Modélisation des systèmes complexes et Points de vue : l'Ingénierie des Modèles centrée utilisateur pour l'Ingénierie Système

Sophie Ebersold

Université de Toulouse

IRIT

Document de Synthèse

Présenté en vue de l'obtention d'une

HABILITATION Á DIRIGER LES RECHERCHES DE L'UNIVERSITÉ DE TOULOUSE

Présentée et soutenue le *12 Mars 2021* **par :**

Sophie Marcaillou-Ebersold

Modélisation des systèmes complexes et Points de vue : l'Ingénierie des Modèles centrée utilisateur pour l'Ingénierie Système

JURY

Mireille Blay Fornarino	Professeure d'Université	Rapporteur
Olivier Barais	Professeur d'Université	Rapporteur
Xavier Blanc	Professeur d'Université	Rapporteur
Jean-Michel Bruel	Professeur d'Université	Garant
Benoit Combemale	Professeur d'Université	Examinateur
Bernard Coulette	Professeur d'Université	Examinateur
Bertrand Meyer	Provost and Professor of Software Engineering	Examinateur
Mahmoud Nassar	Professeur de l'Enseignement Supérieur	Examinateur

École doctorale et spécialité :
Informatique EDMITT
Unité de Recherche :
IRIT
Garant :
Jean-Michel Bruel
Rapporteurs :
Mireille Blay Fornarino, *Olivier Barais* et *Xavier Blanc*

Modélisation des systèmes complexes et Points de vue : l'Ingénierie des Modèles centrée utilisateur pour l'Ingénierie Système

Sophie Ebersold

A Christiane

Remerciements

La rédaction d'une Habilitation à Diriger les Recherches est pour moi le moyen de faire un point sur mes activités de recherche de ces quinze dernières années et d'en produire la synthèse. Ce projet a mis longtemps à voir le jour et je ne me serais pas lancée dans sa rédaction sans les encouragements de femmes de ma communauté de recherche. Je suis particulièrement reconnaissante à Mireille Blay, mais également à Thérèse, Régine, Mounia, Hanh Nhi, Christelle, Corinne, Laurence, Florence et à toutes celles qui m'ont poussée à soutenir une HDR. Je remercie également Franck Morvan et Jean-Michel Bruel qui m'ont rassurée sur la légitimité de cette démarche.

Je tiens à remercier tous ceux qui m'ont proposé leur aide et m'ont encouragée dans ce travail, en particulier Benoit Combemale pour ses conseils avisés et son soutien constant, et bien entendu mon garant, Jean-Michel Bruel.

Ce travail est le fruit de collaborations actives avec Mahmoud Nassar, Adil Anwar, Mahmoud El Hamlaoui, Saloua Bennani, Florian Galinier que j'ai co-encadrés lors de leurs thèses. Je les remercie tous pour ces années de discussions riches et productives.

J'ai un merci particulier à adresser à Bernard Coulette, puisque d'encadrant de ma thèse, il est devenu Directeur des thèses que j'ai co-encadrées avec lui. Je le remercie de m'avoir fait confiance et de m'avoir donné l'opportunité de collaborer avec ces chercheurs marocains que j'apprécie beaucoup.

Collaborer avec Jean-Michel Bruel et Benoit Combemale a également été pour moi source de beaucoup de plaisir. Les réflexions que nous avons pu avoir au lancement de l'équipe SM@RT ou encore lors des séminaires IRIT/UT2J, et les séminaires d'équipe à Villebrumier ou à Rennes, restent de grands moments de recherche et d'amitié.

Je suis très heureuse d'avoir eu l'occasion ces dernières années de travailler avec Bertrand Meyer. Membre de mon jury de thèse, Bertrand est à présent membre de mon jury d'HDR et je l'en remercie. Travailler à ses côtés m'apprend énormément ; je retiendrai notamment sa précision et sa rigueur scientifique, son goût du partage, ses leçons d'écriture.

J'aimerais aussi remercier Annie Meyer pour son hospitalité.

Merci également à toute l'équipe d'Inopolis, Marya, Sasha, Manuel pour les échanges riches et fructueux que nous avons eus, aux côtés de Bertrand et Jean-Michel.

Merci enfin à mes collègues du DMI et de l'IRIT qui m'ont permis de bénéficier d'un CRCT et de me lancer dans la rédaction de ce mémoire. Je n'oublie pas l'équipe DiverSE de l'IRISA que je remercie de m'avoir si bien accueillie lors de ce CRCT.

Pour finir par un élément important, je voudrais dire que mon jury d'HDR est pour moi à l'image de ces années de recherche, et des personnalités marquantes de ma carrière. Je tiens à remercier mes rapporteurs Mireille Blay, Xavier Blanc et Olivier Barais, qui m'ont fait l'honneur et l'amitié de bien vouloir juger ce travail. J'adresse un grand merci à tous les membres de ce jury, à la fois pour leur participation et pour le rôle que chacun a pu jouer, à un moment ou à un autre, dans cette production scientifique.

Résumé

A l'heure actuelle les systèmes complexes de par leur caractère critique, leur nombre, leur taille, ou encore les tâches distribuées et réparties qu'ils supportent, restent un sujet d'études très ouvert. Beaucoup de progrès ont été faits dans le domaine de la recherche dans le but de rendre les systèmes complexes fiables, que ce soit dans un cadre de formalisation des exigences, de multi-modélisation et d'inter-opérabilité, de définition de langages dédiés et d'environnements d'ingénierie adaptés, d'automatisation de la génération de code, ou encore de vérification et de validation des modèles produits. Néanmoins, les méthodes, techniques, outils apportés, s'ils donnent satisfaction, nécessitent encore des améliorations. En effet l'industrie est toujours en demande de moyens permettant de gagner en efficacité tant dans la production des éléments logiciels que dans l'automatisation des vérifications. Les méthodes traditionnelles perdurent et les habitudes évoluent lentement. Les nouvelles technologies du Génie Logiciel, et en particulier de l'Ingénierie Dirigée par les Modèles, ont du mal à s'implanter dans le domaine des systèmes complexes.

Mes travaux de recherche visent à fournir des éléments conceptuels permettant de mieux maîtriser le développement de ces systèmes. Ils s'appuient sur les modèles qui sont produits lors du développement de systèmes, les replacent dans un cadre d'Ingénierie Dirigée par les Modèles, en centrant la démarche sur les parties prenantes de ces systèmes.

A travers la synthèse de quatre thèses qui se sont enchaînées, chacune répondant à des points laissés en suspens par la précédente, je montre dans ce manuscrit comment les mécanismes que nous avons proposés s'inscrivent dans une amélioration de l'Ingénierie Système, en s'appuyant sur l'Ingénierie Dirigée par les Modèles pour réaliser l'alignement de modèles hétérogènes, en mettant l'utilisateur au centre des préoccupations et en proposant des techniques collaboratives et un langage semi-formel d'expression des exigences.

Table des matières

Première partie

Modélisation des systèmes complexes et Points de vue

Chapitre 1

Introduction

A l'heure actuelle les systèmes complexes conservent leur importance en termes de nombre, de taille, de criticité, de tâches distribuées et réparties, mais aussi, et c'est lié, d'études garantissant leur fiabilité. Beaucoup de progrès ont été faits dans le domaine de la recherche dans le but de rendre les systèmes complexes fiables, que ce soit dans le cadre de formalisation des exigences [44], de multi-modélisation et d'inter-opérabilité [198], de définition de langages dédiés et d'environnements d'ingénierie adaptés [84], d'automatisation de la génération de code [129], de vérification et validation des modèles produits [93]. Il n'en reste pas moins que les méthodes, techniques, outils apportés, s'ils donnent satisfaction, nécessitent encore des améliorations. En effet l'industrie est toujours en demande de moyens permettant de gagner en efficacité tant dans la production des éléments logiciels que dans l'automatisation (fiable, cela va sans dire) des vérifications. Les méthodes traditionnelles perdurent et les habitudes évoluent lentement. Les nouvelles technologies du Génie Logiciel, et en particulier de l'Ingénierie Dirigée par les Modèles [49], ont du mal à s'implanter dans le domaine des systèmes complexes.

Mes travaux de recherche visent essentiellement à fournir des éléments conceptuels permettant de mieux maîtriser le développement de ces systèmes. Nous proposons de nous appuyer sur les modèles qui de fait sont produits lors du développement de systèmes, de les replacer dans un cadre d'Ingénierie Dirigée par les Modèles, en centrant la démarche proposée sur les utilisateurs de ces systèmes. Ici nous ne réduisons pas l'utilisateur au rôle d'utilisateur final, mais à toute partie prenante du système, de son analyse à sa mise en œuvre.

C'est une synthèse de ces travaux, menés sur les douze dernières années, que je vais présenter dans ce mémoire. Ils ont permis d'enrichir le domaine de l'Ingénierie Système par des techniques et des outils centrés utilisateurs, mettant en œuvre l'ingénierie des modèles. Les résultats obtenus sont fondés sur la co-supervision de 5 thèses de doctorat et de mémoires de master recherche. La plupart des travaux que je vais présenter ont été menés dans le cadre de co-tutelles de thèses, et plus précisément dans le contexte de projets Franco-Marocains. Une synthèse temporelle de ces travaux est présentée sur la Figure 1.1 suivante.

1.1 Contexte général : Modélisation des systèmes complexes

Mes travaux se sont toujours situés dans l'ingénierie système [142], et plus particulièrement dans le domaine de la modélisation des systèmes complexes. L'objectif que nous avons poursuivi a été de fournir des outils et des méthodes prenant en compte les spécificités des différentes parties prenantes du système global et des artefacts utilisés et produits par ces mêmes parties prenantes.

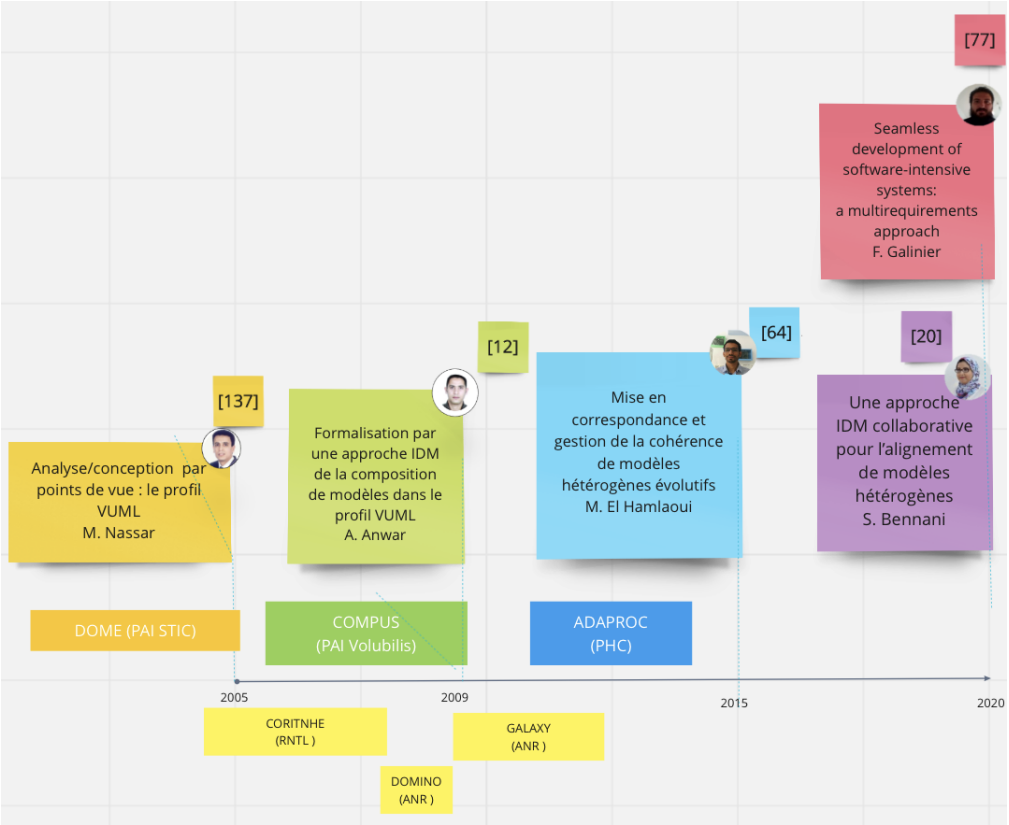


FIGURE 1.1 – Echelonnement de mes principaux travaux de recherche : de VUML à CAHM et RSML ; Thèses et Projets

1.1.1 Systèmes complexes, Modélisation, Points de vue

La notion de système complexe s'intègre dans l'Ingénierie Système telle qu'elle a été définie par différents organismes de certification. En effet, l'ANSI [12], l'IEE [63], l'INCOSE [115, 116] ou encore l'ISO/IEC [3], s'accordent sur la définition suivante d'un système :

Définition 1. Système : collection d'éléments hétérogènes indépendants, intégrés pour atteindre un ou plusieurs buts communs.

Nous adoptons cette définition pour caractériser un système complexe comme suit.

Définition 2. Système Complexe : système composé de sous-systèmes de natures différentes (logiciel embarqué, automate, composants ou équipements mécaniques, capteurs...), réalisés par des équipes hétérogènes (développeurs, automaticiens, mécaniciens, ...), selon des processus variés et soumis à des règles ou normes de sécurité, sûreté, etc.

La démarche de modélisation pour l'ingénierie système, Model-Based Systems Engineering ou MBSE [151, 172], s'adresse à la conception de systèmes complexes. En effet, modéliser le système permet d'en appréhender la complexité et le MBSE promeut l'utilisation de modèles en tant qu'éléments essentiels de l'ingénierie système. La réalisation d'un système complexe passe alors par la production de modèles de ce système.

Le rôle des modèles restant pour nous celui de JL Lemoigne [136], nous utilisons les définitions suivantes :

Définition 3. Modèle : Artefact obtenu par modélisation.

Définition 4. Modélisation : «Action d'élaboration et de construction intentionnelle, par composition de symboles, de modèles susceptibles de rendre intelligible un phénomène perçu complexe et d'amplifier le raisonnement de l'acteur projetant une intervention délibérée au sein du phénomène : raisonnement visant notamment à anticiper les conséquences de ces projets d'action possibles» [136]

La réalisation d'un système, aussi complexe soit-il, fait indubitablement intervenir plusieurs systèmes, plusieurs métiers, plusieurs technologies. L'étude des besoins de la modélisation des systèmes complexes, en particulier des systèmes spatiaux, nous a amenés à considérer la multitude de modèles qui co-existent lors de la réalisation et de la mise en œuvre d'un système. Ces modèles donnent chacun une vision partielle du système et constituent autant de points de vue sur la conception du système qu'il y a de métiers, de technologies, de méthodes et de processus impliqués. Le terme de points de vue a une longue histoire dans le domaine de l'ingénierie logicielle et des systèmes, et en particulier dans la modélisation par points de vue de Finkelstein et al. [82]. Selon l'ISO/IEC/IEEE 42010, un point de vue formalise l'idée qu'il existe différentes façons de voir le même système. Dans ISO/IEC/IEEE 42010, les points de vue font partie intégrante des descriptions d'architecture et peuvent également être spécifiés séparément. Cette norme reconnaît qu'un système complexe puisse être réalisé par élaboration de modèles indépendants, attachés à des préoccupations distinctes.

Ce qui nous intéresse particulièrement dans cette notion de point de vue est qu'elle permet de prendre en compte les acteurs du système dès les premières étapes d'analyse et de conception. De ce fait, elle permet aux différentes parties prenantes de se focaliser sur leur domaine d'expertise, tout en étant intégrés à un tout cohérent.

Nous considérons la définition suivante :

Définition 5. Point de vue : Un point de vue sur un système est une abstraction de la représentation du système répondant aux préoccupations d'un type donné de parties prenantes, indépendamment des préoccupations des autres parties prenantes de ce même système.

L'idée que nous poursuivons consiste à s'appuyer sur les modèles partiels produits (les modèles par points de vue), pour réaliser le système en les considérant comme des abstractions, des outils de communication, des outils de vérification, des vecteurs de mise en œuvre [20].

Nous ne cherchons pas à produire un modèle global du système mais à prendre en considération les modèles partiels qui sont produits lors de la réalisation du système dans le cadre d'une modélisation globale cohérente. Nous considérons ici les modèles de développement et non les modèles d'exécution tels que définis par R. France et B. Rumpe dans [85].

1.1.2 IDM, DSML, Composition de modèles partiels

Depuis la proposition de l'architecture Model Driven Architecture (MDA) [181], et plus généralement du Model Driven Engineering (MDE) [41, 62], l'Ingénierie Dirigée par les Modèles a contribué à centrer les activités de développement sur le paradigme de modèle qui est considéré dorénavant comme une entité de première classe [30]. L'un des intérêts de cette approche est de considérer les modèles à différents niveaux d'abstraction afin de faciliter leur réutilisation au cours du processus de développement. Les technologies de l'Ingénierie Dirigée par les Modèles fournissent également des supports à la manipulation des modèles : interrogation, transformation, composition, analyse, exécution [34].

La mise en œuvre de l'IDM a fait émerger la nécessité de fournir aux différents métiers des langages de modélisation adaptés. Elle a permis le développement d'applications basées sur la définition de modèles plus proches du domaine du problème que du domaine d'implémentation, réduisant ainsi la complexité des plates-formes [126]. Pour ce faire, l'IDM utilise des langages de modélisation spécifiques à un domaine.

Les **Domain Specific Modeling Language** (DSML) formalisent, par exemple, la structure, le comportement et les exigences des applications de domaines particuliers. Ils permettent de capturer les abstractions et la sémantique du domaine. Ils constituent des langages de modélisation définis pour permettre aux utilisateurs de se percevoir comme travaillant directement avec les concepts du domaine.

L'émergence des langages dédiés aux domaines et plus précisément aux métiers, s'est accompagnée de l'accroissement du nombre de DSML. Ces efforts ont également produit des outils supports à la production à la fois de langages ad-hoc et de modèles décrits dans ces DSML [192]. Le constat que l'on fait ensuite est bien entendu la nécessité (i) de s'adapter à la présence de multiples modèles partiels hétérogènes et (ii) de permettre l'inter-opérabilité des modèles produits.

Dans ce contexte de complexité croissante des systèmes logiciels, la multi-modélisation devient donc une activité majeure du processus de développement, notamment dans la phase de conception où plusieurs modèles sont généralement produits et doivent cohabiter et/ou être composés pour fournir un modèle global cohérent du système. Plusieurs approches s'intéressent à la multi-modélisation, que ce soient les approches traditionnelles comme la modélisation par points de vue [81] ou la modélisation par sujets [165], ou encore le développement par aspects [19, 174, 123], et l'ingénierie multi-modèles [153, 48, 57].

Afin de garantir notamment l'interopérabilité, la globalisation des modèles [61], l'automatisation des vérifications et la séparation des préoccupations en termes de réalisation du système, les modèles produits doivent pouvoir être transformés, composés, fusionnés, ou encore alignés. A l'instar des conclusions faites par M. Blay à propos des compositions de modèles dans le chapitre 1.2.4.2 de [33], nous considérons la composition de modèles comme une généralisation de la fusion et de l'alignement de modèles.

Définition 6. Composition de modèles : Consiste à mettre en relation des modèles indépendants les uns des autres et contribuant chacun partiellement à un objectif commun, pour réaliser un modèle global correspondant à ce but commun.

L'automatisation de la composition de modèles constitue encore un enjeu majeur du Génie Logiciel. Face à la problématique de la composition de modèles, deux axes de solutions sont possibles, (i) passer par l'espace des langages [60] ou (ii) rester au niveau modèle, dans une philosophie tout modèle. C'est le choix que nous avons fait en nous appuyant sur l'IDM et

les technologies qui l'accompagnent. La modélisation, enrichie de transformations, peut être utilisée pour fournir un cadre formel lors de la composition avec des domaines spécifiques.

1.2 Contexte applicatif et scientifique

Dans les systèmes complexes, de nombreux parties prenantes (ingénieurs, analystes, opérateurs, managers, etc.) issus de différents domaines (mécanique, électronique, logiciel, etc.) doivent travailler ensemble, et parfois même avec des acteurs étrangers au domaine d'application (avocats, commerciaux, autorités de certification, etc.). Ce problème a été identifié par l'INCOSE dans [116] comme l'un des défis majeurs de l'ingénierie des systèmes. La multitude des modèles hétérogènes produits lors du développement d'un système complexe (développement pris au sens large, de l'analyse des besoins à la réalisation du système) [196, 152], pose toujours le problème de la redondance des informations, de l'incohérence qui peut être introduite par des intérêts divergents et questionne encore l'apport d'une vision globale unifiée [39, 110, 132, 60, 58]. On voit bien notamment que les préoccupations posées et en partie traitées dans le cadre du MBSE s'intéressent à ces problèmes [172, 78, 151].

Les solutions que nous proposons dans nos travaux apportent des éléments de réponse à ces questions à la fois en phase de conception et en phase d'analyse du système logiciel.

Face à la complexité grandissante des systèmes logiciels, et à la nécessité de prendre en compte la collaboration entre plusieurs concepteurs, nous avons proposé une première approche d'analyse/conception centralisée par points de vue des systèmes complexes, VBOOL [143]. Nous avons élaboré une notation et un langage support, permettant de définir un modèle global du système et pour chaque acteur du système un accès correspondant à son propre point de vue. Cette approche, imposant aux utilisateurs de mettre en œuvre un nouveau formalisme unique et partagé par tous, s'est avérée intéressante pour ces utilisateurs mais en raison de l'évolution des standards, puis des outils et langages dédiés qui sont apparus ensuite, nous avons décidé de la faire évoluer. En effet, dans les années 2000 par exemple, la notation UML est devenue le standard de fait pour exprimer les modèles.

1.2.1 Méthodologie de développement orientée points de vue basée sur UML

Dans l'objectif de proposer une méthodologie de développement orientée points de vue basée sur des standards, nous avons d'abord défini un profil UML appelé VUML (View based Unified Modeling Language) [155]. VUML offre un formalisme de modélisation de systèmes logiciels par une approche combinant objets et points de vue. Sur le plan méthodologique, VUML propose une démarche centrée utilisateur qui permet d'intégrer la notion de point de vue dans le processus de développement des systèmes. Dans cette approche, plusieurs modèles de conception modélisant les besoins de chaque acteur interagissant avec le système sont développés par différents concepteurs. Ces modèles doivent être ensuite composés pour produire un modèle VUML unique représentant le modèle de conception global du système. L'ajout d'un point de vue (par exemple pour prendre en compte un nouvel acteur) peut nécessiter le développement d'un modèle de conception supplémentaire qui devra être composé à son tour avec le modèle VUML existant. Le principal ajout à UML est celui du concept de classe multivues. Une classe multivues est une unité d'abstraction et d'encapsulation qui permet de stocker et restituer l'information en fonction du profil de l'utilisateur. Elle offre des mécanismes de gestion des droits d'accès aux informations, de changement dynamique de points de vue et de gestion de la cohérence entre les vues dépendantes. De plus, VUML propose un modèle de composants multivues qui permet de représenter une classe multivues au niveau du diagramme de composants.

Sur le plan sémantique, VUML étend le méta-modèle d'UML et introduit un certain nombre de stéréotypes regroupés sous forme d'un profil UML. Nous avons fait une étude préliminaire de la fusion de modèles au sein du profil VUML [156], mais, en l'absence d'une approche systématique et réutilisable, nous n'avons pas pu la formaliser et donc l'automatiser. En effet, la tâche de fusion — même réduite dans un premier temps à la fusion de deux diagrammes de classes— est un processus relativement complexe qui consiste, en étant synthétique à : (i) harmoniser les diagrammes de classes de façon à éliminer les éventuels conflits résultant de modélisations séparées (ii) mettre les modèles en correspondance pour identifier d'une part les classes identiques qui doivent rester comme telles dans le modèle final, d'autre part les classes devant être fusionnées pour produire une classe multivue, et enfin les relations entre classes qui peuvent varier d'un modèle à l'autre. Dans le cas de différences entre modèles sur les relations entre classes de même nom — par exemple association versus spécialisation —, il est nécessaire d'appliquer une heuristique fondée éventuellement sur un patron de conception ou de demander l'intervention du concepteur qui contrôle la fusion, (iii) réaliser la fusion proprement dite en élaborant notamment les classes multivues sous la forme de structures composées d'une base et de vues. Ces éléments [157] constituent le point de départ des recherches présentées dans ce mémoire et ne seront pas plus détaillées.

1.2.2 Evolution des pratiques et des standards

- **Des modèles dans le standard UML, pré-existants au modèle global**

Afin de mieux correspondre à la réalité industrielle et de ne pas imposer de construire globalement les systèmes mais de permettre une élaboration a priori de modèles par points de vue de ce système, nous avons décidé de considérer ces modèles partiels pré-existants au modèle général. L'une des difficultés à résoudre avec cette approche est la composition de ces modèles partiels, élaborés séparément par points de vue. Une étude sur cette problématique a débouché sur la proposition d'une démarche d'analyse/conception, cette fois décentralisée, qui passe par une étape de composition des modèles partiels [58]. Pour traiter cette composition des modèles partiels, nous avons décidé d'appliquer des techniques d'IDM. Pour cela, comme dans [22], nous avons considéré la composition de diagrammes UML (diagrammes de classes) comme des transformations exogènes prenant en entrée plusieurs modèles conformes à UML et produisant en sortie un modèle conforme à VUML [15]. C'est ce que nous avons réalisé à travers la thèse de Adil Anwar soutenue en 2009 [14], et que nous détaillerons dans le chapitre 3 de ce mémoire.

Pour réaliser cette composition de modèles élaborés séparément sous la forme d'une transformation de modèles, nous avions fait l'hypothèse d'une uniformité descriptive entre les modèles partiels à composer. Dans un environnement collaboratif où coopèrent plusieurs concepteurs de métiers différents, cette hypothèse s'avère très contraignante pour les parties prenantes. En effet, l'évolution de l'IDM [179] a amené à envisager non pas la présence de modèles établis dans un langage commun, tel qu'UML, mais dans des langages plus proches des spécialités qui les mettent en œuvre.

- **Des modèles pré-existants au modèle global du système, hétérogènes**

Nous avons donc considéré des systèmes composés de modèles existants, issus de domaines métiers différents et établis dans des langages ad-hoc, des DSML, donc hétérogènes [111, 112]. L'objectif que nous avons poursuivi, et mis en œuvre à travers la thèse de Mahmoud El Hamlaoui [99], a été de garantir la cohérence du système global par la mise en correspondance des modèles de conception partiels et hétérogènes qui constituent ce système. Une solution aurait pu consister à transformer chaque modèle partiel en un modèle UML et à appliquer ensuite l'approche VUML. Cette technique, dont le principe est d'utiliser VUML comme langage pivot, a

été écartée car jugée trop lourde à mettre en œuvre. En effet, non seulement il faut dans ce cas réaliser des transformations des modèles partiels hétérogènes vers des modèles UML, ce qui n'est pas forcément simple, mais il faut surtout réécrire ces transformations à chaque évolution des modèles partiels. D'autres solutions ont également été étudiées notamment la composition des modèles par tissage [69], mais elle s'est avérée inadaptée. En effet, notre objectif était de fournir une approche qui permette de connecter plusieurs modèles partiels par des relations n-aires tout en supportant l'évolution à la fois du nombre de modèles, et de la nature de leurs connexions. Nous avons proposé la construction d'un modèle virtuel par connexion des modèles fournis en nous inspirant de [58]. Nous avons proposé un processus de mise en correspondance des modèles (fondé sur un méta-modèle de correspondance permettant de connecter les différents modèles) et nous l'avons étendu pour l'intégrer au processus de détection des changements et de leur gestion que nous avons proposé [104]. Ceci sera détaillé dans le chapitre 4, où nous montrons les contributions qui ont été faites pour supporter l'hétérogénéité de conception des systèmes complexes. '

Nous y développerons aussi une perspective intéressante de ce travail qui a consisté à considérer ces modèles évolutifs et issus de collaborations. Le travail effectué dans le cadre du Master2R de Thilelli Debiane [68] a consisté à proposer une extension du processus de mise en correspondance et de synchronisation de modèles hétérogènes défini dans la thèse de Mahmoud El Hamlaoui [99], pour prendre en compte la dimension collaborative de la réalité industrielle. Ce travail s'est poursuivi dans le cadre de la thèse de Saloua Bennani [24] qui a proposé des outils et des méthodes supports à la synchronisation collaborative de modèles de conception hétérogènes évolutifs. Cette approche, détaillée dans le chapitre 5, s'appuie sur des processus collaboratifs impliquant les utilisateurs des modèles considérés, et propose un outillage adapté. En effet, l'outil HMCS (Heterogeneous Matching and Consistency management Suite) réalisé dans le cadre de la thèse de M. El Hamlaoui [102, 99] est actuellement en cours d'extension dans le cadre de l'outil Collab-HMCS qui supporte la dimension collaborative de l'approche développée dans la thèse de S. Bennani.

- **Des modèles d'exigences hétérogènes**

L'alignement des modèles hétérogènes est une problématique qui reprend les notions de vues multiples, de leur complétude et de leur mise en cohérence, au niveau des modèles de conception, mais aussi possiblement au niveau des modèles de réalisation, des modèles d'exigences, ou de leurs méta-modèles respectifs. L'un des problèmes actuels de l'ingénierie système reste le risque d'incohérence lié à l'hétérogénéité des artefacts générés et plus particulièrement à la difficulté d'établir des liens entre ces différents artefacts, produits à différentes phases du cycle de vie de développement du système. Dans la thèse de Florian Galinier, l'objectif que nous avons poursuivi a été de proposer et développer une méthode outillée qui mette en œuvre de manière concrète les principes du développement « sans couture » (lean development ou seamless development) [148], visant à intégrer les exigences dans le code. Cette approche permet de n'avoir qu'un seul ensemble cohérent d'artefacts, qui évoluent de manière coordonnée au fur et à mesure du développement. L'une des pistes étudiées a consisté à adapter l'approche multirequirements proposée par Bertrand Meyer [148] aux spécificités des exigences systèmes et à mettre les modèles au centre du processus.

Nous avons ainsi considéré le processus logiciel dans une phase amont de la conception que nous avons traitée dans les autres thèses.

Les approches basées sur des modèles sont de plus en plus utilisées, en particulier dans les systèmes complexes tels que les systèmes aéronautiques. Des approches comme KAOS [188], SysML [164, 45], ou autres [28, 193], permettent aux ingénieurs d'obtenir, d'exprimer, de classer et de tracer les exigences. Nous avons défini RSML, un langage d'interface d'expression des exigences [87], utilisable et exploitable par les parties prenantes, qui fournit les constructions

nécessaires aux mécanismes de traçabilité, preuves, etc. inhérents aux diverses préoccupations, et par conséquent aux divers points de vue. En utilisant des techniques et outils de l'IDM et à l'aide de transformations [66], nous proposons également des passerelles vers des langages existants tels que SysML ou MSWord/MSExcel, et nous permettons aux ingénieurs de travailler avec leurs outils habituels, du type DOORS [114] ou Reqtify [183] par exemple. Nous solutionnons ainsi le problème de l'ingénierie des exigences lié à l'utilisation du langage naturel, en proposant des outils plus formels mais accessibles, transformant les expressions habituelles des exigences à la fois vers et à partir de RSML. RSML peut alors être considéré comme un pivot entre différents espaces d'expression. C'est aussi un pivot entre les domaines formels et non formels. Ceci sera présenté dans le chapitre 6.

1.3 Organisation de ce mémoire

L'idée que nous poursuivons est que la production des systèmes complexes peut être améliorée par la proposition de méthodes et d'outils d'analyse et de conception prenant en compte les différents points de vue sur ces systèmes, et par conséquent leurs différentes parties prenantes. Fonder un système complexe sur ses modèles partiels implique de faire de ceux-ci un tout cohérent. Nous proposons donc des mécanismes permettant d'assurer la validité du système en permettant d'aligner les modèles et de maintenir cet alignement dans les cas d'évolution. Nous avons dans un premier temps considéré la phase de conception des systèmes. Dans ce contexte de complexité croissante des systèmes logiciels, où la multi-modélisation est une activité majeure du processus de développement, la phase de conception est fondamentale. Plusieurs modèles de conception y sont produits et doivent cohabiter et/ou être composés pour produire un modèle global cohérent du système ; modèle global dont la cohérence doit aussi être assurée lors des évolutions des modèles qui le constituent [85]. Les modèles produits en phase de conception sont fondés sur les éléments établis dans la phase amont du cycle de développement qu'est l'analyse. Pour pouvoir raisonner, automatiser ou encore déduire les relations entre les modèles, nous nous sommes aussi intéressés à l'expression des exigences et aux modèles produits en phase d'analyse.

Dans le but de fournir un modèle global cohérent d'un système complexe, en se fondant sur la réalité de ses multi-utilisateurs, multi-modèles, nous sommes passés d'utilisateurs isolés auxquels nous imposions de travailler tous avec le même langage (d'abord nouveau, puis une norme), à des utilisateurs travaillant chacun avec ses propres outils et collaborant. La Figure 1.2 se veut une illustration de cette synthèse.

Les objectifs et les défis que nous avons relevés pour y parvenir font l'objet du chapitre 2 suivant.

Nous avons proposé des mécanismes permettant d'assurer la composition des modèles partiels par alignement de modèles homogènes (Chapitre 3), et de modèles hétérogènes (Chapitre 4), en intégrant les utilisateurs dans une démarche collaborative (Chapitre 5). Nous avons centré notre proposition sur l'utilisateur également dans la production d'un langage dédié à l'expression des exigences, suffisamment simple pour être utilisé sans connaissance formelle préalable, mais suffisamment formel pour pouvoir être soumis à des vérifications et des validations (Chapitre 6). Ce langage s'inscrit dans une démarche sans couture et fournit des ponts entre plusieurs espaces technologiques.

La dernière partie conclut d'abord ce mémoire en montrant comment les contributions décrites relèvent les défis annoncés (Chapitre 7). Le Chapitre 8 évoque ensuite quelques perspectives et les orientations futures que nous comptons donner à nos travaux.

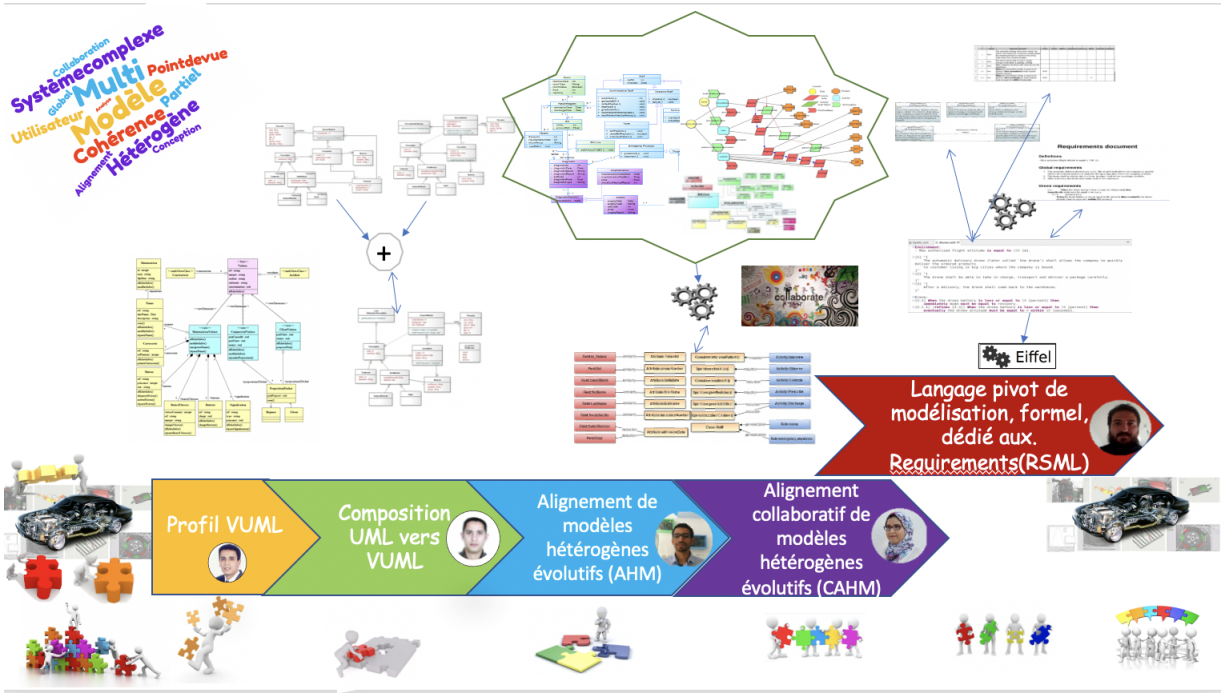


FIGURE 1.2 – Synthèse de nos contributions

Chapitre 2

Objectifs et défis de la modélisation par points de vue des systèmes complexes, centrée utilisateur

Lorsque l'on parle de système complexe et de modélisation de ces systèmes, on adresse tout de suite un grand nombre de domaines métiers et de spécialistes de ces métiers.

La modélisation d'une voiture par exemple va impliquer des modèles électroniques, mécaniques et logiciels. Il va en résulter un ensemble de modèles réalisés indépendamment, selon des points de vue différents, décrivant une même entité (cf. Figure 2.1).

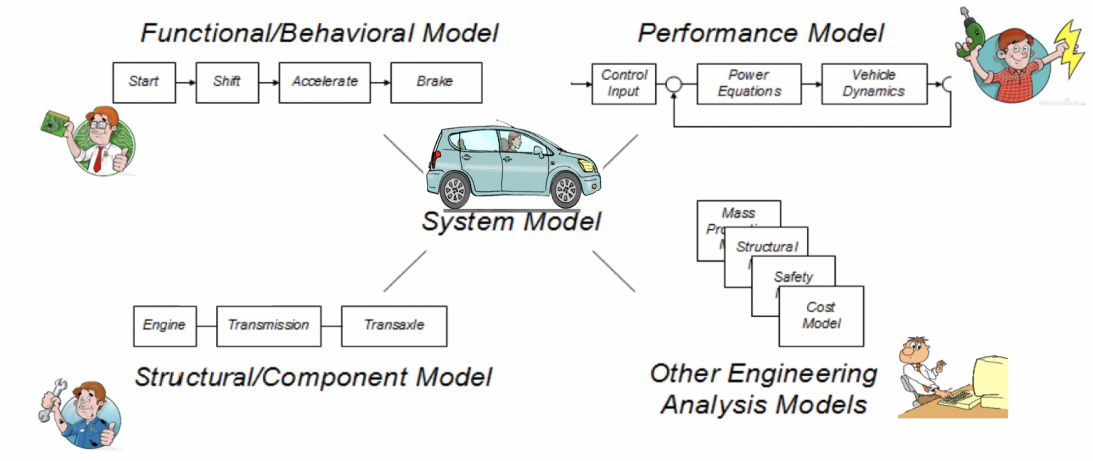


FIGURE 2.1 – Modélisation d'une voiture selon des points de vue différents, mettant en œuvre plusieurs modèles réalisés indépendamment - extrait de la présentation de [101]

Nous qualifions les modèles produits de **partiels**,car ils **donnent une représentation du système dans le cadre d'un point de vue donné et ne représentent donc que partiellement le système global**.

2.1 Multitude de modèles partiels et modèle global cohérent

Dans ce contexte où les systèmes sont représentés par l'ensemble de leurs modèles partiels, nombreux, manipulés par des utilisateurs aux profils variés, **la conception du système dans sa globalité doit être complète, cohérente et efficace.**

Elle doit être complète pour couvrir tous les aspects du système et tous les besoins identifiés de ce système.

Elle doit être cohérente et les différents modèles ne doivent pas introduire de risques d'incohérence et encore moins de contradictions.

Elle doit être efficace dans la mesure où les redondances peuvent être coûteuses et vont devoir être évitées.

Les questions qui se posent et auxquelles nous répondons sont par conséquent les suivantes :

Question 1. *Comment garantir la validité des modèles de conception dans leur ensemble ?*

Question 2. *Comment garantir le cohérence des modèles de conception dans leur ensemble ?*

Question 3. *Comment garantir la cohérence de la conception par rapport à l'analyse ?*

L'un des premiers objectifs que nous avons poursuivi a été de **favoriser la conception d'un modèle global tout en continuant à produire des modèles partiels adaptés à des points de vue variés.**

En effet, prendre en considération chacun des modèles comme faisant partie d'un tout et le lier aux autres modèles produits en les considérant dans le cadre d'un modèle global (cf. Figure 2.2), permet de mettre en évidence puis d'éviter les incohérences inhérentes à une modélisation répartie.

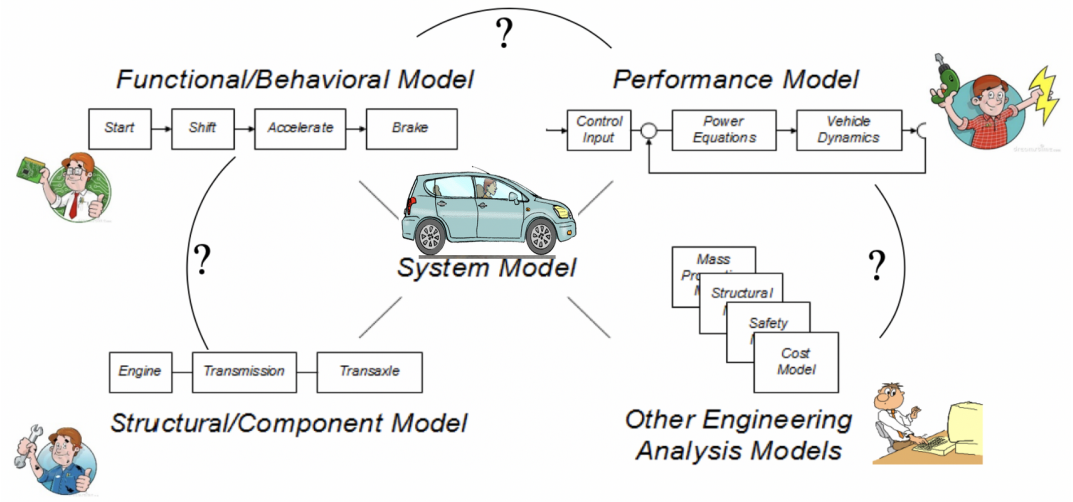


FIGURE 2.2 – Comment considérer le modèle global issu de la modélisation répartie ?

Le défi en réponse à ces questions consiste donc essentiellement à : *Fournir aux utilisateurs les moyens d'obtenir un modèle global du système tout en conservant la production de modèles partiels*

Les premières études concluantes effectuées en Ingénierie des modèles, devant la multitude de méthodes de modélisation conceptuelle (avec chacune ses points forts et ses points faibles), ont amené à établir une norme de modélisation. UML est le langage de modélisation qui été choisi par l'OMG. Nous avons donc dans un premier temps cherché à inscrire notre démarche dans le respect de cette norme.

Dans ce but, le défi a consisté dans un premier temps à *fournir aux utilisateurs une démarche leur permettant de produire des modèles partiels indépendants dans un même standard, UML, et à générer un modèle global commun.*

Plus précisément, un premier défi a consisté à *fournir une démarche et un opération de composition des modèles UML*

2.2 Multitude de modèles partiels hétérogènes et modèle global cohérent

Les concepteurs impliqués dans la modélisation d'un système complexe sont issus de domaines d'activités distincts et produisent des modèles partiels de ce système, selon leurs points de vue (Figure 2.3).

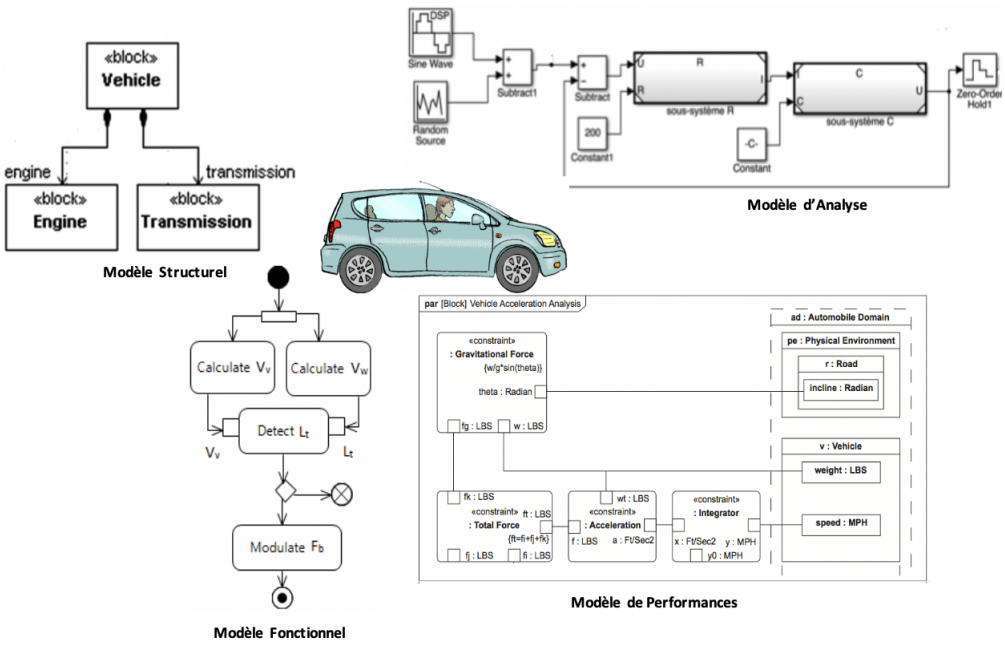


FIGURE 2.3 – Modélisation d'une voiture selon des points de vue différents

Ils sont habitués à évoluer dans des environnements dédiés, à utiliser des formalismes, des outils, des langages, des technologies qui leur sont familiers. L'utilisation d'un formalisme unique s'avère trop difficile à mettre en place et à imposer à ces utilisateurs, qui vont produire des modèles spécifiques à leurs points de vue.

La question qui vient naturellement alors est celle de la spécificité des besoins de ces différents utilisateurs, en termes d'outils de modélisation. La modélisation des systèmes complexes doit prendre en considération les besoins de chaque partie prenante du système en lui fournissant un environnement dédié à son point de vue sur le système.

Plusieurs approches ont été développées pour faire face à la modélisation **décentralisée** des systèmes complexes.

L'approche générale la plus utilisée dans l'industrie est la multi-modélisation [38], largement mise en œuvre en génie logiciel. Elle permet aux concepteurs de se concentrer sur différentes parties du système de manière isolée. Elle consiste ainsi à élaborer des modèles séparés qui correspondent à différents points de vue [110, 132, 39, 40].

La conception de systèmes complexes repose alors sur un ensemble hétérogène de langages, d'outils et d'environnements qui sont utilisés séparément par des concepteurs travaillant sur différents aspects du système [195, 85, 145]. En outre, ces experts en modélisation peuvent être situés dans des zones géographiques éloignées, comme c'est le cas dans le développement distribué et la collaboration de grands éditeurs.

La question qui se pose ici est donc de donner sa place à chaque partie prenante en tant que élément à part entière de la réalisation du système global. En effet, permettre à chaque *stakeholder* de travailler dans son environnement propre avec des techniques et des outils liés à son domaine d'expertise, s'avère une contrainte incontournable.

Question 4. *Comment assurer la cohérence de l'ensemble en permettant à chacun de travailler avec ses propres outils ?*

Permettre aux parties prenantes d'utiliser des langages spécifiques à leurs domaines métiers, des DSL [42], induit des modèles hétérogènes qui soient conformes à différents méta-modèles (Figure 2.4).

L'objectif que nous poursuivons est de fournir un modèle global du système tout en conservant la production de modèles partiels assurant que les besoins des parties prenantes sont pris en compte.

Dans [85], R. France et B. Rumpe, en 2007, identifient les défis suivants du MDE : "Les principaux défis auxquels les chercheurs sont confrontés lorsqu'ils sont tentés de réaliser la vision de l'IDM peuvent être regroupés dans les catégories suivantes :

- 1- Modélisation des défis linguistiques : ces défis découlent des préoccupations liées à la création et à l'utilisation d'abstractions au niveau des problèmes dans les langages de modélisation dédiés et à l'analyse rigoureuse des modèles
- 2- Séparation des défis des préoccupations : ces défis découlent des problèmes associés à la modélisation de systèmes utilisant des points de vue multiples et se chevauchant qui utilisent des langages potentiellement hétérogènes.
- 3- Manipulation des modèles et défis de gestion : ces défis découlent des problèmes liés (1) à la définition, à l'analyse et à l'utilisation des transformations de modèles, (2) au maintien des liens de traçabilité entre les éléments de modèle pour soutenir l'évolution

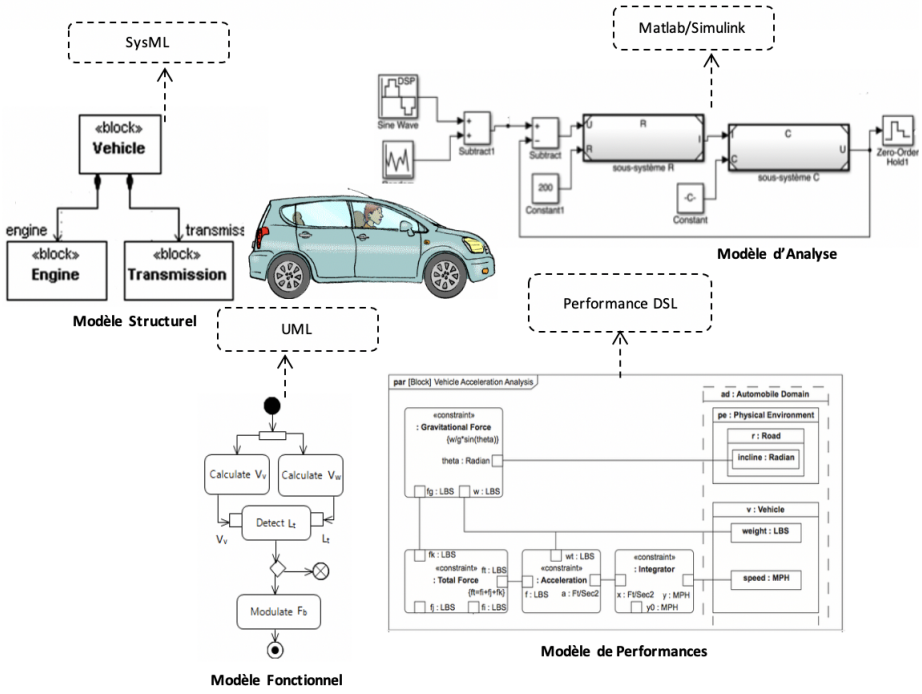


FIGURE 2.4 – Modélisation d'une voiture selon des points de vue différents, mettant en œuvre plusieurs méta-modèles

du modèle et la conception de l'aller-retour, (3) au maintien de la cohérence entre les points de vue, (4) au suivi des versions et (5) à l'utilisation des modèles pendant le cycle de vie."

Notre objectif de **Fournir un modèle global du système tout en conservant la production de modèles partiels assurant que les besoins des parties prenantes sont pris en compte** s'inscrit typiquement dans des éléments de réponses à la catégorie 2 de ces défis.

> Le défi que nous relevons ne consiste donc pas à fournir des DSL et des environnements dédiés aux parties prenantes (défi 1 identifié par R. France et B. Rumpe dans [85], mais de leur *permettre de continuer à évoluer dans leurs environnements propres pour contribuer à un modèle commun par la production de modèles partiels hétérogènes.*

Parmi les problèmes qui se posent dans une situation de modélisation multiple, l'un des principaux concerne l'utilisation conjointe de terminologies et de termes différents pour représenter ou exprimer un même concept. Ce problème, typiquement connu sous le nom de problème d'hétérogénéité est un problème commun largement partagé par la communauté des systèmes logiciels complexes [196], [152].
L'initiative GEMOC a fourni une classification des différents niveaux d'hétérogénéité en génie logiciel. Dans [21], les auteurs proposent une classification à trois niveaux : (i) hétérogénéité des systèmes (comme l'orchestration de sous-systèmes), (ii) hétérogénéité des plates-formes d'exécution (c'est le cas par exemple des moteurs de simulation), et enfin (iii) hétérogénéité de la modélisation qui traite de la coordination des modèles propres à un domaine.

Notre préoccupation se situe à ce troisième niveau.

L'un des objectifs que nous nous sommes fixés est de fournir des moyens d'aligner les modèles hétérogènes partiels d'un système, pour fournir un modèle global cohérent.

La première solution qui vient à l'esprit est de composer les modèles des différents points de vue en une unique représentation. De façon générale, les approches de composition proposées dans la littérature [73, 131, 200] reposant sur l'élaboration d'un modèle global présentent trois inconvénients majeurs liés à l'hétérogénéité des modèles. Le premier concerne la structure du méta-modèle associé au modèle composé. En effet, il n'y a pas de consensus sur la manière de le construire : à partir de l'union de tous les éléments provenant des modèles sources, ou de leur intersection, ou d'une autre combinaison ? La deuxième question concerne la sémantique utilisée pour représenter un élément d'un modèle composé étant donné que les modèles sources peuvent utiliser des sémantiques différentes. Enfin, un modèle global obtenu par composition peut être énorme dans le cas d'un système très complexe et donc très difficile à maintenir.

Pour assurer une cohérence globale des modèles partiels d'un même système, une solution peut être également de capturer les *correspondances* inter-modèles. Cette technique est appelée *alignement de modèles* par analogie avec *alignement d'ontologies*[180].

Le défi consiste ici à *réaliser cet alignement (Figure 2.5).*

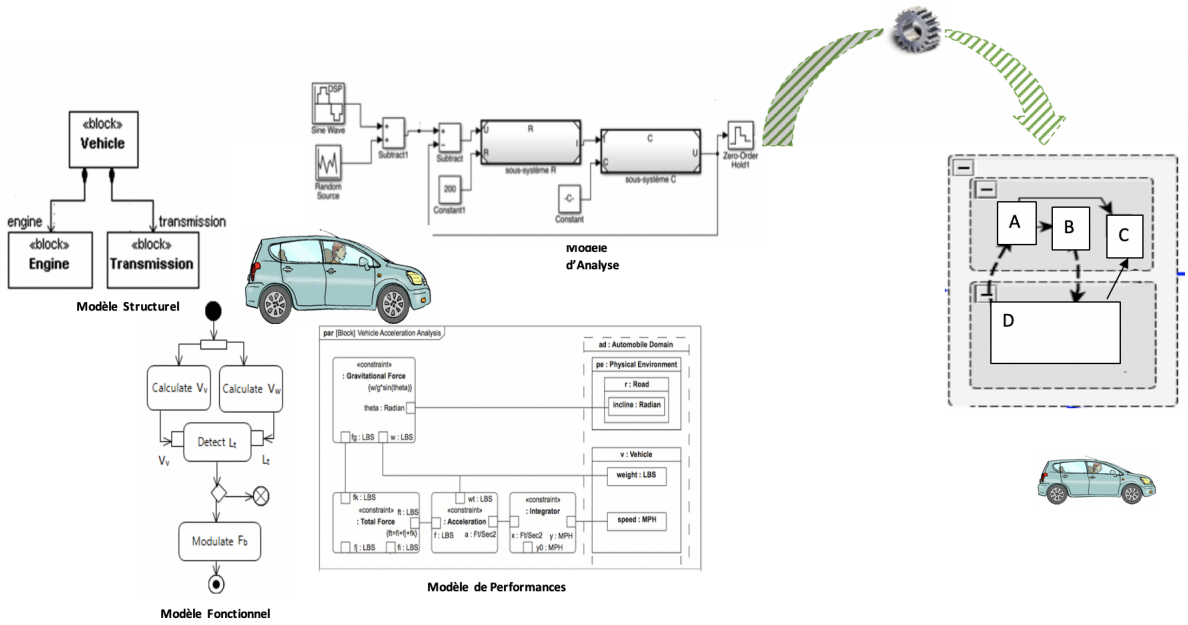


FIGURE 2.5 – Alignement des modèles partiels - production d'un modèle global

2.3 Evolution des modèles partiels hétérogènes et Gestion de la cohérence

L'alignement des modèles partiels qui constituent le système dans sa globalité, garantit la cohérence du modèle global. Mais, en particulier en phase de conception du système, ces modèles sont susceptibles d'évoluer. Les évolutions peuvent concerner un concept, une valeur, une relation dans un modèle donné, mais peuvent aussi être liées à l'ajout d'un nouveau modèle (l'introduction d'un nouveau point de vue par exemple).

R. France et B. Rumpe [85] ont identifié plusieurs problèmes inhérents à l'évolution des modèles : "3- Manipulation des modèles et défis de gestion : Ces défis découlent des problèmes liés ... (3) au maintien de la cohérence entre les points de vue, (4) au suivi des versions et (5) à l'utilisation des modèles pendant le cycle de vie."

Le défi que nous avons relevé consiste à *réaliser l'alignement des modèles hétérogènes, mais aussi à le maintenir dans les cas d'évolution d'un ou plusieurs modèles, ou encore d'introduction d'un nouveau modèle*.

En effet, une fois que le modèle de correspondance d'un système donné est établi, il doit être maintenu alors que les modèles par points de vue évoluent.

- **Remarque :** Nous ne considérons que les changements au niveau des modèles. L'évolution de méta-modèle est considérée comme nécessitant l'établissement de nouveaux modèles. L'impact du changement de l'un des méta-modèles (donc de l'un des DSL) utilisé est donc soit la suppression soit l'ajout d'un nouveau modèle.

L'objectif que nous poursuivons est de **fournir un support à la gestion de l'évolution des modèles**.

Les changements opérés dans les modèles qui cohabitent engendrent un bouleversement au sein du modèle global. Ils nécessitent donc (i) d'être capturés, (ii) d'être traités dans le sens où (iii) ils révéleront des modifications interdites ou bien (iv) ils devront être impactés aux autres modèles.

Le défi pour répondre à cet objectif consiste à *fournir et mettre en œuvre des mécanismes de gestion des évolutions des modèles partiels.*

Lorsqu'un modèle évolue, l'impact sur les autres modèles peut être important et nécessite une étude afin de calculer son incidence sur la cohérence du modèle global.

Si un expert du domaine peut avoir une vision assez large pour prendre les décisions qui s'imposent en termes d'impacts sur les autres modèles partiels et les relations qui lient les différents modèles concernés, une aide précieuse peut lui être apportée par les concepteurs des modèles concernés, qui ont une connaissance plus précise de leur métier. C'est pourquoi il nous a semblé important de donner une dimension collaborative au processus de maintien de la cohérence du système.

Néanmoins, la collaboration est également un critère clé pour garantir l'exactitude de l'alignement des modèles partiels. En effet, un alignement peut être considéré comme plus précis

s'il prend en compte tous les besoins des parties prenantes concernées. Pour cela, nous avons décidé d'étendre notre approche centralisée en y ajoutant une perspective collaborative dés la phase d'alignement des modèles.

L'objectif que nous poursuivons est donc de **fournir des processus collaboratifs permettant d'aligner les modèles partiels hétérogènes et de maintenir l'alignement en cas d'évolution**.

Le défi consiste à *mettre en œuvre des techniques collaboratives et à les appliquer à des processus de conception décentralisée*

2.4 Cohérence des multiples modèles d'analyse

L'un des problèmes actuels de l'ingénierie système reste le risque d'incohérence lié à l'hétérogénéité des artefacts générés et plus particulièrement à la difficulté d'établir des liens entre ces différents artefacts, qui sont produits à différentes phases du cycle de vie de développement du système. C'est pourquoi nous nous sommes aussi intéressés à la phase amont de la conception des systèmes, qu'est celle de leur analyse.

Si l'on remonte d'une phase dans le processus de production du système, l'analyse et la gestion des exigences qu'elle suppose, relèvent de problèmes et de défis similaires. En effet, l'alignement des modèles hétérogènes est une problématique qui adresse les notions de vues multiples, de leur complétude et de leur mise en cohérence, au niveau des modèles de conception, mais aussi au niveau des modèles de réalisation, des modèles d'exigences, ou encore de leurs méta-modèles respectifs.

Les approches basées sur des modèles sont de plus en plus utilisées, en particulier dans les systèmes complexes tels que les systèmes aéronautiques. Des approches comme KAOS [134], SysML [164, 45], ou autres [176, 28], permettent aux parties prenantes d'obtenir, d'exprimer, de classer et de tracer les exigences. En phase d'analyse, il parait donc nécessaire de leur permettre de continuer à travailler, en Langage Naturel ou avec des langages de modélisation semi-formels, tout en offrant des possibilités de vérification, à travers une formalisation plus ou moins rigoureuse.

L'objectif que nous avons poursuivi dans un premier temps a été de **fournir un langage d'interface d'expression des exigences, utilisable et exploitable par les parties prenantes, qui fournisse les constructions nécessaires aux mécanismes de traçabilité, preuves, etc. inhérents aux divers métiers, et par conséquent aux divers points de vue.** ‘

Par ailleurs, devant la réalité des usages des spécialistes de l'ingénierie des exigences dans le monde industriel, souvent fondés sur l'utilisation du langage naturel [150, 5], nous proposons de nous appuyer sur les techniques couramment utilisées et de fournir des supports à la traçabilité, à la vérification et aux preuves.

L'objectif poursuivi consiste donc ensuite à **gérer l'hétérogénéité des artefacts produits lors de l'expression des exigences, en garantissant leur inter-opérabilité et la cohérence de l'ensemble.**

Le défi à relever ici est double.

Il va s'agir en effet dans un premier temps d'*utiliser des techniques et outils de l'IDM pour fournir des éléments permettant d'exprimer les exigences dans un langage suffisamment proche du langage naturel pour être communément utilisé et suffisamment formel pour supporter des vérifications et des preuves.*

Par ailleurs, ce langage, support à des mécanismes de preuves et de vérification, de traçabilité, etc ... doit pouvoir s'interfacer avec les techniques et outils existants afin de les faire bénéficier des possibilités offertes par la formalisation.

Le défi associé consiste ensuite à *utiliser les mécanismes fournis par l'IDM pour supporter la gestion d'exigences via des modèles semi-formels et de garantir la cohérence des modèle formels ou semi-formels.*

L'approche que nous avons proposée intervient pour (i) permettre aux parties prenantes de travailler avec leurs outils habituels, éventuellement dans un cadre collaboratif, (ii) vérifier la cohérence des exigences entre elles mais aussi (iii) assurer la cohérence des exigences et de la conception, ou directement du code.

L'un des objectifs que nous avons poursuivis **a été de définir une approche permettant de n'avoir qu'un seul ensemble cohérent d'artefacts, qui évoluent de manière coordonnée au fur et à mesure du développement.**

Le défi associé consiste à *proposer et développer une méthode outillée qui mette en œuvre de manière concrète les principes du développement « sans couture » et du modèle unique.*

Deuxième partie

Contribution : Mécanismes intégrés supports à la modélisation par points de vue

Chapitre 3

Composition de modèles de conception homogènes basés sur le standard UML

Cette section reprend les éléments exposés dans l'article *"A Rule-Driven Approach for composing Viewpoint-oriented Models"*, Dans : Journal of Object Technology, ETH Swiss Federal Institute of Technology, Vol. 9 N. 2, p. 89-114, 2010 [15]. Elle fait état des résultats obtenus dans le cadre de la thèse d'Adil Anwar, *"Formalisation par une approche IDM de la composition de modèles dans le profil VUML"*, soutenue le 11 décembre 2009 à Toulouse [14].

Dans ce chapitre, je donnerai d'abord un bref aperçu du profil VUML et décrirai l'approche de modélisation que nous avons proposée, basée sur les vues dans un contexte d'Ingénierie Dirigée par les Modèles. Elle permet de composer des diagrammes de classes UML en un seul diagramme de classes VUML (appelé modèle global). Au centre de l'approche, une opération de composition de modèles (UML -> VUML), est définie formellement. Ceci a été mis en œuvre sur l'étude de cas d'un système de gestion des dossiers médicaux partagés (SGDMP).

3.1 Contexte

3.1.1 Profil VUML

Le profil VUML a été développé pour répondre aux besoins d'analyse et de conception de systèmes complexes selon différents points de vue. Dans VUML, un point de vue représente la perspective selon laquelle un acteur donné interagit avec le système. Un point de vue y exprime donc les exigences et les droits d'un acteur. Une vue est le résultat de l'application d'un point de vue sur une entité donnée du système. Le principal concept ajouté par VUML à UML est celui de classe multiview qui est composée d'une classe de base (partagée par tous les points de vue), et d'un ensemble de vues (elles mêmes des classes, extensions de la classe de base) ; chaque vue étant spécifique à un point de vue donné (cf. Figure 3.1).

La syntaxe abstraite et la sémantique de VUML sont décrites par un méta-modèle, un ensemble de règles bien formées exprimées en OCL [162], et un ensemble de descriptions textuelles en langage naturel. Sur le plan méthodologique, un processus informel permet d'analyser et de concevoir des systèmes logiciels en fonction des points de vue. Les modèles partiels produits en UML sont fusionnés en un modèle VUML. Cette intégration, manuelle, nécessitait l'apport de moyens d'automatisation du processus. Un générateur de code multi-cibles a

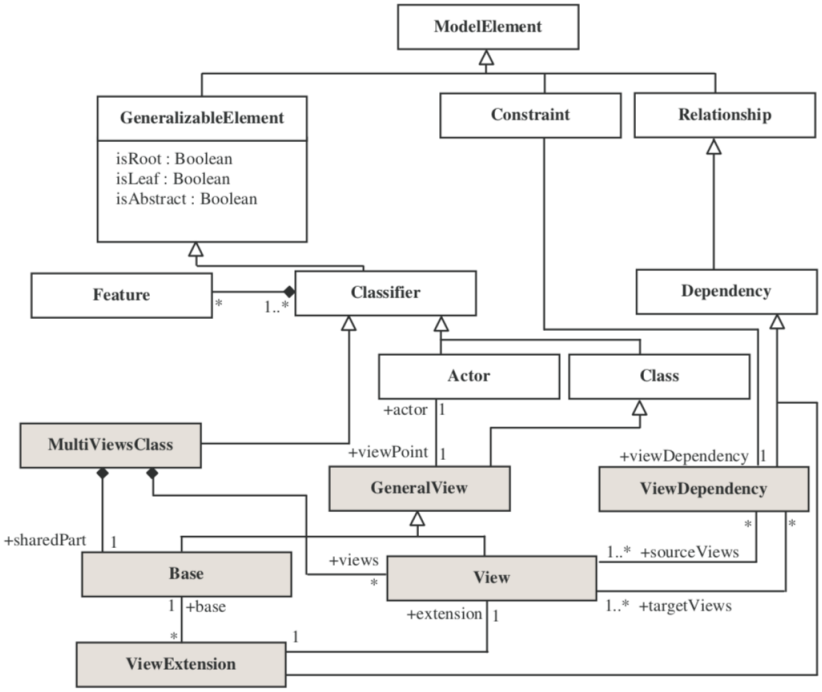


FIGURE 3.1 – Méta-modèle de VUML

également été développé pour produire du code (testé sur Eiffel et Java), à partir d'un modèle VUML.

3.1.2 Processus automatisé de conception VUML basé sur l'IDM

Pour traiter les questions de composition des modèles, l'IDM est apparue comme une solution prometteuse, en considérant certaines étapes de la composition comme des cas particuliers de transformations de modèles. Pour cette raison, les techniques connues d'IDM [30] et particulièrement le concept de transformation ont été utilisés pour formaliser et mettre en œuvre la composition de plusieurs modèles de points de vue réalisés en UML, en un seul modèle VUML. Plus précisément, la composition a été considérée comme une transformation exogène du même niveau d'abstraction (car elle prend en entrée un ensemble de modèles PIM exprimés en UML et génère en sortie un autre modèle PIM qui est conforme au profil VUML). VUML vise à réduire la complexité de conception des systèmes logiciels par décomposition, en fonction des points de vue des acteurs sur le système, c'est à dire de leurs besoins et de leurs droits d'accès. Cette séparation horizontale des préoccupations complète l'approche verticale de l'IDM en proposant une méthodologie qui permette de développer des modèles à chaque niveau d'abstraction. Cependant, pour intégrer pleinement une approche pilotée par les modèles, il est important de définir et d'automatiser les transformations entre les modèles impliqués. La description d'un système en VUML s'inscrit dans la démarche présentée sur la Figure 3.2 ci-dessous.

La première phase du processus de conception avec VUML est l'identification des besoins des acteurs. L'objectif principal est de créer un modèle d'exigences (diagramme de cas d'utilisation UML). De plus, afin de réduire les incohérences éventuelles des modèles réalisés lors de la phase de conception, le modèle des exigences est enrichi d'un glossaire qui précise les concepts de base du domaine et sert de référence pour les concepteurs du système. La deuxième phase du processus, décentralisée, consiste à développer des modèles distincts, chacun représentant un point de vue. Le résultat de cette phase est un ensemble de modèles UML adaptés

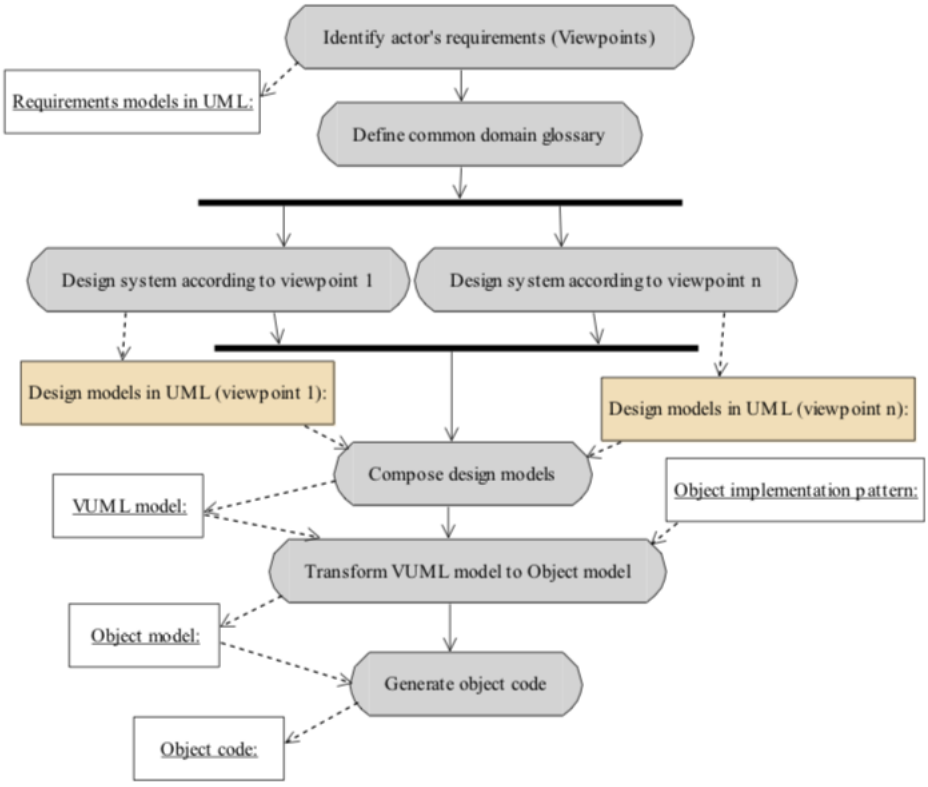


FIGURE 3.2 – Processus dirigé par les modèles de VUML

aux besoins des acteurs impliqués, soit un ensemble de modèles de points de vue. Nous avons focalisé notre étude sur les modèles de conception statiques (modèles de classes).

La troisième phase du processus se compose de trois étapes. La première est une étape de pré-composition qui révèle et corrige les différents conflits sur ces modèles (noms, structure, etc.). Cette étape est jusqu'à présent effectuée manuellement par un ou plusieurs concepteurs et n'a pas été traitée dans la thèse d'Adil Anwar. La deuxième étape, automatique, vise à composer des modèles PIM. Cette composition est une transformation exogène car elle prend en entrée n modèles PIM définis en UML et génère en sortie un modèle conforme au profil VUML. Une fois le modèle VUML généré, la troisième étape consiste à l'affiner. (Cette opération de raffinement n'est pas représentée sur la Figure 3.2.)

Elle consiste à identifier et décrire les dépendances possibles entre les vues d'une classe multivues donnée, afin d'assurer la cohérence du modèle du système. Ces dépendances sont modélisées dans VUML par des relations de dépendance stéréotypées par "viewDependency", et annotées par des contraintes exprimées en langage OCL.

La dernière phase du processus de conception met en œuvre une plate-forme d'exécution. Le modèle VUML est transformé en un modèle d'implantation, fonction de la plate-forme. Cette phase est réalisée en appliquant un modèle de génération de code objet tel que décrit dans [14]. La technique appliquée combine l'utilisation de la transformation de modèle et des modèles de conception (comme décrit dans [122], et donne lieu au développement d'un transformateur modèle à modèle en ATL (VUML2JAVA).

3.1.3 Étude de cas

Pour illustrer notre approche et ses objectifs, nous avons considéré un système de gestion de dossiers médicaux partagés (SGDMP). Pour simplifier, nous avons limité notre étude aux acteurs et activités suivants :

- Les patients : suivent les traitements prescrits par les médecins et subissent des analyses en laboratoire. Ils peuvent également consulter leur dossier médical.
- Les médecins : effectuent des diagnostics et des consultations, rédigent des ordonnances, prescrivent des médicaments et consultent des rapports médicaux.

a. Modélisation des points de vue du SGDMP

Pendant cette phase, le système est modélisé selon des points de vue donnés. Considérons le point de vue du médecin et, par exemple, le scénario "Enregistrer un traitement" du cas d'utilisation "Prescrire un traitement". Le but de ce scénario, lancé par le médecin après une consultation, est de créer un "Formulaire de consultation" qui serve à enregistrer tous les actes cliniques concernant un patient donné et les décisions prises par le médecin pendant la consultation. En procédant de manière incrémentale avec tous les cas d'utilisation, les objets identifiés ainsi que les méthodes associées apparaissant dans les diagrammes de séquences, permettent de construire un diagramme de classes lié au point de vue du médecin (Figure 3.3). On peut noter que le médecin a accès à l'information sur les analyses effectuées en laboratoire, ainsi qu'aux rapports créés par d'autres professionnels de la santé (médecins, infirmières, etc.).

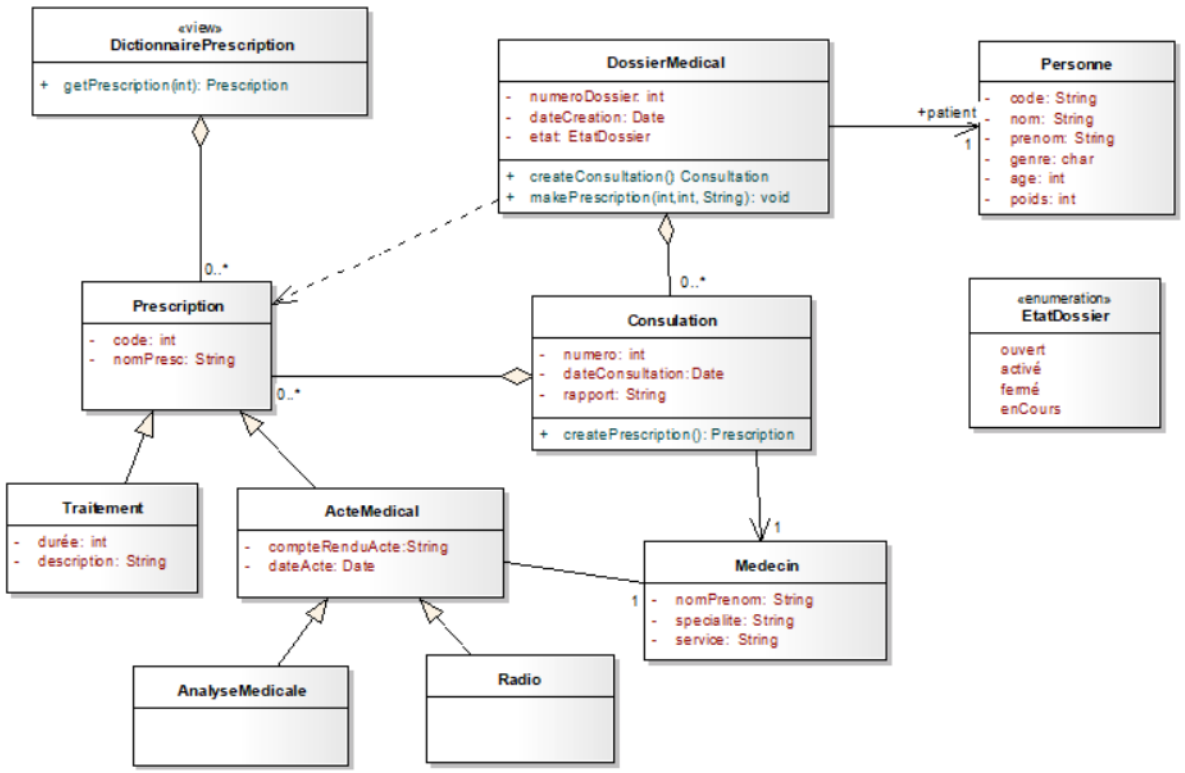


FIGURE 3.3 – Diagramme de classe du point de vue du médecin

Du point de vue du patient, un processus similaire est réalisé et produit le diagramme de classes de la Figure 3.4. Il montre que le patient a accès aux ordonnances des médecins qui le concernent, ainsi qu'à la liste des fiches de paiement associées à ses traitements, mais qu'il n'a

accès directement ni aux rapports des laboratoires d'analyses, ni aux rapports rédigés par les infirmières (qui sont réservés aux médecins).

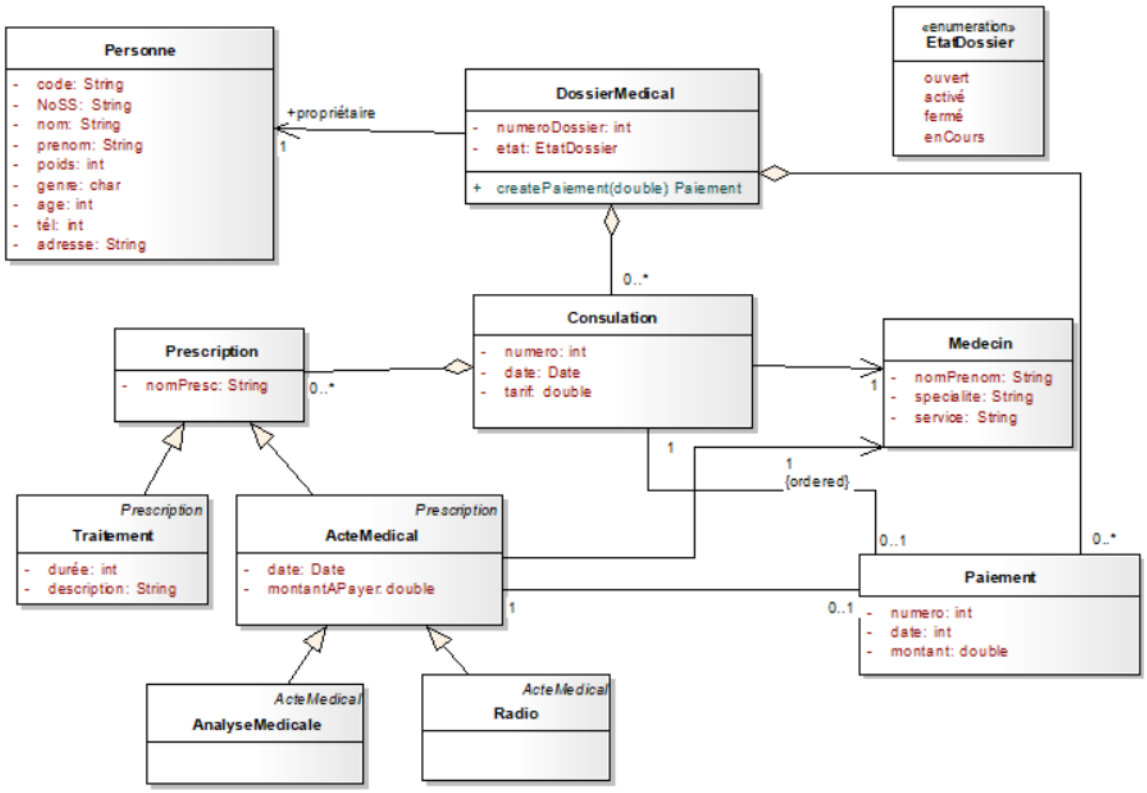


FIGURE 3.4 – Modèle du Point de vue du patient

b. Modélisation VUML du SGDMP

Dans cette phase, les modèles de points de vue sont composés pour produire un modèle VUML. La Figure 3.5 illustre le modèle VUML résultant de la composition des deux modèles de points de vue présentés sur les figures 3.3 et 3.4). Les classes apparaissant dans deux modèles de points de vue, avec le même nom et avec des propriétés différentes (attributs, opérations, associations, etc.), sont fusionnées sous la forme d'une classe multi-vues. La Figure 3.5 montre deux classes multi-vues : *Consultation* et *DosssierMedical*. Les propriétés de la classe *DosssierMedical* qui sont partagées par les deux points de vue considérés sont dans la classe stéréotypées par "base" ; les propriétés qui sont spécifiques à un point de vue sont dans des classes stéréotypées par "view". Les classes multivues sont stéréotypées "multiViewsClass" lorsqu'elles ne sont pas développées sur le diagramme.

Pour nommer les classes stéréotypées par "view", nous avons adopté la notation recommandée par VUML (nom de l'acteur + nom de la classe de base). Cette stratégie permet d'assurer la traçabilité entre les éléments des modèles de points de vue et le modèle VUML.

Notons que la classe *Medecin* n'est pas une classe multivues car elle est exactement la même (nom et contenu) dans les deux modèles de points de vue.

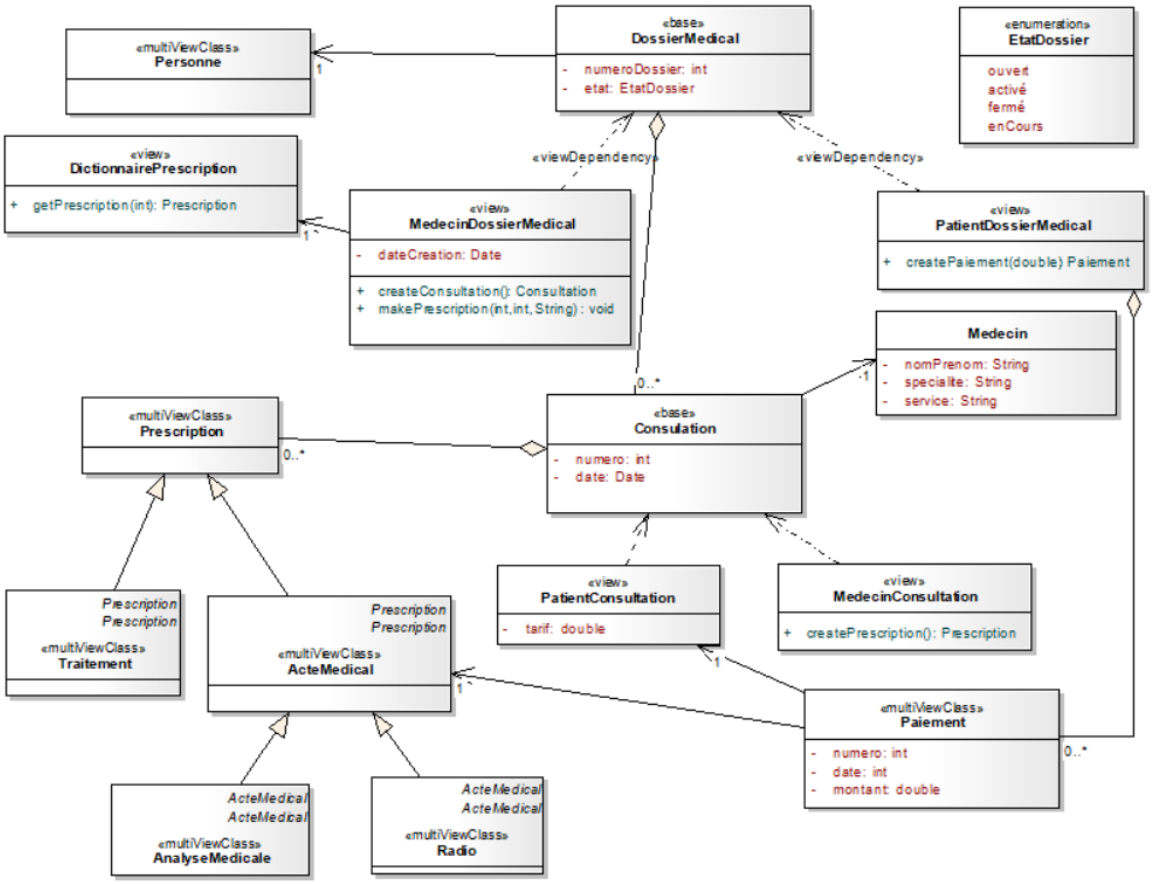


FIGURE 3.5 – Modèle VUML du SGDMP

3.2 Processus de composition de modèles UML

Dans cette section, je détaillerai le processus de composition du modèle que nous avons proposé. Il est également structuré en trois phases :

- une phase de pré-composition,
- une phase de composition,
- une phase de post-composition qui est semi-automatique (guidée par l'utilisateur).

Le processus de composition est représenté par un flux d'activité sur la Figure 3.6.

3.2.1 Phase de pré-composition

L'objectif principal de cette phase est d'harmoniser les modèles partiels afin d'éliminer les conflits possibles (nom, structure, etc.) résultant d'une modélisation séparée. Cela se fait entre autres par la résolution des conflits en déterminant les incohérences et les similitudes entre les éléments des modèles de conception des différents points de vue. Cette phase fait face à des conflits tels que la polysémie (même nom et significations différentes), la synonymie (même sens et noms différents) et les incohérences structurelles (en particulier les relations de généralisation ou d'association). Dans ce dernier cas, il est nécessaire d'appliquer une heuristique qui peut être basée sur des motifs, ou d'exiger l'intervention du concepteur qui contrôle le processus de composition. Par ailleurs, certains des conflits identifiés ici peuvent également être dus au fait que les différents acteurs du système peuvent avoir des objectifs contradictoires. Ce problème particulier, n'est pas traité dans le cadre de cette approche.

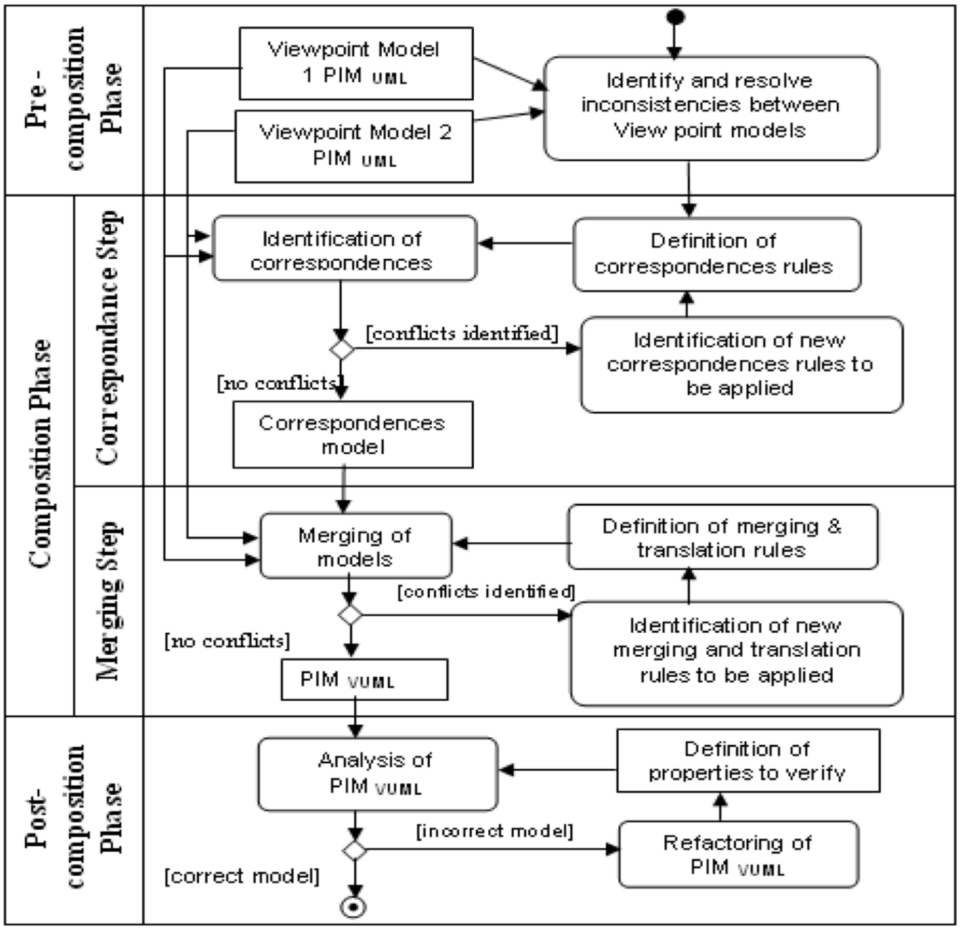


FIGURE 3.6 – Processus de composition dans VUML

3.2.2 Phase de composition

Nous nous sommes fondés sur [83, 79] et [131] qui soutiennent que l'automatisation de la composition des modèles comprend deux tâches différentes qui devraient être effectuées par deux opérateurs distincts : un opérateur de correspondance et un opérateur de fusion. Cette séparation facilite la maintenabilité du processus de composition puisque l'opération de correspondance est plus stable que l'opération de fusion qui peut obéir à des stratégies modifiables. C'est pourquoi nous avons proposé une opération de composition en deux étapes : une phase de mise en correspondance et une phase de fusion (Figure 3.7). L'étape de mise en correspondance (Correspondance) consiste à identifier les liens entre les modèles à composer (nous ne considérons ici que deux modèles sources). Il est régi par des règles de correspondance qui mettent en œuvre des stratégies de comparaison entre les éléments de modèle. La comparaison des éléments est basée sur des propriétés internes définies au niveau du méta-modèle. Par exemple, un sous-ensemble de propriétés internes d'une classe UML peut être représenté par nom, isAbstract, ownedAttribute, ownedOperation, qui sont des propriétés de la classe métaclasse dans le méta-modèle UML [163]. Une règle de correspondance, appliquée à deux éléments décrivant le même concept dans des modèles source différents, crée une relation de correspondance entre ces éléments. Cette relation est ensuite enregistrée dans un modèle de correspondances.

L'étape de Fusion dépend du méta-modèle cible. Dans notre contexte applicatif, cette étape de fusion vise à produire un modèle VUML dont les éléments sont stéréotypés selon le profil VUML. En fait, les éléments VUML sont créés en appliquant des règles de fusion (*merging rules*) et des règles de traduction (*translation rules*). La stratégie de fusion dépend principa-

lement du type de lien. Les règles de fusion s'appliquent aux éléments qui sont liés les uns aux autres par des relations de correspondance. En ce qui concerne les règles de traduction, elles s'appliquent aux éléments qui n'ont pas de correspondant dans le modèle opposé (ils sont simplement copiés dans le modèle VUML).

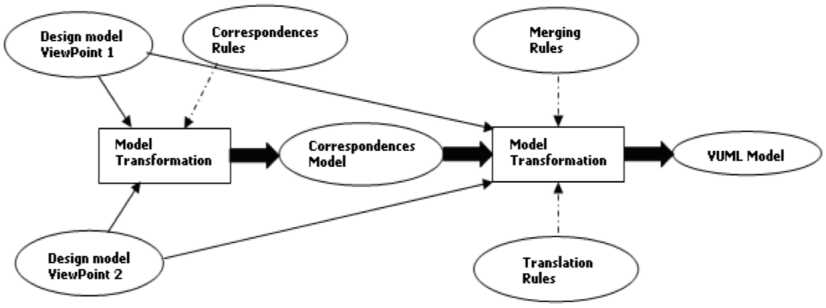


FIGURE 3.7 – Chaîne de transformation d'une composition VUML

3.2.3 Phase postérieure à la composition

Après la phase de composition, une étape d'analyse est effectuée (voir Figure 3.8) afin de découvrir d'éventuelles erreurs de composition. Le modèle VUML composé est vérifié par rapport aux propriétés souhaitées et vis-à-vis de sa conformité aux règles de bonne formation. Lorsqu'une règle est violée, une erreur est détectée et un élément de problème est créé et stocké dans un modèle d'erreurs conforme à un méta-modèle d'erreurs tel que définit dans [31]. Le modèle d'erreurs peut être analysé puis importé dans un outil de *refactoring* de modèles dédié à la résolution de tels problèmes. Les règles de bonne formation définies au niveau du méta-modèle pour exprimer la sémantique statique de VUML - en particulier celles relatives aux constructions du langage - sont utilisées pour développer une technique de preuve basée sur les propriétés, comme décrit dans [64].

La Figure 3.8 décrit le principe de la transformation utilisée pour vérifier le modèle VUML composé. Elle produit un modèle de diagnostic détaillant les erreurs identifiées.

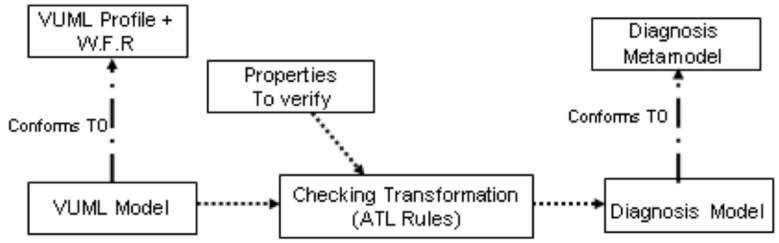


FIGURE 3.8 – Vérification d'un modèle VUML avec transformation ATL

Cette technique est basée sur une extension d'OCL permettant de produire des informations détaillées dans un modèle de sortie. Elle met en œuvre les règles de contrôle définies en ATL. En effet, une règle ATL est définie pour chaque contrainte à vérifier. Le contexte de la contrainte OCL définit le type de la source du motif de la règle, tandis que la condition de garde est la négation de l'expression booléenne associée à la contrainte. Enfin, le type du motif cible caractérise une erreur (Problème). Ce type, défini dans le méta-modèle de diagnostic de J. Bézivin et al. [31], donne des précisions sur l'erreur (gravité, localisation, description, etc.).

3.3 Cadre générique pour la composition de modèles

Un modèle de composition devrait fournir des moyens de prendre en charge les caractéristiques communes pour la construction d'un opérateur de composition. L'enquête présentée dans [30] résume un ensemble de base d'exigences minimales. Nous avons défini un opérateur de composition générique à travers trois composantes : *relations, transformations et stratégies*.

- La composante *relations* permet de définir et de capturer les relations entre les éléments du modèle.
- La composante *transformations* permet d'effectuer des transformations entre les modèles impliqués.
- La composante *stratégies* permet de définir des stratégies de transformation. Ce sont les stratégies qui spécifient la sémantique des règles de transformation.

Pour décrire ces trois types de composantes, nous avons défini trois méta-modèles : le méta-modèle de correspondances, le méta-modèle des règles de transformation et le méta-modèle des stratégies de transformation, qui sont décrits dans les sections ci-dessous.

3.3.1 Méta-modèle générique de correspondances

Le méta-modèle de correspondance définit les différents types de relations entre les éléments du modèle indépendamment de tout domaine d'application donné (Figure 3.9). Nous avons étendu le méta-modèle de tissage de base proposé dans [69] pour répondre aux exigences de composition et pour gérer de nouveaux types de relations. Les métaclasses centrales dans le méta-modèle de correspondance sont donc :

- *CorrespondenceRelationship* : définit la relation entre les éléments des modèles sources. La définition d'une nouvelle relation se fait par une spécialisation de cette métaclasse, ce qui permet la définition sémantique de chaque relation (ex : égalité, équivalence, dépendance, etc.).
- *CorrespondenceRelationshipEnd* : représente l'extrémité d'une relation de correspondance.
- *CorrespondenceElementRef* : modèlise les concepts de référence. Elle contient un nom d'attribut qui représente le nom de l'élément référencé, et un attribut ref qui agit comme un identificateur persistant d'élément de modèle.
- *ReferencePackage* : est un conteneur pour les éléments de référence. Une instance de cette métaclasse contient toutes les références des éléments liés.

La métaclasse *CorrespondenceRelationship* doit être spécialisée pour créer différents types de relations. C'est une solution pratique pour établir une sémantique donnée pour chaque relation de correspondance. Par exemple, la métaclasse *PartialCorrespondenceR* indique que les éléments liés par une instance de cette relation sont deux vues qui représentent le même concept, mais diffèrent par certaines propriétés (par exemple : deux classes avec le même nom mais ayant des attributs ou opérations différents).

La métaclasse *TotalCorrespondenceR* définit un type particulier de relation entre des éléments qui représentent des vues cohérentes du même concept (c'est-à-dire l'égalité dans le sens où deux éléments apparaissent de la même manière dans plusieurs modèles).

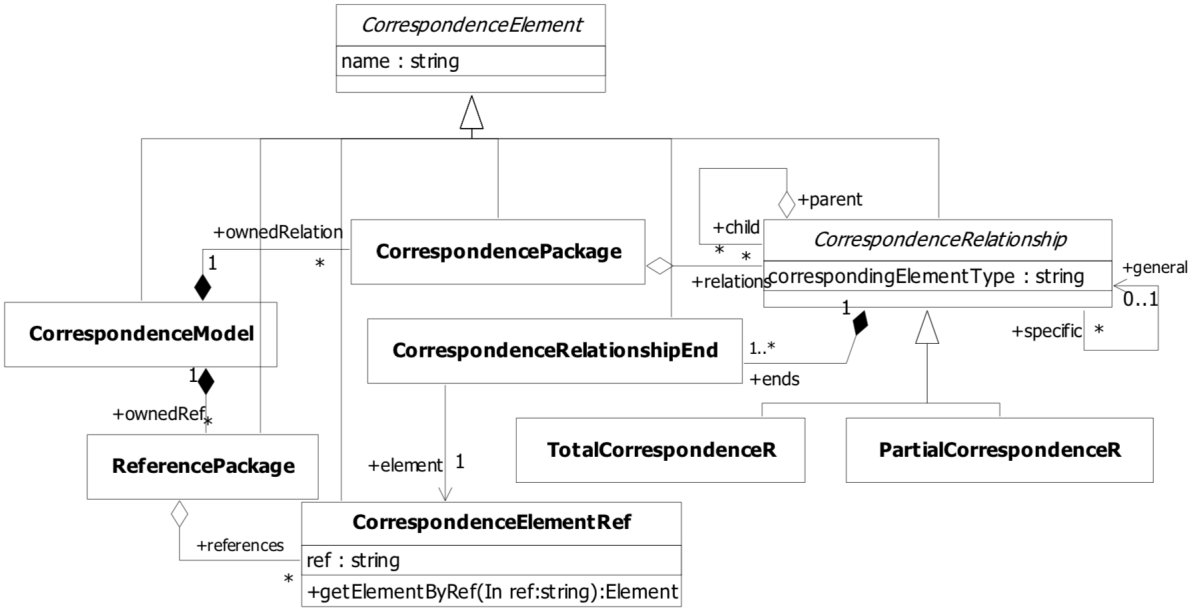


FIGURE 3.9 – Méta-modèle de correspondance générique

3.3.2 Méta-modèle des règles de transformation

Le processus de composition est donc guidé par des règles de transformation structurées en trois catégories : correspondance, fusion et traduction. Cela permet d'abord d'établir des correspondances entre les modèles d'entrée, ensuite de fusionner ces modèles dans un modèle global. Les éléments qui n'ont pas d'élément correspondant dans le modèle opposé sont transformés selon des règles de traduction. Comme les stratégies de transformation dépendent de chaque domaine spécifique, nous ne les considérons pas dans la partie générique du méta-modèle. Le méta-modèle des règles de transformation est une description générale des règles de transformation (Figure 3.10).

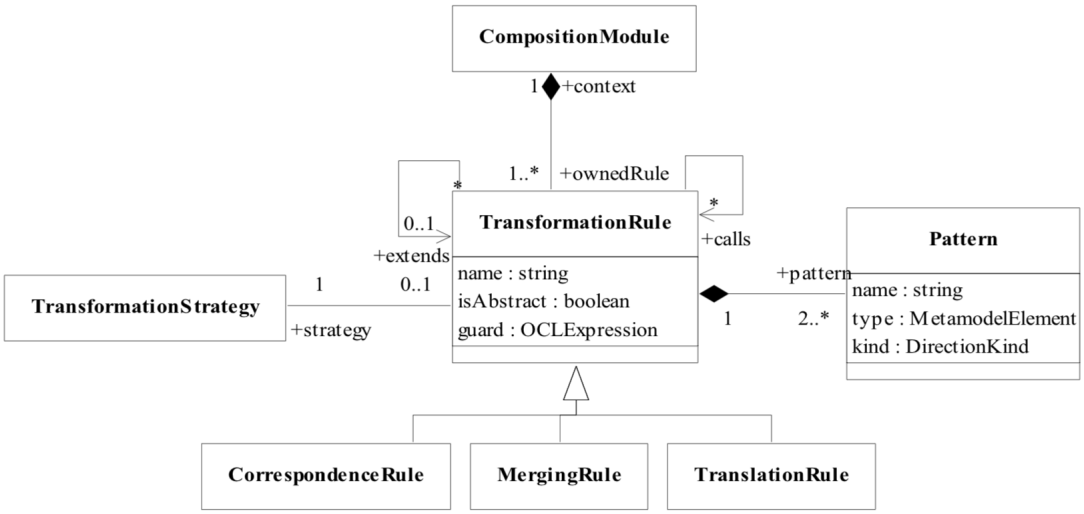


FIGURE 3.10 – Méta-modèle des règles de transformation

Chaque règle de transformation est composée d'au moins deux modèles utilisés pour détecter les éléments des modèles source et cible. Les types de motifs proviennent des éléments des méta-modèles impliqués dans la transformation. Si l'on considère une règle de correspondance, le type du motif d'entrée provient du méta-modèle source, alors que le type du motif de sortie provient du méta-modèle de correspondance. Dans la section suivante, nous allons voir comment ce méta-modèle permet de définir (lorsqu'il est instancié) des règles de transformation spécifiques pour composer des modèles dans un domaine particulier.

3.3.3 Méta-modèle des stratégies de transformation

Jusqu'à présent, nous avons explicité les aspects structurels de notre opérateur de composition. Cependant, ces aspects structurels ne suffisent pas à fournir une définition complète de cet opérateur, car il faut aussi décrire son comportement. Nous utilisons des stratégies de transformation pour spécifier l'aspect comportemental de chaque règle de transformation (Figure 3.11). Les stratégies sont définies par Kolovos et al. [131] comme des algorithmes enfichables qui peuvent être attachés à des règles de transformation pour implémenter une fonctionnalité récursive et réutilisable, elles peuvent être déduites de la structure du méta-modèle. L'utilisation de stratégies a l'avantage de minimiser l'intervention manuelle du développeur. Les stratégies de correspondance définissent la logique de comparaison entre les éléments du modèle. Nous distinguons trois types de stratégies de correspondance. Le premier type est basé sur les signatures décrites dans [175]. La signature d'un élément est décrite par un ensemble de propriétés internes (nom, type, cardinalité, etc.) définies dans le méta-modèle. Pour les éléments de type Class, la stratégie dépend des valeurs de leurs méta-propriétés. Par exemple, si l'on considère le couple (nom, isAbstract), la comparaison de deux classes définies dans deux modèles différents est réduite à la comparaison des valeurs de ces deux propriétés. Les stratégies de correspondance peuvent également être basées sur des relations entre des éléments tels que l'héritage ou la délégation ; dans ce cas, la stratégie de correspondance dépend des informations connues sur les éléments reliés à chaque élément des différents modèles.

Contrairement aux stratégies de correspondance, les stratégies de fusion dépendent du type et de la sémantique associés aux relations de correspondance qui lient les éléments source, ainsi que de la structure et de la sémantique des éléments à créer dans le méta-modèle cible. *UnionMergingStrategy* est utilisé lorsque différents modèles source contiennent des classes avec le même nom mais avec des propriétés différentes. Une simple union des propriétés initiales donne les propriétés de la classe résultante.

TotalMergingStrategy est utilisé pour fusionner deux classes, qui sont conformes.

PartialMergingStrategy permet de créer deux éléments ou plus dans le modèle cible.

3.4 Processus de spécialisation générique

Cette section décrit comment une composition générique peut être spécialisée pour créer un opérateur de composition particulier adapté à un langage de modélisation spécifique. Cette spécialisation peut être divisée en quatre étapes (Figure 3.12) : (1) Spécialisation du méta-modèle de correspondance, (2) Définition d'un modèle stratégique, (3) Définition d'un modèle de règles de transformation, (4) Génération de transformations par composition.

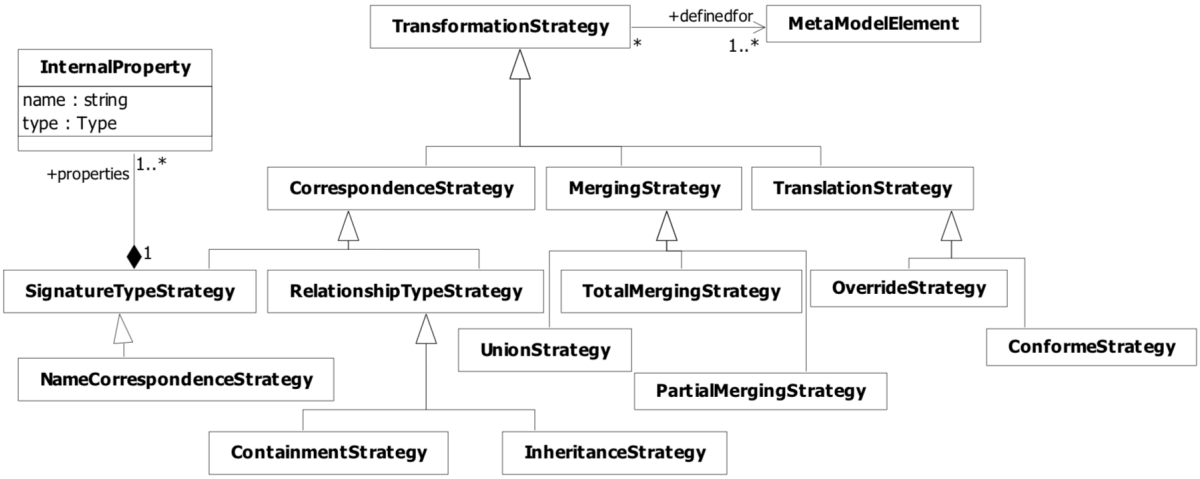


FIGURE 3.11 – Méta-modèle des stratégies de transformation

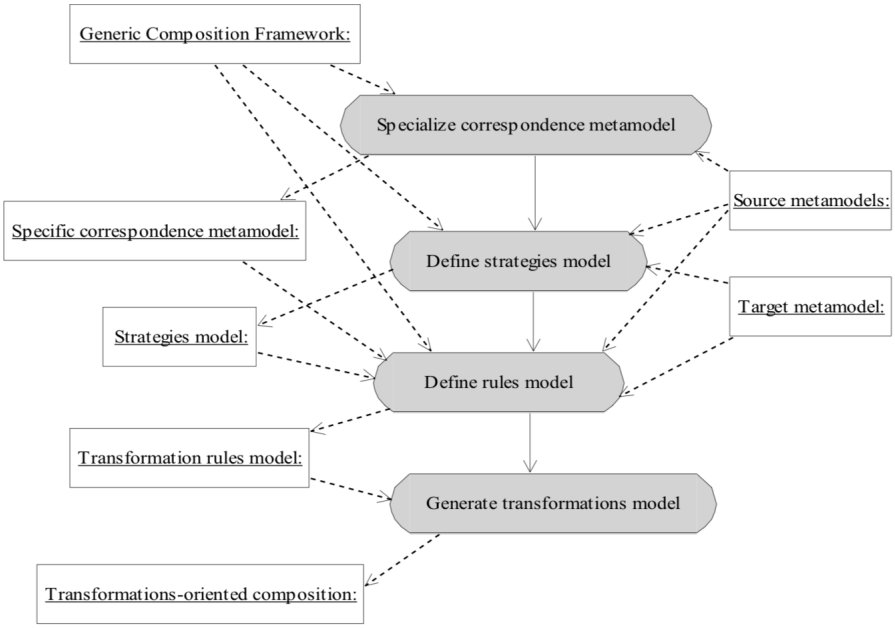


FIGURE 3.12 – Processus de spécialisation du cadre générique

3.4.1 Spécialisation du méta-modèle de correspondance

La première étape du processus de spécialisation consiste à étendre le méta-modèle de correspondance générique à un domaine d'application spécifique (Figure 3.13). Il est nécessaire d'établir différents types de relations entre les éléments du méta-modèle en fonction de leur sémantique, cette tâche n'est pas triviale, car elle nécessite une connaissance approfondie du domaine applicatif sous-jacent [70]. Par exemple, si nous considérons UML2 comme un langage source, pour exprimer la similarité entre les éléments de classe, nous définissons un nouveau type de relation appelé ClassSimilarityRelationship. L'ensemble des relations de correspondance spécifiques constitue le méta-modèle de correspondance spécifique.

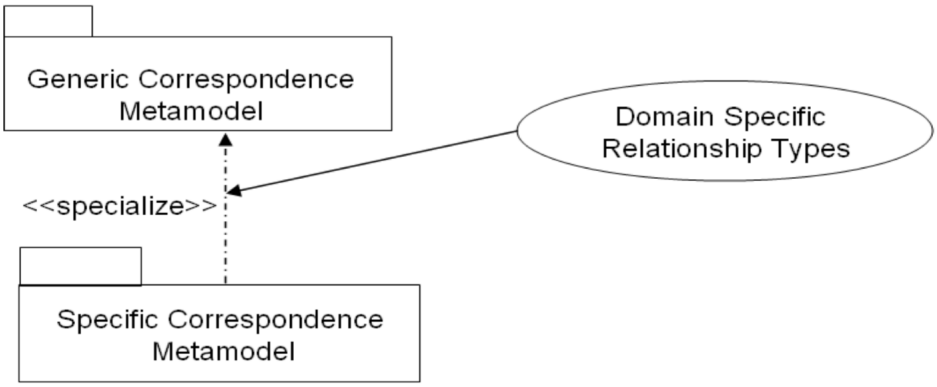


FIGURE 3.13 – Spécialisation du méta-modèle générique de correspondance

3.4.2 Définition du modèle de stratégies

Dans cette étape, l'objectif est de définir des stratégies spécifiques pour la composition des éléments.

Les stratégies de correspondance spécifient la sémantique de comparaison entre les éléments de modèle ; elles spécifient la sémantique des règles de correspondance spécifiques. Les stratégies de correspondance sont conçues par un modèle conforme au méta-modèle des stratégies de transformation (Figure 3.12). Comme pour les règles de correspondance, les règles spécifiques de fusion et de traduction doivent être complétées par des stratégies de transformation appropriées.

Les stratégies de fusion définissent comment les éléments liés par une relation de correspondance spécifique sont fusionnés pour créer des éléments du méta-modèle cible.

Les stratégies de traduction précisent comment les éléments qui n'ont pas d'éléments correspondants dans le modèle opposé sont transformés dans le modèle cible. Par défaut, ces éléments sont copiés dans le modèle cible. Cette stratégie de traduction par défaut peut être annulée afin de couvrir des cas plus spécifiques.

3.4.3 Définition du modèle de règles de transformation

Cette étape vise à définir un modèle de règles de transformation spécifiques conforme au méta-modèle des règles de transformation (Figure 3.11).

(i) Tout d'abord, la métaclasse CorrespondenceRule est instanciée. Cette instance a un nom, elle peut être spécifiée comme concrète ou abstraite, et elle est spécifiée par un ensemble de patterns source et cible. Les types de motifs source sont instanciés par des éléments des méta-modèles source et le type de motif cible est instancié par une relation de correspondance spécifique.

(ii) Puis la spécification de cette règle est complétée en spécifiant une stratégie de correspondance appropriée. Un ensemble de règles de fusion et de traduction en fonction du méta-modèle cible sont spécifiées ensuite. La métaclasse MergingRule définie dans le méta-modèle des règles de transformation est instanciée en spécifiant ses modèles source et cible. Dans ce cas, le modèle source a comme type une relation de correspondance spécifique.

(iii) Enfin, pour la métaclasse TranslationRule, il faut définir les éléments des méta-modèles source et cible associés aux patterns. Le modèle de transformation spécifique est ensuite

enrichi du modèle de stratégies de composition défini à l'étape précédente. Ceci permet l'intégration des capacités de composition spécifiées par ces stratégies.

Il en résulte un modèle de règles de transformation spécifique. Ce modèle peut être considéré comme une spécification de haut niveau des règles de transformation et peut être utilisé pour générer automatiquement des transformations exécutables pour la composition.

3.4.4 Génération de transformations orientées composition

La dernière étape du processus de spécialisation consiste à produire des transformations exécutables mettant en œuvre l'opération de composition. Ces transformations exécutables sont obtenues en transformant le modèle de règles de transformation spécifique. Cette tâche est effectuée par un type particulier de transformation appelé Transformation d'ordre supérieur (HOT)[70] car elle génère une transformation. Une transformation HOT prend en entrée un modèle de transformation et produit en sortie un modèle de transformation qui est conforme à un méta-modèle de langage de transformation (Figure 3.14). Plus précisément, les éléments du modèle de composition sont transformés en codes de composition spécifiques. Par exemple, si une règle de fusion utilise la stratégie TotalMergingStrategy, le modèle de code produit doit implémenter une fonctionnalité qui combine les éléments source en un seul élément de sortie.

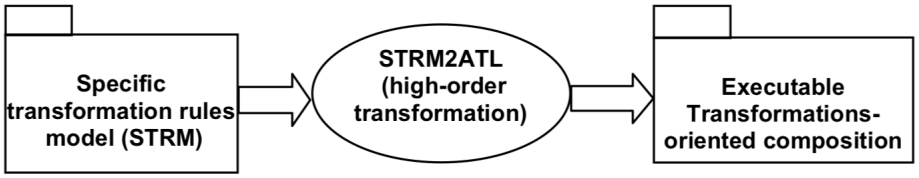


FIGURE 3.14 – Génération de transformations exécutables

3.5 Application à l'étude de cas du SGDMP

Pour illustrer l'approche, cette section décrit comment le cadre de composition générique peut être appliqué pour composer des modèles conformes au méta-modèle UML. Examinons l'exemple du SGDMP présenté dans la section 3.1.3). Dans cet exemple, nous fusionnons deux diagrammes de classes développés indépendamment selon deux points de vue. Le résultat de la composition est un modèle VUML conforme au profil VUML (figures 3.5 et 3.15).

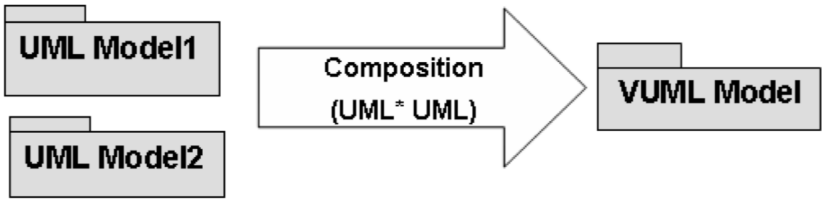


FIGURE 3.15 – Scénario de composition pour l'application SGDMP

3.5.1 Spécification des relations de correspondance

Pour définir des relations de correspondance spécifiques pour les éléments de diagramme de classes UML, nous avons spécialisé le méta-modèle de correspondance cgénérique selon le processus décrit ci-dessus. Par exemple, pour les éléments Classe, nous avons identifié deux relations de correspondance : ClassConformityRelationship et ClassSimilarityRelationship. Nous avons également défini des relations de correspondance pour d'autres éléments tels que les attributs, les opérations, etc.

3.5.2 Spécification des stratégies de transformation

Pour définir la composition des modèles UML produisant un modèle VUML, il faut d'abord définir des stratégies de correspondance. Ces stratégies précisent les règles de sémantique des correspondances. Une stratégie de correspondance par défaut appliquée aux éléments dont la métaclasse est définie comme une sous-classe de NamedElement est basée sur le nom de la propriété. Cette opération dépend fortement du langage de modélisation. À cette fin, pour chaque scénario d'application, une stratégie de correspondance spécifique doit être définie. Dans notre application, nous avons défini deux stratégies de correspondance entre les éléments Classe : conformité et similarité. La conformité est assurée lorsque deux classes apparaissent dans deux modèles de points de vue ayant le même nom et les mêmes propriétés (attributs, opérations, associations) ; elles sont sémantiquement équivalentes (représentent deux vues du même concept dans la terminologie VUML). Il y a similarité lorsque deux classes apparaissent sous le même nom mais ne sont pas conformes.

De même, les stratégies de fusion définissent la sémantique des éléments de fusion en fonction de la relation de correspondance qui les relie. Par exemple, nous avons défini la stratégie TotalMergingStrategy comme une stratégie de fusion de deux classes liées par une ClassConformityRelationship. Elle décrit comment fusionner des classes sources afin de créer une classe cible. Alors que la stratégie de fusion de classes similaires spécifie la création d'une classe multi-vues (base et vues).

3.5.3 Spécification des règles de transformation

Nous avons défini un ensemble de règles de transformation pour composer des diagrammes de classes UML en un modèle de classes VUML. Les règles de transformation sont réutilisées à partir du framework générique. Selon le processus de spécialisation décrit précédemment, nous distinguons les règles de correspondance, qui s'appliquent aux éléments des modèles sources et qui créent des relations spécifiques, des règles de fusion d'éléments correspondants ou des traduction d'éléments. En effet, toutes les règles de la première catégorie sont dérivées des éléments du méta-modèle UML. Les règles de transformation de la seconde catégorie sont dérivées des méta-modèles source et cible ; elles spécialisent le cadre générique pour définir un opérateur de fusion spécifique pour les modèles structurels UML. La Figure 3.16 montre une hiérarchie des règles de correspondance définies pour cet exemple. Certaines règles sont définies comme des sous-règles de la règle ModelElementCorrespondenceRule. Cela permet de factoriser une partie du code commun, offrant plus de réutilisabilité et de flexibilité à l'opérateur de correspondance.

3.5.4 Mise en œuvre en ATL

Pour mettre en œuvre et valider notre approche, nous avions besoin d'un langage de transformation basé sur des règles. Nous voulions nous concentrer sur des approches déclaratives permettant de définir des transformations à un haut niveau d'abstraction sans nous soucier de

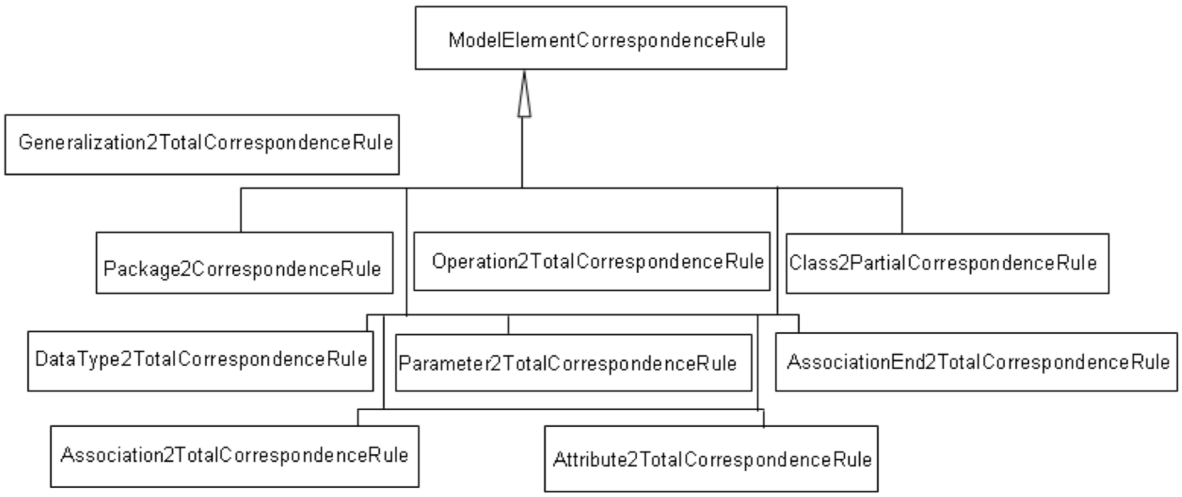


FIGURE 3.16 – Règles de correspondance pour les éléments de diagramme de classes UML

l'exécution finale et de l'application des règles. La gestion de l'application des règles peut être déléguée à un moteur de règles, mais il est souvent nécessaire d'aider ce moteur en introduisant un code impératif en fonction de la complexité de la transformation. Pour ce faire, nous avons choisi ATL [122], langage de transformation hybride permettant de combiner les approches déclarative et impérative. ATL est un composant standard d'Eclipse pour la transformation de modèles, intégré au projet M2M [76].

Le listing3.1 illustre un exemple en ATL de règles de transformation utilisées pour la composition VUML. ' (Il s'agit d'un sous-ensemble des règles implantées). Notons que la composition a été implémentée avec deux modules de transformation (lignes 1-2) et (lignes 10-12) : le premier module implémente des règles de correspondance et génère un modèle de correspondances, tandis que le second implémente des règles de fusion et de traduction et produit le modèle cible VUML. La règle Class2PartialCorrespondence (Lignes 3-9) stipule que deux éléments de classe seront liés par une relation de correspondance partielle s'ils sont définis dans deux modèles différents avec le même nom. Cette règle est déclarée comme une spécialisation (mot-clé extend) de ModelElementCorrespondenceRule qui est définie comme une règle abstraite. Le mécanisme d'héritage permet de factoriser le code commun entre plusieurs règles de transformation. La règle PartialCorrespondence2Base (Lignes 13-37) spécifie que pour chaque relation PartialCorrespondenceRelationship définie qui relie deux classes, une classe d'éléments UML2 est créée dans le modèle composé. Cette règle met en œuvre la stratégie TotalMerging-Strategy. Enfin, la règle Class2Class (Lignes 38-46) implémente la stratégie de traduction par défaut qui exprime qu'une classe n'ayant pas de classe correspondante dans la classe opposée est traduite (copiée) telle quelle dans le modèle cible.

```
1 module UML2Corresp;
2 create MC : MMC from MPV1 : UML2, MPV2 : UML2;
3 rule Class2PartialCorrespondence extends
4 ModelElementCorrespondenceRule{
5 from e1 : UML2!Class, e2 : UML2!Class(e1.name = e2.name)
6 to r : MMC! PartialCorrespondenceR (
7       correspondingElementType <-'Class'
8       )
9   }


10 module CorrespUML2VUML;
11 create VUML : UML2 from MPV1 : UML2, MPV2 : UML2, PRO :
12 UML2, MC : MMC;
13rule PartialCorrespondence2Base{-- specefic to VUML
14 from r : MMC! PartialCorrespondenceR
15 to c : UML2!Class(
16 name <- thisModule.getRefElement(r).name,
17 visibility <-
18 thisModule.getRefElement(r).getElement.visibility,
19 isAbstract <-
20 thisModule.getRefElement(r).getElement.isAbstract
21 )
22 do{
23 thisModule.VUMLModel.packagedElement<-c;
24 c.applyStereotype(thisModule.base);
25 for (iterator in r.getListeAttrEquality){
26 c.ownedAttribute <-
27 thisModule.AttributeEquality2Attribute(iterator);
28 }
29 for (iterator in r.getListeOpEquivalence){
30 c.ownedOperation <-
31 thisModule.OperationEquivalence2Operation(iterator);
32 }
33 if (not r.general.oclIsUndefined()){
34 c.generalization <-
35 thisModule.resolveTemp(r.general,'g');
36 }
37 }}
38 rule Class2Class{
39 from c1: UML2!Class(c1.isNotMultiviewClass)
40 to c2: UML2!Class(
41 name <- c1.name,
42 isAbstract <- c1.isAbstract,
43 visibility <- c1.visibility,
44 ownedAttribute<-c1.ownedAttribute,
45 ownedOperation<-c1.ownedOperation
46 )}
```

Listing 3.1 – Régles de transformation ATL UML2VUML

3.6 Conclusion

Le travail réalisé dans le cadre de la thèse de Adil Anwar a permis de formaliser l'opération de composition de modèles de conception réalisés selon une modélisation orientée points de vue. Notre travail s'est focalisé sur l'opération de composition de modèles de conception issus d'une phase de conception décentralisée.

Sur le plan méthodologique, nous avons proposé un processus de composition de modèles. Ce processus est composé de trois phases : la première phase permet de résoudre les conflits (syntaxiques, sémantiques, structuraux, etc.). Nous avons élaboré une méthode de résolution semi-automatique ou interactive en identifiant pour chaque type de conflit une règle de résolution.

La deuxième phase est une phase de composition automatique qui permet de produire le modèle VUML en s'appuyant sur un ensemble de règles de transformation.

La troisième phase est une phase de vérification de certaines propriétés du modèle composé. Nous pouvons par exemple analyser la cohérence statique du modèle VUML en vérifiant le respect des règles de bonne formation (W.F.R).

La phase de composition sur laquelle a porté plus particulièrement cette thèse, s'appuie sur deux artefacts principaux : un modèle de correspondance – qui joue le rôle de produit intermédiaire dans le processus – et des règles de transformation. Le modèle de correspondance permet de stocker les différentes relations de correspondances établies entre les éléments des modèles d'entrée. Les règles de transformation sont déclinées en règles de correspondance, de fusion et de translation. Ces règles permettent d'abord de mettre en correspondance les modèles en entrée, puis de les fusionner afin de produire un modèle global VUML.

L'utilisation du langage ATL (Atlas Transformation Language) nous a permis de coder la plupart des règles sous une forme déclarative. L'ordre d'exécution de ce type de règles est implicitement spécifié par le développeur et pris en charge par le moteur d'exécution du langage.

Nous avons appliqué ces résultats au profil VUML, en prenant comme cas d'étude un système de gestion de dossiers médicaux partagés.

La phase de validation de cette approche a donné lieu à un prototype de composition de diagrammes UML en un diagramme de classe VUML, qui intègre également des opérations telles que la vérification du modèle VUML généré et la génération de code Java.

Ainsi, nous avons atteint l'objectif fixé de :

Favoriser la conception d'un modèle global tout en continuant à produire des modèles partiels adaptés à des points de vue variés.

Et relevé les premiers défis que nous avons identifiés dans le cadre de cette étude :

Fournir aux utilisateurs les moyens d'obtenir un modèle global du système tout en conservant la production de modèles partiels

Fournir aux utilisateurs une démarche leur permettant de produire des modèles partiels indépendants dans un même standard et à générer un modèle global commun.

Et plus précisément :

Fournir une démarche de composition des modèles UML en un modèle VUML
Fournir un opérateur de composition des modèles UML en un modèle VUML
Fournir des moyens automatiques de vérification de la validité du modèle obtenu

Avec la composition de modèles proposée dans VUML, nous répondons aux questions :

Question 1 : Comment garantir la validité des modèles de conception dans leur ensemble ?

Question 2 : Comment garantir le cohérence des modèles de conception dans leur ensemble ?

3.7 Discussion

Les travaux présentés dans cette section ont bien sur des limites et soulèvent certaines questions. Dans notre analyse des systèmes basée sur les vues, des conflits entre les modèles de points de vue peuvent apparaître en raison de conceptions distinctes. Nous avons identifié de telles incohérences qui peuvent être syntaxiques (p. ex. conflits d'homonymie, conflits structurels) ou sémantiques (p. ex. conflits de synonymie). La gestion de ces conflits n'est pas faite à l'étape de la composition et n'a pas été pris en compte dans cette étude.

Jusqu'à présent, la comparaison des éléments du modèle source est basée sur les propriétés syntaxiques définies dans le méta-modèle source. Par exemple, deux opérations ayant la même signature (nom, paramètres et type de retour) sont considérées comme équivalentes ce qui n'est évidemment pas toujours le cas. Pour résoudre ce problème, on peut soit effectuer une étape de réconciliation semi-automatique entre les concepteurs, soit renforcer la sémantique (en particulier les aspects comportementaux) associée au méta-modèle source afin d'élaborer des stratégies de comparaison plus fines.

Pour valider notre travail, nous avons choisi ATL afin de bénéficier d'une approche semi-déclarative. Ce choix semble correct pour une étude de cas mais peut présenter des limites d'extensibilité avec de grands modèles et de nombreuses règles. Des langages impératifs basés sur la transformation tels que le kermeta pourraient êttre expérimentés.

Le modèle VUML résultant du processus de composition peut être utilisé de plusieurs façons. Une méthode classique consiste à produire du code objet via un générateur de code. Nous avons ainsi développé un générateur de code générique et une première instance de ce générateur ciblant Java [65]. D'autres langages tels que C++ ou Eiffel pourraient être supportés. De plus, si l'on a besoin d'étendre un modèle donné en VUML afin d'intégrer un nouveau point de vue, le modèle VUML peut être composé à son tour avec un nouveau modèle de point de vue produit séparément. Pour ce faire, on peut appliquer notre approche en remplaçant le méta-modèle UML source par le méta-modèle VUML

Enfin, notons que l'activité de tissage du processus de spécialisation pourrait être automatisée. Pour ce faire, il faudrait appliquer la technique HOT pour générer les règles de transformation exécutables dans un langage de transformation spécifique. Par conséquent, l'étape de programmation manuelle de la transformation serait considérablement réduite.

Néanmoins, ce qui nous a paru le plus important à considérer en perspective de ces travaux est lié à l'évolution des paradigmes de modélisation des systèmes complexes et aux progrès faits dans l'Ingénierie des Modèles qui a amené non plus à considérer un langage de modélisation unique, mais des langages de modélisation différents, liés aux métiers : les Domain Specific Language. Force a été de constater que les modèles à composer ne devaient plus être considérés comme homogènes, mais hétérogènes. C'est le problème que nous avons tenté de résoudre dans le cadre de la thèse de Mahmoud El Hamlaoui que je présenterai dans le chapitre suivant.

Chapitre 4

Alignement de modèles de conception hétérogènes

Cette section reprend les éléments exposés dans *Handling Heterogeneous models Matching and Consistency via MDE* exposé lors de ENASE 2018, Funchal, Madeira, Portugal, March 23-24, 2018, Revised Selected Papers, pp. 288-313 [100] et *Alignment of viewpoint heterogeneous design models : Emergency Department Case Study* dans International Workshop On the Globalization of Modeling Languages (GEMOC) co-located with MODELS 2016 [101].
Cette section résume les résultats obtenus dans le cadre de la thèse de Mahmoud El Hamlaoui, "Mise en correspondance et gestion de la cohérence de modèles hétérogènes évolutifs", soutenue le 18 septembre 2015 à Toulouse [99].

Je présenterai dans ce chapitre la proposition que nous avons faite dans le cadre de la thèse de Mahmoud El Hamlaoui pour répondre au problème de l'alignement de modèles hétérogènes évolutifs. Elle est illustrée par une étude de cas réelle, qui a permis de valider l'approche : un système de gestion d'un Service d'Urgences.

Je donnerai d'abord une vue d'ensemble de l'approche d'alignement dont l'objectif est la construction d'un modèle, appelé modèle de correspondance, permettant de connecter des modèles partiels fournis en entrée. Je décrirai ensuite en détail l'approche de gestion de la cohérence, et de synchronisation des changements qui peuvent intervenir au sein des modèles connectés.

4.1 Contexte

4.1.1 Des modèles partiels hétérogènes évolutifs

La conception de systèmes complexes fait appel à un ensemble varié d'experts en modélisation provenant de différents secteurs d'activité. Ces concepteurs peuvent être situés dans des zones géographiques éloignées, comme c'est le cas pour le développement collaboratif distribué dans les grandes entreprises de logiciels. Plusieurs approches ont été développées pour faire face à la modélisation de systèmes complexes. La plus utilisée est l'approche multi-modélisation [38]. Elle consiste à élaborer des modèles partiels distincts qui correspondent à différentes vues métier du système [40]. Cette approche aide les concepteurs à se concentrer isolément sur les différentes parties (modèles partiels) du système.

Cependant, à un moment donné, il est indispensable de construire un modèle global pour comprendre et exploiter efficacement l'ensemble du système [103].

Par ailleurs, ces modèles ne sont plus écrits dans une norme de modélisation, mais dans des DSL distincts, adaptés aux métiers et sont donc hétérogènes.

Nous utilisons donc des modèles partiels, par point de vue, hétérogènes, que nous composons pour réduire les risques d'incohérence et faciliter la maintenance de l'ensemble. En effet, ces modèles sont susceptibles d'évoluer, pendant et après leur conception, de nouveau modèles peuvent être produits, etc ... Le modèle global peut donc être régulièrement remis en question. Il faut noter que notre approche ne concerne pas les changements internes aux modèles ; il appartient aux concepteurs des modèles partiels de gérer les répercussions internes des changements apportés sur leurs modèles et d'assurer leur validité.

4.1.2 Etude de cas : un Service d'Urgence

Les Services d'Urgence (SU) représentent une branche essentielle du système de santé de tout pays. Ces services sont généralement confrontés à des situations d'urgence (accidents, catastrophes naturelles, attentats terroristes, guerres, épidémies, etc.) qui nécessitent des compétences particulières grâce à une approche multidisciplinaire où les points de vue sont complémentaires. Il faut donc tenir compte d'un besoin de coordination entre les parties prenantes dans la phase de conception d'un tel système, afin que les différents modèles développés dans cette phase puissent être synchronisés. La gestion non optimale des urgences, que l'on constate assez souvent, provient en partie d'une prise en compte insuffisante de ces facteurs au moment de la conception. En outre, les modèles peuvent évoluer parce que les lois, les règlements, les règles commerciales, les procédures d'exploitation, les contraintes de sécurité et la protection des données personnelles, etc. peuvent changer. Dans cette étude, nous nous situons spécifiquement dans le contexte de l'urgence de l'hôpital public de Montpellier.

De nombreux acteurs participent au bon fonctionnement d'un service d'urgence, qu'il s'agisse d'infirmières auxiliaires ou de médecins urgentistes (chirurgiens). Par souci de simplicité, nous avons limité notre étude de cas à trois domaines d'activité gérés séparément par les acteurs suivants :

- Concepteur de rapports médicaux : responsable de la construction de maquettes numériques qui définissent un rapport d'examen d'urgence (EER). Il crée un modèle conforme à un méta-modèle de forme,
- Concepteur de logiciels : responsable de la représentation des données organisationnelles du système d'information à travers un méta-modèle orienté objet.
- Concepteur de processus : Il définit les protocoles médicaux à appliquer par le personnel. Il crée un modèle qui s'exprime à travers un méta-modèle basé sur les processus.

Dans ce qui suit, nous présentons les exigences identifiées pour l'élaboration des différents modèles concernés et leurs méta-modèles respectifs.

Modèle organisationnel

Le modèle organisationnel (un extrait est présenté sur la Figure 4.1) représente l'organisation d'un service d'urgence. Un SU est décrit par son nom, son adresse et son personnel (personnel administratif, personnel technique, infirmier, médecin urgentiste et chirurgien), ses patients et ses salles d'examen. Pour chaque membre du personnel, il est possible de connaître ses certifications et compétences, ses langues parlées, sa date d'embauche et son planning. Le personnel technique entretient et commande les machines. Le personnel administratif réserve

les salles, perçoit les paiements, etc. Un médecin urgentiste conseille et traite ses patients. Il pose des diagnostics et effectue des actes, éventuellement chirurgicaux, qui sont répertoriés.

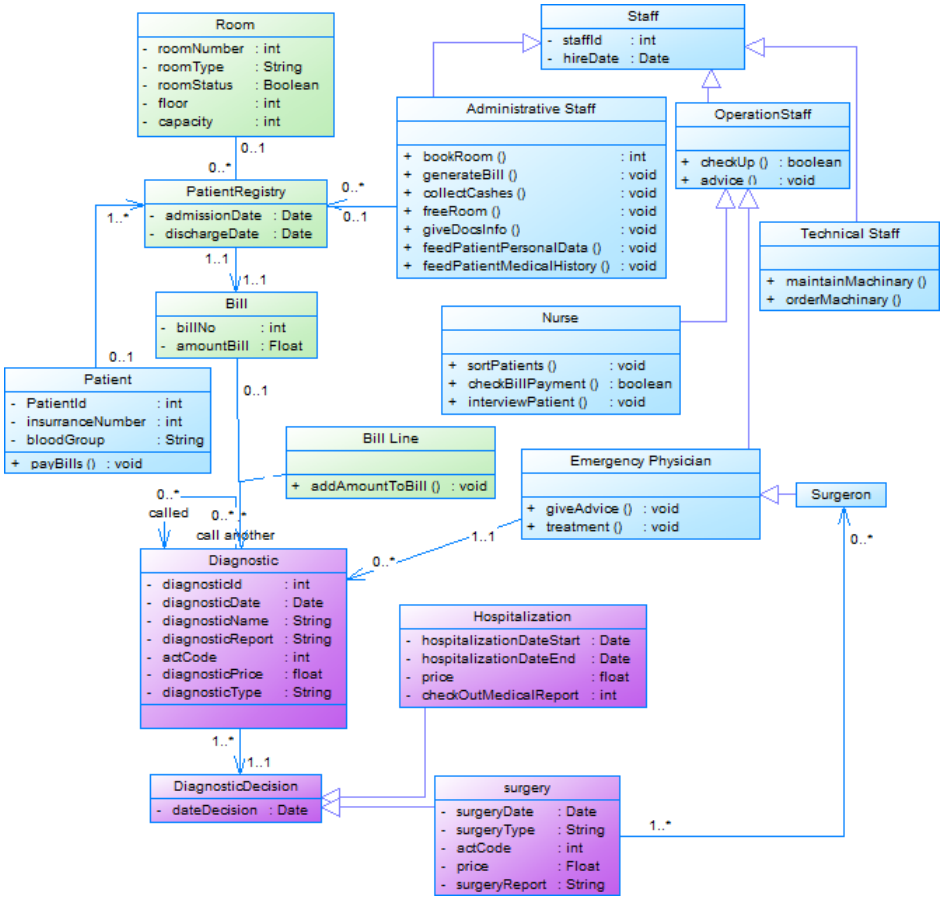


FIGURE 4.1 – Modèle organisationnel (Organizational M)

Le modèle organisationnel est conforme au méta-modèle illustré à la Figure 4.2.

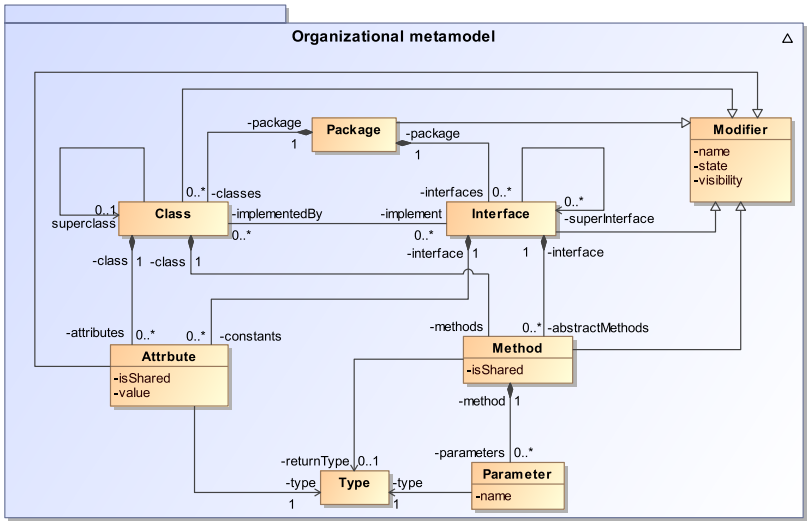


FIGURE 4.2 – Meta-modèle organisationnel (Organizational MM)

Ce dernier représente une version simplifiée d'un méta-modèle de conception sous une

forme orientée objet. Le concept de base est le *Package*, qui contient les classes et les interfaces. Celles-ci sont composées d'attributs et de méthodes.

Modèle des protocoles médicaux

Le modèle des protocoles médicaux (extrait de la Figure 4.3) vise à décrire les différents protocoles à appliquer par chaque catégorie de personnel. Tout d'abord, une infirmière reçoit chaque patient et l'oriente vers un service approprié (médical ou chirurgical). Ensuite, un assistant sanitaire installe le patient dans une chambre. Par la suite, une entrevue protocolaire est effectuée par une infirmière afin de recueillir des données administratives et médicales. Un médecin urgentiste effectue une consultation et fait une ordonnance. L'infirmière retourne auprès du patient et exécute la prescription du médecin.

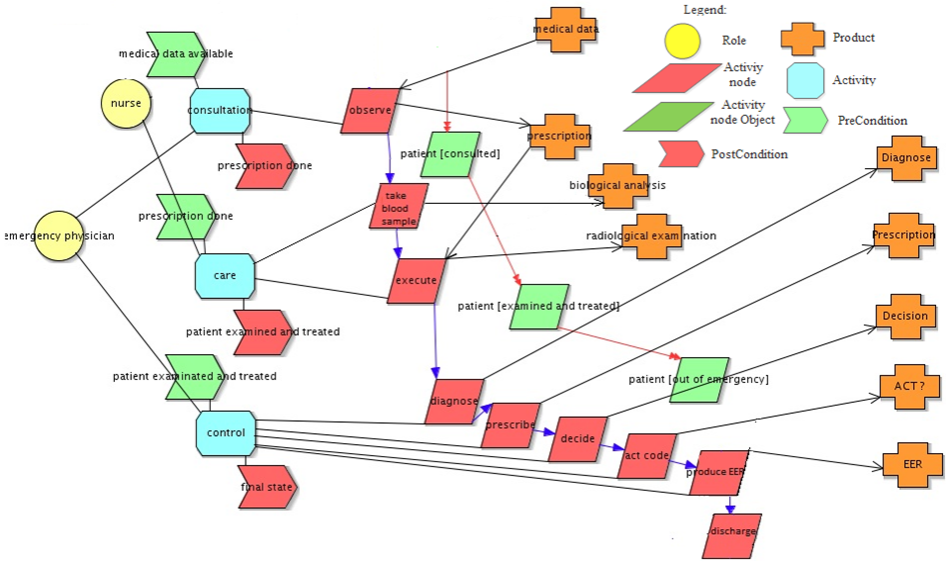


FIGURE 4.3 – Modèle des protocoles médicaux (Process M)

Le méta-modèle représenté dans la Figure 4.4 permet de construire le modèle des protocoles médicaux.

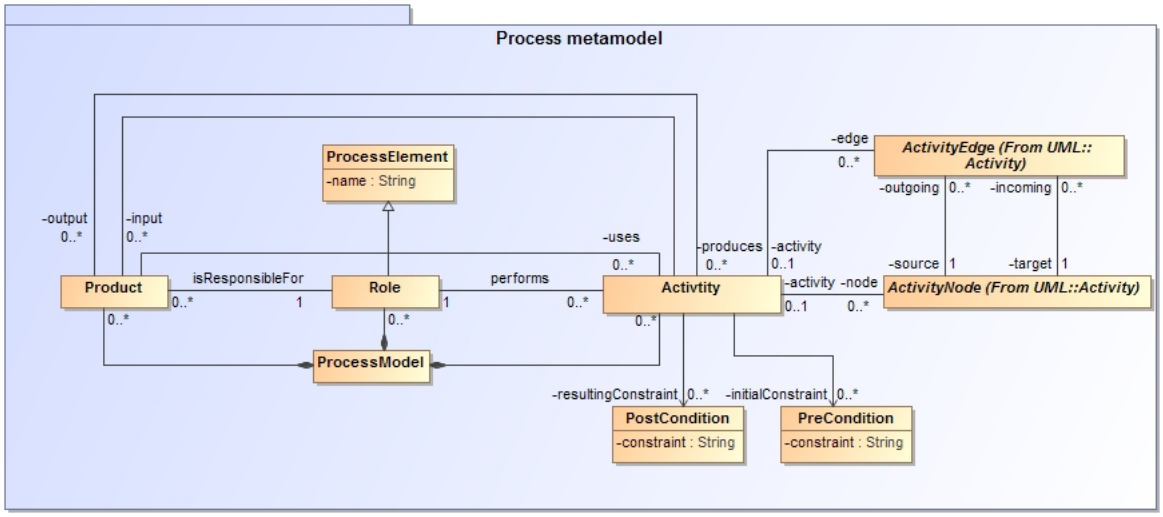


FIGURE 4.4 – Meta-modèle des protocoles médicaux (Process MM)

Il définit un concept appelé "ProcessModel" qui inclut les produits, les rôles et les activités, ainsi que les relations entre eux. Les activités sont reliées les unes aux autres par le concept abstrait ActivityEdge. Une activité est également liée au concept abstrait ActivityNode.

Modèle de rapport d'examen d'urgence (REU)

Un modèle REU (Figure 4.5) représente l'information qui sera produite par un médecin urgentiste pour un patient donné. D'une manière générale, il contient l'identifiant du médecin, des informations sur le patient (numéro de sécurité sociale, prénom, nom, prénom et âge), la date d'arrivée du patient et son observation clinique. Ce dernier est réalisé à l'aide d'abréviations codifiées et d'explications associées. Il identifie la pathologie afin d'orienter le patient vers le service approprié au sein du même établissement ou d'un autre.

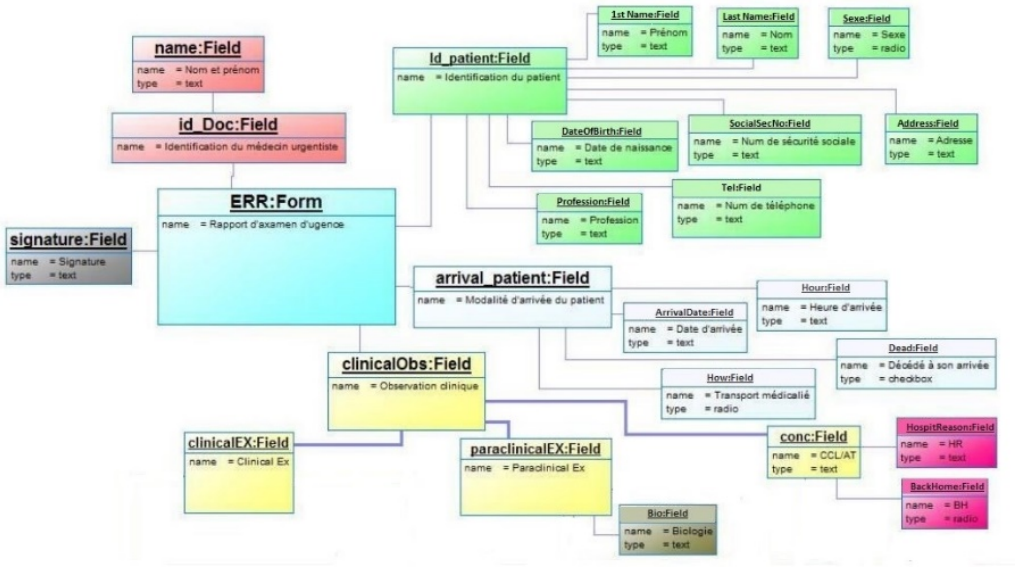


FIGURE 4.5 – Modèle des formulaires (Form M)

Le méta-modèle proposé dans la Figure 4.6 permet la représentation d'un formulaire avec un ensemble de champs éventuellement composites.

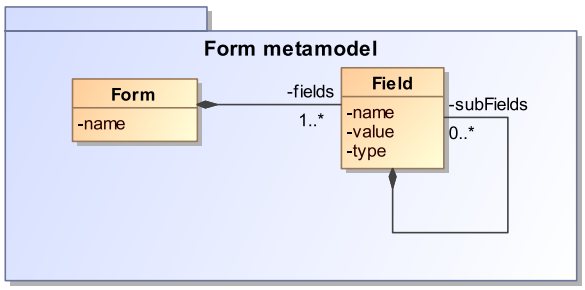


FIGURE 4.6 – Meta-modèle des formulaires (Form MM)

4.2 AHM : Approche d'alignement de modèles de conception ("Models Matching")

L'appariement de modèles consiste à établir les correspondances qui existent entre leurs éléments respectifs ; on parlera aussi de mise en correspondance, ou d'alignement, par analogie avec l'alignement d'ontologies [180]. Nous avons proposé pour cela d'analyser les modèles d'entrée (et leurs méta-modèles respectifs) afin d'identifier les correspondances qui existent entre leurs éléments. Les correspondances entre éléments de modèles mises en évidence sont stockées dans un modèle de correspondance, appelé M1C. Ce modèle, mettant en correspondance ses différents modèles partiels, constitue donc une vision globale du système.

Pour rappel, nous ne considérons que les correspondances inter-modèles. Ainsi, les correspondances entre les éléments d'un même modèle partiel ne font pas l'objet de notre étude.

La section suivante présente l'élaboration de M1C ainsi que le processus d'alignement itératif proposé.

4.2.1 Méta-modèle de correspondance

Pour identifier les correspondances entre les éléments des modèles source, nous nous fondons sur les correspondances existant entre les éléments de leurs méta-modèles respectifs. Ceci permet de définir une correspondance une seule fois au niveau méta-modèle (M2) et de la réutiliser pour les éléments des différents modèles (de niveau M1), eux-mêmes instances des méta-modèles. Les correspondances établies au niveau des méta-modèles servent donc de guide lors de l'établissement des correspondances entre les modèles. Les correspondances sont des éléments complexes, car elles contiennent les éléments de modèles liés, mais aussi la relation qui les relie, le type de cette relation, sa sémantique, etc. Notons que les correspondances que nous manipulons sont n-aires.

Pour exprimer les correspondances, nous avons défini un méta-modèle de correspondance. Le méta-modèle de correspondance représenté sur la Figure 4.7 identifie les différents concepts à partir desquels le modèle de correspondance est créé.

Dans le but de ne pas créer de modèles de correspondances trop lourds à gérer, nous avons opté pour une virtualisation du modèle de correspondance, selon [58]. C'est en effet le mécanisme de virtualisation qui est utilisé pour accéder aux éléments de modèles des différentes correspondances [104]. Ainsi, les éléments des modèles partiels ne sont pas copiés dans le modèle global mais référencés. Tout élément est repéré par sa référence et c'est celle-ci qui est manipulée dans le modèle de correspondance. Le modèle de correspondance obtenu est donc un modèle global virtuel.

Le méta-modèle introduit la notion de modèle de correspondance (CorrespondenceModel), qui contient les correspondances établies entre des (méta)éléments de (méta)modèles distincts à travers leurs références.

La métaclasse *CorrespondenceModel* contient des correspondances établies entre au moins deux (méta)éléments (*RefElement*) de différents (méta)modèles par leurs *références*. Nous stockons les références sous forme de chaîne de caractères pour encapsuler les informations sur leur modèle source et leur méta-modèle.

La métaclasse *correspondance* est composée d'au moins deux éléments référencés et de la relation (*Relationship*) qui les relie.

Un des intérêts de l'approche proposée est qu'elle permet 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*, permettent notamment ces connections multiples.

L'attribut *bidirectional* spécifie si la relation est bidirectionnelle. Dans ce cas, les (méta)éléments concernés sont tous des éléments sources et il n'y a pas de (méta)élément de type cible. (Pre-

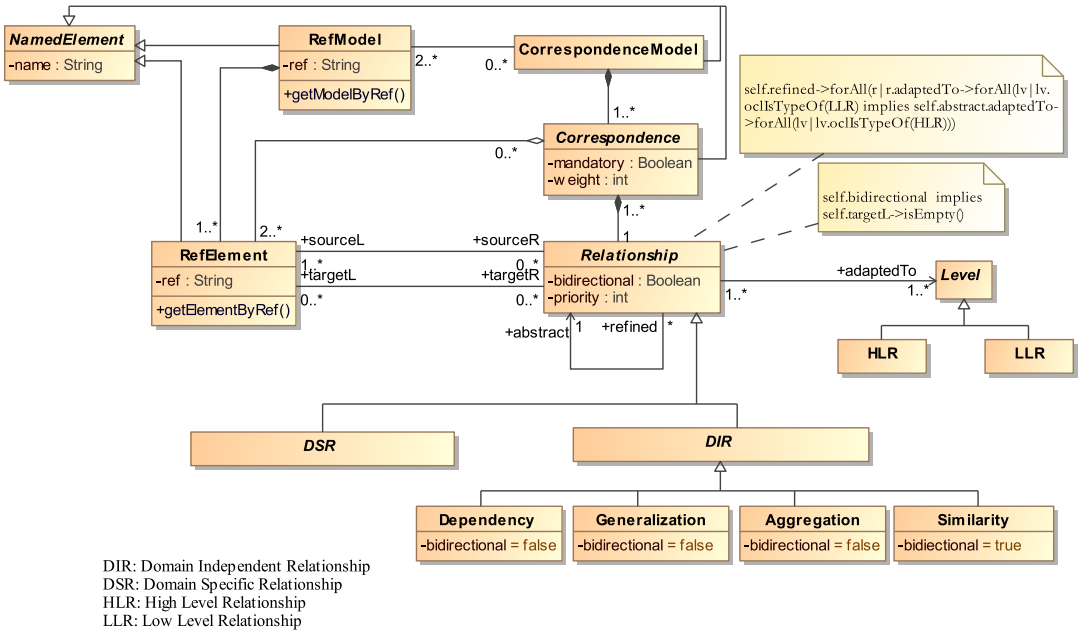


FIGURE 4.7 – Noyau du méta-modèle de correspondance, MMC

nons l'exemple d'une égalité, tous les éléments reliés sont égaux, la relation qui les lie n'est pas dirigée, les éléments font partie d'un même n-uplet, sans relation d'ordre ou d'inférence, ils sont tous source d'une même relation). Ceci est spécifié par la règle OCL suivante : :*self.bidirectional implies self.target → isEmpty ()*.

La métaclasse *Correspondence* possède trois attributs dédiés à la gestion de la cohérence que nous verrons dans la section 4.3. L'attribut *mandatory* indique si une correspondance est obligatoire pour le système étudié, *weight* représente le coefficient de pondération associé à chaque correspondance et *priority* accorde une valeur de priorité à chaque type de relation.

Pour représenter les différents types de relations pouvant exister entre des éléments de modèles, nous proposons de prendre en compte la diversité des relations dans le méta-modèle.

Nous définissons pour cela deux types de relations spécialisables. *Relationship* est une généralisation abstraite de *DIR* (Domain Independant Relationship) qui modélise les relations indépendantes du domaine et de *DSR* (Domain Specific Relationship), les relations spécifiques du domaine. La spécialisation de DIR permet de représenter des relations génériques et indépendantes du domaine d'application. Cependant, ces relations peuvent être insuffisantes pour un domaine donné. Dans ce cas, il est possible d'ajouter des relations spécifiques par une spécialisation de la métaclasse DSR.

La métaclasse abstraite *Level* est associée à une relation et décrit si celle-ci représente une relation dite de haut niveau (HLR au niveau M2) ou une relation de bas niveau (LLR au niveau M1) ou les deux. Ainsi, les *HLR* permettent de représenter les relations qui sont utilisées dans les correspondances inter méta-modèles, tandis que *LLR* représentent les relations qui sont utilisées dans les correspondances inter modèles.

La règle OCL : *self.refined → forAll (r | r.r.adaptedTo → forAll (lv | lv | lv.oclIsTypeOf (LLR) implies self.abstract.adaptedTo → forAll (lv | lv.oclIsTypeOf (HLR)))*

précise que toute relation utilisable au niveau M2 [1] est utilisable au niveau M1. Le contraire

1. Nous notons ici les niveaux de métamodélisation selon la pyramide de modélisation de l'OMG [161], où M0 dénote le monde réel, M1 le niveau modèle et M2 le niveau des méta-modèles (M3 étant le MOF)

n'est pas vrai.

D'autres annotations contiennent des expressions sémantiques qui sont associées aux relations. Nous verrons plus tard comment elles sont générées et exploitées dans le processus d'alignement des modèles (section 4.2.5).

4.2.2 Processus d'alignement de modèles

Certaines des correspondances identifiées entre les 3 modèles du SU apparaissent sur la Figure 4.8 ci-dessous.

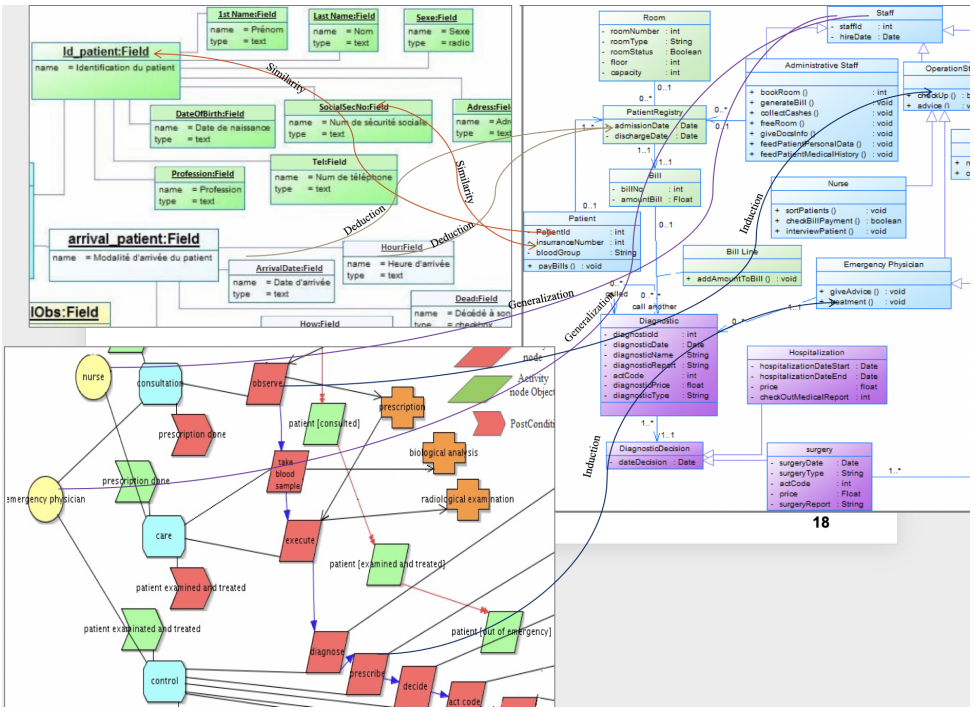


FIGURE 4.8 – Extrait de correspondances entre les 3 modèles du SU

Le modèle de correspondance qui les contient ne peut pas être construit de manière linéaire. Il suit un processus que nous appelons processus d'alignement, que nous avons formalisé. La Figure 4.9 montre une itération du processus d'alignement proposé. Elle décrit les étapes nécessaires à la réalisation de l'appariement entre modèles partiels hétérogènes, afin d'obtenir un modèle de correspondance.

Le processus implique deux parties prenantes, un intégrateur expert qui est le superviseur du domaine d'application, et un outil qui assiste le superviseur et met en œuvre les traitements automatiques du processus.

Le processus prend en entrée les différents méta-modèles des modèles à aligner et le noyau générique du méta-modèle de correspondance (MMC). Par la suite, le superviseur vérifie si le MMC contient toutes les relations nécessaires pour établir des correspondances entre les (méta)modèles partiels (étape 1). S'il suppose que les relations proposées ne sont pas suffisantes pour décrire le domaine d'application donné, il spécialise la métaclasse DSR de MMC, en rajoutant les relations qu'il estime nécessaires (étape 2).

Ainsi, pour une description correcte du SU, trois relations spécifiques au Domaine ont été rajoutées par un expert du domaine. *Deduction*, *Requirement* et *Induction*. La première permet de déduire la valeur d'un élément à partir de celle d'un autre élément. Le deuxième permet d'identifier les champs dont une tâche a besoin pour se dérouler correctement. La troisième permet de représenter les opérations qu'une tâche met en œuvre lors de son exécution. (Nous

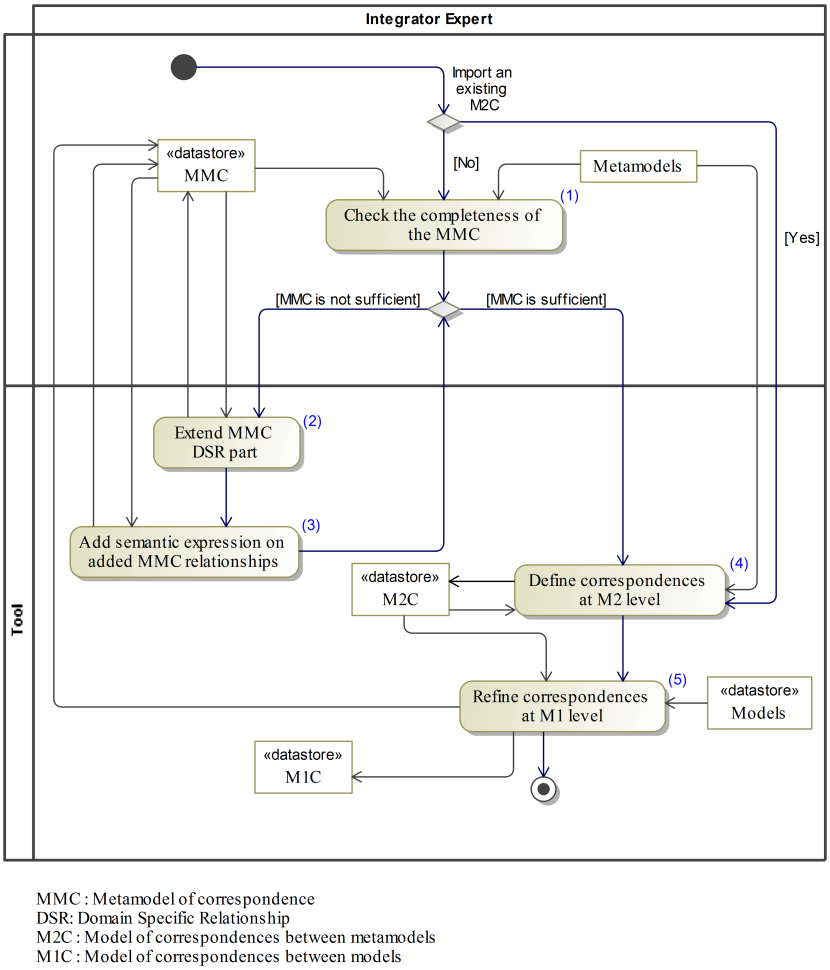


FIGURE 4.9 – Sous-processus d'appariement

verrons plus tard section 4.2.5 qu'une sémantique opérationnelle est attribuée par l'expert du domaine à chaque type de relation, lors de l'étape 3 du processus d'alignement).
Une fois que le MMC est spécialisé, l'opération d'alignement peut commencer. L'expert identifie les correspondances entre les méta-éléments afin de produire le modèle de correspondance des méta-modèles impliqués (étape 4). Ce modèle de correspondance de niveau M2 est appelé M2C. Il stocke les correspondances de haut niveau (les HLC) entre méta-éléments.

La Figure 4.10 résume les HLC définies pour le SU.

A titre d'exemple de correspondances notons sur cette figure la DSR *Induction*, qui relie le méta-élément *Activity* du méta-modèle des protocoles médicaux *Process MM* au méta-élément *Operation* du méta-méta-modèle organisationnel *Organizationnal MM*, (signifiant qu'une opération peut potentiellement contribuer à la réalisation d'une tâche). Une DIR, *Similarity*, est également définie entre les méta-éléments *Field* du méta-modèle *Form MM* et *Attribute* du méta-modèle *Organizationnal MM*.

L'étape suivante (étape 5) consiste à produire les correspondances inter-modèles (les LLC). Pour cela, on va procéder à un raffinage des HLC. L'outil support développe semi-automatiquement les produits cartésiens des HLC du M2C et effectue un raffinage, décomposé en une opération de reproduction, suivie d'une opération de sélection, au niveau M1, pour produire le M1C.

On définit le raffinage par :une relation Rf, qui a une HLC du M2C associe toutes les LLC de niveau M1 qui en sont déduites. Cy Rf Cx, avec Cx et Cy deux correspondances, si et seulement si Cy est définie à partir de Cx. On pose comme postulat que Rf est non réflexive et non

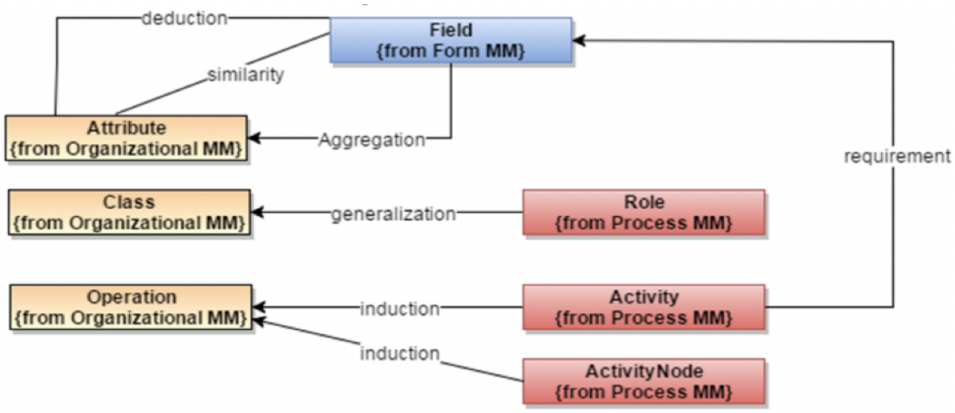


FIGURE 4.10 – Modèle M2C du SU

symétrique. La première propriété indique qu'une HLC ne raffine pas une autre HLC alors que la seconde exprime le fait qu'une LLC est obtenue par raffinage d'une HLC et non l'inverse.

La Figure 4.11 illustre le raffinage des HLCs en LLCs et les concepts de base de notre approche. Le modèle M2C contient des HLCs qui sont les correspondances reliant des méta-éléments appartenant aux différents méta-modèles. Le modèle M1C, conforme aussi au MMC, est obtenu par raffinage du M2C. Le M1C contient des LLC qui sont les correspondances reliant des éléments appartenant aux différents modèles.

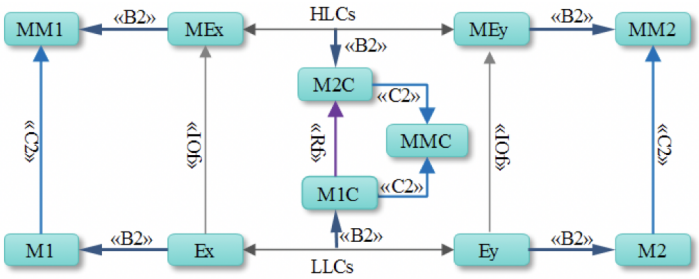


FIGURE 4.11 – Principe du raffinage des correspondances

où :

- MM1 et MM2 sont respectivement les méta-modèles des modèles M1 et M2 à apparier,
- MEx et MEy sont respectivement les méta-éléments dont Ex et Ey sont les instances,
- MMC, M2C et M1C sont respectivement le méta-modèle de correspondance que nous avons proposé, le méta-modèle de correspondance et le modèle de correspondance des modèles M1 et M2 à apparier
- Rf est la relation de raffinage,
- IOf est la relation InstanceOf
- B2 est la relation "Belongs to"
- C2 est la relation "Conforms to"

Il faut noter que dans cette approche, il est possible d'établir des correspondances de dimension supérieure à 2 entre un nombre de modèles également supérieur à 2. Dans la figure ci-dessus nous avons pris un exemple de correspondances binaire uniquement pour ne pas surcharger le schéma explicatif.

Le raffinage se déroule différemment selon que l'on apporte des précisions et/ou des contraintes à la correspondance établie au niveau modèle. Nous distinguons le raffinage par propagation et le raffinage par extension. Dans les deux cas, la première opération consiste à réaliser une propagation des HLC. Si les correspondances doivent comporter des informations plus précises au regard des modèles partiels et du domaine d'application, elles sont étendues.

Dans l'étude de cas du SU, 16 LLC ont été produites comme le montre la Figure 4.12.

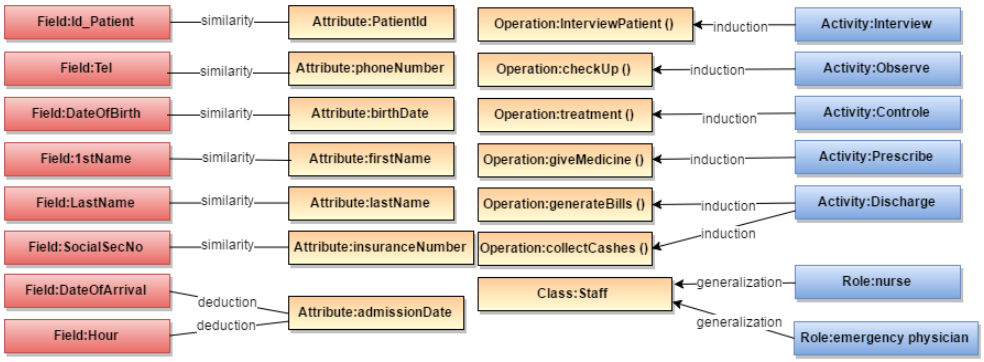


FIGURE 4.12 – Modèle M1C du SU

4.2.3 Raffinage par propagation

Le raffinage par propagation est défini par la composition d'une fonction de reproduction et d'une sélection :

$$M1C = Propagation(M2C) = Reproduction \circ Selection(M2C)$$

Reproduction

La reproduction fonctionne selon le schéma suivant (Figure 4.13), où l'on voit que toute instance de B est reliée à toute instance de Z et est reliée à toute instance de Y. (Notons que les instances de A ne sont pas liées).

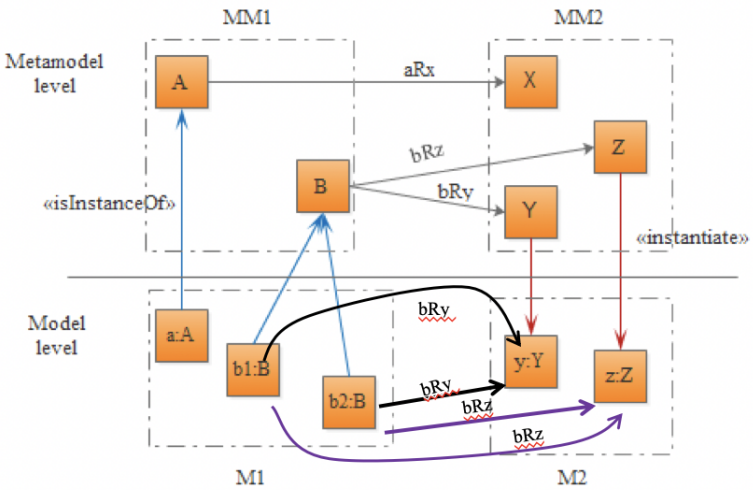


FIGURE 4.13 – Reproduction du M2C

Cette opération est un homomorphisme entre les correspondances du M2C et celles du M1C. Elle consiste à dupliquer au niveau M1, sous la forme de LLC, toutes les correspondances HLC définies au niveau M2. Il y a autant de LLC potentielles pour une HLC donnée que de produits cartésiens d'instances des méta-éléments impliqués dans cette HLC.

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

- Cn|M, l'ensemble des correspondances d'un modèle *M*,
- me/M, un élément *me* d'un modèle *M*,
- Ci[ma/MA, mb/MB], une correspondance *Ci* entre les éléments *ma* du modèle *MA* et *mb* du modèle *MB*

Si $\exists Cx,\ tel\ que\ Cx[mme1/MM1, mme2/MM2],\ avec\ M1 \mathcal{X} MM1,\ et\ M2 \mathcal{X} MM2,$
Alors :
$\forall me1/M1\ InstanceOf\ mme1/MM1,$
$\forall me2/M2\ InstanceOf\ mme2/MM2,$
$\exists Cy, [me1/M1, me2/M2]\ avec\ CyRfCx$

Ce qui explique que l'instance de A apparaissant sur la Figure 4.13 ne soit pas liée, puisque A est reliée à X au niveau M2 et qu'il n'y a pas d'instance de X au niveau M1.

Sélection

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. Cette Opération, que nous appelons Sélection détermine les correspondances qui constitueront le M1C.

L'opération de sélection consiste donc à filtrer les correspondances produites par l'opération de reproduction afin de ne conserver que celles qui sont valides. Les LLC sont filtrées grâce à l'expression sémantique associée aux types des relations qui les constituent. Nous avons en effet enrichi le meta-modèle de correspondance (cf. Figure 4.15) afin qu'une sémantique soit associée à chaque relation impliquée dans une correspondance. Cette sémantique peut-être plus ou moins formelle, allant de l'expression en langage naturel, à une expression formelle, en passant par sa description dans un langage de programmation (type Java). Ceci sera détaillé plus loin (section 4.2.5).

Le raffinage doit permettre aussi de compléter, en cas de besoin, la description des LLCs avec les fonctionnalités requises pour renseigner précisément le type de relation utilisé dans la correspondance. Nous avons proposé dans ce but une autre forme de raffinage, le raffinage par extension.

4.2.4 Raffinage par Extension

Les LLC créées par reproduction ne satisfont pas forcément les souhaits de l'expert. Il peut avoir à opérer des choix sur certaines actions à effectuer (Cariou et al., 2009) afin de préserver les propriétés souhaitées, ou pour ajouter des détails ou des informations sur les correspondances.

Le raffinage par extension, permet de redéfinir les correspondances afin d'ajouter aux types de relation des contraintes et/ou des traitements spécifiques au domaine d'application traité. Dans ce cas, l'extension prend la forme d'un nouveau type de relation. C'est une LLR qui hérite du type de relation qui a permis son extension. Dans le cas du SU, l'expert a considéré que certaines correspondances basées sur *Similarity* et *Aggregation* n'étaient pas suffisantes.

Ces types ont été remplacés respectivement par *Equality* et *Composition* via un raffinage par extension. La Figure 4.14 liste les correspondances concernées.

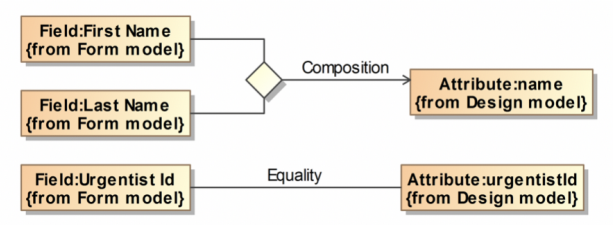


FIGURE 4.14 – LLCs obtenues par raffinage par extension

Les éléments *Attribute :urgentistid* et *Field :UrgentistId* décrivent le même concept et doivent donc contenir la même valeur d'où l'*Equality* qui les relie. Parallèlement, la correspondance ayant comme type de relation *Aggregation* reliant les éléments *Field :FirstName* et *Field :LastName* avec *Attribute :name*, est remplacée par une correspondance contenant une relation de type *Composition*.

4.2.5 Sémantique opérationnelle associée aux relations

Une activité fondamentale du processus d'alignement consiste donc à enrichir le MMC d'une expression sémantique (SE) pour chaque relation. C'est l'activité que l'expert du domaine effectue à l'étape 3 du processus 4.9 pour toute nouvelle relation introduite ; une expression sémantique étant par défaut associée aux relations de type DIR du MMC. Cette sémantique des DIR, fortement liée au domaine d'application, est modifiable en amont du processus d'alignement. Nous avons proposé un DSL d'expression sémantique qui est tissée dans le MMC sous la forme d'annotations, comme le révèle la Figure 4.15.

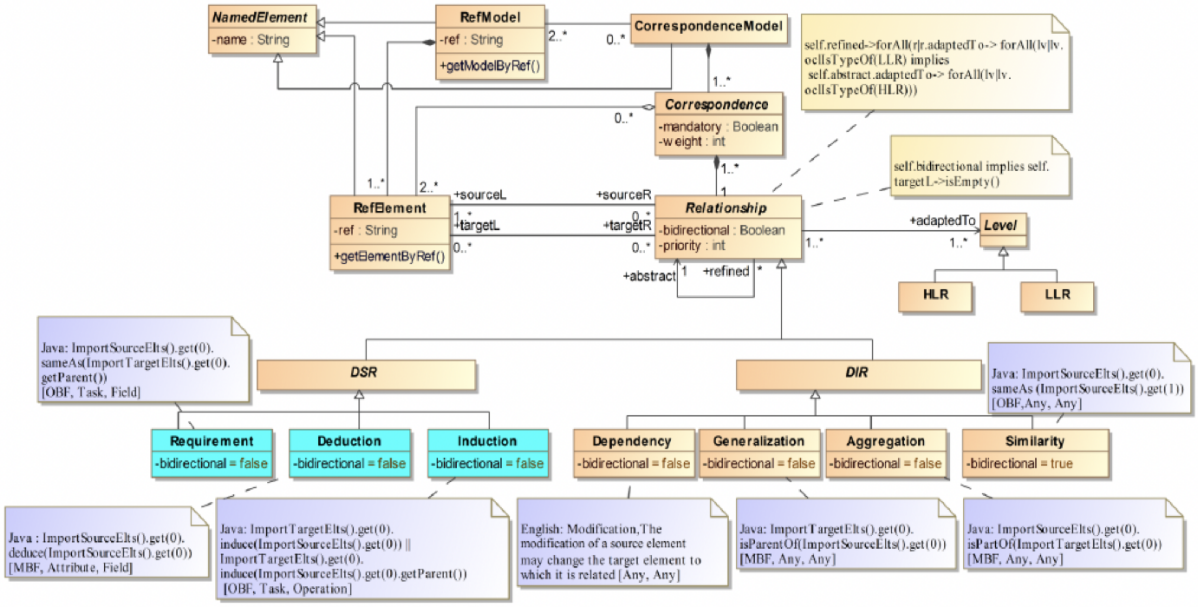


FIGURE 4.15 – MMC étendu par le tissage des expressions sémantiques des relations

DSL d'expression sémantique

L'intérêt de l'utilisation de ce DSL est non seulement (i) d'exploiter les actions pour filtrer les correspondances lors du raffinage afin de ne garder que celles qui correspondent à la sémantique de leur relation, mais aussi (ii) d'avoir une définition structurée commune de chaque type de relation et (iii) de permettre de construire le M2C de façon assistée à l'aide des informations sur les types d'éléments reliés.

Ce DSL est défini de la façon suivante (Figure 4.16) :

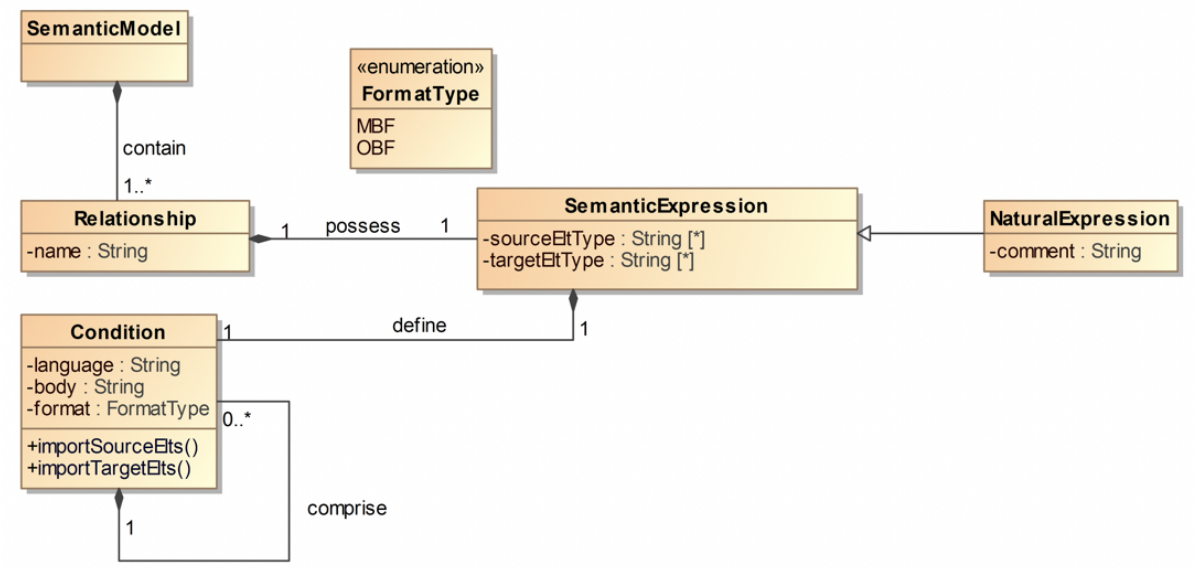


FIGURE 4.16 – DSL pour l"expression sémantique des relations inter-(méta)modèles

Une expression de sémantique opérationnelle est définie pour chaque relation. Elle est composée du type d'élément(s) source(s) et du type d'élément(s) cible(s) s'il y en a (i.e. si la relation est dirigée). La *condition* de l'expression sémantique définit le corps de l'action (*body*) à exécuter dans un *langage* tel que OCL [53], Java [31], Alloy [35], B [42], etc. L'attribut *format* précise si le corps de la condition est basé sur des éléments de modèle (MBF : Model Based Format) ou sur ceux d'une ontologie (OBF : Ontology Based Format). (Dans ce dernier cas, une transformation de l'espace technologique des modèles vers l'espace technologique des ontologies est nécessaire). La condition définit également deux opérations de récupération des éléments sources et cibles d'une correspondance. On notera que l'expert a la possibilité d'utiliser une expression (structurée) en langage naturel. La structuration est assurée par la définition du type des éléments à relier ainsi que la précision de la langue dans laquelle l'expression est écrite.

Un modèle d'expressions sémantiques, conforme à ce DSL, est alors créé et toute relation présente dans le MMC (sous-classes de DIR et DSR) se voit attribuer une sémantique. Ainsi, nous avons obtenu pour le SU le modèle d'expression sémantique suivant (Figure 4.17).

Dans l'exemple du MMC du SU, la sémantique (décrite en Java) de la relation *Similarity*, impose que cette dernière soit utilisée dans une correspondance impliquant deux éléments (récupérés par l'opération importSourceElts()) ayant des types indifférents (Any, Any). La condition stipule explicitement, en utilisant la fonction *sameAs*, que les éléments liés sont similaires.

Mise en œuvre dans l'opération de sélection

L'expert intégrateur précise la sémantique de chaque relation. Cette spécification, qui consiste à enrichir le MMC d'expressions, présentées sous forme d'annotations UML, va faciliter la mise

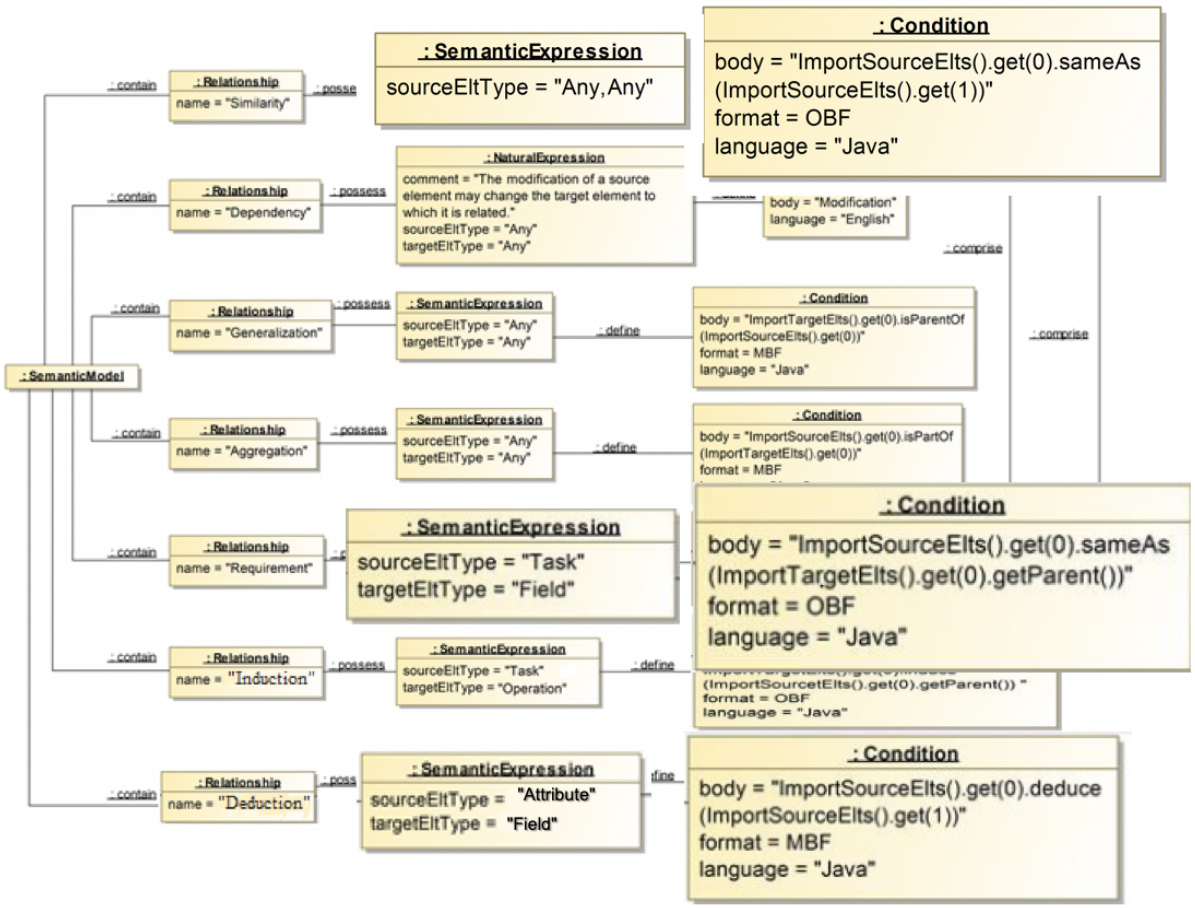


FIGURE 4.17 – Modèle d'expression sémantique des LLR du SU

en correspondance. Ainsi, lors de la sélection, la sémantique opérationnelle des relations ayant une expression formelle, va être exécutée automatiquement par l'outil support. (Cette exécution nécessite un interprète du langage dans lequel l'expression est écrite (une machine virtuelle Java dans notre cas)). Ceci va assurer que seules les relations valides seront reproduites dans le M1C. Ainsi, dans l'exemple du SU, l'existence de *similarity* entre *Field* et *Attribute* (Figure 4.10) provoque lors de la sélection l'exécution de sameAs entre toutes les instances concernées. "id-Patient".sameAs("PatientId") renvoie true, la LLC correspondante est maintenue dans M1C (Figure 4.12). A contrario, "Tel".sameAs("PatientId") renvoie false, la LLC correspondante ne fait pas partie du M1C.

Les relations avec expression informelle (en langage naturel), seront filtrées par le superviseur qui pourra décider de les conserver ou au contraire de les éliminer, en se fondant sur cette expression. La phase de sélection s'effectue donc de façon semi-automatique, certaines relations étant filtrées manuellement.

Nous allons voir que cette sémantique est également utile dans la gestion de la cohérence des modèles lors des évolutions (section 4.3 suivante).

4.3 Gestion de la cohérence

L'évolution des modèles n'est généralement pas coordonnée entre les concepteurs des différents modèles partiels et chaque modèle peut évoluer, indépendamment des autres, mettant en cause la cohérence de l'ensemble et du M1C. Une première solution garantissant un M1C correct consiste à exécuter le processus d'alignement après chaque changement. Cette solution n'est

évidemment pas satisfaisante, en raison de l'effort humain requis pour l'activité d'alignement d'une part et de l'absence de suivi automatique des évolutions d'autre part.

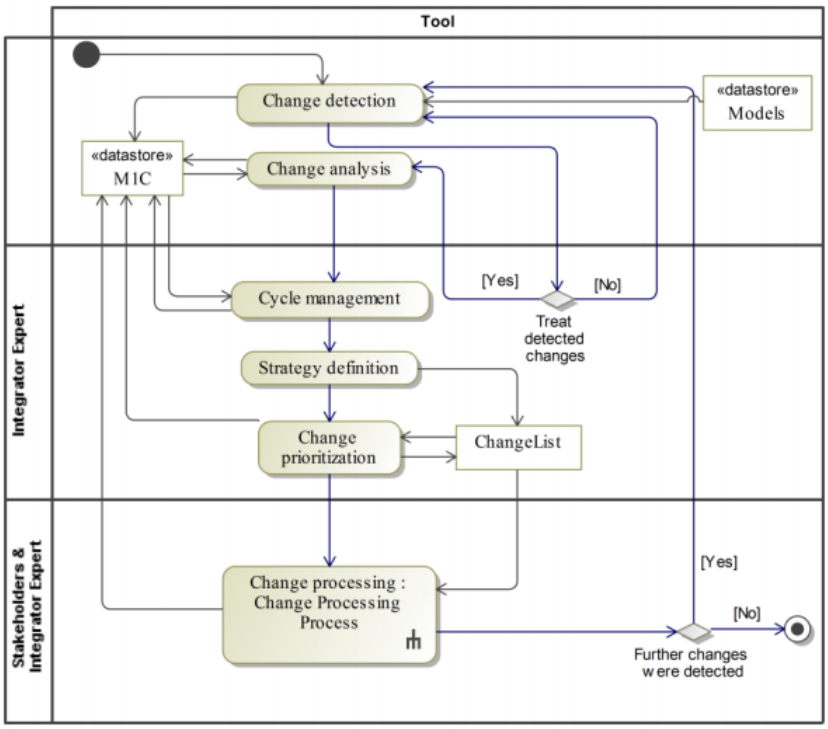


FIGURE 4.18 – Processus de gestion de la cohérence

La propagation des changements dans AHM est semi-automatisée. Elle se limite à l'ajout, la modification et la suppression des éléments de modèles source. Elle est mono-utilisateur et suppose qu'un rôle expert peut dérouler les deux processus de l'approche. Cette approche considère toutefois, que l'expert peut demander aux concepteurs des points de vue métier de l'assister, mais ceci n'est pas formalisé dans la description des processus.

L'approche AHM que nous avons proposée fournit un processus de gestion des évolutions. Ce processus est automatiquement activé à l'aide du patron Observer [91] à la fin du processus d'alignement. Il prend en entrée les modèles partiels du système et leur modèle de correspondance (M1C). Il s'effectue en six étapes décrites sur la Figure 4.18 : détection des changements, analyse des changements, gestion des cycles, stratégie de planification des changements, "priorisation" des changements et traitement des changements. La première étape nécessite un suivi automatique continu tandis que les autres sont déclenchées successivement à la demande de l'expert. Ce processus est supporté par un outil, HMCS, et implique l'intervention de l'expert dans des phases qui nécessitent son expertise ou une configuration humaine. Dans cette section, nous allons détailler ces six étapes.‘

4.3.1 Détection des modifications

La première étape, garantissant le maintien de la cohérence du modèle global du système, consiste à détecter les évolutions qui ont lieu au sein des modèles partiels. Ces changements sont détectés dés qu'ils se produisent, à l'aide du patron Observer qui a été introduit dans le MMC. Nous avons fait ce choix parmi plusieurs techniques de différentiation du type EMFCompare [46] qui fournit un algorithme générique pour calculer les différences entre deux versions d'un même modèle. Le problème que posent ces techniques dans notre cas est triple. (i) l'ensemble

du modèle doit être analysé pour mettre en évidence les différences entre deux versions d'un modèle, sur la base de techniques de calculs de distance, qui peuvent être longs. (ii) le nombre de modèles est limité car la comparaison ne peut se faire qu'entre deux modèles d'un même domaine d'activité. (iii) il y a un gaspillage de ressources puisqu'il faut conserver la version initiale du modèle d'entrée en même temps que sa version courante.

Les changements détectés par l'implantation du patron Observer sont ensuite ajoutés au M1C par l'intermédiaire des métaclasses MMC *History, DiffElt, AddedElt, DeletedElt, ModifiedElt* (partie 1 de la Figure 4.19).

History est utilisée pour garder une trace des modifications appliquées.

DiffElt permet d'enregistrer la trace des éléments qui ont évolué. Le premier de ses deux attributs contient le type de la modification qui a lieu. (Nous verrons section 4.3.2 que nous classons les changements afin de distinguer ceux que l'on peut traiter automatiquement, des autres). Le deuxième contient la référence de l'élément concerné, construite à partir du nom de l'élément, de son méta-élément et du nom du modèle.

DeletedElt représente un élément qui n'existe plus dans le modèle original mais qui est maintenu à des fins de traçage.

AddedElt et *ModifiedElt* représentent respectivement un élément nouvellement ajouté et un élément de modèle qui a subi une modification.

Le MMC étendu intègre également un concept qui représente le cœur du modèle de l'observateur (partie 2 de la Figure 4.19). La métaclasse *Observer* spécifie l'élément du modèle à observer. Il s'agit d'une généralisation de la métaclasse *Subject* qui a trois méthodes. Deux d'entre elles (*attach* et *detach*) permettent de fixer ou de détacher un objet observateur d'un élément de modèle. La troisième méthode (*notify*) permet de notifier le M1C des changements qui ont eu lieu. La méthode *update* de la métaclasse Observer est utilisée pendant la phase de traitement des changements, afin de maintenir la cohérence des modèles du domaine.

La troisième partie de la Figure 4.19, la métaclasse *impact*, définit le type d'impact de chaque changement (*impactType*) et la *solution* adaptée. Un changement sur un élément peut affecter directement ou indirectement des éléments qui sont liés à cet élément par une correspondance, comme nous le verrons dans la section 4.3.2.

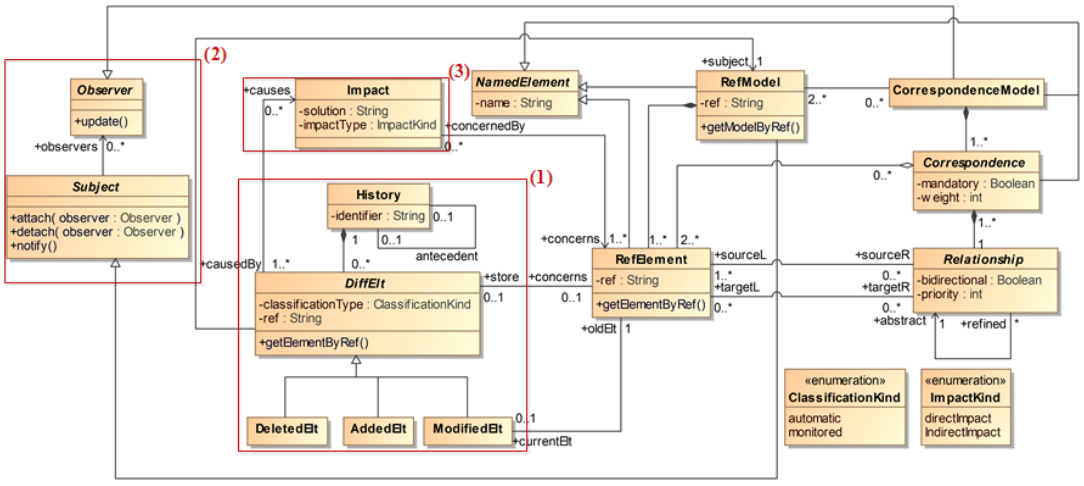


FIGURE 4.19 – Extension du Méta-Modèle de Correspondences (MMC) pour la gestion de la cohérence

Cas du SU

Les deux premières colonnes du tableau 4.1 ci-dessous résument les évolutions réalisées sur les modèles du SU, qui ont été automatiquement détectés via l'outil support et ajoutés au M1C.

Le champ *DateOfbirth* a été supprimé du modèle de formulaires, tandis que les champs *SocialSecNo* et *Conc* ont été modifiés et deviennent respectivement *PatientRecordNo* et *Description*. Par ailleurs, un rôle *Secretary* a été ajouté au modèle des protocoles médicaux. Il est responsable lors du départ du patient, d'imprimer les ordonnances ainsi que le compte rendu d'examen (CRE) d'urgence, puis d'éditer la facture et d'encaisser le paiement. Les éléments *Task :Print prescription*, *Task :Print emergency report*, *Task :Edit invoice* et *Task :Receive payment* ont donc été ajoutés au modèle des protocoles médicaux.

TABLE 4.1 – Aperçu des changements subis par le SU, de leurs répercussions et de leur classification

Type	**Model's Element**	**Change Analysis**		**Change Priorization**	
		Classif.	Influenced Elts.	Weight	Order
A	*Role :Secretary*	Auto.	-	0	**4**
A	*Activity :Print Prescription*	Auto.	-	0	**5**
A	*Activity :Print Emergency Report*	Auto.	-	0	**6**
A	*Activity :Edit Invoice*	Auto.	-	0	**7**
A	*Activity :Receive Payment*	Auto.	-	0	**8**
D	*Field :DateOfBirth*	Auto.	*Attribute :birthdate (Directly)*	1	**3**
D	*Field :DateOfArrival*	Auto.	-	0	
M	Old : *Field :SocialSecNo* New : *Field :PatientRecordNo*	Moni.	*Attribute :insuranceNumber(Directly)*	1	**2**
M	Old : *Field :Conc* New : *Field :Description*	Moni.	*Attribute :description(Directly)* *Activity :Diagnose(Directly)*	5	**1**

A : Addition, D : Suppression(Deletion), M : Modification
Classif : Classification, Auto : Automatique, Moni : Guidé (monitored)

4.3.2 Analyse des changements et gestion des cycles d'impacts

Dans la phase d'analyse des changements (étape 2 du processus de gestion de la cohérence 4.18), les changements sont d'abord classés en deux catégories : ceux qui sont gérés en mode automatique et ceux qui sont gérés en mode guidé. Ces modes de gestion des modifications visent à appréhender au mieux les conséquences d'un changement en attribuant à chaque mode des actions spécifiques. Ainsi,

- le Mode automatique (Automatic Evolution) concerne les changements qui mènent à des actions automatiques effectuées sur les modèles. Les types d'actions concernés sont l'ajout et la suppression,
- le Mode guidé (Monitored Evolution) concerne les changements du type modification. Lorsqu'un élément est modifié, les correspondances dont il fait partie doivent être réévaluées en fonction de la sémantique des types de relations utilisées. Selon Cicchetti et Ciccozzi [56], lorsque la sémantique du type de relation entre en jeu, les problèmes de gestion de versions deviennent plus complexes et ne peuvent pas être traitées automatiquement. En d'autres termes, le support d'automatisation est réduit et les changements sont susceptibles de faire intervenir une assistance humaine afin de décider de la répercussion du changement. C'est pourquoi ce type d'évolution nécessite d'être guidé.

Cette activité va donc consister à affecter un type de classification à chaque changement et à déterminer les éléments du M1C qui peuvent être impactés par ce changement.

L'extension du MMC (partie 3 de la Figure 4.19) permet de retrouver pour un élément modifié la ou les correspondances auxquelles il appartient et donc de retrouver le ou les éléments qui peuvent être affectés. Les éléments directement impactés sont des éléments directement liés à un élément modifié, alors que les éléments indirectement impactés sont des éléments qui peuvent être affectés par un effet en cascade.

Supposons que nous ayons deux correspondances (Figure 4.20), la première reliant un élément A à un élément B par la relation *Rx* et la seconde reliant l'élément B à un élément C par la relation *Ry*. Si l'élément A est modifié, l'élément B peut être influencé directement par ce changement, tandis que l'élément C peut être affecté indirectement par un effet de cascade lié au changement de B.

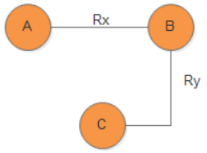


FIGURE 4.20 – Example de relations pouvant provoquer des impacts en cascade

Imaginons maintenant que C soit relié à A par une correspondance, on est dans le cas décrit sur la Figure 4.21 suivante où la modification de A peut impacter B puis indirectement C. La modification de C peut alors impacter A et on a fait face à un cycle de modifications.

Une fois les changements et leurs impacts directs ou indirects détectés, l'outil support capture automatiquement les cycles des effets en cascade. Lors de l'étape 3 du processus, tout cycle détecté est signalé à l'expert qui est responsable de rompre le cycle en supprimant l'une des correspondances.

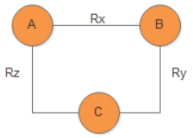


FIGURE 4.21 – Example de cycle d'impacts

La colonne "Analyse des changements" du tableau 4.1 décrit, pour chaque changement, le type de classification automatique ou guidé (*Classif*) et les éléments que le changement affecte directement ou indirectement (*Influenced Elts*). Ces éléments sont automatiquement intégrés au M1C. Par exemple, supprimer le *Field :DateOfBirth* a une influence directe sur *Attribute :birthdate*, tandis que supprimer *Field :DateOfArrival* n'a aucune influence sur les autres éléments des modèles. Ces deux modifications sont classées en mode automatique.

De même, les éléments de modèle ajoutés (par exemple : *Task :Print prescription*, *Task :print emergency report*, etc.) sont classés aussi en mode automatique et leur ajout n'affecte pas la cohérence du M1C existant.

Les éléments concernés par les modifications sont classés en mode guidé. En effet, la modification de l'élément *Field :SocialSecNo* a une influence directe sur l'élément *Attribute :insuranceNumber*. La modification de l'élément *Field :Conc* a une influence indirecte sur les éléments *Activity :Diagnose* et *Attribute :description*.

L'étape suivante va consister pour l'expert à décider de l'ordre de traitement des changements identifiés.

4.3.3 Définition de la stratégie et priorisation des changements

Les étapes 4 et 5 suivantes vont consister pour l'expert à choisir une stratégie d'ordonnancement des traitements des changements et à établir des priorités entre ceux-ci, afin de produire une liste ordonnée de changements à traiter.

Deux stratégies ont été proposées : une stratégie fondée sur le type de classification (Classification Based Strategy) et une stratégie fondée sur le type d'impact (Impact Based Strategy). La stratégie fondée sur le mode de classification consiste à créer une liste des changements qui contient dans l'ordre, les changements en mode automatique suivis de ceux en mode guidé - ou l'inverse, selon le choix de l'expert

La seconde stratégie produit une liste de changements fonction du type d'impact de chacun d'entre eux. Elle commence donc par les changements qui ont une influence directe et indirecte suivis de ceux qui ont une influence directe uniquement - ou l'inverse, selon le choix de l'expert.

Dans le cas du SU, la stratégie adoptée est celle à base de classification, en commençant par le mode guidé suivi du mode automatique. Ce choix est justifié par le besoin de traiter les changements de type modification avant ceux de type suppression ou ajout. Cela a donné lieu à la première liste suivante :

1. Ancien Field :Social Security Number Nouveau Field :Patient Record Number
2. Ancien Field :Explanation Nouveau Field :Description
3. Pool :Secretary
4. Task :Print prescription
5. Task :Print emergency report
6. Task :Edit invoice
7. Task :Receive payment
8. Field :Age

Néanmoins, (i) on a ici plusieurs éléments en mode guidé (les modifications) et plusieurs autres en mode automatique (les ajouts). (ii) Dans cette approche, nous traitons l'évolution d'un seul changement à la fois ; les changements d'un même mode doivent être classés entre eux. (iii) L'ordre de traitement des évolutions de modèles est important car l'impact sur le système dépend de cet ordre dans le sens où le résultat diffère selon l'ordre choisi.

Ces deux stratégies d'ordonnancement fonctionnent bien pour des changements qui ont des modes de changement différents ou différents types d'impact, mais nécessitent d'ordonner les traitements des changements qui ont le même type d'impact ou le même mode de classification.

Pour permettre de déterminer cet ordre de traitement des changements quels qu'ils soient, des coefficients de pondération (attribut *weight*) sont associés aux correspondances.

4.3.4 Calcul des coefficients de pondération

Un coefficient de pondération est attribué à chaque correspondance afin de déterminer un ordre de traitement.

Lorsque le M2C est construit, des valeurs de priorité sont affectés aux relations (attribut *priority* de la métaclasse *Relationship* du MMC 4.7). Ces valeurs permettant de classer les relations selon leur importance supposée, interviennent dans le calcul du coefficient de pondération utilisé lors de l'ordonnancement des traitements des changements.

Ainsi, pour le SU, l'expert a affecté une valeur de priorité à chaque type de relation. Les valeurs de priorité : 1, 1, 2, 2, 3, 5, 5 sont affectés respectivement aux types Similarité, Dépendance, Agrégation, Généralisation, Déduction, Induction et Exigence.

L'ordre de traitement est défini par le calcul automatique du coefficient de pondération de chaque correspondance.

Ce coefficient est calculé par la formule suivante :

$$\begin{aligned} weight &= \sum_{k=0}^{n}(DirectlyAffectedElement_k * priority) \\ &+ \sum_{k=0}^{n}(IndirectlyAffectedElement_k * priority) \end{aligned}$$

Lorsque plusieurs modifications ont le même coefficient de pondération, il appartient à l'expert de décider de l'ordre de traitement.

Le résultat de la priorisation est présenté sous forme d'une nouvelle liste ordonnée en fonction des coefficients de pondération calculés. Les changements sont traités dans l'ordre qui suit :

1. Field :Description (weight=6),
2. Field :Social Security Number (weight=1),
3. Field :Age (weight=3),
4. Pool :Secretary (weight=0),
5. Task :Print prescription (weight=0),
6. Task :Print emergency report (weight=0),
7. Task :Edit invoice (weight=0),
8. Task :Receive payment (weight=0).

Le poids est égal au nombre d'éléments impactés par les changement, multiplié par la priorité de la relation qui les lie. Le tableau ci-dessous 4.22 résume les calculs des coefficients de pondération et l'ordre obtenu.

Partie 1			**Partie 2**		
Type de Chgt.[1]	**Elément de modèle**		**Mode de Class.**[5]	**Coefficient de pondération**	**Ordre**
M.[3]	Ancien	*Field:Social Security Number*	Guidé	Weight = Attribute:insuranceNumber * prioritySimilarity = 1	2
	Nouveau	*Field:Patient Record Number*			
M.	Ancien	*Field:Explanation*	Guidé	Weight = Task:Diagnose * priorityRequirement + Attribute:Details * prioritySimilarity = 6	1
	Nouveau	*Field:Description*			
A.[2]	*Pool:Secretary*		Auto.[6]	0	4
A.	*Task:Print prescription*		Auto.	0	5
A.	*Task:Print emergency report*		Auto.	0	6
A	*Task:Edit invoice*		Auto.	0	7
A.	*Task:Receive payment*		Auto.	0	8
S.[4]	*Field:Age*		Auto.	Weight = Attribute:name * priorityDeduction = 3	3

FIGURE 4.22 – Calcul des coefficients de pondération des traitements des changements

L'ordre attribué aux changements de type ajout est aléatoire étant donné qu'ils sont traités à la fin du processus de traitement des évolutions (cf. Figure 4.23) lors de la mise en correspondance.

Une fois les modifications ordonnées, leur traitement commence.

4.3.5 Traitement des évolutions

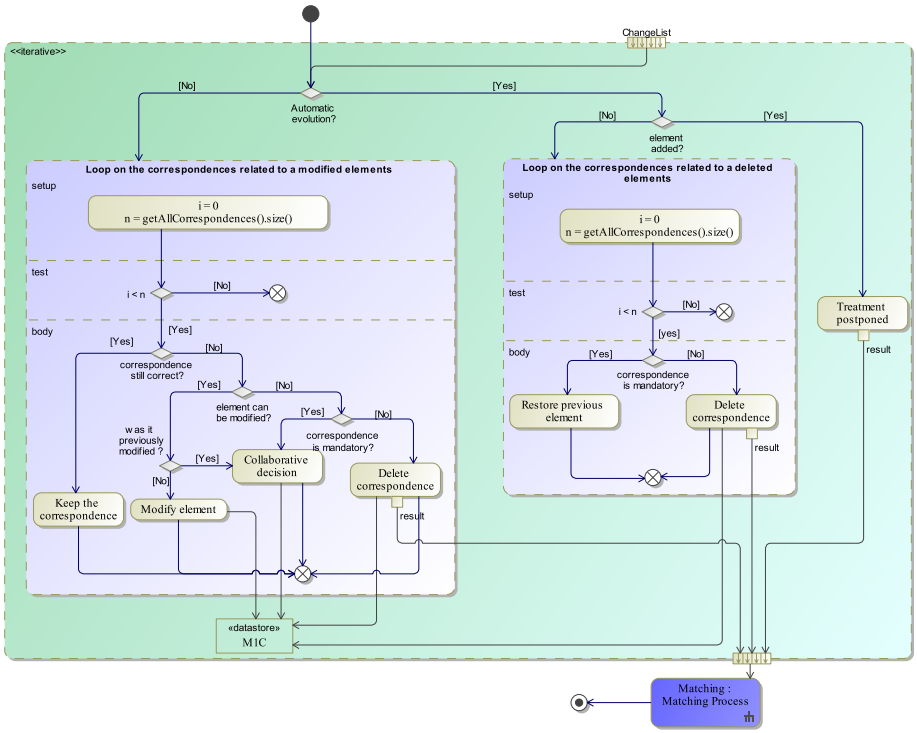


FIGURE 4.23 – Etapes du processus de traitement des évolutions

Dans cette étape, M1C et les modèles partiels peuvent être modifiés pour tenir compte des changements qui ont été détectés. La Figure 4.23 présente le processus de traitement des évolutions. Les modifications classées en mode automatique sont traitées automatiquement.

Dans le cas où un élément a été ajouté, le processus de comparaison est relancé à la fin du processus de modification pour traiter les éléments ajoutés en une seule fois. Si un élément a été supprimé, toutes les correspondances impliquant cet élément deviennent orphelines. Une correspondance orpheline est une correspondance pour laquelle il manque une de ses extrémités - un élément de modèle. Lorsqu'une correspondance devient orpheline, l'expert vérifie si elle est obligatoire pour le système concerné (la valeur de son attribut *mandatory* est true). Dans le cas positif, l'élément supprimé est restauré, sinon la correspondance est supprimée du modèle de correspondance.

Les changements en mode guidé sont gérés de manière semi-automatique. En effet, une modification ne sera traitée que si la sémantique associée à la relation contenue dans la correspondance concernée n'est plus correcte. Cela signifie qu'elle est possible. Il est alors nécessaire de modifier chacun des éléments qui lui sont liés. Dans le cas contraire, la correspondance est supprimée si elle n'est pas obligatoire (*mandatory* = *false*). Dans ce cas où la correspondance n'est plus valide (au regard de la sémantique associée à sa relation), et où elle est obligatoire, une prise de décision de l'expert est nécessaire. Il doit décider de la modification à apporter et du traitement de ses impacts. On notera que pour ce faire il peut se faire assister des concepteurs des modèles partiels concernés (modèle à l'origine de l'évolution ou impacté par celle-ci).

Dans le cas du SU, les deux premiers changements (Field :Social Security et Field :Explanation), voient les correspondances concernées maintenues car elles restent valides après leur modification. Cette validité est vérifiée par l'exécution du corps de la condition associée au type de relation impliquée dans la correspondance (que nous avons vu section 4.2.5). La suppression du troisième élément (*Field :Age*), entraîne la suppression de la correspondance puisque (i) elle

Ordre	Type de Chgt.[1]	Elément de modèle		Mode de Class.[5]	Action effectuée
1	M.[3]	Ancien	*Field:Social Security Number*	Guidé	Maintien de la correspondance
		Nouveau	*Field:Patient Record Number*		
2	M.[4]	Ancien	*Field:Explanation*	Guidé	Maintien de la correspondance
		Nouveau	*Field:Description*		
3	S.[3]	*Field:Age*		Auto.[6]	Suppression de la correspondance
4	A.[2]	*Pool:Secretary*		Auto.	Mise en correspondance
5	A.	*Task:Print prescription*		Auto.	Mise en correspondance
6	A.	*Task:Print emergency report*		Auto.	Mise en correspondance
7	A	*Task:Edit invoice*		Auto.	Mise en correspondance
8	A.	*Task:Receive payment*		Auto.	Mise en correspondance
9	A.	*Pool:Secretary*		Auto.[6]	Mise en correspondance

FIGURE 4.24 – Résumé de changements identifiés sur le SU

n'est pas obligatoire (mandatory = false) et (ii) elle est devenue orpheline. En ce qui concerne les autres changements de type ajout, l'opération de mise en correspondance est invoquée à la fin du processus pour chercher l'existence d'éventuelles correspondances et pour les créer ensuite. Le tableau 4.24 résume l'état des changements identifiés.

4.4 Outil support : Heterogeneous Matching and Consistency management Suite

L'outil support développé par Mahmoud El Hamlaoui, HMCS, accompagne l'approche présentée tout au long du processus d'alignement et de gestion de la cohérence. Je ne le présenterai que brièvement dans cette section car il était alors à l'état de prototype et a été étendu dans le cadre de la thèse de Saloua Benanni.

Il aide d'abord l'expert à établir les correspondances de niveau M2 (voir Figure 4.25) pour produire le M2C.

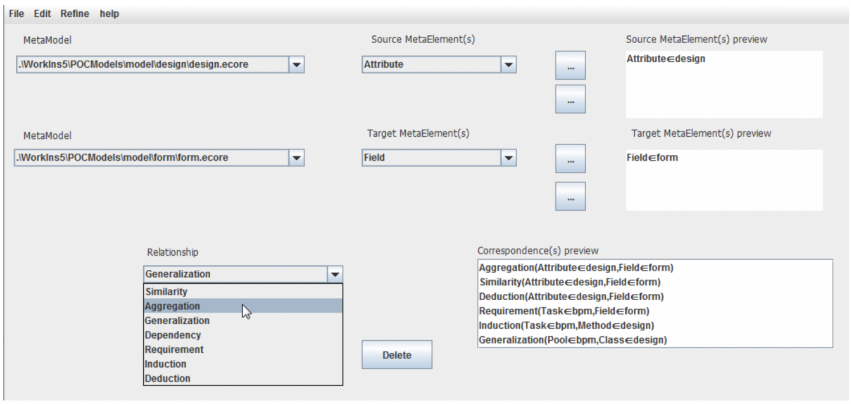


FIGURE 4.25 – Editeur graphique - création du méta-modèle de correspondance M2C du SU

Il génère ensuite automatiquement les LLC grâce à la sémantique des relations, lorsque celle-ci est formalisée. Il laisse la possibilité à l'expert de supprimer celles dont la sémantique est fausse, lorsque celle-ci était exprimée en Language naturel et qu'aucune inférence n'a pu être faite. Les LLC sont répertoriés et décrites, conformément au méta-modèle de correspondance (Figure 4.26)

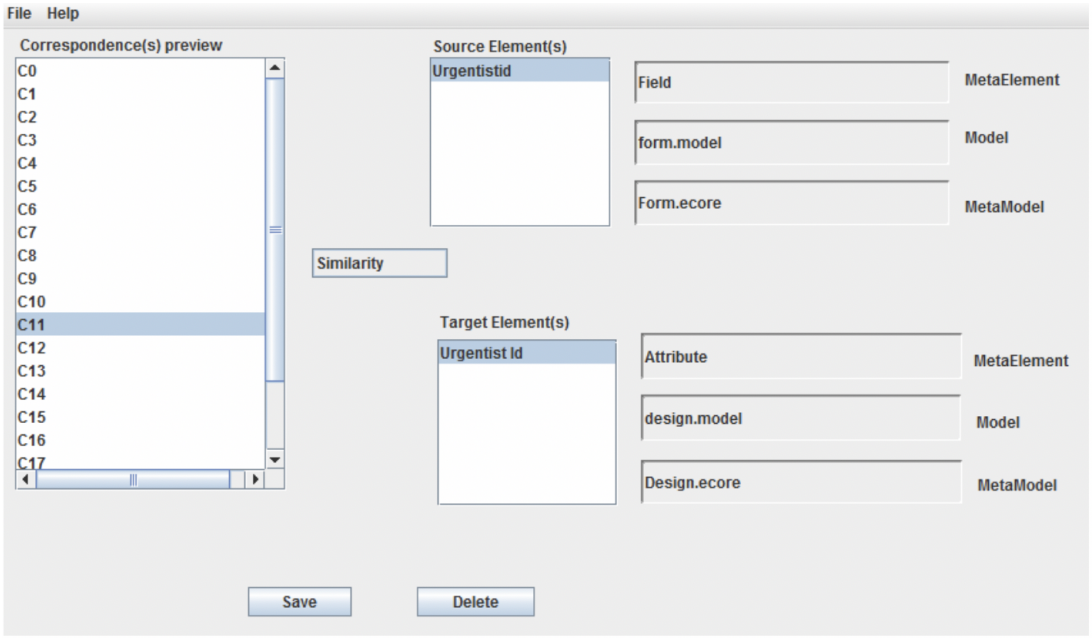


FIGURE 4.26 – Editeur graphique pour la création du modèle de correspondance M1C

La Figure 4.27 illustre les fonctionnalités apportées pour gérer les traitements des évolutions.

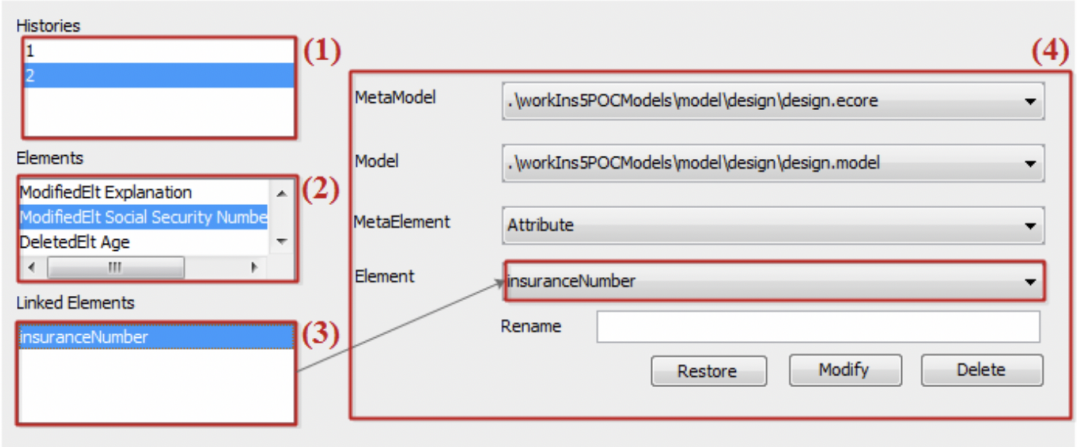


FIGURE 4.27 – Résumé de changements identifiés sur le SU

On affiche pour chaque historique (encadré 1) la liste des changements (encadré 2) ordonnée à l'aide des coefficients de pondération calculés. Par exemple, dans le cas de la deuxième évolution à traiter, la modification de *Social Security Number*, la liste des éléments susceptibles d'être impactés est affichée dans l'encadré 3. A ce stade du prototype, l'implantation de la sémantique permettant d'exécuter les corps de la condition associés aux types de relation n'était pas encore terminée. Les changements de type modification, ont été traités via l'encadré 4 qui permet une

fois l'expert assuré de la validité d'une correspondance, de décider, en fonction du processus de traitement des changements, de l'action à effectuer. L'action *restore* permet de récupérer la version précédente de l'élément et l'action *modify* de modifier l'élément de l'extrémité de la correspondance. Si la correspondance n'est plus valide, l'action *delete* l'enlève du modèle de correspondance.

HMCS est fondé sur Eclipse avec la structure décrite ci-dessous (Figure 4.28)

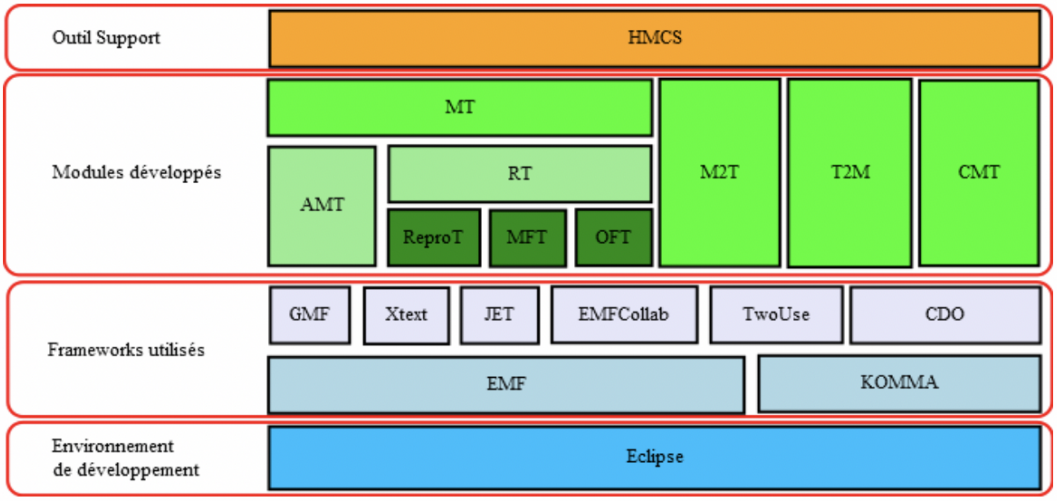


FIGURE 4.28 – Architecture technique de HMCS

où :

- le module MT supporte l'alignement des modèles. Il est lui même constitué de deux modules :
  - **AMT**, Assisted Matching Tool, dont le rôle est de créer le modèle M2C à partir des méta-modèles décrits en utilisant Eclipse Ecore. (Il peut être obtenu à partir d'un éditeur graphique ou bien d'un éditeur textuel synchronisé de façon bidirectionnelle). Pour celà, il s'appuie sur les modules **M2T** (Model to Text) et **T2M** (text to Model).
  - **RT**, Refining Tool, dont le rôle est de produire le modèle M1C pour les modèles source en entrée. Il intègre deux niveaux de raffinage : Propagation et Extension Le RT est composé lui même sur trois modules principaux :
    - **ReproT**, Reproduction Tool, permet de mettre en place la première étape du raffinement par propagation qu'est la reproduction.
    - MFT, Model Filtering Too et OFT, Ontology Filtering Tool supportent l'opération de sélection.
      - **MFT** renforce l'automatisation de la mise en correspondance en utilisant les expressions sémantiques des types de relation
      - **OFT** est sollicité quand les expressions sémantiques reposent sur des fonctions décrites en utilisant une base de connaissances (*Condition* avec un *FormatType* OBF.
- CMT, Consistency Management Tool, est le support à la gestion des évolutions de modèles et à la cohérence du M1C.

Pour plus de détail sur la réalisation de cet outil, il faut aller consulter [99] [2].

2. Chapitre VI, pp. 136-152

4.5 Conclusion

Nos travaux de recherche générale portent sur la conception de systèmes d'information complexes dans un contexte multi-vues. Au cours du cycle de modélisation, les modèles partiels peuvent évoluer fréquemment, en raison de la définition de nouvelles exigences ou de contraintes. Ainsi, plusieurs changements peuvent survenir dans différents modèles du système. Pour gérer la cohérence entre ces modèles, nous avons proposé une approche qui (1) établit semi-automatiquement un modèle de correspondance entre ces modèles,(2) utilise ensuite ce modèle de correspondance pour traiter les changements, identifiés automatiquement.

L'apport de cette approche est donc double. La première contribution est celle d'un processus permettant la création d'une vue globale du système par l'intermédiaire d'une composition fondée sur la mise en correspondance de plusieurs modèles partiels. Les correspondances, n-aires, identifiées entres les éléments des modèles se basent sur des types de relations instanciées à partir d'un méta-modèle de correspondance. Ce dernier est extensible (selon les spécificités du domaine d'application considéré) et permet de supporter les concepts relatifs à ce domaine. Les correspondances sont d'abord identifiées entre les méta-éléments des méta-modèles des modèles partiels. Les correspondances entre les éléments de modèles sont ensuite obtenues par un mécanisme de raffinement, supporté par un langage d'expression de la sémantique des relations : SED (Semantic Expression DSL). La composition est dite « virtuelle » dans la mesure où les éléments figurant dans une correspondance ne sont que des références aux éléments appartenant aux modèles partiels. De ce fait, les modèles inter-connectés par ces correspondances forment un modèle global virtuel.

La seconde contribution est relative au maintien de la cohérence des modèles partiels et du modèle global. En effet, les modèles évoluant dans le temps, le changement d'un élément ou de plusieurs éléments participant à des correspondances, peut entraîner l'incohérence du modèle global. Pour maintenir la cohérence du modèle global, nous proposons un second processus permettant (1) d'identifier automatiquement les changements réalisés (2) de les classifier en focniton de l'imoprtance de leurs répercusisons éventuelles et (3) de proposer des traitements adaptés, voire de les réaliser automatiquement.

Ce travail a été concrétisé par le développement d'un outil support nommé HMCS (Heterogeneous Matching and Consistency management Suite), basé sur la plateforme Eclipse.

L'approche a été validée et illustrée à travers un cas d'étude portant sur la gestion du Service d'Urgence d'un hôpital (ce travail a été mené en collaboration avec le CHU de Montpellier).

En proposant cette approche, nous avons répondu aux objectifs que nous nous étions fixés de :

Fournir un modèle global du système tout en conservant la production de modèles partiels assurant que les besoins des parties prenantes sont pris en compte.

Fournir des moyens d'aligner les modèles hétérogènes partiels d'un système, pour fournir un modèle global cohérent.

Fournir un support à la gestion de l'évolution des modèles.

Et nous avons relevé plusieurs défis, tels que :

Réaliser l'alignement des modèles hétérogènes, mais aussi maintenir la cohérence de l'ensemble dans les cas d'évolution d'un ou plusieurs modèles, ou encore d'introduction d'un nouveau modèle.

Fournir et mettre en œuvre des mécanismes de gestion des évolutions des modèles partiels.

Avec l'alignement de modèles hétérogènes supporté par AHM, nous répondons aux questions :

Question 1 : Comment garantir la validité des modèles de conception dans leur ensemble ?

Question 2 : Comment garantir le cohérence des modèles de conception dans leur ensemble ?

Question 4 : Comment assurer la cohérence de l'ensemble en permettant à chacun de travailler avec ses propres outils ?

4.6 Discussion

Dans le cadre de cette approche, nous nous sommes intéressés à l'apport des ontologies pour réaliser l'alignement des modèles partiels [105]. Sur la base d'un alignement commun au niveau méta-modèle, un ensemble de correspondances au niveau modèle est généré, puis affiné à l'aide d'approches de correspondance syntaxique.

Les résultats obtenus ont montré des pistes intéressantes, comme la possibilité de s'appuyer sur des techniques éprouvées pour réaliser l'alignement, fondé sur des éléments syntaxiques, ou des relations typiques comme la similarité.

Néanmoins, des limitations fortes, dues notamment au fait que la plupart des ontologies travaillent au niveau terminologique, tandis que nous travaillons essentiellement au niveau des assertions.

Bien que nous nous soyons concentrés dans cette étude sur la relation de similitude, comme la plupart des approches existantes en matière d'alignement de modèles, d'autres types de relations devraient pouvoir être exploitées.

En effet, les évolutions qui ont eu lieu dans le domaine des ontologies avec la possibilité de raisonner et de faire de l'inférence de règles, devrait permettre d'améliorer le processus d'alignement des modèles partiels et les techniques de maintien de la cohérence de l'ensemble du système [11].

Par ailleurs, si des métriques de la qualité de l'outil HMCS, quantifiées à l'aide de CodePro Analytix, ont été réalisées(cf. [99] [3]), l'évaluation de l'apport de l'utilisation de l'outil support qui a été menée ne l'a pas été dans des conditions satisfaisantes et n'a pas permis de véritablement conclure. Nous envisageons donc de mener cette étude de façon plus rigoureuse dans nos travaux futurs.

3. Chapitre VI, Fig. VI-14

Dans les perspectives à ce travail, il faut aussi envisager d'adapter cette approche pour soutenir la collaboration dans l'alignement des modèles et la gestion de la cohérence des modèles de correspondance. En effet, dans des systèmes réellement complexes, le rôle d'expert - ayant une vision et connaissance globale de toutes les préoccupations des différents points de vue - peut difficilement être défini. Nous avons noté que pour prendre certaines décisions, notamment lors du calcul de la répercussion de certaines évolutions, l'expert devait s'appuyer sur la connaissance des concepteurs concernés. Nous proposons que les concepteurs travaillent en étroite collaboration pour participer à l'élaboration du système global d'une part et au maintien de sa cohérence d'autre part. L'idée est de continuer à respecter l'indépendance de chaque "métier" en lui permettant de réaliser son propre modèle partiel, avec les langages et outils dédiées, mais de ne plus cloisonner le production des modèles. C'est ce sur quoi nous avons travaillé dans le cadre de la thèse de Saloua Bennani (cf. chapitre 5 suivant).

Chapitre 5

Alignement de modèles de conception hétérogènes ; l'évolution des modèles prise en charge dans un cadre collaboratif

Cette section reprend les éléments exposés dans les articles *"A Collaborative Decision Approach for Alignment of Heterogeneous Models"*, dans les actes de WETICE 2019, pp. 112-117 [25] et "*Collaborative model-based matching of heterogeneous models"*, dans CSCWD 2018,pp. 443-448 [26].
Cette section résume les résultats obtenus dans le cadre de la thèse de Saloua Bennani, "*Une approche IDM collaborative pour l'alignement de modèles hétérogènes*", soutenue le 16 juillet 2020 à Rabat [24].

Dans ce chapitre, je décrirai l'approche CAHM que nous proposons. CAHM, ou Collaborative Alignement of Heterogeneous Models, s'appuie sur un méta-modèle de collaboration, MMCollab, et des politiques de co-décision formalisées par des patrons de prise de décisions en groupe.

Elle propose un processus collaboratif composé de deux sous-processus : un premier dédié à la mise en correspondance de modèles hétérogènes et un deuxième pour le maintien de la cohérence inter-modèles.

L'approche proposée sera, comme la précédente (Chapitre 4), illustrée sur l'étude de cas du Service d'Urgences. Le fait d'utiliser un même cas d'étude pour illustrer l'approche permettra notamment de mieux cerner l'apport de la collaboration. L'outil support HMCS a été étendu par Saloua Bennani pour prendre en compte la dimension collaborative, dans le cadre de l'outil HMCS-Collab.

Je présenterai d'abord le méta-modèle de collaboration MMCollab. Ensuite nous verrons la modélisation de prise de décisions en groupe (GDM pour *Group Decision Making*) que nous avons proposée, en mettant un accent particulier sur le paquetage dédié. Cinq politiques de décision instanciées à partir de modèles de GDM seront évoquées.

CAHM permet de mettre en œuvre la gestion collaborative des changements en s'appuyant sur MMCollab, et sur le méta-modèle de correspondance que nous avons vu au chapitre 4. Celui-ci a été étendu afin de mieux capturer et gérer les changements (Il sera détaillé dans la Section 5.4.1). La figure 5.1 ci-dessous donne un aperçu synthétique de l'approche : elle a pour objectif la mise en correspondance des modèles partiels hétérogènes et évolutifs et s'appuie pour cela sur les deux méta-modèles MMC et MMCollab.

Elle fait intervenir :

(i) un expert chargé de la définition de la sémantique des relations du méta-modèle de correspondance,

(ii) un *manager* de la collaboration,

(iii) les différents concepteurs métiers impliqués.

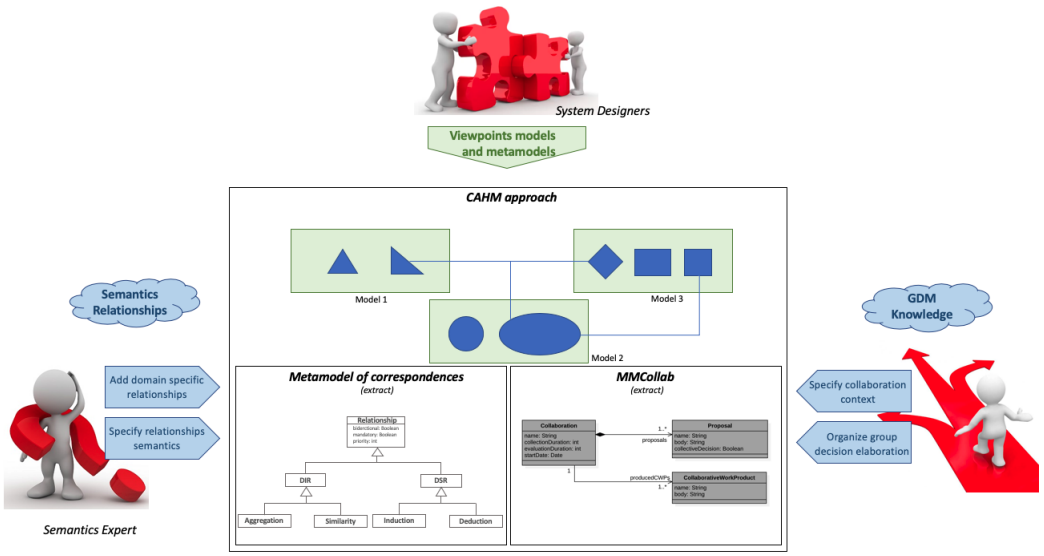


FIGURE 5.1 – L'approche CAHM

L'architecture de l'outil HMCS-Collab supportant, dans le cadre de prises de décision collaboratives, la mise en correspondance et la gestion de la cohérence en cas d'évolution, fera l'objet de la section 5.5.

5.1 Contexte

Cette étude s'appuie sur le travail qui a été mené dans le cadre de la thèse de Mahmoud El Hamlaoui (cf. Chapitre 4) et a donc considéré comme bases les éléments suivants :

- La modélisation de systèmes complexes faisant intervenir des concepteurs de domaines d'activité distincts.
- La production par ces concepteurs de modèles partiels, adaptés à leurs points de vue respectifs du système
- L'utilisation pour ce faire, de langages dédiés spécifiques au domaine (DSL) [42].
- L'approche AHM présentée dans le chapitre 4 précédent, qui propose :

  1. L'alignement de modèles hétérogènes par le biais d'un modèle de correspondance virtuel. En effet, pour assurer une cohérence globale entre les modèles partiels d'un système, la solution que nous avons choisie consiste à établir et maintenir les correspondances inter-modèles ; Une *correspondance* étant constituée d'une relation sémantique reliant au moins deux éléments.

2. Le maintien de la cohérence de l'ensemble des modèles partiels connectés, s'appuyant sur ce modèle de correspondance
3. Le rôle majeur de l'expert du domaine chargé de donner les règles de l'alignement des modèles concernés (par l'intermédiaire de la définition de correspondances au niveau des méta-modèles (M2))
4. Le rôle majeur de l'expert du domaine chargé de prendre seul les décisions non automatisables, que ce soit (i) lors de l'établissement des correspondances inter-méta-modèles, (ii) lors de la sélection des correspondances valides de niveau modèle (M1), (iii) ou encore lors du traitement des évolutions de modèles et de leurs répercussions.

Nous avons donc supposé jusqu'ici qu'un seul acteur, l'expert intégrateur du système, puisse effectuer seul l'appariement des modèles [94, 100, 101, 50].
Si l'hypothèse de l'acteur unique peut raisonnablement s'appliquer aux systèmes ayant un nombre limité de points de vue, elle n'est plus valide dans le cas de systèmes complexes. En effet, quel que soit le degré d'expertise de l'acteur qui effectue l'alignement, il ne peut pas saisir les préoccupations techniques et fonctionnelles de tous les points de vue impliqués, en particulier dans le cas de modèles fortement hétérogènes et de gros systèmes.

Nous avons donc envisagé dans cette étude de ne plus faire supporter le poids des décisions au seul expert intégrateur du système. L'idée que nous poursuivons est, en effet, d'intégrer l'approche dans un contexte collaboratif, impliquant tous les acteurs concernés.

Faire intervenir les différents acteurs métiers, à la fois lors de l'élaboration du modèle de correspondance et lors du maintien de sa cohérence, permet l'intégration de connaissances et de préoccupations plus larges, et facilite l'alignement des modèles tout en garantissant la cohérence et la fiabilité de l'ensemble.

Le premier postulat que nous posons est par conséquent que la conception de ce type de systèmes ne peut être que meilleure si elle est faite dans un cadre collaboratif [72, 86].

Le deuxième postulat consiste à dire, comme nous l'avons toujours fait et comme [51], que l'IDM permet aux acteurs métier de construire et utiliser les outils dont ils ont besoin dans leurs domaines (Modeling by the People, for the People [125]).

Le troisième postulat est que la définition des liens inter-modèles selon les spécificités du domaine d'application est de la responsabilité des acteurs métiers concernés.

Mais impliquer les acteurs métier dans la définition des liens entre modèles soulève la question de la manière dont ils peuvent collaborer. Bien que les pratiques industrielles favorisent la conception collaborative, l'alignement collaboratif de modèles hétérogènes se fait encore dans la pratique de manière informelle, ce qui est fastidieux et source d'erreurs. C'est pourquoi nous avons proposé d'étendre l'approche AHM par une démarche collaborative outillée, CAHM (collaborative AHM).

5.2 Modélisation de la prise de décision en groupe

Pour répondre à ce besoin de collaboration, nous proposons en effet une approche semi-automatique d'alignement collaboratif de modèles hétérogènes [26]. Elle combine l'ingénierie basée sur les modèles et la prise de décision en groupe (GDM [23]). Elle permet dans un premier temps d'établir les correspondances entre les modèles hétérogènes, puis dans un deuxième temps d'assurer la cohérence du modèle de correspondance ainsi obtenu, et des modèles source quand ils sont modifiés. Elle est fondée sur un méta-modèle de collaboration, MMCollab [26] présenté dans la section suivante 5.2.1.

5.2.1 Le méta-modèle MMCollab

MMCollab peut être instancié pour décrire chaque session de collaboration où les acteurs font des propositions, les évaluent ou les affinent pour aboutir à une décision collective.

Chaque paquetage de MMCollab couvre une partie de la modélisation de la prise de décision collaborative, à savoir :

- l'organisation des acteurs (paquetage *Actors)*,
- l'organisation des propositions (paquetage *Proposal*),
- l'évaluation des propositions (paquetage *Evaluation*),
- l'élaboration de la décision collective (paquetage *CollectiveDecision*) et
- des concepts de base (paquetage *CoreConcepts*).

5.2.2 Vue d'ensemble de MMCollab

Collaboration est le concept central de MMCollab, décrit dans la figure 5.2. Il s'agit d'une spécialisation du concept *Activity* de SPEM [161].

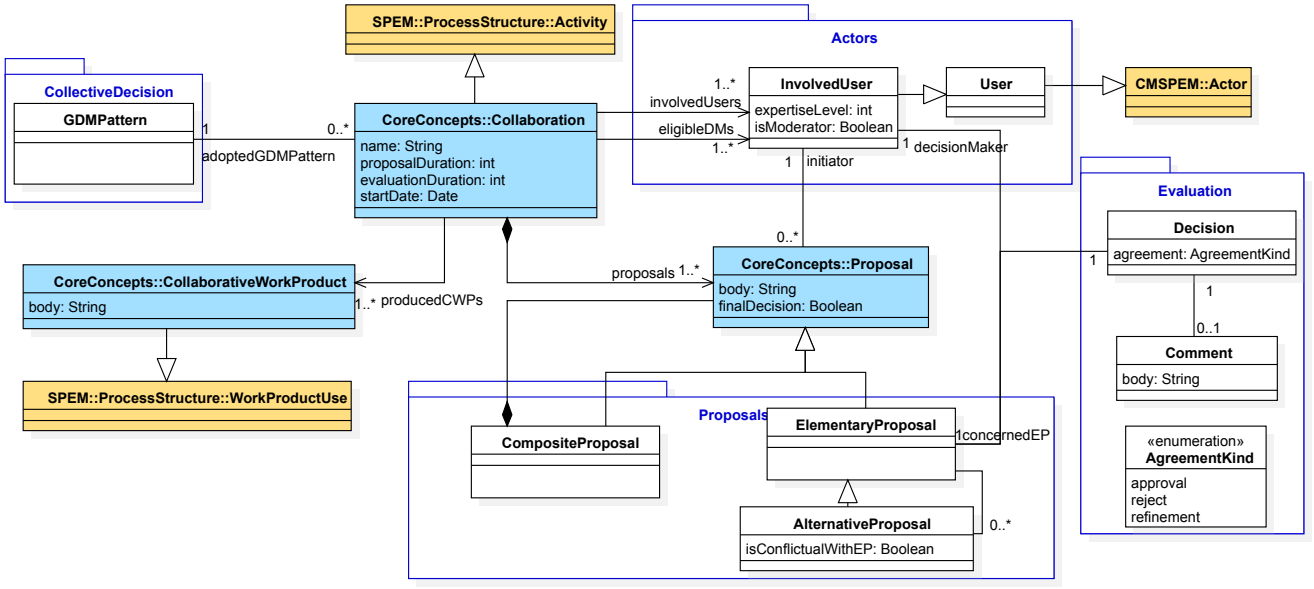


FIGURE 5.2 – Aperçu du méta-modèle de collaboration (MMCollab)

Une collaboration comprend un ensemble de propositions (*proposals*). Une **proposition** décrit une solution proposée au problème identifié, et doit faire l'objet d'une prise de décision.

Une collaboration implique un ensemble de **participants** (*involvedUsers*), dont un modérateur de la collaboration (attribut *isModerator==true* de *InvolvedUser*).

Une collaboration est mise en place via un patron de prise de décision en groupe, *GDMPattern*, qui va permettre de décrire la politique de décision en groupe adoptée (ceci fera l'objet de la section 5.2.3).

Le rôle d'un **modérateur** est de choisir cette politique de décision (i.e. le modèle de patron de prise de décision en groupe à adopter, *adoptedGDMPattern*). Un des rôles du modérateur est également de fixer les durées maximum de la collaboration. Celles-ci concernent le temps de collecte des propositions et le temps d'évaluation des propositions faites (*collectionDuration* et *evaluationDuration*). Par défaut, l'acteur qui a fait la première proposition est considéré comme le modérateur de la collaboration.

Une liste de **décideurs éligibles** (*eligibleDMs*) est initialisée avec les participants qui satisfont au modèle de patron de prise de décision en groupe (GDMP) adopté. Chaque décideur se voit attribuer un poids décisionnel (une estimation de son niveau d'expertise, *expertiselevel*).

Une proposition peut être composite ou élémentaire. *CompositeProposal* est une sorte de transaction atomique, composée d'un arbre de propositions dont les feuilles sont des propositions dites élémentaires (*elementaryProposal*) (EP), qui sont soit approuvées soit rejetées, ensemble. Chaque EP provient d'un initiateur (*initiator*) et doit être évaluée par les décideurs éligibles. Un **décideur** (*decisionMaker*) est un participant qui est en mesure d'évaluer une proposition.

L'**évaluation** consiste à produire une décision individuelle (*decision*). La décision peut être une approbation, un rejet ou un affinage (valeurs *approval*, *reject* ou un *refinement* de l'énumération *AgreementKind*).

Lorsqu'un décideur rejette une EP, il doit justifier son choix par un commentaire (*comment*). S'il estime qu'une EP doit être affinée, il fournit une proposition alternative, *AlternativeProposal* (AP). L'attribut *isConflictualWithEP* d'une *AP* précise si cette AP est en conflit avec la EP à laquelle elle est rattachée.

La proposition est caractérisée par sa description (*body*) et une **décision finale** (*finalDecision*), qui représente la décision finale du groupe en ce qui la concerne. La valeur de l'attribut *finalDecision* d'une proposition est fixée en fonction de la politique de décision adoptée et des évaluations faites des propositions composites, alternatives, etc ... au cours de la collaboration.

Une collaboration produit un ou plusieurs *CollaborativeWorkProduct* (spécialisation de *WorkProductUse* de SPEM 2.0 Process Structure) qui rassemble **l'ensemble des propositions approuvées**.

5.2.3 Le paquetage CollectiveDecision

Le paquetage ***CollectiveDecision***, présenté dans la figure 5.3, fait l'objet d'une section particulière car il est central dans le processus collaboratif de prise de décision. En effet, il rassemble les concepts nécessaires à l'élaboration d'une décision collective dans un contexte de prise de décision en groupe. Le principal concept de ce paquetage est celui de **patron de prise de décision en groupes** *GDMPattern*. Il s'agit d'une spécialisation de Pattern - défini selon la structure largement utilisée dans la conception de logiciels pour décrire les patrons [91], c'est-à-dire une intention (*intention*), un ensemble de contextes (*application*), un ensemble d'utilisations connues (*unknown use*), une solution (*solution*) et une association réflexive (*parent-child*).

Outre ses caractéristiques héritées, un *GDMPattern* se compose d'une Méthode de participation (*Participation Method*) et d'une Méthode de co-décision (*Co-decision Method*).
La **méthode de participation** précise comment les parties prenantes participent à la prise de décision. Elle est caractérisée par l'énumération *ParticipationType*. Elle est ainsi démocratique (*democratic*) lorsque toutes les parties prenantes sont impliquées et limitée (*restricted*) lorsque seul un sous-ensemble d'acteurs est sélectionné.

Pour chaque *ParticipationMethod*, certains paramètres peuvent être spécifiés. Elle peut être par exemple (valeurs de *ParameterKind*), basée sur l'anonymat des participants (*anonymous*) ou sur l'attribution de coefficients de confiance (ou d'expertise) accordés aux participants (*weighted*). En cas de participation restreinte, le critère de sélection des parties prenantes devra également être précisé. Un participant peut être sélectionné (i) parce qu'il est expert dans le domaine concerné par la prise de décision, ou (ii) tout simplement parce qu'il est disponible pour le processus de prise de décision (*expertise* vs. *disponibility*).

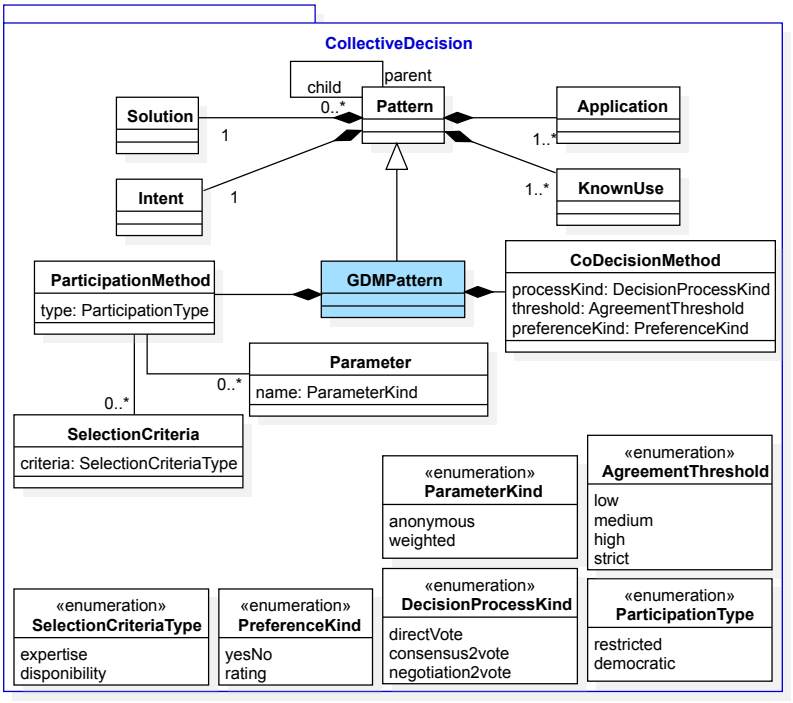


FIGURE 5.3 – *CollectiveDecision* package de MMCollab

La **méthode de co-décision** est déterminée par trois attributs :

- Les seuils facilitent la prise de décision en groupe. En effet, les groupes peuvent utiliser des fourchettes d'accord, fixées par un seuil (***threshold***), pour la validation des propositions. Un seuil strict (*strict*) signifie qu'un accord à 100% est requis alors que les seuils faibles (*low*), moyens (*medium*), ou (élevés *high*) évitent à la prise de décision d'être soumise à un accord strict.

- Le type de processus (***processKind***) précise le processus d'évaluation des propositions. Étant donné que les parties prenantes peuvent se trouver à différents endroits, même les processus consensuels ou de négociation donnent lieu à un vote final pour recueillir les avis. Nous proposons donc trois sortes de processus de décision dans l'énumération *DecisionProcessKind* : vote direct (*direct vote*), vote con-sensuel *Consensus2vote* (exige un seuil strict) et vote après négociation (*negotiation2vote*) (avec un seuil bas, moyen ou élevé).

- Le type de préférence (***preferenceKind***) précise comment les propositions sont évaluées : par une graduation (*rating*) ou de façon binaire par une acceptation, ou un rejet (*yesNo*).

5.2.4 Exemples de Politiques de décision, instances du patron de GDM

Une *Politique de décision* (DP, pour *Decision Policy*) est une instance d'un patron de GDM. Plus précisément, une DP est une combinaison d'instances d'éléments qui composent un patron de GDM, soit une Méthode de participation et une Méthode de co-décision, et par conséquent une combinaison d'instances d'éléments qui caractérisent à la fois une méthode de participation et une méthode de co-décision :

- le type de participation (*type*),

- le type de processus de décision (*processusKind*),

- le seuil d'accord (*threshold*), et
- le type de préférence (*préférenceKind*)

La combinaison de ces éléments nous a permis de définir cinq politiques de décision qui décrivent les politiques couramment utilisées dans le cadre d'une prise de décision (classes mises en évidence sur la figure 5.4) ci-dessous.

Ces cinq DP peuvent être classées en fonction du type de participation associé (Restreint (*RestrictedDP*) vs Démocratique (*DemocraticDP*)), ou encore en fonction du nombre de tours nécessaires pour arriver à une décision (*SingleElectionDP* vs *DemocraticDP*).

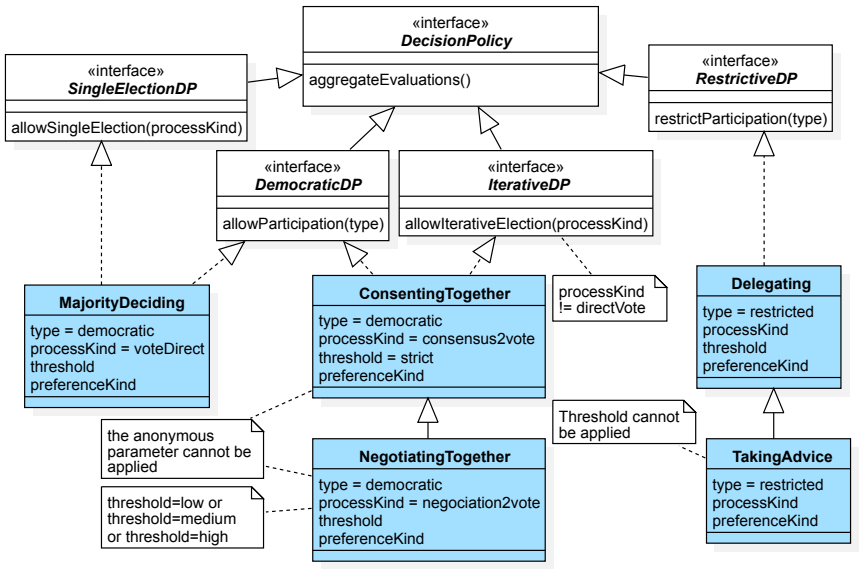


FIGURE 5.4 – Instances de politiques de décision du (*GDMPattern*) et leurs dépendances

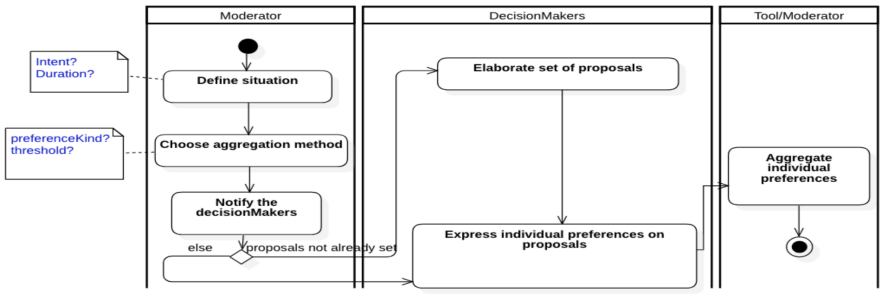
- *MajorityDeciding* est une politique de décision démocratique. Elle hérite également de la DP à élection unique (*SingleElectionDP*) car elle est réalisée en un seul tour de consultation. Cela signifie que si les décideurs n'ont pas atteint le seuil défini à la fin de la collaboration, ils doivent soit ajuster le seuil, soit ré-évaluer les propositions.
- *ConsentingTogether* et *NegotiatingTogether* sont des DP itératives, ce qui signifie qu'elles peuvent être répétées jusqu'à ce que le seuil fixé soit atteint. *ConsentingTogether* exige un seuil strict (100% d'accord) tandis que *NegotiatingTogether* fonctionne avec un seuil bas, moyen ou élevé.
- *Delegating* et *TakingAdvice* sont des DP restreintes ; elles nécessitent donc de préciser les critères de sélection des décideurs.

Ces politiques de décision ne sont pas figées et peuvent être étendues en fonction des contextes d'application, en explorant les combinaisons possibles des éléments qui les composent. Par exemple, les *processKind* et *threshold* de la politique de décision *Delegating* ne sont pas constants. Ils peuvent donc prendre toutes les valeurs possibles et fournir une politique de décision similaire à la décision à la majorité, ou au consentement ou encore à la négociation, mais dans un mode restrictif par exemple.

Pour faciliter le choix parmi ces politiques de décision, nous fournissons un manuel descriptif qui inventorie les patrons proposés, selon le modèle de Gamma [91]. Le tableau 5.1 en donne un exemple avec la description de la politique de prise de décision à la majorité, *MajorityDeciding*.

TABLE 5.1 – Description de la politique de prise de décision à la majorité

Name	*Majority Deciding*
Intent	Parvenir à une décision qui tienne compte des avis de toutes les parties prenantes. La, ou les, proposition(s) approuvée(s) par la majorité du groupe est, ou sont, adoptée(s).
Applications	Ce modèle doit être utilisé dans le cas où : - les compétences nécessaires et le niveau d'expertise des décideurs sont équivalents. - la prise de décision est soumise à des contraintes de temps. En effet, ce patron nécessite peu de temps car la décision est prise par vote à un seul tour.
Known uses	Elections à un seul tour organisées soit en présence, soit par vote électronique.
Solution	La mise en œuvre de ce patron passe par six étapes (cf.diagramme d'activités ci-dessous) : Tout d'abord, le modérateur définit les caractéristiques de la collaboration (intention et durée). Ensuite, il fixe le seuil et le type de préférence de la méthode de co-décision (le type de processus est fixé à vote direct). Ensuite, il informe les acteurs auxquels il attribue le rôle de décideur. Si les propositions ne sont pas établies au préalable, les décideurs commencent par en dresser la liste. Ils expriment ensuite leurs préférences individuelles. Enfin, les préférences individuelles sont regroupées (automatiquement par un outil ou éventuellement par le modérateur), et les propositions dépassant le seuil sont approuvées. Elles constituent la décision de groupe. Plusieurs propositions peuvent être approuvées si elles ne sont pas contradictoires.
Related patterns	Delegating *Majority Deciding* et *Delegating* diffèrent dans le type de participation et le poids décisionnel accordé aux acteurs. *Delegating* permet de faire une sélection préalable des acteurs impliqués, alors que *Majority Deciding* est démocratique.



5.3 Application à l'alignement des modèles du cas d'étude du Service d'Urgence

Nous appliquons la modélisation de la collaboration proposée à la mise en correspondance des modèles du système de service d'urgence d'un hôpital (SU), décrit section 4.1 du chapitre 4. Les modèles partiels décrivant ce système ont été définis en coopération avec les médecins urgentistes d'un hôpital français et ont été élaborés par des équipes de conception différentes, dans le cadre d'une étude de cas impliquant plusieurs équipes de recherche [101].

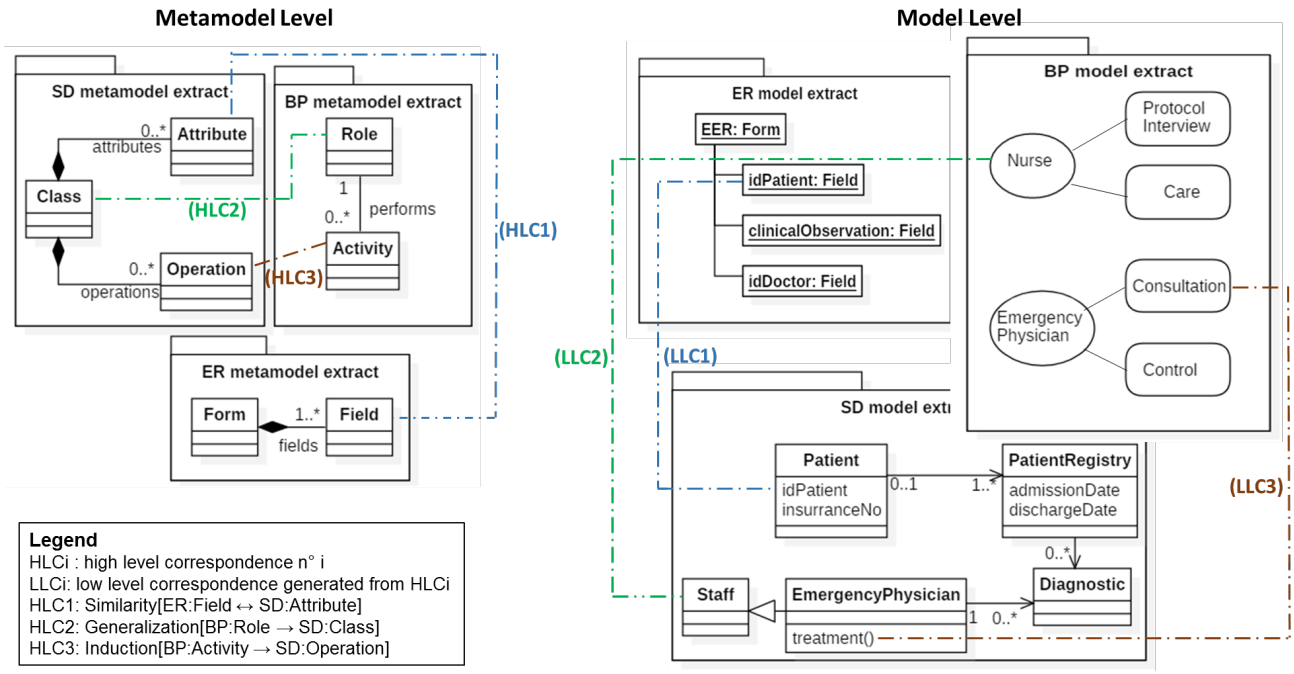


FIGURE 5.5 – Extraits des modèles partiels du SU et de leurs connexions (HLC et LLC)

La figure 5.5 présente un extrait de ces méta-modèles, de leurs modèles respectifs et de certaines de leurs connexions.

- Le modèle SD contient des classes concernant les personnels hospitaliers, les patients, leurs antécédents médicaux, etc ...
- Les rôles et les activités des différents intervenants sont décrits dans le modèle BP.
- Dans le modèle ER, les champs qui forment le rapport médical sont décrits (numéro de sécurité sociale, observations cliniques, etc.).

Ces modèles, hétérogènes, incluent des éléments communs ou dépendants qui doivent être orchestrés pour assurer la cohérence du système. Dans ce qui suit, nous allons voir le processus collaboratif mis en œuvre pour aligner ces modèles.

5.3.1 Processus collaboratif d'alignement

Nous avons instancié MMCollab et défini un processus de mise en correspondances collaboratif.

Les acteurs du processus sont :

- Des **coordinateurs locaux** : Chaque équipe de concepteurs désigne un coordinateur local (LC pour *local coordinator*), qui participe à ce processus.

- Le **modérateur** de la session, choisi par les LCs par consensus.
- Un **expert en sémantique** qui définira la sémantique des DSR à ajouter.
- HMCS-Collab : L'outil qui étend HMCS, le prototype support de l'approche AHM. HMCS-Collab supporte la collaboration lors de la mise en correspondances et du maintien de la cohérence.

L'objectif du processus d'alignement est de fournir le modèle de correspondance M1C connectant les modèles locaux. Il reste dans ses grandes étapes identique à celui que nous avions proposé dans AHM (4) ; ces étapes deviennent collaboratives.
Celles-ci consistent essentiellement :

1) à étendre le méta-modèle de correspondance si nécessaire par des Correspondances et des Relations Spécifiques au Domaine (DSR)
2) à établir les correspondances entre méta-modèles (HLC) pour construire le modèle de correspondance M2C.

Pour chaque activité collaborative (c'est-à-dire *Extension de MMC* et *Production du M2C*), le modérateur choisit la politique de décision à adopter. Selon cette politique de décision, une proposition peut être considérée comme (i) une action individuelle qui est ensuite évaluée en collaboration dans le cas d'une politique de vote ou (ii) une action de collaboration où une partie prenante initie un changement et les autres l'affinent par un "brainstorming" afin d'arriver à un consensus.

La figure 5.6 illustre le processus d'alignement collaboratif, selon le formalisme CM-SPEM [124]. Il produit un modèle de correspondance entre des éléments de modèles hétérogènes à travers quatre activités principales. Ce modèle de processus, met aussi en évidence les principaux *CollaborativeWorkproducts* produits et consommés par chaque activité, et les acteurs participant.

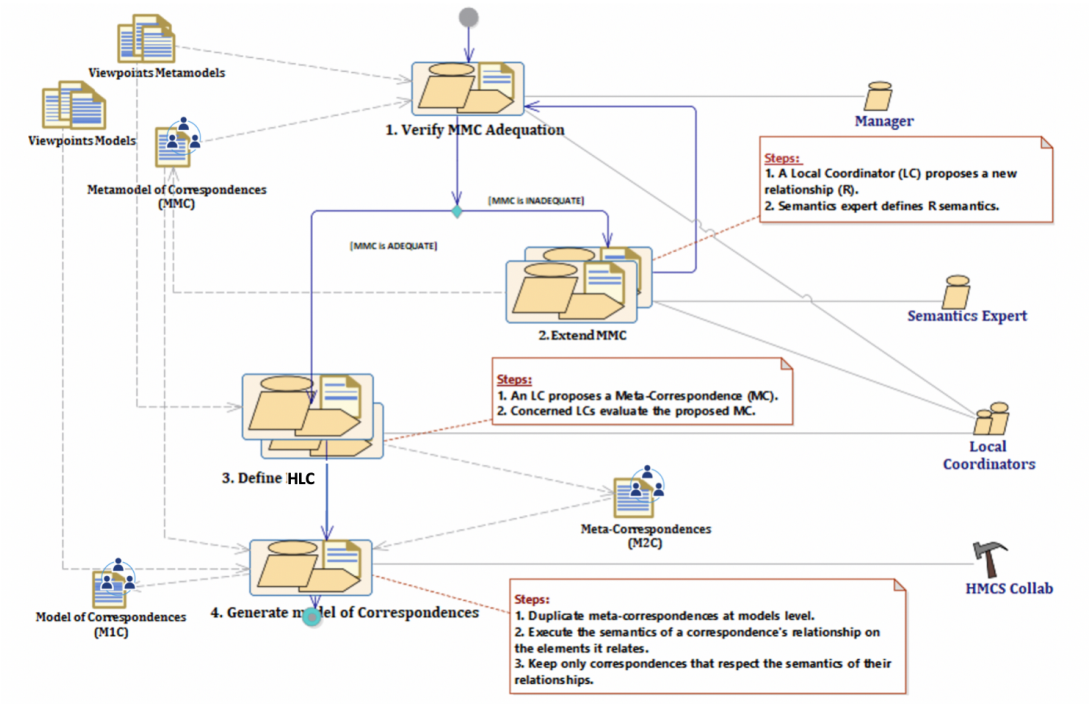


FIGURE 5.6 – Processus d'alignement collaboratif (modélisé avec CM-SPEM)

Ce processus a quatre activités majeures :

1. **Activité 1- Vérifier la couverture du MMC (*Verify MMC Adequation*)** Les coordinateurs locaux des points de vue du système vérifient individuellement l'adéquation du MMC générique au domaine d'application étudié.

2. **Activité 2- Etendre le MMC (*Extend MMC*)** Si le MMC générique ne suffit pas à décrire les correspondances possibles pour le domaine étudié, sa partie spécifique est spécialisée en fonction de la politique décisionnelle adoptée. Les coordinateurs locaux proposent de rajouter des DSR. Une fois qu'une relation a été proposée et validée, sa sémantique est mise en œuvre par l'expert en sémantique (dans le cas où une expression sémantique formelle peut lui être associée).

   En résumé, Extend MMC est une *Collaboration* au sens de MMCollab. Les *proposals* sont les nouvelles relations DSR à inclure dans le MMC et le MMC étendu est le *CollaborativeWorkProduct* résultant.

3. **Activité 3- Produire le M2C (*Produce M2C*)** Chaque coordinateur local peut proposer des correspondances potentielles qui impliquent des éléments de son méta-modèle. En utilisant le MMC et les méta-modèles des domaines métiers concernés, il précise le ou les méta-éléments impliqués dans chaque HLC proposée (éléments de son méta-modèle et des méta-modèles liés) et le type de relation qui les connecte. Une fois que ces correspondances sont validées par les autres coordinateurs locaux concernées, l'outil support à l'approche, HMCS-Collab (qui sera décrit dans la Section 5.5), combine ces évaluations pour générer le modèle de correspondance entre méta-éléments, M2C. La figure 5.7 montre le diagramme de cette activité. Il contient 2 types spécifiques de produits de la collaboration, *CollaborativeWorkProduct* : la liste des correspondances proposées (indexée par *P*) et le résultat de l'évaluation de ces propositions (indexée par *S*).

   L'activité *produceM2C* est la deuxième instance de *Collaboration* du processus de mise en correspondance. Les *proposals* sont les correspondances entre méta-modèles proposées et les *CollaborativeWorkProduct*s sont les correspondances approuvées.

4. **Activité 4- Produire le M1C (*Produce M1C*)** HMCS-Collab produit automatiquement le M1C en propageant les correspondances du niveau des méta-modèles, au niveau des modèles. Il génère d'abord pour chaque correspondance de niveau M2, le produit cartésien des instances de méta-éléments qui y sont impliqués, puis ne conserve que les correspondances qui respectent la sémantique associée à leurs relations, comme le fait HMCS. Dans le cas de relations avec expression informelle (en langage naturel), qui nécessiteraient lors du filtre l'intervention d'un superviseur (comme dans AHM) pour décider de conserver ou au contraire d'éliminer les correspondances concernées, une session collaborative sera initiée.

5.3.2 Application au système SU

Dans le processus d'alignement des modèles proposé, nous avons identifié deux activités collaboratives : (1) établir des relations (sémantiques) pour enrichir le MMC et (2) produire des HLC. Pour simplifier, nous supposons que quatre relations ont été utilisées pour décrire les correspondances du système de SU : Similarité, Généralisation, Induction et Déduction.

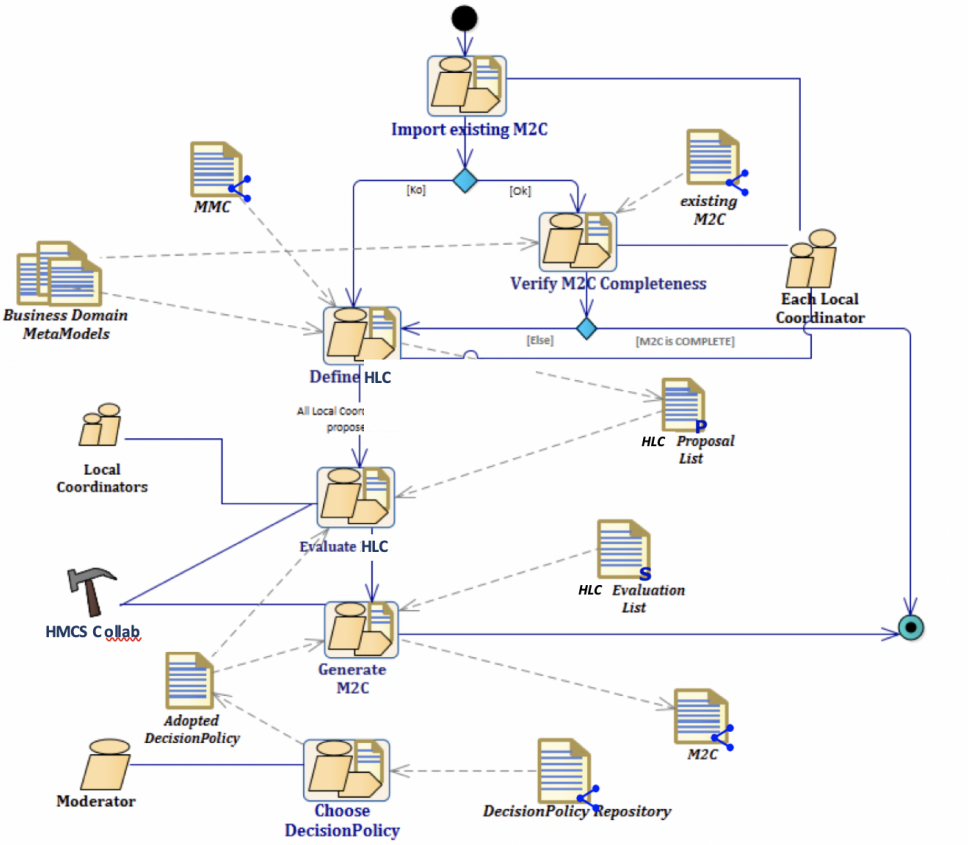


FIGURE 5.7 – Détails de l'activité produire le M2C (en CM-SPEM)

Le scénario de collaboration illustré ci-dessous est la production de HLC, où :

- SD_{LC}, BP_{LC} et ER_{LC} font respectivement référence aux coordinateurs locaux de SD, BP et ER.
- Une HLC est représentée à l'aide de la syntaxe suivante (où $\rightarrow$ est utilisée pour les relations asymétriques et $\leftrightarrow$ pour les relations symétriques) : *Relationship "[" metamodel " :" meta-element ($\rightarrow$ or $\leftrightarrow$) metamodel " :" meta-element "]"*
- Le tableau 5.2 résume les méta-correspondances proposées, leur initiateur et les décideurs (DM).

TABLE 5.2 – Proposed HLCs

N	Initiateur	High Level Correspondence	DM
1	BP_{LC}	Similarity [BP :Role $\leftrightarrow$ SD :Class]	SD_{LC}
2	SD_{LC}	Similarity[ER :Field $\leftrightarrow$ SD :Attribute]	ER_{LC}
3	BP_{LC}	Induction[BP :Activity, SD :Operation $\rightarrow$ ER :Field]	SD_{LC}, ER_{LC}
4	BP_{LC}	Generalization[BP :Role $\rightarrow$ SD :Class]	SD_{LC}
5	ER_{LC}	Deduction[ER :Field $\rightarrow$ SD :Attribute]	SD_{LC}

Une fois que les HLC ont été proposées, elles sont soumises à des évaluations par les décideurs éligibles. BP_{LC} est considéré comme le modérateur de la collaboration puisqu'il est

le premier acteur à initier une proposition. Supposons qu'il opte pour une politique de décision itérative, il doit donc choisir entre *ConsentingTogether* et *NegotiatingTogether*. Nous supposons dans la suite qu'il choisit la seconde.

HLC1, HLC2, HLC4 et HLC5 sont binaires, évaluées par un seul décideur, et ne nécessitent donc pas de faire l'objet d'un processus collaboratif.

L'évaluation collaborative de HLC3 est détaillée sur la Figure 5.8).

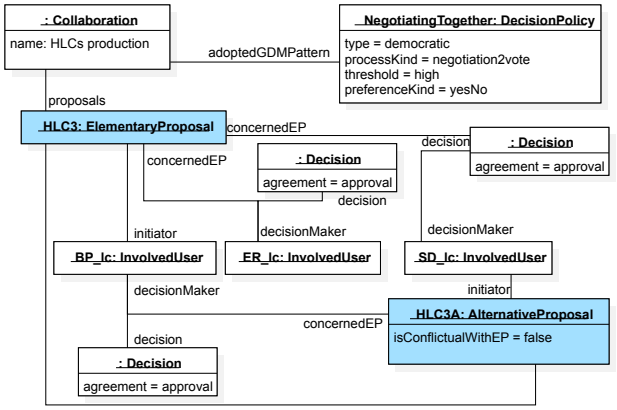
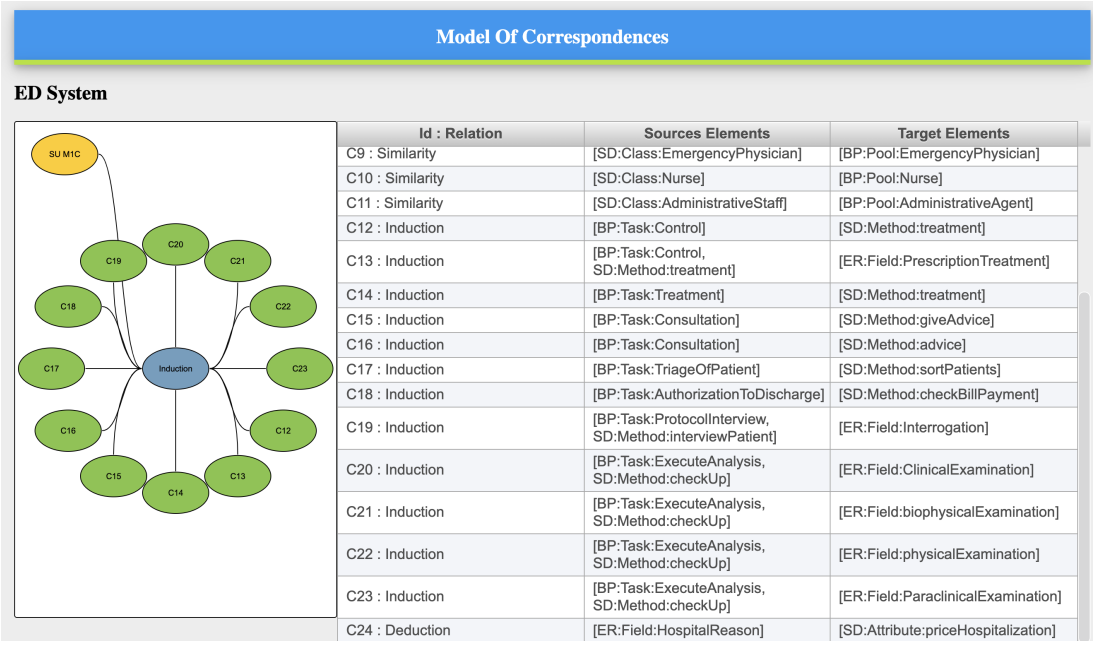


FIGURE 5.8 – Instanciation du package *Evaluation* de MMCollab

HLC3 est initié par BP_{LC}. On compte deux décideurs (SD_{LC} et ER_{LC}). ER_{LC} donne son accord sur HLC3 tandis que SD_{LC} l'affine par une proposition alternative (HLC3A) : *Induction[BP :Activité → SD :Opération]*. SD_{LC} est donc l'initiateur de HLC3A et BP_{LC} le décideur. Étant donné que HLC3A et HLC3 ne sont pas conflictuelles (comme l'a indiqué SD_{LC} lorsqu'il a défini HLC3A), HLC3 et HLC3A peuvent toutes deux être maintenues.
À l'issue de cette activité, toutes les HLC proposées sont approuvées.

Les LLC sont automatiquement dérivées des HLC. Seules les correspondances valides au regard de la sémantique de leur relation sont stockées dans le modèle de correspondance, dont un extrait est présenté Figure 5.9.



Id : Relation	Sources Elements	Target Elements
C9 : Similarity	[SD:Class:EmergencyPhysician]	[BP:Pool:EmergencyPhysician]
C10 : Similarity	[SD:Class:Nurse]	[BP:Pool:Nurse]
C11 : Similarity	[SD:Class:AdministrativeStaff]	[BP:Pool:AdministrativeAgent]
C12 : Induction	[BP:Task:Control]	[SD:Method:treatment]
C13 : Induction	[BP:Task:Control, SD:Method:treatment]	[ER:Field:PrescriptionTreatment]
C14 : Induction	[BP:Task:Treatment]	[SD:Method:treatment]
C15 : Induction	[BP:Task:Consultation]	[SD:Method:giveAdvice]
C16 : Induction	[BP:Task:Consultation]	[SD:Method:advice]
C17 : Induction	[BP:Task:TriageOfPatient]	[SD:Method:sortPatients]
C18 : Induction	[BP:Task:AuthorizationToDischarge]	[SD:Method:checkBillPayment]
C19 : Induction	[BP:Task:ProtocolInterview, SD:Method:interviewPatient]	[ER:Field:Interrogation]
C20 : Induction	[BP:Task:ExecuteAnalysis, SD:Method:checkUp]	[ER:Field:ClinicalExamination]
C21 : Induction	[BP:Task:ExecuteAnalysis, SD:Method:checkUp]	[ER:Field:biophysicalExamination]
C22 : Induction	[BP:Task:ExecuteAnalysis, SD:Method:checkUp]	[ER:Field:physicalExamination]
C23 : Induction	[BP:Task:ExecuteAnalysis, SD:Method:checkUp]	[ER:Field:ParaclinicalExamination]
C24 : Deduction	[ER:Field:HospitalReason]	[SD:Attribute:priceHospitalization]

FIGURE 5.9 – Extrait du modèle de correspondance du SU

5.4 Application à la gestion des changements du cas d'étude du Service d'Urgence

Dans le cadre de cette étude, il nous a d'abord semblé nécessaire de préciser les concepts inhérents à la gestion des évolutions des modèles, que ce soit pour capturer les changements, pour calculer et optimiser leurs impacts, ou encore pour traiter leurs impacts. Ces évolutions peuvent concerner des méta-modèles ou des modèles.

Elles peuvent être :

- l'ajout d'un point de vue (un méta-modèle et le modèle qui lui est conforme),
- la suppression d'un méta-modèle,
- la modification d'un méta-modèle (changements apportés à des éléments de méta-modèle),
- l'ajout d'un nouveau modèle, conforme à un méta-modèle existant,
- la suppression d'un modèle,
- la modification d'un modèle (changements apportés à des éléments de modèle).

Nous avons limité les évolutions à traiter car la cohérence interne d'un méta-modèle sort du cadre de cette étude. Ainsi, les changements d'éléments d'un méta-modèle ne sont pas supportés.

Si un méta-modèle subit des changements internes, nous considérerons qu'il y a suppression de ce méta-modèle puis ajout d'un nouveau méta-modèle.

Nous interdisons l'ajout d'un modèle qui serait conforme à un méta-modèle déjà impliqué dans un point de vue (i.e. déjà instancié) : les modèles sont hétérogènes (donc deux modèles ne peuvent pas être conformes au même méta-modèle), et tous les méta-modèles sont différents.

Toujours dans l'objectif de réduire les risques d'erreurs et de minimiser les efforts humains en (i) automatisant au maximum, (ii) offrant un support collaboratif, nous avons d'abord étendu MMC afin de mieux supporter la gestion de la cohérence des modèles alignés.

5.4.1 Méta-modèle de correspondance étendu

Les concepts nécessaires à la définition et à la gestion des correspondances entre les modèles sont rassemblés dans le méta-modèle de correspondance MMC présenté dans la Figure 5.10.

Le MMC peut être considéré comme constitué de deux sous-ensemble d'éléments :

- Un sous ensemble de concepts (à droite de la figure 5.10 - en bleu) est dédié à la mise en correspondances, comme que nous l'avons vu dans le Chapitre 4 précédent).
- Un deuxième sous-ensemble (à gauche de la figure 5.10 - jaune) est dédié à la gestion des évolutions des modèles et constitue l'extension apportée à MMC.

Un ***Repository*** conserve l'historique des évolutions des (méta-)modèles source. Il est composé d'un ensemble de versions ***ChangesVersion*** qui caractérise les changements opérés sur les modèles.

Chaque nouvelle version de modèle induit donc une nouvelle version de changement *ChangesVersion*, identifiée par un id et constituée d'un ensemble de changements (*Change*). Chaque changement est associé au modèle *RefModel* ou à l'élément de modèle *RefElement* concerné par

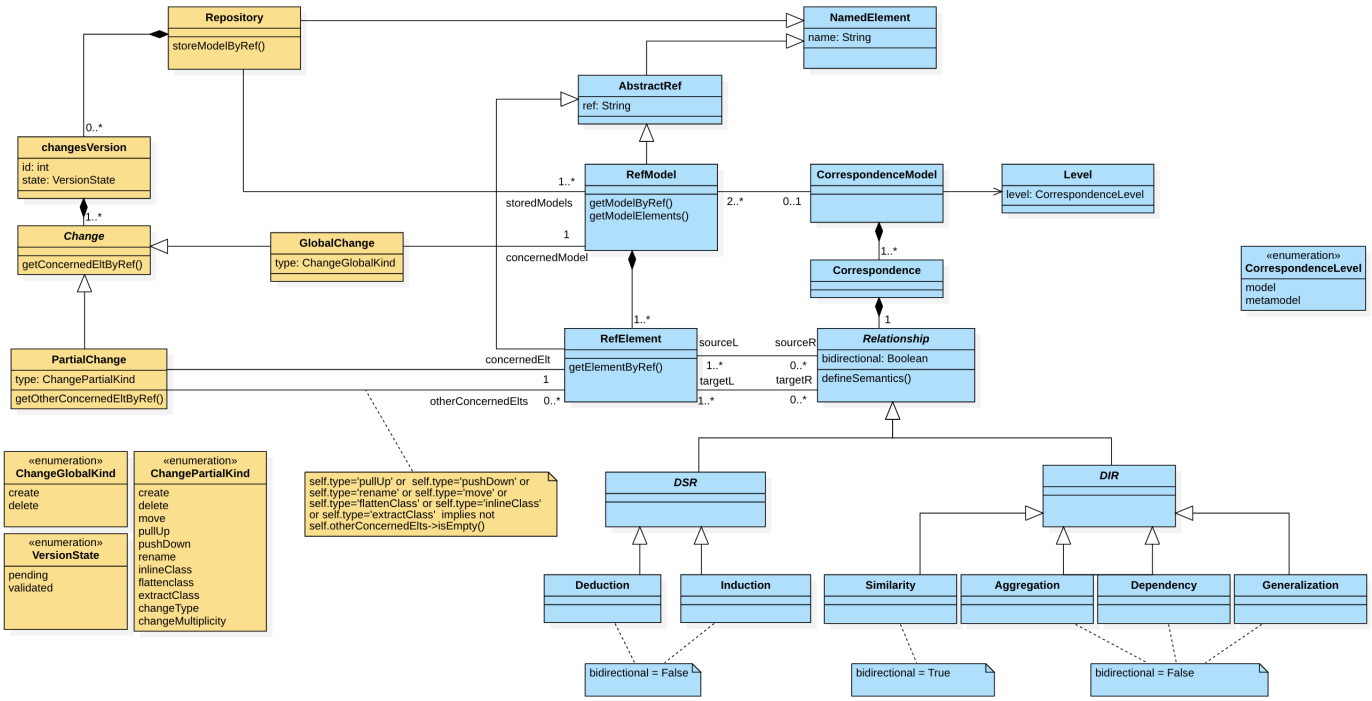


FIGURE 5.10 – MMC Etendu

ce changement. Un *ChangesVersion* possède un état *State*. Un état en attente (*pending*) signifie que l'impact des modifications apportées à un (méta-)modèle sur les autres (méta-)modèles n'a pas encore été analysé et considéré ; un état validé (*validated*) signifie que les impacts des modifications apportées à un (méta-)modèle sur les autres (méta-)modèles ont été analysés et traités.

Nous considérons que les changements (***Change***) peuvent être de deux types, global ou partiel et nous les traitons différemment selon ce type. Un changement global (***GlobalChange***) concerne la création ou la suppression d'un modèle ou d'un méta-modèle (*concernedModel*). Ses valeurs possibles sont répertoriées dans l'énumération *ChangeGlobalKind*.
Un changement partiel (***PartialChange***) concerne un élément de modèle (*ConcernedElt*). Les changements partiels sont répertoriés dans l'énumération *ChangePartialKind*.

Selon [108] et [128], les changements apportés à un modèle peuvent être :

- *Create* : ajout d'une propriété ou d'une classe,
- *Delete* : suppression d'une propriété ou d'une classe,
- *Move* : déplacement d'une propriété d'une classe dans une autre. Il s'agit d'un :
  - *pull up* si la propriété est déplacée vers une classe parente et d'un
  - *push down* si la propriété est déplacée vers une classe descendante.
- *Rename* : renommage d'un élément,
- *Extract class* : lorsqu'une classe devient en "surpoids" avec trop de méthodes et que son objectif n'est pas clair, Extract class permet la création d'une nouvelle classe et le déplacement de méthodes et de données de la première vers la seconde,
- *Inline class* : suppression d'une classe après avoir déplacé ses propriétés dans une autre classe,

- *Flatten class* : suppression d'une classe après déplacement d'un ensemble de ses propriétés dans ses sous-classes,
- *Change multiplicity* : changement de la multiplicité d'une association,
- *Change type* : changement du type d'une propriété.

Ces changements partiels peuvent être classés en changements atomiques (*create, delete, rename, change type, change multiplicity*) et changements composites (*move, extract class, inline class, flatten class*). Dans CAHM, nous traitons tout changement composite comme une suite de changements atomiques.

Pull up, *push down*, *extract class*, *inline class*, *flatten class* et *rename* impliquent plusieurs éléments : *concernedElt* et *otherConcernedElts*. Par exemple, le *pull up* d'un élément *i* d'une classe *A* vers une classe *B* est un changement partiel qui a *i* pour *concernedElt* et *A* et *B* pour *otherConcernedElts*. Le *rename* d'un élément *e* par *f*, a *e* pour *concernedElt* et *f* pour *otherConcernedElts*.
Nous ne traitons pas *change multiplicity*, puisque la multiplicité n'intervient pas dans l'établissement des correspondances.
Dans l'exemple du SU, quand on renomme Field :SocialSecNo par Field :PatientRecordNo, le *concernedElt* est SocialSecNo et l'*otherConcernedElts* est Field : PatientInsuranceNo.

Le processus proposé pour gérer l'alignement collaboratif de modèles hétérogènes évolutifs est basé sur ce méta-modèle de correspondance pour les aspects d'alignement et d'évolution d'une part, et sur la prise de décision en groupe collaborative d'autre part.

5.4.2 Processus collaboratif de gestion de la cohérence

Afin d'assurer que tout changement n'introduise pas d'incohérence au sein du système, nous avons proposé dans l'approche AHM de détecter et de catégoriser les changements puis de séparer ceux dont l'impact était "automatisable", des autres. Les changements dits "guidés" doivent faire l'objet d'un traitement manuel de la part de l'expert du système. Il lui est conseillé de s'appuyer sur les connaissances des concepteurs métiers, mais l'approche ne formalise pas ce processus. Forts de la modélisation de la prise de décision en groupe supportée par MMCollab, nous proposons d'intégrer un processus collaboratif de gestion des changements qui ne sont pas automatisables.

Ainsi, l'approche CAHM, propose un processus de gestion de l'évolution des modèle permettant de capturer les changements, de calculer leurs impacts, de les classer selon leur importance en termes d'impacts et de les traiter automatiquement quand c'est possible, via un processus collaboratif faisant intervenir toutes les parties prenantes, sinon.

Ce processus met en œuvre l'outil support HMCS-Collab dans les trois phases qui le constituent. Outre les *coordinateurs locaux* des différents points de vue du système, et l'*outil support*, un *manager* de la collaboration est également impliqué dans ce processus, décrit sur la figure 5.11 ci-dessous.

Ce processus passe par trois phases principales :

1) Détection des changements
2) Analyse des impacts des changements
3) Traitement des incohérences

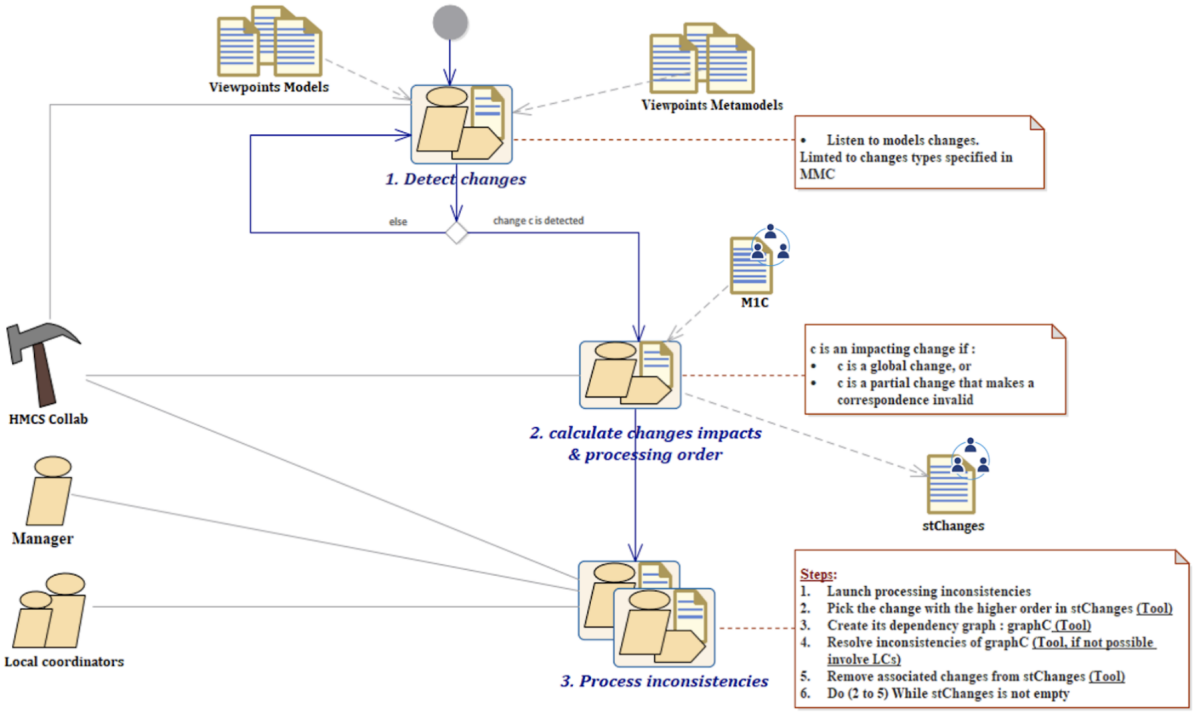


FIGURE 5.11 – Processus collaboratif de maintien de la cohérence

1. Phase de détection des changements

Les changements intervenant sur des éléments du modèle sont signalés à la volée. En effet, l'outil HMCS-Collab capture automatiquement les changements supportés par MMC (addition, suppression, mise à jour, pull-up, push-down), en se basant sur des identifiants et sur un mécanisme d'observation du répertoire (*Repository* du MMC) qui contient tous les changements.

Le système SU a subit les changements suivants qui ont été repertoriés : le modèle ER subit des changements partiels, le *field physio* a été spécialisé par quatre nouveaux *field*, *hormonalEx*, *immunologicalEx*, *serologicalEx*, *functionalEx* comme résumé sur la figure suivante.

La *Task :Control* du modèle BP a subit un changement de type renommage ; son nouveau nom est *medicalMonitoring*. Enfin, un nouveau point de vue a été ajouté au système. Il s'agit du point de vue organisationnel. Les méta-modèle et modèle ont été importés. Le manager précise l'acteur de la collaboration qui doit gérer ce point de vue.Ce changement étant un changement global, les changements détectés doivent être traités sans attendre. Nous passons donc à l'étape 2 du processus de CAHM-phase 2 (cf Fig 5.11).

2. Phase d'analyse des impacts des changements

Un changement a un ou plusieurs impacts s'il est :

- Un changement global ou
- Un changement partiel de type ajout ou
- Un changement partiel d'un élément utilisé dans au moins une correspondance qui provoque l'invalidité de l'une de ces correspondances au moins.

La validité des correspondances impliquant l'élément concerné doit donc être vérifiée pour chaque changement.

Pour ce faire, CAHM propose d'exécuter la partie opérationnelle de la sémantique des relations utilisée par les correspondances concernées, sur l'ensemble des éléments qu'elles connectent.

S'il y a des incohérences, des correspondances qui ne sont plus valides, les coordonnateurs locaux doivent décider ensemble de l'évolution de ces correspondances (phase 3) en fonction du changement en cours de traitement. Cette décision peut être difficile à prendre, surtout si l'élément concerné est impliqué dans plusieurs correspondances. C'est pourquoi, pour aider les coordonnateurs locaux à évaluer les répercussions potentielles de chaque changement, CAHM permet 1) de classer les changements dans un ordre décroissant de priorité de traitement, 2) de s'appuyer sur un ensemble de résolutions éprouvées stockées dans un catalogue, 3) de mettre en œuvre un processs collaboratif de prise de décision.

L'ordre dans lequel les traitements sont faits dépend des types respectifs des changements :

- Un changement global est traité en priorité car il change la structure du point de vue métier.
- Un changement partiel de type ajout n'est pas prioritaire car il n'a aucune incidence sur le modèle de correspondance déjà établi.
- L'ordre de priorité de traitement d'un changement partiel de type modification ou suppression correspond au nombre de correspondances qu'il rend invalides.

Un algorithme permettant de calculer l'ordre des traitements à opérer, basé sur un calcul d'impacts, est présenté sur la figure 5.12.

```
   Input      : c // a change
                stChanges // stack of changes
   Output     : stChanges // stack of changes
 1 if c.type == 'Global' then
 2 |  stChanges.insertAtTop(c) // c has the highest priority
 3 else // c.type == 'Partial'
 4 |  if c.subtype != 'A' // c.subtype == 'D' or c.subtype == 'M'
   |   then
 5 |  |  ce = c.getChangedElement()
 6 |  |  ce_corresps = getCorresps(ce) // get related elements to
   |  |   ce in M1C (directly and indirectly)
 7 |  |  for each corresp in ce_corresps do
 8 |  |  |  if verifySemantics(corresp) == False then
 9 |  |  |  |  ce.getInvalidCorresps().add(corresp)
10 |  |  |  end
11 |  |  end
12 |  |  if ce.getInvalidCorresps().size() > 0 then
13 |  |  |  stChanges.position(c) // Put c in its right position
   |  |  |   and Shift the rest of higher changes by a position
   |  |  |   to the top; a change c1 has higher priority than c
   |  |  |   if c1.getInvalidCorresps()>c.getInvalidCorresps()
14 |  |  end
15 |  else // c.subtype == 'A'
16 |  |  stChanges.insertAtBottom(c)// c has the lowest priority
17 |  end
18 end
```

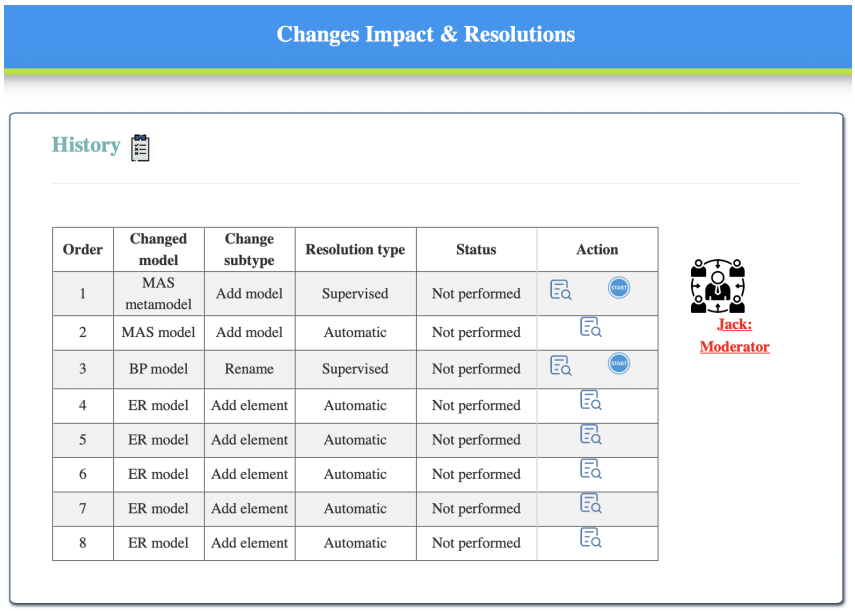
FIGURE 5.12 – Algorithme de calcul de priorité de traitement des changements

Les traitements à effectuer sont empilés dans une pile de traitements (*stChanges*) selon l'ordre décroissant des priorités qui leur ont été affectées.

L'état de la *ChangesVersion* est validé pour tout changement non impactant (state = validated).

3. Phase de traitement des changements

Une fois les correspondances non valides identifiées et les priorités de traitement des changements calculées, ces correspondances doivent être mises à jour. Elles sont traitées selon l'ordre décroissant des priorités calculées précédemment. La Figure 5.13 illustre la liste ordonnée des changements du système SU.



Order	Changed model	Change subtype	Resolution type	Status	Action
1	MAS metamodel	Add model	Supervised	Not performed	
2	MAS model	Add model	Automatic	Not performed	
3	BP model	Rename	Supervised	Not performed	
4	ER model	Add element	Automatic	Not performed	
5	ER model	Add element	Automatic	Not performed	
6	ER model	Add element	Automatic	Not performed	
7	ER model	Add element	Automatic	Not performed	
8	ER model	Add element	Automatic	Not performed	

FIGURE 5.13 – Interface de classement des impacts des changements détectés.

Le système SU a subi des changements pour répondre aux feedbacks des acteurs ayant mis en œuvre la phase 1. Ainsi, le *Field physio* du modèle *ER* a été spécialisé en ajoutant les *fields hormonalEx*, *immunologicalEx*, *serologicalEx* et *functionalEx* comme résumé sur la Figure 5.14. Ces quatre *Field* correspondent à des spécialisations d'examens physiologiques.

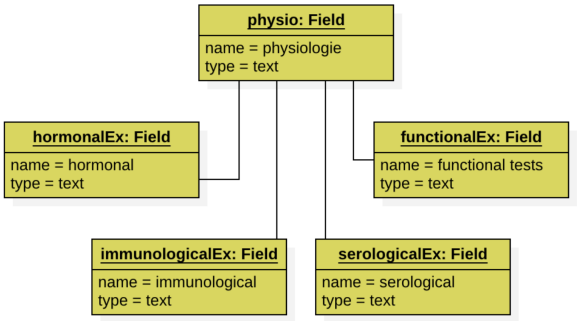


FIGURE 5.14 – Spécialisation de *Field Physio* lors de l'évolution du modèle ER.

Le modèle BP subit aussi un changement. La *Task :Control* est renommée, et son nouveau nom est *MedicalMonitoring*.

En plus de ces changements partiels, Jack, le modérateur, ajoute un nouveau point de vue pour compléter la représentation du système. Ce point de vue offre un degré élevé de simultanéité et d'appropriation décentralisée des tâches, des informations et des ressources impliquées dans un processus contrairement à ce qui est fait dans le point de vue processus. Il a pour objectif d'optimiser le flux des patients, d'ordonnancer l'activité du processus de soin et de construire des plannings pour le personnel médical. Ce méta-modèle de système multi agents (MAS) est décrit sur la Figure 5.15.

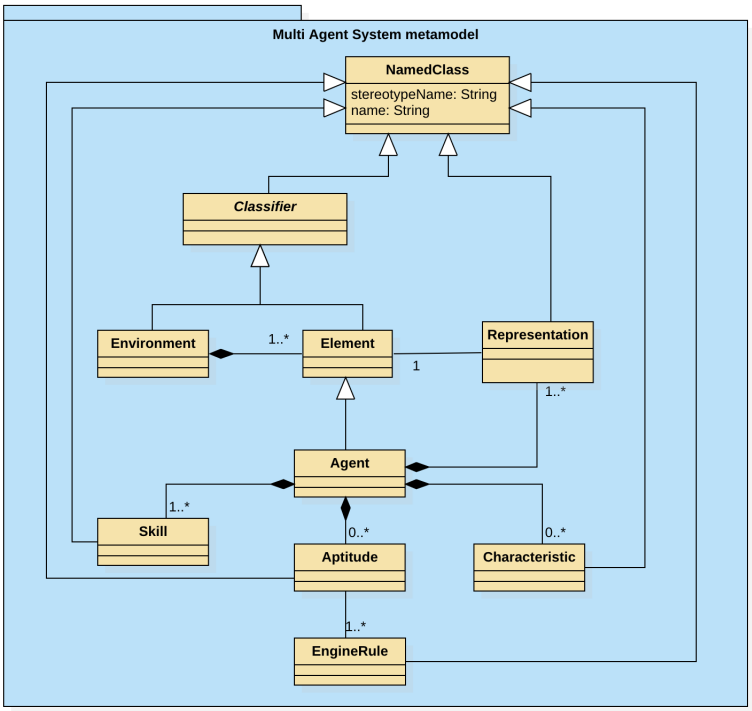


FIGURE 5.15 – Méta-modèle du point de vue MAS

L'instanciation d'un modèle de MAS garantit à terme une prise en charge rapide et de qualité, tout en planifiant de façon semi-automatique les ressources du système hospitalier. Le modèle ajouté au SU, présenté sur la Figure 5.16 est basé sur le travail de thèse de [9].

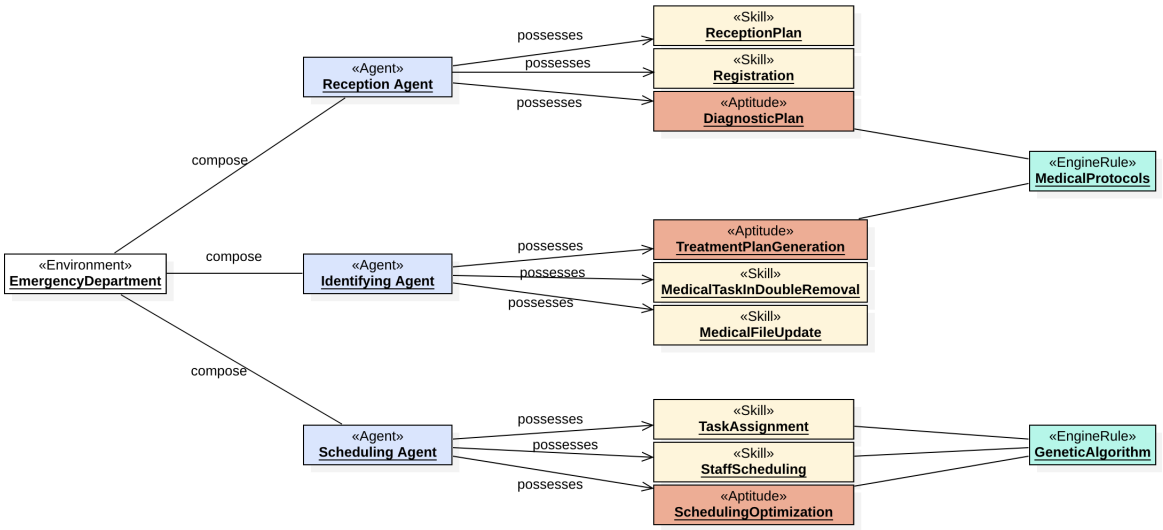


FIGURE 5.16 – Modèle du point de vue MAS

Jack importe le méta-modèle et le modèle du point de vue et choisit Dan comme acteur responsable de ce point de vue.

Les modèles source sont verrouillés durant l'activité de traitement pour que les coordinateurs locaux n'apportent pas d'autres changements susceptibles d'interférer avec les modifications en cours.

La Figure 5.17 illustre l'enchaînement des étapes de cette activité (sans mentionner les artefacts en entrée/sortie).

Le traitement des incohérences est lancé automatiquement en cas de changement global

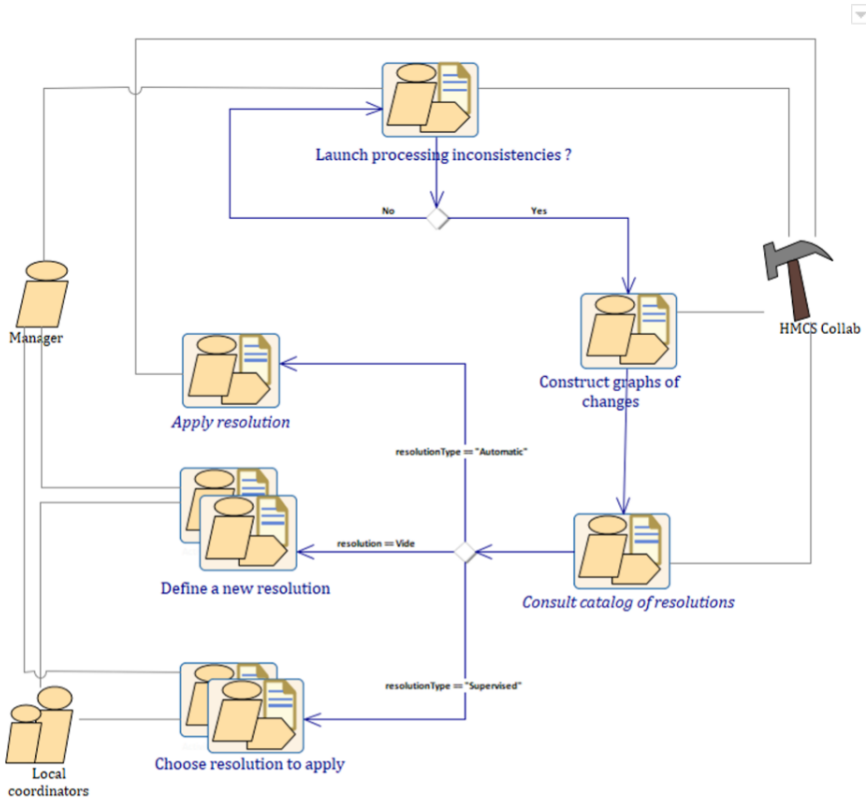


FIGURE 5.17 – Traitement des changements

(qu'il ait lieu au niveau méta-modèle ou au niveau modèle). Dans le cas d'un changement partiel, le traitement des incohérences est déclenché par le manager, généralement à la suite d'une requête d'un des coordinateurs locaux qui estime qu'il vient d'apporter des changements partiels importants au modèle qui le concerne.

Une fois que le traitement est lancé, HMCS-Collab récupère le changement ayant le plus grand impact. Afin d'identifier les impacts du changement considéré, un graphe de ses dépendances est alors établi dans certains cas (***Construct graph of changes***).

Graphe des dépendances d'un changement

Ce graphe a pour objectif de tracer les éléments directement ou indirectement reliés à l'élément changé (qu'il soit un modèle, un méta-modèle ou un élément de modèle) et d'identifier les correspondances concernées. Il permet de relever les incohérences (les correspondances qui ne sont plus valides) et sert de support à la phase de résolutions de ces incohérences.

Le graphe des dépendances d'un changement partiel est construit différemment du graphe des dépendances d'un changement global. Par ailleurs, tous les changements ne nécessitent pas la construction de ce graphe :

- Dans le cas d'un changement partiel dû à un élément modifié E_c, par exemple, le graphe a pour racine E_c et pour arcs les correspondances impliquant E_c, directement ou indirectement. C'est à dire que les noeuds du graphe sont des triplets constitués d'un élément de modèle, de sa distance à la racine et de l'ensemble des correspondances "suivies".

  Dans l'exemple ci-dessous (figure 5.18), un M1C est défini entre trois modèles dont les éléments sont nommés respectivement avec les alphabets latin, grec et cyrillique. C1, C2, C3 et C4 sont les noms des correspondances. α, ë, γ, л d et θ sont les éléments reliés par ces correspondances. Considérons le graphe des dépendances de c. c, α, ë et γ étant

directement liés à c, ils sont à une distance de 1. л est à une distance de 2 puisqu'il est indirectement lié à c alors que d et θ ne sont pas inclus dans le graphe de c. Le graphe des dépendances de c est : graphC(c)= (c,0,null),(α, 1,C1),(ë,1,C1), (γ, 1,C2),(л,2, {C2,C3})

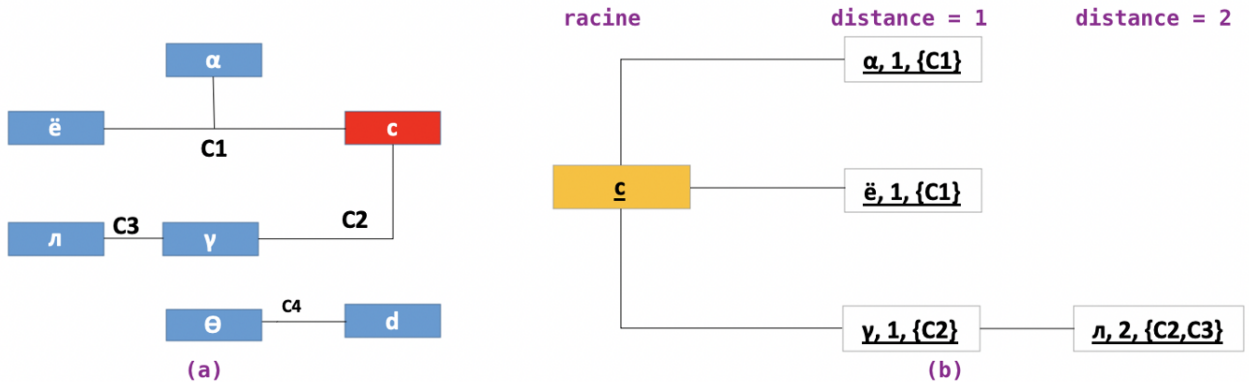


FIGURE 5.18 – Exemple de graphe de dépendances : (a) : M1C - (b) GraphC(c)

- Dans le cas d'un changement global de type suppression, le graphe est constitué des graphes de tous les éléments du modèle supprimé.
- Dans le cas d'un changement de type ajout, qu'il soit global ou partiel, il n'y a pas de graphe, puisque l'élément n'est impliqué dans aucune correspondance existante.

Ainsi, pour les changements considérés du SU, on construit, entre autres, le graphe suivant (Figure 5.19).

La racine du graphe est l'élément *Task :Control* du modèle BP. Le graphe comporte deux éléments qui sont reliés à *Task :Control* dans le M1C, *Method :treatment* et *Field :Prescription-Treatment*. Pour chaque noeud du graphe, l'interface précise le nom du noeud, sa distance à la racine et les correspondances entre la racine et l'élément considéré au niveau du noeud. (L'extrait de l'interface de HMCS_Collab de la figure 5.19, montre que celui fournit au modérateur de la collaboration la possibilité de visualiser les détails des correspondances impliquées dans ce graphe.)

Une fois que tous les graphes nécessaires ont été créés, le traitement des changements démarre, dans l'ordre des priorités.

Traitement des changements : Résolution

- Dans le cas particulier d'un changement global de type suppression, le traitement est automatique : Toutes les correspondances situées sur les arcs des graphes sont supprimées du M1C. Si c'est un méta-modèle, toutes les HLCs qui connectent les éléments de ce méta-modèle avec les autres méta-modèles sont aussi supprimées.
- Un ajout quant à lui nécessite une nouvelle mise en correspondance, collaborative. Celle-ci se fait de manière incrémentale, en apportant les modifications utiles au méta-modèle de correspondance et au M2C existant si c'est un méta-modèle qui a été ajouté (i.e. en considérant les HLC impliquant le nouvel élément), puis en générant les HLC pour les modèles instances. Si un modèle ou un élément de modèle ont été ajoutés, le M1C est automatiquement régénéré.

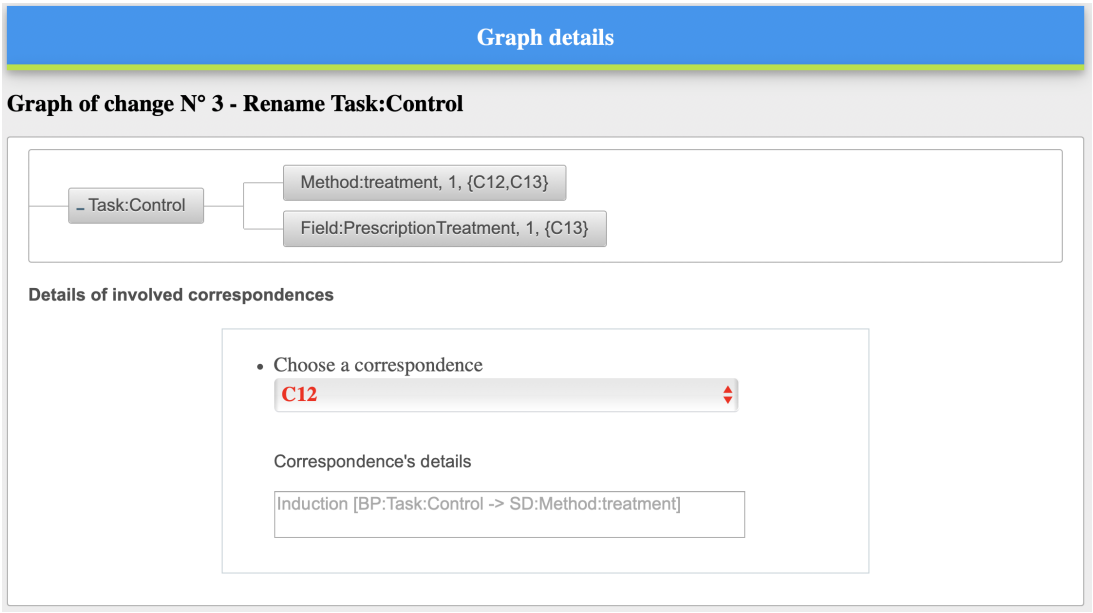


FIGURE 5.19 – Interface de visualisation du graphe de dépendances de *Task :Control*.

- Dans le cas général d'un changement partiel, le traitement d'un changement passe par la résolution des incohérences contenues dans son graphe des dépendances. HMCS-Collab identifie d'abord les correspondances contenues dans le graphe qui ne sont plus valides.

Dans l'exemple du SU, l'incohérence suivante a notamment été identifiée :
La co-existence de C12 : Induction [BP :Task :Control -> SD :Method :treatment] et de C13 : Induction [BP :Task :Control, SD :Method :treatment -> ER :Field :PrescriptionTreatment].

Ainsi, le manager de la collaboration et les coordinateurs locaux ont connaissance des incohérences à résoudre. Ils s'appuient sur ce graphe pour identifier et décider des résolutions nécessaires, afin d'obtenir un graphe correct à la fin du processus de traitement.

Ils ont également à disposition un catalogue de résolutions prédéfinies. L'étape suivante consiste à vérifier l'existence de solutions appropriées dans ce catalogue (***Consult catalog of resolutions***).

Ce catalogue contient une liste extensible de résolutions classées en fonction du type de changement.

Dans cette étape du processus, HMCS-Collab recherche d'abord les solutions appropriées dans le catalogue de résolutions prédéfinies. Il les propose en support à la prise de décision sur les solutions appropriées. Certaines résolutions présentes dans ce catalogue permettent une mise en œuvre automatique. C'est le cas notamment d'un changement global de type suppression où les graphes de tous les éléments du modèle supprimé ont été construits et où toutes les correspondances situées sur les arcs de ces graphes sont supprimées du M1C. Si c'est un méta-modèle, toutes les HLCs qui connectent les éléments de ce méta-modèle avec les autres méta-modèles sont aussi supprimées.

De même, un ajout nécessite automatiquement une nouvelle mise en correspondance.

Le catalogue des résolutions est détaillé dans l'Annexe 5.b. de la thèse de Saloua Bennani, l'extrait ci-dessous (Table 5.3) donne un aperçu de son contenu et de sa structure sur les exemples d'un changement global intervenu au niveau M2, et d'un changement global de niveau M1.
La colonne *Résolution* résume les résolutions possibles, tandis que la colonne *Type de résolution* précise la nature de chaque résolution (i.e., *automatique* ou au contraire *supervisée*).

Les résolutions ***automatiques*** sont opérées par l'outil HMCS-Collab (***Apply resolution***). Les résolutions ***supervisées*** sont les résolutions qui nécessitent une intervention humaine. En effet, selon le cas de figure, soit elles présentent plusieurs solutions possibles pour un même changement et nécessitent alors une sélection (l'opérateur « or » entre au moins deux résolutions exprime les alternatives possibles), soit elles ne proposent aucune solution pré-déterminée.

TABLE 5.3 – Extrait du Catalogue de résolutions des incohérences

Change			**Resolution**	**Resolution**
Level	**Type**	**Subtype**		**Type**
Metamodel	Global	Delete MM1	(**RM1** or **RM2**) and **R1**, with : **RM1** : Remove meta-correspondences (MM1) **R1** : Remove correspondences(M1) **RM2** : Adjust meta-correspondances (MM1)	 Automatic Automatic Supervised
		Add MM1	**RM3** : Define meta-correspondences (MM1)	Supervised
		Delete + add MM1	**RM1** or **RM2** or **RM3**	Supervised
Model	Global	Delete M1	**R1** : Remove correspondences(M1)	Automatic
		Add M1	**RM4.0** : Propagate meta-correspondences (MM1)	Automatic
		Delete + add M1	**R1** and **RM4.0**	Automatic

Quand plusieurs résolutions sont proposées au catalogue, la résolution à adopter doit être choisie collaborativement. Quand aucune solution n'est prédéfinie, la résolution doit être proposée collaborativement. Les coordinateurs locaux et le manager sont donc impliqués pour choisir (***Choose resolution to apply***) ou proposer des solutions possibles et décider de la résolution à appliquer (***Define a new resolution***), et à ajouter au catalogue.

Les *proposals* pour cette collaboration sont les résolutions proposées par les collaborateurs. La décision de groupe consiste à trancher sur la résolution à appliquer. Le *collaborativeWorkProduct* potentiel en est le catalogue des résolutions alimenté d'une nouvelle résolution.
Le traitement des changements se poursuit jusqu'à ce que la pile des changements à traiter soit vide.

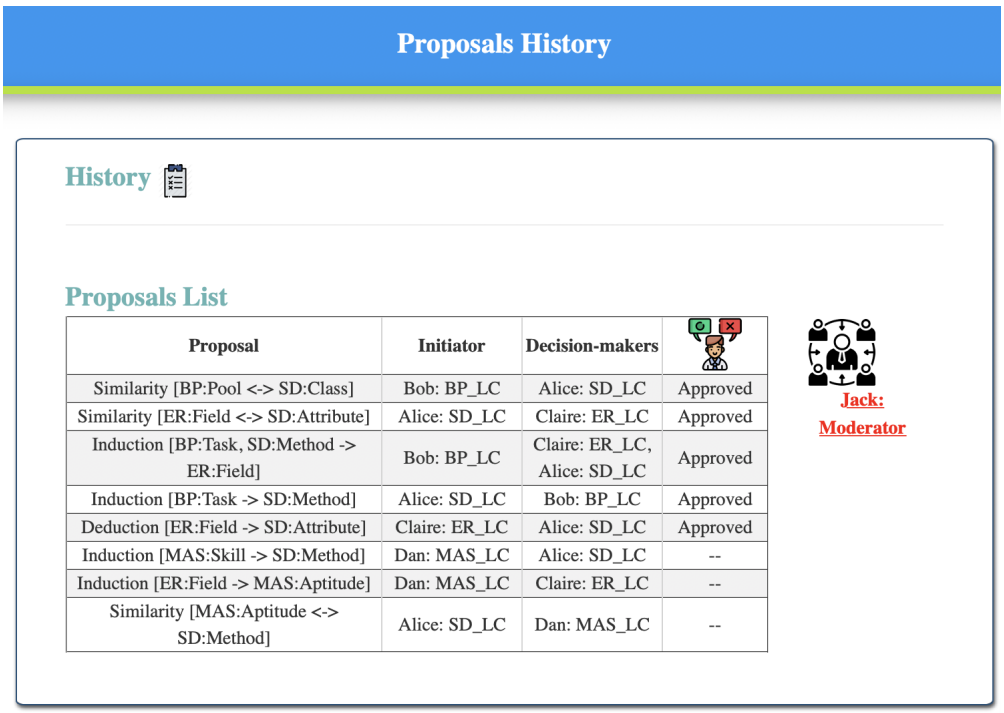


FIGURE 5.20 – Liste des méta-correspondances après l'ajout du point de vue MAS, et avant leur évaluation.

Dans l'exemple du SU, la collaboration a impliqué, dans l'ordre des traitements réalisés (cf Figure 5.13 :

(1) Ajout du méta-modèle MAS.

L'ajout d'un nouveau point de vue et du méta-modèle *MAS* (Fig. 5.15 implique de proposer de nouvelles correspondances au niveau méta et d'évaluer les éventuels conflits que cet ajout pourrait créer.

La Figure 5.20 illustre les méta-correspondances considérées pour le système SU. Les cinq premières méta-correspondances ont été approuvées lors de la phase 1 et ont été conservées car l'ajout de nouvelles méta-correspondances n'a pas provoqué de conflit. (La collaboration concernant les méta-correspondances nouvellement proposées a suivi la politique de décision **Consenting Together** et les trois propositions ont été approuvées suite à l'évaluation).

(2) Ajout du modèle MAS.

L'ajout du modèle *MAS* (Fig. 5.16) implique la propagation des méta-correspondances qui viennent d'être ajoutées et la suppression des correspondances issues des méta-correspondances non retenues. Aucune méta-correspondance de la phase 1 n'a été écartée. Ainsi, l'ajout du modèle *MAS* revient à propager uniquement les trois méta-correspondances nouvellement approuvées. Cette action est faite automatiquement par l'outil HMCS-Collab.

La propagation des trois méta-correspondances donne neuf correspondances ajoutées au M1C :

- Induction [MAS :Skill :ReceptionPlan $->$ SD :Method :feedPatientPersonalData]
- Induction [MAS :Skill :MedicalFileUpdate $->$ SD :Method :addMedicalHistory]
- Induction [MAS :Skill :MedicalFileUpdate $->$ SD :Method :addDisease]

- Induction [MAS :Skill :MedicalFileUpdate − > SD :Method :feedPatientMedical-History]
- Induction [MAS :Skill :MedicalFileUpdate − > SD :Method :addAllergy]
- Induction [MAS :Skill :TaskAssignment − > SD :Method :generateSchedule]
- Induction [MAS :Skill :StaffScheduling − > SD :Method :generateSchedule]
- Induction [ER :Field :HospitalReason − > MAS :Aptitude :DiagnosticPlan]
- Induction [ER :Field :HospitalReason − > MAS :Aptitude :TreatmentPlanGeneration]

(3) Renommage de l'attribut *Task :Control* du modèle BP

Le renommage de *Task :Control* nécessite une collaboration pour décider du traitement des incohérences qu'il a engendrées (incohérence des correspondances C12 et C13). Le catalogue de résolutions [1] envisage deux résolutions pour ce type de changement, R2 et R6, avec *R2* : *remove correspondences(e)* et *R6* : *maintain correspondences(e)*.

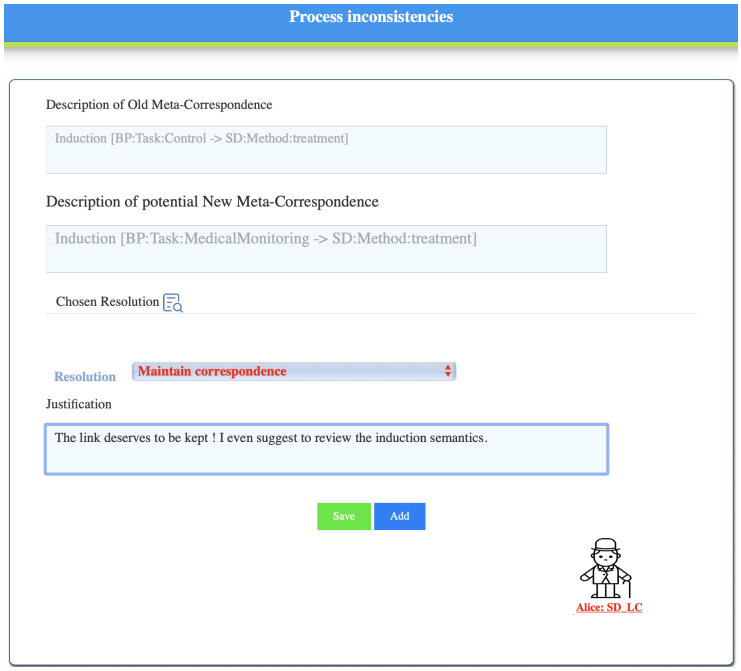


FIGURE 5.21 – Choix de la résolution à appliquer pour le changement *rename Task :Control*.

Le graphe de dépendances de ce renommage contenant des éléments issus des modèles *SD* et *ER* en plus du modèle *BP*, la collaboration implique les coordinateurs locaux de ces trois modèles : Bob, Alice et Claire. La Figure 5.21 illustre le choix de résolution effectué par Alice. Elle choisit de maintenir la correspondance C12 et explique son choix aux autres concepteurs. De ce fait C12 et C13 sont maintenues avec une nouvelle source, *Task :MedicalMonitoring*.

(4) Ajout de quatre *Field* au modèle ER : quatre changements.

Les ajouts des *Field HormonalEx*, *ImmunologicalEx*, *SerologicalEx*, et *FunctionalEx* sont traités de façon supervisée. En effet, le catalogue de résolutions prévoit de supprimer la correspondance impliquant la classe parente et d'étudier la possibilité de forcer le lien pour ses sous-classes.

1. cf. Annexe 5.b de ref thèse Saloua

Le *Field physio* est impliqué dans la correspondance Induction [BP :Task :ExecuteAnalysis, SD :Method :checkUp -> ER :Field :physicalExamination].

La gestion de l'ajout de ses sous classes dans le modèle ER, nécessite l'accord des trois coordinateurs locaux concernés, Bob, Alice et Claire. Ils collaborent pour prendre les décisions qui s'imposent, notamment au sujet du modèle ER :

(a) Une *induction* doit-elle être définie entre *BP :Task :ExecuteAnalysis*, *SD :Method :checkUp* et *ER :Field :HormonalExamination* ?

(b) de même pour chacune des sous-classes de *ER :Field :Physio* créée ?

(c) Faut-il ajouter des entités ad-hoc dans les modèles PB et SD ?

(d) Si oui, une *induction* doit-elle être définie entre *ER :Field :HormonalExamination* et les entités adaptées ajoutées aux deux autres modèles ?

(e) etc.

Finalement, le procesuss de collaboration a abouti à une décision finale par consensus (selon les préconisations du catalogue de résolutions) : La correspondance *induction* est forcée avec les sous-classes de *Physio* ; ce qui enrichit le M1C des correspondances :

- Induction [BP :Task :ExecuteAnalysis, SD :Method :checkUp -> ER :Field :HormonalExamination] ;
- Induction [BP :Task :ExecuteAnalysis, SD :Method :checkUp -> ER :Field :ImmunologicalExamination] ;
- Induction [BP :Task :ExecuteAnalysis, SD :Method :checkUp -> ER :Field :SerologicalExamination] ;
- Induction [BP :Task :ExecuteAnalysis, SD :Method :checkUp -> ER :Field :FunctionalExamination].

5.5 Tool support : Module d'aide à la décision

HMCS-Collab enrichit HMCS d'un support à la collaboration. HMCS est constitué d'un ensemble de modules assurant la correspondance, la gestion de la cohérence et la transformation des modèles. Pour soutenir l'alignement collaboratif des modèles, nous avons ajouté deux modules : *Outil de collaboration (CollabT)* et *Outil de prise de décision (DMT)*. Je mettrai l'accent ici sur le module DMT (Decision Making Tool) qui est dédié à la prise de décision en groupe (GDM). Le module DMT (Figure 5.22) permet de produire une décision collaborative pour une proposition donnée en exploitant les données des utilisateurs (*UDB*), les politiques décisionnelles mises en œuvre (*DMP*), les propositions (*PDB*) et leurs évaluations (*EDB*).
Quatre gestionnaires ont accès à ces quatre bases de données.

- *UDB Extractor* extrait pour chaque proposition (1.a), la liste des utilisateurs concernés (1.b).
- Cette liste est ensuite transférée à *Notification center* (2.a) qui notifie les utilisateurs concernés (2.b).

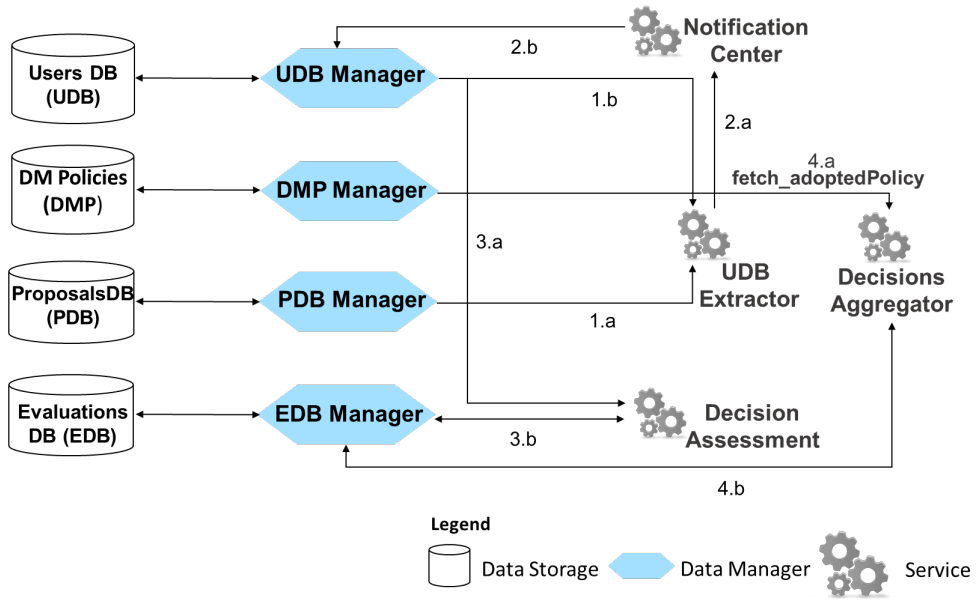


FIGURE 5.22 – Decision Making Tool (DMT)

- Ces utilisateurs évaluent alors individuellement les propositions et fournissent des décisions (3.a) via le service *Decision Assessment*. Ces décisions modifient l'EDB via le gestionnaire EDB (3.b).
- Enfin, le service *Decisions Aggregator* produit une décision de groupe en combinant les décisions individuelles (4.b) selon la politique décisionnelle adoptée (4.a).

La Figure 5.23 résume l'architecture globale de HMCS-Collab. Elle inclut à la fois les modules développés et les technologies utilisées.

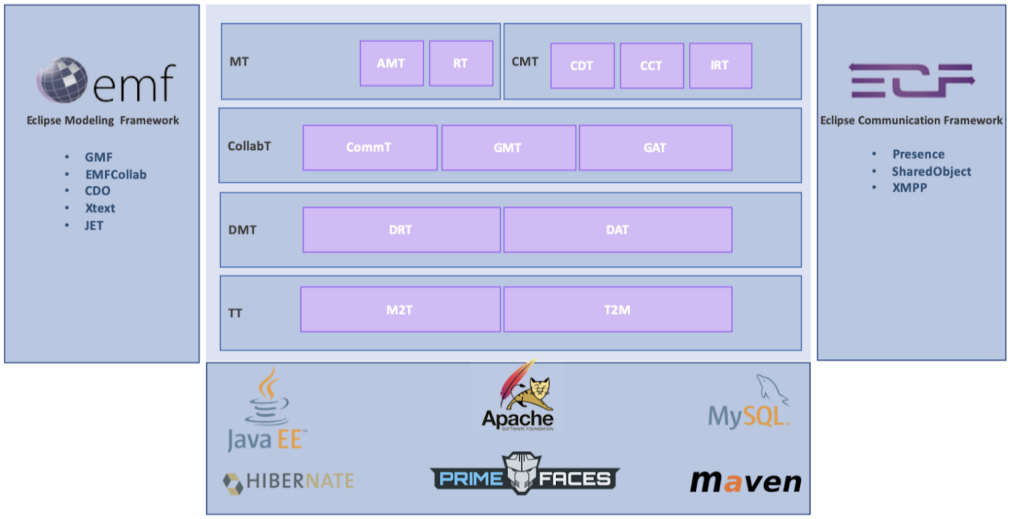


FIGURE 5.23 – Architecture globale de HMCS-Collab

On y retrouve les modules HMCS (4) et les couches rajoutées pour supporter la collaboration :

- Decision Making Tool (DMT) : contient un ensemble de politiques de prise de décision et la mise en œuvre de leurs méthodes d'agrégation. Il est divisé en deux sous-modules : (1) Decision policies Repository Tool ((DRT) qui met en œuvre des politiques de prise de

décision et (2) Decision Aggregator Tool (DAT) qui agrège les évaluations individuelles (en effectuant des calculs tels que somme, moyenne, etc.) pour obtenir les décisions collectives concernant les propositions.

- Collaboration Tool (CollabT) : assure les mécanismes de collaboration (par exemple, la communication et la gestion de groupe, et la sensibilisation du groupe, ...) via un outil de communication (CommT), un outil de gestion de groupe (GMT) et un outil de sensibilisation de groupe (GAT) ;

HMCS-Collab s'appuie notamment sur EMF-Collab, solution Eclipse légère permettant à plusieurs utilisateurs de modifier simultanément un même modèle EMF.

Les détails de l'implantation de l'outil sont dans le chapitre 6 de la thèse de Saloua Bennani [24]

5.6 Conclusion

Nos travaux de recherche portent sur la conception de systèmes complexes dans un contexte multi-vues, donc multi-modèles hétérogènes, et multi-utilisateurs, au sens "stakeholders". Dans ce cadre, la cohérence entre les modèles partiels constituant le modèle global du système, et le maintien de cette cohérence quand les modèles connectés évoluent, ne peut reposer sur le seul rôle d'un expert du domaine d'application.

La collaboration est un critère clé pour garantir l'exactitude de l'alignement des modèles partiels. En effet, un alignement peut être considéré comme plus précis s'il prend en compte tous les besoins des parties prenantes concernées. Pour cela, nous avons décidé d'étendre l'approche centralisée AHM en y ajoutant une perspective collaborative dés la phase d'alignement des modèles.

Par ailleurs, lorsqu'un modèle évolue, l'impact sur les autres modèles peut être important et nécessite une étude poussée pour calculer son incidence sur la cohérence du modèle global. Si un expert du domaine peut avoir une vision assez large pour prendre les décisions qui s'imposent en termes d'impacts sur les autres modèles partiels et les relations qui lient les différents modèles concernés, une aide précieuse peut lui être apportée par les concepteurs de ces modèles, qui ont une connaissance plus précise de leur métier. C'est pourquoi il nous a semblé important de donner également une dimension collaborative au processus de maintien de la cohérence du système.

L'approche CAHM est fondée sur un processus de prise de décisions en groupe. Elle permet la communication et la coordination entre les intervenants de plusieurs secteurs d'activités afin d'établir des liens sémantiques entre les concepts de leurs modèles et de maintenir ces liens. Son apport est triple.

(i) La première contribution est celle d'un méta-modèle de collaboration et de patrons de prises de décision en groupe, qui permettent de définir des politiques décisionnelles configurables, à appliquer et applicables (i) en phase de mise en correspondances des modèles et (ii) lors de la gestion des évolutions de ces modèles

Cette approche supporte la communication et la coordination entre les intervenants de plusieurs secteurs d'activités. Elle leur permet (i) d'établir des liens sémantiques entre les concepts de leurs modèles, (ii) de décider collectivement des incidences des changements apportés.

(ii) Nous avons pu ainsi proposer un processus collaboratif de mise en correspondance des modèles qui fait intervenir les parties prenantes quand nécessaire pour garantir une meilleure qualité des connections établies entre les modèles partiels. Ceci constitue la deuxième contribution.

(iii) La troisième contribution est relative au maintien de la cohérence des modèles partiels et du modèle global. Nous proposons un second processus permettant (1) d'identifier automatiquement les changements réalisés (2) de les classifier en fonction de l'importance de leurs répercussions éventuelles et (3) de proposer des traitements adaptés, voire de les réaliser automatiquement. Ce processus est supporté par un mécanisme permettant, dans le cadre d'une extension du méta-modèle de correspondance que nous avons proposé, de conserver un historique de tous les changements apportés aux modèles connectés.

Ces processus sont outillés par HMCS_Collab qui intègre la collaboration via une plateforme légère et s'appuie sur Eclipse pour les mécanismes liés à la manipulation de modèles.

L'approche a été validée et illustrée à travers un cas d'étude portant sur la gestion du Service d'Urgence d'un hôpital (ce travail a été mené en collaboration avec le CHU de Montpellier). Il a également été mis en œuvre sur l'étude de cas d'un Système de Gestion de Conférences [26].

En proposant cette approche CAHM, nous avons répondu aux objectifs que nous nous étions fixés de :

fournir des processus collaboratifs permettant d'aligner les modèles partiels hétérogènes et de maintenir l'alignement en cas d'évolution.

En effet, le défi qui consistait à

mettre en œuvre des techniques collaboratives et à les appliquer à des processus de conception décentralisée

a été relevé par les mécanismes proposés.

CAHM et les mécanismes qu'elle intègre, fournissent ainsi des éléments de réponse aux questions :

Question 1 : Comment garantir la validité des modèles de conception dans leur ensemble ?

Question 2 : Comment garantir le cohérence des modèles de conception dans leur ensemble ?

Question 4 : Comment assurer la cohérence de l'ensemble en permettant à chacun de travailler avec ses propres outils ?

5.7 Discussion

Une évaluation de CAHM a été ménée dans le cadre d'une expérience de mise en œuvre sur le Système d'Urgence d'un hopital.

Deux groupes ont été constitués, dans deux slots temporels distincts. Le premier groupe a conçu les modèles de façon complètement distribuée et indépendante. Son travail s'est arrêté là. Un expert du domaine a ensuite travaillé en mode centralisé, chargé de connecter seul les modèles partiels [27]. Le second groupe d'utilisateurs était constitué d'étudiants en dernière année d'école d'ingénieur. Il faut noter, que lors de ces expériences, l'outil n'était pas opérationnel sur les aspects collaboratifs. La collaboration a pris forme à travers un fichier partagé, pour définir et évaluer les méta-correspondances.

Nous avons tout d'abord comparé les méta-correspondances et les modèles de correspondances obtenus. Nous avons ensuite exploité les résultats d'un sondage réalisé auprès des étudiants qui avaient joué le rôle de coordinateurs locaux.

Le gain de CAHM par rapport à l'approche centralisée AHM apparaît en termes de :

- *Efficacité* : Dans l'approche AHM, certaines HLC ne donnaient pas de LLC ; ce qui prouve que ces HLC étaient en fait inutiles. A contrario, toutes les HLC établies collaborativement et validées sont propagées dans le M1C.

  En outre, lorsque l'alignement est effectué par un seul acteur, moins de correspondances sont produites et l'alignement nécessite plusieurs itérations pour obtenir le même modèle de correspondance que celui obtenu par la collaboration de tous les coordinateurs locaux.

- *Flexibilité* : Dans CAHM, l'alignement des modèles est effectué collectivement et les actions des acteurs sont combinées, de sorte qu'une solide connaissance de tous les points de vue du système global (rôle de l'expert dans l'approche centralisée) n'est plus nécessaire ; Celles des acteurs se complètent. Chaque acteur peut ainsi concevoir plus librement sa solution en dehors de toute considération des autres points de vue.

- *Qualité de la décision* : Chaque acteur traite couramment son point de vue et la possibilité qui lui est offerte de discuter des connections avec les autres concepteurs lui permet de mieux appréhender non seulement le système dans sa globalité, mais aussi son propre point de vue sur celui-ci. Par ailleurs, la définition des correspondances entre méta-modèles en collaboration sur deux étapes (proposition, évaluation) répartit les responsabilités entre les *coordinateurs locaux* concernés et garantit que toutes les correspondances établies sont pertinentes pour le système étudié (puisqu'elles sont validées par différents acteurs).

- *Satisfaction* : les commentaires des acteurs soulignent que le processus CAHM leur facilite la tâche car ils n'ont pas eu à faire une analyse approfondie des autres méta-modèles.

Bien entendu, certains inconvénients sont inhérents à la collaboration dans ce contexte. Les acteurs ont par exemple relevé le problème des délais de communication. Néanmoins, une certaine durée peut être tolérée au vu des avantages d'une telle approche, tels que (i) la réduction de l'effort total, (ii) la moindre dépendance d'un expert, (iii) l'identification plus rapide des incohérences, etc.

A l'heure actuelle, nous n'avons pas tenu compte du temps nécessaire pour effectuer l'alignement, car nous nous sommes focalisés sur le gain, en termes de qualité de l'alignement notamment, d'une approche collaborative.

Bien entendu, l'étude des alignements proposés dans le domaine des ontologies et l'automatisation qu'elle pourrait permettre est à mettre en perspective avec une approche collaborative.

Mais nous pouvons considérer que la deuxième sera toujours nécessaire pour compléter la première. Nous proposons un sujet de thèse sur ce sujet. Ceci sera repris dans les perspectives globales de cette HDR.

Dans les perspectives à ce travail, il faut aussi envisager de ne plus considérer que des modèles de conception, mais aussi des modèles d'analyse, de tests, de réalisation, ... L'une des problématiques actuelles de l'ingénierie système reste le risque d'incohérence lié à l'hétérogénéité des artefacts générés et plus particulièrement à la difficulté d'établir des liens entre ces différents artefacts, produits à différentes phases du cycle de vie de développement du système. C'est ce sur quoi nous avons travaillé dans le cadre de la thèse de Florian Galinier qui est focalisée sur les modèles d'analyse à travers la modélisation et la formalisation des exigences (cf. chapitre 6 suivant).

Chapitre 6

Vers l'intégration de modèles d'exigences hétérogènes

Cette section reprend les éléments exposés dans l'article [90] "Seamless Integration of Multirequirements in Complex Systems", IEEE 25th International Requirements Engineering Conference Workshops.
Elle résume une partie des résultats obtenus dans le cadre de la thèse de Florian Galinier, "Seamless development of software-intensive systems : a multirequirements approach", qui sera soutenue début 2021.

Pour cette étude nous sommes partis du double constat que :

(i) Le Model-Based Systems Engineering [115, 151] encourage l'utilisation des modèles comme artefacts de base de l'ingénierie

(ii) La gestion des exigences est l'un des aspects fondamentaux de la mise en place de systèmes fiables.

Les exigences étant des éléments contractuels entre les ingénieurs et les autres parties prenantes (comme les clients), elles sont principalement exprimées en langage naturel et la plupart des méthodes et outils couramment utilisés traitent des exigences en langage naturel.

Mais seule la formalisation offre toutes les garanties de vérification, et de validation. Elle permet aux utilisateurs de prouver que le système répond à ses exigences et qu'il fera effectivement ce qu'il est censé faire. Néanmoins, si la formalisation des exigences est un bon moyen de s'assurer que le système est correct, elle est coûteuse, car elle effectuée par des spécialistes et la plupart du temps ne concerne que les parties critiques. En effet, l'expression des exigences dans le langage naturel, par opposition à l'expression formelle, soulève le débat de leur utilisation dans l'industrie.

L'usage de la formalisation est ralenti par la force de l'habitude. La formalisation des exigences s'avère souvent inutilisable et donc inutilisée par les acteurs qui participent à la mise en œuvre des systèmes. Dans la plupart des cas, les exigences sont tracées au cours du processus de développement et restent tout du long en langage naturel. Or, la traçabilité entre les exigences du système et le système lui-même est également fondamentale dans la mesure où elle permet d'identifier la couverture des exigences par le code, soit la partie des exigences qui n'est éventuellement pas satisfaite, ou encore ce qui sera affecté en cas de modification des exigences.

L'un des principaux objectifs de l'ingénierie des exigences est donc de généraliser l'utilisation de la formalisation des exigences. Mais l'introduction de nouvelles approches nécessite

de pouvoir composer avec les approches existantes. Pour que leur introduction soit utile, il faut aussi qu'elles correspondent à la réalité des usages, c'est-à-dire qu'elles restent proches du langage naturel.

C'est dans ce but, que nous avons proposé le langage RSML, (Requirement Specific Modelling Language), langage de modélisation spécifique à l'expression des exigences. Sa nouveauté réside dans l'utilisation d'un langage naturel contraint et la transformation automatique des exigences en éléments formels définis dans un langage de programmation. Ceci offre la possibilité de vérifier automatiquement les exigences écrites au départ en RSML, en s'appuyant sur l'environnement du langage cible choisi.

Le choix du langage s'est porté sur Eiffel [146] pour plusieurs raisons :

- Ce travail, a fait l'objet d'une thèse co-supervisée par Bertrand Meyer. Nous avons donc pu bénéficier du support de spécialistes de ce langage.
- Le langage Eiffel, par le mécanisme de programmation par contrat qu'il intègre, supporte la formalisation des exigences que nous réalisons par des assertions sémantiques.
- Eiffel s'accompagne d'un prouveur qui permet de faire de la vérification de couverture d'exigences ([148]).
- L'environnement de programmation d'Eiffel, Eiffel Studio, fournit des mécanismes de référencement (notes EIS), supports à la traçabilité, sur lesquel il est possible de s'appuyer pour réaliser un développement sans couture.

L'avantage du Langage Naturel Contraint est qu'il fournit d'avantage de formalisation que le langage naturel tout en étant plus accessible aux non-experts. (Comme nous voulons fournir une description formelle des systèmes, nous ne considérons pas le traitement du langage naturel. En effet, le *NLP* [1] [113] [8] [139] ne permet pas encore la description formelle d'une expression en langage naturel, comme le nécessitent les vérificateurs et les prouveurs).

Parallèlement, la modélisation peut être utilisée pour fournir un cadre formel lors de la composition avec des domaines spécifiques, notamment en fournissant des langages ou des transformations vers des langages formels existants. Nous avons proposé d'utiliser de telles techniques pour aborder la question de l'introduction de la formalisation des exigences tout en fournissant des outils abordables pour tous les types d'acteurs (qu'ils soient spécialistes de la sémantique ou pas). Une solution au problème de la prise en compte de plusieurs langages consiste à utiliser un paradigme unique - c'est le principe du modèle unique [167], [148]. L'approche que nous proposons est basée sur ce principe.

Nous avons utilisé l'exemple d'un drone de Parrot pour illustrer l'application de notre approche. La description des exigences du drone en Langage naturel est disponible en ligne [2].

Dans la première partie de ce chapitre section 6.1, je présenterai les concepts du langage et sa syntaxe. Dans la deuxième partie, section 6.1.3, je montrerai brièvement comment nous proposons d'utiliser la description sémantique générée en Eiffel et des relations sémantiques entre les exigences pour réaliser la vérification du code et la satisfaction des exigences. Je montrerai ensuite comment, en considérant RSML comme un langage pivot, nous offrons la possibilité de préserver les habitudes des parties prenantes (section 6.4). Nous proposons en effet d'utiliser les pratiques existantes de l'IDM (telles que les transformations [32]) comme un moyen de combler le fossé entre le langage naturel et la formalisation. Enfin, dans section 6.5, je donnerai une évaluation de l'approche, qui exploite les résultats d'une expérience d'utilisation de RSML sur un panel d'utilisateurs variés.

1. Natural Language Processing
2. https ://gitlab.com/fgalinier/rsml-examples/blob/master/AutomaticDeliveryDrone.md

6.1 RSML un langage dédié à l'expression des exigences

Les langages proches des non-experts et qui représentent un pont entre les deux mondes de la spécification des exigences informelles et formelles ont encore besoin d'améliorations, ce qui conduit à la nécessité d'un nouveau langage et d'une nouvelle méthode, proches des non-experts, permettant l'utilisation de méthodes de validation et de vérification.

Afin de faciliter la tâche des différents intervenants, nous avons proposé RSML, un langage d'expression d'exigences proche du langage naturel mais suffisamment formel pour pouvoir le traduire automatiquement en langage Eiffel et fournir ainsi des moyens de vérification de la satisfaction des exigences.

RSML est fondé sur un méta-modèle qui fera l'objet de la section 6.1.1. Il fait partie d'un projet dont l'objectif est de fournir un langage de modélisation (section 6.1.2), lisible et manipulable par les ingénieurs des exigences, permettant d'exprimer à la fois les exigences, les relations entre elles, et leurs relations avec d'autres artefacts. Ce langage est basé sur le langage Eiffel, plus formel, pour permettre l'analyse automatique des exigences à l'aide d'outils tels qu'Autoproof [187] (je reviendrai sur ce point dans la section 6.1.3). Il a été développé avec Xtext [3] dans Gemoc Studio [4] [61].

La Figure 6.1 ci-dessous décrit le processus suivi dans le cadre de l'expression d'exigences en RSML. La génération de la formalisation des exigences en Eiffel est automatique, à partir du moment où celles-ci ont été décrites en RSML dans un premier temps.

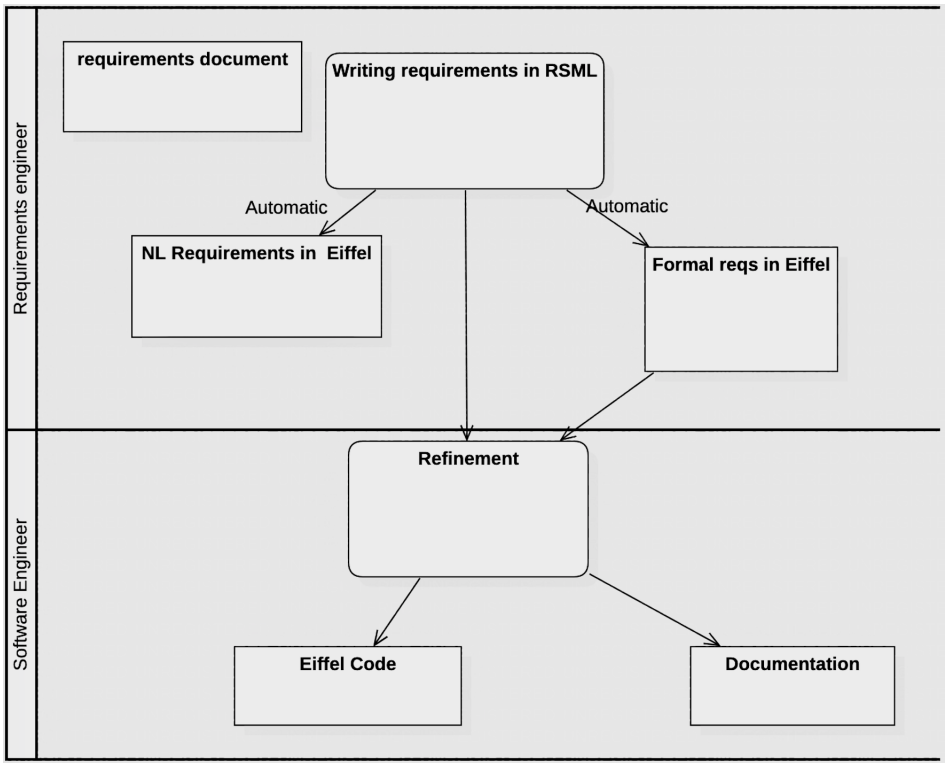


FIGURE 6.1 – Processus suivi dans le cadre de l'écriture d'exigences en RSML (Diag. d'activité UML)

RSML offre donc une interface au spécialiste des exigences qui lui permet de décrire des exigences, et d'obtenir leur description formelle, sans connaître de langage formel spécifique.

La version présentée dans ce chapitre est une version stabilisée de RSML, mais le langage évolue régulièrement pour gagner en expressivité.

3. https://www.eclipse.org/Xtext/documentation/index.html
4. http://gemoc.org/studio.html

6.1.1 Les concepts de RSML : Un méta-modèle dédié à l'expression d'exigences

Les différents acteurs impliqués dans un projet travaillent généralement avec des outils différents. En fonction de leurs antécédents, certains utilisateurs préféreront travailler avec des approches textuelles, tandis que d'autres sont habitués à mettre en œuvre des approches mathématiques.

Nous avons également constaté que :

- D'une part, la plupart des approches et des outils utilisés sont basés sur le texte. Ils ne requièrent pas de niveau d'expertise élevé puisqu'ils considèrent, la plupart du temps, les besoins dans leur ensemble (comme une unité textuelle). Ces approches et outils sont axés sur la traçabilité et les relations entre les exigences.

- D'autre part, les approches formelles se concentrent sur ce qui compose une exigence - quelles sont ses hypothèses et quelles sont les propriétés à vérifier. De plus, même s'il existe quelques tentatives d'exprimer les exigences dans un formalisme ad-hoc tel que ReqIF [75], il n'existe pas de norme sur la représentation des exigences et des éléments liés à ces exigences largement utilisée de nos jours, ni pour les langages naturels, ni pour les approches formelles.

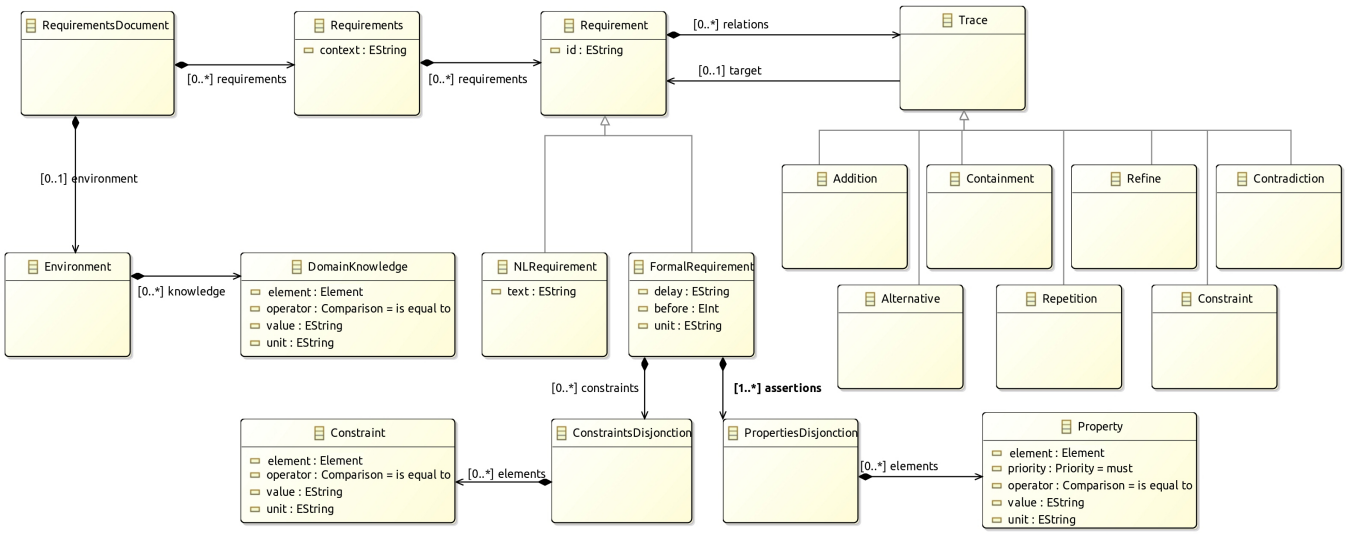


FIGURE 6.2 – Extrait du méta-modèle de RSML

Les concepts de RSML sont détaillés ci-dessous dans la description du méta-modèle décrit Figure 6.2.

Notre approche est basée sur la distinction de Jackson-Zave entre l'environnement et le système [118]. Ainsi, nous avons organisé le méta-modèle RSML en trois parties :

(i) Description du domaine (cf. Figure 6.3),

(ii) Description des exigences, qui distingue exigences en langue naturelle et exigences formelles (cf. Figure 6.4),

(iii) Description des relations entre exigences (cf. Figure 6.5)

La partie (i) du méta-modèle (Figure 6.3) permet aux utilisateurs de représenter les données du domaine, qui seront utilisées comme hypothèses d'environnement. Les hypothèses

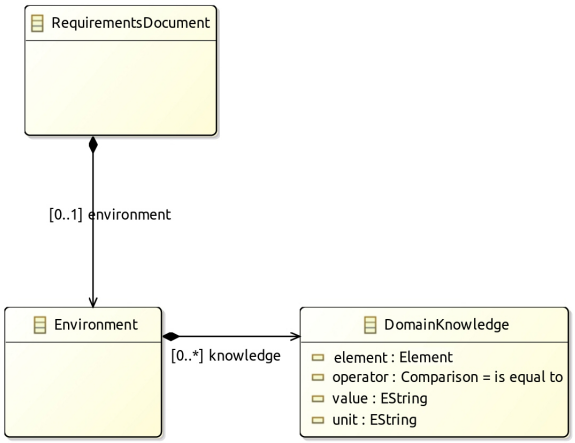


FIGURE 6.3 – Partie (i) du meta-modèle de RSML

concernent les élements (*element*), des artefacts identifiés (par une chaîne de caractères) qui peuvent ensuite être utilisés dans toutes les exigences, et sont formalisés sous forme d'invariants pour l'ensemble du système. Par exemple, une loi fixant que *"l'altitude de vol maximale autorisée pour un drône est de 150 mètres"* peut être exprimée en instanciant cette partie du méta-modèle.

Les hypothèses ainsi définies seront prises en compte pour toutes les exigences.

Les exigences proprement dites sont exprimées dans la partie (ii) du méta-modèle, décrit Figure 6.4.

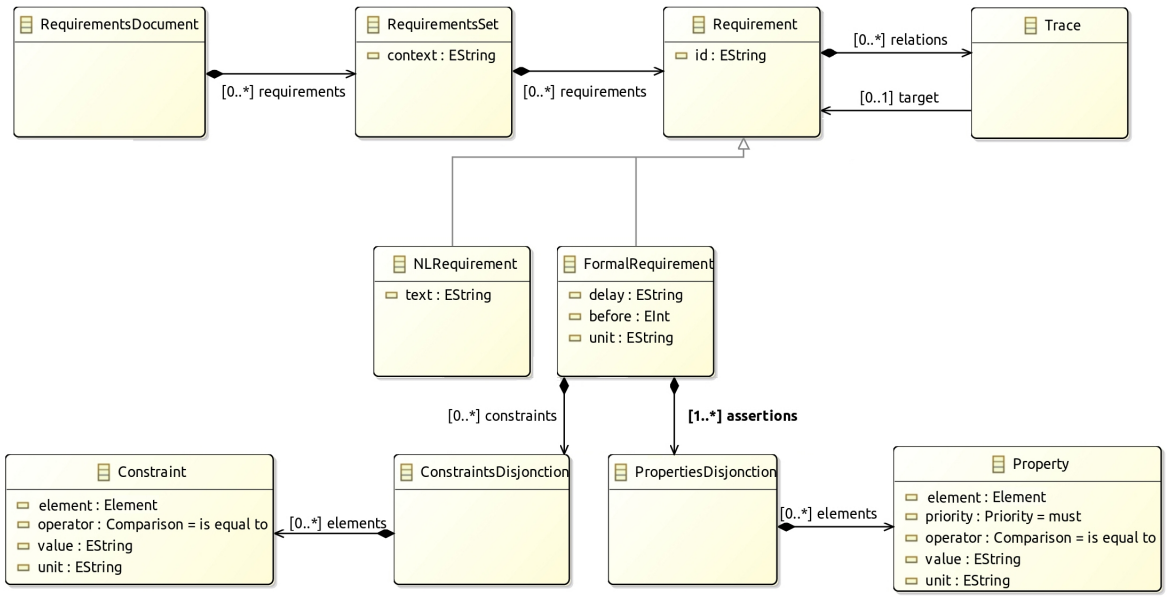


FIGURE 6.4 – Partie (ii) du meta-modèle de RSM

Elles doivent toutes être identifiées de manière unique pour faciliter la traçabilité. Nous avons choisi de laisser la possibilité d'exprimer les exigences en langage naturel ou dans une représentation formelle. Les exigences en langue naturelle auront alors un seul attribut de type texte (comme les exigences SysML ou KAOS).

Les exigences formelles sont exprimées sous la forme de propriétés (*Property*) que le système doit vérifier, dans un certain contexte. Le contexte est formalisé par un ensemble de contraintes (*Constraint*), et sera utilisé comme hypothèse pour les exigences.Les *propriétés* et les *contraintes* traitent toutes des *éléments*, et sont toutes identifiées par une chaîne de ca-

ractères. Ces éléments seront comparés à une valeur (*value*), qui peut être une valeur primitive (telle qu'un nombre ou une chaîne de caractères), un état ou une référence à un autre *élément* [5].

Les exigences exprimées en langage naturel n'ont pas de sémantique spécifique, mais les liens qu'elles peuvent avoir avec d'autres exigences ont une sémantique. Nous pouvons déduire de cette sémantique une traçabilité, support à des vérifications et à une certaine propagation de la satisfaction des exigences. Cette traçabilité que nous qualifions d'horizontale entre les exigences est exprimée par le concept de *Trace*.

Nous distinguons en effet deux formes de traçabilité : La traçabilité verticale (classique), équivalente à de la navigation de raffinage d'exigences, depuis leur expression en langage Naturel jusqu'à leur implantation dans le code, telle que définie dans [2] ou [97], et la traçabilité horizontale, exprimée ici par *Trace*, que nous définissons ainsi :

> **Traçabilité horizontale :** navigation fondée sur des relations sémantiques entre exigences

La section 6.3 présente brièvement comment la sémantique de la traçabilité horizontale est supportée par RSML et comment elle est mise en œuvre à des fins de validation et de vérification [88]. Je détaillerai la partie (iii) du méta-modèle de RSML (Figure 6.5) dans cette section 6.3

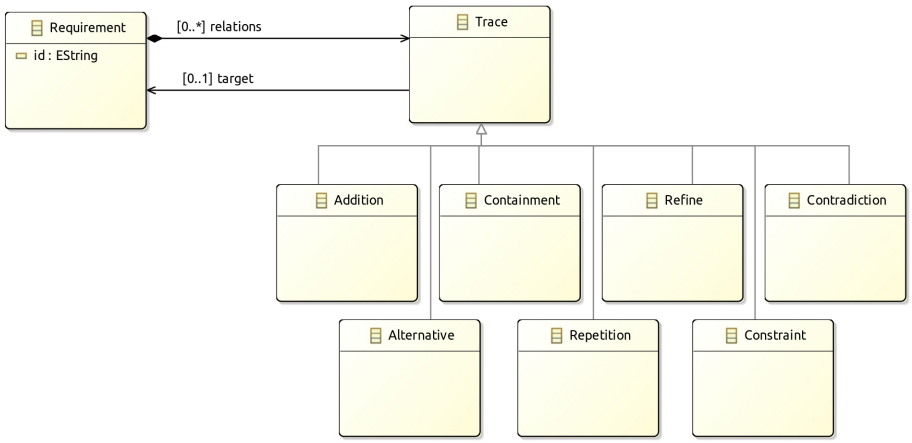


FIGURE 6.5 – Partie (iii) du meta-modèle de RSML

En résumé, le méta-modèle de RSML supporte la description du domaine (au moyen de données environnementales valuées), l'expression des exigences et un ensemble de relations inter-exigences.

6.1.2 Syntaxe de RSML, un langage de modélisation spécifique aux exigences

Les techniques et outils de l'IDM nous ont permis de définir, à partir de ce méta-modèle, le DSL RSML. Nous l'avons développé en utilisant l'environnement GEMOC [6]. RSML peut être comparé aux outils de développement dirigé par le comportement (BDD [7]) tels que Cucumber [194]. Toutefois, contrairement à ces outils, RSML est défini sémantiquement.

La syntaxe complète de RSML n'est pas décrite ici mais est accessible en ligne [8].

5. Voir `https://gitlab.com/fgalinier/RSML/blob/master/grammar/grammar.pdf` pour plus de détails sur la grammaire des exigences formelles
6. `http://gemoc.org/studio.html`
7. Behavior Driven Development
8. https ://gitlab.com/fgalinier/RSML/blob/master/grammar/grammar.pdf

L'exemple donné Figure 6.6 ci-dessous illustre la manière dont la description du domaine, les exigences en langage naturel et les exigences formelles, ainsi que leurs relations, peuvent être exprimées en RSML.

SysML.xmi | drone.rsml

```
Environment:
- Max authorized flight altitude is equal to 150 [m].

[1] "[
    The automatic delivery drone (later called `the drone') shall allows the company to quickly
deliver the ordered products
    to customer living in big cities where the company is based.
]"
[2] "[
    The drone shall be able to take in charge, transport and deliver a package carefully.
]"
[3] "[
    After a delivery, the drone shall come back to the warehouse.
]"

Drone:
[2.1] When the drone battery is less or equal to 10 [percent] then
    immediately mode must be equal to recovery.
[2.1.1] (refines [2.1]) When the drone battery is less or equal to 10 [percent] then
    eventually the drone altitude must be equal to 0 within 30 [seconds].
```

FIGURE 6.6 – Exemple de code RSML

RSML permet d'exprimer les exigences et les relations entre elles, en s'inspirant d'approches comme EARS [144]. Toutes sortes d'exigences peuvent être traitées avec RSML, qu'elles soient fonctionnelles, temporelles, de performance, ..., qu'elles soient exprimées en langage naturel ou formelles. Pour l'instant, les exigences qui peuvent être exprimées sont limitées à celles qui permettent de faire des comparaisons entre les propriétés. (Dans un but de généralisation, une des perspectives à ce travail est de mettre en place des quantificateurs dans RSML.)

La description du domaine est d'abord donnée dans la liste qui suit le mot-clé *Environnement*. L'exemple de section 6.1.1 est exprimé comme suit : "*L'altitude autorisée pour un vol est au maximum de 150 [m]*", mais peut également être exprimé par : "*L'altitude de vol maximale autorisée est égale à 150 [m]*".

La description du domaine suit les règles de grammaire suivantes :

⟨*environment*⟩ ::= (‘Env’ | ‘Environment’) ‘:’ ⟨*newline*⟩ ⟨*domain knowledge*⟩+

⟨*domain knowledge*⟩ ::= ‘–’ (⟨*simple definition*⟩ | ⟨*interval definition*⟩ | ⟨*range definition*⟩) ‘.’ ⟨*newline*⟩

⟨*simple definition*⟩ ::= ⟨*element*⟩ ⟨*comparison operator*⟩ ⟨*value*⟩ (‘[’ ⟨*unit*⟩ ‘]’)?

⟨*interval definition*⟩ ::= ⟨*element*⟩ (‘is in’ | ‘is out’) ⟨*value*⟩ ‘and’ ⟨*value*⟩

⟨*range definition*⟩ ::= ⟨*element*⟩ ‘can be’ ⟨*value*⟩ (‘,’ ⟨*value*⟩)*

⟨*element*⟩ ::= ⟨*word*⟩+

⟨*comparison operator*⟩ ::= 'is' ('equal to')?
| 'less than'
| 'less or equal to'
| 'greater than'
| 'greater or equal to'
| 'different to'

⟨*value*⟩ ::= ⟨*number*⟩ | ⟨*state*⟩ | ⟨*element*⟩

⟨*unit*⟩ ::= ⟨*word*⟩ ('^' ⟨*number*⟩)? ('/' ⟨*unit*⟩)?

Ce qui donne dans notre cas :

```
Environment:
- Max authorized flight altitude is equal to 150 [m].
```

Les exigences proprement dites sont ensuite exprimées. Elles sont étiquetées par leur identifiant unique (entre crochets).

Leur expression suit la règle de grammaire :

⟨*requirements set*⟩ ::= (⟨*word*⟩+ ':' ⟨*newline*⟩)? ⟨*requirement*⟩+

⟨*requirement*⟩ ::= '[' ⟨*id*⟩ ']' ('(' ⟨*relationships*⟩')')?
(⟨*natural language requirement*⟩ | ⟨*formal requirement*⟩) ⟨*newline*⟩

⟨*id*⟩ ::= ⟨*digit*⟩ ('.' ⟨*id*⟩)?

⟨*relationships*⟩ ::= ⟨*relationship kind*⟩ ⟨*id*⟩ ('and' ⟨*relationships*⟩)?

⟨*relationship kind*⟩ ::= 'refines'
| 'part of'
| 'is an addition of'
| 'is a constrained version of'
| 'contradicts'
| 'repeats'
| 'is an alternative of'
| 'trace'

Par exemple, dans l'exemple donné 6.6, les exigences 1, 2 et 3 sont celles du système, tandis que les exigences 2.1 et 2.1.1 sont exprimées dans le contexte du drone.

Contrairemennt aux exigences formelles, les exigences en langage naturel sont signalées par la présence de guillemets (les exigences en langage naturel sur plusieurs lignes sont comprises entre `"[` et `]"`, tandis que celles sur une ligne sont entourées de guillemets classiques), ainsi que le spécifie la règle de grammaire :

⟨*natural language requirement*⟩ ::= ⟨*single line text*⟩ | ⟨*multiline text*⟩

⟨*single line text*⟩ ::= '"' .* '"'

⟨*multiline text*⟩ ::= '"[' (. | ⟨*newline*⟩)* ']"'

Ainsi nous obtenons :

```
[1] "[
    The automatic delivery drone (later called `the drone') shall allows the company to quickly
deliver the ordered products
    to customer living in big cities where the company is based.
]"
[2] "[
    The drone shall be able to take in charge, transport and deliver a package carefully.
]"
[3] "[
    After a delivery, the drone shall come back to the warehouse.
]"
```

Les exigences formelles utilisent une syntaxe similaire à celle de Cucumber : (i) les contraintes sont introduites par when, (ii) alors que les propriétés sont introduites par then (non obligatoire s'il n'y a pas de contrainte).
Leur expression suit la grammaire suivante :

⟨*formal requirement*⟩ ::= (⟨*properties disjunction*⟩ ('and' ⟨*properties disjunction*⟩)?) '.'
| 'When' ⟨*constraints disjunction*⟩ ('and' ⟨*constraints disjunction*⟩)? 'then' 'immediately' ⟨*properties disjunction*⟩ ('and' ⟨*properties disjunction*⟩)? '.'
| 'When' ⟨*constraints disjunction*⟩ ('and' ⟨*constraints disjunction*⟩)? 'then' 'eventually' ⟨*properties disjunction*⟩ ('and' ⟨*properties disjunction*⟩)? ('within' ⟨*integer*⟩ '[' ⟨*Unit*⟩ ']')?) '.'

⟨*constraints disjunction*⟩ ::= ⟨*constraint*⟩ ('or' ⟨*constraint*⟩)?

⟨*constraint*⟩ ::= ⟨*element*⟩ ⟨*comparison operator*⟩ ⟨*value*⟩ ('[' ⟨*unit*⟩ ']')?

⟨*properties disjunction*⟩ ::= ⟨*property*⟩ ('or' ⟨*property*⟩)?

⟨*property*⟩ ::= ⟨*element*⟩ ⟨*priority*⟩ ⟨*modal comparison operator*⟩ ⟨*value*⟩ ('[' ⟨*unit*⟩ ']')?

⟨*modal comparison operator*⟩ ::= 'be equal to'
| 'be less than'
| 'be greater than'
| 'be less or equal to'
| 'be greater or equal to'
| 'be not equal to''

Ce qui donne ici :

```
Drone:
[2.1] When the drone battery is less or equal to 10 [percent] then
    immediately mode must be equal to recovery.
[2.1.1] (refines [2.1]) When the drone battery is less or equal to 10 [percent] then
    eventually the drone altitude must be equal to 0 within 30 [seconds].
```

D'avantage de détails sur la manière d'écrire les exigences avec RSML sont disponibles dans le fichier RSML/README [9]

9. https://gitlab.com/fgalinier/RSML/blob/master/README.md

6.1.3 Sémantique de RSML

Tous les éléments du méta-modèle de RSML ont une sémantique précise, basée sur Autoreq [158] (langage orienté objet reposant sur Eiffel [147] permettant d'exprimer et de vérifier des exigences), et à chacun d'entre eux est associé un modèle Autoreq permettant une transformation automatique de RSML en Eiffel. Cette sémantique est définie à l'aide du *Design by Contracts* [146], qui permet d'exprimer les exigences sous forme d'assertions sémantiques en utilisant directement la sémantique d'un langage de programmation (ici Autoreq).

Dans [158], une représentation des exigences à l'aide de fonctions Eiffel est proposée. Cela permet, d'une part, la promulgation d'exigences et, d'autre part, la preuve de la satisfaction des exigences par le système, grâce au prouveur Autoproof [187]. En effet, en fournissant une représentation des exigences en Eiffel, les utilisateurs peuvent spécifier (et même implémenter) le système de manière transparente [90] dans un langage unique et exécuter le prouveur pour vérifier que le système est conforme à ses exigences.

Les exigences exprimées en RSML sont automatiquement traduites en Eiffel, formalisées et raffinées. Nous avons choisi de continuer à utiliser le langage de programmation Eiffel parce qu'il intègre la programmation par contrat, qu'il est un langage à objets qui favorise le raffinement par spécialisation, et pour bénéficier de son outil de preuve Autoproof et de ses mécanismes de traçabilité.

Le mécanisme de notes [10] intégré à l'environnement de développment Eiffel Studio et en particulier EIS, permet de créer des liens entre différentes vues du code, entre le code et des documents, entre le code et les exigences correspondantes, entre différents artefacts de code. Ceci garantit une traçabilité entre les documents de spécification des exigences et l'implantation de celles-ci, notamment, que nous avons jugée fondamentale pour garantir le validité d'un système. C'est ce que j'évoquerai dans la section 6.3.

Notons que cette approche peut néanmoins être adaptée à d'autres langages qui fournissent une conception par contrat et un outil support, tel que JML [137] par exemple.

La représentation formelle de RSML est fondée sur une définition sémantique en Eiffel de chacun de ses éléments. Cette sémantique, détaillée dans [158], permet d'exprimer des exigences temporelles sous forme d'hypothèses (pré-conditions) et d'affirmations (post-conditions). Le méta-modèle RSML fournit une abstraction permettant d'exprimer ce type d'exigences, *formal requirements*. Par exemple, l'exigence 2.1 de la Figure 6.6 est ainsi formalisée en une fonction Eiffel (décrite dans le Listing 6.1).

```
when_drone_battery_is_less_or_equal_to_10_then_immediately_mode
_must_be_recovery
note
        EIS: "src=drone.rsml", "ref=[2.1]"
        src: [2.1] When the drone battery is less or equal to 10
           [percent] then immediately mode must be equal to recovery.
require
        when_drone_battery_is_less_or_equal_to_10_percent:
            drone_battery <= 10
deferred
ensure
        check_mode_must_be_equal_to_recovery: mode = recovery
end
```

Listing 6.1 – Sémantique de l'exigence 2.1 en Eiffel

10. https://www.eiffel.org/doc/eiffelstudio/Eiffel+Information+System

Ainsi, toute exigence est traduite en une fonction Eiffel. On voit ici que les exigences sont exprimées sous la forme de fonctions abstraites, limitées par des pré-conditions exprimant les contraintes contenues dans la clause when de RSML, soit l'hypothèse requise pour évaluer ces exigences (et exécuter les fonctions correspondantes). Les propriétés que les exigences doivent posséder (exprimées dans la clause then de RSML) sont décrites dans des post-condition (ce sont des affirmation que les fonctions doivent satisfaire lors de leur exécution). La note EIS qui apparait sur le listing 6.1 permet de faire le lien (cliquable) avec le document original drone.rsml et de référencer le bookmark qui correspond ici à la description de l'exigence concernée ([2.1]).

Le mécanisme de notes intégré à l'environnement EiffelStudio consiste à rajouter des méta-données dans le code. La note EIS, par exemple, est définie par EiffelStudio comme un lien vers un fichier externe. Elle permet d'associer à l'artefact en cours de définition, un bookmark dans un document.

La note *src* spécifie un lien entre deux artefacts de code. Nous utilisons *src* pour spécifier qu'une classe ou une fonction d'exigence est une formalisation d'une exigence en langage naturel. Ainsi le lien entre l'exigence en langage naturel [2.1] et sa formalisation en Eiffel dans la fonction d'exigence *when_drone_battery_is_less_or_equal_to_10_then_immediately_mode_must_be_recovery* est fait par la note *src* (comme sur le listing 6.1 ci-dessus).

6.2 De RSML à Eiffel

Dans l'approche que nous proposons, l'utilisateur peut exprimer les exigences en langage naturel, puis affiner progressivement chacune d'entre elles en des exigences plus formelles, et enfin lier toutes les exigences dans un même formalisme.

Les exigences sont regroupées par contexte, dans un projet Eiffel, et vont donner lieu à des fonctions d'exigences regroupées dans des classes, correspondant à ces contextes. Chaque contexte accède aux définitions de l'environnement.

Toutes les données environnementales sont contenues dans une classe dédiée, `DOMAIN_KNOWLEDGE` (cf. listing 6.2).

Deux graphes de classes sont générés, liés par des relations de type *src*. D'une part celui des classes d'exigences en langage naturel et d'autre part l'ensemble des classes d'exigences en langage formel qui réutilisent les définitions de l'environnement. Les classes déduites de la spécification RSML du Drone [11], sont résumées sur la Figure 6.7.

11. `https://gitlab.com/fgalinier/RSML/-/blob/master/use-case/drone.rsml`

```
note
        EIS: "src=use-case_drone.rsml", "ref=Environment"
        description: "[This class contains the domain knowledge
        that will be used by requirements.]"

deferred class DOMAIN_KNOWLEDGE
feature
        -- States range
     ...
        return : DOUBLE = 3.0
        activated : DOUBLE = 4.0
        control_mode : DOUBLE = 5.0
        normal_mode : DOUBLE = 6.0
        delivering : DOUBLE = 7.0
        destination_for_parcel_from_webservice : DOUBLE = 8.0
        confirmed : DOUBLE = 9.0
        recovery_mode : DOUBLE = 10.0
        release : DOUBLE = 11.0
        ...
        -- States
        ...
        duration: DOUBLE
        max_weight_of_parcel : DOUBLE
        max_size_of_parcel : DOUBLE
        low_battery_status : DOUBLE
        minimum_safe_height : DOUBLE
        drone_location : DOUBLE
        parcel_position : DOUBLE
        drone_status : DOUBLE
        assigned_parcel : DOUBLE
        attached_parcel : DOUBLE
        drone_mode : DOUBLE
        drone_altitude : DOUBLE
        destination : DOUBLE
        drone_position : DOUBLE
        drone_battery_status : DOUBLE
        application_status : DOUBLE
        malfunction_status : DOUBLE
        battery_status_in_application : DOUBLE
    ....

invariant
    ...
        max_weight_of_parcel_is_10: max_weight_of_parcel = 10 -- kg
        max_size_of_parcel_is_1: max_size_of_parcel = 1 -- m
        low_battery_status_is_10: low_battery_status = 10 -- percent
        minimum_safe_height_is_50: minimum_safe_height = 50 -- m
end
```

Listing 6.2 – Extrait du code Eiffel de la classe de description de l'environnement du Drone

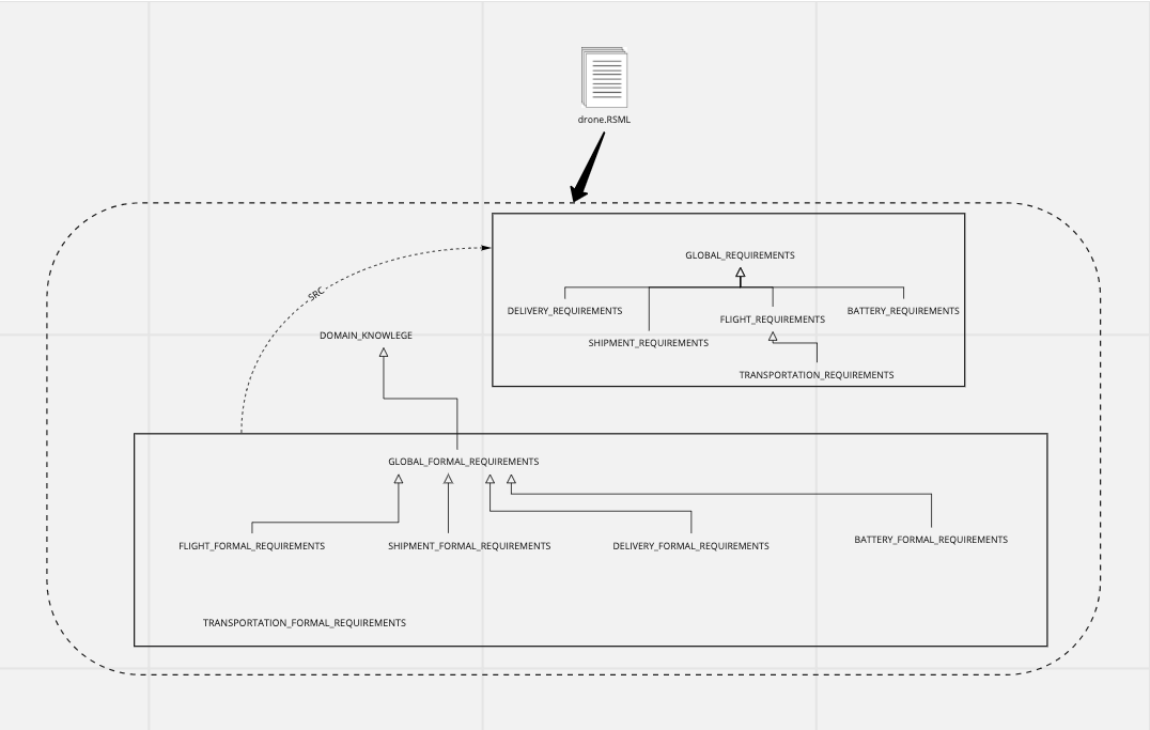


FIGURE 6.7 – Description de la hiérarchie de classes d'exigences du système Drone avant raffinement

Dans le listing 6.3, la classe `GLOBAL_REQUIREMENTS` contient toutes les exigences en langage naturel liées au contexte *Drone* décrit dans la spécification RSML du Drone et dont la Figure 6.8 présente un extrait.

[0.1] "[The aim of the automatic delivery drone (later called "the drone") is to transport parcels from the commercial enterprise warehouse to its customers. The drone is a quadricopter equipped with a clamp.]"

[0.2] "[When activated, the drone shall take over a parcel (pick it in the warehouse and deliver it).]"

[0.3] "[The drone shall be controlled by a web application, and being accessible all the time by this application.]"

[0.4] "[In normal mode, the drone shall be fully automated, but an operator must be able to switch to control mode to take control of the drone with the application.]"

[0.4.1] (refines [0.4]) Drone mode could be equal to normal mode or drone mode could be equal to control mode.

[0.5] "[The drone is propelled by four rotors, each driven by a brushless motor. The power shall be enough to transport parcels weighing up to 10 kg and with a size up to 1x1 m.]"

Behavior:

[1.1] "[The drone shall pick up a parcel, go to the destination and drop it off when activated on the web application.]"

[1.1.1] (part of [1.1]) When the drone status is equal to activated then eventually attached parcel shall be not equal to null within 120 [s].

[1.1.2] (part of [1.1]) When attached parcel is not equal to null then eventually drone location shall be equal to destination within 1500 [s].

[1.1.3] (part of [1.1] and part of [1.4]) When drone location is equal to destination then eventually attached parcel shall be equal to null within 120 [s].

FIGURE 6.8 – Extrait de la spécification RSML du système Drone

La classe *GLOBAL_REQUIREMENTS* donne une description textuelle des exigences et fait le lien, via des notes EIS, avec le code RSML correspondant. Elle est abstraite (*deferred*) car ces exigences doivent être formalisées puis raffinées pour être finalement implantées. On peut

constater que l'exigence [0.4.1] est exprimée formellement en RSML. Elle donne donc lieu à une fonction d'exigence dont l'identifiant correspond à la description en RSML.

```
note
        EIS: "src=use-case_drone.rsml"
        description: "[Global requirements of the system]"
deferred class GLOBAL_REQUIREMENTS
feature
-- Requirements
    ...
        requirement_0_2_doc: STRING
        note
            EIS:"src=use-case_drone.rsml", "ref=[0.2]"
        do
            Result := "[
                When activated, the drone shall take over a
                parcel (pick it in the warehouse and deliver
                it).
                        ]"
        end

    requirement_0_4_doc: STRING
        note
            EIS:"src=use-case_drone.rsml", "ref=[0.4]"
        do
            Result := "[
                    In normal mode, the drone shall be fully
                    automated, but an operator must be able to
                    switch to control mode to take control of
                    the drone with the application.
                        ]"
        end

    drone_mode_could_be_normal_mode_or_drone_mode_could_be_
    control_mode_doc: STRING
        note
            EIS: "src=use-case_drone.rsml", "ref=[0.4.1]"
            refines: "{GLOBAL_REQUIREMENTS}.requirement_0_4_doc"
        do
            Result := "[
                    [0.4.1] (refines [0.4]) Drone mode could be
                    equal to normal mode or drone mode could be
                    equal to control mode.
                        ]"
        end
    ...

end
```

Listing 6.3 – Extrait de la Description des exigences du Drone en Eiffel

La classe *GLOBAL_REQUIREMENTS* fournit une vue documentation des exigences et s'inscrit dans l'approche sans couture, car si les classes d'exigences formelles sont automatiquement obtenues depuis la description RSML, elles peuvent l'être aussi manuellement depuis ces classes.

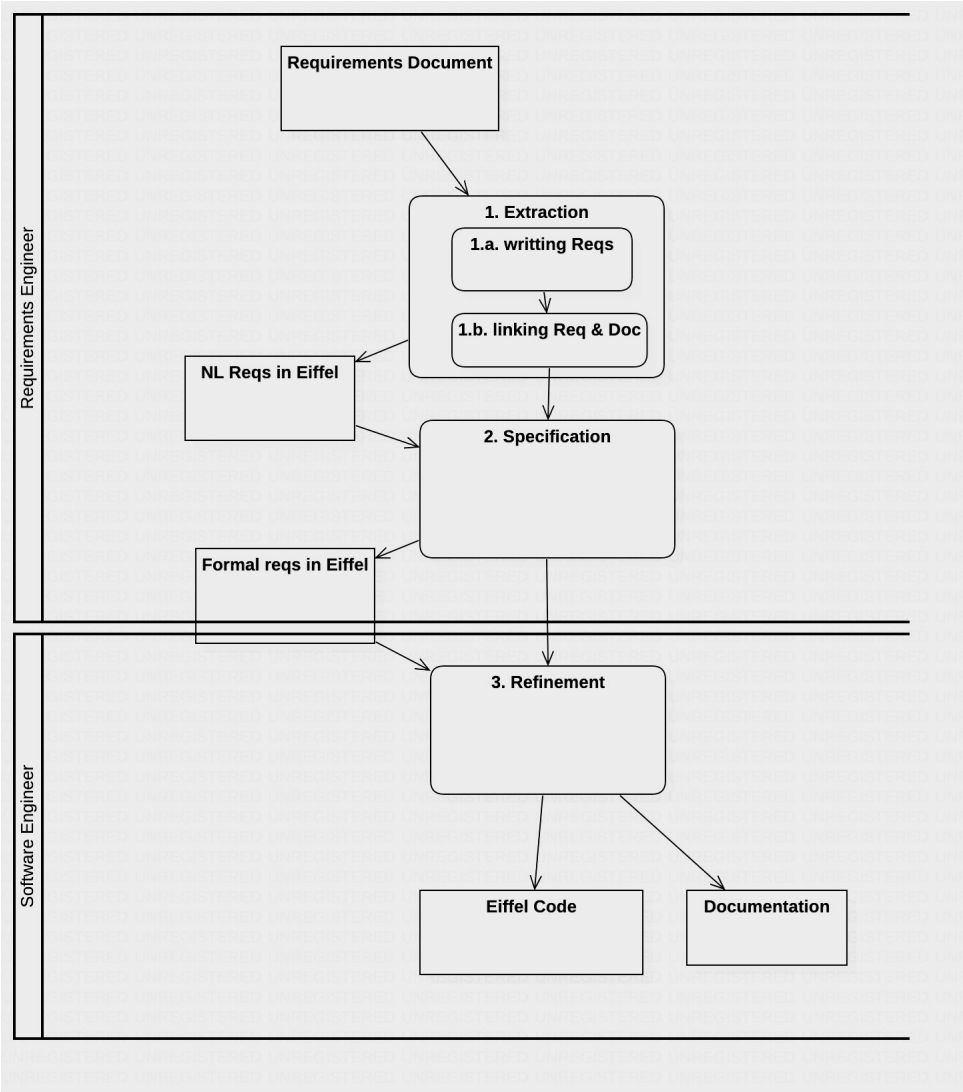


FIGURE 6.9 – Processus de l'approche d'intégration sans couture des exigences dans le code (SIRCOD) - Diagramme d'activité UML

Dans l'approche SIRCOD, décrite dans la thèse de Florian Galinier [89], tout le processus d'obtention du code depuis les exigences contenues dans un document est réalisé en Eiffel, par un raffinement sans couture. Les classes Eiffel décrivant en langage naturel les exigences sont d'abord réalisées, puis raffinées en classes formelles, manuellement, selon le processus décrit Figure 6.9. Je ne détaillerai pas plus cette approche que l'on peut trouver dans le chapitre qui la décrit de [89].

Les classes DELIVERY_REQUIREMENTS, SHIPMENT_REQUIREMENTS, etc. fournissent une vue documentation des contextes DELIVERY, SHIPMENT, etc. de la specification RSML dont un extrait est donné dans la Figure 6.8 (le code Eiffel du projet drone est accessible en ligne [12]).

Chaque classe donne lieu à la création d'une classe équivalente, formelle (_FORMAL_), qui précise pour chaque fonction d'exigence un lien *src* avec sa description informelle. Ainsi, les fonctions d'exigences de la classe GLOBAL_FORMAL_REQUIREMENTS contiennent une note *src* vers les fonctions de la classe GLOBAL_REQUIREMENTS (voir listing 6.4).

12. https ://gitlab.com/fgalinier/seamless_refinement//tree/master/drone_project

```
note
        EIS: "src=use-case_drone.rsml", "ref=globals requirements"
        description: "[This class contains globals requirements]"
deferred class GLOBAL_FORMAL_REQUIREMENTS
inherit
        DOMAIN_KNOWLEDGE
feature -- Requirements
        ...
        requirement_0_2
        note
                src: "{GLOBAL_REQUIREMENTS}.requirement_0_2_doc"
        deferred
        end

    ...

        requirement_0_4
        note
                src: "{GLOBAL_REQUIREMENTS}.requirement_0_4_doc"
        deferred
        end

    drone_mode_could_be_normal_mode_or_drone_mode_could_be_
    control_mode
        note
            src: "{GLOBAL_REQUIREMENTS}.drone_mode_could_be_
            _normal_mode_or_drone_mode_could_be_control_mode_doc"
            refines: "{GLOBAL_REQUIREMENTS}.requirement_0_4"
        local
            old_duration: like duration
        do
        from
                old_duration := duration
                -- Add call to specification
        until
                ((drone_mode = normal_mode or drone_mode =
                control_mode))
                or, (duration - old_duration) > 31536000
                -- seconds (equivalent to 1 year)
        loop
                -- Add call to specification
        end

        check assert: (drone_mode = normal_mode or
        drone_mode = control_mode) end
        check assert: (duration - old_duration) <= 31536000 end
        -- seconds (equivalent to 1 year)
        end

         ...
end
```

Listing 6.4 – Extrait de l'expression des exigences du drone en Eiffel

Ce code peut être utilisé de plusieurs façons. D'abord en tant que cadre de développement, car les fonctions d'exigences peuvent être utilisées pendant le développement pour vérifier que toutes les exigences sont mises en œuvre, et qu'elles le sont de la bonne manière. Ensuite pour la validation, en analysant si la formalisation des exigences et leurs relations sont correctes (afin d'éventuellement détecter certaines incohérences).

6.3 Raffiner les exigences et définir des liens entre elles

RSML permet aux utilisateurs non spécialistes d'exprimer les exigences, sans aucune considération de la manière dont elles sont réellement formalisées. Cette formalisation peut être utilisée pour la validation ou comme premier canevas pour un développement *correct par construction*. En effet, puisque ces fonctions sont abstraites, chaque classe héritant de la classe GLOBAL_FORMAL_REQUIREMENTS par exemple, doit prendre en compte ces fonctions différées : soit en les implémentant directement, soit en les implémentant dans ses classes descendantes. Cela se fait dans la troisième activité du processus présenté sur la Figure 6.1, appelée *Raffinement*, qui est itérative.

6.3.1 Raffinement des exigences

L'idée poursuivie est de partir d'exigences assez simples et abstraites, et de les enrichir de manière itérative, en les rendant de plus en plus concrètes. Ce processus s'inscrit dans l'approche du raffinement tel que définit en ingénierie des exigences par [135] et [199]. Le paradigme orienté objet et la conception par contrat sont tout à fait adaptés à cette approche. Selon le principe de substitution de Liskov, les pré-conditions et les post-conditions ont les mêmes propriétés que dans le raffinement défini par Abrial dans [6].

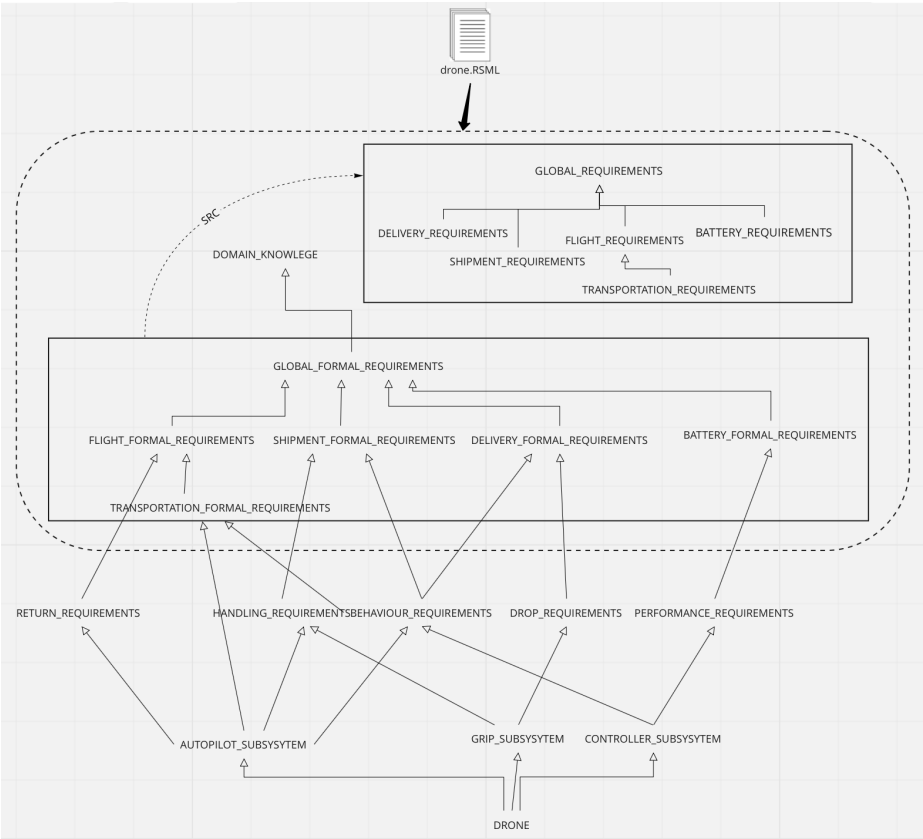


FIGURE 6.10 – Description de la hiérarchie de classes d'exigences du système Drone

Notons aussi que le langage Eiffel nous permet de bénéficier de l'héritage multiple. Ainsi, une classe d'exigences peut spécialiser par héritage plusieurs classes formelles. Ceci nous permet d'obtenir l'ensemble des exigences du drone en fusionnant l'ensemble des exigences de ses sous-sytèmes. Ainsi, dans le cas du drone, le graphe des classes obtenues sera celui de la Figure 6.3.1.

Les fonctions d'exigences en langage naturel doivent être formalisées. La fonction d'exigences *when_drone_status_is_activated_then_eventually_attached_parcel_should_not_be_null* déclarée dans le Listing 6.5, sera implantée par la fonction de même nom du Listing 6.6. En effet, la classe BEHAVIOR_REQUIREMENTS hérite de DELIVERY_FORMAL_REQUIREMENTS et a pour charge d'implanter les méthodes, restées abstraites, qui concernent le comportement du drone.

Cette méthodologie garantit que toutes les exigences ont non seulement été prises en compte par l'ingénieur, mais aussi qu'elles ont été satisfaites. En effet, une spécification incorrecte, entraînera une erreur lors de l'analyse statique ou des tests. Par ailleurs, les pré et post-conditions doivent être vérifiées selon les adaptations liées à l'héritage.

```
note
    EIS: "src=use-case_drone.rsml", "type=trace"
    description: "[ Delivery requirements of the system ]"

deferred class DELIVERY_FORMAL_REQUIREMENTS
inherit
    GLOBAL_FORMAL_REQUIREMENTS
feature -- Requirements
...
    when_drone_location_is_destination_then_eventually_attached_
    parcel_should_be_null
    note
        src: "{DELIVERY_REQUIREMENTS}.when_drone_location_is_destination_
        then_eventually_attached_parcel_should_be_null_doc
        parts: "{DELIVERY_FORMAL_REQUIREMENTS}.requirement_1_1"
        parts: "{DELIVERY_FORMAL_REQUIREMENTS}.requirement_1_4"
    deferred
    end
...
    when_drone_status_is_activated_then_immediately_assigned_
    parcel_should_not_be_null_and_parcel_position_should_not_be_null
    note
        src: "{DELIVERY_REQUIREMENTS}.when_drone_status_is_activated_
        then_immediately_assigned_parcel_should_not_be_null_and_
        parcel_position_should_not_be_null_doc"
    deferred
    end
...
end
```

Listing 6.5 – Classe d'exigences formelles et lien *src* avec sa classe d'exigences en LN

```
note
    EIS: "src=use-case_drone.rsml", "ref=Behavior"
    description: "[This class contains requirements in the context of:
    Delivery, in the context of:Behavior. ]"
deferred class BEHAVIOR_REQUIREMENTS
inherit DELIVERY_FORMAL_REQUIREMENTS
    redefines
    when_drone_status_is_activated_then_eventually_attached_parcel_
    should_not_be_null
    when_drone_status_is_activated_then_immediately_assigned_parcel_
    should_not_be_null_and_parcel_position_should_not_be_null
feature -- Requirements
    when_drone_status_is_activated_then_eventually_attached_parcel_
        should_not_be_null
    note
       parts: "{BEHAVIOR_REQUIREMENTS}.requirement_1_1"
                  -- inherited from DELIVERY_FORMAL_REQUIREMENTS
    local
       old_duration: like duration
    do
       from
            check assume: (drone_status = activated) end
            old_duration := duration
            -- Add call to specification
       until
            ((attached_parcel /= null)) or
            (duration - old_duration) > 120
       loop
            check assume: (drone_status = activated) end
            -- Add call to specification
       end
       check assert: (attached_parcel /= null) end
       check assert: (duration - old_duration) <= 120 end
    end
...
    when_drone_status_is_activated_then_immediately_assigned_parcel_
    should_not_be_null_and_parcel_position_should_not_be_null
    note
       parts: "{BEHAVIOR_REQUIREMENTS}.requirement_1_2"
    require
       when_drone_status_is_equal_to_activated:
            (drone_status = activated)
    deferred
    ensure
       check_assigned_parcel_shall_be_not_equal_to_null:
            (assigned_parcel /= null)
       check_parcel_position_shall_be_not_equal_to_null:
            (parcel_position /= null)
    end
end
```

Listing 6.6 – Spécialisation et Implantation d'exigences de DELIVERY_REQUIREMENTS dans BEHAVIOR_REQUIREMENTS

6.3.2 Liens entre exigences et traçabilité

En mettant en œuvre l'héritage et en notant des liens src, nous conservons un lien entre les différents artefacts liés aux exigences, et nous assurons une traçabilité verticale, depuis la toute première représentation en langage naturel des exigences jusqu'à leur implantation finale.

La traçabilité horizontale est quant à elle obtenue en incluant les relations qui peuvent exister entre les exigences. C'est le mécanisme des notes de Eiffel, qui rend possible de refléter ces relations entre exigences.

Nous l'avons en effet utilisé pour spécifier les liens entre les exigences et les artefacts associés, comme, dans le Listing 6.7, où la fonction est enrichie d'une note d'addition (*adds*), qui donne l'information que *when_drone_location_is_destination_then_eventually_attached_parcel_should_be_null_and_drop_location_status_should_be_confirmed* de la classe DROP_REQUIREMENTS est une extension de *when_drone_location_is_destination_then_eventually_attached_parcel_should_be_null* de la classe BEHAVIOR_REQUIREMENTS.

```
note
    EIS: "src=use-case_drone.rsml", "ref=Drop"
    description: "[This class contains requirements in the context
        of: Delivery, in the context of: Drop.]"
deferred class DROP_REQUIREMENTS
inherit DELIVERY_FORMAL_REQUIREMENTS
    redefine
      when_drone_location_is_destination_then_eventually_attached_par
      cel_should_be_null_and_drop_location_status_should_be_confirmed
      ....
feature -- Requirements
    when_drone_location_is_destination_then_eventually_attached_par
    cel_should_be_null_and_drop_location_status_should_be_confirmed
    note
       adds: "{BEHAVIOR_REQUIREMENTS}.when_drone_location_is_destina
       tion_then_eventually_attached_parcel_should_be_null"
    local
       old_duration: like duration
    do
       from
            check assume: (drone_location = destination) end
            old_duration := duration
                -- Add call to specification
       until
            ((attached_parcel = null) and
            (drop_location_status = confirmed)) or
            (duration - old_duration) > 31536000 -- seconds
       loop
            check assume: (drone_location = destination) end
                -- Add call to specification
       end
       check assert: (attached_parcel = null) end
       check assert: (drop_location_status = confirmed) end
       check assert: (duration - old_duration) <= 31536000 end
       end
        ...
end
```

Listing 6.7 – Exemple de relations entre exigences formelles

La sémantique des relations entre les artefacts influence la couverture de la satisfaction des exigences, et sa vérification. Dans l'exemple de ces deux fonctions, (Listing 6.7), lorsque *when_drone_location_is_destination_then_eventually_attached_parcel_should_be_null_and_-drop_location_status_should_be_confirmed* est satisfait, *when_drone_location_is_destination_-then_eventually_attached_parcel_should_be_null* l'est aussi.

Relations dans le méta-modèle de RSML

Nous avons défini ainsi sept relations dans le méta-modèle, spécialisation du concept de *Trace* :

- addition : l'exigence source ajoute une propriété à l'exigence cible
- containment : l'exigence source contient l'exigence cible
- refines : l'exigence source étend l'exigence cible
- contradiction : l'exigence source et l'exigence cible ne peuvent pas être satisfaites en même temps
- alternative : l'exigence source est une alternative à l'exigence cible
- repetition : l'exigence source est une répétition de l'exigence cible
- constraint : l'exigence source contraint le contexte de l'exigence cible

Chacune de ces relations a une sémantique définie et spécifiée par des contraintes OCL. Je donnerai ici la règle OCL de l'addition (adds) uniquement, mais la spécification des relations est consultable en ligne [13]‘

- Addition implique que les deux *Formal Requirements* partagent le même contexte et que les propriétés de l'exigence cible sont un sous-ensemble des propriétés de l'exigence source.

  Ceci peut être vérifié par :

```
context Addition inv:
  (self.source.oclIsTypeOf(FormalRequirement) and
 self.target.oclIsTypeOf(FormalRequirement)) implies
 (self.target.constraints = self.source.constraints and
 self.source.properties->includesAll(self.target.properties)
  )
```

- Refine implique que les propriétés de l'exigence cible sont un sous-ensemble de l'exigence source et que les contraintes de l'exigence étendue, la cible, sont plus restrictives que les contraintes de l'exigence source.
- Containment est une "addition" particulière car l'union des propriétés des sous-exigences contenues doit être égale à l'ensemble des propriétés du conteneur.
- Constraint implique que les deux *Formal Requirements* partagent les mêmes propriétés et que la contrainte d'exigence cible est un sous-ensemble de la contrainte d'exigence source.

13. https://gitlab.com/fgalinier/RSML/-/blob/master/relationships/Demonstration_of_relationships_semantics.pdf

- Alternative implique que les propriétés de l'exigence source sont un sous-ensemble des propriétés de l'exigence cible et même chose pour les contraintes.
- Repetition implique que les propriétés et les contraintes des deux exigences sont identiques.
- Contradiction signifie que e les propriétés et les contraintes des deux exigences sont contradictoire. Dans ce cas une règle OCL ne peut que vérifier que l'une des deux contraintes d'exigences définies est un sous-ensemble de l'autre (c'est-à-dire que les deux exigences sont considérées dans le même contexte), puisque nous n'analysons pas la sémantique des propriétés elles-mêmes dans le méta-modèle.

Expression des relations en RSML

Ces relations s'expriment dans la syntaxe de RSML par :

⟨*requirement*⟩ ::= '[' ⟨*id*⟩ ']' ('(' ⟨*relationships*⟩ ')')?
(⟨*natural language requirement*⟩ | ⟨*formal requirement*⟩) ⟨*newline*⟩

⟨*relationships*⟩ ::= ⟨*relationship kind*⟩ ⟨*id*⟩ ('and' ⟨*relationships*⟩)?

⟨*relationship kind*⟩ ::= 'refines'
| 'part of'
| 'is an addition of'
| 'is a constrained version of'
| 'contradicts'
| 'repeats'
| 'is an alternative of'
| 'trace'

Expression des relations en Eiffel

Chacune de ces relations est également formalisée en Eiffel. Nous avons proposé d'étendre l'approche suivie jusqu'ici en faisant correspondre une exigence à une fonction d'exigence Eiffel, en faisant correspondre les relations définies ci-dessus aux concepts des langages de programmation orientés objet en général, et d'Eiffel en particulier. Grâce à la représentation formelle des exigences et des relations présentée ci-dessus, nous avons montré comment nous pouvons propager la satisfaction entre les exigences à travers leurs relations []. Nous ne prétendons pas être exhaustifs avec cet ensemble de relations, notre objectif étant de fournir un moyen simple de les exprimer afin de déduire des propriétés de propagation de satisfaction et ainsi d'optimiser le contrôle de la cohérence de l'ensemble des exigences produites.

Nous avons considéré lors de l'implantation des exigences en Eiffel, qu'une exigence R est une fonction d'une classe, qu'une contrainte $\mathcal{C}_R$ de R est une *pré-condition* de cette fonction et une propriété $\mathcal{P}_R$ une *post-condition*. Nous faisons correspondre la représentation formelle des exigences aux concepts Eiffel, en nous fondant sur cette représentation. J'illustrerai ici l'addition sur l'exemple du drone, mais le détail de l'ensemble des relations est disponible en ligne [14].

14. https://gitlab.com/fgalinier/RSML/-/blob/master/relationships/Demonstration_of_relationships_semantics.pdf

Addition

Pour gérer l'*addition* nous pouvons ajouter de nouvelles propriétés en ajoutant de nouvelles post-conditions. Ainsi, pour garder la trace qu'une exigence R_6 est un ajout à une exigence R_5, on utilise le mécanisme d'appel. R_6 possède le même corps que R_5 avec de nouvelles propriétés à satisfaire. C'est le cas dans l'exemple (Listing 6.8) ci-dessous, `Req_6` est un appel à `Req_5` avec de nouvelles post-conditions.

```
Req_5
note
    src: "[When a parcel is handled by the drone, the drone
    shall transport it to its destination]"
require
    parcel_handled: parcel /= Void
do

ensure
    transport_to_destination: location = parcel.destination and
    parcel = Void
end

Req_6
note
    src: "[ The drone shall retrieve destination of handled parcel
    from an existing webservice and transport it  to this destination ]"
do
    Req_5
ensure
    destination_from_webservice:
          location=webservice.destination(parcel)
end
```

Listing 6.8 – Representation d'une addition en Eiffel

La propagation de la satisfaction de l'addition est préservée puisque `Req_6` ne peut être prouvé que si `Req_5` l'est aussi.

Ces relations peuvent ensuite être utilisées pour retracer les exigences et vérifier que toutes les exigences sont correctement satisfaites.

La traçabilité horizontale qu'elles induisent, offre une analyse facilité de la couverture des exigences en permettant notamment de réduire l'ensemble des exigences à vérifier.

La traçabilité verticale est garantie d'une part par l'héritage entre les classes des différents raffinements d'une exigence, et d'autre part par le lien induit dans la note *src*.

6.4 Un pont entre différents espaces technologiques

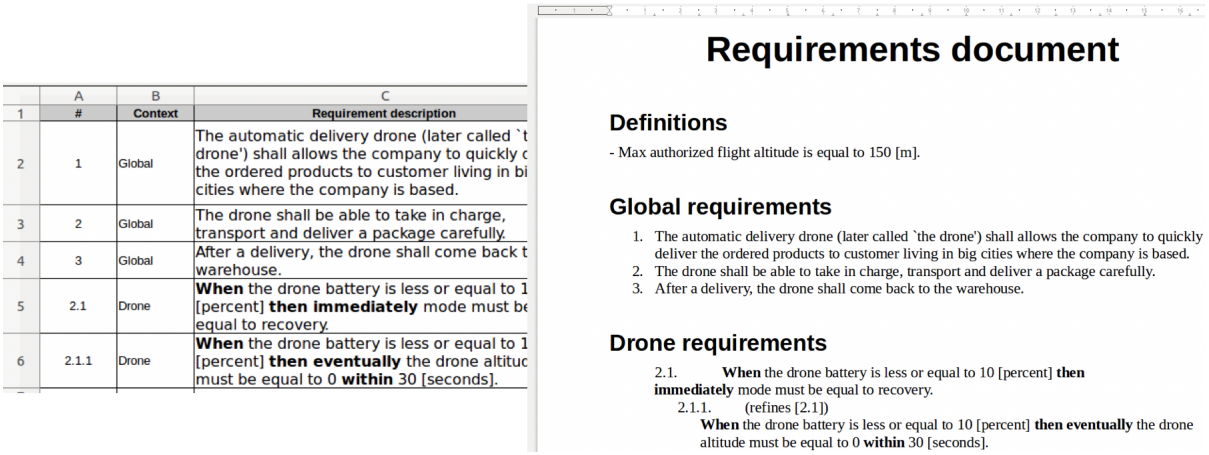
Nous proposons d'utiliser des techniques de l'IDM pour aborder la question de l'introduction d'éléments formels dans les exigences tout en fournissant des outils abordables pour tous les types d'acteurs (qu'ils soient ou non spécialistes de la sémantique). Nous nous adressons en particulier aux acteurs qui ont l'habitude d'utiliser des langages de modélisation ou le langage naturel. Nous leur suggérons d'utiliser RSML pour profiter de la puissance des annotations légères qu'il offre.

Néanmoins, nous sommes pleinement conscients que les utilisateurs finaux peuvent préférer continuer à mettre en œuvre leurs pratiques. Pour cette raison, nous avons prévu de construire des ponts entre RSML et d'autres approches d'ingénierie des exigences.

6.4.1 De RSML vers des représentations textuelles

Même si RSML est proche de la langue naturelle, certains utilisateurs peuvent préférer disposer du document d'exigences dans des formats plus conventionnels (par exemple, pour des raisons contractuelles), tels que PDF, Microsoft Word Documents ou même des documents Excel (voir Figure 6.11).

L'un des principaux avantages de l'utilisation d'un langage de modélisation est de bénéficier de techniques de l'IDM, en particulier des transformations de modèles [66], pour construire plus facilement des passerelles vers d'autres approches. Plusieurs outils, tels que ATL [121] ou Kermeta [120] entre autres, permettent de décrire ces transformations, fondées sur le méta-modèle de RSML. Notons que toutes les transformations ont été écrites par programmation en utilisant le framework de modélisation Eclipse (EMF) [15] pour faciliter leur intégration dans l'environnement de développement de RSML [16]. Dans la Figure 6.11, les artefacts RSML ont été transformés en documents MS-Word et MS-Excel.



	A	B	C
1	#	Context	Requirement description
2	1	Global	The automatic delivery drone (later called `t drone') shall allows the company to quickly c the ordered products to customer living in bi cities where the company is based.
3	2	Global	The drone shall be able to take in charge, transport and deliver a package carefully.
4	3	Global	After a delivery, the drone shall come back t warehouse.
5	2.1	Drone	**When** the drone battery is less or equal to 1 [percent] **then immediately** mode must be equal to recovery.
6	2.1.1	Drone	**When** the drone battery is less or equal to 1 [percent] **then eventually** the drone altituc must be equal to 0 **within** 30 [seconds].

FIGURE 6.11 – Exemples de documents générés à partir de l'exemple RSML de la Figure 6.6.

Ceci illustre la puissance des transformations des modèles : En utilisant le modèle des exigences et en faisant correspondre les méta-éléments correspondants, issus du méta-modèle de RSML, à des éléments d'un langage cible, nous pouvons facilement fournir une représentation des exigences adaptée à l'utilisateur.

Il est cependant important de noter que, s'il n'y a pas de format spécifique dans le document textuel, la transformation à partir de ce type de document ne peut pas être effectuée. Cela souligne la nécessité pour les documents d'exigences, d'utiliser des représentations standard comme [4, 63] ou même des modèles pour écrire les exigences comme [74, 144]. ou encore de les structurer à la manière de [149].

6.4.2 De RSML vers d'autres approches d'ingénierie des exigences et inversement

Nous avons réalisé une étude approfondie des approches d'ingénierie des exigences basées sur le langage naturel, semi-formelles et formelles dans [44]. Cette étude met en avant le rôle de la formalisation dans les exigences d'un système et les avantages, souvent complémentaires des différents types d'approches. Si la transformation de RSML en représentation textuelle permet aux utilisateurs de passer à leurs outils textuels habituels, les transformations vers d'autres représentations peuvent également être intéressantes.

Par exemple, SysML [164]est de plus en plus utilisé dans l'ingénierie des systèmes basés sur des modèles et fournit un diagramme des exigences dédié. Les exigences ainsi exprimées peuvent ensuite être liées à d'autres artefacts, tels que d'autres exigences, des blocs (parties de systèmes) ou encore des cas de tests. Par conséquent, pour permettre aux utilisateurs de bénéficier de la sémantique de RSML (y

15. `http://www.eclipse.org/modeling/emf/`
16. https ://gitlab.com/fgalinier/RSML/-/blob/master/README.md

compris de son environnement et de ses outils de validation), tout en utilisant SysML, nous proposons une transformation de RSML en SysML.

Nous avons écrit des règles de transformation faisant correspondre les exigences et les relations entre les artefacts RSML et SysML, en utilisant la sémantique définie dans la norme SysML. (SysML est utilisé avec une sémantique différente selon les outils et le contexte, mais il est assez facile de modifier les règles de transformation pour les adapter). L'exemple donné Figure 6.6 est transformé en diagramme des exigences SysML dans la Figure 6.12 ci-dessous. Avec des outils tels que Papyrus [77] ou Enterprise

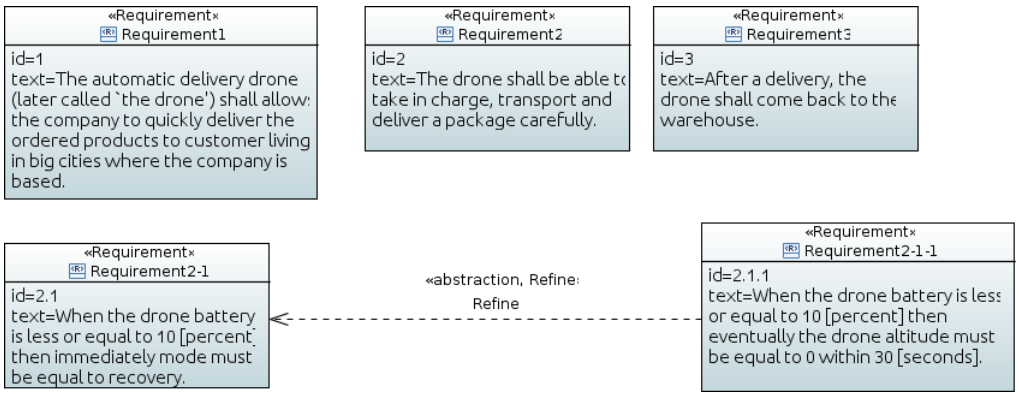


FIGURE 6.12 – Representation en SysML des exigences RSML de la Figure 6.6

Architect [182], les ingénieurs peuvent utiliser le diagramme des exigences SysML pour intégrer les exigences exprimées en RSML dans les projets de conception.

Par ailleurs, s'il peut être intéressant d'écrire des exigences RSML et de les transformer en SysML, la transformation inverse peut également être utile. En effet, l'utilisateur peut vouloir importer des exigences existantes exprimées en SysML, ou continuer à modéliser les exigences à l'aide de représentations graphiques SysML plutôt que d'utiliser la notation textuelle RSML.

Ainsi, la Figure 6.13 fournit le code RSML résultant de la transformation du diagramme des exigences SysML de la Figure 6.12. Le contexte et la connaissance du domaine n'y apparaissent pas puisqu'ils ne font pas partie du méta-modèle de SysML.

SysML.xmi | drone.rsml

```
[1] "[
    The automatic delivery drone (later called `the drone') shall allows the company to quickly
deliver the ordered products
    to customer living in big cities where the company is based.
]"

[2] "[
    The drone shall be able to take in charge, transport and deliver a package carefully.
]"

[3] "[
    After a delivery, the drone shall come back to the warehouse.
]"

[2.1] "[ When the drone battery is less or equal to 10 [percent] then immediately mode must be
equal to recovery. ]"

[2.1.1] (refines [2.1]) "[ When the drone battery is less or equal to 10 [percent] then
    eventually the drone altitude must be equal to 0 within 30 [seconds]. ]"
```

FIGURE 6.13 – Transformation SysML2RSML Figure 6.12

Étant donné que les exigences SysML ne sont que textuelles, elles ont été transformées en exigences en langages naturel dans RSML. *id* et *text* ont été mis en correspondance avec le champ ad-hoc dans RSML.

Afin d'améliorer la qualité des exigences générées, il est possible d'ajouter un analyseur syntaxique des exigences textuelles, permettant de détecter si certaines d'entre elles correspondent à la syntaxe

RSML des exigences formelles. Il serait ainsi possible de transformer les exigences 2.1 et 2.1.1 en exigences formelles RSML, en supprimant simplement les guillemets (puisqu'elles utilisent déjà la syntaxe adéquate).

En outre, on peut noter que les relations entre SysML et RSML sont préservées. Il est donc possible d'utiliser la formalisation RSML pour valider les relations entre les exigences SysML. Nous avons en effet veillé à ce que chaque type de relations SysML soit associé à une relation RSML (par exemple, la relation « raffiner» de SysML est associée au raffinement de RSML, la relation «deriveRqt» de SysML à l'ajout de RSML, etc.). L'ensemble des exigences peut ainsi être analysé pour vérifier la validité des relations, en utilisant la sémantique associée.

Une fois que nos travaux en cours sur la formalisation des relations d'exigences seront terminés, nous avons également l'intention de valider l'utilité de traduire ces relations en relations d'autres approches. Nous pourrons ainsi analyser un diagramme d'exigences SysML, en nous fondant sur sa représentation en RSML.

J"ai présenté ici les transformations vers et à partir de SysML, essentiellement dans le but d'illustrer les possibilités qu'offre la combinaison de RSML et des transformations de modèle. Cette approche peut être étendue à d'autres formalismes tels que KAOS ou i* [197] par exemple, qui permettront aux utilisateurs de concilier leurs outils habituels et les avantages de ces approches, avec les bénéfices de la formalisation fournie par RSML.

Par ailleurs, nous avons utilisé dans le cadre de ce travail une formalisation en Eiffel, mais d'autres représentations formelles, telles que Event-B ou Alloy, peuvent également être mises en correspondance avec RSML, et peuvent donc être des cibles de transformations.

6.5 Mise en œuvre de RSML

Pour illustrer notre approche, nous avons utilisé une implémentation Xtext de RSML dans Gemoc Studio [17] : Dans un premier temps, pour valider que cette approche est indépendante du domaine, nous l'avons appliquée à différents cas d'utilisation bien connus comme le système de train d'atterrissage (LGS) [35] ou le système du service d'ambulance de Londres (LAS) [138]. Afin de pouvoir avoir des remarques et des tests de l'utilisation de RSML, nous avons rendu publiques les descriptions en RSML du LGS [18] et du LAS [19].

Dans les deux cas, nous avons constaté que l'éditeur disponible de RSML amène les utilisateurs à exprimer facilement les exigences. En fait, les utilisateurs non spécialistes ont produit la version RSML des exigences qui leur ont été données en langage naturel.

Les formulations ont été vérifiées avec Autoproof et nous avons découvert des erreurs de validation et de vérification. Dans l'étude de cas du LGS par exemple, l'utilisation d'Autoreq a permis de détecter des erreurs dans une spécification existante de l'Abstract State Machine du système [158]. Ainsi, l'utilisation conjointe de la sémantique de RSML et d'Autoproof permet de vérifier le système lui-même, en ce qui concerne ses exigences. Dans le LAS, le problème révélé par Autoproof a mis en évidence qu'une exigence était mal comprise et devait être corrigée pour obtenir un ensemble d'exigences valables [87].

Nous avons également appliqué notre approche à un cas d'utilisation plus important. Les exigences utilisées pour ce nouveau cas d'utilisation concernaient un système de drônes et ont été décrites en langage naturel dans un document [20]

Nous avons d'abord produit une spécification en RSML, en formalisant certaines des exigences figurant sur le document (et en en rédigeant d'autres en utilisant le langage naturel) [21].

Nous avons obtenu 6 descriptions du domaine et 51 exigences RSML. L'outil a généré 51 exigences (20 exigences formelles et 31 exigences en langage naturel), réparties en 8 classes Eiffel. Dans un

17. `https://gitlab.com/fgalinier/RSML`
18. `https://gitlab.com/fgalinier/rsml-examples`
19. `https://gitlab.com/fgalinier/LAS`
20. `https://gitlab.com/fgalinier/RSML/blob/master/use-case/ADDE.md`
21. disponible ici : `https://gitlab.com/fgalinier/RSML/blob/master/use-case/drone.rsml`

deuxième temps, nous les avons affinées en 8 nouvelles classes, avec 51 nouvelles exigences formelles (raffinement d'autres exigences). Ces exigences écrites en code Eiffel fournissent un cadre pour développer une version (simulée) du système. Nous avons ensuite mis en œuvre 10 classes (avec 32 méthodes qui étaient des améliorations directes des fonctions d'exigences formelles). L'utilisation d'Autoproof et du *Design-By-Contracts*, nous a permis de nous assurer qu'à tout moment du projet, le système développé répondait correctement aux exigences (et que toutes les exigences étaient satisfaites).

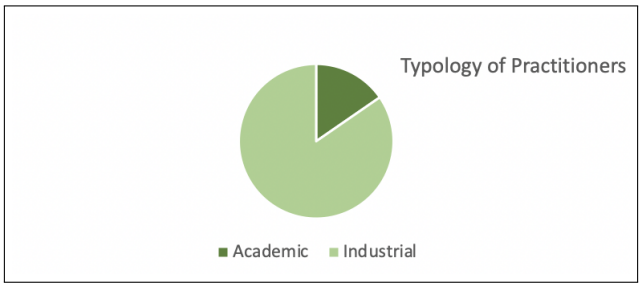


FIGURE 6.14 – Profil des participants

Enfin, nous avons mené une étude pour valider l'utilité d'une description en RSML. Nous avons travaillé avec un ensemble d'utilisateurs, la plupart d'entre eux exprimant généralement les exigences en langage naturel ou dans des langages de modélisation tels que SysML. Nous leur avons d'abord présenté RSML, en leur fournissant quelques explications sur la syntaxe et l'outil support [22]. Nous leur avons ensuite demandé d'exprimer les exigences du projet proposé. Les réactions et les antécédents des utilisateurs ont été recueillis dans le cadre d'une enquête [23] Nous avons obtenu 11 réponses à partir de l'ensemble initial, avec une grande majorité d'utilisateurs industriels provenant d'entreprises comme Airbus, Thales, Capgemini ou Orange (d'autres utilisateurs provenant de start-up).

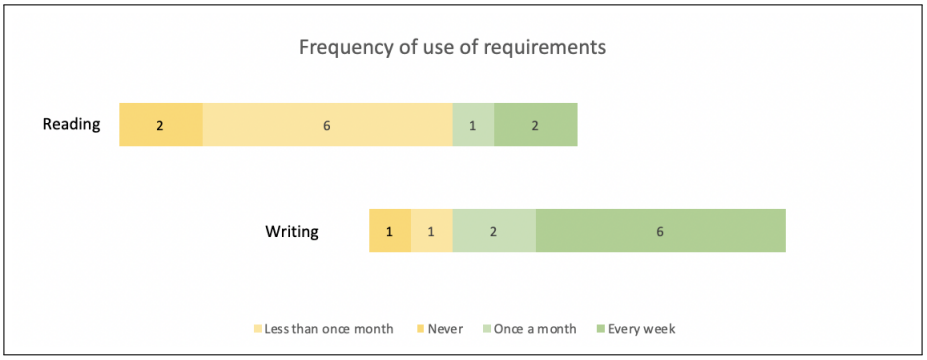


FIGURE 6.15 – Usages courants des prospects : fréquence d'utilisation des exigences

Parmi les testeurs, on peut voir Figure 6.14 et Figure 6.15 que :

- 81% des utilisateurs étaient industriels, les 19% autres étaient des universitaires (seniors ou doctorants),
- 54% lisent régulièrement des exigences (chaque semaine), 46% moins souvent,
- 18% rédigent des exigences fréquemment, 63% une fois par mois, 19% jamais.

En ce qui concernent leurs habitudes et l'impact des exigences dans leur travail (cf. Figure 6.16) :

22. Cf. `https://gitlab.com/fgalinier/RSML/blob/master/README.md`.
23. `https://docs.google.com/forms/d/e/1FAIpQLSe9YhPSjn06J2sH0oo7qLvCIdr12YZPcFosXaZy4n1h viewform?usp=pp_url` - Voir aussi `https://docs.google.com/spreadsheets/d/1Iwc-yrwsHlLxAnpYxbJAPAvvByHDxFxrJpVwhhkbepg/edit?usp=sharing` pour l'ensemble des données.

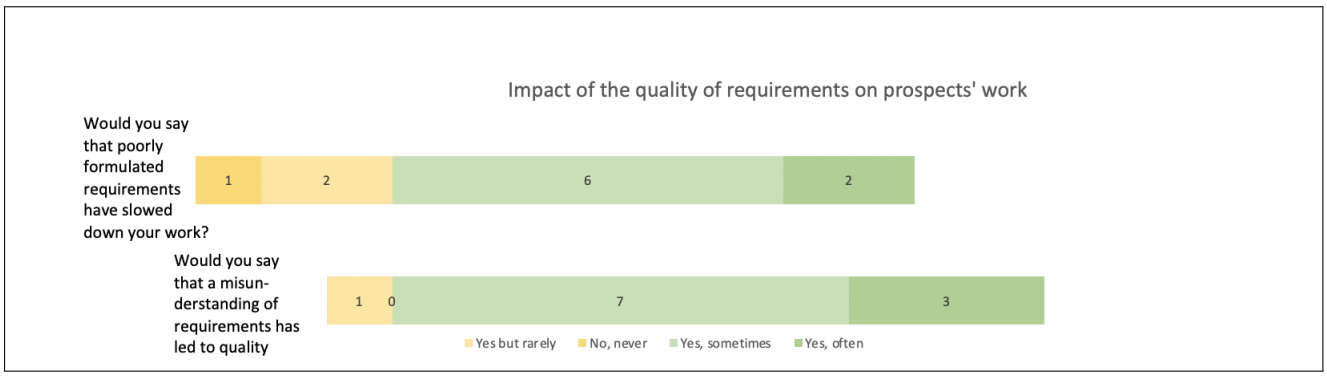


FIGURE 6.16 – Impact d'exigences mal formulées sur le travail des participants

- Tous sont d'accord sur le fait que des exigences mal formulées ont ralenti leur travail (63% parfois, 27% souvent, 10% rarement).
- Seuls 27% ont répondu qu'une mauvaise compréhension des exigences n'avait pas (ou rarement) entraîné de problèmes de qualité dans leur travail (pour 18% c'est souvent le cas, pour 54% parfois).
- Seulement 28% ont utilisé des approches spécifiques aux exigences (comme Cucumber, Gherkin, User Stories). , moins de 10% ont utilisé des outils dédiés aux exigences.

A propos de l'*utilisabilité* de RSML et de son outil support, nous avons demandé un retour d'informations sur la facilité d'utilisation, la qualité des exigences obtenues, les perspectives d'adoption de RSML. Les notes allaient de 0 (pas d'accord) à 5 (tout à fait d'accord). La *facilité d'écriture des exigences RSML* a obtenu une note moyenne de 3 (avec 27% de 2, 27% de 3 et 36% de 4), tandis que le *facilité de lecture des exigences RSML* a obtenu une note moyenne de 4,54 (sans note inférieure à 4). Une majorité d'utilisateurs a déclaré qu'il est plus facile d'écrire et de lire les exigences RSML que les exigences formelles. (cf. Figure 6.17)

Le questionnaire nous a également permis de constater qu'un effort d'apprentissage doit être fait, étant donné qu'à l'affirmation concernant RSML "*J'ai appris à l'utiliser rapidement*", nous obtenons une note moyenne de 2,45 (avec une note maximale de 3), et la note moyenne sur la confiance de l'utilisateur dans les exigences qu'il a écrites en RSML n'est que de 2,91.

Du point de vue de la perspective d'intégration de RSML dans les entreprises, à la question : "*Dans mon travail, les exigences de rédaction avec RSML peuvent m'aider à être plus efficace*", la note moyenne est de 3 (avec 64% de 3 ou plus). À la question "*Dans mon travail, les exigences décrites en RSML peuvent m'aider à être plus productif*", la réponse moyenne est de 2,81 (avec 64% de 3 ou plus). 55% donnent un 4 ou un 5 à l'affirmation suivante : "*Dans mon travail, les exigences décrites en RSML peuvent m'aider à assurer la qualité des systèmes*". (avec une note moyenne de 3,36).

Concernant la *qualité* de la description RSML du système de drônes envoyée par les testeurs, on peut dire que par rapport à la description que nous avons faite au départ, les résultats sont satisfaisants à 75%.

Enfin, les résultats obtenus sont encourageants car les utilisateurs habitués à travailler avec des exigences ont trouvé l'approche satisfaisante et les utilisateurs occasionnels d'exigences ou habitués à travailler en langue naturelle (i) n'ont pas eu de difficulté majeure à utiliser l'approche, et (ii) l'ont trouvée intéressante quant au niveau de formalisation fourni.

Biais à la validité de cette étude

L'évaluation empirique que nous avons réalisée est très préliminaire et fondamentale. Elle n'avait pas pour but de valider pleinement l'approche, mais de nous donner un premier retour sur son applicabilité industrielle. Nous sommes néanmoins conscients des limites de notre étude, en raison de sa nature informelle. De nombreux autres biais pourraient être exprimés, comme la faible quantité de population, mais nous nous sommes assurés de la représentativité de cette population et de l'impartialité des réponses apportées.

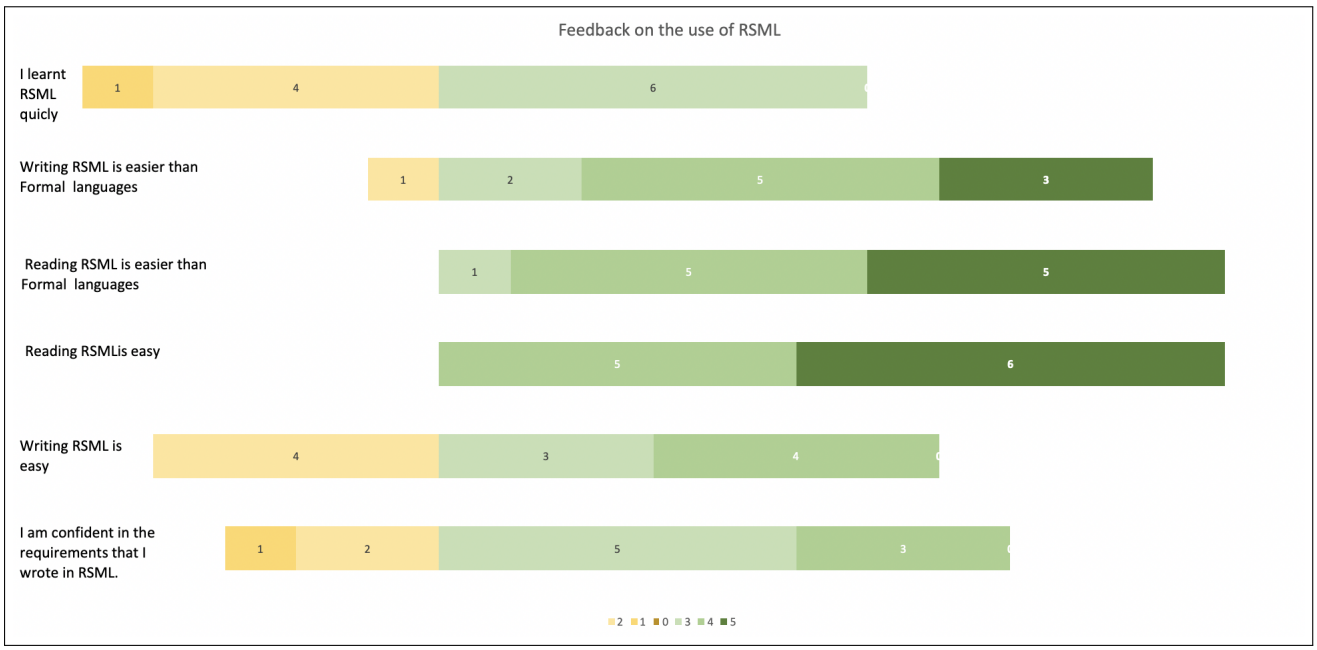


FIGURE 6.17 – Synthèse des évaluations sur la facilité d'utilisation et l'efficacité de RSML

6.6 Conclusion sur RSML

L'approche que nous proposons conduit à une utilisation plus facile de la représentation formelle des exigences.

RSML est un langage dédié aux exigences qui est proche du langage naturel, tout en supportant une sémantique formelle. Il est basé sur un méta-modèle permettant l'expression des exigences à la fois en langage naturel et dans un langage formel. RSML est défini sémantiquement et basé sur la conception par contrats. Ses expressions peuvent ainsi être analysées à l'aide de solveurs et les utilisateurs non-experts peuvent ainsi bénéficier de cette formalisation inhérente au langage de façon transparente.

RSML vise à être intégré dans une approche homogène, à réduire l'écart entre les exigences en langage naturel et les représentations formelles, et à fournir des moyens de s'adresser à plusieurs types de parties prenantes (utilisateurs d'outils de modélisation, utilisateurs de langages naturels ou même utilisateurs d'approches plus formelles).

Les premières expériences nous ont donné de bons retours sur l'utilisation de RSML. Sa mise en œuvre sur des études de cas et l'expérimentation menée sur un pannel d'utilisateurs s'est avérée utile pour confirmer qu'il :

(i) est facilement utilisable par les parties prenantes habituées à travailler en langage naturel,

(ii) permet une expression formelle des exigences et ainsi leur validation par un prouveur et

(iii) permet la vérification des systèmes par rapport à leurs exigences grâce à des liens de traçabilité.

Les avantages de l'approche que nous avons proposée autour de RSML sont multiples : d'une part, elle permet de définir un *Domain Specific Modelling Language* accessible aux non-experts, qui fournisse une description formelle des exigences et d'autre part, son adoption du principe du modèle unique offre des garanties de traçabilité des exigences tout au long du cycle de développement.

Par ailleurs, en s'inscrivant dans l'idée de globalisation des langages de modélisation [92], elle peut être utilisée pour combler l'écart avec d'autres approches, et par là même d'intégrer des modèles d'analyse hétérogènes.

En proposant RSML nous avons atteint l'objectif fixé de

fournir un langage d'interface d'expression des exigences, utilisable et exploitable par les parties prenantes, qui fournisse les constructions nécessaires aux mécanismes de traçabilité, preuves, etc. inhérents aux divers métiers, et par conséquent aux divers points de vue. ‘

Nous avons mis en œuvre *des techniques et outils de l'IDM pour fournir des éléments permettant d'exprimer les exigences dans un langage suffisamment proche du langage naturel pour être communément utilisé et suffisamment formel pour supporter des vérifications et des preuves.*

En utilisant les techniques et les outils du MBSE, nous proposons également quelques ponts vers des langages existants pour permettre aux ingénieurs de travailler avec leurs outils habituels. Nous traitons plusieurs questions d'ingénierie des exigences, en fournissant un moyen de transformer les expressions d'exigences habituelles à la fois vers et depuis RSML (que nous considérons alors comme un pivot entre différents espaces d'expression). En considérant RSML comme un langage pivot, nous avons fournit des pistes pour

gérer l'hétérogénéité des artefacts produits lors de l'expression des exigences, en garantissant leur inter-opérabilité et la cohérence de l'ensemble.

Nous avons ici *utilisé les mécanismes fournis par l'IDM pour supporter la gestion d'exigences via des modèles semi-formels et garantir la cohérence des modèles produits.*

Par ailleurs, en intégrant RSML à l'approche SIRCOD,

nous avons défini une approche permettant de n'avoir qu'un seul ensemble cohérent d'artefacts, qui évoluent de manière coordonnée au fur et à mesure du développement.

Nous avons en effet *proposé et développé une méthode outillée qui mette en œuvre de manière concrète les principes du développement « sans couture » et du modèle unique.*

Ainsi, RSML, et SIRCOD, apportent des éléments de réponse aux questions :

Question 3 : Comment garantir la cohérence de la conception par rapport à l'analyse ?

Question 4 : Comment assurer la cohérence de l'ensemble en permettant à chacun de travailler avec ses propres outils ?

6.7 Discussion

Certaines validations sont bien entendu encore nécessaires. Cette approche et ses outils supports seront testés par un plus grand nombre de personnes, avec des niveaux d'expertise plus tranchés en matière d'ingénierie des exigences et sur une étude de cas industrielle. Cela permettrait de valider l'évolutivité de l'approche, en termes de nombre et de confidentialité des parties prenantes, de nombre et de types d'exigences, de nombre et de types de relations entre elles.

Par ailleurs, afin d'éprouver les techniques de transformations inter espaces de formalisation des exigences, nous ajoutons actuellement de nouvelles transformations vers d'autres approches, comme KAOS. Ceci devrait confirmer l'extensibilité de notre approche dans un contexte plus global.

Un autre travail en cours porte sur les définitions sémantiques des relations entre exigences. Il est basé sur une approche mathématique, qui doit encore être validée [149]. La définition sémantique conduit en effet à un environnement complet de validation des exigences [87] :

(i) les exigences, si elles sont formelles, peuvent être utilisées dans le processus de validation à l'aide d'un prouveur,

(ii) en utilisant les relations et leur sémantique, la validité d'une relation peut être détectée et

(iii) par propagation de la satisfaction sur les relations [88], les exigences non formelles peuvent également être validées en utilisant les deux points précédents.

Troisième partie

Conclusion et Perspectives

Chapitre 7

Conclusion : Ingénierie Système mettant en œuvre des techniques de l'IDM centrée utilisateur

Nos contributions participent à l'amélioration de la modélisation des systèmes complexes en intégrant des mécanismes de points de vue (opérateurs de composition, modèle global virtuel, gestion de l'évolution, outils supports) mis à disposition des parties prenantes, d'abord dans le cadre de modèles de conception homogènes, puis de modèles de conception hétérogènes et enfin de modèles d'exigences hétérogènes.

Nous fournissons à des utilisateurs isolés par leur propre point de vue sur le système, des approches d'ingénierie des systèmes basées sur les techniques de l'IDM, et des mécanismes collaboratifs et semi-automatiques qui facilitent leur modélisation du système à réaliser.

L'implication des parties prenantes et de leurs points de vue garantit, notamment grâce à la collaboration mise en œuvre, un modèle global cohérent du système considéré. Nous permettons en effet de considérer la modélisation d'un système complexe, non plus comme un ensemble de modèles partiels hétérogènes dont la cohérence est à garantir par seul un expert du domaine, mais comme un ensemble structuré de modèles partiels hétérogènes dont la cohérence est garantie par la collaboration de ses parties prenantes et l'utilisation d'outils de vérification.

La Figure 7.1 se veut une illustration de la synthèse de nos travaux et de leurs enchaînements.

Notons que nos travaux contribuent également à permettre à de jeunes chercheurs d'améliorer leurs compétences, dans le cadre de coopérations franco-marocaines qui perdurent encore. Nous avons monté avec succès des projets de chercheur invité, nous proposons des sujets de thèse communs dans le cadre de thèses en co-tutelle et nous collaborons au montage de projets Hubert Curien ou de bourses Eiffel.

Les chercheurs avec lesquels nous avons collaborés sont enseignants chercheurs au Maroc, ou en passe de le devenir. Ils sont très impliqués dans l'enseignement du Génie Logiciel, la recherche et les réseaux de collaboration internationaux. M. Nassar par exemple, après avoir été Responsable du département Informatique de l'ENSIAS de Rabat en est actuellement le Responsable des Relations Internationales et le directeur d'un CEDOC (Ecole doctorale).

L'une de nos contributions principale est un support à l'alignement collaboratif de modèles de conception hétérogènes évolutifs

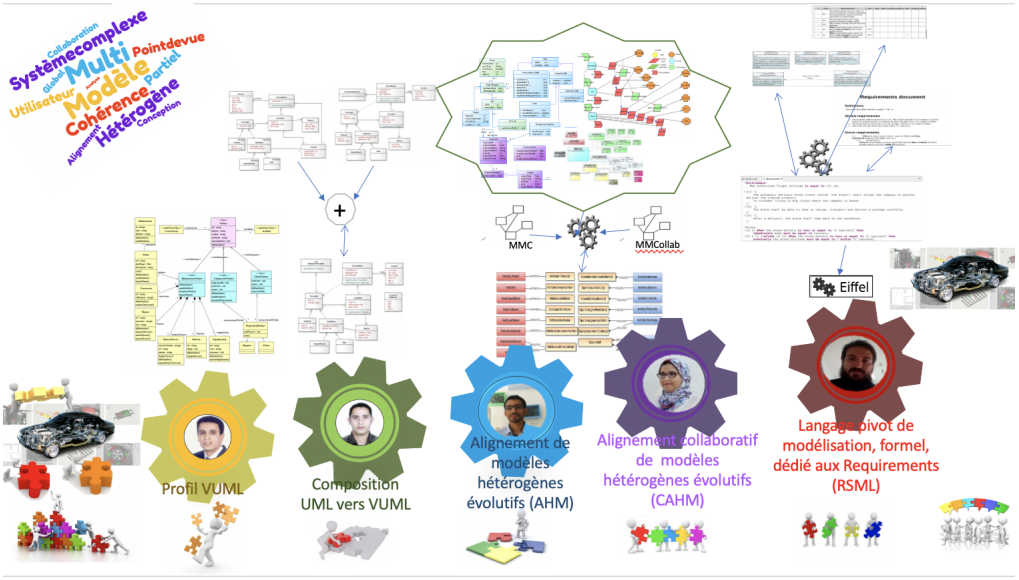


FIGURE 7.1 – Synthèse de nos contributions

7.1 Alignement collaboratif de modèles de conception hétérogènes évolutifs

7.1.1 Contribution

Avec l'approche Collaborative Alignement of Heterogeneous Models, CAHM, qui s'appuie sur Alignement of Heterogeneous Models (AHM, développé dans la thèse de M. El Hamlaoui) et précédemment View-based UML (VUML, proposé dans les travaux de A. Anwar), nous avons proposé, dans le cadre de la thèse de Saloua Bennani, un ensemble de concepts qui supportent l'alignement de plusieurs modèles hétérogènes évolutifs et le maintien de cet alignement en cas d'évolution d'un ou plusieurs des modèles liés.

Cette approche est fondée sur le méta-modèle MMCollab défini dans cette thèse. MMCollab est dédié à la description de la collaboration en général et de la prise de décision en groupe en particulier. Il structure les prises de décision collaboratives en paquetages : les acteurs, le choix de la politique de prise de décision, la collecte des propositions sur lesquelles vont porter les évaluations individuelles et les décisions collectives relatives à ces propositions. Nous avons élaboré ensuite un ensemble extensible de politiques de prise de décision en groupe prêtes à l'utilisation, grâce à la mise en œuvre du patron GDMPattern introduit dans MMCollab.

CAHM s'accompagne d'un processus permettant d'établir des liens inter-modèles et de les maintenir en cas d'évolution des modèles liés. Pour cela, cette approche propose :

- un processus global semi-automatique où les utilisateurs, acteurs métier du système, établissent collaborativement des liens typés entre les éléments des méta-modèles des modèles à aligner, des méta-correspondances,
- un processus outillé de propagation et de filtre de ces méta-correspondances au niveau des modèles, pour produire un modèle de correspondance qui garantit la cohérence du système global. Cette opération se base sur des expressions sémantiques qui sont associées aux types de relations qui caractérisent les (méta-)correspondances entre éléments de (méta-)modèles,
- un outil support HMCS-Collab, (pour Heterogeneous models Matching and Consistency management Suite - Collaborative version).

HMCS-Collab, l'outil support à l'approche, assure :

- les sessions collaboratives d'élaboration des méta-correspondances,
- la propagation automatique des méta-correspondances du niveau des méta-modèles à celui des modèles, et par là même la génération du modèle de correspondance,
- des sessions de prise de décision collective de propagation des méta-correspondances dont la sémantique n'est pas suffisamment formelle, pour compléter le modèle de correspondance,
- la détection automatique des changements opérés sur les modèles source,
- l'application de résolutions automatiques présentes dans un catalogue des résolutions,
- en cas d'incohérences qui ne peuvent être traitées automatiquement, des sessions collaboratives de choix de résolutions possibles ou d'élaboration de nouvelles résolutions quand aucune solution ne convient au groupe de décideurs.

7.1.2 Positionnement vis-à-vis de l'état de l'art

Certaines approches existantes prennent en compte différents points de vue intervenant dans la modélisation des systèmes complexes. Mais la plupart de ces approches, si elles prennent en compte les préoccupations d'un métier [92], les alignements qui peuvent être faits entre ces modèles [37, 47, 50, 58, 140, 190] ou encore l'évolution de ces modèles [107, 127], restent partielles dans la mesure où elles ne cherchent à traiter qu'une partie des problèmes. Les approches de composition de différents points de vue en une seule représentation, proposées dans la littérature [20, 73, 130, 131, 200], reposent sur l'élaboration d'un modèle global et présentent trois inconvénients majeurs liés à l'hétérogénéité des modèles.

Le premier concerne la structure du méta-modèle associé au modèle composé ; en effet, il n'y a pas de consensus sur la manière de le construire : à partir de l'union de tous les éléments provenant des modèles sources ou de leur intersection, ou d'une autre combinaison ?

La deuxième question concerne la sémantique utilisée pour représenter un élément d'un modèle composé étant donné que les modèles sources peuvent utiliser des sémantiques différentes.

Enfin, un modèle global obtenu par composition peut être d'une taille très importante dans le cas de systèmes complexes et donc très difficile à maintenir.

En général, les approches telles que celles de Brunelière et al. ou encore Cicchetti et al. [48, 57], présentent des lacunes à deux moments du processus de mise en correspondance : avant et après la création du modèle de correspondance.

En amont de la création du modèle de correspondance, on constate un manque d'équilibre entre la capacité d'exprimer les correspondances et leur réutilisabilité. Les approches existantes se basent principalement sur un seul de ces éléments, car la réutilisabilité se fait au prix d'une moindre expressivité et inversement, une garantie de réutilisabilité se fait au détriment de l'expressivité. De plus, ces approches ne gèrent que des correspondances binaires et ne peuvent donc pas établir de correspondances n-aires complexes reliant un élément de modèle à un ensemble d'éléments appartenant à d'autres modèles.

En aval de la création du modèle de correspondance, on peut noter que les approches étudiées produisent un modèle de correspondance pour chaque paire de modèles d'entrée ; ainsi pour n modèles d'entrée, (n * (n-1))/2 modèles de correspondances doivent être créés, ce qui conduit à un grand nombre de modèles séparés sans aucune connexion entre eux et rend leur gestion non seulement difficile, mais presque impossible à automatiser.

Nous produisons un modèle unique de correspondances entre les modèles d'entrée, qui supporte des relations n-aires. Nous garantissons la réutilisation des éléments obtenus et nous assurons leur pérennité en proposant d'enrichir les modèles par des concepts permettant de gérer

l'évolution des modèles et en donnant des heuristiques de collaboration gouvernant le processus dans son ensemble. En effet, les différents experts métiers doivent pouvoir collaborer pour alléger la tâche de contrôle de la mise en correspondance et de prise de décision dans les cas d'évolution. Les approches existantes ne prennent pas en compte cette double dimension d'une modélisation collaborative a posteriori [141, 117].

7.1.3 Bilan

L'approche que nous proposons implique l'utilisateur et supporte la mise en correspondance collaborative de plusieurs modèles de conception partiels et hétérogènes tout en assurant le maintien de la cohérence de l'ensemble par une gestion collaborative de l'évolution des modèles.

Elle répond en cela à trois des quatre questions que nous nous sommes posées au départ à propos de l'ingénierie des systèmes complexes :

Question 1 : Comment garantir la validité des modèles de conception dans leur ensemble ?

Question 2 : Comment garantir la cohérence des modèles de conception dans leur ensemble ?

Question 4 : Comment assurer la cohérence de l'ensemble en permettant à chacun de travailler avec ses propres outils ?

La réponse à la question 3 est donnée ci-dessous.

7.2 Langage pour l'ingénieur des exigences, dédié à la production de modèles d'analyse sans rupture

7.2.1 Contribution

L'approche que j'ai présentée ici propose un langage, Requirement Specific Modelling Language (RSML), décrit par un méta-modèle qui a été défini dans la thèse de Florian Galinier. Elle propose aussi un processus et un outil dédiés à l'expression d'exigences. Les exigences sont définies dans un langage naturel contraint, adapté aux pratiques courantes des parties prenantes en Ingénierie des Exigences, qu'ils soient experts des méthodes formelles ou qu'ils ne le soient pas, comme c'est le plus souvent le cas [5, 185].

L'utilisation de l'IDM permet de combler le fossé avec d'autres approches, permettant à davantage de spécialistes des exigences d'être impliqués dans un projet, en fournissant un moyen de transformer des exigences d'un langage de modélisation (SysML par exemple) en RSML.

Fournir un niveau de formalisation permettant de vérifier la spécification est fondamental pour garantir la fiabilité des systèmes. La sémantique de RSML est supportée par l'approche SIRCOD, qui fournit un moyen simple d'exprimer formellement les exigences en Eiffel (et de s'assurer qu'elles sont toutes formalisées), au moyen notamment d'assertions sémantiques

intégrées au langage et des mécanismes d'héritage multiple, de redéfinition, d'abstraction, etc. Les exigences peuvent ainsi être soumises à un prouveur (Autoproof).

Pour à la fois faciliter l'écriture des exigences en RSML et apporter des éléments de vérification, nous avons également pris en compte deux types de traçabilité entre exigences : la traçabilité verticale qui consiste à assurer qu'une exigence est bien prise en compte dans le système réalisé au fur et à mesure de son développement, et la traçabilité horizontale qui consiste à permettre une navigation entre exigences, fondée sur des relations dont nous avons fourni la sémantique.

Je noterai enfin que RSML permet d'exprimer tout type d'exigences, fonctionnelles, non-fonctionnelles, ou encore environnementales, car les exigences en langage naturel non formalisables, du type "The system must be efficient" sont prises en compte par l'approche.

7.2.2 Positionnement vis-à-vis de l'état de l'art

Les approches formelles d'expression des exigences, si elles ont tendance à fournir des éléments plus proches de l'utilisateur [160, 95], ne permettent pas une intégration sans couture des exigences dans le code. En effet, ces approches permettent de faire de la vérification et de la validation d'exigences mais ne constituent pas un moyen de rendre ces exigences opérationnelles.

Ces approches supportent l'expression des exigences avec une sémantique simple et des relations entre exigences, comme raffinage ou décomposition par exemple, associées à une sémantique non formelle. Ces relations donnent une vision hiérarchique du système en termes d'exigences et sont des vecteurs de la traçabilité des exigences entre les différentes tâches de développement du système. Cependant, l'établissement de tels liens reste un travail difficile, et même si des outils automatiques existent (comme [17] ou [191]), une intervention humaine est toujours nécessaire pour les vérifier [1].

Certaines approches visent à introduire une certaine formalisation dans l'expression des exigences, tout en restant proches des pratiques des parties prenantes. Stimulus [119] par exemple, fournit un environnement et un langage faciles à utiliser pour les non-spécialistes, avec une définition sémantique basée sur Lucid Synchrone [59] et Lutin [173] qui peut être vérifiée par modélisation. Stimulus offre un moyen simple d'exprimer formellement les besoins et de les vérifier, mais il ne supporte pas l'expression de la traçabilité entre les exigences et les artefacts de développement. Pour surmonter ce problème, plusieurs approches ont tenté de normaliser la façon d'exprimer les exigences. C'est le cas d'URML [109, 29] ou même d'URN [13]. Dans ces travaux, les auteurs proposent de définir un modèle pour les exigences et les artefacts qui peuvent être liés à ces exigences. Cependant, à l'instar de SysML [10] ou de KAOS [135], ces approches sont trop spécifiques pour être utilisées de manière standard.

Dans [96], les auteurs proposent de garder des langues distinctes et de jeter des ponts entre elles. Ils utilisent des modèles virtuels pour fédérer différents langages dans un espace commun. En mettant en correspondance les artefacts de plusieurs modèles (tels que KAOS ou même des documents Word), ils permettent de relier les artefacts dans plusieurs outils. Cependant, il s'agit d'une approche générique, non spécifique aux exigences, qui oblige les utilisateurs à écrire de nouveaux pilotes pour inclure de nouveaux outils. Elle tente de concilier dans un modèle virtuel commun plusieurs espaces techniques et ne permet donc pas de définir de sémantique pour ce modèle, qui sera plutôt introduite par l'un des espaces techniques.

7.2.3 Bilan

Avec RSML, nous répondons à un besoin de langage à la fois suffisamment informel pour permettre une manipulation aisée par les parties prenantes et suffisamment formel pour permettre de procéder à des vérifications et faciliter des validations. Ceci a lieu dans le cadre d'une approche homogène et sans couture visant à intégrer les exigences dans le code et supportée par un environnement de développement.

L'approche proposée fournit des éléments de réponse aux questions concernant l'ingénierie des systèmes complexes suivantes :

Question 3 : Comment garantir la cohérence de la conception par rapport à l'analyse ?

Question 4 : Comment assurer la cohérence de l'ensemble en permettant à chacun de travailler avec ses propres outils ?

Chapitre 8

Perspectives : Ingénierie des connaissances et apprentissage automatique pour l'alignement collaboratif de modèles hétérogènes

8.1 Ingénierie des connaissances et apprentissage automatique pour l'alignement de modèles hétérogènes

8.1.1 Alignement de modèles de conception

Lors de la mise en correspondance des modèles de conception, nous faisons la supposition d'une absence d'incohérence entre les modèles partiels à composer. Si les conflits syntaxiques peuvent être facilement identifiés et résolus, il n'en est pas de même des conflits sémantiques qui incluent les conflits structurels. Pour mettre en œuvre la mise en correspondance des modèles de conception de façon plus efficace, il faudrait réaliser en amont une opération d'alignement syntaxique et sémantique de ces modèles. Cette phase d'harmonisation consiste à identifier les dépendances entre les modèles partiels, à caractériser les incohérences portant sur des termes (conflits de synonymie et d'homonymie par exemple) ou sur des structures (conflits entre relations de spécialisation et d'association, etc.) et à résoudre les incohérences de façon semi-automatique.

Pour réaliser cette résolution de conflits de façon automatisée, nous pensons qu'il faut faire appel à des techniques d'ingénierie des connaissances - comme les ontologies - qui sont employées avec succès dans le domaine du web sémantique, en les combinant à des heuristiques de conception [168, 166, 170]. L'originalité consiste ici à marier l'ingénierie des connaissances avec l'IDM pour automatiser l'alignement sémantique de modèles. Nous avons fait une première expérimentation, en collaboration avec une chercheur spécialisée en ontologies, qui a donné lieu à une publication dans le workshop international GEMOC [105] et nous proposons ensemble un sujet de thèse. L'objectif est non seulement de fournir des éléments utiles à la mise en correspondance des modèles, mais également de permettre une meilleure automatisation de la prise en compte de l'évolution des modèles partiels composés. La solution à apporter nous semble donc devoir procéder à une analyse syntaxique et sémantique des modèles partiels en lien avec les domaines d'expertises concernés et la modélisation des connaissances mises en œuvre, en s'appuyant sur le domaine de l'ingénierie des connaissances [189].

Nous proposons parallèlement de supporter le processus d'alignement de CAHM en y intégrant des techniques d'apprentissage automatique. Ceci devrait aider les parties prenantes,

en apportant tout d'abord une aide à la détection automatique des correspondances potentielles inter-modèles et un filtre plus efficace des correspondances pertinentes, puis en identifiant le patron de prise de décision le plus adapté à la validation des correspondances établies [133]. Ceci, de par le support et l'automatisation qui en découle, s'inscrit notamment dans un objectif de passage à l'échelle de CAHM.

Par ailleurs, nous pensons à explorer de nouvelles approches de modélisation et de méta-modélisation exploitant les techniques d'apprentissage automatique [106, 52, 169].

Nous partageons l'idée que l'alignement des modèles hétérogènes évolutifs d'un système complexe doit également bénéficier de l'émergence de l'apprentissage automatique. Cette approche pourrait intervenir à plusieurs niveaux, que ce soit pour :

(i) le guidage des parties prenantes lors de la réalisation distribuée de leurs modèles [171],

(ii) l'optimisation du processus automatique de mise en correspondance des concepts des modèles partiels, s'appuyant sur des ontologie de domaine, mais aussi apprenant des filtres mis en place, et permettant notamment d'alimenter le catalogue de résolutions proposé dans CAHM, de façon automatique [67],

(iii) faciliter la réutilisation de modèles existants et des modèles de correspondances associés aux modèles de conception par exemple ([159]),

(iv) permettre ainsi le passage à l'échelle des approches que nous avons proposées.

8.1.2 Alignement de modèles d'exigences

L'intégration de la notion de point de vue à la phase d'analyse permet d'identifier en premier lieu l'ensemble des champs de préoccupation. Cette identification est une description précise des différents phénomènes fonctionnels, technologiques et métiers d'un système. Les modèles qui sont élaborés séparément en phase d'analyse sont fournis selon plusieurs points de vue. Ils sont ainsi focalisés sur des domaines distincts et réalisés dans des langages différents.

Comme les modèles de conception, ils vont pouvoir être alignés pour assurer la complétude et la cohérence globale du système considéré. C'est l'une des perspectives de l'étude que nous avons menée autour du langage RSML. Nous avons proposé avec RSML, un langage proche des utilisateurs qui permet de créer un pont entre des modèles d'analyse (modèles SysML par exemple) et une spécification formelle (en Eiffel) des exigences représentées dans ces modèles.

L'idée que nous poursuivons ici consiste à proposer des alignements de modèles d'exigences qui permettraient d'obtenir un seul programme RSML, d'où un modèle unique des exigences du système pourrait être déduit et vérifié.

La solution à étudier consiste à utiliser RSML comme langage pivot entre des espaces technologiques différents pour gérer l'hétérogénéité des artefacts produits lors de l'expression des exigences, en garantissant la cohérence de l'ensemble. Nous prévoyons que cette étude puisse être menée selon 3 axes :

1. En nous fondant sur les transformations de modèles, nous avons montré que nous pouvions obtenir un programme RSML à partir de modèles d'exigences, notamment semi-formels comme SysML. Nous avons vu que les relations entre SysML et RSML sont préservées et qu'il est donc possible d'utiliser la formalisation RSML pour valider les relations entre les exigences SysML. En associant chaque type de relations SysML à une relation RSML, en se fondant sur la sémantique associée et en s'appuyant sur une formalisation des relations d'exigences, nous pouvons assurer l'analyse de l'ensemble des

exigences et la vérification de la validité des relations. Ce mécanisme rend possible l'analyse d'un diagramme d'exigences SysML, en se fondant sur sa représentation en RSML.

Cette approche peut être étendue à d'autres formalismes tels que KAOS ou i* [197] par exemple, qui permettront aux utilisateurs de concilier leurs outils habituels et les avantages de ces approches, avec les bénéfices de la formalisation fournie par RSML.

Nous avons également commencé une étude pour supporter la transformation en RSML de langages formels d'expression des exigences comme Event-B ou Alloy. La syntaxe et la sémantique bien définies de ces langages devraient faciliter leur transformation en RSML.

Les éléments non formels, comme un document en langage naturel, doivent par contre intégrer des structurations rigoureuses pour pouvoir être traduits efficacement en RSML. L'utilisation de techniques éprouvées de Natural Language Processing [7, 80, 16, 184, 98] qui permettent d'analyser un texte, ses termes et sa structure, nous semblent pouvoir être utilisées pour permettre une transformation satisfaisante en termes de formalisation RSML, d'un document exprimé en langage naturel. Certaines structure lexicales du type "when this then do that" sont tout à fait repérables et transformables ; il serait intéressant d'étudier dans quelle mesure les techniques de NLP permettent une généralisation à des expressions plus vagues. Des études ont également montré qu'un modèle SysML ou UML pouvait en partie être généré à partir d'une analyse de texte, en utilisant ces techniques [55, 178].

2. Ensuite, nous pensons qu'il est possible d'effectuer un alignement des différents programmes RSML générés, voire de les fusionner, en s'appuyant sur le méta-modèle de RSML. Ceci permettrait de révéler les répétitions, mais aussi les incohérences éventuelles (en s'appuyant sur l'outil de preuve Autoproof) et garantirait un seul ensemble cohérent d'exigences.

3. Enfin, pour optimiser ce processus, nous proposons d'envisager une harmonisation préalable des modèles d'exigences fondée sur les termes et les structures. Nous prévoyons une étude de cas de l'utilisation des ontologies [71, 18] et des alignements d'ontologies [186], pour un alignement des modèles d'exigences, dont le glossaire commun défini en amont de la production des modèles partiels devrait simplifier le processus d'alignement.

8.2 Intégration des patrons de collaboration à l'Ingénierie Système

La collaboration reste un élément fondamental que nous voulons continuer à exploiter. En effet, elle apparaît comme un élément clé de la réussite des projets, en facilitant le contrôle de la couverture du domaine et de la cohérence des artefacts produits, tant au niveau de l'analyse que de la conception. (C'est également le cas en phase de développement, mais ceci sort du cadre de notre travail).

8.2.1 Intégration de la collaboration proposée dans CAHM à RSML

Nous projetons pour commencer de nous appuyer sur les éléments fournis dans le cadre de la thèse de Saloua Bennani, pour permettre aux spécialistes des exigences de collaborer

dans la réalisation de leurs modèles RSML respectifs et d'aligner plus facilement ces modèles. CAHM intègre un certain nombre de patrons de collaborations et de préconisations sur leur utilisation. L'approche a été appliquée à l'alignement de modèles de conception et à la prise en compte de l'évolution de ces modèles. Ces patrons sont génériques et devraient s'adapter à toute situation de prise de décision collective. C'est pourquoi il nous semble intéressant d'étudier son application à d'autres types de modèles et d'alignement.

Par ailleurs, les modèles RSML d'un système peuvent être construits de façon décentralisée, notamment dans le cas de systèmes complexes. Chaque sous-système par exemple donnera lieu à son propre modèle RSML. Le modèle devra être cohérent dans sa globalité et ses exigences vérifiées, comme celles de ses sous-systèmes, ou encore celles de leurs interfaces. Nous pensons que la gestion de cette cohérence ne peut être que favorisée par son intégration dans une collaboration active des parties prenantes. L'une des perspectives que nous voyons aux patrons collaboratifs de CAHM consiste donc à inscrire la production de programmes RSML dans un processus collaboratif.

Ensuite, la collaboration appliquée à la gestion de l'évolution des modèles, telles que proposée dans CAHM, présente un intérêt essentiel dans les cas de changements d'exigences et par conséquent d'évolution d'un ou plusieurs des modèles concernés. L'impact d'un changement doit pouvoir être calculé soit au niveau du modèle initial, soit au niveau du modèle RSML correspondant. L'enjeu ici réside dans le double objectif de mettre en œuvre les patrons de collaboration pour prendre les bonnes décisions quant aux évolutions possibles et leurs impacts, mais aussi de voir comment le catalogue de résolutions doit être constitué et alimenté, en fonction de la spécificité des modèles d'exigences.

Enfin, si RSML peut être utilisé comme langage pivot pour aligner des modèles d'exigences hétérogènes, faire collaborer les spécialistes des différents formalismes ou métiers [43] en utilisant un outil support permettant d'aligner des modèles d'analyse partiels avant leur transformation en RSML, garantirait une meilleure cohérence du modèle global et par là même du système final. Une extension de l'intégration de la collaboration à CAHM, consistera donc à proposer un support à l'élaboration (et l'évolution) de modèles d'exigences partiels hétérogènes. Nous comptons donc nous appuyer sur des éléments formels de collaboration pour intégrer les parties prenantes au processus de gestion de la cohérence des modèles d'exigences, ce qui est fondamental pour obtenir des résultats probants.

Nous avons mis en œuvre la collaboration pour améliorer l'alignement de modèles de conception hétérogènes évolutifs. Nous proposons d'une part de l'appliquer à la production de modèles d'analyse hétérogènes, mais nous prévoyons d'autre part de la généraliser à différents processus mis en œuvre dans le cadre de l'ingénierie système.

8.2.2 Intégration des patrons de collaboration de CAHM à l'Ingénierie Système

Des études de l'intégration de la collaboration au MBSE ont montré que ceci permettait d'améliorer la gestion de la complexité inhérente à ce type d'activité [53].

En ingénierie système, les outils collaboratifs sont nombreux et tous les jours améliorés [36]. Cependant, ils demeurent pour certains des "usines à gaz", où si la collaboration est exploitée, elle reste d'avantage une juxtaposition de propositions qu'une communication partagée.

En effet, la plupart des outils "collaboratifs", comme les outils dominants qui sont à disposition dans le cadre du DevOps par exemple [1], ne couvrent pas la totalité des aspects fondamentaux listés ci-dessous :

- communiquer de manière simple et rapide pour favoriser une prise de décision efficace,
- partager le travail dans l'objectif d'une plus grande efficacité de toutes les parties prenantes,
- suivre le calendrier des tâches et l'avancement des tâches prévues pour permettre une meilleure gestion de projet,
- accéder à des espaces de travail partagés synchrones et modérés pour favoriser la cohérence continue de la solution en cours de production,
- avoir à disposition des espaces d'alertes et de prises de décisions collectives pour garantir l'implication de toutes les parties prenantes concernées.

Nous proposons d'étudier la mise en œuvre de la collaboration supportée par CAHM, et sa couverture des éléments listés ci-dessus dans les différentes activités de modélisation de l'ingénierie système. Nous pensons étudier son application, dans un premier temps, au MBSE [54] et aux processus de production et d'utilisation de modèles SysML notamment.

Le MBSE et l'ingénierie système étant très sensibles à la collaboration, les techniques collaboratives doivent également être améliorées par l'intégration d'éléments psycho-sociologiques. Nous pensons en effet que des études psycho-sociologiques sur les habitudes et les usages des parties prenantes apporteraient des éléments intéressants et peut-être surprenants [185] qui devraient permettre d'améliorer les processus et les outils à disposition de ces utilisateurs.

8.2.3 De l'intérêt d'intégrer des éléments d'apprentissage automatique dans les patrons de collaboration de CAHM au service du MBSE

Enfin, nous voyons des connexions évidentes entre ces futurs travaux (que nous comptons mettre en oeuvre dans des thèses et des stages de masters). En effet, le processus collaboratif que nous proposons peut être amélioré par l'intégration de techniques d'apprentissage automatique. Intégrer des bots par exemple à des processus collaboratifs peut s'avérer efficace pour améliorer la communication entre parties prenantes et la prise de décisions [171].

Coupler collaboration et apprentissage automatique au service de l'ingénierie système, en utilisant des techniques comme celles mises en œuvre dans [171, 169], et dans [52, 106], permettrait de garantir à la fois une meilleure qualité, une plus grande efficacité, et un support intelligent aux processus de modélisation des systèmes complexes [177, 154].

L'objectif que nous nous fixons avec ces perspectives, consiste à plus ou moins long terme à assembler les différents mécanismes que nous avons fournis et que nous comptons étendre pour obtenir un environnemt d'alignement collaboratif intelligent de modèles hétérogènes (cf. Figure 8.1).

1. `https://mindmajix.com/10-tools-for-effective-devops-collaboration`

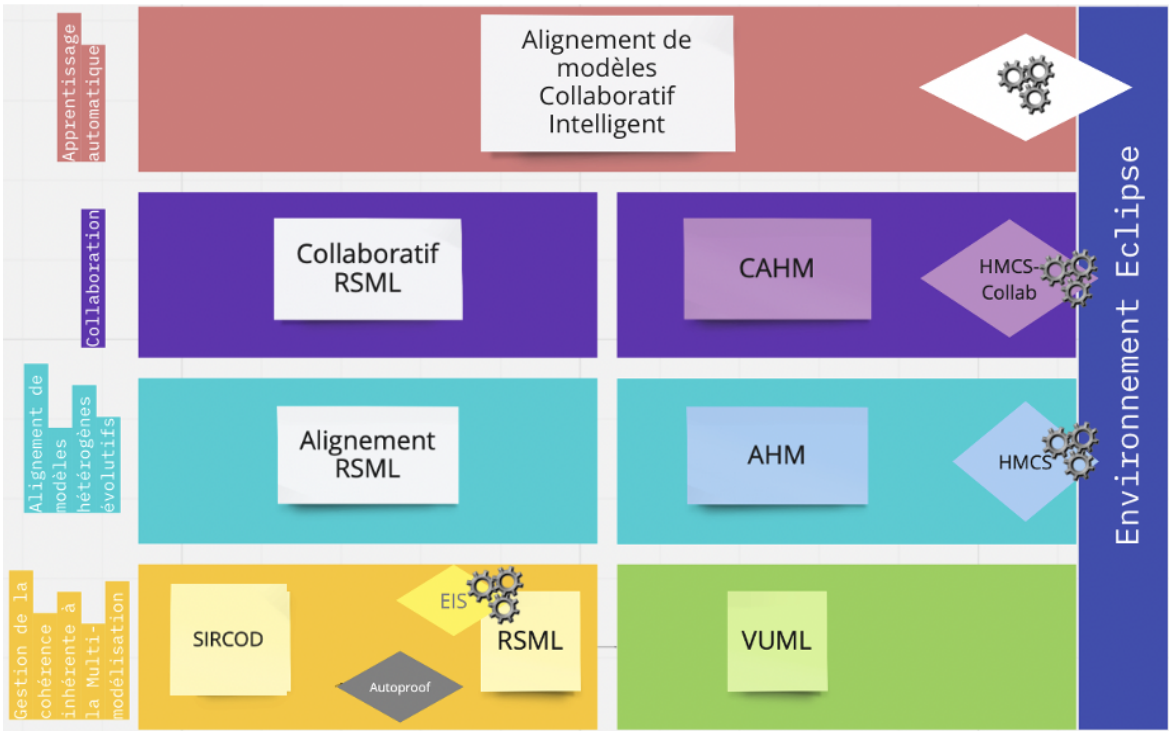


FIGURE 8.1 – Perspectives à nos contributions

Bibliographie

[1] Vetting automatically generated trace links : what information is useful to human analysts ? In *2018 IEEE 26th International Requirements Engineering Conference (RE*, page 52–63. IEEE.

[2] ISO/IEC/IEEE International Standard - Systems and software engineering – vocabulary. pages 1–418, 2010. ISO/IEC/IEEE 24765 :2010(E).

[3] ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering. pages 1–94, 2011.

[4] Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering. pages 1–104, Nov 2018. ISO/IEC/IEEE 29148 :2018(E).

[5] Mohamad , Colin Neill, and Phillip Laplante. State of practice in requirements engineering : Contemporary data. *Innovations in Systems and Software Engineering : A NASA Journal*, 12 2014.

[6] Jean-Raymond Abrial. *Modeling in Event-B : System and Software Engineering*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[7] S. Abualhaija, C. Arora, M. Sabetzadeh, L. C. Briand, and E. Vaz. A machine learning-based approach for demarcating requirements in textual specifications. In *RE*, pages 51–62. IEEE, 2019.

[8] D. Aceituna, H. Do, G. S. Walia, and S. W. Lee. Evaluating the use of model-based requirements verification method : A feasibility study. In *Workshop on Empirical Requirements Engineering (EmpiRE 2011)*, pages 13–20, 2011.

[9] Ines Ajmi. *Outils et modèles collaboratifs pour la gestion des tensions dans les services des urgences pédiatriques*. PhD thesis, Ecole centrale de Lille, 2015.

[10] Albert Albers and Christian Zingel. Challenges of model-based systems engineering : A study towards unified term understanding and the state of usage of sysml. 01 2013.

[11] Alsayed Algergawy, Daniel Faria, Alfio Ferrara, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Naouel Karam, Abderrahmane Khiat, Patrick Lambrix, Huanyu Li, Stefano Montanelli, Heiko Paulheim, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Élodie Thiéblin, Cássia Trojahn, Jana Vatascinová, Ondrej Zamazal, and Lu Zhou. Results of the ontology alignment evaluation initiative 2019. In *Proceedings of the 14th International Workshop on Ontology Matching, co-located with the 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand*, volume 2536, pages 46–85, 2019.

[12] Government Electronics American National Standards Institute and Information Technology Association. ANSI/EIA/-632, Processes for Engineering a System. 1999, 2014. Document Number ANSI/GEIA EIA-632.

[13] Daniel Amyot. Introduction to the user requirements notation : learning by example. *Computer Networks*, 42(3) :285–301, June 2003.

[14] Adil Anwar. *Formalisation par une approche IDMde la composition de modèles dans le profil VUML.* These, Université Toulouse 2, France en cotutelle avec l'Université Mohammed V de Rabat, Maroc, year=2009, month = 12, day = 09.

[15] Adil Anwar, Sophie Ebersold, Bernard Coulette, Mahmoud Nassar, Abdelaziz Kriouile, et al. A rule-driven approach for composing viewpoint-oriented models. volume 9, pages 89–114, 2010. Journal of Object Technology.

[16] C. Arora, M. Sabetzadeh, L. C. Briand, and F. Zimmer. Automated checking of conformance to requirements templates using natural language processing. *IEEE Trans. Software Eng.*, 41(10) :944–968, 2015.

[17] H.U. Asuncion, A.U. Asuncion, and R.N. Taylor. Software traceability with topic modeling. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 1, page 95–104. IEEE.

[18] J. Ayerdi, A. Garciandia, A. Arrieta, W. Afzal, E. Enoiu, A. Agirre, G. Sagardui, M. Arratibel, and O. Sellin. Towards a taxonomy for eliciting design-operation continuum requirements of cyber-physical systems. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 280–290, 2020.

[19] Elisa Baniassad and Siobhan Clarke. Theme : An approach for aspect-oriented analysis and design. In *Proceedings of the 26th International Conference on Software Engineering*, ICSE '04, page 158–167, USA, 2004. IEEE Computer Society.

[20] O. Barais. *Utilisation de la modélisation à l'exécution : objectif, challenges et bénéfices*. Habilitation à Diriger les Recherches, Université de Rennes 1, IRISA, 2014.

[21] Benoit Baudry, Benoit Combemale, Julien DeAntoni, Frédéric Mallet, Equipe AOSTE, and Sophia Antipolis. Action spécifique 2011 gemoc du gdr gpl ingénierie du logiciel pour les systèmes hétérogènes. 2012.

[22] Benoit Baudry, Franck Fleurey, Robert France, and Raghu Reddy. Exploring the relationship between model composition and model transformation. 05 2020.

[23] Valerie Belton and Jacques Pictet. A framework for group decision using a mcda model : sharing, aggregating or comparing individual information ? volume 6, pages 283–303. Taylor & Francis, 1997. Journal of decision systems.

[24] Saloua Bennani. *"Une approche IDM collaborative pour l'alignement de modèles hétérogènes*. These, Université Toulouse Jean Jaures, France et Université Mohamed V de Rabat, Maroc, 2020.

[25] Saloua Bennani, Sophie Ebersold, Bernard Coulette Bibliography, and Mahmoud Nassar. A collaborative decision approach for alignment of heterogeneous models. In *Dans : IEEE International Workshop on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 2019), Capri, Italy, 12/06/2019-14/06/2019*. IEEE Computer Society, juin 2019.

[26] Saloua Bennani, Mahmoud El Hamlaoui, Sophie Ebersold, Bernard Coulette, and Mahmoud Nassar. Collaborative model-based matching of heterogeneous models. In *International Conference on Computer Supported Cooperative Work in Design (CSCWD), Nanjing, China*. IEEExplore, 2018.

[27] Saloua Bennani, Iliass Ait El Kouch, Mahmoud El Hamlaoui, Sophie Ebersold, Bernard Coulette, and Mahmoud Nassar. A formalization of group decision making in multi-viewpoints design. volume 13, page 58. Canadian Center of Science and Education, Jan 2020. Computer and Information Science.

[28] Brian Berenbach, Florian Schneider, and Helmut Naughton. The use of a requirements modeling language for industrial applications. In *2012 20th IEEE International Requirements Engineering Conference (RE)*, pages 285–290. IEEE, September 2012.

[29] Brian Berenbach, Florian Schneider, and Helmut Naughton. The use of a requirements modeling language for industrial applications. In *2012 20th IEEE Springer Requirements Engineering Conf. (RE)*, pages 285–290. IEEE, September 2012.

[30] Jean Bézivin, Salim Bouzitouna, Marcos Didonet Del Fabro, Marie-Pierre Gervais, Frédéric Jouault, Dimitrios Kolovos, Ivan Kurtev, and Richard F Paige. A canonical scheme for model composition. In Arend Rensink and Jos Warmer, editors, *European Conference on Model Driven Architecture-Foundations and Applications*, pages 346–360. Springer, 2006.

[31] Jean Bézivin, Ivan Kurtev, et al. Model-based technology integration with the technical space concept. In *Metainformatics Symposium*, volume 20, pages 44–49, 2005.

[32] Xavier Blanc. *MDA en action : Ingénierie logicielle guidée par les modèles*. Eyrolles, 2005.

[33] Mireille Blay-Fornarino. *Interprétations de la composition d'activités*. Habilitation à diriger des recherches, Université Nice Sophia Antipolis, April 2009.

[34] Mireille Blay-Fornarino and Laurence Duchien. *Numéro spécial TSI - Ingénierie Dirigée par les Modèles*, volume 29 :4-5. TSI - Hermès Science - Lavoisier, April 2010.

[35] Frédéric Boniol and Virginie Wiels. The Landing Gear System Case Study. In Frédéric Boniol, Virginie Wiels, Yamine Ait Ameur, and Klaus-Dieter Schewe, editors, *ABZ 2014 : The Landing Gear Case Study*, number 433 in Communications in Computer and Information Science, pages 1–18. Springer International Publishing, June 2014.

[36] Francis Bordeleau, Benoit Combemale, Romina Eramo, Mark Van Den Brand, and Manuel Wimmer. Tool-Support of Socio-Technical Coordination in the Context of Heterogeneous Modeling : A Research Statement and Associated Roadmap. In *GEMOC 2018 - 6th International Workshop on The Globalization of Modeling Languages*, pages 1–3, Copenhagen, Denmark, October 2018.

[37] Artur Boronat. Moment : a formal framework for model management. 2007. PhD in Computer Science, Universitat Politenica de Valencia (UPV), Spain.

[38] Artur Boronat, Alexander Knapp, Jose Meseguer, and Martin Wirsing. What is a multi-modeling language ? volume 5486, pages 71–87, 06 2008.

[39] F. Boulanger, C. Jacquet, C. Hardebolle, and E. Rouis. Modeling heterogeneous points of view with modhel'x. In *International Conference on Model Driven Engineering Languages and Systems*, pages 310–324. Springer, 2009.

[40] Frédéric Boulanger, Christophe Jacquet, Cécile Hardebolle, and Ayman Dogui. Heterogeneous model composition in ModHel'X : The power window case study. In *Workshop on the Globalization of Modeling Languages at MODELS 2013*, 2013.

[41] M. Brambilla, J. Cabot, and M. Wimmer. *Model-driven Software Engineering in Practice*. Synthesis Lectures on Software. Morgan & Claypool, 2012.

[42] Manfred Broy, Martin Feilkas, Markus Herrmannsdoerfer, Stefano Merenda, and Daniel Ratiu. Seamless model-based development : From isolated tools to integrated model engineering environments. volume 98, pages 526–545. IEEE, 2010. Proceedings of the IEEE.

[43] Jean-Michel Bruel, Benoit Combemale, Esther Guerra, Jean-Marc Jézéquel, Jörg Kienzle, Juan de Lara, Gunter Mussbacher, Eugene Syriani, and Hans Vangheluwe. Model transformation reuse across metamodels. In Arend Rensink and Jesús Sánchez Cuadrado, editors, *Theory and Practice of Model Transformation*, pages 92–109, Cham, 2018. Springer International Publishing.

[44] Jean-Michel Bruel, Sophie Ebersold, Florian Galinier, Manuel Mazzara, Bertrand Meyer, and Alexandr Naumchev. The role of formalism in system requirements (full version). volume cs.SE/1911.02564, 2019. arXiv.

[45] J.M. Bruel and S. Gerard. *SysML in Action with Papyrus*. Elsevier Science, 2019.

[46] C. Brun and A. Pierantonio. Model differences in the eclipse modeling framework. volume 9, pages 29–34, 2008. UPGRADE, The European Journal for the Informatics Professional.

[47] Cédric Brun and Alfonso Pierantonio. Model differences in the eclipse modeling framework. volume 9, pages 29–34, 2008. UPGRADE, The European Journal for the Informatics Professional.

[48] Hugo Bruneliere, Erik Burger, Jordi Cabot, and Manuel Wimmer. A feature-based survey of model view approaches, intl. journal of SOftware & SYstems Modeling. volume 18, pages 1931–1952. Springer, 2019. Software & Systems Modeling.

[49] Hugo Bruneliere, Romina Eramo, Abel Gómez, Valentin Besnard, Jean Michel Bruel, Martin Gogolla, Andreas Kästner, and Adrian Rutle. Model-driven engineering for design-runtime interaction in complex systems : Scientific challenges and roadmap. In *Federation of International Conferences on Software Technologies : Applications and Foundations*, pages 536–543. Springer, 2018.

[50] Hugo Brunelière, Jokin Garcia Perez, Manuel Wimmer, and Jordi Cabot. Emf views : A view mechanism for integrating heterogeneous models. In *International Conference on Conceptual Modeling*, pages 317–325. Springer, 2015.

[51] Antonio Bucchiarone, Jordi Cabot, Richard F Paige, and Alfonso Pierantonio. Grand challenges in model-driven engineering : an analysis of the state of the research. volume 19, pages 5–13, 2020.

[52] Jordi Cabot, Robert Clarisó, Marco Brambilla, and Sébastien Gérard. *Cognifying Model-Driven Software Engineering*, pages 154–160. 01 2018.

[53] Laura Roa Castro. *Management de la complexité organisationnelle des projets en ingénierie systèmes : Mise en place d'une approche socio-technique pour l'amélioration des aspects collaboratifs*. PhD thesis, 2017.

[54] Mohammad Chami, Aiste Aleksandraviciene, Aurelijus Morkevicius, and Jean-Michel Bruel. Towards solving mbse adoption challenges : The d3 mbse adoption toolbox. *INCOSE International Symposium*, 28 :1463–1477, 07 2018.

[55] Mohammad Chami, Christophe Zoghbi, and Jean-Michel Bruel. *A First Step towards AI for MBSE : Generating a Part of SysML Models from Text Using AI*, pages 123–136. 11 2019.

[56] A. Cicchetti and F. Ciccozzi. Towards a novel model versioning approach based on the separation between linguistic and ontological aspects. In *ME 2013–Models and Evolution Workshop Proceedings*, page 60. CEUR-WS, 2013.

[57] Antonio Cicchetti, Federico Ciccozzi, and Alfonso Pierantonio. Multi-view approaches for software and system modelling : a systematic literature review, in intnl. journal of software & systems modeling. pages 1–27. Springer, 2019. Software & Systems Modeling.

[58] Cauê Clasen, Frédéric Jouault, and Jordi Cabot. Virtualemf : a model virtualization tool. In *International Conference on Conceptual Modeling*, pages 332–335. Springer, 2011.

[59] J.L. Colaço, B. Pagano, and Pouzet. M. : A conservative extension of synchronous dataflow with state machines. In *Proc. of the 5th ACM Int. Conference on Embedded Software*, page 173–182, New York, NY, USA.

[60] B. Combemale. *Towards Language-Oriented Modeling*. Habilitation à Diriger les Recherches, Université de Rennes 1, IRISA, 2015.

[61] Benoit Combemale, Julien Deantoni, Benoit Baudry, Robert B. France, Jean-Marc Jézéquel, and Jeff Gray. Globalizing Modeling Languages. pages 10–13, June 2014.

[62] Benoit Combemale, Robert France, Jean-Marc Jézéquel, Bernhard Rumpe, Jim R.H. Steel, and Didier Vojtisek. *Engineering Modeling Languages*. Chapman and Hall/CRC, November 2016.

[63] IEEE Computer Society Software Engineering Standards Committee and IEEE-SA Standards Board. IEEE Recommended Practice for Software Requirements Specifications. Institute of Electrical and Electronics Engineers, 1998.

[64] P. Cousot. Methods and logics for proving programs. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, chapter 15, pages 843–993. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1990.

[65] Xavier Crégut, Sophie Ebersold, Mahmoud Nassar, and Bernard Coulette. *Un patron de génération de code pour le profil VUML. Dans : Objets, Composants et Modèles (OCM*. BibTeX, Berne, Suisse.

[66] Krzysztof Czarnecki and Simon Helsen. Classification of model transformation approaches. In *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, volume 45, pages 1–17. USA, 2003.

[67] Javier Luis Cánovas Izquierdo and Jordi Cabot. Jsondiscoverer : Visualizing the schema lurking behind json documents. *Knowledge-Based Systems*, 103 :52 – 55, 2016.

[68] Thileli debiane. Mise en cohérence de modèles hétérogènes évolutifs lors d'un développement collaboratif, mémoire de master2 recherche it toulouse, 07 2014.

[69] Marcos Didonet Del Fabro, Jean Bézivin, Frédéric Jouault, Erwann Breton, and Guillaume Gueltas. Amw : a generic model weaver. In *1 ere Journées sur l'Ingénierie Dirigée par les Modèles (IDM05)*, pages 105–114, 2005.

[70] Marcos Didonet Del Fabro, Jean Bézivin, and Patrick Valduriez. Weaving models with the eclipse amw plugin. In *Eclipse Modeling Symposium, Eclipse Summit Europe*, volume 2006, pages 37–44, 2006.

[71] G. Deshpande, Q. Motger, C. Palomares, I. Kamra, K. Biesialska, X. Franch, G. Ruhe, and J. Ho. Requirements dependency extraction by integrating active learning with ontology-based retrieval. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 78–89, 2020.

[72] Davide Di Ruscio, Mirco Franzago, Ivano Malavolta, and Henry Muccini. Envisioning the future of collaborative model-driven software engineering. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 219–221. IEEE, 2017.

[73] Zoé Drey, Cyril Faucher, Franck Fleurey, Vincent Mahé, and Didier Vojtisek. Kermeta language reference manual. 2009. Manuscript available online http ://www. kermeta. org.

[74] Matthew B Dwyer, George S Avrunin, and James C Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No. 99CB37002)*, pages 411–420. IEEE, 1999.

[75] Christof Ebert and Michael Jastram. Reqif : Seamless requirements interchange format between business partners. volume 29, pages 82–87. IEEE, 2012. IEEE software.

[76] Eclipse Foundation. Eclipse/m2m project web page, 2007.

[77] Eclipse Foundation. *Papyrus*. 2015.

[78] Jeff Estefan. Survey of model-based systems engineering (mbse) methodologies. volume 25, 01 2008.

[79] Kleinner Farias and Toacy Oliveira. A guidance for model composition. page 27, 08 2007.

[80] A. Ferrari, F. Dell'Orletta, A. Esuli, V. Gervasi, and S. Gnesi. Natural language requirements processing : A 4d vision. *IEEE Softw.*, 34(6) :28–35, 2017.

[81] Anthony Finkelstein, Jeff Kramer, and Michael Goedicke. *Viewpoint oriented software development*. University of London, Imperial College of Science and Technology, Department . . . , 1991.

[82] Anthony Finkelstein, Jeff Kramer, Bashar Nuseibeh, Ludwik Finkelstein, and Michael Goedicke. Viewpoints : A framework for integrating multiple perspectives in system development. volume 2, pages 31–57. World Scientific, 1992. International Journal of Software Engineering and Knowledge Engineering.

[83] Franck Fleurey, Benoit Baudry, Robert France, and Sudipto Ghosh. A generic approach for automatic model composition. In *International Conference on Model Driven Engineering Languages and Systems*, pages 7–15. Springer, 2007.

[84] M. Fowler. *Domain-Specific Languages*. Addison-Wesley Signature Series (Fowler). Pearson Education, 2010.

[85] Robert France and Bernhard Rumpe. Model-driven development of complex software : A research roadmap. In *2007 Future of Software Engineering, ICSE*, pages 37–54. IEEE Computer Society, 2007.

[86] Mirco Franzago, Davide Di Ruscio, Ivano Malavolta, and Henry Muccini. Collaborative model-driven software engineering : a classification framework and a research map. volume 44, pages 1146–1175. IEEE, 2017. IEEE Transactions on Software Engineering.

[87] Florian Galinier. A DSL for Requirements in the Context of a Seamless Approach. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, pages 932–935, New York, NY, USA, 2018. ACM.

[88] Florian Galinier. Formal Semantics of Requirements Relationships for Traceability. Rapport de recherche 2019_04_30, IRIT, Université Paul Sabatier, Toulouse, avril 2019.

[89] Florian Galinier. *Seamless development of software-intensive systems : a multirequirements approach*. These, Université Paul Sabatier, Toulouse 3, France, 2020.

[90] Florian Galinier, Jean-Michel Bruel, Sophie Ebersold, and Bertrand Meyer. Seamless integration of multirequirements in complex systems. In *IEEE 25th International Requirements Engineering Conference Workshops, RE 2017 Workshops, Lisbon, Portugal, September 4-8, 2017*, pages 21–25. IEEE Computer Society, 2017.

[91] Erich Gamma. *Design patterns : elements of reusable object-oriented software*. Pearson Education India, 1995.

[92] G.E.M.O.C. Initiative on the globalization of modeling languages. []Online] Available : http ://gemoc.org/ins/.

[93] Martin Gogolla, Frank Hilken, and Khanh-Hoang Doan. Achieving model quality through model validation, verification and exploration. volume 54, pages 474 – 511, 2018.

[94] Fahad R Golra, Antoine Beugnard, Fabien Dagnat, Sylvain Guerin, and Christophe Guychard. Addressing Modularity for Heterogeneous Multi-model Systems Using Model Federation. In *Companion Proceedings of the 15th International Conference on Modularity*, MODULARITY Companion 2016, pages 206–211, Malaga, Spain, 2016. ACM.

[95] Fahad Rafique Golra, Fabien Dagnat, Jeanine Souquières, and Imen Sayar Sylvain Guérin. Bridging the gap between informal requirements and formal specifications using model federation ». In *: International Conference on Software Engineering and Formal Methods (SEFM). T*, page 10886. Springer International Publishing. isbn :.

[96] F.R. Golra, A. Beugnard, F. Dagnat, S. Guerin, and C. Guychard. Addressing modularity for heterogeneous multi-model systems using model federation. In *15th International Conference on Modularity,.Malaga*, page 206–211, Spain.

[97] O. C. Z. Gotel and C. W. Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of IEEE International Conference on Requirements Engineering*, pages 94–101, 1994.

[98] T. Güneş and F. B. Aydemir. Automated goal model extraction from user stories using nlp. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 382–387, 2020.

[99] Mahmoud El Hamlaoui. *Mise en correspondance et gestion de la cohérence de modèles hétérogènes évolutifs*. These, Université Toulouse Jean Jaurès en cotutelle avec l'Université Mohammed V de Rabat, 2015.

[100] Mahmoud El Hamlaoui, Saloua Bennani, Mahmoud Nassar, Sophie Ebersold, and Bernard Coulette. A MDE Approach for Heterogeneous Models Consistency. In *International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), Madeira*. SciTePress, 2018.

[101] Mahmoud El Hamlaoui, Bernard Coulette, Sophie Ebersold, Saloua Bennani, Mahmoud Nassar, Antoine Beugnard, Jean-Christophe Bach, Yassine Jamoussi, and Hanh-Nhi Tran. Alignment of viewpoint heterogeneous design models : Emergency Department Case Study. In *4th International Workshop On the Globalization of Modeling Languages (GEMOC 2016) co-located with ACM/IEEE MODELS 2016*, pages 18–27, Saint-Malo, France, 2016.

[102] Mahmoud El Hamlaoui, Sophie Ebersold, Adil Anwar, Bernard Coulette, and Mahmoud Nassar. Towards a framework for heterogeneous models matching. volume 8, pages 132–151, 2014. Journal of Software Engineering.

[103] Mahmoud El Hamlaoui, Sophie Ebersold, Bernard Coulette, Adil Anwar, and Mahmoud Nassar. Maintien de la cohérence de modèles de conception hétérogènes. volume 34, pages 667–702, 2015. Technique et Science Informatiques.

[104] Mahmoud El Hamlaoui, Sophie Ebersold, Bernard Coulette, Mahmoud Nassar, and Adil Anwar. Heterogeneous models matching for consistency management. In *IEEE International Conference on Research Challenges in Information Science - RCIS 2014*, pages pp. 1–12, Marrakech, Morocco, 2014.

[105] Mahmoud El Hamlaoui, Cassia Trojahn, Sophie Ebersold, and Bernard Coulette. Towards an ontology-based approach for heterogeneous model matching. In *International Workshop On the Globalization of Modeling Languages (GEMOC), co-located with ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS*, Valencia, SPAIN. CEUR Workshop Proceedings.

[106] T. Hartmann, A. Moawad, F. Fouquet, and Y. Le Traon. The next evolution of mde : A seamless integration of machine learning into domain modeling. In *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 180–180, 2017.

[107] R. Hebig, D. Khelladi, and R. Bendraou. Approaches to co-evolution of metamodels and models : A survey". volume 43, pages 396–414,. Institute of Electrical and Electronics Engineers.

[108] Regina Hebig, Djamel Eddine Khelladi, and Reda Bendraou. Surveying the corpus of model resolution strategies for metamodel evolution. In *2015 Asia-Pacific Software Engineering Conference (APSEC)*, pages 135–142. IEEE, Dec 2015.

[109] J. Helming, M. Koegel, F. Schneider, M. Haeger, C. Kaminski, B. Bruegge, and B. Berenbach. Towards a unified Requirements Modeling Language. In *2010 Fifth International Workshop on Requirements Engineering Visualization*, pages 53–57, September 2010.

[110] Rich Hilliard et al. Viewpoint modeling. In *Proceedings of 1st ICSE Workshop on Describing Software Architecture with UML (DSAU@ICSE)*, volume 7, Toronto (Canada), 2001. ACM.

[111] John Hutchinson, Mark Rouncefield, and Jon Whittle. Model-driven engineering practices in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, page 633–642, New York, NY, USA, 2011. Association for Computing Machinery.

[112] John Hutchinson, Jon Whittle, Mark Rouncefield, and Steinar Kristoffersen. Empirical assessment of mde in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, page 471–480, New York, NY, USA, 2011. Association for Computing Machinery.

[113] Reiner Hähnle, Kristofer Johannisson, and Aarne Ranta. An Authoring Tool for Informal and Formal Requirements Specifications. In Ralf-Detlef Kutsche and Herbert Weber, editors, *Fundamental Approaches to Software Engineering*, number 2306 in Lecture Notes in Computer Science, pages 233–248. Springer Berlin Heidelberg, April 2002. DOI : 10.1007/3-540-45923-5_16.

[114] IBM. *Rational Doors*. 2015.

[115] INCOSE. *AFIS chapter, MBSE WG, Requirement follow-up during architecture definition*.

[116] INCOSE. *SE Vision 2025*. 2014.

[117] J.L.C. Izquierdo and J. Cabot. Collaboro : a collaborative (meta) modeling tool. volume 2, page 84.

[118] Michael Jackson and Pamela Zave. Deriving specifications from requirements : an example. In *1995 17th Int. Conf. on Software Engineering*, pages 15–15. IEEE, 1995.

[119] B. Jeannet and F. Gaucher. Debugging real-time systems requirements : simulate the "what" before the "how". In *Embedded World Conf., Nürnberg*. Germany.

[120] Jean-Marc Jézéquel, Olivier Barais, and Franck Fleurey. Model driven language engineering with kermeta. In *International Summer School on Generative and Transformational Techniques in Software Engineering*, pages 201–221. Springer, 2009.

[121] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL : A model transformation tool. volume 72, pages 31–39. Elsevier, 2008. Science of computer programming.

[122] Frédéric Jouault and Ivan Kurtev. Transforming models with atl. In Jean-Michel Bruel, editor, *Satellite Events at the MoDELS 2005 Conference*, pages 128–138, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[123] Jean-Marc Jézéquel. Model driven design and aspect weaving. volume 7, 05 2008.

[124] Komlan Akpédjé Kedji, Redouane Lbath, Bernard Coulette, Mahmoud Nassar, Laurent Baresse, and Florin Racaru. Supporting collaborative development using process models : a tooled integration-focused approach. volume 26, pages 890–909. Wiley Online Library, 2014. Journal of Software : Evolution and Process.

[125] Steven Kelly. Modelling by the people, for the people. In Martina Seidl and Steffen Zschaler, editors, *Software Technologies : Applications and Foundations*, pages 178–183, Cham, 2018. Springer International Publishing.

[126] Stuart Kent. Model driven engineering. In Michael Butler, Luigia Petre, and Kaisa Sere, editors, *Integrated Formal Methods*, pages 286–298, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[127] D. Khelladi, R. Hebig, R. Bendraou, J. Robin, and M.P. Gervais. Detecting complex changes and refactorings during (meta)model evolution". volume 62, Elsevier. Information Systems.

[128] Djamel Eddine Khelladi, Reda Bendraou, Regina Hebig, and Marie-Pierre Gervais. A semi-automatic maintenance and co-evolution of OCL constraints with (meta)model evolution. volume 134, pages 242–260. Elsevier, December 2017.

[129] John Klein, Harry Levinson, and Jay Marchetti. Model-driven engineering : Automatic code generation and beyond. Technical Report CMU/SEI-2015-TN-005, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2015.

[130] D. Kolovos. Establishing correspondences between models with the epsilon comparison language. In *Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications*, page 146–157.

[131] Dimitrios S Kolovos, Richard F Paige, and Fiona AC Polack. Merging models with the epsilon merging language (eml). In *International Conference on Model Driven Engineering Languages and Systems*, pages 215–229. Springer, 2006.

[132] Henk Koning and Hans van Vliet. A method for defining ieee std 1471 viewpoints. volume 79, pages 120–131. Elsevier, 2006. Journal of Systems and Software.

[133] Lars Kotthoff, Ian P. Gent, and Ian Miguel. An evaluation of machine learning in algorithm selection for search problems. *AI Commun.*, 25(3) :257–270, August 2012.

[134] A.Van Lamsweerde. *Requirements engineering : From system goals to UML models to software*, volume 10. John Wiley Sons, Chichester, UK.

[135] Axel van Lamsweerde. Goal-oriented requirements engineering : a guided tour. In *Proc. 5th IEEE Int. Symposium on Requirements Engineering*, pages 249–262, 2001.

[136] Jean-Louis Le Moigne. *La théorie du système général : théorie de la modélisation*. FeniXX, 1994.

[137] Gary T Leavens, Albert L Baker, and Clyde Ruby. Preliminary design of jml : A behavioral interface specification language for java. volume 31, pages 1–38. ACM, 2006. ACM SIGSOFT Software Engineering Notes.

[138] Emmanuel Letier. *Reasoning about agents in goal-oriented requirements engineering*. PhD thesis, PhD thesis, Université catholique de Louvain, 2001.

[139] Feng-Lin Li, Jennifer Horkoff, Alexander Borgida, Giancarlo Guizzardi, Lin Liu, and John Mylopoulos. From Stakeholder Requirements to Formal Specifications Through Refinement. In Samuel A. Fricker and Kurt Schneider, editors, *Requirements Engineering : Foundation for Software Quality*, Lecture Notes in Computer Science, pages 164–180. Springer International Publishing, March 2015. DOI : 10.1007/978-3-319-16101-3_11.

[140] Yuehua Lin, Jeff Gray, and Frédéric Jouault. Dsmdiff : a differentiation tool for domain-specific models. volume 16, pages 349–361. Springer, 2007. European Journal of Information Systems.

[141] I. Malavolta, H. Muccini, and S. Rekha. Enhancing architecture design decisions evolution with group decision making principles. In *International Workshop on Software Engineering for Resilient Systems*, page 9–23. Springer, Cham.

[142] S. Marcaillou, B. Coulette, and A. Kriouile. Visibility : A new relationship for complex system modelling. In *Int. Conference TOOLS USA*, pages pp. 153–161, Santa Barbara (USA), 1994.

[143] Sophie Marcaillou Ebersold. *Intégration de la notion de points de vue dans la modélisation par objets : le langage VBOOL.* PhD thesis, Toulouse 3, 1995.

[144] Alistair Mavin, Philip Wilkinson, Adrian Harwood, and Mark Novak. Easy approach to requirements syntax (ears). In *2009 17th IEEE International Requirements Engineering Conference*, pages 317–322. IEEE, 2009.

[145] Marjan Mernik, Jan Heering, and Anthony M Sloane. When and how to develop domain-specific languages. volume 37, pages 316–344. ACM, 2005. ACM computing surveys (CSUR).

[146] Bertrand Meyer. Applying 'design by contract'. volume 25, pages 40–51, October 1992.

[147] Bertrand Meyer. *Eiffel : The Language*. Prentice-Hall, Inc., USA, 1992.

[148] Bertrand Meyer. Multirequirements. In Norbert Seyff and Anne Koziolek, editors, *Modelling and Quality in Requirements Engineering (Martin Glinz Festscrhift)*, 2013.

[149] Bertrand Meyer, Jean-Michel Bruel, Sophie Ebersold, Florian Galinier, and Alexandr Naumchev. The anatomy of requirements. volume abs/1906.06614, 2019.

[150] Luisa Mich, Mariangela Franch, and Pier Luigi Novi Inverardi. Market research for requirements analysis using linguistic tools. *Requirements Engineering*, 9 :40–56, 01 2004.

[151] Patrice Micouin. *Model-Based Systems Engineering : Fundamentals and Methods*. John Wiley Sons eds, 01 2014.

[152] Pieter J Mosterman and Justyna Zander. Cyber-physical systems challenges : a needs analysis for collaborating embedded software systems. volume 15, pages 5–16. Springer, 2016. Software & Systems Modeling.

[153] Pierre-Alain Muller, Franck Fleurey, and Jean-Marc Jézéquel. Weaving executability into object-oriented meta-languages. In Lionel Briand and Clay Williams, editors, *Model Driven Engineering Languages and Systems*, pages 264–278, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[154] Gunter Mussbacher, Benoit Combemale, Silvia Abrahão, Nelly Bencomo, Loli Burgueño, Gregor Engels, Jörg Kienzle, Thomas Kühn, Sébastien Mosser, Houari Sahraoui, and Martin Weyssow. Towards an assessment grid for intelligent modeling assistance. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems : Companion Proceedings*, MODELS '20, New York, NY, USA, 2020. Association for Computing Machinery.

[155] Mahmoud Nassar. VUML : a Viewpoint oriented UML Extension. In *18th IEEE International Conference on Automated Software Engineering, 2003. Proceedings.*, pages 373–376, Montreal, Canada, 2003. IEEE.

[156] Mahmoud Nassar. *Analyse/conception par points de vue : le profil VUML*. PhD thesis, 2005. Thèse de doctorat dirigée par Coulette, Bernard Informatique Toulouse, INPT 2005.

[157] Mahmoud Nassar, Jérémie Guiochet, Bernard Coulette, Xavier Crégut, and Sophie Ebersold. Vers un profil UML pour la conception de composants multivues. volume 11, pages 83–113, http ://www.editions-hermes.fr/, décembre 2005. Hermès Science.

[158] Alexandr Naumchev, Bertrand Meyer, Manuel Mazzara, Florian Galinier, Jean-Michel Bruel, and Sophie Ebersold. AutoReq : Expressing and verifying requirements for control systems. volume 51, pages 131 – 142, 2019.

[159] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, A. Pierantonio, and L. Iovino. Automated classification of metamodel repositories : A machine learning approach. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 272–282, 2019.

[160] Thuy Nguyen. Verification of behavioural requirements for complex systems with form-l, a modelica extension. In *26th International Conference on Software Systems Engineering and their Applications. EDF RD, 6 quai Watier, 78110*, Chatou, FRANCE.

[161] OMG. Software & systems process engineering meta-model specification. volume 2, 2008. OMG Std., Rev.

[162] Object Management Group (OMG). Uml2 ocl final adopted specification.

[163] Object Management Group (OMG). UML 2.0, September 2003. https ://www.omg.org/spec/UML/2.0/Beta1.

[164] Object Management Group OMG. *OMG Systems Modeling Language (OMG SysML™), V1.0*. 2007. OMG Document Number : formal/2007-09-01 Standard document URL : http ://www.omg.org/spec/SysML/1.0/PDF.

[165] Harold Ossher, Matthew Kaplan, Alexander Katz, William Harrison, and Vincent Kruskal. Specifying subject-oriented composition. volume 2, pages 179–202, 1996.

[166] W.C. OWL. Web ontology language structural specification and functional-style syntax.

[167] R. Paige and J. Ostroff. The Single Model Principle. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, RE '01, pages 292–, Washington, DC, USA, 2001. IEEE Computer Society.

[168] F.S. Parreiras, S. Staab, and A. Winter. *TwoUse : Integrating UML models and OWL ontologies*. Inst. für Informatik.

[169] Tanumoy Pati, Dennis Feiock, and James Hill. Proactive modeling : Auto-generating models from their semantics and constraints. In *SPLASH 2012 : DSM 2012 - Proceedings of the 2012 ACM Workshop on Domain-Specific Modeling*, pages 7–12, 10 2012.

[170] L. Patil and M. Atique. A novel feature selection based on information gain using wordnet. In *Science and Information Conference (SAI), 2013, 2013*, page 625–629, London (UK. IEEE.

[171] Sara Perez, Esther Guerra, Juan Lara, and Francisco Jurado. The rise of the (modelling) bots : Towards assisted modelling via social networks. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 723–728, 10 2017.

[172] A. L. Ramos, J. V. Ferreira, and J. Barceló. Model-based systems engineering : An emerging approach for modern systems. volume 42, pages 101–111, 2012. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).

[173] P. Raymond, Y. Roux, and E. Jahier. Specifying and executing reactive scenarios with lutin. In *Electronic Notes in Theoretical Computer Science*, volume 203, page 19–34.

[174] R. Navateja Reddy, Robert B. France, Sudipto Ghosh, and Franck Fleurey. Model composition-a signature-based approach. 2005.

[175] Y. R. Reddy, S. Ghosh, R. B. France, G. Straw, J. M. Bieman, N. McEachen, E. Song, and G. Georg. Directives for composing aspect-oriented design class models. In Awais Rashid and Mehmet Aksit, editors, *Transactions on Aspect-Oriented Software Development I*, pages 75–105, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[176] Jean-François Roy, Jason Kealey, and Daniel Amyot. Towards Integrated Tool Support for the User Requirements Notation. In Reinhard Gotzhein and Rick Reed, editors, *System Analysis and Modeling : Language Profiles : 5th International Workshop, SAM 2006, Kaiserslautern, Germany, May 31 - June 2, 2006, Revised Selected Papers*, pages 198–215, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. DOI : 10.1007/11951148_13.

[177] Rijul Saini, Gunter Mussbacher, Jin L. C. Guo, and Jörg Kienzle. Domobot : A bot for automated and interactive domain modelling. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems : Companion Proceedings*, MODELS '20, New York, NY, USA, 2020. Association for Computing Machinery.

[178] Omer Salih and Abd-El-Kader Sahraoui. From requirements engineering to uml using natural language processing – survey study. *European Journal of Engineering Research and Science*, 2 :44, 01 2017.

[179] D.C. Schmidt. Model-driven engineering. In *IEEE Computer*, volume 39, pages 25–31, 2006.

[180] Pavel Shvaiko and Jérôme Euzenat. Ontology Matching : State of the Art and Future Challenges. volume 25, pages 158–176, 2013.

[181] Richard Soley et al. Model driven architecture. volume 308, page 5, 2000. OMG white paper.

[182] Sparx Systems. *Enterprise Architect*. 2017.

[183] Dassault Systems. CATIA Reqtify, 2016.

[184] D. Torre, S. Abualhaija, M. Sabetzadeh, L. Briand, K. Baetens, P. Goes, and S. Forastier. An ai-assisted approach for checking the completeness of privacy policies against gdpr. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 136–146, 2020.

[185] Nirnaya Tripathi, Eriks Klotins, Rafael Prikladnicki, Markku Oivo, Leandro Bento Pompermaier, Arun Sojan Kudakacheril, Michael Unterkalmsteiner, Kari Liukkunen, and Tony Gorschek. An anatomy of requirements engineering in software startups using multi-vocal literature and case survey. *Journal of Systems and Software*, 146 :130 – 151, 2018.

[186] Cassia Trojahn dos Santos. *Towards ontology matching maturity : contributions to complex, holistic and foundational ontology matching*. Habilitation à diriger les recherches, Université de Toulouse, Toulouse, France, 2019.

[187] Julian Tschannen, Carlo A. Furia, Martin Nordio, and Nadia Polikarpova. AutoProof : Auto-Active Functional Verification of Object-Oriented Programs. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *Lecture Notes in Computer Science*, pages 566–580. Springer-Verlag, Berlin, Heidelberg, April 2015.

[188] Axel Van Lamsweerde. *Requirements engineering : From system goals to UML models to software*, volume 10. Chichester, UK : John Wiley & Sons, 2009.

[189] K. Vanherpen, J. Denil, I. Dávid, De Meulenaere, Mosterman P., Torngren. P., M., and A. Qamar. Ontological reasoning for consistency in the design of cyber-physical systems. In *2016 1st International Workshop on Cyber-Physical Production Systems (CPPS*, page 1–8. Vienna.

[190] K. Voigt, P. Ivanov, and A. Rummler. Matchbox : combined meta-model matching for semi-automatic mapping generation. In *ACM Symposium on Applied Computing (SAC), 2010*, page 2281–2288, New York (USA. ACM.

[191] Wang W., N. Niu, H. Liu, and Z. Niu. Enhancing automated requirements traceability by resolving polysemy. In *2018 IEEE 26th International Requirements Engineering Conference (RE*, page 40–51. IEEE.

[192] J. Whittle, J. Hutchinson, and M. Rouncefield. The state of practice in model-driven engineering. volume 31, pages 79–85, 2014. IEEE Software.

[193] J. Whittle, P. Sawyer, N. Bencomo, B.H.C. Cheng, and J.-M. Bruel. Relax : Incorporating uncertainty into the specification of self-adaptive systems. In *2009 17th IEEE International Requirements Engineering Conference*, page 79–88.

[194] Matt Wynne, Aslak Hellesoy, and Steve Tooke. *The cucumber book : behaviour-driven development for testers and developers*. Pragmatic Bookshelf, 2017.

[195] Guang Yang, Alberto Sangiovanni-Vincentelli, Yosinori Watanabe, and Felice Balarin. Separation of concerns : overhead in modeling and efficient simulation techniques. In *Proceedings of the 4th ACM international conference on Embedded software*, pages 44–53. ACM, 2004.

[196] Abdullah Yousafzai, Victor Chang, Abdullah Gani, and Rafidah Md Noor. Multimedia augmented m-learning : Issues, trends and open challenges. volume 36, pages 784–792. Elsevier, 2016. International Journal of Information Management.

[197] Eric SK Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997, Proc. of the 3rd IEEE Int. Symposium on*, pages 226–235. IEEE, 1997.

[198] Gregory Zacharewicz. Model-based approaches for interoperability of next generation organization information systems : State of the art and future challenges. In Mohammad S. Obaidat, Tuncer I. Ören, and Helena Szczerbicka, editors, *Proceedings of the 9th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH 2019, Prague, Czech Republic, July 29-31, 2019*, page 7. SciTePress, 2019.

[199] Pamela Zave and Michael Jackson. Four dark corners of requirements engineering. volume 6, pages 1–30, New York, NY, USA, January 1997. ACM.

[200] Alanna Zito, Zinovy Diskin, and Juergen Dingel. Package merge in uml 2 : Practice vs. theory ? In *International Conference on Model Driven Engineering Languages and Systems*, pages 185–199. Springer, 2006.

Table des figures

Liste des tableaux

Résumé

A l'heure actuelle les systèmes complexes de par leur caractère critique, leur nombre, leur taille, ou encore les tâches distribuées et réparties qu'ils supportent, restent un sujet d'études très ouvert. Beaucoup de progrès ont été faits dans le domaine de la recherche dans le but de rendre les systèmes complexes fiables, que ce soit dans un cadre de formalisation des exigences, de multi-modélisation et d'inter-opérabilité, de définition de langages dédiés et d'environnements d'ingénierie adaptés, d'automatisation de la génération de code, ou encore de vérification et de validation des modèles produits. Néanmoins, les méthodes, techniques, outils apportés, s'ils donnent satisfaction, nécessitent encore des améliorations. En effet l'industrie est toujours en demande de moyens permettant de gagner en efficacité tant dans la production des éléments logiciels que dans l'automatisation des vérifications. Les méthodes traditionnelles perdurent et les habitudes évoluent lentement. Les nouvelles technologies du Génie Logiciel, et en particulier de l'Ingénierie Dirigée par les Modèles, ont du mal à s'implanter dans le domaine des systèmes complexes.

Mes travaux de recherche visent à fournir des éléments conceptuels permettant de mieux maîtriser le développement de ces systèmes. Ils s'appuient sur les modèles qui sont produits lors du développement de systèmes, les replacent dans un cadre d'Ingénierie Dirigée par les Modèles, en centrant la démarche sur les parties prenantes de ces systèmes.

A travers la synthèse de quatre thèses qui se sont enchaînées, chacune répondant à des points laissés en suspens par la précédente, je montre dans ce manuscrit comment les mécanismes que nous avons proposés s'inscrivent dans une amélioration de l'Ingénierie Système, en s'appuyant sur l'Ingénierie Dirigée par les Modèles pour réaliser l'alignement de modèles hétérogènes, en mettant l'utilisateur au centre des préoccupations et en proposant des techniques collaboratives et un langage semi-formel d'expression des exigences.