, la fusion syntaxique et sémantique des DSML a été étudiée par (Wouters, 2013) ; l'approche propose de construire un unique langage de modélisation à partir des différents DSML. Plusieurs notations visuelles adaptées aux attentes des experts de chaque domaine métier sont produites à partir de la même syntaxe abstraite résultant de l'opération de fusion. Ce travail est fondé sur le langage xOWL, extension du langage OWL. L'intégration sémantique repose sur la transformation sous forme d'ontologies des concepts définis dans les différents DSML pour supporter plusieurs notations du même artefact (modèle). Ce travail suppose que les modèles n'existent pas initialement (ou bien sont ignorés) et se concentre sur l'intégration des différents DSML en un unique langage de modélisation utilisé par les différents concepteurs métier selon plusieurs notations graphiques. C'est une alternative intéressante aux travaux qui proposent la gestion des modèles hétérogènes.

L'approche MDI (*Multi Document Integration*) présentée dans (Konigs *et al.* 2006) généralise le formalisme TGG et propose une approche d'intégration de modèles hétérogènes et distribués. Le processus d'intégration est basé premièrement

sur un ensemble de règles déclaratives appelées règles d'intégration de données. Ces règles déclaratives sont utilisées pour spécifier les liens de correspondance entre les modèles concernés. Par la suite, à partir de ces règles, sont dérivées un ensemble de règles opérationnelles qui assurent la cohérence entre les modèles. Ces règles sont responsables de la propagation des changements d'un élément de modèle à un autre. Cependant, les contraintes posées sur les attributs définis par TGG doivent être relativement simples. Par ailleurs, cette approche ne traite pas la gestion des conflits.

MOMENT (Boronat *et al.*, 2007) est un framework fondé sur EMF pour la gestion de modèles. Ce framework fournit un ensemble de modules génériques et d'opérateurs de gestion de modèles tels que l'opérateur de fusion. L'approche se base sur trois concepts : la relation d'équivalence, la stratégie de résolution des conflits et le rafraîchissement de constructions. L'opérateur de fusion utilise la relation d'équivalence définie pour un métamodèle donné pour détecter des éléments dupliqués entre les deux modèles en entrée. Cette approche suppose que les modèles d'entrée sont cohérents et ne traitent pas les relations complexes comme la relation d'agrégation. Cela rend l'étape de « matching » facile à automatiser. L'approche traite par contre la propagation des changements en proposant une mise-jour sur place des modèles reliés.

Plusieurs approches actuelles traitent un ou plusieurs aspects de la problématique de l'évolution de modèles. Nous avons étudié celles qui paraissent les plus représentatives, à savoir COPE (Herrmannsdoerfer *et al.,* 2008), EMFMigrate (Di Ruscio *et al.,* 2011) et celle développée par Cicchetti *et al.* (2008). Après analyse, on relève plusieurs lacunes. En effet, ces approches ne définissent pas de pré-phase de classification, ce que nous jugeons indispensable pour pouvoir traiter automatiquement certains changements. En outre, la plupart de ces approches, excepté celle de Cichetti *et al.,* se concentrent sur la migration de modèles à l'issue de l'adaptation du métamodèle correspondant, ceci afin de préserver le lien de conformité entre le modèle et son métamodèle (principe de co-évolution). Ces approches ne traitent donc que le niveau vertical de l'évolution, sans prendre en considération l'évolution horizontale. Or, la synchronisation de modèles s'inscrit dans ce second type d'évolution.

D'une façon générale, les approches de mise en correspondance étudiées présentent des lacunes sur deux axes : avant et après la création du modèle de correspondance. En ce qui concerne le premier axe, nous notons d'abord le manque d'équilibre entre le pouvoir d'expression des concepts de mise en correspondance et leur réutilisabilité. Les approches existantes se basant généralement sur l'un des deux critères uniquement (par exemple la réutilisation se fait au détriment de l'expressivité et vice versa). Ensuite, ces approches n'exploitent que des correspondances binaires et ne permettent donc pas d'établir des correspondances complexes reliant un élément d'un modèle à un nombre quelconque d'éléments d'autres modèles. À propos du deuxième axe, on note que les approches de mise en correspondance produisent un modèle de correspondances pour chaque paire de modèles, ce qui a pour conséquence un grand nombre de modèles séparés, rendant ainsi leur gestion difficile et presque impossible à automatiser. En outre, les modèles évoluant dans le temps, la modification de l'un d'eux peut entraîner l'incohérence

du modèle de correspondances d'où la nécessité de répercuter les modifications, ou tout au moins d'identifier les éléments de modèles qui seront impactés par les changements. Au final, les approches étudiées ne traitent pas complètement l'aspect d'évolution du système (liens et éléments de modèles en cours d'évolution). Ceci vient du fait que ces dernières n'exploitent pas les correspondances préalablement établies. Il ne leur est alors pas possible d'offrir un mécanisme de synchronisation qui garantisse la cohérence du système global.

3. Cas d'étude : *Bug Tracking System* (BTS)

Pour illustrer la problématique de notre travail, nous avons choisi un cas d'étude issu de projets open source réels concernant le suivi d'anomalies : BTS (*Bug Tracking System*). Le choix de cet exemple s'explique par le fait qu'il est de taille raisonnable pour être décrit dans un article tout en faisant intervenir plusieurs concepteurs travaillant sur différents aspects du système, des exigences utilisateur jusqu'à l'architecture conceptuelle. Bien que purement logiciel, le système considéré permet d'illustrer la notion de mise en cohérence de modèles hétérogènes.

Un tel système a pour objectif d'offrir aux acteurs, en fonction de leurs différents statuts (chef d'équipe, développeurs, testeurs…), la possibilité de signaler des dysfonctionnements et de les commenter, de suivre l'état d'une anomalie, d'aviser leurs collaborateurs des problèmes rencontrés, de suggérer des solutions ou des possibilités de contournement, etc. Dans le domaine de gestion d'anomalies, nous considérons trois métiers : la gestion des exigences utilisateurs, la gestion des anomalies et la modélisation des processus. Pour chaque domaine métier, nous identifions un acteur qui réalisera un modèle décrit dans un langage dédié au métier (figure 1) :

– L'analyste : responsable de la modélisation des besoins sous forme d'exigences.

– L'architecte logiciel : responsable de la modélisation des anomalies.

– L'ingénieur des procédés : responsable de la modélisation du processus métier.

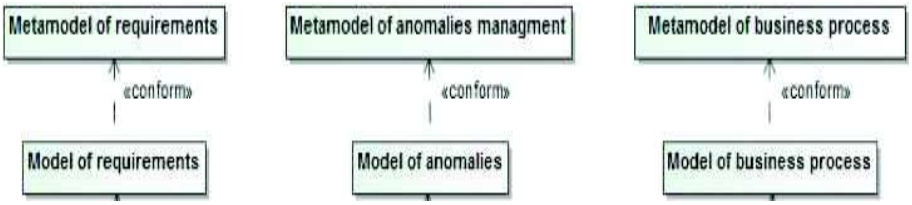

Figure 1. Modèles du système BTS

Dans les sections suivantes, nous présentons les trois modèles produits par les trois acteurs présentés, les métamodèles associés, et donnons quelques exemples de correspondances entre les modèles.

3.1. Modèle d'exigences

Pour évaluer la qualité et la validité de n'importe quel projet, il faut vérifier que ce dernier respecte les exigences exprimées par l'analyste. Un métamodèle d'expression d'exigences, inspiré de la notation SysML (SysML, 2008), a été proposé (figure 2). Il permet de construire un modèle exprimant les différentes exigences du système. Celles-ci spécifient, en utilisant une syntaxe textuelle, les fonctionnalités que le système doit remplir. Elles sont liées les unes aux autres et également à d'autres éléments de modèle à l'aide de différents types de relations (*copy, contain, derive*).

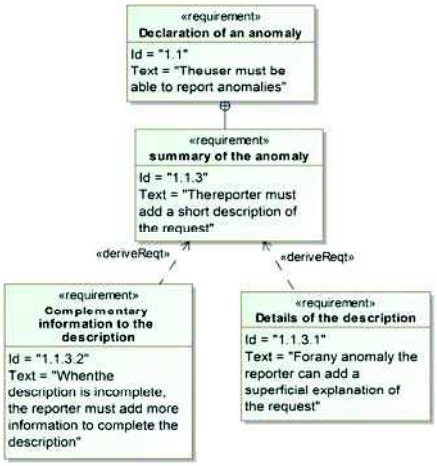

Figure 2. Modèle d'exigences (extrait)

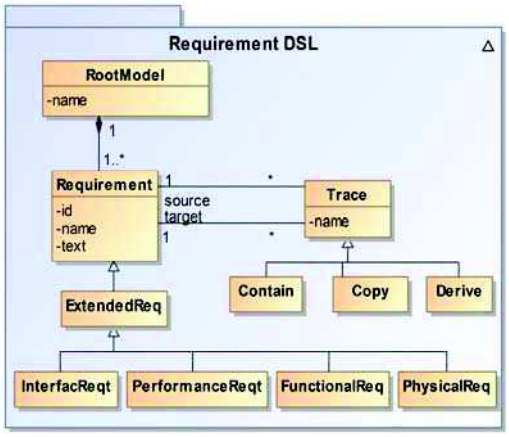

Figure 3. Métamodèle d'expression des exigences

Les exigences que le système BTS doit être capable de satisfaire sont ainsi décrites dans le modèle de la figure 3. Pour des raisons de simplicité, nous nous sommes limités à quelques exigences. Par exemple l'exigence avec id = '1.1' est liée à la déclaration d'une anomalie. Elle comprend une sous-exigence (id = '1.1.3') liée au résumé de l'anomalie, raffinée à son tour afin d'ajouter des contraintes supplémentaires à respecter lors de la déclaration d'une anomalie.

3.2. Modèle d'anomalies

Dans la figure 4, nous proposons un métamodèle définissant de façon générale et minimaliste des entités et des associations entre elles. Il est simple mais suffisant pour permettre d'élaborer un modèle d'anomalies qui est en fait un sous-ensemble du modèle de conception. Nous avons choisi une solution open-source dans le domaine de la gestion d'anomalies appelée Mantis (2010) pour représenter le modèle de conception logiciel.

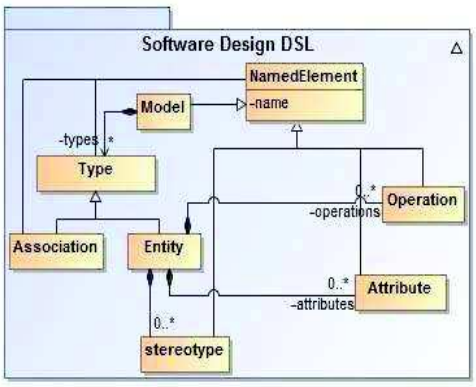

Figure 4. Métamodèle utilisé pour la conception

La figure 5 donne un aperçu de ce modèle. Le terme "Issue" est utilisé pour définir le concept d'anomalie. Une anomalie est caractérisée par un identifiant unique, un ensemble d'informations sur la version du produit et les étapes à reproduire pour aboutir au problème soulevé ("summary", "description", "stepsToReproduce", …). Deux type d'acteurs sont impliqués avec pour rôles respectifs "assignedTo" et "reporter". Le premier représente celui qui signale l'anomalie, tandis que le second indique l'acteur à qui le traitement de l'anomalie est affecté.

3.3. Modèle de processus

Le traitement d'une anomalie peut aussi être vu comme un processus « métier » que les différents acteurs doivent mettre en œuvre pour pouvoir la résoudre. Nous

supposons que le concepteur de procédé utilise la notation BPMN (BPMN, 2011) pour modéliser le processus métier. Cette notation étant standard, nous n'avons pas jugé utile d'en donner la description.

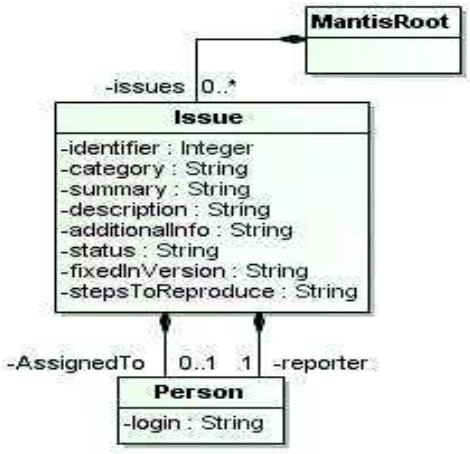

Figure 5. Extrait du modèle de conception d'anomalies

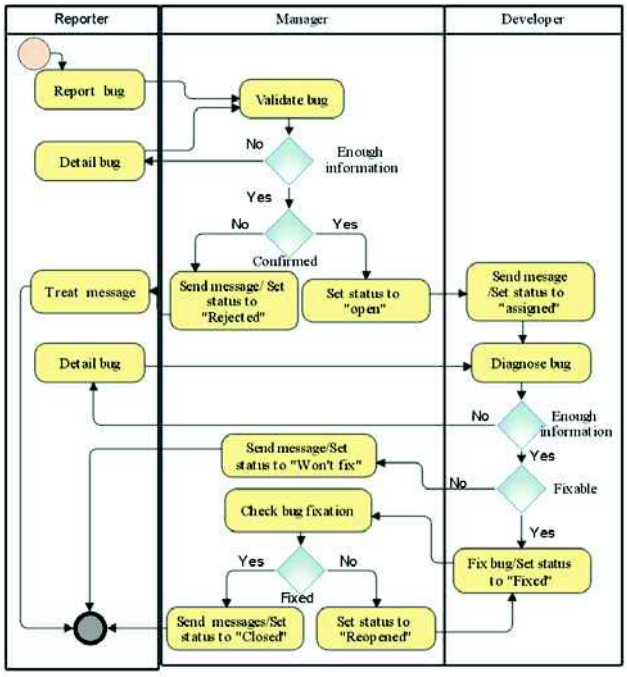

Figure 6. Extrait du modèle du processus métier du BTS en BPMN

Un exemple du processus exprimé en BPMN (2011) est présenté dans le modèle de la figure 6. Les rôles requis dans ce modèle sont *developper*, *reporter* et *manager*. Après avoir signalé une anomalie, le *reporter* doit mettre le statut de l'anomalie à « new», un mail est automatiquement envoyé au chef de projet qui a le rôle *manager*, puisqu'il n'intervient pas directement dans la correction de l'anomalie. Une fois que le chef de projet valide l'anomalie, il doit l'affecter à un *developper*, et changer le statut à "open". Si l'anomalie n'est pas validée par le chef de projet, il la réaffecte au *reporter*, qui l'a signalée, pour demander par exemple un complément de description. Une fois que le *developper* a corrigé l'anomalie, il doit en informer son chef de projet, qui à son tour vérifie la correction de l'anomalie et change son statut à "Fixed". Le *manager*, averti de ce changement, revérifie et change le statut de l'anomalie à "Closed", si elle a été corrigée.

Nous supposons que les trois modèles partiels créés pour la mise en place de l'application BTS ont été réalisés de façon indépendante. Néanmoins, ils ont naturellement des liens entre eux. Ces liens dits de « correspondance » peuvent être mis en évidence par un expert du domaine d'application.

3.4. Correspondances entre les modèles partiels du BTS

La figure 7 illustre quelques correspondances binaires ou ternaires entre les éléments des trois modèles partiels du BTS mises en évidence par l'expert du domaine. Elles sont fondées sur les relations suivantes: "Equality", "Similarity", "CoDependency", "Verify" et "UpdateValue". Par exemple l'élément *reporter* du modèle de conception est le même que l'élément *reporter* du modèle du processus métier. Ces deux éléments sont reliés par une correspondance possédant la relation *Equality*. De même, une correspondance ternaire possédant une relation *Similarity* a été établie entre les éléments des trois modèles, respectivement, *requirement, issue* et *detect bug*.

L'objectif que nous poursuivons dans notre approche est dans un premier temps de fournir un support à l'identification de ces correspondances afin de réaliser un modèle contenant les correspondances entre ces modèles partiels, selon un mode semi-automatique. Cette mise en correspondance des modèles est l'objet de la section suivante.

4. Mise en correspondance de modèles hétérogènes

L'objectif de cette section est de décrire la première phase du processus global de notre approche (voir figure 8), visant la construction d'un modèle permettant de connecter des modèles partiels fournis en entrée. Le modèle construit est appelé *modèle de correspondances*.

Pour identifier l'existence de correspondances entre les éléments des modèles source, nous nous fondons sur les correspondances existant entre les éléments de leurs métamodèles respectifs. Ceci permet de définir une correspondance une seule fois au niveau métamodèle et de la réutiliser pour les éléments des différents

modèles, eux-mêmes instances des métamodèles. Les correspondances établies au niveau des métamodèles serviront donc de guide lors de l'établissement des correspondances entre les modèles.

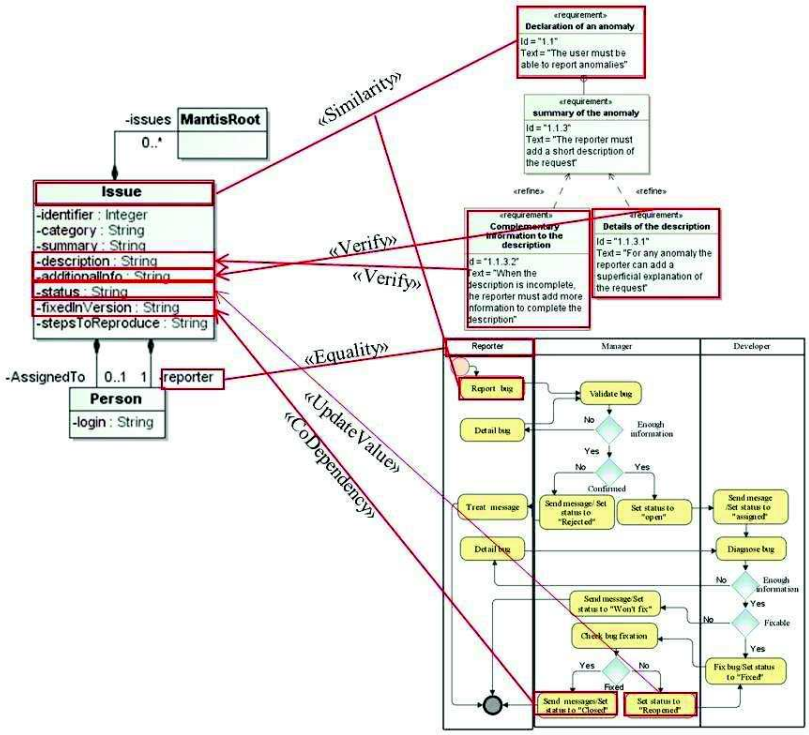

Figure 7. Exemple de correspondances entre les modèles partiels du BTS

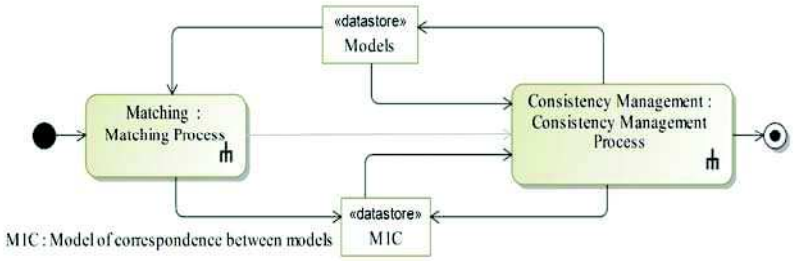

Figure 8. Processus global de l'approche

Pour exprimer les correspondances, nous avons défini un métamodèle de correspondances. Il faut noter que nous ne cherchons à identifier que les correspondances inter-modèles et non les correspondances intra-modèles qui existent entre les éléments d'un même modèle.

4.1. Métamodèle de correspondances

Il y a différentes relations possibles entre deux éléments de modèles. En effet, une relation peut être scalaire (ex. : =, >, ...), fonctionnelle (conversion, concaténation, …) ou encore conceptuelle (agrégation, héritage, composition, ...). Dans (Pottinger *et al.,* 2003), les auteurs définissent deux catégories sémantiques de relations : égalité et similarité. Plusieurs égalités sont référencées dans la littérature (égalité de noms, de synonymes et d'hyponymes, par exemple). De même on trouve plusieurs similarités : similarité de types, similarité de noms et similarité de relations. Pour représenter les différents types de relations pouvant exister entre des éléments de modèles, nous proposons un métamodèle générique, métamodèle de correspondances, appelé « MMC » (ElHamlaoui *et al.,* 2013b), permettant de prendre en compte la diversité des relations et des DSL (voir figure 9).

4.1.1. Concepts

Les concepts de ce métamodèle sont les suivants :

– *CorrespondenceModel* : il s'agit ici du concept clé de modèle de correspondances. Il contient les correspondances établies entre des éléments de modèles distincts.

– *Correspondence* : une correspondance est composée d'éléments de modèles (ses extrémités) et de la relation (*Relationship*) qui les lie. Il faut noter qu'une correspondance n'est pas forcément binaire. Elle peut contenir plus de deux éléments qui sont liés par la même relation.

– *Relationship* : ce concept (abstrait) désigne les relations entre (méta-)éléments de différents (méta-)modèles. Associé à *RefElement*, il permet de définir des correspondances n-aires, mettant en relation plusieurs éléments en même temps. Il est spécialisé par les classes *DomainIndependentRelationship* and *DomainSpecificRelationship*.

– *HLR* (*High Level Relationship*) : métaclasse abstraite dont la spécialisation permettra de définir les relations permettant de créer des correspondances au niveau des métamodèles.

– *LLR* (*Low Level Relationship*) : métaclasse abstraite dont la spécialisation permettra de définir les relations permettant de créer des correspondances au niveau des modèles.

– *DIR* (*DomainIndependentRelationship*) : cette métaclasse représente les relations génériques existant dans les différents domaines d'application. L'agrégation, la similarité, la dépendance et la généralisation sont des spécialisations de ce concept.

– *DSR* (*DomainSpecicRelationship*) : elle représente les relations intervenant dans un domaine d'application spécifique. De nouveaux types de relations sont obtenus par spécialisation de cette métaclasse.

4.1.2. Expressivité et multiplicité apportées par le métamodèle proposé

Le métamodèle que nous avons proposé permet d'étendre l'ensemble des concepts supportés par héritage de la classe *DomainSpecificRelationship (DSR)*. En effet, les relations préétablies dans ce métamodèle, les DIR − indépendantes du domaine − telles que agrégation, similarité, dépendance et généralisation (voir figure 9), sont génériques et peuvent être utiles quel que soit le domaine, mais peuvent être insuffisantes.

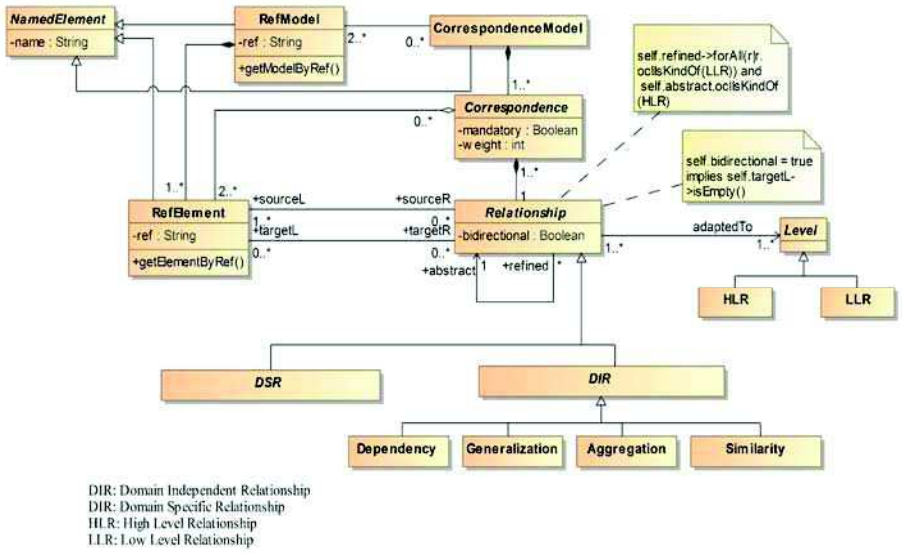

Figure 9. Métamodèle de correspondances MMC (partie générique)

Dans le cas du système BTS notamment, de nouvelles relations ont dû être introduites. Par exemple, une exigence du modèle d'exigences exprime des contraintes sur des attributs d'anomalies du modèle de conception. Il est donc judicieux de les connecter par une vérification. On met ainsi en évidence la nécessité de créer une nouvelle correspondance et par conséquent une nouvelle relation, que nous appelons *Verify*. Il est de même nécessaire de rajouter une relation (*UpdateValue)* entre un attribut d'anomalie et une tâche. La figure 10 illustre ces ajouts par spécialisation de la métaclasse *DomainSpecificRelationship*.

Un des intérêts de l'approche proposée est qu'elle permet également de supporter des correspondances n-aires entre plusieurs modèles. Les cardinalités associées à *RefModel* dans son association à *CorrespondenceModel* et dans l'agrégation de

Correspondence ainsi que les rôles de *Relationship* dans ses associations à *RefElement* – apparaissant dans la figure 10 – permettent notamment ces connections multiples.

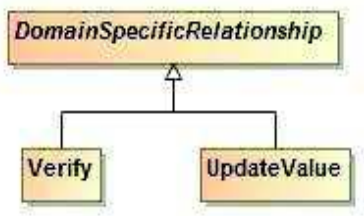

Figure 10. Prise en compte des spécificités d'un domaine

4.1.3. Expression de la sémantique des types de relations : le DSL SEL

Afin d'attribuer une sémantique aux différents types de relations, qui peuvent par ailleurs différer d'un domaine d'application à un autre (par exemple, le type de relation *Require* peut avoir une signification différente dans un domaine X, de celle utilisée pour le BTS), nous avons proposé d'associer à chaque type de relation, une sémantique formelle ou semi-formelle. Celle-ci permettra d'assister l'expert dans l'établissement des correspondances. Nous avons ainsi défini un DSL dédié à l'expression de cette sémantique (figure 11).

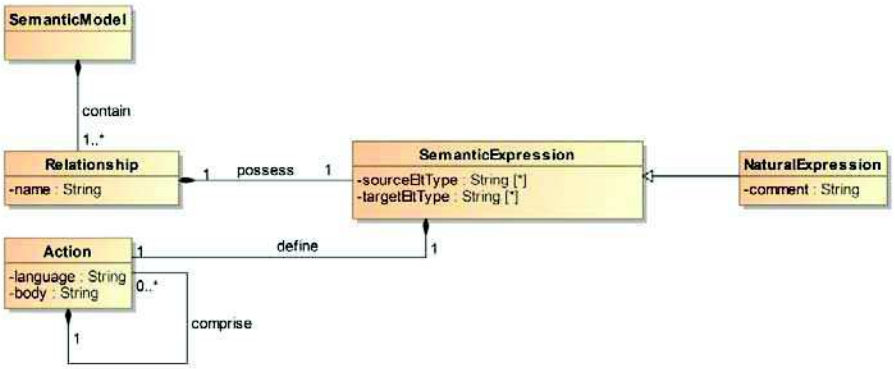

Figure 11. DSL d'expression de la sémantique des relations (SEL)

Pour la production du modèle d'expression sémantique associé au modèle de correspondance, nous proposons une assistance à l'expert. Ainsi, un programme parcourt le MMC pour récupérer automatiquement la liste des types de relations de base (DIR) et celles ajoutées par l'expert du domaine (DSR). Par la suite une expression est associée à chaque type de relation. Chaque expression se compose du

type d'élément (ou liste d'éléments) source, un type d'élément (ou liste d'éléments) cible (i.e. dans le cas du type de relation *Similarity* on n'aura que des éléments source qui peuvent avoir différents types). L'expression permet aussi de spécifier l'action à exécuter dans un langage qui devra être précisé (tel que : OCL, Java, HQL, …). Dans la mesure où l'expression avec un langage formel n'est pas possible, l'expert a la possibilité d'utiliser une expression structurée en langage naturel.

La figure 12 montre un exemple du modèle d'expression de la sémantique des relations du BTS

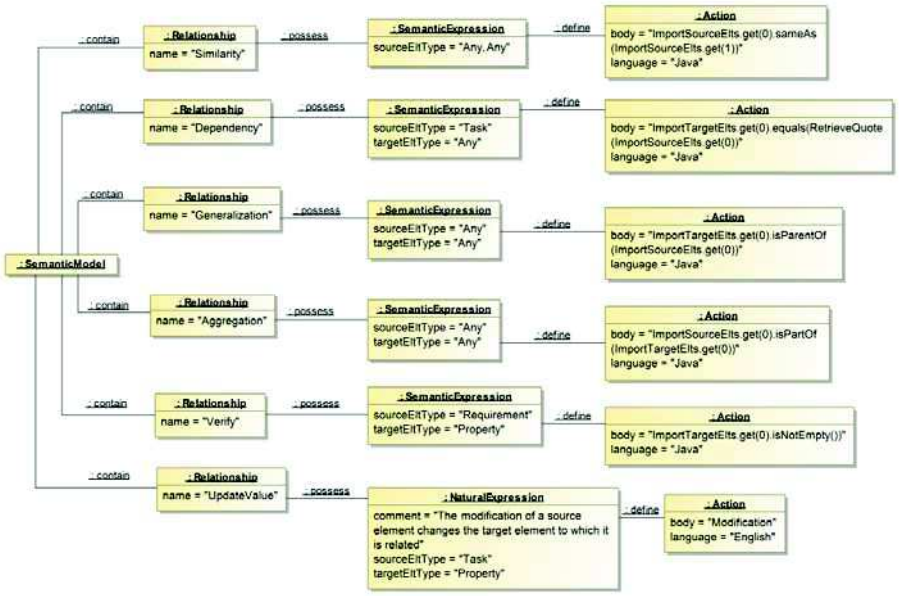

Figure 12. Modèle d'expression de sémantique

Un type de relation S*imilarity*, par exemple, est utilisé dans une correspondance impliquant deux éléments (*sourceElt[0], sourceElt[1])* de nature indifférentes (*Any* pour le premier élément et *Any* pour le second). Il est décrit par une expression en Langage Java. L'action de l'expression explicite, à l'aide de la fonction *sameAs*, que les deux éléments sont similaires. La construction de la fonction *sameAs* récupère l'information à partir d'une ontologie de domaine. Si cette dernière n'existe pas, la fonction fait appel à un thésaurus générique. Nous avons choisi d'utiliser WordNet (Patil, 2013) via son API JAWS. Si aucun résultat n'a été trouvé, la fonction *sameAs* utilise en dernier recours une implémentation de la méthode *levenshteinDistance* définie dans l'API d'alignement. A noter que l'utilisation de cette métrique est un choix car rien n'empêche d'en utiliser d'autres. Un panorama des métriques structurelles est présenté dans (Cheatham, 2013).

Concernant le type de DSR *Verify*, il est utilisé pour relier un élément de type *Requirement* à un autre de type *Property*. L'action de cette expression est décrite en Java. Le type de relation *UpdateValue*, difficilement formalisable, est quant à lui exprimé en langage naturel en précisant les types d'éléments concernés par ce type de relation, ceci afin que tous les acteurs du domaine en aient la même compréhension.

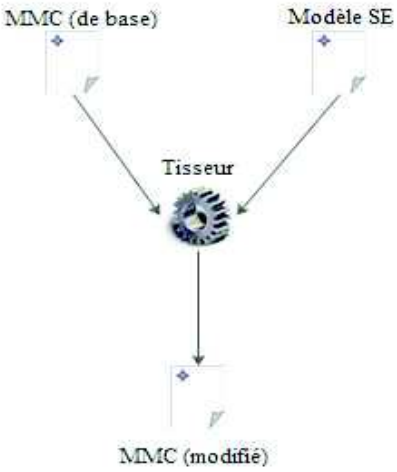

Figure 13. Principe du tissage des expressions sémantiques au MMC

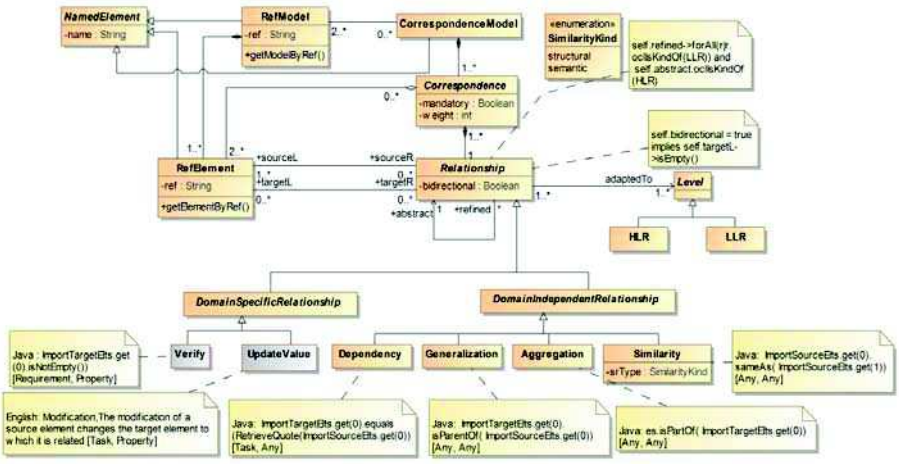

Figure 14. Métamodèle de correspondances annoté

Le modèle d'expression de la sémantique (Modèle *SE*) est ensuite tissé au MMC à la manière du développement par aspects (Filman, 2004) (figure 13). Dans notre

cas, il consiste simplement à greffer les différentes expressions sous la forme d'annotations sur les types de relations du MMC, comme le montre le MMC annoté résultant (figure 14).

L'intérêt de l'utilisation du DSL SEL est premièrement d'avoir une définition structurée commune de chaque type de relation, deuxièmement de construire le M2C de façon assistée à l'aide des informations sur les types d'éléments reliés et troisièmement d'exploiter les actions pour filtrer les correspondances lors du raffinage afin de ne garder que celles qui sont en concordance avec la sémantique des relations qu'elles contiennent. À cet effet, nous proposons dans la section suivante un mécanisme (de type raffinage) permettant d'obtenir les correspondances entre les modèles à partir des correspondances établies entre les métamodèles.

4.2. Raffinage des correspondances

Nous distinguons les correspondances entre méta-éléments ou HLC ("High Level Correspondance"), des correspondances de type LLC ("Low Level Correspondance") que l'on retrouve au niveau des modèles. Une LLC contient une relation de bas niveau (LLR), alors qu'une HLC contient une relation de haut niveau (HLR). Une HLC permet d'anticiper la complexité de la mise en correspondances en établissant d'abord des correspondances au niveau des métamodèles impliqués. Les LLC sont obtenues à partir des HLC par un mécanisme de raffinage semi-automatique.

Un raffinage de HLC en LLC est similaire à une transformation d'un PIM (*Platform Independent Model*) en PSM (*Platform Specific Model*). Il s'agit d'abord d'une projection filtrée des correspondances établies au niveau M2, sur le niveau M1. Il faut ensuite pouvoir compléter la description des LLC avec les précisions requises par les modèles partiels.

Dans la suite, nous noterons $\mathcal{R}_f$ la relation refinage. Considérant deux correspondances Cx et Cy, Cy $\mathcal{R}_f$ Cx, signifie que Cy est un raffinage de Cx.

4.2.1. Création des HLC

Les correspondances HLC sont définies une seule fois au niveau des métamodèles où il est plus facile d'établir les relations (il y a beaucoup moins d'éléments à relier au niveau métamodèle qu'au niveau modèle). L'identification des correspondances à ce niveau est établie de façon assistée en s'appuyant sur les annotations ajoutées au MMC via le DSL SEL. Plus précisément, grâce à l'information sur les types d'éléments (renseignés dans les annotations), il est possible d'appliquer un filtre sur l'ensemble des méta-éléments et de ne présenter à l'expert du domaine que les méta-éléments à relier.

Les correspondances HLC sont contenues dans un modèle appelé « M2C » (car il relie des métamodèles de niveau M2) conforme au métamodèle de correspondances MMC (figure 9).

4.2.2. Application à l'étude de cas BTS

Dans le système BTS, des éléments de métamodèles sont en correspondance. La sémantique des relations est établie par l'expert qui, par héritage, rajoute au métamodèle les concepts nécessaires, dans la partie spécifique au domaine (figure 10)

Ainsi, les relations génériques (exprimées par la partie indépendante du domaine) du métamodèle et les relations spécifiques permettent de connecter les métamodèles sources par les correspondances associées.

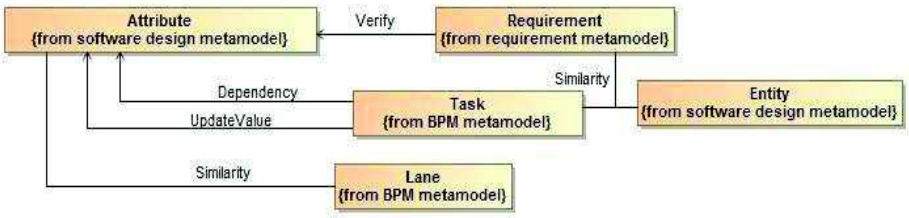

Figure 15. Exemples de HLC pour l'exemple du BTS

La figure 15 illustre des correspondances entre les métamodèles identifiés pour le système BTS. On remarque notamment la correspondance ternaire *Similarity* qui lie *Requirement* à *Entity* et *Task.* En effet, une exigence est similaire à une tâche en terme de processus, comme elle est similaire à la notion d'entité en terme de conception.

4.2.3. Création des LLC

Les HLC contenues dans le modèle de correspondances M2C sont raffinées pour créer des LLC entre chacun des éléments de modèle instances des méta-éléments reliés par les correspondances HLC. Les LLC sont stockées dans un modèle appelé « M1C » (car il relie des modèles de niveau M1), également conforme au MMC.

Le point de départ consiste à identifier les éléments à relier. Toutes les instances de méta-éléments reliés doivent aussi potentiellement l'être au niveau modèle. La création des correspondances par raffinage se déroulera différemment selon que l'on apporte (ou pas) des précisions, à la correspondance établie au niveau méta. Dans les deux cas, la première opération consistera à dupliquer les HLC. Si les correspondances doivent comporter des informations plus précises au regard des modèles partiels et du domaine d'application, elles seront ensuite étendues.

Raffinage par duplication

Ce type de raffinage induit un homomorphisme entre les correspondances du M2C et celles du M1C. Il s'agit d'une duplication de toutes les relations de correspondances définies dans le M2C. En d'autres termes, il y a autant de LLC

pour une HLC donnée que d'instances de tuples d'éléments pour les méta-éléments reliés par la HLC concernée.

Ceci se formalise, pour le cas d'une correspondance binaire, par :

Soient :

{Cn|M} l'ensemble des correspondances Cn d'un modèle M et me/M, un élément d'un modèle M,

Ci,[ma/MA, mb/MB], une correspondance entre les éléments ma du modèle MA et mb du modèle MB.

Si ∃Cx tel que :

Cx,[mme1/MM1, mme2/MM2], avec M1 χ MM1, M2 χ MM2,

Alors :

$\forall$ me1/M1 InstanceOf mme1/MM1, me2/M2 InstanceOf mme2/MM2,

$\exists$ {Cy,[me1/M1, me2/M2]} avec Cy $\mathcal{R}_f$ Cx

Bien entendu, étant donné le nombre potentiellement très grand de correspondances ainsi créées, on peut facilement imaginer que toutes ne seront pas utiles. Les correspondances LLC ne seront donc stockées dans le M1C que si elles sont pertinentes.

Les LLC sont filtrées grâce aux différentes expressions associées aux types de relations. Pour les types de relations ayant une expression semi-formelle, c'est à l'expert, assisté par un outil, de décider de garder ou de supprimer la correspondance en fonction de l'expression associée au type de la relation.

Raffinage par extension

Les LLC créées par duplication ne satisfont pas forcément le besoin de l'expert. Il peut avoir à opérer des choix sur certaines actions à effectuer comme indiqué dans (Barbier *et al.,* 2009) pour préserver les propriétés souhaitables, par exemple, ou ajouter des détails ou des informations sur les correspondances, de sorte à en préciser la sémantique.

Le raffinage de type « extension » permet de redéfinir les correspondances afin d'ajouter aux relations davantage de contraintes, et/ou des traitements spécifiques au domaine traité. Dans ce cas, l'extension prendra la forme d'une nouvelle relation, instance d'une nouvelle métaclasse, héritant de la métaclasse LLR.

Pour le BTS, par exemple, deux nouvelles relations, *Equality* et *CoDependency*, ont été ajoutées à l'ensemble des relations LLR (figure 16). L'introduction de la relation *Equality* permet, par exemple, de produire la correspondance attendue entre les éléments *reporter* des modèles partiels (cf. section 3.4).

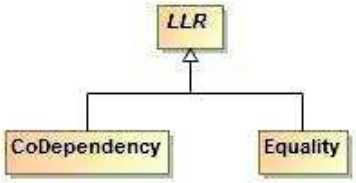

Figure 16. Exemple d'ajout de relations de bas niveau comme spécialisation de la métaclasse LLR

4.2.4 Application à l'étude de cas BTS

Dans l'étude de cas que nous menons, les HLC mises en évidence (figure 17), sont raffinées et permettent de produite les LLC présentées sur les deux figures ci-dessous. La figure 17 présente celles qui sont obtenues par duplication, la figure 18 illustre le raffinage par extension.

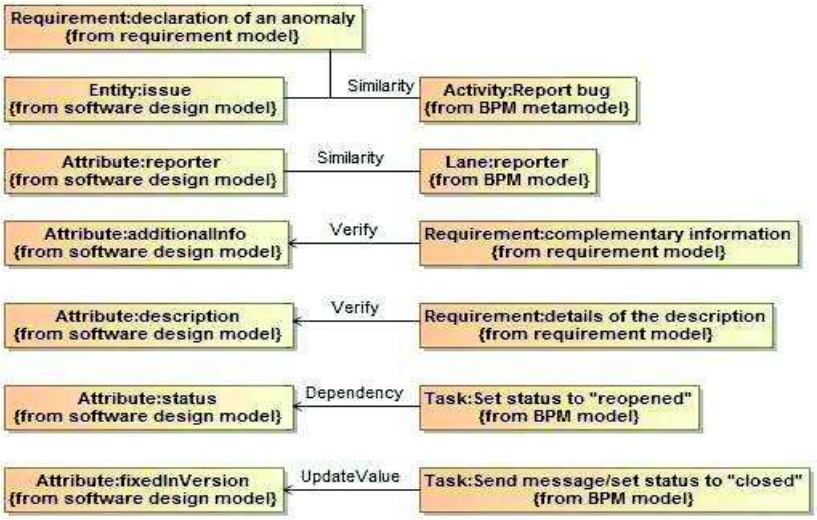

Figure 17. Exemples de LLC du BTS obtenues par duplication

Sur la figure 17, on remarque la similarité ternaire entre les élements *issue*, *declaration of an anomaly* et *Detect a Bug* qui a été identifiée sur la figure 7 précédemment (section 3.4). On voit également sur la figure 18 la correspondance *Equality* entre les éléments *reporter* issus des deux modèles partiels *Conception des anomalies* et *Processus métier*.

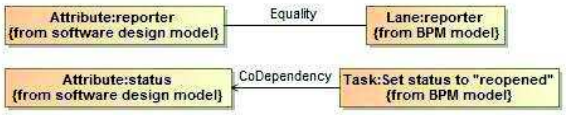

Figure 18.Exemples de LLC du BTS obtenues par extension

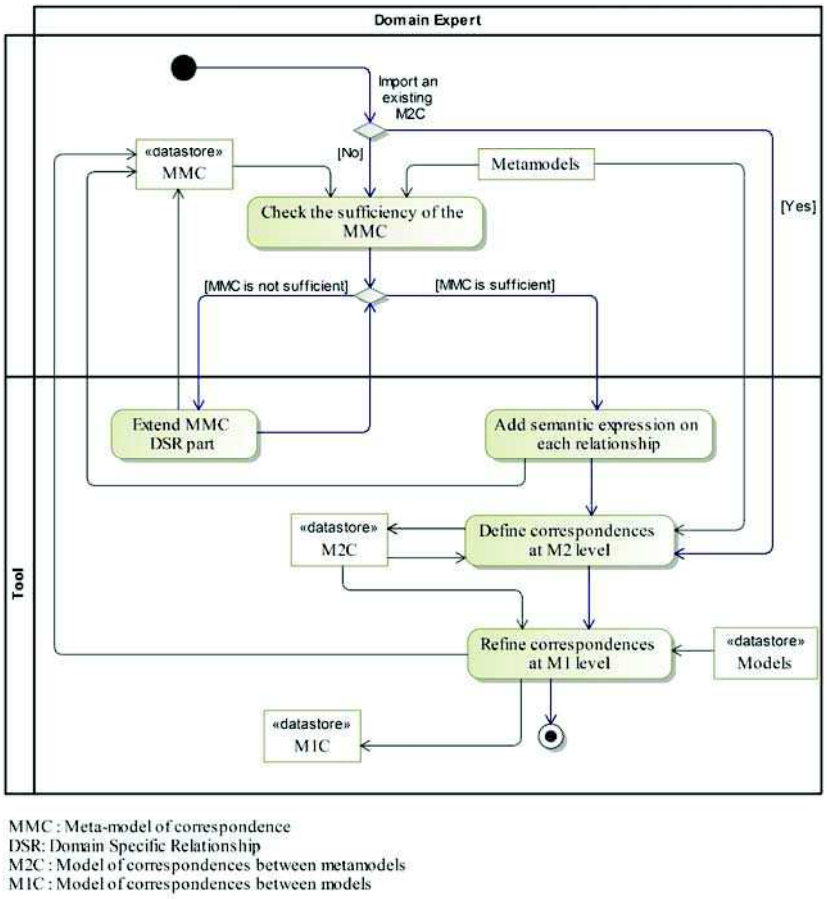

Figure 19. Processus de mise en correspondance

4.3. Processus de mise en correspondance

La proposition faite dans cette section a consisté dans un premier temps à identifier les concepts nécessaires à la mise en correspondance de modèles partiels ; Il s'agit de la réalisation d'un métamodèle de correspondances qui permette de

générer les modèles de correspondances entre modèles partiels, quel que soit le domaine d'application visé. En effet, ce métamodèle propose des éléments génériques spécialisant le concept *DomainIndependantRelationship* et une extension possible grâce au concept *DomainSpecificRelationship*. Une activité préalable à mener pour produire un modèle de correspondances consiste à vérifier l'expressivité du métamodèle fourni et à le compléter, si nécessaire, par des relations spécifiques au domaine d'activité.

4.3.1. Description

Nous proposons donc un processus de mise en correspondance (figure 19) commençant par cette activité d'extension éventuelle du métamodèle MMC, à partir des métamodèles source. Cette activité est menée par un expert du domaine qui a une vision globale du domaine et des correspondances possibles. L'étape suivante consiste à enrichir le MMC avec une expression de la sémantique de chaque type de relation sous forme d'annotations.

La deuxième partie du processus consiste à établir les correspondances. Cette activité, menée sous la supervision de l'expert, peut être en partie outillée. En effet, la déclaration des HLC peut être supportée par un outil. De même, la sélection des LLC conservées au niveau des modèles, peut être facilitée dans un mode semi-automatique

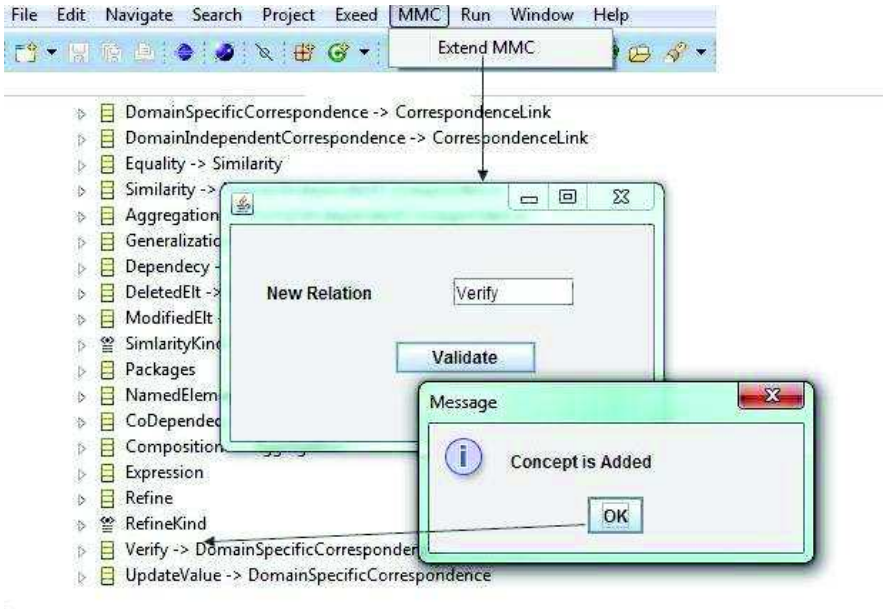

Figure 20. Extension du métamodèle sous l'outil HMS

Nous avons implanté l'approche dans un outil appelé Heterogeneous Matching Suite (HMS), qui supporte notamment le processus de mise en correspondance. Son utilisation permet la déclaration des correspondances entre modèles partiels et la définition des HLC au niveau M2, puis le raffinage en LLC au niveau M1. La figure 20 présente l'interface de l'outil pour l'extension du MMC. L'outil HMS a été développé sous Eclipse (Gronback., 2009).

4.3.2. Application à l'étude de cas BTS

La figure 21 illustre le modèle de correspondances du BTS obtenu en utilisant HMS. Les parties (1), (2), (3) représentent respectivement les références aux modèles du processus métier, des exigences, et des anomalies ainsi que les éléments de modèles qui sont utilisés dans les 6 correspondances représentées dans la partie (4) (cf. figure 7).

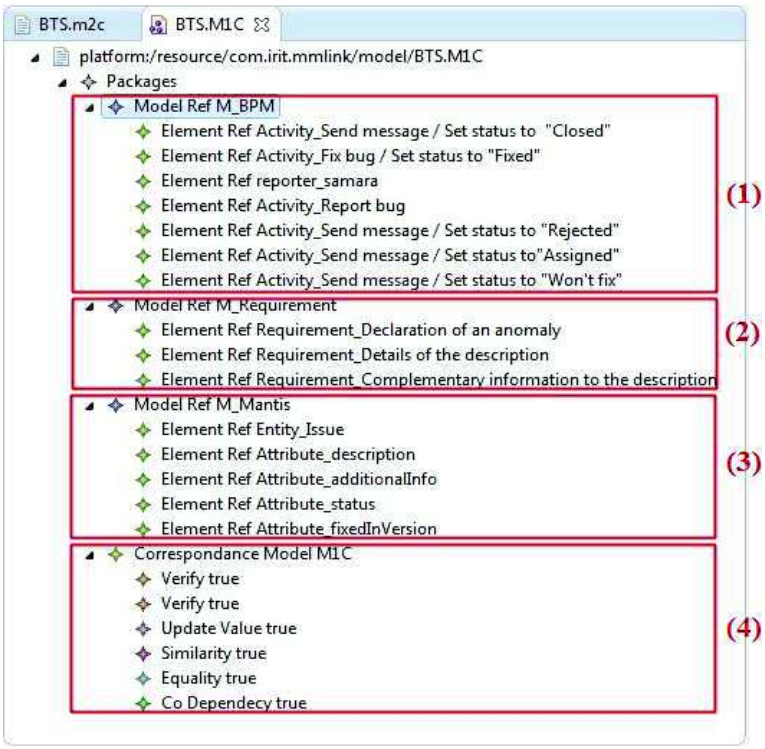

Figure 21. Extrait du modèle de correspondances M1C produit sous HMS

5. Maintien de la cohérence de l'ensemble des modèles

Outre l'intérêt du modèle de correspondances M1C souligné précédemment pour l'expression de la sémantique du réseau de modèles partiels, l'autre intérêt du M1C

est son exploitation pour la gestion de la cohérence dans le cas où les modèles partiels évoluent. En effet, les modèles partiels sont toujours évolutifs dans la réalité, et à l'image de leur élaboration qui a été faite séparément par les différents concepteurs, leur évolution se fait également de façon séparée et non coordonnée. Notre objectif est donc de maintenir la cohérence du réseau de modèles constitué selon la démarche présentée ci-dessus. Nous proposons pour cela un processus semi-automatique permettant de (i) mettre à jour le modèle de correspondance, (ii) calculer les impacts d'un changement sur un modèle donné, (iii) proposer des modifications pour maintenir la cohérence du système (cf. figure 22). Le processus que nous proposons et décrivons dans les sous-sections suivantes, comprend trois étapes majeures : l'identification des changements ("change detection"), la classification des changements ("change classification") et la synchronisation des changements ("change processing").

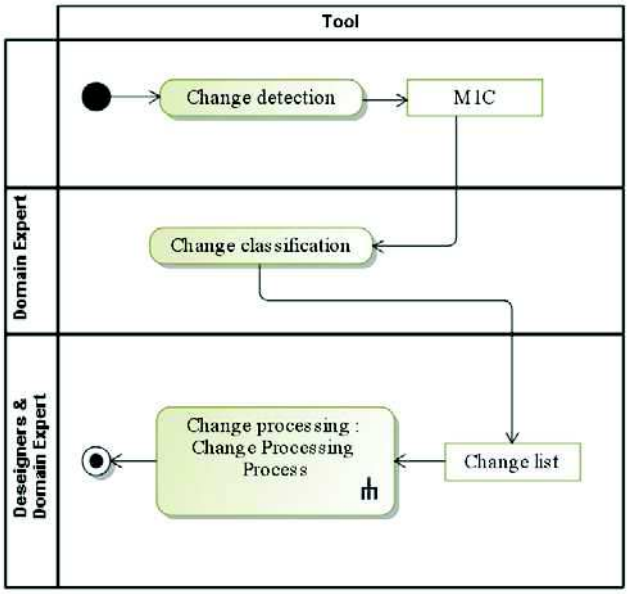

Figure 22. Processus de maintien de la cohérence lors des évolutions de modèle partiel

5.1. Identification des changements

5.1.1. Détection des changements

Une fois que le processus de mise en correspondance a produit le modèle de correspondances M1C, le processus de maintien de la cohérence est activé. Il prend en entrée l'ensemble des modèles partiels constituant le réseau de modèles du

système (les modèles susceptibles d'avoir subi une évolution), et le modèle de correspondances M1C.

La phase de détection des changements est automatiquement déclenchée. Elle a pour objectif d'identifier les éléments qui ont changé (*i.e.* modifiés, ajoutés, supprimés) par l'intermédiaire du patron de conception Observateur de Gamma. La figure 23 présente l'application de ce patron à notre approche sous la forme de son intégration dans le MMC. Son utilisation permet de notifier le modèle de correspondances des différents changements qui ont eu lieu via la méthode *notify*. La méthode *update* sera utilisée lors de la phase de synchronisation des changements pour maintenir la cohérence des modèles.

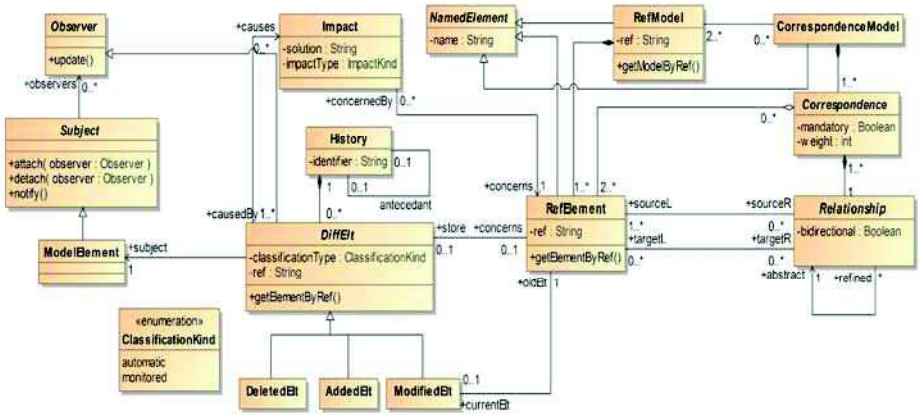

Figure 23. Intégration du patron de conception Observer dans le MMC

5.1.2. Enrichissement du métamodèle de correspondance

Maintenant que nous sommes capables de détecter les changements opérés sur les éléments de modèle, nous devons pouvoir les prendre en compte dans le modèle de correspondances. Le métamodèle MMC est étendu afin de supporter les concepts associés à ces évolutions (nous nous sommes inspirés des opérations CRUD) (Barbier *et al.,* 2010).

Les concepts introduits dans le MMC (figure 23), qui permettent de prendre en compte les différents types d'évolution des éléments de modèles, sont les suivants :

– *DiffElt* : métaclasse abstraite qui permet d'enregistrer, une fois spécialisée, la trace des éléments ayant évolué. Elle possède deux attributs. Le premier contient la classification du changement (cf. section 5.2) et le second est la référence de l'élément (construite à partir de son nom, de son méta-élément et du nom du modèle).

– *DeletedElt* : Elément de modèle qui n'existe plus, suite à une opération de suppression. (Quand un élément est supprimé d'un modèle source, nous le

maintenons dans le modèle de correspondances d'où cette métaclasse permettant de conserver la trace de la suppression).

– *AddedElt* : Nouvel élément de modèle qui a été ajouté.

– *ModifiedElt* : Nouvel état d'un élément de modèle, défini comme le résultat d'une opération de modification d'un élément du modèle initial.

Nous avons donc retenu trois types de changements à ce niveau là : ajout, suppression, ou modification d'éléments de modèles. Le mode de traitement d'un changement est fortement lié à son type. La deuxième phase du processus va consister à classer les changements, pour adapter leur traitement.

5.1.3. Exemple : application au BTS

L'expert a identifié une correspondance de type *UpdateValue* entre les éléments *fixedInVersion* de type *Attribute* du modèle des anomalies et *Set status to « closed »* de type *Task* du modèle des processus métier (figure 24) (voir modèles partiels sections 3.2 et 3.3).

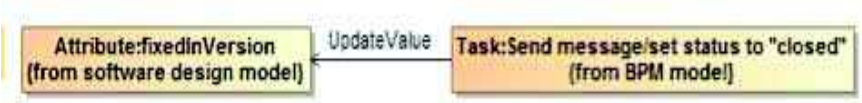

Figure 24. Extrait du modèle de correspondances du BTS

Cette correspondance exprime que l'on doit créer une nouvelle version, lorsque le statut d'une anomalie traitée passe à *closed*. Si par exemple, l'élément *fixedInVersion* est remplacé par un élément *targetMilestone* – ce qui signifie qu'on exploitera différemment le couple *(traitement des anomalies, génération de versions)* – un changement sera détecté.

5.2. Classification des changements

5.2.1. Modes de gestion des changements

Les modes de gestion des modifications proposés visent à gérer au mieux les impacts, en impliquant au minimum l'expert du domaine. Ils permettent d'attribuer à chaque type de changement une action spécifique.

Le M1C a été modifié et contient les changements qui ont eu lieu, en fonction de leur type. Nous allons d'abord produire une liste des changements, puis les classer selon leur mode de traitement : automatique ou guidé.

– Mode automatique ("Automatic Evolution") : il concerne les modifications qui mènent à des actions automatiques effectuées sur les modèles. Par exemple, si nous supprimons un élément de modèle, toute correspondance englobant cet élément devient orpheline. (Nous définissons une correspondance orpheline comme une

correspondance dont une des extrémités − un élément de modèle - est manquante). Quand une correspondance est orpheline, elle est supprimée du modèle de correspondance.

– Mode guidé (“Monitored Evolution”) : il concerne les évolutions qui nécessitent une assistance humaine afin de décider de l’impact (ou des impacts) du changement. Il relève de la responsabilité de l’expert du domaine de décider s’il convient de maintenir la relation, et dans ce cas si c’est avec les nouveaux éléments de modèle impliqués, ou si elle doit être modifiée.

5.2.2. Exemple : application au BTS

Le changement concernant l’élément *fixedInVersion* sera géré dans le mode guidé étant donné que la sémantique de l’élément a changé. Nous sommes ici dans le cas d’un changement de type modification.

5.3. Synchronisation des changements

C’est dans la phase finale, dite de synchronisation des changements, que les modèles vont évoluer en accord avec les changements qui ont eu lieu, par application de traitements spécifiques.

5.3.1. Processus proposé

Certaines modifications liées aux changements détectés nécessitent l’approbation de l’expert du domaine et des concepteurs, alors que d’autres pourront être gérés automatiquement. La figure 25, présente le processus suivi lors de cette étape de traitement de l’impact des changements.

Les modèles sont modifiés en adéquation avec les changements identifiés. Les évolutions classées dans la catégorie *Automatic Evolution* sont traitées automatiquement. Les éléments des modèles partiels source ne doivent pas être modifiés, sans accord de l’acteur concerné. Par conséquent, les seules évolutions automatiques possibles consistent à supprimer du M1C des correspondances ou à relancer une phase de mise en correspondance. Ces évolutions correspondent à l’ajout ou à la suppression d’éléments de modèles.

Les autres évolutions, classées dans la catégorie *Monitored Evolution*, sont gérées de façon semi-automatique en offrant des suggestions par rapport aux solutions possibles. La correspondance est maintenue si, après évolution de l’élément de modèle, elle est toujours valide. On contrôle notamment si l’évolution entraine un impact en cascade cyclique. Supposons qu’on ait modifié un élément A, ce qui a déclenché l’évolution d’un élément B ; ceci peut aussi entrainer l’évolution d’un troisième élément C lié à B. On appelle cette succession d’évolution des impacts en cascade. C’est dans le cas où l’élément C est relié à l’élément A qui a provoqué au départ ces évolutions, qu’on parle d’impact en cascade cyclique.

Quand un élément A est modifié, il faut généralement modifier chacun des éléments auxquels il est relié en regard des sémantiques associées aux

correspondances établies. L'ordre des modifications dépendra d'une pondération associée à chaque relation par l'expert, en fonction de sa prédominance dans un cycle d'évolution, par exemple.

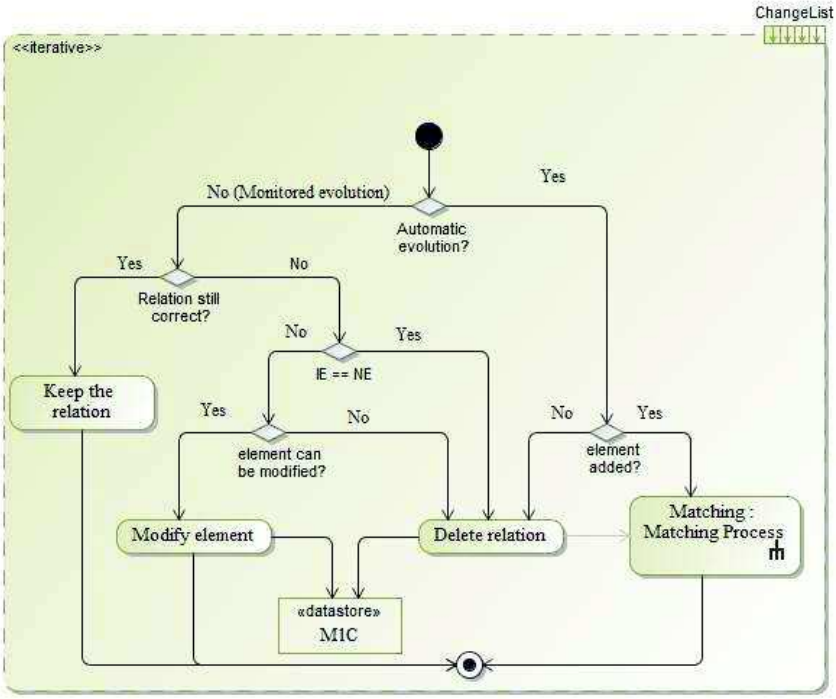

Figure 25. Processus de traitement des évolutions

Dans le cas d'un impact en cascade cyclique, la modification de l'élément de modèle en question n'est pas autorisée. En effet, partant du principe que le M1C est cohérent au départ, on considère que si A a été modifié, il a provoqué la modification de B, mais aussi celle de C. Logiquement, la modification de C provoquée par B, ne devrait pas avoir d'effet supplémentaire et il n'y a pas de raison que C provoque la modification de A.

5.3.2. Exemple : application au BTS

L'élément *fixedInVersion* du modèle partiel initial *conception d'anomalies* a été remplacé par un élément *targetMilestone*. Cette modification a été capturée, classée en mode guidé et répercutée pour mettre à jour le modèle de correspondances. Ainsi (figure 26), l'élément *targetMilestone* apparaît à la place de *fixedInVersion*, la correspondance entre *fixedInVersion* et la tâche *Send messsage/set status to « closed »* a été supprimée. Une nouvelle mise en correspondance a permis de générer la correspondance entre *targetMilestone* et la tâche *set status to open*.

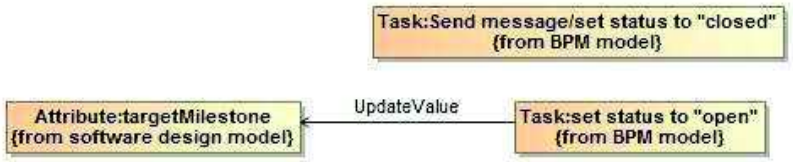

Figure 26. Synchronisation des changements appliquée au système BTS

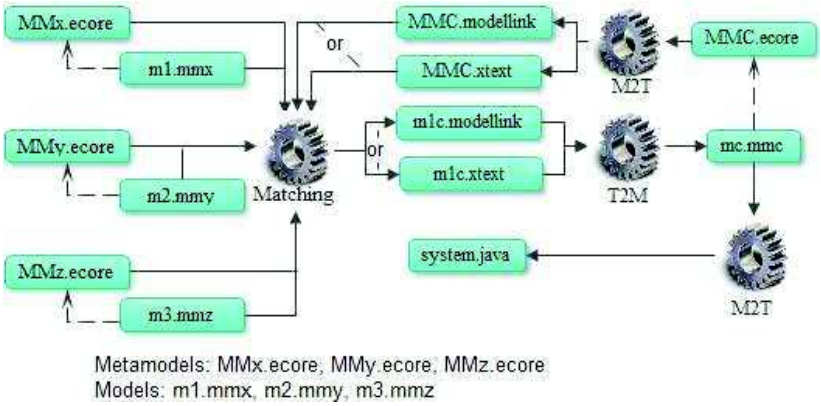

Figure 27. Architecture du prototype Heterogenous Matching Suite (HMS)

5.3.3. Implantation dans l'outil HMS

Cette phase de synchronisation des modèles est actuellement en cours de développement sous l'outil HMS. Notre objectif est dans un premier temps d'implanter la synchronisation en mode automatique et dans un deuxième temps de fournir l'aide nécessaire à la gestion guidée des impacts. La mise en place de coefficients de pondération associés aux correspondances devrait permettre de déterminer l'ordre de traitement des changements.

Actuellement, HMS se présente comme une suite composée de trois modules (représentés par des engrenages dans l'architecture générale de l'outil illustrée sur la figure 27) : Matching, M2T (Model To Text) and T2M (Text To Model). Les rectangles représentent les différents (meta-)modèles en entrée et sortie de chacun des modules. Deux vues synchronisées sont disponibles : graphique et textuelle. La première, intuitive, s'appuie sur l'outil Modelink (cf. Modellink). Pour que les correspondances puissent être établies graphiquement, le métamodèle de correspondances a été mis en conformité avec la syntaxe ModelLink. Le module M2T implémente une transformation qui « sérialise » (en s'appuyant sur la technologie JET (Java Emitter Template) (Steinberg *et al.,* 2009) le métamodèle de correspondances en deux modèles, selon le type de visualisation choisi.

C'est le module de *Matching* qui permet d'établir les correspondances entre modèles une fois que ces derniers sont saisis. L'acteur impliqué (en général l'expert du domaine), établit les correspondances au niveau des métamodèles. Il choisit ensuite le type de raffinage qu'il veut mettre en œuvre pour la production du M1C. (L'implantation du raffinage se fait par une transformation écrite en Java/EMF).

Le module T2M "parse" le modèle de correspondances construit, encore une fois selon le mode de visualisation choisi, en un modèle conforme au métamodèle MMC.

6. Discussion

La thématique de la synchronisation de modèles partiels d'un système complexe est vaste et ne peut pas être traitée dans un seul article. Par rapport à la proposition décrite dans les sections précédentes, il reste certains points à discuter et bien sûr à creuser.

Sémantique des relations de correspondance

La sémantique des liens entre les modèles est un point important de toute approche de multimodélisation. Dans notre approche, elle permet bien entendu de donner du sens aux correspondances, mais aussi de servir de filtre lors de la génération des correspondances de niveau modèle. Le résultat global est le renforcement de la sémantique du réseau de modèles produit par le processus proposé. Nous donnons la possibilité à l'expert d'associer une expression sémantique à chaque relation. Pour lui permettre de procéder de manière incrémentale, ce qui peut être utile pour la conception d'un système complexe, le DSL SEL offre la possibilité de spécifier une expression formelle ou semi-formelle en langage structuré. Cette manière de procéder permet de satisfaire la pratique des concepteurs mais limite l'application des techniques de vérification formelle pour assurer, par exemple, la cohérence des expressions sémantiques.

Automatisation de la mise en correspondance des modèles et passage à l'échelle

Pour que notre approche passe à l'échelle, il faut bien sûr que le processus présenté ne soit pas exécuté de façon manuelle. Notre approche supporte l'assistance à l'expert du domaine par l'intermédiaire de plusieurs mécanismes – exploitation de la sémantique des relations de correspondance, utilisation de techniques issues de l'alignement d'ontologies – mais il reste encore du travail à faire sur ces aspects et à valider l'approche sur des exemples de systèmes complexes de dimension industrielle.

Ordre de traitement des évolutions de modèles partiels

L'ordre de traitement des évolutions de modèles est important car sans coordination, le traitement des évolutions pourrait conduire à des cycles ingérables. Nous pensons qu'il n'est pas possible de faire l'économie de ce coordinateur – a priori l'expert du domaine – ou bien alors il faut mettre des contraintes comme par exemple figer certaines versions de modèles partiels, pour ne pas permettre n'importe quelle évolution à tout moment. Nous avons aussi pensé à mettre en place

des coefficients de pondération associés aux correspondances pour permettre de déterminer un ordre de traitement des changements sans blocage. Cette question est néanmoins ouverte.

Prise en compte de l'aspect collaboratif

Dans la réalité d'un système complexe, le processus que nous avons décrit ci-avant est éminemment collaboratif. Ainsi les différents acteurs de la modélisation doivent collaborer pour faire face à cette complexité. Cette collaboration peut s'opérer selon divers scénarios qui dépendent de plusieurs paramètres : degré de complexité du système, organisation du projet, disponibilité des experts et des concepteurs… Par exemple les concepteurs d'un métier donné peuvent collaborer entre eux sous la coordination d'un concepteur en chef pour produire un modèle partiel. Quand les modèles partiels sont réalisés, l'expert du domaine opère, avec l'assistance d'un outil support, à la mise en correspondance et au traitement de la synchronisation, en collaborant lui aussi avec les concepteurs responsables. Comme indiqué dans la section suivante, ce travail sur l'aspect collaboratif, que nous avons commencé à aborder, fait partie des perspectives principales de notre travail.

7. Conclusion et perspectives

Le travail présenté dans cet article s'inscrit dans le cadre de la modélisation de systèmes complexes. Il propose un processus couvrant la mise en correspondance de modèles de conception multi-DSL et la synchronisation de ces modèles lorsqu'ils évoluent.

Le processus comprend d'abord une phase de mise en correspondance des modèles hétérogènes produisant un modèle de correspondances. Cette phase s'appuie sur un métamodèle de correspondances et sur la définition d'une relation de Raffinage entre deux niveaux de modélisation permettant de déduire les correspondances entre modèles à partir des correspondances établies entre des concepts de niveau métamodèle. Le résultat est un modèle de correspondances conforme au métamodèle de correspondances. La seconde phase du processus permet, en exploitant le modèle de correspondances établi lors de la première phase, de traiter les changements identifiés sur les modèles partiels de façon à maintenir la cohérence des modèles interconnectés. Cette phase de synchronisation s'appuie sur une classification des types de liens de correspondance identifiés, en leur associant, quand c'est possible, une sémantique opérationnelle qui permette de gérer automatiquement les impacts des modifications. Le processus est supporté par un prototype opérationnel d'outil développé sous Eclipse.

L'originalité de notre proposition s'exprime plus particulièrement dans la mise en correspondance des modèles. Plusieurs critères de qualité la caractérisent :

– Généricité : le métamodèle de correspondances MMC fournit une partie générique – commune à tous les domaines – qui permet de décrire la plupart des relations courantes.

– Extensibilité : le MMC peut être étendu en fonction des particularités du domaine d'application considéré, afin de gérer les relations propres à des domaines spécifiques. Cela se fait par spécialisations de la métaclasse *DomainSpecificRelationship*.

– Flexibilité : le MMC permet de relier n modèles (à travers leurs éléments de modèle) et d'exprimer des cardinalités de type n-aire pour chaque correspondance possible.

– Légèreté : le modèle de correspondances est construit de manière virtuelle dans la mesure où il ne contient que des références aux éléments « physiques » des modèles partiels.

Pour tester l'approche et obtenir ce que l'on peut qualifier de preuve de concept, nous avons développé une étude de cas représentative des problèmes que nous cherchons à résoudre.

Perspectives

Comme le montre la section précédente concernant les points pouvant porter à discussion, il y a plusieurs tâches à réaliser pour compléter le travail actuel.

Pour améliorer le passage à l'échelle et automatiser (au moins partiellement) le processus, nous allons poursuivre notre investigation des techniques d'ingénierie des connaissances (déjà partiellement utilisées comme nous l'avons vu dans la section 4.1.3), pour améliorer l'expression des correspondances. Ce travail s'effectue en collaboration avec des spécialistes de l'alignement d'ontologies.

Par ailleurs, dans l'approche présentée ici, nous sommes partis du principe que les tâches non automatisables du processus étaient effectuées par un expert ayant une connaissance globale des différents modèles partiels, et donc capable d'identifier avec précision les liens de correspondance. Cette hypothèse rend le processus très dépendant de cet expert et donc relativement centralisé. Dans la réalité des développements industriels, ce travail est réalisé par plusieurs concepteurs qui travaillent de façon collaborative (comme nous l'avons mentionné dans la discussion). Nous avons donc initié une étude visant à remplacer le processus actuel par un processus collaboratif. Ce passage se traduira par une redéfinition des tâches à effectuer par les différents concepteurs, et par l'utilisation d'outils collaboratifs. Nous utilisons pour cela les résultats que nous avons obtenus dans le cadre du projet ANR GALAXY (Kedji *et al.,* 2014).

D'un point de vue outillage, nous sommes en train d'enrichir le framework HMS (*Heterogeneous Matching Suite*), notamment le support à la synchronisation des modèles lors de leur évolution.

Remerciements

Ce travail a été financé en partie par le PHC Volubilis ADAPROC (2011-2013).

Bibliographie

Anwar A., Ebersold S., Coulette B., Nassar M., Kriouile A. (2010). A rule-driven approach for composing viewpoint-oriented models. *Journal of Object Technology*, vol. 9, p. 89-114.

Barbier F., Cariou E., Belloir N. (2009). Contrats de transformation pour la validation de raffinement de modèles, *Actes Ingénierie Dirigée par les Modèles 2009*, Nancy, France.

Barbier F., Eveillard S., Youbi K., Cariou E. (2010). *Model-driven reverse engineering of cobol-based applications*. Information Systems Transformation: Architecture Driven Modernization Case Studies, p. 283-299.

Baudry B., Combemale B., DeAntoni J., Mallet F. (2011). *Ingénierie du logiciel pour les systèmes hétérogènes*. Rapport d'activités, Action Spécifique GeMoC du GDR GPL, 2011.

Boronat A., Cars J. A. , Ramos I., Letelier P. (2007). Formal model merging applied to class diagram integration, *Electronic Notes in Theoretical Computer Science*, vol. 166, p. 5-26.

BPMN (2011). *Omg bpmn-v2.0*, http://www.omg.org/spec/BPMN/2.0/PDF.

Brun C., Pierantonio A. (2008). Model differences in the eclipse modeling framework. UPGRADE, *The European Journal for the Informatics Professional*, 29–34.

Cheatham M., Hitzler P. (2013). The role of string similarity metrics in ontology alignment, *International Semantic Web Conference 2013*, Sydney, Australia.

Cicchetti A., Di Ruscio D., Eramo R., Pierantonio A. (2008). Automating co-evolution in model-driven engineering. *Enterprise Distributed Object Computing Conference 2008*, Munich, Germany.

Clasen C., Jouault F., Cabot J. (2011). Virtual Composition of EMF Models. *Actes Ingénierie Dirigée par les Modèles 2011*, Lille, France.

Clasen C., Jouault F., Cabot J. (2011). Virtualemf: a model virtualization tool. *International Conference on Conceptual Modeling 2011*, Brussels, Belgium.

Del Fabro M.D., Bezivin J., Jouault F., Breton E., Gueltas G. (2005). AMW: a generic model weaver. *Actes Ingénierie Dirigée par les Modèles 2005*, Paris, France.

Di Ruscio D., Iovino L., Pierantonio A. (2001). What is needed for managing co-evolution in mde?. *International Workshop on Model Comparison in Practice 2011*, Zürich, Switzerland.

Drey Z., Faucher C., Fleurey F., Mahé V., Vojtisek D. (2009). *Kermeta language*. Reference Manual.

El Hamlaoui M., Ebersold S., Anwar A., Nassar M. Coulette B. (2013). A Process for Maintaining Heterogeneous Models Consistency through Change Synchronization. *International Conference on Computer Systems and Applications 2013*, Fes, Morocco.

El Hamlaoui M., Ebersold S., Coulette B., Anwar A., Nassar M. A process for defining a unique correspondence model to relate heterogeneous models. *International Conference on Evaluation of novel Approaches to software engineering 2013*, Angers, France.

El Hamlaoui M, Trojahn C., Ebersold S., Coulette B. Towards an Ontology-based Approa for Heterogeneous Model Matching. *International Workshop on The Globalization Modeling Languages 2014*, Valencia, Spain.

Gemoc (2012). *The GEMOC initiative On the Globalization of Modeling Languag* http://www.gemoc.org

Gronback R. (2009). *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolk* Addison-Wesley Professional, 1 edition.

Herrmannsdoerfer M., Benz S., Juergens E. (2008). Cope: A language for the coupl evolution of metamodels and models. *International Workshop on Model Co-Evoluti and Consistency Management, 2008*, Budapest, Hungary.

Filman R., Elrad T., Clarke S. (2004). *Aspect-oriented Software Development*. In Addiso Wesley Professional.

Kedji K.A., Lbath R., Coulette B., Nassar M., Baresse L., Racaru F. (2012). Supporti collaborative development using process models: a Tooled Integration-focused Approac *Journal of Software: Evolution and Process*, vol. 26, p. 890-909.

Kolovos D., Paige R., Polack F. (2006). Model comparison: a foundation for moc composition and model transformation testing. *International workshop on Glob integrated model management 2006*, Shangai, China.

Kônigs A., Schrr A. (2006). MDI: A rule-based multi-document and tool integrati approach. *SOftware and SYstems Modeling 2006*, vol. 5, p. 349-368.

Lin Y., Gray J., Jouault F. (2007). DSMDiff: a differentiation tool for domain-speci models. *European Journal of Information Systems*, vol. 16, p. 349–361.

Mantis bug tracker (2010). Mantis bug tracker , http://www.mantisbt.org/index.php.

Modelink (2012). Epsilon Modelink, http://www.eclipse.org/epsilon/doc/modelink.

Nassar.N (2003). Vuml: a viewpoint oriented uml extension. *International Conference Automated Software Engineering 2003*. Montreal, Canada.

Ober I., Coulette B., Lakhrissi Y. (2008). Behavioral Modelling and Composition of Obje Slices Using Event Observation. *International Conference on Model Driven Engineeri Languages and Systems 2008*, Toulouse, France.

Patil, L.H. (2013). A novel feature selection based on information gain using wordn *Science And Information conference 2013*, London, England.

Pottinger R.A., Bernstein P.A (2003). Merging models based on given correspondenc *International Conference on Very large Data Bases 2003*, Toronto, Canada.

Schöttle M., Kienzle J. (2013). On the Challenges of Composing Multi-View Mode *International Workshop on The Globalization of Modeling Languages 2013*, Mian United States.

Steinberg D., Budinsky F., Paternostro M., Merks E. (2009). *EMF: Eclipse Modeli Framework*. Addison-Wesley.

SysML (2008). Omg sysml-v1.1. http://www.sysml.org/docs/specs/OMGSysML-v1.1-08-1 01.pdf.

Voigt k., Ivanov P., Rummler A. (2010). Matchbox: combined meta-model matching for semi-automatic mapping generation. *Symposium on Applied Computing 2010,* New Yok, United States.

Wouters L. (2013). *Multi-domain Expert-User Modeling Infrastructure*. Thèse, Université Pierre et Marie Curie.

Xing z. et Stroulia E. (2005). UMLDiff: an algorithm for object-oriented design differencing. *International Conference on Automated software engineering 2005*. New Yok, United States.