



HAL
open science

Approche orientée service pour la configuration de méthodes outillées

Adrian Iacovelli

► **To cite this version:**

Adrian Iacovelli. Approche orientée service pour la configuration de méthodes outillées. Autre [cs.OH]. Université Panthéon-Sorbonne - Paris I, 2012. Français. NNT: . tel-00764048

HAL Id: tel-00764048

<https://theses.hal.science/tel-00764048>

Submitted on 12 Dec 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE DE DOCTORAT
DE L'UNIVERSITE DE PARIS 1-PANTHEON-SORBONNE

Spécialité : Informatique

Adrian Iacovelli

Pour l'obtention du titre de :

DOCTEUR DE L'UNIVERSITE PARIS I – PANTHEON - SORBONNE

**Approche orientée service pour
la configuration de méthodes outillées**

Soutenu le 13 septembre 2012 devant le jury composé de :

| | |
|---------------------|--------------------|
| Mme Carine SOUVEYET | Directeur de thèse |
| Mme Corine CAUVET | Rapporteur |
| Mme Jolita RALYTÉ | Rapporteur |
| M. Oscar PASTOR | Membre du jury |
| Mme Colette ROLLAND | Membre du jury |

Remerciements

Je tiens à remercier en premier lieu, Carine Souveyet, professeur à l'Université Paris 1 pour avoir accepté la direction scientifique de cette thèse, m'avoir épaulé tout au long de la réalisation de ce travail de recherche et prodigué de précieux conseils qui m'ont permis d'aller constamment de l'avant et de tirer un grand enseignement de cette période passée à l'Université Paris 1.

Je remercie chaleureusement Colette Rolland, professeur à l'Université Paris 1 pour m'avoir accueilli au Centre de Recherche en Informatique, laboratoire de recherche de l'Université Paris 1 et m'avoir encouragé et soutenu dans la réalisation de ce travail.

Je remercie sincèrement Corine Cauvet, professeur à l'Université d'Aix-Marseille et Jolita Ralyté maître de recherche et d'enseignement à l'Université de Genève pour m'avoir fait l'honneur d'accepter le rôle de rapporteur de cette thèse.

Je remercie également Oscar Pastor, professeur à l'Université de Valence et Colette Rolland, professeur à l'Université Paris 1 pour avoir bien voulu faire partie du jury de cette thèse.

Je remercie aussi Rebecca Deneckère pour s'être attelée à la relecture de cette thèse, pour son dynamisme et ses conseils qui m'ont été d'un soutien quotidien dans l'élaboration de ce travail.

Mes remerciements vont aussi à toute l'équipe du Centre de Recherche en Informatique pour la chaleureuse ambiance qui règne dans ce laboratoire, notamment Camille, Charlotte, Daniel, Françoise, Irina, Manuele, Assia, Bruno, Elena, Hicham, Kahina, Ramzi, Raul, Sana et Salma pour votre enthousiasme et votre soutien quotidien.

Enfin, je remercie toute ma famille et mes amis pour leurs encouragements et leur patience. Une pensée particulière pour Charley, Christelle, Benoît, Bérangère, Damien, Josselyn, Florent, Nicolas, Marianne, Mathieu, Mickaël, Patrick, Sébastien et Solenn pour tous les moments partagés au cours de ma vie estudiantine et pour m'avoir toujours soutenu dans mes projets.

Résumé

L'ingénierie des méthodes situationnelles (IMS) s'intéresse aux méthodes de développement des systèmes d'information. Ce domaine, qui a vu le jour il y a une vingtaine d'années, propose une vision modulaire des méthodes en blocs de construction élémentaire appelés composants de méthode. Il traite de la réutilisation de ces composants de méthode pour assembler ou adapter des méthodes conformément à leur situation d'utilisation. En dépit d'un effort de recherche significatif dans cette discipline, on observe que les solutions développées ces vingt dernières années n'ont pas été intégrées au monde professionnel à grande échelle.

Nous proposons l'approche "Méthodes avec une architecture orientée Service" (MaaS) pour répondre au manque de considération de la dimension usage dans les approches d'IMS. Dans cette approche, nous proposons d'appliquer le paradigme orienté service au domaine de l'ingénierie des méthodes pour outiller les méthodes situationnelles et favoriser leur usage. Notre proposition s'articule autour des trois acteurs de l'architecture orientée service : l'annuaire de service, le fournisseur et le client. La première proposition de cette approche est une ontologie des métamodèles de composants de méthode (OMCM) apportant une base sémantique commune au domaine de l'IMS. Cette ontologie permet de décrire de la même façon les composants de méthode issus d'approches différentes et de réaliser un annuaire unique pour toutes les approches IMS. Du point de vue du fournisseur, le métamodèle de service méthodologique atomique et le métamodèle de ligne de méthode constituent notre deuxième proposition. Ces deux métamodèles permettent, d'une part, d'apporter une dimension outils aux composants et aux méthodes situationnelles et, d'autre part, d'intégrer le concept de variabilité dans la définition des méthodes afin de les adapter par configuration. L'orientation service de ces outils apporte une solution interopérable entre les composants de méthode issus d'approches d'IMS différentes. Au niveau du client, notre troisième proposition consiste en une adaptation de la démarche de qualité Capability Maturity Model Integration (CMMI) aux approches d'IMS pour leur intégration progressive en entreprise, et en une plateforme d'exécution des services méthodologiques atomiques (SMA) et des lignes de méthode (LM) indispensable à leurs usages en entreprise.

Cette approche est illustrée par un cas application d'une ligne de méthode pour la phase de lancement d'un projet agile de développement d'un système d'information. Ce cas d'application capture les variantes dans les activités de lancement de projet de trois méthodes agiles (XP, DSDM et Scrum) en une ligne de méthode et permet ainsi de choisir la bonne pratique de la méthode agile la plus adaptée au contexte de projet.

Cette thèse propose d'accentuer l'usage des approches IMS par la prise en compte de la dimension outil et de la variabilité dans les méthodes.

Mots clefs : ingénierie des méthodes situationnelles, architecture orienté service, service méthodologique, service méthodologique atomique, ligne de méthode.

Abstract

Situational Method Engineering (SME) is a prolific research domain where competing SME approaches have been defined and used for composing, adapting or/and configuring a method into modular constructs according to their own modularization vision since the late 90's. However, the diversity of these solutions implies some drawbacks constituting an obstacle to their usage in industry.

The Method as a Service (MaaS) approach is proposed in this thesis to overcome this lack of consideration of situational methods usage dimension. In this approach we propose to rethink the SME domain with the service oriented paradigm according to the three actors of the service oriented architecture (service registry, provider and consumer). This solution includes four elements: the ontology of method components metamodels (OMCM), the atomic method service (AMS) metamodel, the method line (ML) metamodel and the MaaS Platform.

The OMCM ontology is providing a semantic common ground for the SME approaches. This ontology is used to describe all method components in the same manner to publish them in a registry.

The AMS and ML metamodels provide an interoperable service oriented solution for the tool support dimension of method components and methods. The ML metamodel incorporates the variability concept in the method definitions to facilitate their configurations and their adaptations.

The MaaS Platform architecture defines a frame for building method service execution platforms in different technologies like web portals or IDE.

This thesis proposes to handle the tool support dimension and the variability concept in methods by providing an original service oriented reuse platform.

Keywords: situational method engineering, service oriented architecture, method service, atomic method service, method line.

Table des matières

| | |
|---|-----------|
| Chapitre 1. Introduction | 1 |
| 1.1. Domaine de la thèse | 1 |
| 1.1.1 L'ingénierie des méthodes | 1 |
| 1.1.2 L'ingénierie des méthodes situationnelles | 2 |
| 1.2. Constat | 5 |
| 1.3. Problématique | 7 |
| 1.4. Méthode de résolution et propositions | 9 |
| 1.5. Plan de la thèse..... | 10 |
| Chapitre 2. Etat de l'art | 12 |
| 2.1. Introduction..... | 12 |
| 2.2. Cadre de référence | 13 |
| 2.2.1 L'axe objectif | 14 |
| 2.2.2 L'axe usage | 15 |
| 2.2.3 L'axe processus..... | 16 |
| 2.2.4 L'axe sujet..... | 17 |
| 2.3. Les approches d'ingénierie des méthodes situationnelles..... | 18 |
| 2.3.1 Les fragments de méthode | 18 |
| 2.3.2 Les chunks | 22 |
| 2.3.3 Les composants de méthode | 25 |
| 2.3.4 Les composants de méthode OPF (Open Process Framework) | 28 |
| 2.3.5 Les services méthodologiques Service Oriented Meta-Method (SO2M) | 31 |
| 2.4. Comparaison des approches..... | 34 |
| 2.5. Conclusion | 38 |
| Chapitre 3. L'approche MaaS..... | 40 |
| 3.1. Introduction..... | 40 |
| 3.2. Rappel du constat, de la problématique et de l'objectif de recherche | 40 |
| 3.2.1 Constat | 40 |

| | | |
|--|--|------------|
| 3.2.2 | Problématique | 42 |
| 3.2.3 | Objectif de recherche | 42 |
| 3.3. | Aperçu de la solution | 43 |
| 3.3.1 | Une ontologie de métamodèles de composant de méthode..... | 43 |
| 3.3.2 | La définition d'un composant de méthode standard et de méthodes configurables | 45 |
| 3.3.3 | Le processus de l'approche MaaS | 47 |
| 3.4. | Positionnement de l'approche par rapport au cadre de référence | 49 |
| 3.5. | Enquête auprès de professionnels | 51 |
| 3.5.1 | Critères d'évaluation de l'hypothèse H1 | 52 |
| 3.5.2 | Critères d'évaluation de l'hypothèse H2 | 52 |
| 3.5.3 | Résultats..... | 53 |
| 3.6. | Conclusions..... | 55 |
| Chapitre 4. Ontologie de métamodèles de composant de méthode..... | | 57 |
| 4.1. | Introduction..... | 57 |
| 4.2. | L'ontologie OMCM..... | 58 |
| 4.2.1 | Les ontologies | 58 |
| 4.2.2 | Description des concepts de l'ontologie OMCM | 60 |
| 4.2.3 | Correspondance entre l'ontologie OMCM et les métamodèles de composant de méthode | 66 |
| 4.2.4 | Synthèse | 77 |
| 4.3. | Utilisation de l'ontologie OMCM dans une architecture orientée service | 77 |
| 4.3.1 | Les architectures orientées services (AOS)..... | 78 |
| 4.3.2 | YASA : une technique de recherche de service sémantique | 80 |
| 4.3.3 | Publication et recherche dans l'annuaire de service méthodologique | 88 |
| 4.3.4 | Exemple de recherche de service méthodologique | 98 |
| 4.4. | Conclusion | 103 |
| Chapitre 5. Métamodèles de service méthodologique..... | | 105 |
| 5.1. | Introduction..... | 105 |
| 5.2. | Support outillé pour les composants de méthode..... | 106 |

| | | |
|--|---|------------|
| 5.2.1 | Verrous technologiques liés à l'architecture orientée service d'une méthode | 106 |
| 5.2.2 | Le métamodèle de service méthodologique atomique | 119 |
| 5.3. | Les lignes de méthode..... | 123 |
| 5.3.1 | La variabilité des processus | 124 |
| 5.3.2 | Le métamodèle de ligne de méthode..... | 129 |
| 5.3.3 | Construction d'une ligne de méthode | 136 |
| 5.3.4 | Spécification des guides de configuration..... | 159 |
| 5.3.5 | Spécification des aspects produits de la ligne de méthode..... | 166 |
| 5.4. | Un processus d'IMS progressif | 179 |
| 5.5. | Conclusion | 182 |
| Chapitre 6. Cas d'application | | 185 |
| 6.1. | Introduction..... | 185 |
| 6.2. | Cas d'application : une ligne de méthode d'un lancement de projet agile..... | 185 |
| 6.2.1 | Les méthodes agiles : étape de lancement d'un projet..... | 186 |
| 6.2.2 | Dynamic System Development Method | 186 |
| 6.2.3 | Extreme Programming | 188 |
| 6.2.4 | Scrum | 190 |
| 6.3. | Modèle d'acquisition de la ligne de méthode d'un lancement de projet agile | 192 |
| 6.3.1 | Modélisation BPMN de la ligne de méthode de lancement de projet agile | 192 |
| 6.3.2 | Guides de configuration pour la ligne de méthode de lancement de projet agile | 196 |
| 6.4. | Détail des composants conceptuels et techniques de la ligne de méthode..... | 205 |
| 6.4.1 | Composant C1 : Etude de marché..... | 205 |
| 6.4.2 | Composant C2 : Etude technique..... | 209 |
| 6.4.3 | Composant C3 : Abandon du projet..... | 213 |
| 6.4.4 | Composant C4 : Définition de la liste des besoins du produit | 216 |
| 6.4.5 | Composant C5 : Définition de besoin utilisateur | 219 |
| 6.4.6 | Composant C6 : Définition des besoins de haut niveau..... | 222 |
| 6.4.7 | Composant C7 : Planification générale des livrables..... | 225 |

| | | |
|--|--|------------|
| 6.4.8 | Composant C8 : Définition du plan général des livrables..... | 228 |
| 6.4.9 | Composant C9 : Définition du plan directeur | 231 |
| 6.4.10 | Composant C10 : Affinement des besoins | 234 |
| 6.4.11 | Composant C11 : Définition du planning de développement | 236 |
| 6.4.12 | Modèle de produit global de la ligne de méthode | 239 |
| 6.5. | Modèle de la ligne de méthode d'un lancement de projet agile | 243 |
| 6.6. | Modèle de transformation des entrées/sorties | 247 |
| 6.7. | Conclusion | 251 |
| Chapitre 7. Plateforme MaaS | | 252 |
| 7.1. | Introduction..... | 252 |
| 7.2. | Une plateforme pour l'approche MaaS..... | 253 |
| 7.2.1 | Spécifications de la plateforme de recherche et d'exécution de services méthodologiques | 253 |
| 7.2.2 | Recherche des services méthodologiques | 256 |
| 7.2.3 | Exécution d'un SMA | 257 |
| 7.2.4 | Exécution d'une ligne de méthode..... | 258 |
| 7.2.5 | Caractéristiques de l'architecture | 260 |
| 7.3. | Réalisation d'un prototype avec les technologies des portails web et WRSP | 261 |
| 7.4. | Réalisation d'un prototype avec la technologie OSGI pour son utilisation dans un environnement de développement intégré | 265 |
| 7.5. | Conclusion | 271 |
| Chapitre 8. Conclusion | | 272 |
| 8.1. | Contributions..... | 272 |
| 8.2. | Perspectives..... | 274 |
| Bibliographie | | 276 |

Chapitre 1. Introduction

1.1. Domaine de la thèse

Les travaux présentés dans cette thèse participent au domaine de l'ingénierie des méthodes et plus précisément à l'ingénierie des méthodes situationnelles. Ce domaine traite de l'ensemble des techniques de construction des méthodes. L'ingénierie des méthodes situationnelles s'intéresse aux principes de réutilisation et de modularité pour fournir une construction de méthode par réutilisation de composants issus d'autres méthodes et ainsi produire une méthode adaptée à sa situation d'utilisation.

1.1.1 L'ingénierie des méthodes

Le domaine de l'ingénierie des méthodes est apparu dans le milieu des années 90 en réponse à une demande constante d'adaptation des méthodes de développement de système d'information. En effet, les entreprises se reposent de plus en plus sur leur Système d'Information (SI) pour garantir leur compétitivité. Il en résulte une application des SI dans des domaines de plus en plus variés. En conséquence, les attentes vis-à-vis des SI sont croissantes afin de concevoir des SI en adéquation avec le cœur de métier des entreprises. La complexité des SI augmente donc de manière drastique, imposant des développements de plus en plus longs, complexes et coûteux. Dans ce contexte, il devient nécessaire d'appliquer des méthodes pour l'ingénierie des SI (Rolland, 2005) et (Ralyté, 2001a).

D'après Seligmann (Seligmann, 1989) et Rolland (Plihon, 1995), une méthode peut être décrite selon cinq aspects interdépendants : l'*aspect paradigme* ("way of thinking"), l'*aspect produit* ("way of modeling"), l'*aspect processus* ("way of working"), l'*aspect organisationnel* ("way of controlling") et l'*aspect outil* ("way of supporting"). La manière de penser d'une méthode, ou aspect paradigme, est définie au travers de paradigmes décrivant la philosophie, la "vision du monde" préconisée par la méthode et mis en place à l'aide de concepts et leurs interconnexions. L'aspect produit définit l'ensemble des modèles des éléments à produire de la méthode, chaque modèle reposant sur un paradigme bien défini. La manière d'appliquer la méthode est dictée par son aspect processus. Cet aspect regroupe l'ensemble des activités et leur agencement permettant de réaliser les éléments de la méthode à produire. L'aspect organisationnel expose la manière de gérer l'application de la méthode pour le développement du SI, elle regroupe la façon de gérer les différentes activités de la méthode dans l'organisation et la répartition des ressources pour la réalisation des tâches à effectuer. L'aspect outil a pour objectif d'apporter des outils nécessaires au support de l'application des méthodes lors du développement du SI.

Il existe plusieurs définitions du concept de méthode en accord avec ces cinq aspects. La plupart d'entre elles convergent vers l'idée qu'une méthode est basée sur des modèles (système de concepts) et consiste en plusieurs étapes qui doivent/peuvent être exécutées dans un ordre donné (Ralyté, 2001a). Une méthode de développement d'un SI est, selon Harmsen (Harmsen, 1997), *"une collection de procédures, de techniques, de descriptions de produit et d'outils pour le support effectif, efficace et consistant du processus d'ingénierie d'un SI"*.

Le domaine de l'ingénierie des méthodes rassemble toutes les techniques de définition de nouvelles méthodes d'ingénierie des SI. Plusieurs définitions de ce domaine existent. Les plus restrictives proposent de construire de nouvelles méthodes à partir de méthodes existantes (Punter, 1996). Les plus génériques définissent l'ingénierie des méthodes comme (Brinkkemper, 1996) : *"une discipline de conceptualisation, de construction et d'adaptation de méthodes, de techniques et d'outils pour le développement des systèmes d'information"*.

L'utilisation des méthodes d'ingénierie des SI dans les entreprises est néanmoins influencée par le contexte des projets auxquels elles sont appliquées. L'expérience a montré que les méthodes ne sont pas universelles. Elles nécessitent d'être adaptées aux facteurs de contingence spécifiques à chaque projet. Ce constat a amené, au sein de l'ingénierie des méthodes, à la définition de l'ingénierie des méthodes situationnelles comme un nouvel axe de recherche.

1.1.2 L'ingénierie des méthodes situationnelles

D'après (Lyytinen, 1987), (Wijers, 1990), (Aaen, 1992), (Yourdon, 1992) et (Russo, 1995), l'analyse de la pratique des méthodes montre qu'elles ne sont pas appliquées telles qu'elles sont définies. En effet, certaines étapes sont ajoutées, d'autres supprimées, les documents attendus sont adaptés au contexte du projet, les décisions ne sont pas toujours prises au moment prévu etc. D'une manière générale, on constate que les méthodes sont trop rigides et, par la même ne sont donc pas universelles.

Les différentes approches d'ingénierie des méthodes tendent à résoudre ces problèmes en définissant l'ensemble des activités visant à la construction de méthodes adaptées aux spécificités des projets d'ingénierie des SI. Dans (Welke, 1992), l'ingénierie des méthodes situationnelles est définie comme : *"la discipline visant à construire et à adapter une méthode de développement de SI et les outils associés à chacun des projets spécifiques auxquels elle est appliquée"*.

La figure 1.1 illustre le processus de construction et d'adaptation de méthodes de l'ingénierie des méthodes situationnelles qui est commun à la plupart des approches existantes.

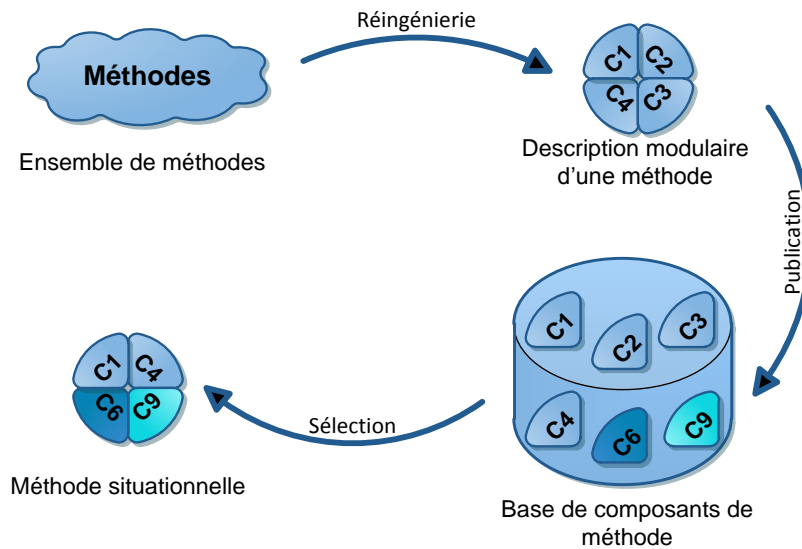


Figure 1.1 : Processus d'ingénierie des méthodes situationnelles

La figure 1.1 montre les différentes activités intervenant dans l'élaboration d'une nouvelle méthode correspondant aux besoins spécifiques d'un projet. Le point initial du processus d'ingénierie d'une méthode situationnelle est la définition des méthodes existantes. Ces méthodes sont préalablement étudiées au cours d'une étape de réingénierie visant à les découper en composants de méthode. Ces derniers sont formalisés conformément au métamodèle de composant préconisé par l'approche situationnelle. Au cours de cette étape de formalisation, les méthodes sont découpées en "blocs" autonomes. Pour chacun de ces blocs les différents aspects de la méthode (Seligmann, 1989) sont identifiés et mis en correspondance avec la structure du composant de méthode de l'approche utilisée. Une fois le découpage d'une méthode achevé, les composants résultants sont stockés dans une base de composants de méthode. Lorsqu'une méthode doit être construite conformément aux spécificités d'un projet, les composants les plus adaptés à la situation sont extraits de la base de composants et assemblés pour constituer une nouvelle méthode situationnelle.

Cette notion de composant de méthode est transverse à la majorité des approches d'ingénierie des méthodes situationnelles. Leur structure est dépendante de l'approche dans laquelle ils sont utilisés. En d'autres termes, il existe autant de types de composant de méthodes que d'approches d'ingénierie des méthodes situationnelles. Historiquement, cette notion permettant la définition modulaire d'une méthode, est apparue dans l'approche de Brinkkemper (Brinkkemper, 1996) sous le terme de fragments de méthode et est définie comme : *"une partie cohérente d'une méthode d'ingénierie des systèmes d'information"*. Les composants de méthode peuvent être définis à différents niveaux de granularité. Le plus haut niveau de granularité est porté par le composant représentant la méthode elle-même. Le plus bas niveau de granularité est un composant atomique correspondant à la plus petite partie cohérente de la méthode. Les niveaux de granularité intermédiaires de la méthode sont définis

par des composants agrégats correspondant à un assemblage de composants atomiques ou de composants agrégats de granularité inférieure.

La plupart des composants de méthode sont associés à un descripteur afin de favoriser leur réutilisation. Le descripteur contient les informations nécessaires à la récupération du composant dans une base de composants. Il comporte également les informations sur la situation et le contexte de réutilisation du composant pour son application pratique.

Le corps d'un composant de méthode doit, quant à lui, être conforme aux cinq aspects d'une méthode. En fonction des approches, la couverture de ces aspects est plus ou moins importante pour les composants qu'elles proposent. Néanmoins deux aspects, l'aspect produit et l'aspect processus, sont la base de la méthode et sont prises en compte dans toutes les approches. Au niveau de la méthode et des composants, ces deux aspects sont respectivement dérivés en modèles de processus et modèles de produit. L'utilisation de l'aspect paradigme est souvent implicite et correspond aux paradigmes de modélisation utilisés dans la définition de ces deux types de modèles.

Par définition, le modèle de processus du corps d'un composant de méthode modélise la dynamique de la méthode (Ralyté, 2001a). Un processus est une démarche mise en œuvre par un ensemble d'activités inter-reliées dont le but commun est la définition d'un produit. Le produit étant le résultat de l'application du processus, sa qualité dépend fortement du processus appliqué pour l'obtenir. En conséquence, un modèle de processus est fortement lié au modèle de ou des produits qu'il est destiné à réaliser (Ralyté, 2001a).

Les caractéristiques du produit résultant de l'application d'une méthode sont décrites dans un modèle de produit. Il est possible d'avoir plusieurs modèles de produit pour en représenter ses différentes facettes, de même pour représenter ses différents niveaux de détails (granularité) ou encore pour modéliser ses différents niveaux d'abstractions (Ralyté, 2001a).

Les diverses approches proposées dans la littérature incluent donc toutes à minima dans leurs composants : le modèle de produit et le modèle de processus. Néanmoins, elles prennent en charge de manière très inégale les autres aspects d'une méthode (l'aspect organisationnel et l'aspect outil).

Dans cette thèse, nous nous limiterons à l'étude des cinq approches les plus importantes du domaine. L'approche la plus simple et également la première approche à avoir vu le jour est celle de Brinkkemper (Brinkkemper, 1998). Elle traite de composants de méthode appelés fragments de méthode inter-reliés décrivant soit une partie produit, soit une partie processus de la méthode. Une autre approche est celle développée dans (Agerfalk, 2003) et (Wistrand, 2004), manipulant des composants de méthode rassemblés dans des packages de méthode et le choix des composants de méthode pour la création de nouvelles méthodes est dirigé par les buts. Dans l'approche (Ralyté, 2001b), (Ralyté, 2001c), les composants de méthode, ou composant de méthode chunk, sont perçus

comme un bloc insécable regroupant un modèle de processus et un modèle de produit. A l'instar des composants de méthode, leur situation de réutilisation est également décrite par le but que réalise le composant. Parmi les approches les plus récentes, nous trouvons l'approche Open Process Framework (OPF) (Henderson-Sellers, 2002), et (OPF, 2012). Les composants de méthode de cette approche, appelés fragment de méthode OPF sont construits sur la norme de métamodélisation ISO/IEC 24744 (ISO/IEC 24744, 2010). Dans cette approche, chaque élément de la méthode est vu comme un fragment de méthode. Une des dernières approches d'ingénierie des méthodes situationnelles à avoir été publiée dans (Guzélian, 2007a) et (Guzélian, 2007b) est une approche basée sur le paradigme des architectures orientée service où chaque composant de méthode est alors perçu comme un service.

L'ingénierie des méthodes est un domaine de recherche très dynamique depuis une quinzaine d'années mais son adoption par les entreprises reste limitée. Nous discuterons de cette limitation dans la section suivante.

1.2. Constat

L'utilisation des méthodes de développement des SI reste limitée dans la plupart des entreprises. Il existe toujours des cas d'échecs de développement de SI dus à des problèmes de méthodologie. La bonne gestion des méthodes de développement est vitale pour les entreprises. En effet, l'utilisation de méthodes pour le développement des SI permet d'améliorer les chances de réussite des projets et d'augmenter la qualité des systèmes produits. De plus, les systèmes d'information sont devenus incontournables pour les entreprises. En effet, ils doivent supporter tous les acteurs de l'entreprise dans leurs missions.

A cela s'ajoute le fait que les entreprises sont confrontées de nos jours à des évolutions de leurs systèmes d'information de plus en plus rapides. En effet, elles doivent répondre à des contraintes de marché dont la fréquence d'évolution a grandement augmenté. Les SI des entreprises sont en constante évolution. Le concept même de "béta permanente" est apparu pour refléter ce développement et l'amélioration continue du SI. Il est par conséquent nécessaire d'avoir une réponse méthodologique adaptée à ces nouvelles formes de développement.

Le développement d'un SI étant stratégique pour l'entreprise, les domaines d'application des SI sont de plus en plus variés et la complexité des SI croît de plus en plus. Depuis une quinzaine d'année il devient indispensable d'utiliser une méthodologie adaptée pour produire des SI de qualité.

Les méthodes sont le plus fréquemment définies de manière monolithique en se voulant universelles. Cependant, devant la diversité des projets et des domaines d'application, les méthodes ne peuvent pas être appliquées à la lettre. Les utilisateurs des méthodes ont la nécessité de les adapter en fonction du contexte de leurs projets et les méthodes classiques ne fournissent pas toujours cette flexibilité ou cette permissivité de configuration.

Malgré ces besoins de méthode pour le développement des SI et ces besoins d'adaptation de la méthode aux facteurs de contingence des projets, les approches d'ingénierie des méthodes situationnelles ont des difficultés à percer dans le domaine professionnel. Une quinzaine d'années après la publication de la première approche, elles rencontrent toujours des difficultés à s'implanter dans les entreprises alors qu'elles apportent des éléments de réponse à ces problèmes.

En pratique, dans les entreprises, les utilisateurs de méthode rencontrent de nos jours certaines difficultés dans l'application de leurs processus méthodologiques. Ils ont conscience de l'importance des méthodes dans la réussite de leurs projets de développement mais ont des problèmes pour trouver une méthode adaptée à leur contexte et produisent donc des adaptations ad hoc de leurs méthodes. Nous pouvons constater un manque de formation, d'accompagnement, d'outils et de maintenance des méthodes au cours des projets. Fréquemment, les connaissances sur les méthodes ou les adaptations ad hoc ne sont pas capitalisées. L'application de méthodes ou l'adaptation de méthodes peuvent également rencontrer une forte résistance aux changements de la part des utilisateurs dans l'entreprise si les changements sont trop rapides et fréquents et s'il existe encore quelques personnes réfractaires aux méthodes de développement.

Du point de vue des approches d'ingénierie des méthodes situationnelles, ces difficultés font du contexte organisationnel actuel des entreprises un terrain non propice à leur implantation. Elles souffrent également d'un certain nombre de lacunes freinant leur application.

En 2007, une analyse (Agerfalk, 2007) des approches a été établie par la communauté du domaine de l'ingénierie des méthodes. Le constat de cette analyse montre que la trop grande diversité des approches d'ingénierie des méthodes situationnelles génère une confusion chez les utilisateurs dans leur application. Cette diversité implique également des difficultés pour la communauté à définir une base commune pour la définition des approches d'ingénierie des méthodes situationnelles.

Le concept de composant de méthode est transverse à la majeure partie des approches d'ingénierie des méthodes situationnelles. Cependant, d'une approche à l'autre, l'ensemble des concepts constituant la définition et la structure des composants diffèrent. Entre les différentes approches, la couverture des cinq aspects d'une méthode par les composants varie grandement. Autrement dit, en fonction de l'approche, les concepts utilisés dans la structure de son composant de méthode associé ne couvrent que partiellement la définition d'une méthode selon Seligmann (Seligmann, 1989). On trouve, par exemple des aspects méthodologiques dans certaines approches et pas dans d'autres, mais, le plus souvent, c'est l'usage des méthodes ou l'aspect outil qui n'est pas pris en compte.

Malgré la définition d'une norme ISO/IEC 24744 (ISO/IEC 24744, 2010) pour la définition des composants, elle n'est que peu suivie en pratique par les différentes approches et les acteurs de l'ingénierie des méthodes. La plupart des composants de méthode produits aujourd'hui sont centralisés

dans des bases de données de composants propriétaires et leurs définitions sont issues d'approches différentes. Cette multitude des approches d'ingénierie des méthodes et le manque de standardisation dans le domaine provoquent des problèmes d'interopérabilité en pratique dans l'assemblage entre les différents composants pour construire de nouvelles méthodes. Nous pouvons également constater en pratique que les utilisateurs des méthodes préfèrent configurer les méthodes qu'ils utilisent déjà pour répondre aux nouvelles contraintes de leurs projets plutôt que de créer de nouvelles méthodes. Dans cette dernière décennie, les approches d'ingénierie des méthodes situationnelles se sont focalisées sur la vision modulaire des méthodes en utilisant par analogie les travaux réalisés dans le domaine de l'ingénierie des logiciels, notamment les principes qui ont pris place dans l'évolution de l'ingénierie des logiciels vers la définition des approches d'ingénierie des logiciels par composants. La dimension de l'usage des méthodes nouvellement créées ou adaptées par l'utilisation de l'ingénierie des méthodes n'a pas ou peu été prise en compte dans les travaux sur les approches d'ingénierie des méthodes (IM).

L'ensemble de ces mauvaises conditions d'application des méthodes de développement des SI dans les entreprises, ajoutées aux lacunes des approches d'ingénierie des méthodes situationnelles conduisent à une faible exploitation des possibilités offertes par ces dernières.

1.3. Problématique

La somme des lacunes que nous venons d'énoncer nous amène à nous interroger sur l'application des méthodes et leurs adaptations aux contextes de projets particuliers des entreprises.

Dans le domaine de l'ingénierie des méthodes situationnelles, nous proposons d'étudier dans cette thèse la mise en place et l'utilisation des approches d'ingénierie des méthodes situationnelles dans les entreprises. Le fait que ces approches n'ont pas réussi à percer dans le monde industriel au cours des deux dernières décennies soulève la problématique suivante :

Comment améliorer l'usage de l'ingénierie des méthodes situationnelle dans les entreprises ?

Cette problématique rassemble en un seul point l'ensemble des questions de recherche soulevées par l'usage des approches d'ingénierie des méthodes situationnelles en pratique.

La diversité des approches du domaine de l'ingénierie des méthodes situationnelles induit en pratique une confusion chez les utilisateurs de l'ingénierie des méthodes. Malgré les différentes tentatives qui ont pu être réalisées dans le domaine, il n'existe pas de base commune pour les approches d'ingénierie des méthodes situationnelles. Les approches ont certains concepts en commun mais, inversement, chacune d'entre elles manipule de nouveaux concepts.

➤ *Quelle base commune pour le domaine de l'ingénierie des méthodes situationnelles ?*

Actuellement, les solutions proposées dans le domaine de l'ingénierie des méthodes sont diverses et, pour la majeure partie, non standardisées. Les composants de méthode ne sont pas interopérables entre eux et sont centralisés dans des bases de données propriétaires.

- *Comment favoriser l'interopérabilité des solutions d'ingénierie des méthodes ? Quels sont les standards applicables aux solutions proposées ?*

L'usage des méthodes n'est pas pris en compte dans les approches d'ingénierie des méthodes situationnelles. L'aspect outil n'est pas couvert dans la définition structurelle des composants associés aux approches. Ceci implique que le support outillé de la méthode générée par l'approche n'est pas pris en charge.

- *Quelles sont les possibilités techniques pour apporter un support outillé aux méthodes résultantes de l'application d'un processus d'ingénierie des méthodes ?*

Nous pouvons constater, dans les entreprises, que lorsque les utilisateurs de méthode ont le besoin d'adapter une méthode pour un projet, ils se contentent de modifier et paramétrer une méthode existante déjà utilisée.

- *Lors de l'adaptation d'une méthode à un contexte de projet, un utilisateur a-t-il besoin d'assembler une nouvelle méthode ou alors de configurer une méthode existante ?*

Le manque d'accompagnement et les changements trop brutaux dans la mise en place de nouvelles méthodes dans les entreprises provoquent une forte résistance aux changements et une faible acceptation des nouvelles méthodes de la part des utilisateurs.

- *Comment intégrer progressivement une nouvelle approche dans une entreprise afin de réduire la résistance aux changements des utilisateurs et augmenter l'acceptation de cette dernière ?*

La problématique de cette thèse s'inscrit pleinement dans le domaine de l'ingénierie des méthodes situationnelles. L'objectif principal de notre travail est de proposer une nouvelle approche d'ingénierie des méthodes situationnelles tenant compte de l'usage des méthodes adaptées aux contextes des projets. Cette approche devra répondre aux caractéristiques suivantes :

- *L'uniformisation et la standardisation* : notre approche doit définir un socle commun pour les approches d'ingénierie des méthodes situationnelles existantes dans le but de proposer des composants de méthode standardisés et interopérables.
- *Répondre aux besoins de l'entreprise* : répondre au plus près aux besoins des entreprises concernant leurs pratiques d'adaptations de méthode.
- *Une définition des outils pour l'ingénierie des méthodes* : les méthodes créées ou configurées à partir de notre approche se devront de posséder un support outillé pour leur application dans

les meilleures conditions. Une architecture pour la définition d'une plateforme de construction d'outils de support des méthodes doit être incluse dans notre approche. Cette architecture doit également être compatible avec les contraintes spécifiques du domaine de l'ingénierie des méthodes.

- *Une intégration de l'ingénierie des méthodes dans les entreprises* : notre approche doit proposer un processus progressif d'intégration de celle-ci en entreprise afin de prendre en compte la résistance aux changements des utilisateurs et des organisations.

1.4. Méthode de résolution et propositions

Notre solution visant à l'amélioration de l'usage de l'ingénierie des méthodes dans les entreprises repose sur l'objectif de recherche suivant :

- Objectif de recherche : repenser l'ingénierie des méthodes situationnelles au travers du paradigme orienté service.

Par l'application d'une architecture orientée services à l'ingénierie des méthodes, nous proposons une approche dirigée par les besoins des utilisateurs, manipulant des composants de méthode sous la forme de "services" développés par un fournisseur de méthodes.

Nous pensons améliorer l'interopérabilité, la distribution et l'usage des composants de méthode avec l'utilisation de standards existants dans le domaine de l'ingénierie des services. Ces standards seront appliqués à la définition, la découverte et la configuration des composants de méthode afin de créer de nouveaux composants appelés "services méthodologiques" basés sur le paradigme des architectures orientés services (AOS).

Chaque service méthodologique inclut un outil supportant le composant de méthode dont il fournit une implémentation. En d'autres termes, un service méthodologique apporte l'aspect outil manquant à la plupart des approches de l'ingénierie des méthodes situationnelles actuelles. La solution apportée par un service méthodologique est une implémentation du modèle de processus du composant de méthode qu'il supporte. Les entrées consommées par cette implémentation et les sorties produites sont, quant à elles, définies par le ou les modèles de produit du composant de méthode implémenté.

Cette approche sous forme de services pour la définition des outils permet de créer rapidement un support à une méthode adaptée aux spécificités d'un contexte de projet. Un autre avantage est d'utiliser cette modularité liée à l'utilisation de service pour introduire progressivement un processus outillé d'ingénierie des méthodes dans une entreprise.

Nous nommerons cette nouvelle approche d'ingénierie des méthodes situationnelles "Méthodes avec une architecture orientée Service" (MaaS) par analogie aux approches de type "Logiciel en Tant que Service" (LTS) du domaine de l'ingénierie des logiciels.

L'approche proposée se découpe en trois parties (figure 1.2) :

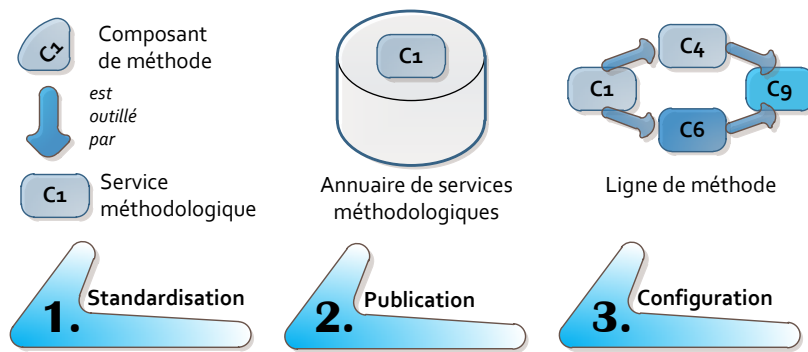


Figure 1.2 : MaaS, une approche orientée service

Le point de départ de cette approche est un ensemble de composants de méthode existants définis à l'aide d'une approche d'ingénierie des méthodes situationnelles.

Telle que représenté à la figure 1.2, la première partie de notre approche concerne la création d'un support outillé sous la forme de services méthodologiques englobant les composants de méthode dans leur aspect outil.

La deuxième étape de l'approche concerne le stockage et la récupération de ces services méthodologiques, créés par des fournisseurs de services méthodologiques, dans un annuaire standardisé.

La troisième partie traite de la configuration et de l'exécution des méthodes et de leurs outils adaptés aux facteurs de contingences des projets des entreprises.

La contribution apportée par cette thèse au domaine de l'ingénierie des méthodes situationnelles est une nouvelle approche d'ingénierie des méthodes situationnelles proposant le concept de service méthodologique (standardisé et interopérable). Ce concept permet de prendre en charge l'aspect outil d'une méthode et la variabilité incluse dans celle-ci.

1.5. Plan de la thèse

Cette thèse est organisée en sept chapitres.

Le chapitre 2 présente l'état de l'art des approches d'ingénierie des méthodes situationnelles et les positionne par rapport à un cadre de référence analytique.

Le chapitre 3 offre un aperçu de la démarche orientée service pour la configuration de méthode outillée.

Le chapitre 4 présente l'ontologie de métamodèles de composant de méthode (OMCM) en tant que base commune pour les approches d'IMS. Ce chapitre décrit également l'application de cette ontologie dans un annuaire de services méthodologiques lors la description et la recherche de services.

Le chapitre 5 décrit le métamodèle de service méthodologique atomique et le métamodèle de ligne de méthode ainsi que leur opérationnalisation. Il présente également l'intégration progressive de notre approche dans une entreprise.

Le chapitre 6 illustre l'approche MaaS par un cas d'application d'élaboration d'une ligne de méthode pour l'étape de lancement d'un projet conformément aux méthodes agiles les plus utilisées en entreprise.

Le chapitre 7 décrit la spécification conceptuelle de la plateforme de l'approche MaaS et de ses implémentations selon divers environnements techniques orientés service.

Le chapitre 8 conclut le travail présenté dans cette thèse et propose des perspectives de recherche de ce travail.

Chapitre 2. Etat de l'art

2.1. Introduction

Ce chapitre traite de l'état de l'art des recherches dans le domaine de l'ingénierie des méthodes. Il présente une comparaison des différentes approches d'ingénierie des méthodes situationnelles. Nous proposons de nous focaliser sur l'élément central de ces approches, le composant de méthode. En effet, le concept de composant est fortement dépendant de l'approche dont il est issu. Par conséquent, nous allons placer les composants de méthode au centre de cette étude et nous intéresser plus particulièrement à leurs démarches de construction et leur facilité d'utilisation.

Historiquement, la première approche d'ingénierie des méthodes situationnelles est l'approche à base de "fragments de méthode" (Brinkkemper, 1996), (Brinkkemper, 1998), (Brinkkemper, 1999), (Harmsen, 1994) et (Harmsen, 1997). Cette approche préconise très tôt une vision modulaire d'une méthode qui inspirera par la suite tous les autres composants des approches d'IMS telles que : les chunks (Rolland, 1996), (Rolland, 1998), (Ralyté, 2001b), (Ralyté, 2001c), (Ralyté, 2003) et (Mirbel, 2006a), les composants de méthode (Agerfalk, 2003), (Wistrand, 2004), (Karlsson, 2005) et (Karlsson, 2006), les composants Open Process Framework (OPF) (Henderson-Sellers, 1996), (Firesmith, 2002), (Gonzalez-Perez, 2007) et (OPF, 2012) et les services méthodologiques Service-Oriented Meta-Method (SO2M) (Guzélian, 2007a) et (Guzélian, 2007b). Toutes ces approches d'IMS s'accordent sur la définition d'une méthode proposée par Seligmann (Seligmann, 1989).

Dans ce chapitre, nous proposons d'étudier les approches d'ingénierie des méthodes citées ci-dessus au travers d'un cadre de référence dont la structure est inspirée de (Prieto-Diaz, 1991). L'originalité de ce cadre est de pouvoir observer un concept central selon plusieurs points de vue (axes). Cette observation est formalisée par l'identification d'un ensemble de propriétés (facettes) selon le point de vue choisi.

Le cadre de référence présenté dans ce chapitre est centré sur les approches et leurs composants de méthode. Son objectif est d'évaluer les démarches et les composants des approches d'IMS en observant les caractéristiques des composants et de leur cycle de vie. Cette comparaison, effectuée sur les cinq approches d'IMS les plus connues, nous permet de mettre en avant leurs efforts d'utilisabilité selon : le composant, la démarche et l'interopérabilité des composants, les différences entre les approches et de préciser l'étendue de la couverture d'une démarche vis-à-vis des aspects de la définition d'une méthode introduite par (Seligmann, 1989) et (Plihon, 1995).

Dans un premier temps, nous présentons à la section 2.2 le cadre de référence utilisé pour structurer l'état de l'art. Les cinq approches d'IMS sont décrites à la section 2.3 en les évaluant en fonction du cadre de référence. La section 2.4 synthétise la comparaison de ces cinq approches. Enfin, les limites

de ces approches en termes d'utilisabilité sont mises en avant lors de la conclusion (section 2.5) et reliées à la problématique adressée par cette thèse.

2.2. Cadre de référence

Le métamodèle utilisé pour réaliser ce cadre de référence est basé sur celui introduit par (Prieto-Diaz, 1991). L'objectif de l'approche de (Prieto-Diaz, 1991) est d'établir une classification précise des composants logiciels en se basant sur leur descripteur, leur association à un lexique de termes (thésaurus) et sur un graphe des facettes. Le cadre proposé pour effectuer cette classification repose sur l'observation d'un sujet selon quatre points de vue différents que nous nommerons "axes". Chacun de ces quatre axes est personnalisé par un ensemble de "facettes" en fonction de ce sujet. Une facette (ou "attribut") représente une caractéristique observable pertinente pour la classification du sujet de l'étude.

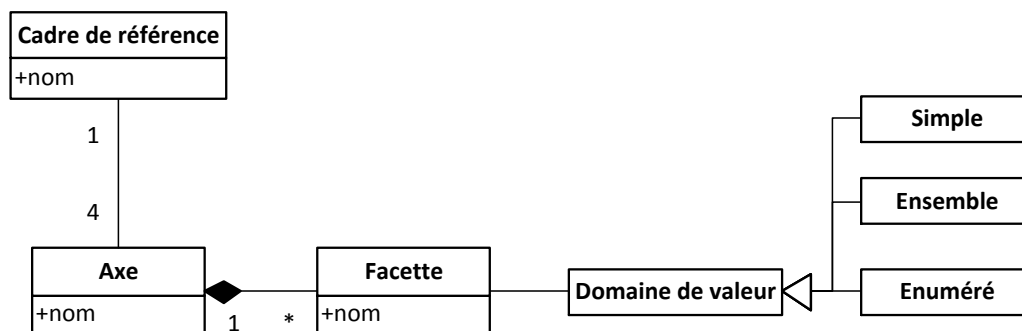


Figure 2.1 : Métamodèle utilisé pour le cadre de référence

Chaque axe est caractérisé par son nom et un ensemble de facettes. Celles-ci permettent de canaliser l'observation du sujet selon le point de vue défini dans l'axe. Une facette est également identifiée par son nom et est caractérisée par un domaine de valeurs (voir figure 2.1).

Trois types de domaines de valeurs peuvent être associés à une facette: le type "simple", le type "énuméré" et le type "ensemble". Un domaine de valeur *simple* correspond à un type de base prédéfini tel que les booléens, entiers, réels ou encore chaînes de caractères. Un domaine de valeur *énuméré* (ou *Enum*) correspond à une seule valeur désignée parmi un ensemble de valeurs prédéfinies, par exemple la "valeur b" choisie parmi l'ensemble de valeurs suivant: Enum{valeur a, valeur b, valeur c}. Un domaine de valeur *ensemble* (ou *Set*) correspond à une collection de valeurs sélectionnées dans un ensemble de valeurs prédéfinies, par exemple les valeurs {valeur a, valeur b} sélectionnées parmi l'ensemble de valeurs suivant : Set{valeur a, valeur b, valeur c}.

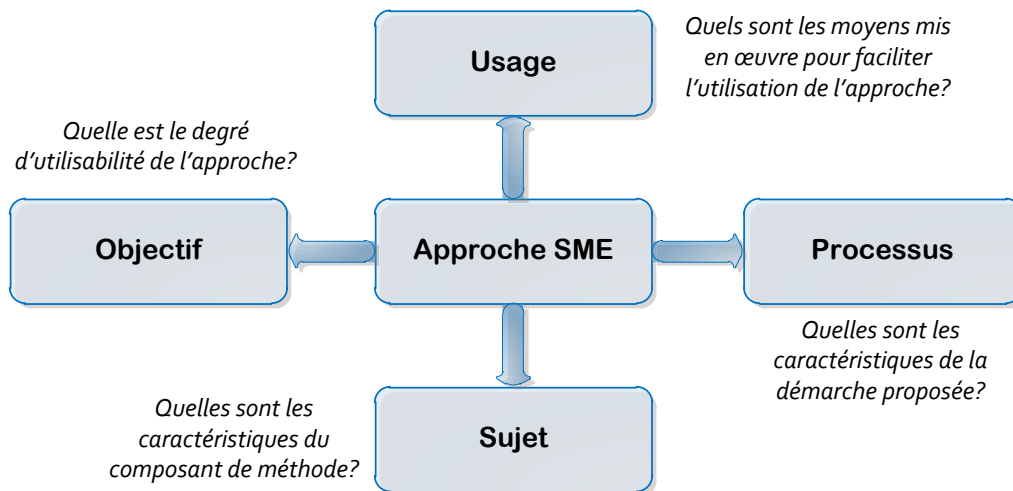


Figure 2.2 : Cadre de référence pour la comparaison des composants de méthode.

La figure 2.2 présente les quatre axes (objectif, usage, processus et sujet) qui nous semblent pertinents pour la comparaison des approches et de leur composant. Une approche d'IMS s'organise autour d'un métamodèle de composant de méthode et propose un processus de construction et de réutilisation de ces composants. Les propriétés discriminantes du composant de méthode constituent l'axe *sujet* du cadre de référence. L'axe *processus* permet d'identifier les caractéristiques de la démarche d'IMS. Le but de cette comparaison est de mesurer la facilité d'utilisation des démarches d'IMS et ceci est le propos de l'axe objectif. Enfin, les moyens mis en œuvre aux niveaux conceptuel et technique pour favoriser l'utilisation de l'approche et des composants sont identifiés par l'axe *usage*. La description de chaque axe du cadre de référence est détaillée dans la suite de cette section.

2.2.1 L'axe objectif

Le degré d'utilisabilité d'une approche d'IMS peut se mesurer selon :

- Le degré d'interopérabilité du composant de méthode (développé selon l'approche d'IMS) avec d'autres composants développés selon d'autres approches.
- La nature de l'interaction de la démarche d'IMS avec l'ingénieur méthode. C'est-à-dire le degré de support apporté à l'ingénieur méthode durant l'application de la démarche.

La notion d'interopérabilité a été introduite et discutée dès l'émergence du domaine de l'ingénierie des méthodes situationnelles (Brinkkemper, 1996). Les approches, dans leur grande majorité, s'intéressent prioritairement à définir des composants de méthode et les répertorient dans une base de composants mais l'interopérabilité de ces composants est limitée à cette base et à ce formalisme. Dans ce cas, nous parlerons d'interopérabilité "interne" à l'approche.

Lorsqu'une approche a le souci de structurer des composants de manière standard, ou qu'elle propose une base de composants issus de différentes approches, la facette "interopérabilité" est dite "externe". Néanmoins, cette interopérabilité peut être limitée par l'environnement supporté par le standard. Par

exemple, un standard basé sur un paradigme donné peut limiter l'utilisation du composant à cet environnement. Dans ce cas, la facette interopérabilité est dite "externe dans le même environnement". Au contraire, la valeur "externe dans des environnements différents" est affectée lorsque l'interopérabilité externe n'est pas limitée.

Il est possible d'augmenter la productivité des ingénieurs méthodes en fonction du degré d'automatisation de l'interaction entre l'ingénieur méthode et la démarche de l'approche. Cette mesure d'interaction peut être découpée en trois niveaux. Les approches d'ingénierie des méthodes situationnelles peuvent fournir un processus complètement "automatisé" ou "assisté" (semi-automatisé) de construction, réutilisation et assemblage des composants de méthode. Dans le cas d'une démarche uniquement conceptuelle, l'application de l'approche est dite "manuelle".

Les facettes de l'axe "objectif" sont :

Interopérabilité : *ENUM*{interne, externe dans le même environnement, externe dans des environnements différents}.

Interactivité : *ENUM*{manuelle, assistée, automatisée}.

2.2.2 L'axe usage

Cet axe permet d'identifier les moyens mis en œuvre par l'approche pour favoriser son utilisation en pratique.

Nous distinguons dans notre observation :

- l'usage du composant de méthode.
- l'usage du processus de l'approche d'IMS.

Les approches d'IMS ont toutes pour définition d'une méthode, celle proposée par Seligmann (Seligmann, 1989) et étendue par Rolland (Plihon, 1995). En effet, une méthode est caractérisée par rapport à cinq aspects : paradigme, produit, processus, organisationnel et outil. Néanmoins, les approches et leur composant couvrent de manière inégale les cinq aspects d'une méthode. En pratique, il est communément admis qu'une méthode définie par exemple sans prendre en compte l'aspect outil ni l'aspect organisationnel est difficilement utilisable. La couverture des aspects dans le composant de méthode est une facette (nommée *couverture des composants*) nous permettant d'évaluer le degré d'utilisabilité des composants. Le domaine de valeur de cette facette est l'ensemble des cinq aspects d'une méthode.

Comme le propose Brimkkemper (Brinkemper, 1996), un processus outillé peut être considéré comme un ensemble d'outils supportant chacun une partie du processus. Par conséquent, nous

caractériserons le support de la démarche d'IMS en mentionnant les parties de processus qui sont outillés, comme par exemple :

- *Recherche de composants*
- *Stockage de composants dans une base*
- *Construction de composants*
- *Manipulation de composants*

Un composant de méthode décrit les aspects paradigme, produit, processus et organisationnel tandis que le support outillé de ce composant représente l'aspect outil de celui-ci. L'opérationnalisation de cet aspect est réalisée par un composant technique (logiciel). La concrétisation d'un composant de méthode par un composant technique est un élément important augmentant le degré d'utilisabilité de l'approche. En effet, si chaque composant possède un support outillé, une méthode basée sur un ensemble de composants pourra elle-même être dite outillée. Cette capacité est représentée par la valeur *implémentation du composant* dans la facette "Outil et implémentation".

Les facettes de l'axe "usage" sont :

Couverture des composants : $SET\{\text{aspect paradigme, aspect produit, aspect processus, aspect organisationnel, aspect outil}\}$.

Outil et implémentation : $SET\{\text{stockage des composants, manipulation de composants, implémentation du composant, recherche de composants, construction de composants}\}$.

2.2.3 L'axe processus

L'axe processus permet d'identifier les caractéristiques discriminantes de la démarche proposée par l'approche d'IMS. Chaque étape de la démarche (la décomposition d'une méthode en composants, la sélection des composants, la construction d'une nouvelle méthode) fait l'objet d'une facette.

L'axe processus permet de mettre en avant les manières de concevoir et d'utiliser les composants de méthode. Les attributs de cet axe ont pour objectif de décrire l'ensemble des activités principales des approches d'IMS interagissant avec les composants (la décomposition d'une méthode en composants, la sélection des composants, la mise en correspondance des composants avec la situation du projet, la construction d'une nouvelle méthode).

La plupart des approches d'IMS nécessitent la réingénierie des méthodes en composants de méthode pour pouvoir les réutiliser ultérieurement durant le processus de construction d'une méthode situationnelle. Cette étape permet de décomposer une méthode selon le principe de décomposition proposé par l'approche. La facette "principe de décomposition" représente le paradigme dirigeant cette étape. Par exemple, le concept de but dirige l'activité de réingénierie des méthodes dans l'approche

des composants de méthode chunk. Le principe de décomposition est propre à chaque approche, le domaine de valeur de cette facette est du type prédéfini "chaîne de caractères".

Une fois que les méthodes sont découpées en composants stockés dans des bases, ceux-ci peuvent être récupérés pour être réutilisés dans d'autres définitions de méthodes. Le mécanisme de réutilisation implique d'une part une activité de recherche du composant à partir d'une base et, d'autre part, une activité de sélection du composant le plus approprié à la situation. La facette discriminante (nommé *principe de recherche/sélection*) liée à l'activité de recherche correspond aux types d'information sur lesquels porte la requête. L'adéquation d'un composant à un contexte de projet (facette *correspondance avec la situation*) s'évalue en fonction du modèle préconisé par chaque approche.

L'étape de construction d'une méthode situationnelle peut être réalisée selon plusieurs stratégies : assemblage, assemblage avec chevauchement et extension, réduction ou encore par construction incrémentale. La stratégie par *assemblage par association* consiste à regrouper des composants (sans chevauchement) en accord avec un contexte spécifique de projet pour former une méthode unique (Ralyté, 2001c). Le regroupement de composants avec chevauchement entre les composants pour former une nouvelle méthode est supporté par la stratégie spécifique d'*assemblage par intégration* (Ralyté, 2001c). Une nouvelle méthode peut être construite par ajout de nouveaux composants à une méthode existante (*extension*) ou par suppression de composants d'une méthode existante (*réduction*) (Wistrand, 2004). Une stratégie plus récente (Weerd, 2007) propose d'adapter une méthode au contexte de projet d'une entreprise de manière incrémentale. En d'autres termes, des composants de méthode sont intégrés, modifiés ou supprimés d'une méthode incrément par incrément sur la durée.

Les facettes de l'axe "processus" sont :

Principe de décomposition : chaîne de caractères.

Principe de recherche/sélection : chaîne de caractères.

Correspondance avec la situation : chaîne de caractères.

Technique de construction : SET{assemblage par association, assemblage par intégration, extension, réduction, incrémentale}.

2.2.4 L'axe sujet

Cet axe concentre l'observation sur les aspects internes du composant de méthode.

Une revue de la littérature existante nous permet d'identifier trois niveaux de représentation possibles des composants de méthode : *intentionnel*, *structurel* et *opérationnel*. Le niveau intentionnel permet de représenter sous forme de buts le contexte de réutilisation d'un composant. Le niveau structurel détermine la structure interne des composants et leurs relations. Le niveau opérationnel, quant à lui, prend en compte l'implémentation d'une ou plusieurs parties du composant. Les niveaux de

représentation couverts par l'approche vis-à-vis des composants fait l'objet de la facette *niveau de représentation*.

La prise en compte des différentes granularités d'un composant peut s'effectuer de manière réflexive, c'est-à-dire qu'une agrégation de composants peut être considérée elle-même comme un composant. La facette *réflexivité* permet de déterminer si le modèle de composant proposé par l'approche est réflexif ou pas.

L'aspect produit d'un composant est défini à l'aide de modèles à différents niveaux d'abstraction. La facette (nommée *niveaux d'abstraction*) permet de décrire les différents niveaux d'abstraction des modèles pris en compte par l'approche : *instance*, *modèle*, *métamodèle* et *méta-métamodèle* (OMG/MOF, 2006).

La dernière facette (nommée *formalisme*) permet d'identifier le niveau du formalisme utilisé pour décrire la structure interne du composant. Comme le propose (Brinkkemper, 2001), cette caractéristique comporte deux classes de formalisme : *conceptuel* et *technique*. Le formalisme technique induit l'usage d'outils pour manipuler cette description, contrairement au formalisme de type conceptuel.

Les facettes de l'axe "sujet" sont :

Niveau de représentation : $SET\{\text{intentionnel, structurel, opérationnel}\}$.

Récurtivité : booléen.

Niveaux d'abstraction : $SET\{\text{instance, modèle, métamodèle, méta-métamodèle}\}$.

Formalisme : $SET\{\text{conceptuel, technique}\}$.

2.3. Les approches d'ingénierie des méthodes situationnelles

La comparaison proposée concerne les quatre approches d'IMS orientées composant (Brinkkemper, 1998), (Ralyté, 2001b) et (Ralyté, 2001c), (Wistrand, 2004) et (Firesmith, 2002) les plus connues du domaine ainsi que celle proposant une orientation service de ses composants (Guzélian, 2007a). Cette section détaille chacune de ces approches en analysant, d'une part, la structure du composant et, d'autre part, la démarche proposée. La synthèse de cette analyse est ensuite spécifiée par l'évaluation de ces facettes selon le cadre de référence présenté dans la section 2.2 de ce chapitre.

2.3.1 Les fragments de méthode

Cette approche est issue des travaux de Brinkkemper (Brinkkemper, 1996), (Brinkkemper, 1998), (Brinkkemper, 1999) et de Harmsem (Harmsen, 1994), (Harmsen, 1997). Elle constitue la première approche d'IMS à avoir vu le jour et les approches d'IMS apparues par la suite s'en sont inspirées. En

d'autres termes, l'approche des fragments de méthode a servi de base au domaine de l'ingénierie des méthodes situationnelles.

2.3.1.1. Structure d'un fragment de méthode

La structure du composant de méthode "fragment de méthode" est dirigé par la définition d'une méthode selon les deux aspects produit et processus. L'originalité de l'approche est de promouvoir la décomposition de la description du produit et du processus d'une méthode en fragments. C'est pour cette raison que deux types de fragment sont proposés : fragments de produit et fragments de processus (voir figure 2.3).

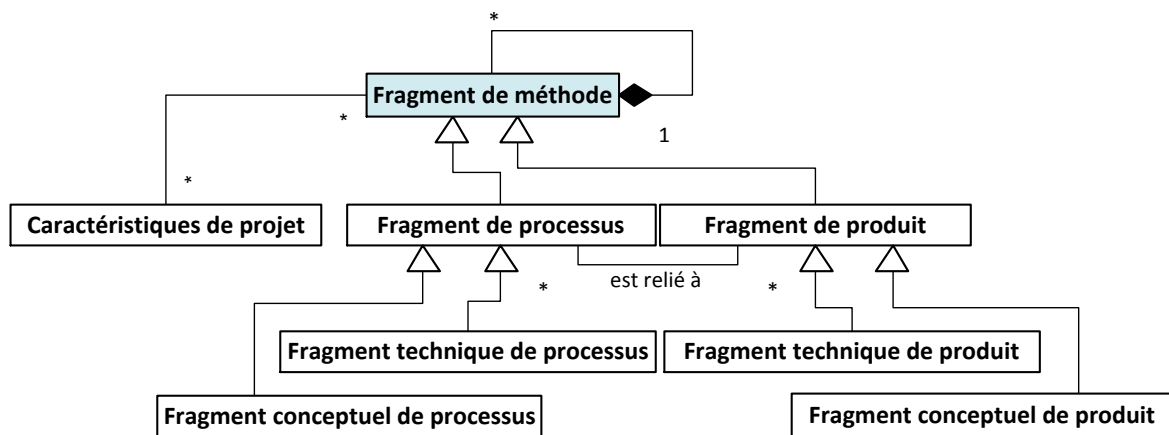


Figure 2.3 : Structure d'un fragment de méthode adapté de (Brinkkemper, 1998)

Clairement, le fragment de méthode n'encapsule pas au sein d'un même concept les aspects produit et processus. Un fragment de produit est une description des produits et des sous-produits devant être construits par l'application d'une méthode. Cela comprend par exemple les livrables, les documents, les modèles, les diagrammes etc. Un fragment de processus, quant à lui, décrit les différentes phases, activités et tâches d'un processus méthodologique destiné à réaliser des produits représentés par des fragments de produits (Harmsen, 1994).

La relation entre fragment de processus et fragment de produit permet, d'une part, de représenter le ou les produit(s) réalisé(s) à partir d'un processus et, d'autre part, d'identifier le ou les processus permettant de réaliser le produit. Cela signifie qu'un fragment de produit peut être requis dans l'application de plusieurs fragments de processus différents. Inversement, un fragment de processus peut être amené à manipuler ou produire des fragments de produit différents.

Il existe un lien de récursivité sur les fragments de méthode. Ainsi, un fragment de produit peut être composé d'un ensemble de fragments de produit définis à une granularité plus fine. De même, pour les fragments de processus, ils peuvent être eux aussi constitués d'autres fragments de processus.

D'après Harmsen (Harmsen, 1994), un fragment de méthode est défini à un niveau conceptuel lorsqu'il correspond à une description des activités ou des modèles, tandis qu'un fragment défini à un niveau technique est un outil pour le support d'un fragment de méthode conceptuel (produit ou processus).

Enfin, plusieurs caractéristiques de projet peuvent être associées à un fragment de méthode pour faciliter la sélection de celui-ci dans une base de méthodes.

2.3.1.2. L'approche associée aux fragments de méthode

Cette approche d'IMS s'organise autour d'une base de méthodes articulée sur la structure des fragments de méthode présentée ci-dessus. Selon la démarche "type" des approches d'IMS (voir figure 1.1 dans le chapitre 1), nous pouvons regretter l'absence dans cette approche de principe de décomposition de méthodes existantes en fragment de méthode. De même, aucune méthodologie ou bonne pratique n'est proposée pour la construction de nouveaux fragments de méthode.

Cependant, cette approche fournit une méthodologie détaillée pour la construction d'une base de méthodes afin de stocker un ensemble de fragments de méthode existants (Harmsen, 1997).

Lors de la construction d'une méthode situationnelle, la première étape consiste à caractériser le contexte du projet (facteurs de contingences) dans lequel cette méthode doit s'appliquer. (Harmsen, 1997) propose trois catégories de facteurs de contingence :

- Les facteurs environnementaux caractérisent l'organisation dans laquelle se déroule le projet.
- Les facteurs organisationnels du projet décrivent la structure de l'équipe, sa taille et les dépendances du projet vis-à-vis d'autres projets en cours ou à venir.
- Les facteurs liés au sujet du projet, décrivant tour à tour les caractéristiques du SI, la qualité des buts et des exigences du projet, ainsi que son degré d'innovation.

Une fois les facteurs de contingence du projet établis, (Harmsen, 1997) propose de les corrélérer avec des indicateurs de succès du projet représentés par des buts non-fonctionnels. A partir de cette corrélation, l'approche fournit un guide de configuration qui permet d'obtenir un scénario méthodologique initial adapté au contexte du projet (c'est-à-dire un patron d'une méthode "idéale").

Les requêtes de recherche des fragments de méthode les plus adaptés à la situation du projet sont formulées en fonction des besoins exprimés dans ce scénario idéal. Le fragment le plus approprié au scénario et au contexte du projet est sélectionné avant d'être agrégé à la méthode en construction. Afin de garantir la qualité de cette méthode nouvellement produite, (Harmsen, 1997) définit un certain nombre de mesures de qualité: la complétude, la consistance, l'efficacité, la robustesse et l'applicabilité.

2.3.1.3. Application du cadre de référence aux fragments de méthode

Axe objectif

Concernant l'axe "objectif" de notre cadre de référence, les fragments de méthode ne proposent pas l'utilisation de standards pour faciliter leur intégration avec d'autres composants de méthode. Nous pouvons donc les caractériser par une *interopérabilité* interne. En revanche, un utilisateur appliquant cette approche pour la construction de nouvelle méthode est assisté par des outils : une base méthode, un langage appelé MEL (Method Engineering Language) et son interpréteur. Ceci permet d'affecter la valeur "assistée" à la facette *interactivité*.

Axe usage

La structure des fragments de méthode traite des aspects produit, processus et paradigme (facette *couverture du composant*). Cependant, l'aspect organisationnel n'est pas directement pris en charge dans les fragments de méthode. Il apparaît avec la notion d'acteur dans la base de méthode, cette notion étant absente des fragments. La méthode finale assemblée ne traite donc pas l'aspect organisationnel. Comme nous l'avons vu précédemment, il est clair que la démarche est supportée par des *outils* : un système de gestion de base de méthode appelé "Decamerone" (Harmsen, 1995) pour le stockage, la manipulation, la recherche et la construction des fragments de méthode. Néanmoins, la méthode situationnelle construite par cette approche n'est pas outillée malgré la présence des concepts de fragments techniques de produit et de processus dans le modèle. En effet, la démarche ne couvre pas la spécification des fragments techniques en pratique. Par conséquent, l'aspect outil n'est pas traité dans les fragments de méthode.

Axe processus

Comme énoncé dans la section 2.3.1.2, l'approche des fragments de méthode ne propose pas de principe de décomposition pour les méthodes existantes. Elle présente cependant les mécanismes de *sélection* de fragments par requête sur le contenu du descripteur du fragment dans la base de méthode. La *correspondance avec la situation* du projet est faite à l'aide d'un scénario de méthode "idéal" déduit en fonction des facteurs de contingences du projet. La *technique de construction* utilisée ensuite sur les fragments sélectionnés est un assemblage par association ou un assemblage par intégration de ces derniers pour former une nouvelle méthode, ou encore plus récemment (Weerd, 2007) une intégration incrémentale de ceux-ci dans le processus d'adaptation d'une méthode.

Axe sujet

Les fragments de méthode ont un *niveau de représentation* structurel, mais également opérationnel avec les fragments de méthode techniques. Le *formalisme* utilisé pour ces représentations est essentiellement conceptuel et les *niveaux d'abstraction* pour la description des fragments sont le

modèle et le métamodèle. Nous pouvons également conclure que le principe de récursivité est utilisé dans cette approche car les fragments peuvent être composés de fragments de granularité inférieure.

2.3.2 Les chunks

L'approche des chunks de méthode a été développée dans (Rolland, 1996), (Rolland, 1998), (Ralyté, 2001b), (Ralyté, 2001c), (Ralyté, 2003), (Mirbel, 2006a). L'approche considère une méthode selon les deux aspects produit et processus et applique le principe d'encapsulation de ces deux aspects aux concepts de "méthode" et de "composant de méthode" appelé chunk. Un contexte de réutilisation est ajouté à l'extérieur du composant pour aider à sa réutilisation.

2.3.2.1. Structure d'un chunk

Un composant de méthode de type chunk est défini comme une partie autonome d'une méthode (Mirbel, 2006a). Ce composant est constitué d'une partie de produit et d'une partie de processus inter reliées. Une particularité du chunk est qu'il possède un seul modèle de produit relié à un seul modèle de processus. Ces deux parties sont encapsulées dans un conteneur abstrait appelé corps.

Le modèle de produit de ce composant est défini par un ensemble de concepts, de relations entre ces concepts, et de contraintes limitant la sémantique des concepts utilisés (Mirbel, 2006a). De plus, des contraintes de niveau modèle (règles de validation) sont également définies et utilisées lors du processus de modélisation pour valider le schéma obtenu. Un modèle de produit préconise à son utilisateur l'usage d'un paradigme particulier et est implémenté à l'aide d'un langage de modélisation.

Le modèle de processus, quant à lui, explique comment construire un produit conforme au modèle de produit qui lui est rattaché. Un modèle de processus est constitué d'un ensemble de directives définies comme des assertions, des bonnes pratiques, des procédures ou encore des moyens d'action représentant une connaissance méthodologique. Ceux sont des guides pour les utilisateurs afin de réaliser une intention dans une situation donnée (Mirbel, 2006a). Une directive peut être "simple" et contenir des informations méthodologiques informelles et textuelles, ou alors, une directive peut être définie comme "tactique" et constituée d'un ensemble d'étapes à exécuter séquentiellement. Enfin, une directive peut être complexe et dite "stratégique". Dans ce dernier cas, une directive est composée d'un ensemble des sous-directives décrivant les différents moyens pour réaliser un produit ou un sous-produit.

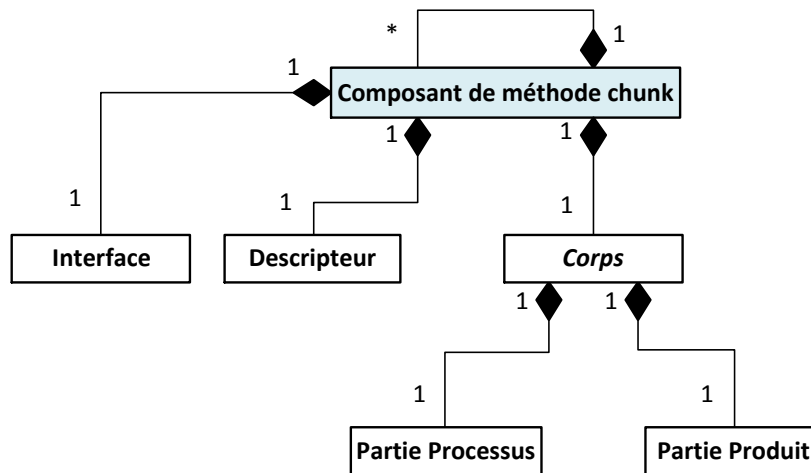


Figure 2.4 : Structure d'un composant de méthode chunk adapté de (Ralyté, 2001b) et (Ralyté, 2001c)

Comme nous pouvons l'observer à la figure 2.4, un chunk peut être défini à différents niveaux de granularité. En effet, un chunk peut être lui-même composé d'autres chunks et une méthode complète sera elle-même perçue comme un composant de méthode.

A chaque chunk est associée une interface, résumant le contexte d'application du composant de méthode au travers d'un couple <situation, intention>. La partie *Intention* explique le but que le chunk permet de satisfaire alors que la partie *Situation* représente le contexte dans lequel le but s'applique (Mirbel, 2006a).

L'approche, propose en plus de l'interface et du corps, un descripteur de composant associé au chunk afin d'améliorer sa réutilisabilité. En effet, ce descripteur permet d'identifier le contexte de réutilisation du chunk en précisant de manière plus détaillée le contexte du projet dans lequel l'application de ce chunk est pertinente (Mirbel, 2006a).

2.3.2.2. L'approche associée aux chunks

L'approche des composants de méthode de type chunk propose un processus pour la réingénierie des méthodes. Ce processus propose une décomposition des méthodes par les buts ou intentions. Chaque méthode est décomposée en directives réalisant des intentions et réutilisables indépendamment de la méthode elle-même (Ralyté, 2001a). Pour chacune de ces directives, un chunk est construit par une application récursive de cette décomposition dirigée par les intentions ou les buts.

Concernant le stockage et la recherche de composants de méthode, l'approche propose de stocker les chunks dans une base de méthode structurée en deux parties. La première partie concerne la connaissance méthodologique réutilisable représentant le contenu des composants de méthode. La seconde partie décrit la méta-connaissance nécessaire pour la sélection d'extraction des composants de la base, c'est-à-dire le descripteur et la signature de chaque chunk (les parties Interface et Descripteur) (Ralyté, 2001a).

Dans un premier temps, l'utilisateur définit les besoins méthodologiques spécifiques à la situation de son projet à l'aide d'une carte d'exigences. Ensuite, l'utilisateur formule une requête en donnant les valeurs des attributs du descripteur et de l'interface du chunk qu'il souhaite. La sélection des chunks s'effectue par l'application de la requête de l'utilisateur sur la base de méthode afin de récupérer un premier ensemble de composants. Des mesures de similarité sont appliquées à chacun des chunks de cet ensemble résultat afin de mesurer leur adéquation avec l'ensemble des exigences situationnelles formulées par l'utilisateur (Ralyté, 2003). Les mesures de similarité prennent en compte les aspects sémantiques et structurels du modèle de produit, mais également du modèle de processus (Carte). La similarité du modèle de processus est mesurée à la fois sur la sémantique ("affinité sémantique des intentions" et "affinité sémantique des stratégies") et la structure ("similarité structurelles des intentions" et "similarité structurelles des stratégies") (Ralyté, 2001b).

Une nouvelle méthode adaptée aux besoins de l'utilisateur est ensuite créée par l'assemblage des composants recherchés dans la base de méthodes. Cette approche propose deux façons d'assembler les composants de méthode : par association ou par intégration. L'association de composants de méthode consiste à associer les chunks C1 et C2 selon un mode producteur/consommateur. En effet, le produit résultant de l'application du chunk C1 est la situation d'entrée du chunk C2. L'intégration des composants de méthode consiste à identifier entre deux chunks C1 et C2 les éléments communs de leurs modèles de processus et de produit afin de les fusionner (Ralyté, 2001b).

2.3.2.3. Application du cadre de référence aux chunks

Axe objectif

L'approche ne propose pas de standards pour l'interopérabilité des chunks avec d'autres composants de méthode. Nous pouvons donc caractériser l'*interopérabilité* des chunks comme interne. Concernant l'*interactivité* de la démarche avec les utilisateurs, l'application de cette approche est exclusivement manuelle.

Axe usage

Un chunk est construit sur les trois aspects d'une méthode qui sont les trois aspects paradigme, produit et processus. En effet, ce type de composant de méthode est défini dans un paradigme donné, celui préconisé par le modèle de produit. Ce dernier est relié au modèle de processus permettant de le produire. Les *outils* supportant cette approche se limitent à une base de méthode pour le stockage et la recherche de chunks et aucun support outillé n'assiste l'étape de construction d'une nouvelle méthode par assemblage ou intégration.

Axe processus

L'approche d'IMS associée aux chunks propose un processus de réingénierie des méthodes en utilisant un *principe de décomposition* dirigé par les intentions. Une fois créés, les nouveaux chunks sont stockés dans une base de méthode. Ils sont ensuite *recherchés et sélectionnés* dans la base à l'aide d'une requête et des mesures de similarité. Ces mesures de similarité s'effectuent entre les composants sélectionnés et la situation du projet pour lequel il est nécessaire de construire une nouvelle méthode. La *situation* du projet est caractérisée à l'aide d'une carte des exigences. Enfin, à l'aide d'une base de composants réutilisables, une nouvelle méthode peut être construite en utilisant les *techniques de construction* par extension, par assemblage par association, en associant les chunks les uns aux autres, ou par assemblage par intégration, en intégrant / fusionnant les chunks entre eux.

Axe sujet

Dans cette approche, les chunks sont *représentés au niveau* intentionnel à la fois au niveau de l'interface et au niveau du descripteur sous forme d'intention ou de but. Dans la partie Corps, les chunks fournissent une représentation conceptuelle des modèles de produit et de processus selon un *formalisme* exclusivement *conceptuel*. Les niveaux d'abstraction de ces représentations pris en compte par l'approche sont : les modèles et les métamodèles. Enfin, le modèle de chunk supporte le principe de *récurtivité* pour représenter des chunks à différentes granularités. Comme vu à la section 2.3.2.1, un chunk peut être atomique ou composite et assemblé à d'autres chunks.

2.3.3 Les composants de méthode

Les composants de méthode sont issus de l'approche proposée par (Agerfalk, 2003), (Wistrand, 2004), (Karlsson, 2005), (Karlsson, 2006). Cette approche est utilisée dans un cadre spécifique de l'IMS, la configuration de méthode (Karlsson, 2004). Les composants de méthode de cette approche introduisent un concept plus large que les composants préconisés par les autres approches.

2.3.3.1. Structure d'un composant de méthode

Un composant de méthode est défini par Karlsson (Karlsson, 2005) comme une partie autonome et indépendante d'une méthode d'ingénierie d'un système. Il doit adresser un aspect particulier du problème à traiter. Un composant représente la partie la plus succincte et applicable possible en pratique d'une méthode (Agerfalk, 2007). Il peut être relié à d'autres composants de méthode comme l'illustre la figure 2.5 afin de constituer une méthode. Ce concept de composant est sémantiquement plus large que ceux des autres approches car il est défini comme une composition d'éléments de méthode. Ce dernier concept englobe tous les éléments constituant une méthode : notations, concepts, artefacts, actions et acteurs.

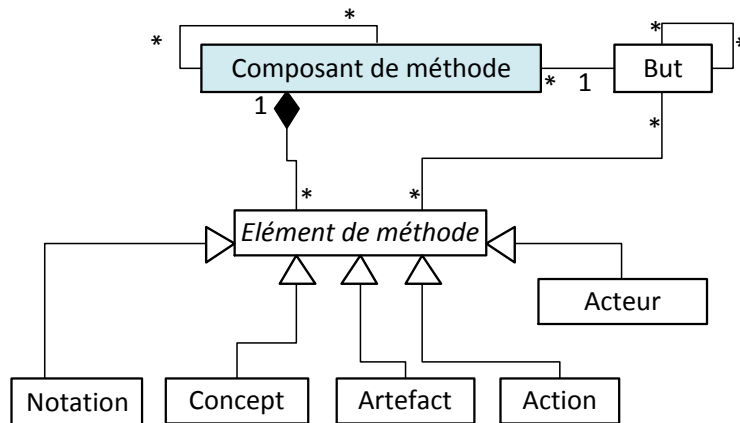


Figure 2.5 : Structure d'un composant de méthode adapté de (Karlsson, 2004) et (Karlsson, 2004)

Une action est définie dans cette approche comme un processus regroupant un ensemble de tâches à réaliser durant l'application de la méthode. Les actions sont les composants centraux d'un modèle de processus (Wistrand, 2004). Les artefacts, quant à eux, sont les entrées et les résultats des actions (modèle de produit). L'application des actions pour la production d'artefact est encadrée par le concept d'acteur définissant le rôle des ressources permettant l'exécution des processus et la manipulation des produits.

Les actions et les artefacts sont décrits à l'aide de modèles. Les concepts peuvent être définis comme les constituants du paradigme utilisé pour la représentation des actions et des artefacts. Les notations sont liées aux concepts dans le sens où elles sont des éléments de modélisation représentant des concepts d'un paradigme pour la création de modèles.

La réutilisation des composants de méthode est facilitée par l'utilisation de but rattaché à chaque composant. Un but peut également être affiné en sous-buts qui seront reliés aux éléments de méthode entrant dans la composition du composant de méthode.

2.3.3.2. L'approche associée aux composants de méthode

L'approche des composants de méthode proposée par Karlsson et ses collègues (Wistrand, 2004), s'organise en deux vues autour du concept de composant de méthode : la vue externe et la vue interne. La vue externe concerne la décomposition d'une méthode d'ingénierie des SI dirigée par les buts en un ensemble de composants de méthode. La vue interne concerne, quant à elle, la structure du composant en lui-même. Pour chacun des composants issus de la décomposition d'une méthode, un ensemble de buts lui est rattaché. Ceux sont soit des buts d'accomplissement, lorsqu'ils contribuent au but global du projet, soit des buts de contradiction, lorsqu'ils identifient des incompatibilités pour ce composant.

Concernant le stockage des composants de méthode, Karlsson (Karlsson, 2004) propose de les rassembler dans des bases de méthode. Aucune autre recommandation sur les portefeuilles méthodologiques n'est indiquée dans cette approche.

La sélection des composants de méthode se fait, quant à elle, par des requêtes basées sur les buts supportés par des composants. Les utilisateurs formulent leurs requêtes sous la forme d'un but correspondant aux exigences de la situation spécifique de leur projet. Une correspondance entre le but de la requête et les buts réalisés par les composants stockés dans la base est ensuite effectuée tout en prenant en compte les buts de contradiction portés par les composants de méthode. En revanche, cette approche ne décrit pas comment est effectuée la correspondance entre la situation du projet et les composants sélectionnés dans la base.

Après avoir passé l'étape de sélection des composants, une nouvelle méthode d'ingénierie des SI peut être créée à partir des composants extraits de la base de méthode. Cette approche propose ainsi de créer de nouvelles méthodes en utilisant les mécanismes d'assemblage, d'extension ou de restriction sur les composants de méthode (Wistrand, 2004). En d'autres termes, il est possible d'adapter une méthode à une situation de projet particulière en associant des composants de méthode entre eux pour former une méthode (assemblage), en fusionnant des composants entre eux (extension), ou alors en réduisant le contenu de composants avant de les intégrer (restriction).

2.3.3.3. Application du cadre de référence aux composants de méthode

Axe objectif

Dans (Wistrand, 2004), il n'est pas fait mention d'un support outillé pour l'application de l'approche qu'ils proposent. Cette approche associée aux composants de méthode a donc une *interactivité* manuelle. De même, aucun standard n'est proposé dans la définition de l'approche et de ses composants, nous pouvons donc qualifier l'attribut interopérabilité comme interne pour cet axe du cadre de référence.

Axe usage

Concernant la *couverture des composants*, cette approche prend en compte les aspects suivants : paradigme, produit, processus et organisationnel. Nous retrouvons les aspects produit, processus et organisationnel respectivement dans les concepts : artefact, action et acteur. Les concepts d'artefacts et d'actions sont décrits dans un composant à l'aide de modèles définis dans des paradigmes donnés (l'aspect paradigme). Comme énoncé à la sous-section "axe objectif" de la section 2.3.3.3, il n'est pas fait mention d'*outil* pour cette approche.

Axe processus

Le processus de l'approche publiée dans (Wistrand, 2004), propose une étape de *décomposition* des méthodes existantes en composants de méthode dirigée les buts. Une fois définis, ces composants sont stockés dans une base de méthode. Ils sont ensuite *sélectionnés* lorsqu'un utilisateur effectue une requête en fonction des buts (identifiés pour chacun des composants). Une fois les composants

récupérés de la base, l'approche ne fait pas mention d'une étape de vérification de la *correspondance* entre les composants sélectionnés et la situation spécifique du projet de l'utilisateur. Les composants sélectionnés sont alors assemblés (assemblage par association ou assemblage par intégration) pour *construire* une nouvelle méthode adaptée au contexte du projet. Il est également possible d'effectuer des extensions ou des réductions sur les composants sélectionnés.

Axe sujet

Le *niveau de représentation* utilisé dans cette approche est le niveau intentionnel par l'identification des buts associés aux composants de méthode, mais aussi le niveau structurel avec la définition des composants de méthode. Quant aux niveaux d'abstraction, les niveaux modèle et métamodèle sont utilisés dans l'approche. Le *formalisme* de description de ces composants est essentiellement conceptuel. Lors de la construction d'une nouvelle méthode par un assemblage de composants, des liens d'association sont établis entre les composants, mais la nouvelle méthode n'est pas considérée elle-même comme un composant. Nous pouvons en conclure que cette approche n'inclut pas de principe de *récurtivité*.

2.3.4 Les composants de méthode OPF (Open Process Framework)

L'approche Open Process Framework est désormais conforme à la norme internationale ISO/IEC 24744 (ISO/IEC 24744, 2010) et a été proposée dans (Henderson-Sellers, 1996), (Firesmith, 2002), (Gonzalez-Perez, 2007), (OPF, 2012). Dans cette approche, chaque composant de méthode est généré à partir d'un métamodèle sous-jacent (Gonzalez-Perez, 2007). Une autre particularité de cette approche est que tous les éléments composant une méthode peuvent être définis comme des composants de méthode OPF.

2.3.4.1. Structure d'un composant de méthode OPF

Un composant de méthode OPF peut être défini comme un concept abstrait permettant de dériver ou généraliser les différents éléments constituant une méthode (OPF, 2012). Comme présenté à la figure 2.6, les concepts d'unité de travail (un processus) et de produit de travail (un produit) sont des composants de méthode. Une unité de travail est définie dans OPF comme un composant de méthode modélisant une opération applicable par un producteur. Une unité de travail est également constituée d'un ensemble d'activités ou de tâches (OPF, 2012).

Un composant de méthode de produit de travail est un modèle de ce qui est produit, utilisé, modifié ou détruit par un ou plusieurs producteurs durant l'application d'un composant de type unité de travail (OPF, 2012).

La modélisation d'un produit de travail est assurée dans un langage supportant un paradigme donné. Un langage est constitué d'un vocabulaire correspondant à l'ensemble de représentations des différents

concepts du paradigme utilisé. Il est également composé d'une grammaire regroupant les différentes règles de modélisation de ce langage.

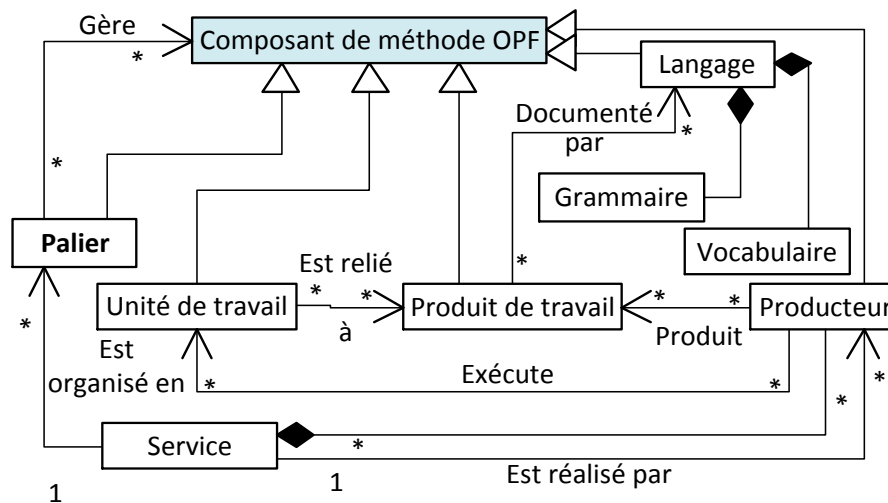


Figure 2.6 : Structure d'un composant de méthode OPF adapté de (OPF, 2012)

L'exécutant d'une unité de travail (processus) est modélisé dans OPF par le concept de producteur (humain ou machine). L'exécution de l'unité de travail par le producteur peut être directe ou indirecte (OPF, 2012). L'ensemble des producteurs collaborent et interviennent dans la réalisation d'un service. Ce dernier est relatif à un projet devant être entrepris par l'application d'un ensemble d'unités de travail afin de réaliser ou maintenir un ensemble de produits de travail (OPF, 2012). Les différents services sont quant à eux organisés temporellement en paliers garantissant une cohérence temporelle et de performance dans l'application d'une méthode.

2.3.4.2. L'approche associée aux composants de méthode OPF

L'approche OPF propose un métamodèle standard pour les méthodes de développement des systèmes d'information. Ainsi, en suivant cette approche, les méthodes existantes sont décomposées en éléments par l'utilisation des principes d'héritage et d'instanciation comme le préconise le paradigme orienté objet. Chaque élément de méthode est ainsi représenté à l'aide d'un concept appelé "clabject". Ce dernier est une entité ayant une double facette d'objet (instance) et de classe (Gonzalez-Perez, 2007).

Une fois créés, les composants de méthode OPF sont stockés dans d'une base de méthode nommée "base de produit de travail". Cette base est définie comme une construction abstraite et est actuellement implémentée sous la forme d'une base de données ou d'un entrepôt de connaissances (Gonzalez-Perez, 2007). Les composants OPF sont indexés dans cette base méthodologique par le concept de service. Ce concept permet d'exprimer la collaboration des acteurs à un projet pour la réalisation du processus afin de construire les produits. Lors de l'étape de recherche dans la base, l'utilisateur formule une requête selon son contexte de projet afin de sélectionner des composants de méthode adaptés à sa situation. Ensuite, l'adéquation des composants à la situation à un instant donné

est effectuée en tenant compte par exemple du but global du projet à atteindre, ainsi que des ressources disponibles au niveau de producteur. Cependant, la question de la temporalité d'exécution des composants par rapport à la situation de l'utilisateur n'est pas traitée (Gonzalez-Perez, 2007).

L'ensemble des composants sélectionnés dans le "base de produit de travail" est assemblé de manière agile par des associations entre les composants au fur et à mesure du projet. Contrairement aux autres approches, OPF est moins focalisé sur l'aspect processus des méthodes. En effet cette approche prône un cycle de vie agile pour son application. En d'autres termes, une nouvelle méthode situationnelle n'est pas créée entièrement avant le démarrage du projet mais est en revanche construite dynamiquement au fur et à mesure du déroulement et de l'évolution du projet. Quelques composants sont sélectionnés et appliqués au début du projet. Les composants de méthode suivants sont sélectionnés au dernier moment durant le déroulement du projet afin de prendre en compte les exigences réelles du projet et non celles qui sont prévus ou envisagés. Cette démarche permet d'adapter de manière continue durant le projet et de prendre en compte les situations réelles du projet plutôt que la situation initiale estimée du projet

2.3.4.3. Application du cadre de référence aux composants de méthode OPF

Axe objectif

L'approche OPF ne proposant qu'une base de données pour le stockage des composants en tant qu'outil, l'application de la démarche est dite manuelle. La valeur *manuelle* est affectée à la facette *interactivité* de cet axe. Le paradigme utilisé dans cette approche est le paradigme objet, celui-ci permet d'avoir une certaine ouverture vers les autres composants orientés objets issus d'autres approches. La facette *interopérabilité* peut donc être considérée comme externe dans le même environnement (orientation objet).

Axe usage

Comme nous l'avons énoncé précédemment, le seul *outil* proposé dans cette approche est la base de composants. Par conséquent, seul le stockage de composant est outillé dans cette approche. De plus, les méthodes créées à partir de cette approche ne sont pas supportées par un outil, l'approche ne couvre pas l'aspect outil des méthodes. En revanche, l'approche OPF propose une couverture des aspects produit et processus par le biais des concepts de produit de travail et d'unité de travail. De plus, cette approche couvre l'aspect paradigme grâce au concept de langage qui est le langage de modélisation utilisé pour représenter les modèles de produit et de processus dans un paradigme donné.

Axe processus

Comme nous l'avons vu à la section 2.3.4.2, les méthodes d'ingénierie des SI existantes sont *décomposées* dans cette approche en composants de méthode OPF en utilisant les principes d'héritage

et d'instanciation. Ils sont ensuite stockés dans une base de méthode et indexés par le contexte de projet (le concept OPF de service). La *sélection* des composants de cette base se fait à partir d'une requête dirigée par le contexte du projet. L'adéquation des composants au projet est évaluée en fonction des buts du projet et des acteurs devant participer à celui-ci. Enfin, les composants sont assemblés de manière dynamique en prenant en compte le chevauchement des composants entre eux. La *construction* d'une méthode situationnelle qui peut être effectuée de manière incrémentale ou non par un assemblage par association ou par un assemblage par intégration en supportant le chevauchement entre les composants.

Axe sujet

Etant basée sur la norme ISO/IEC 24744 (ISO/IEC 24744, 2010), qui est un standard de méta-modélisation des méthodes d'ingénierie des SI, l'approche OPF traite à la fois le *niveau structurel de représentation* avec l'utilisation de "clabjects" pour la définition de chaque élément de méthode et le niveau intentionnel par l'identification du but supporté par le composant dans son contexte d'utilisation. Le *formalisme* de cette approche est technique puisque le formalisme objet peut être facilement opérationnalisé.

Enfin, les *niveaux d'abstraction* utilisés pour les modèles dans cette approche sont : schéma, modèle et métamodèle. Concernant la facette récursivité de cet axe, l'approche ne propose pas à travers son modèle de composant de composer des composants OPF avec d'autres composants de méthode. Par conséquent la récursivité n'est pas supportée.

2.3.5 Les services méthodologiques Service Oriented Meta-Method (SO2M)

Proposée par Cauvet et Guzélian, cette approche est présentée comme un framework orienté service pour la construction de méthodes de développement des SI (Guzélian, 2007a), (Guzélian, 2007b). La particularité de cette approche est l'application du paradigme des AOS à l'IMS et ainsi de prendre en compte l'aspect outil qui était absent des approches d'IMS.

2.3.5.1. Structure d'un service méthodologique SO2M

La figure 2.7 nous montre qu'un service méthodologique SO2M est décomposé en trois parties abstraites encapsulant les autres concepts: une partie identification, une partie processus et une partie ressource. Un service méthodologique peut être défini à différents niveaux de granularité (c'est-à-dire qu'un service méthodologique peut être atomique ou composite et constitué de plusieurs autres services méthodologiques).

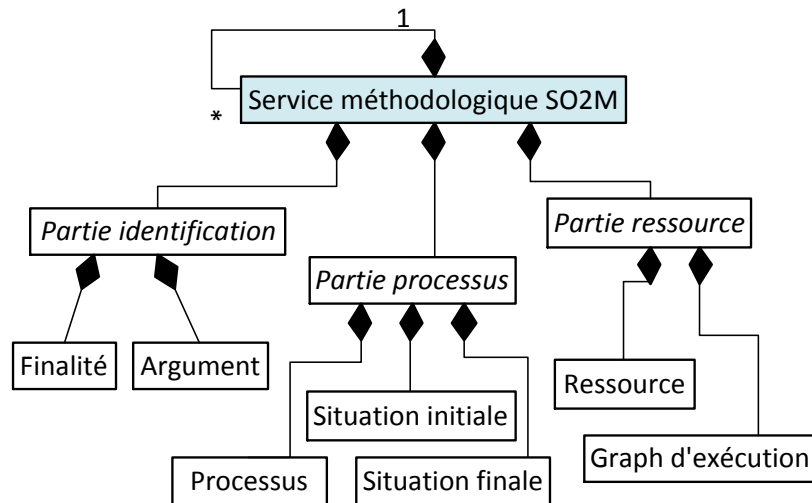


Figure 2.7 : Structure d'un service méthodologique SO2M adapté de (Guzélian, 2007a) et (Guzélian, 2007b)

La partie processus regroupe les constituants d'un processus méthodologique pour la réalisation du but du composant. Le concept de processus de cette partie correspond au processus de la méthode en lui-même. Il peut être atomique et être un processus opérationnel réalisant un but simple ou un sous but. Un processus peut également être composite afin de supporter un but complexe et être décomposé en plusieurs autres processus simples ou composites, reliés à l'aide de points de contrôle (séquentiel ou parallèle). Le processus peut être décisionnel. Dans ce cas particulier, le processus est décomposé en fonction du but complexe qu'il supporte. La décomposition du but peut introduire plusieurs alternatives possibles d'atteindre ce but. Chacune de ces alternatives est décrite par un processus composant. Enfin, un processus peut être défini comme simple et il est supporté par un service opérationnel qui est sélectionné de manière différée au moment de sa concrétisation (Guzélian, 2007a). Les deux autres concepts de cette partie processus sont la caractérisation de la situation initiale et de la situation finale du processus.

La partie ressource de ce type de composant de méthode décrit la solution offerte par le service méthodologique SO2M. Le graphe d'exécution est un diagramme d'activité dont les activités doivent être substituées par des services opérationnels au moment de l'exécution. Les ressources correspondent aux produits utilisés ou créés lors de l'application du processus (Guzélian, 2007a).

La partie identification définit le but du composant de méthode. Le concept "finalité" de cette partie du composant correspond au contexte du projet dans lequel le service méthodologique SO2M est le plus approprié, ainsi qu'au but et au moyen de le réaliser. La finalité a pour objectif de caractériser le problème résolu par le composant de méthode. Le concept "argument", quant à lui, recense les avantages et les inconvénients liés à l'application du service méthodologique SO2M.

2.3.5.2. L'approche associée aux services méthodologiques SO2M

L'approche SO2M est organisée autour d'une base de services méthodologiques dans laquelle sont stockés les services méthodologiques SO2M. Dans cette approche, un service méthodologique est défini comme une surcouche d'un fragment de processus. Cela est rendu possible par l'incorporation d'une couche de description représentant toutes les informations nécessaires à son usage et un mécanisme de composition permettant d'assembler les services dynamiquement. L'approche est basée sur des fragments de processus existants et ne propose malheureusement pas de directives ni de bonnes pratiques pour la décomposition de méthode en services méthodologiques.

Une dimension sémantique est donnée aux services méthodologiques SO2M en enrichissant leur descripteur avec des données définies dans les quatre types d'ontologies de domaine de l'approche avant de les stocker dans une base de données de composant de méthodes.

Ces quatre ontologies de produits, de processus, d'acteurs et de buts, sont reliées au domaine des SI. Elles sont également exploitées lors de la recherche dans la base de services méthodologique. Afin de récupérer des services de la base, les utilisateurs formulent leur requête sous forme de but à l'aide de l'ontologie de but de SO2M. La correspondance entre les requêtes et les descripteurs de services se fait par un calcul de similarité sémantique par rapport à l'ontologie de but.

La correspondance entre le contexte de projet et les services méthodologique SO2M extraits de la base de méthode se réalise à l'aide des ontologies de domaines (produit, processus, but et acteurs) de l'approche.

Une fois la sélection des composants effectuée, une nouvelle méthode situationnelle peut être composée par un assemblage sans chevauchement des composants. La composition des services méthodologiques SO2M est représentée par un arbre ou graphe d'exécution de services. Ces services sont reliés entre eux dans ce graphe par des liens de type séquentiel ou parallèle.

2.3.5.3. Application du cadre de référence aux services méthodologiques SO2M

Axe objectif

L'approche SO2M est basée sur le paradigme des AOS et les services méthodologiques doivent être conformes aux standards développés pour ce type d'architecture. Les composants de cette approche peuvent donc être utilisés conjointement à d'autres composants conformes à ces standards. Les services méthodologiques disposent donc d'une *interopérabilité* externe avec d'autres composants de méthode d'environnements d'exécution hétérogènes. L'*interactivité* de la démarche est assistée dans le cadre de l'exécution des processus méthodologiques implémentés par des services.

Axe usage

Au niveau de l'axe "usage" du cadre de référence, cette approche couvre les aspects paradigme, produit et processus d'une méthode. L'aspect outil n'est pas réellement couvert par l'approche car aucune implémentation de service méthodologique n'est proposée. Le concept de service manipulé dans l'approche reste un concept abstrait. En revanche, concernant les *outils* pour l'application de l'approche, un prototype pour le stockage de descripteurs de service, la recherche et la composition de graphe d'exécution (construction) est proposé.

Axe processus

Comme nous l'avons énoncé à la section 2.3.5.2, l'approche SO2M ne propose pas de guide pour la *décomposition* de méthode. Elle se base sur des fragments de processus déjà existants. La *technique pour construire* de nouvelles méthodes est une composition par assemblage par association sans chevauchement de services méthodologiques. La *sélection* des services de la base de méthode est effectuée par similarité sémantique entre la requête et les descripteurs en exploitant une ontologie de buts. Concernant les services méthodologiques ainsi récupérés, la *correspondance* avec la situation contextuelle du projet est réalisée par le biais des quatre ontologies définies dans l'approche.

Axe sujet

Un descripteur de service méthodologique SO2M est défini sous forme de but, cela correspond *au niveau intentionnel de représentation* du composant de méthode. La représentation structurelle du service méthodologique est assurée par la description des trois parties : identification, processus et ressource du composant. Enfin le niveau de représentation opérationnel est supporté par le graphe d'exécution opérationnel d'un service méthodologique. Cette dernière représentation opérationnelle du composant impacte le *formalisme* utilisé pour cette représentation en lui donnant une orientation plutôt technique. Cela n'empêche pas les services méthodologiques d'être définis aux *niveaux d'abstraction* : modèle, métamodèle et méta-métamodèle. Ce service méthodologique peut être également défini récursivement car il peut être lui-même composé d'autres services méthodologiques.

2.4. Comparaison des approches

Le cadre de référence définit dans ce chapitre a été appliqué aux cinq approches d'IMS les plus connues dans la communauté de l'IM. Le tableau présenté à la table 2.1 reprend les valeurs affectées à chaque facette du cadre de référence pour les cinq approches d'IMS. Dans ce tableau, les facettes sont regroupées selon un axe de comparaison pour une approche d'ingénierie des méthodes donnée. Ce cadre permet d'identifier les caractéristiques discriminantes des approches et de les comparer de manière synthétique et structurée.

Pour l'axe Objectif, toutes les approches fournissent une interopérabilité interne de leur composant de méthode. Seules les approches OPF et SO2M proposent une utilisation de standards dans la définition de composants afin de permettre une interopérabilité externe. Cette interopérabilité est toutefois limitée dans l'approche OPF car elle est basée sur le paradigme orienté objet, ce qui posera inévitablement des problèmes de sérialisation pour des composants définis selon un autre paradigme que celui-ci. De la même manière, les approches sont pour la plupart appliquées de manière manuelle par les utilisateurs. Seules les approches basées sur les fragments de méthode et les services méthodologiques SO2M proposent des outils pour assister les utilisateurs concernant des parties de la démarche. Cela montre un manque de support et d'automatisation dans l'application des approches d'IMS. Nous pouvons également déduire que, malgré les efforts de standardisation proposés par les approches OPF et SO2M, il existe toujours un manque de standards pour les approches d'IMS et d'interopérabilité de leur composant de méthode.

De plus, ce manque de support outillé dans l'axe Usage du cadre de référence. En effet, la majeure partie des approches n'offrent un support outillé que pour le stockage et la recherche des composants de méthode. Les approches SO2M et fragments de méthode proposent également un support outillé lors de la construction des méthodes par assemblage. Néanmoins, les approches ne proposent une assistance outillée que sur certaines parties de la démarche. En effet, l'aspect outil n'est couvert par aucune approche. En d'autres termes, cela signifie que la méthode situationnelle construite par l'application de ces approches n'est, elle-même, pas outillée.

Concernant l'axe Processus du cadre de référence, nous remarquons que les approches ont quasiment toutes les mêmes types d'activité dans leur démarche (réingénierie des méthodes existantes, recherche et sélection de composant réutilisable, correspondance entre le composant sélectionné et la situation du projet ou le besoin de l'utilisateur et enfin construction de la nouvelle méthode situationnelle), comme l'illustre la figure 1.1 du Chapitre 1. En revanche, les moyens utilisés pour accomplir ces activités varient fortement d'une approche à l'autre. Si nous prenons le cas de la décomposition de méthodes existantes en composants pour la phase de réingénierie, il y a quasiment autant de solutions différentes que d'approches d'IMS. Il en va de même pour la sélection des composants stockés dans une base de méthode, ainsi que pour la mise en correspondance des composants sélectionnés à la situation spécifique d'un projet. Nous retrouvons ce constat pour l'activité de construction de nouvelles méthodes situationnelles. Si toutes les approches présentées dans ce chapitre proposent des techniques d'assemblage de composant, certaines proposent en plus des techniques comme l'extension ou la réduction de composants existants. De cette diversité, nous pouvons souligner un manque de standardisation du concept de composant de méthode, et un manque de convergence sur les facteurs qui influence l'application des processus. À l'heure actuelle, la vision processus des approches manque

d'homogénéité et, comme le montre l'approche OPF, l'évolution du contexte du projet durant celui-ci nécessite une constante évolution des méthodes en cours de projet.

L'axe Sujet de ce cadre de référence met en évidence le fait que les approches d'IMS existantes utilisent des niveaux de représentation différents pour la description de leur composant de méthode et leurs processus. Si toutes les approches présentées dans ce chapitre utilisent un niveau de représentation structurel pour leurs composants, quelques-unes définissent les composants de méthode à un niveau opérationnel. Ces dernières, en général, adoptent un formalisme plutôt technique contrairement aux précédentes leur préférant un formalisme plus conceptuel. Toutefois, l'utilisation d'un niveau intentionnel de représentation des composants de méthode sous forme de buts ou d'intentions s'est généralisée pour favoriser la recherche et la sélection des composants. Nous pouvons également constater qu'il existe plusieurs niveaux de granularité utilisés dans les approches pour les composants. Certaines approches considèrent le concept de composant de méthode comme un élément atomique d'une méthodologie tandis que d'autres considèrent qu'un composant de méthode est composite; c'est-à-dire pouvant être constitué d'autres composants. Le contenu des composants de méthode est majoritairement orienté sur les aspects processus et produit des méthodes, néanmoins certaines approches les plus récentes proposent de couvrir l'aspect organisationnel par l'introduction d'une orientation supplémentaire basée sur les acteurs (producteurs). Sur l'échelle des niveaux d'abstraction utilisés pour la représentation des composants de méthode, la couverture de l'ensemble des niveaux diverge d'une approche à l'autre. Cela risque d'engendrer des problèmes pour l'interopérabilité entre les différentes approches lors du passage des produits d'un composant à l'autre.

Si les cinq approches présentées dans ce chapitre ont le même objectif de créer des méthodes adaptées aux situations spécifiques des projets, elles divergent dans les moyens qu'elles proposent pour la mise en œuvre de cet objectif. Comme nous l'avons présenté dans cette section, malgré cette diversité d'action, les approches d'IMS existantes souffrent de leur manque d'adoption dans le monde de l'entreprise.

| Axe | Attributs | Fragment de méthode | Chunk | Composant de méthode | Composant de méthode OPF | Service méthodologique SO2M |
|-----------|------------------------------------|--|---|--|--|--|
| Objectif | Interopérabilité | interne | interne | interne | externe dans le même environnement | externe dans des environnements différents |
| | Interactivité | assistée | manuelle | manuelle | manuelle | assistée |
| Usage | Couverture des composants | paradigme, processus, produit | paradigme, processus, produit | paradigme, processus, produit, organisationnel | paradigme, processus, produit, organisationnel | paradigme, processus, produit |
| | Outil et implémentation | stockage, manipulation, recherche, construction | stockage, recherche | non spécifié | stockage | stockage, recherche, construction |
| Processus | Principe de décomposition | non spécifié | par intention | par but | par héritage et instanciation | non spécifié |
| | Principe de recherche et sélection | requête sur descripteur | mesure de similarité | requête sur le but | requête sur le contexte projet | similarité sémantique |
| | Correspondance avec la situation | scénario | carte des exigences | non spécifié | par but et acteur | par des ontologies de buts, acteurs, processus et produits |
| | Technique de construction | assemblage par association, assemblage par intégration, incrémentale | assemblage par association, assemblage par intégration, extension | assemblage par association, assemblage par intégration, extension, réduction | assemblage par association, assemblage par intégration, incrémentale | assemblage par association |
| Sujet | Niveau de représentation | structurel, opérationnel | intentionnel, structurel | intentionnel, structurel | intentionnel, structurel | intentionnel, structurel, opérationnel |
| | Récursivité | oui | oui | non | non | oui |
| | Niveau d'abstraction | modèle, métamodèle | modèle, métamodèle | modèle, métamodèle | instance, modèle, métamodèle | modèle, métamodèle, méta-métamodèle |
| | Formalisme | conceptuel | conceptuel | conceptuel | technique | technique |

Table 2.1 : Analyse comparative des cinq approches d'IMS sélectionnées

2.5. Conclusion

Ce chapitre reprend les cinq approches d'IMS les plus connues. Nous les avons étudiées sous différents angles au travers d'un cadre de référence. Les quatre axes que forment le cadre de référence nous ont permis d'aborder ces approches existantes selon deux éléments : les composants de méthode et la démarche d'ingénierie des méthodes proposée par l'approche. Cela nous a permis d'étudier les aspects internes des composants : leur structure et leur formalisation, ainsi que leurs usages potentiels (interopérabilité, aspects de la méthode couverts, assistance outillée de la démarche).

L'application de notre cadre de référence sur les cinq approches présentées dans ce chapitre nous a permis de mettre en évidence certaines lacunes qui se résument selon les cinq points suivants :

- Malgré les efforts récents de standardisation dans la communauté de l'IM, il existe toujours un manque de standardisation dans les approches d'IMS qui entraîne d'une part des difficultés d'interopérabilité entre les composants de méthode et, d'autre part, induit un manque d'homogénéité dans la manière de réaliser les activités de la démarche d'ingénierie des méthodes.
- Si la communauté de l'IM s'accorde sur une définition commune d'une méthode en cinq aspects, les approches existantes couvrent au minimum les aspects produit et processus mais avec des manières différentes. En effet, le composant de type chunk encapsule les aspects produit et processus alors que les fragments de méthode considèrent les aspects produit et processus comme des composants de méthode distincts.
- Comme nous pouvons le constater dans le tableau comparatif des approches, l'aspect outil n'est quasiment pas traité par les approches d'IMS ce qui implique que la méthode adaptée à la situation n'est pas outillée.
- De plus, il reste encore des efforts à fournir pour l'assistance et l'automatisation de la démarche des approches d'IMS, qui sont, pour la plupart du temps, appliquées de manière manuelle.
- La plupart des approches d'IMS proposent une adaptation des méthodes en correspondance avec la situation d'un projet au début de celui-ci sans tenir compte des variations possibles au niveau du contexte du projet en cours de réalisation.
- Enfin, la prise en compte dans les approches d'IMS de tous les niveaux d'abstraction (instance, modèle, métamodèle et méta-métamodèle) est importante car elle sert de support à l'interopérabilité des produits transmis entre les composants lors de leur opérationnalisation.

Ces constats viennent conforter la problématique de cette thèse : *Comment améliorer l'usage des approches d'ingénierie des méthodes situationnelles dans les entreprises ?* Dans le sens où l'usage des méthodes situationnelles créées et les approches d'IMS elles-mêmes ne sont pas pris en compte en pratique.

Pour tenter de répondre à ces constats, nous allons, dans le chapitre suivant présenter la démarche de notre approche.

Chapitre 3. L'approche MaaS

3.1. Introduction

Après une analyse de la littérature et une étude sur le terrain auprès d'entreprises françaises, nous pouvons constater que les approches d'IMS existantes présentent des lacunes freinant leur implantation et leur usage dans le milieu professionnel. De plus, il existe dans les entreprises des besoins méthodologiques qui ne trouvent pas satisfaction dans les méthodes d'ingénierie des SI ou les approches d'IMS actuelles. Nous présentons la solution que nous avons choisie pour répondre à ces problèmes.

Notre démarche consiste à appliquer le paradigme orienté service (AOS) dans le domaine de l'ingénierie des méthodes situationnelles pour proposer une approche d'IMS orientée sur l'usage des composants de méthode. Afin d'atteindre cet objectif nous proposons :

- (i) l'élaboration d'une base commune pour le domaine de l'IMS en définissant une ontologie des métamodèles de composant de méthode (OMCM),
- (ii) la définition d'un modèle de composant de méthode exécutable : le service méthodologique permettant de s'affranchir du problème d'interopérabilité des composants,
- (iii) la construction d'un processus d'IMS orienté sur l'usage et l'aspect outil d'une méthode,
- (iv) la création d'un modèle de méthode intégrant le concept de variabilité sous la forme d'une ligne de méthode pour réaliser la configuration de méthode,
- (v) l'usage de niveaux d'application de l'approche permettra une meilleure intégration de celle-ci en entreprise.

Ce chapitre s'organise en trois parties. Le constat de recherche et la problématique introduits au chapitre 1 ainsi que les hypothèses formulées pour y répondre sont rappelés à la section 3.2. Un aperçu détaillé de la solution répondant à la problématique et la mise en place des hypothèses en pratique est le propos de la section 3.3. Le positionnement de cette approche par rapport au cadre de l'état de l'art est présenté à la section 3.4. Une enquête auprès de professionnels a permis de confirmer la pertinence de certaines questions de recherche soulevées au chapitre 1 de cette thèse (section 3.5). Enfin, nous concluons ce chapitre sur les contributions de notre proposition (section 3.6).

3.2. Rappel du constat, de la problématique et de l'objectif de recherche

3.2.1 Constat

Le domaine de l'ingénierie des méthodes situationnelles est apparu il y a une quinzaine d'années en réponse au constat suivant : les méthodes d'ingénierie des SI ont une définition monolithique avec un

objectif d'être universelles, mais en pratique cela n'est pas applicable à cause de la diversité trop importante des contextes de projet et des domaines d'application de ces méthodes. Les approches d'IMS ayant des difficultés à s'implanter dans les entreprises, ce constat est toujours d'actualité. En effet, comme nous avons pu le constater lors d'une enquête auprès d'entreprises (voir section 3.5 de ce chapitre), les entreprises ont besoin d'adapter les méthodes d'ingénierie qu'ils utilisent. Cela est d'autant plus vrai que les situations de projet sont amenées à changer au cours du temps. L'évolution actuelle des marchés impose aux entreprises des évolutions de plus en plus rapides de leur SI. Il en résulte une pression plus grande sur les projets de développement avec des modifications du contexte de projet survenant au cours de la réalisation de ces derniers. Notre enquête a également permis de constater que, lors de l'adaptation de leurs méthodes à la situation de leurs projets, les entreprises préféreraient configurer les méthodes qu'ils utilisent plutôt que construire une nouvelle méthode par assemblage de composants réutilisables pour répondre à un contexte de projet particulier.

Comme le chapitre 2 sur l'état de l'art a pu l'expliquer, les approches d'IMS les plus populaires présentent actuellement des lacunes freinant leur intégration et leur application dans les entreprises. En effet, une interopérabilité entre les composants de méthode issus de différentes approches n'est pas supportée. Cela provient d'un manque d'utilisation de standards dans les approches d'IMS. Ajoutez à cela que l'ensemble des niveaux d'abstraction utilisés dans la description des composants (instance, modèle, métamodèle, méta-métamodèle) n'est pas couvert uniformément par l'ensemble des approches. Ceci peut provoquer des difficultés d'interopérabilité, notamment pour le transfert de produits entre différents composants. Ces données peuvent être définies sur un ou plusieurs niveaux d'abstraction qui ne sont peut-être pas pris en compte par l'un des composants, il ne sera donc pas applicable à ces données. De plus, les méthodes créées par une approche d'IMS manquent de support outillé. Ce manque d'outils est aussi présent dans l'application de la démarche d'IMS ou dans son automatisation. Enfin, l'utilisation de standards pour l'implémentation de ces outils est primordiale car ils doivent pouvoir fonctionner ensemble.

L'usage des approches d'IMS est donc freiné au sein des entreprises sans support outillé adapté. L'interopérabilité entre les composants issus d'approches différentes est difficile à atteindre sans application de standard dans la définition de ces composants. De plus, les composants de méthode sont le moyen idéal pour le stockage et le partage de la connaissance méthodologique dans une base. Enfin, sans processus approprié à l'adaptation de méthode au contexte de projet, les entreprises apportent de nos jours des modifications à leurs méthodes de manière ad hoc lorsqu'ils en éprouvent le besoin. Il existe donc un besoin d'avoir une approche formalisée pour aider à adapter des méthodes aux situations particulières de projet d'entreprise : une approche d'IMS. Le processus d'une approche d'IMS nécessite cependant une introduction progressive dans le milieu de l'entreprise pour former les utilisateurs étape par étape afin d'augmenter son acceptation.

3.2.2 Problématique

Les difficultés pour les approches d'IMS à s'implanter dans les entreprises sont dues en partie au manque d'interopérabilité des supports outillés et les besoins actuels des entreprises à adapter leurs méthodes à leurs projets nous a permis d'énoncer la problématique suivante :

- *Comment améliorer l'usage de l'ingénierie des méthodes situationnelles dans les entreprises ?*

Cette problématique reflète le besoin (i) de modéliser une base commune pour l'ingénierie des méthodes; (ii) de définir un composant de méthode exécutable et interopérable ; (iii) de fournir une modélisation des méthodes incorporant le concept de variabilité ; (iv) mais aussi de formaliser la démarche d'une approche d'IMS afin que son intégration dans l'entreprise puisse être progressive.

3.2.3 Objectif de recherche

Dans l'optique d'améliorer l'usage des approches d'IMS, notre proposition est construite sur l'objectif de recherche suivant :

- *Objectif de recherche : repenser l'ingénierie des méthodes situationnelles au travers du paradigme de services (au sens large) et l'application de l'architecture orientée service (AOS).*

La notion de *service* considérée dans cet objectif est à prendre au sens large comme étant un service centré utilisateur permettant d'aider celui-ci dans la réalisation de l'une de ses intentions. L'*architecture orientée service* représente ici la démarche d'ingénierie ou de réingénierie d'un système en une collection de services de haut niveau, autonomes, décrits, publiés, recherchés, à partir d'un annuaire de services, exécutables et composables. Les plus-values offertes par ces deux éléments sont, d'une part, de promouvoir une architecture flexible et dynamique et, d'autre part, de dissocier les actions de publication, de recherche, d'invocation à distance, de composition de services et de l'implémentation de services par l'exploitation de descripteurs standardisés intégrant un niveau conceptuel et un niveau technique. En effet, la publication, la recherche, l'invocation à distance et la composition de service n'exploitent pas l'implémentation du service mais se basent uniquement sur le descripteur de celui-ci. L'interopérabilité de cette solution orientée service réside dans la proposition d'une plateforme standard pour publier, sélectionner, rechercher, invoquer à distance et composer des services implémentés selon des environnements de développement différents.

Nous proposons de transposer les plus-values de cette solution orientée service dans le domaine de l'ingénierie des méthodes situationnelles.

Cette transposition est envisagée, d'une part, par l'adjonction de l'aspect *outil* au composant de méthode (issu d'approches d'IMS différentes) par le biais d'un service applicatif (service logiciel

gérant l'interaction avec l'utilisateur). Ce couplage permet de fournir un support outillé faisant défaut dans les approches d'IMS orientées composants. Ce couple identifié sous le concept de *service méthodologique* est considéré comme une boîte noire exposée à l'aide d'un descripteur standard.

La solution orientée service transposée à l'ingénierie des méthodes consiste, d'autre part, à construire un annuaire de services méthodologiques permettant d'accueillir des composants de méthode de différentes approches d'IMS avec les fonctionnalités génériques pour les rechercher, les sélectionner, les publier et les invoquer à distance.

L'introduction de la variabilité dans la définition d'une méthode nous paraît un point important pour encourager les entreprises à mieux adapter les méthodes à leur contexte et à plus largement utiliser des approches d'IMS. C'est pour cette raison que nous considérons plutôt la configuration de méthode que la construction d'une nouvelle méthode par assemblage ou composition de composants (ou de services) comme dans beaucoup d'approches d'IMS. Cette stratégie implique la définition d'un modèle de configuration de méthode proposant d'utiliser les composants dans une plateforme d'exécution du processus méthodologique, basée elle aussi sur le paradigme des AOS, tout en incorporant la variabilité nécessaire à l'adaptation de la méthode aux situations changeantes des projets. Enfin, la démarche de l'approche proposée est prévue pour une intégration progressive dans le département méthode des entreprises.

3.3. Aperçu de la solution

Comme nous avons pu le voir précédemment, la solution proposée s'organise autour de l'amélioration de l'usage de l'ingénierie des méthodes situationnelles dans les entreprises. Cela passe par l'application du paradigme des AOS au domaine de l'IMS pour la construction d'une nouvelle approche. Cette proposition s'articule autour des trois éléments suivants :

- Une ontologie de métamodèles de composant de méthode servant de base à la définition du descripteur standard de services,
- Un modèle de services adapté à l'ingénierie des méthodes situationnelles permettant la configuration de méthode et,
- La démarche d'ingénierie ou de réingénierie des composants de méthode en services et leur utilisation pour la configuration de méthodes.

Chacun de ces trois éléments fait l'objet d'une sous-section.

3.3.1 Une ontologie de métamodèles de composant de méthode

Le premier élément de la solution proposée consiste à définir une ontologie de métamodèles de composant de méthode commune aux approches d'IMS actuelles. Pour établir cette ontologie, une

étude détaillée des concepts constituant la structure interne des composants de méthode a été menée en exploitant le socle commun à toutes ces approches : la définition d'une méthode selon Seligman (Seligmann, 1989) et Rolland (Plihon, 1995). L'enjeu de ce travail est d'identifier les similitudes et les singularités des approches afin de pouvoir proposer une ontologie permettant de référencer tous les concepts des définitions existantes de composant de méthode.

La construction de cette ontologie s'est effectuée en quatre étapes commençant par une analyse de la littérature des différents types de composants existants. À partir de cette analyse, l'étape suivante a été d'identifier chaque concept utilisé par ces derniers en fonction des cinq aspects d'une méthode : *paradigme, produit, processus, organisationnel et outil*. Enfin, l'étape d'abstraction sur les concepts utilisés dans chacun des composants a permis de regrouper les concepts communs et de noter les différences entre les approches. Puis, par conceptualisation, nous avons bâti cette ontologie de métamodèles de composant de méthode.

La figure suivante présente la démarche d'utilisation de cette ontologie dans l'approche MaaS.

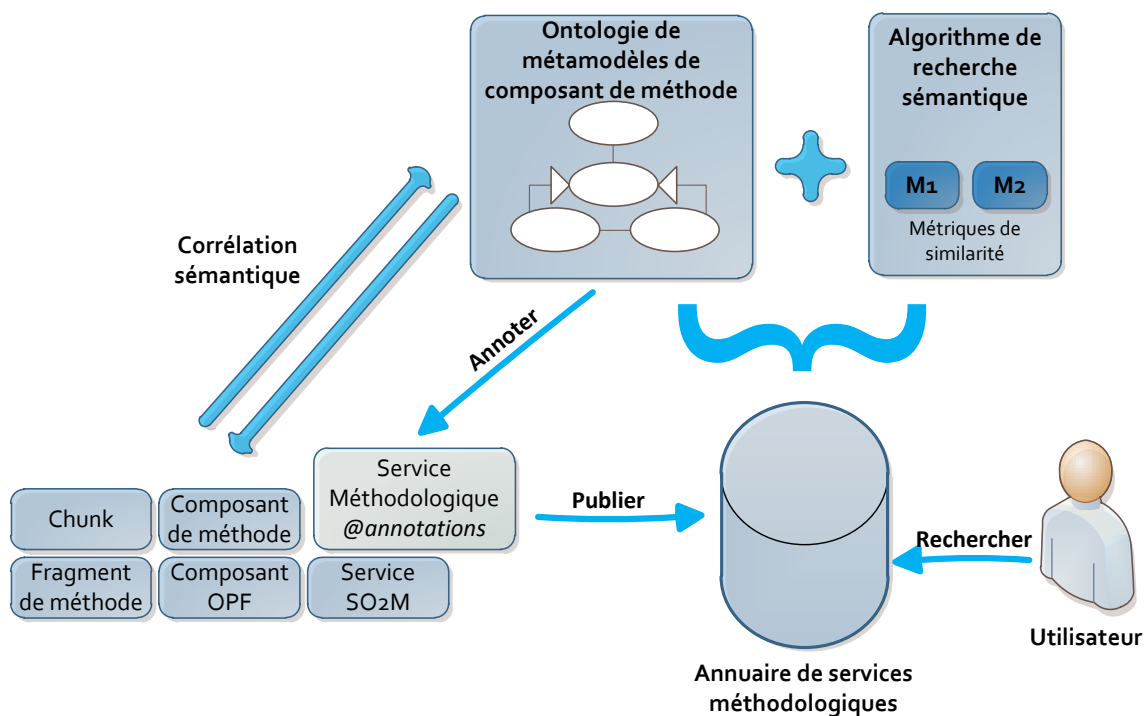


Figure 3.1 : Utilisation de l'ontologie OMCM dans l'approche MaaS

Comme nous pouvons l'observer à la figure 3.1, la sémantique des métamodèles de composant de méthode issus des approches existantes se définit par rapport à l'ontologie OMCM. Il en va de même du composant nommé service méthodologique issu de l'approche MaaS, proposée dans cette thèse. Dans le cadre d'une architecture AOS appliquée à l'ingénierie des méthodes situationnelles, cette ontologie est utilisée pour structurer le descripteur standard attaché au service (ou composant). Ainsi, l'utilisateur peut formuler sa requête de recherche (ses besoins) en fonction de cette ontologie et ainsi

être indépendante de l'approche dont sont issus les composants de méthode. Néanmoins, la publication d'un composant (ou service) de méthode dans l'annuaire nécessite l'application des règles de correspondance entre sa structure interne et l'ontologie afin d'en dériver son descripteur. Le mécanisme choisi pour implémenter ce descripteur est celui des annotations sémantiques. Les services méthodologiques sont ainsi annotés en fonction de la sémantique représentée par les concepts de l'ontologie OMCM. Déployés dans l'annuaire de services, les algorithmes de recherche sémantique effectuent la correspondance entre la requête de l'utilisateur et les annotations des composants de méthode. Ces annotations doivent être définies par le fournisseur à la publication du composant (ou service) et les règles de correspondances à appliquer dépendent de l'approche utilisée pour construire le composant.

Il est à observer que l'ontologie OMCM a une importance qui va au delà de l'approche MaaS puisqu'elle permet, d'une part, d'effectuer les correspondances entre les composants issus d'approches différentes, et d'autre part, elle constitue une base commune pour le domaine des approches d'IMS. Pour avoir une implémentation souple au niveau de l'annuaire, le mécanisme d'annotation et la stratégie d'implémentation de l'annuaire proposée par Yassa (Chabeb, 2008), (Chabeb, 2010) ont été adoptés pour l'annuaire de services méthodologiques. En effet, cette stratégie propose de décrire la structure des annotations dans une ontologie appelée "ontologie technique" couplée à des ontologies de domaines pour affecter des valeurs aux différentes annotations. Cette stratégie permet ainsi de fournir un mécanisme d'annotations, de stockage et de recherche applicable sur n'importe quel type de composant de méthode.

3.3.2 La définition d'un composant de méthode standard et de méthodes configurables

Le deuxième élément de notre solution consiste en la définition d'un métamodèle de service méthodologique atomique (SMA) et d'un métamodèle de méthode configurable appelée ligne de méthode (LM) qui est considérée comme un service méthodologique complexe.

L'objectif de cette approche est d'orienter la conception de celle-ci vers l'usage des composants en pratique. Pour ce faire, les concepts des métamodèles de service méthodologique (SMA et LM) sont, eux-mêmes, positionnés par rapport à l'ontologie OMCM. L'application du paradigme des AOS sur cette corrélation sémantique permet de produire un composant opérationnel et interopérable appelé service méthodologique (SM). Ce type de composant est destiné à apporter le support outillé manquant actuellement aux méthodes créées à partir des approches d'IMS existantes. En effet, un SM contient l'implémentation de son processus méthodologique sous la forme d'un service. La méthode étant vue comme une agrégation de services méthodologiques, l'outil final de celle-ci est également dérivé par agrégation des implémentations des SM qui la composent.

La figure ci-dessous présente la démarche d'utilisation de ces métamodèles de service méthodologique atomique et de ligne de méthode dans l'approche MaaS.

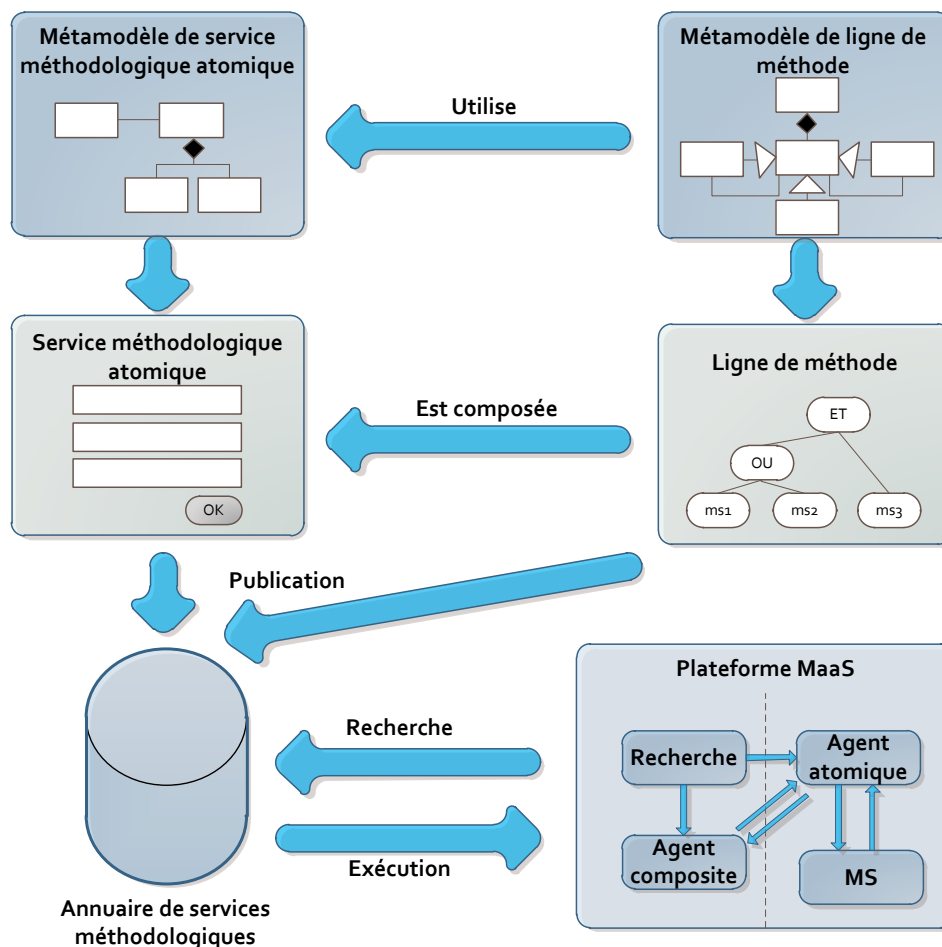


Figure 3.2 : Utilisation des métamodèles de service méthodologique et de ligne de méthode dans MaaS

D'après l'enquête d'entreprises présentée à la section 3.5 de ce chapitre et une étude plus ancienne (Karlsson, 2004), les entreprises préfèrent configurer les méthodes qu'ils utilisent pour s'adapter aux besoins du contexte de leur projet plutôt que de créer de nouvelles méthodes. En allant dans ce sens, nous proposons un modèle de ligne de méthode inspiré de (Kornysheva, 2011), (Deneckere, 2011) et des modèles de variabilité de (Bennasri, 2005) et de (Kaabi, 2007). Cette proposition est construite par conceptualisation et transposition du concept de variabilité au domaine de l'IMS. Les concepts de point de variation et variante sont intégrés dans le processus d'une méthode composée de services méthodologiques. L'opérationnalisation d'une agrégation de SM nécessite un mécanisme de passage des données d'un SM à l'autre. Le mécanisme adopté est celui de BPEL pour l'orchestration de web services : des fonctions de transformation des produits, générés et consommés par les différents services méthodologiques, sont définies au niveau de l'agrégation (ligne de méthode) et utilisées pour exécuter celle-ci. En d'autres termes, une ligne de méthode est une composition de SM incluant des

points de variation. Elle représente une méthode dans son ensemble et peut être configurée à n'importe quel moment du cycle de vie du projet.

Comme présenté à la figure 3.2, l'implémentation d'une ligne de méthode est constituée d'une composition de service méthodologique atomique (simple ou complexe). Une fois assemblée, une ligne est stockée dans l'annuaire de composants de la même manière qu'un SMA. Les services méthodologiques atomiques et les lignes de méthode sont recherchés par les utilisateurs depuis la plateforme MaaS à partir d'un annuaire de SM. Les services atomiques récupérés sont invoqués et exécutés directement sur la plate-forme par le module exécutant les services atomiques. Les lignes de méthode sont, quant à elles, interprétées par le module exécutant les services composites. Ce dernier interagit avec l'utilisateur pour intégrer les choix de celui-ci lors de la configuration de la méthode. Le module exécutant les LM invoque alors les services choisis par les utilisateurs et effectue les correspondances de données entre les entrées et les sorties de produit des services.

Ces métamodèles de service méthodologique atomique et de ligne de méthode apportent l'aspect *outil* aux approches d'IMS. L'introduction de la variabilité dans les méthodes adaptées aux situations de projet permet également d'apporter la flexibilité nécessaire aux environnements de projets actuels où le contexte subit des évolutions rapides.

3.3.3 Le processus de l'approche MaaS

Le dernier élément de la solution MaaS est de proposer un processus d'IMS se focalisant sur l'usage des méthodes situationnelles produites.

Ce processus illustré à la figure 3.3 se découpe en plusieurs activités. La première d'entre elles est une standardisation des composants de méthode, existants dans d'autres approches, en services méthodologiques. Cette activité permet de construire un outil sous forme d'un service implémentant le processus du composant de méthode qui lui est associé. Afin de garantir l'interopérabilité entre les composants, les SM sont créés à partir de standards existants ou d'extensions compatibles avec une architecture AOS.

À partir d'un ensemble de services méthodologiques, un ingénieur méthode peut adapter des méthodes existantes ou partir de zéro pour construire des méthodes configurables. Une ligne de méthode ainsi créée correspond aux exigences particulières de l'entreprise en termes de projets. Cette activité consiste en l'analyse de situation contextuelle du projet d'une entreprise, puis d'une modélisation d'une ligne de méthode basée sur l'adaptation des méthodes déjà utilisées pour qu'elles soient en correspondance avec leur contexte d'utilisation. Cela passe par l'introduction de points de choix (point de variation) dans les méthodes existantes dont l'objectif est de proposer aux utilisateurs de méthodes plusieurs alternatives méthodologiques possibles concurrentes en fonction de leurs besoins (variantes).

Un fournisseur de méthode produit ainsi des SMA et des lignes de méthode qui sont, par la suite, publiés dans un annuaire de services méthodologiques pour être réutilisés. Cette étape permet de rendre les services et les lignes de méthode accessibles à un maximum d'utilisateurs et ainsi favoriser leur réutilisation.

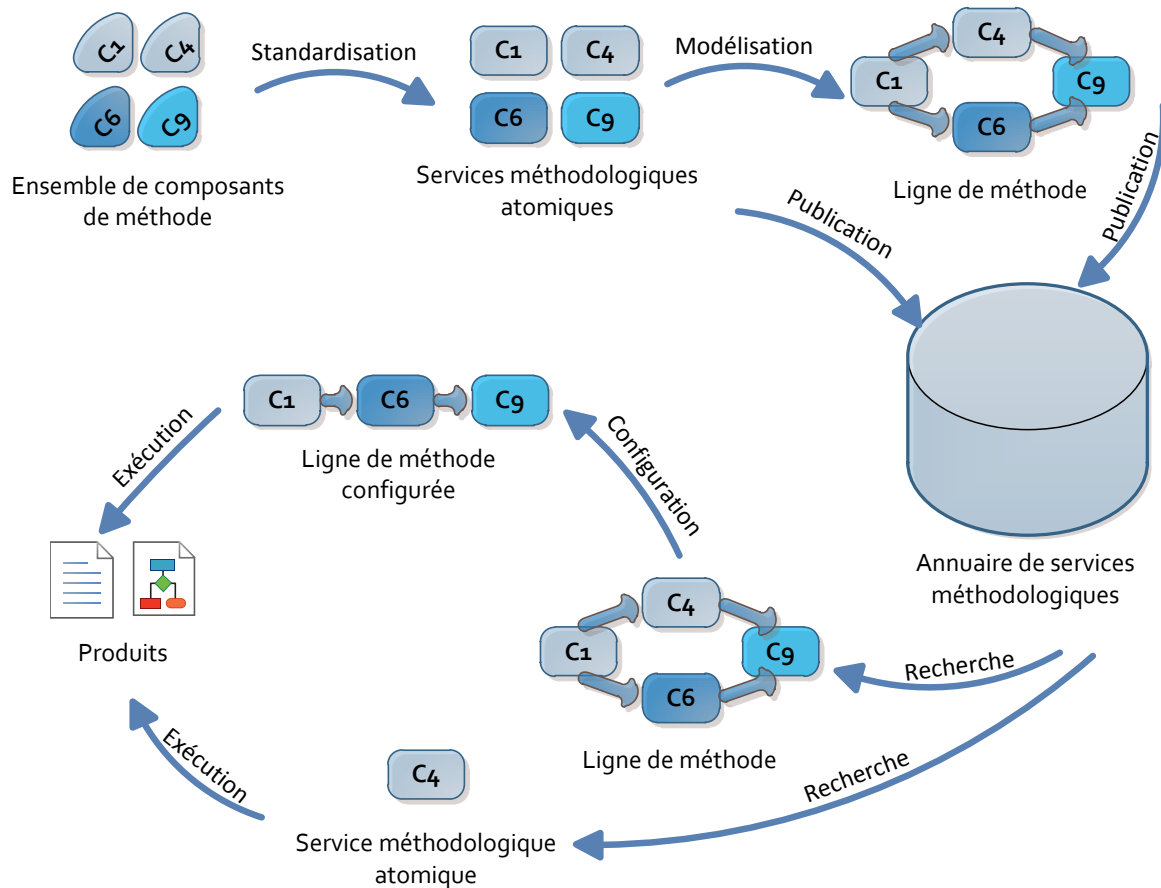


Figure 3.3 : Processus de l'approche MaaS

Une fois stockés dans l'annuaire, les services méthodologiques et les lignes de méthodes sont exposés et peuvent être recherchés par les utilisateurs en fonction de leurs besoins. Les utilisateurs (chefs de projet ou développeurs) construisent leurs requêtes sous la forme d'une interface de service annotée conformément à l'ontologie OMCM (section 3.3.1). La correspondance entre la requête et les composants de la base est effectuée par l'algorithme de recherche sémantique de l'annuaire.

Les services méthodologiques atomiques ainsi récupérés, qu'ils soient simples ou complexes peuvent alors être directement invoqués et exécutés sur la plateforme afin de réaliser ou de transformer les produits.

Les lignes de méthode sélectionnées dans l'annuaire passent en premier lieu par une étape de configuration avant d'être exécutées sur la plateforme. La configuration d'une ligne peut être réalisée à n'importe quel moment au cours de la durée de vie d'un projet. Cela signifie que le choix entre

plusieurs alternatives méthodologiques peut être effectué au cours de l'exécution de la ligne de méthode.

Dans l'objectif de répondre au besoin d'intégration progressive des démarches d'IM dans les entreprises, l'approche MaaS préconise de découper les activités de ce processus en niveaux de maturité conformément à l'approche de qualité processus CMMI (CMMI, 2006). En d'autres termes, nous proposons d'appliquer l'approche CMMI au processus des approches d'IMS. L'adaptation des niveaux de maturité CMMI au domaine de l'ingénierie des méthodes permet d'augmenter progressivement la qualité des processus des approches existantes et de fournir aux entreprises un cadre d'évaluation et d'amélioration de leur gestion des méthodes pour le développement des SI.

Cette approche permet la prise en compte de l'aspect outil des méthodes situationnelles faisant défaut dans les approches d'IMS actuelles. L'application du paradigme des AOS dans le cadre de l'approche MaaS s'effectue en étendant les standards orientés service afin de les adapter aux exigences du domaine de l'ingénierie des méthodes. Cela apporte une interopérabilité entre les services méthodologiques pour la construction de lignes de méthode. Le processus de l'approche MaaS peut s'appliquer dans l'entreprise de manière graduelle conformément aux niveaux de maturité. Ceci permet une intégration progressive des pratiques de l'ingénierie des méthodes dans l'entreprise.

3.4. Positionnement de l'approche par rapport au cadre de référence

L'approche proposée dans cette thèse peut se positionner par rapport au cadre de référence par les facettes suivantes :

| vue | facette | L'approche Maas |
|-----------|----------------------------------|---|
| Objectifs | Interopérabilité | externe dans des environnements différents |
| | Interactivité | assistée |
| Usage | Couverture des composants | paradigme, processus, produit, outil |
| | Outils-Implémentation | Stockage, recherche, exécution et configuration |
| Processus | Principe de décomposition | Non pertinent |
| | Recherche et Sélection | Selon le métamodèle de composant proposé |
| | Correspondance avec la situation | Ontologie de domaine (Produit) |
| | Technique de construction | Configuration (variabilité) |
| Produit | Niveau | Intentionnel, structurel et technique |
| | Récurtivité | Non |
| | Niveau d'abstraction | Modèle, Métamodèle et méta- |

| | | |
|--|------------|------------|
| | | métamodèle |
| | Formalisme | conceptuel |

Table 3.1 : Positionnement de l'approche MaaS

L'orientation service de cette approche a été envisagée pour, d'une part, proposer une interopérabilité avec les approches d'IMS existantes (*externe dans des environnements différents*) et, d'autre part, pour prendre en charge l'aspect *outil* en plus des quatre aspects traditionnellement traités dans les composants de méthode (*paradigme, produit, processus et organisationnel*). De plus, comme dans l'orientation service, l'approche propose des outils pour rechercher, publier, exécuter et configurer des services méthodologiques. L'étude des vues produit/processus de l'approche permet de dégager quelques caractéristiques principales :

- Le service méthodologique intègre les visions intentionnelle, structurelle et opérationnelle d'un composant de méthode. En effet, la partie intentionnelle est gardée pour améliorer sa réutilisation, et le service agrège en un concept unique la partie conceptuelle du composant de méthode et son support outillé (partie opérationnelle) sous forme de service applicatif.
- La variabilité est un élément de construction important, c'est pour cette raison que nous privilégions la configuration de méthode plutôt que la composition de services comme dans SO2M. Ceci nous a amené à développer un modèle de ligne de méthode au dessus du concept de service méthodologique afin de spécifier cette variabilité. C'est pour cette raison que l'approche a la facette récursivité évaluée à *Non*.
- Les niveaux d'abstraction des modèles gérés dans l'approche vont du modèle en passant par le métamodèle mais aussi le méta-métamodèle si l'on veut pouvoir offrir de l'interopérabilité au niveau opérationnel.
- La recherche et la sélection de services exploitent la partie conceptuelle grâce à l'ontologie OMCM qui peut servir de base de représentation commune à l'ensemble des approches d'IMS mais aussi la partie opérationnelle à l'aide du descripteur opérationnel (WSDL dans notre cas).
- Le choix d'implémentation de l'annuaire de services méthodologiques et du descripteur sémantique du service méthodologique conduit à introduire des annotations sémantiques au descripteur opérationnel de service standard (WSDL). Celles-ci doivent être conformes à l'ontologie technique (ontologie OMCM) et aux ontologies de domaine (produits). L'objectif de cette implémentation est, d'une part, d'exploiter au maximum les standards de l'orientation service (WSDL, UDDI, SOAP) pour garantir l'interopérabilité opérationnelle et, d'autre part, de prendre en compte le plus possible la sémantique particulière du domaine de l'IMS dans la description, la recherche et la publication de services afin de proposer des outils dédiés à l'IMS.

3.5. Enquête auprès de professionnels

La solution proposée dans cette thèse a pour objectif l'amélioration de l'usage de l'ingénierie des méthodes situationnelles en entreprise. Cette solution s'articule autour de (i) l'élaboration ontologie générique des métamodèles de composant, (ii) la définition d'un métamodèle de composant de méthode exécutable et interopérable, (iii) l'identification de l'architecture d'une plateforme pour l'exécution des services méthodologiques, (iv) la création d'un métamodèle de méthode intégrant le concept de variabilité sous la forme d'une ligne de méthode, (v) la construction d'un processus d'IMS orienté sur l'aspect outil d'une méthode pour que son intégration en entreprise soit progressive.

- (i) Le besoin de modélisation d'un socle commun pour le domaine de l'IMS prend sa source dans l'analyse des différentes approches établie dans (Agerfalk, 2007). Ce constat de diversité des approches est renforcé par la comparaison des approches d'IMS dans l'état de l'art présenté au chapitre 2 de cette thèse. En effet, les approches ont une couverture partielle des cinq aspects d'une méthode et elles utilisent des niveaux d'abstraction différents dans la modélisation de leurs composants.
- (ii) Les manques de standardisation et d'interopérabilité identifiés dans cet état de l'art renforcent le besoin de définir un métamodèle de service méthodologique interopérable.
- (iii) Les lacunes de support outillé pour les composants et méthodes adaptées impliquent le besoin de définir un métamodèle de service méthodologique exécutable ainsi que l'architecture de sa plateforme d'exécution.
- (iv) Concernant la création d'un métamodèle de ligne de méthode, le besoin d'une intégration du concept de variabilité dans les méthodes est lié au manque de configuration à la volée des méthodes situationnelles que nous avons identifié dans l'état de l'art. Nous avons néanmoins effectué une enquête auprès des industriels afin de conforter l'hypothèse que les entreprises préfèrent configurer leurs méthodes plutôt que d'en construire de nouvelles par assemblage.
- (v) De la même manière, une confrontation avec les industriels était nécessaire afin de vérifier l'hypothèse qu'une démarche de gestion de méthodes ou une démarche d'IMS est intégrée progressivement dans une entreprise pour en faciliter son usage.

Nous avons formulé les deux hypothèses suivantes :

- H1 : *Une entreprise préférera configurer les méthodes plutôt que composer de nouvelles méthodes par assemblage.*
- H2 : *Lorsqu'une démarche d'IM est intégrée à une entreprise celle-ci est réalisée de manière progressive.*

Pour ces deux hypothèses, nous avons identifié un ensemble de critères d'évaluation. Puis à partir de ces critères, un questionnaire a été établi dans l'objectif d'interroger des personnes ayant un rapport quotidien avec les méthodes de développement de SI afin de valider ou d'invalidier ces hypothèses.

3.5.1 Critères d'évaluation de l'hypothèse H1

Cette hypothèse s'inscrit dans le thème de l'adaptation des méthodes. Nous souhaitons mesurer la préférence des personnes interrogées entre deux critères concernant le type d'adaptation effectuée sur leurs méthodes et la flexibilité des méthodes dans leurs adaptations. Nous voulons vérifier que les entreprises effectuent des modifications partielles sur les méthodes afin de les adapter à leur contexte de projets. En d'autres termes, qu'elles ont besoin d'une flexibilité partielle de leurs méthodes, ce qui est propice à une configuration de méthodes. Ces deux critères nous permettent d'établir si les entreprises préfèrent configurer des méthodes ou créer de nouvelles méthodes par composition.

Les critères d'évaluation de H1 sont :

- Degré de flexibilité nécessaire pour des méthodes : *valeur {aucune, partielle, totale}*.
- Type d'adaptation : *valeur {configuration, composition}*.

Les questions permettant de déterminer la valeur de ces critères sont :

- Degré de flexibilité nécessaire pour des méthodes : *pour un projet particulier lorsque que vous avez besoin d'adapter ou de configurer une méthode, quelle est l'ampleur des modifications à apporter à la méthode ?*
- Type d'adaptation : *lorsque vous devez adapter une méthode pour un projet, quelle adaptation effectuez-vous ?*

3.5.2 Critères d'évaluation de l'hypothèse H2

L'objectif de cette hypothèse est d'établir un lien entre la présence d'activités de gestion des méthodes dans une entreprise et la manière dont elles ont été intégrées aux processus de celle-ci. Nous évaluons la présence d'activités de gestion ou d'ingénierie de méthodes dans une entreprise par la présence : d'activités de mesure de la qualité de l'application des méthodes, d'activités de support des méthodes, d'activités de maintenance des méthodes, de service de gestion des méthodes. Nous souhaitons ensuite mesurer si l'intégration de ces processus de gestion des méthodes est faite de manière progressive. Si l'hypothèse H2 est vérifiée, la stratégie des entreprises en matière d'intégration de leur gestion des méthodes est l'intégration progressive. L'usage d'une démarche d'ingénierie des méthodes sera donc favorisé par une intégration progressive aux processus de l'entreprise.

Les critères d'évaluation de H2 sont :

- Activités de gestion (ou d'ingénierie) des méthodes dans l'entreprise : *booléen*.

- Intégration progressive de l'ingénierie des méthodes : *booléen*.

Les questions permettant de déterminer la valeur de ces critères sont :

- Activités de gestion (ou d'ingénierie) des méthodes dans l'entreprise : *disposez-vous d'un service ou département responsable des méthodes de l'entreprise ? Lorsque vous appliquez une méthode dans l'entreprise : avez-vous un processus vérifiant la qualité de l'application de cette méthode ? ou fournissez-vous aux utilisateurs des activités de support pour faciliter son application ? ou révisiez-vous la méthode en lui intégrant les adaptations fréquemment réalisées ?*
- Intégration progressive de l'ingénierie des méthodes : *est-ce que les activités de gestion des méthodes de la question précédente ont été introduites dans l'entreprise de manière progressive (par pallier) ?*

3.5.3 Résultats

Dans l'objectif de vérifier ces hypothèses, 142 personnes ont été interrogées ayant un profil familier avec l'utilisation des méthodes de développement de SI en entreprises. En d'autres termes, personnes travaillant en entreprise et ayant un des profils suivants : développeur, chef de projet ou ingénieur méthode. Ces personnes ont été interrogées par le biais d'un questionnaire en ligne que nous leurs avons envoyé par mail. Nous avons utilisé un site internet comme média de recueil des résultats de ce questionnaire.

Les résultats de cette étude sont présentés pour chaque hypothèse sous la forme d'un tableau croisé. Pour chaque tableau le compte des réponses à la première question de l'hypothèse est ventilé en fonction des réponses à la deuxième question pour cette même hypothèse. Par exemple, la participation d'une personne ayant répondu "configuration" et "partielle" respectivement aux questions sur le type des adaptations et le degré de flexibilité de l'hypothèse H1 est comptabilisée dans la cellule à l'intersection de la colonne "configuration" et de la ligne "partielle". Après avoir comptabilisé l'ensemble des résultats, cette cellule contient le nombre de personnes ayant répondu "configuration" et "partielle" aux questions de l'hypothèse H1.

Le tableau suivant présente les résultats du questionnaire pour l'hypothèse H1 : *préférence des entreprises pour la configuration des méthodes plutôt que l'assemblage de nouvelles méthodes*.

| | | Type des adaptations | | | Total |
|----------------------|--------------|----------------------|-------------|---------------|-------|
| | | Sans réponse | Composition | Configuration | |
| Degré de flexibilité | Sans réponse | 48 | 1 | 1 | 50 |

| | | | | | |
|--------------|-----------|----|----|-----------|-----|
| | Aucune | 5 | 0 | 4 | 9 |
| | Partielle | 4 | 11 | 65 | 80 |
| | Totale | 0 | 0 | 3 | 3 |
| Total | | 57 | 12 | 73 | 142 |

Figure 3.5 : Tableau croisé des résultats du questionnaire pour l'hypothèse H1

Comme présenté à la figure 3.5, sur 142 personnes 83 se sont exprimées sur les deux questions. Sur ces 83 personnes seules 11 préfèrent composer de nouvelles méthodes contre 72 personnes préférant configurer les méthodes qu'elles utilisent. De plus; parmi ces 72 personnes, 65 ont besoin d'effectuer des adaptations partielles de leurs méthodes à leur contexte de projet. Nous pouvons donc distinguer une tendance à préférer la configuration de méthodes pour une adaptation partielle de leurs méthodes.

Les résultats du questionnaire pour l'hypothèse H2 d'intégration progressive d'une démarche d'IMS dans une entreprise sont présentés sur le tableau suivant.

| | | Intégration progressive de l'ingénierie des méthodes | | | Total |
|---|--------------|--|-----|-----------|-------|
| | | Sans réponse | non | oui | |
| Activités de gestion (ou d'ingénierie) des méthodes dans l'entreprise | Sans réponse | 52 | 0 | 0 | 52 |
| | non | 1 | 19 | 5 | 25 |
| | oui | 10 | 8 | 47 | 65 |
| Total | | 63 | 27 | 52 | 142 |

Figure 3.6 : Tableau croisé des résultats du questionnaire pour l'hypothèse H2

Sur les 142 personnes interrogées 79 ont répondu aux deux questions et 47 d'entre elles ont déclaré que les activités de gestion et d'ingénierie des méthodes ont été intégrées progressivement dans leur entreprise. Seules 8 personnes ont déclaré que les activités de gestion et d'ingénierie des méthodes n'ont pas été intégrées progressivement et 24 personnes déclarent ne pas avoir de telles activités dans leur entreprise. A partir de ces résultats, nous pouvons également constater que les méthodes sont devenues des processus à part entière dans l'entreprise. Une tendance d'intégration progressive des démarches de gestion ou d'ingénierie des méthodes dans les processus d'entreprise se dégage de ces résultats.

Les résultats du questionnaire montrent une tendance des entreprises, d'une part, sur la préférence à la configuration des méthodes et, d'autre part, sur l'intégration progressive des démarches d'IMS. Ceux-

ci sont en accord avec nos deux hypothèses et viennent conforter les aspects de la configuration de méthode et de l'intégration progressive des démarches d'IMS de notre solution.

3.6. Conclusions

Dans ce chapitre nous avons présenté un aperçu de la solution proposée dans cette thèse. L'objectif auquel doit répondre celle-ci est d'améliorer l'usage de l'ingénierie des méthodes situationnelles dans les entreprises.

Pour ce faire, nous avons d'abord introduit une ontologie OMCM pour les cinq approches d'IMS les plus connues du domaine. Elle constitue une base commune pour la définition de composant de méthode et elle permet d'établir des correspondances entre les composants de différentes approches.

Le deuxième élément de la solution MaaS consiste en la définition d'un composant en correspondance avec le métamodèle de composant de méthode appelé service méthodologique. Ce nouveau type de composant met l'accent sur l'aspect outil des méthodes qui n'était jusqu'alors pas pris en compte dans la plupart des approches d'IMS existantes. Cette nouvelle dimension des services méthodologiques permet d'apporter les outils nécessaires à l'application des méthodes de développement des SI en pratique afin de favoriser leur usage. La définition d'un métamodèle de ligne de méthode permet également d'apporter la flexibilité nécessaire aux méthodes pour s'adapter à la diversité des situations de projet de développement actuel. Enfin, la plateforme d'exécution des SMA et des lignes de méthode apporte un support outillé pour l'application de notre nouvelle approche d'IMS appelée MaaS.

Nous avons ensuite présenté le processus de cette approche MaaS. Ce processus est orienté sur son usage et sur l'usage des méthodes situationnelles dans les entreprises. L'utilisation du paradigme des AOS au domaine de l'IMS pour la construction des services méthodologiques, des lignes de méthode, et du processus de l'approche MaaS favorise l'utilisation de standards dans leurs implémentations et permet une interopérabilité entre les différents services et outils. Une application de l'approche CMMI sur ce processus d'approche d'IMS permet également d'intégrer progressivement le domaine de l'IMS dans les entreprises.

Les chapitres suivants de cette thèse détaillent le contenu de cette solution et sont organisés de la façon suivante :

- Le chapitre 4 présente l'ontologie OMCM et son utilisation dans l'annuaire de services méthodologiques.
- Le chapitre 5 définit les modèles de services méthodologiques et de ligne de méthode. Il présente également la décomposition du processus de l'approche MaaS en niveaux de maturité pour son intégration dans une entreprise.

- Le chapitre 6 présente un cas pratique d'application de l'approche MaaS sur les méthodes agile "Extreme Programming" (Wells, 2009), "Dynamic System Development Method" (DSDM, 2009) et "SCRUM" (Schwaber, 2011) communément utilisées dans les entreprises.
- Le chapitre 7 décrit l'architecture de la plateforme de l'approche MaaS et de sa réalisation dans deux environnements techniques : les portails web utilisés conjointement aux services WSRP (OASIS/WSRP, 2008) et l'environnement de développement intégré Eclipse avec les composants logiciels OSGI (OSGI, 2011).

Chapitre 4. Ontologie de métamodèles de composant de méthode

4.1. Introduction

Ce chapitre présente une ontologie de métamodèles de composant de méthode (OMCM) pouvant servir de socle commun aux approches d'IMS.

Les méthodes de développement des SI sont le sujet de recherche du domaine de l'ingénierie des méthodes. Un des intérêts principaux de ce domaine est la décomposition de ces méthodes en parties modulaires dans un objectif de réutilisation, d'optimisation des méthodes mais également afin de garantir leur flexibilité et leur adaptabilité (Agerfalk, 2007). C'est de cet intérêt que sont apparus les travaux du domaine et de la communauté de l'ingénierie des méthodes situationnelles. Ce nouveau domaine est depuis une quinzaine d'années un domaine de recherche prolifique où plusieurs approches d'IMS sont en compétition. Chacune de ces approches a proposé une définition, publié et utilisé sa propre vision de ce qu'est une méthode modulaire. Cette diversité nous montre la richesse des travaux de recherche qui ont été accomplis mais elle implique également une série de lacunes déjà énoncées dans le chapitre 2 de cette thèse. Ces lacunes engendrent des confusions non souhaitées pour les utilisateurs n'étant pas experts du domaine de l'IM (Agerfalk, 2007), dues au manque de standards et d'interopérabilité pour les composants de méthode, ainsi qu'au manque de support outillé (Deneckere, 2008).

Aujourd'hui, obtenir une base commune pour le domaine de l'IMS est un problème d'actualité pour les chercheurs de cette communauté (Agerfalk, 2007). D'après les auteurs de la table ronde effectuée en 2007 autour de ce sujet, et dont les résultats ont été présentés dans (Agerfalk, 2007), il existe deux possibilités de réponses à ce sujet : (1) les différences entre les approches d'IMS sont mineures et un accord sur une définition commune d'un concept de bloc modulaire de méthode peut être atteint, ou alors (2) la diversité impliquée par les approches d'IMS est utile car elles desservent toutes des objectifs différents et, dans ce cas, toutes sont nécessaires.

Nous proposons dans ce chapitre d'atteindre cet objectif de base commune par la définition d'une ontologie des métamodèles de composant de méthode. Cette ontologie consiste en une base sémantique commune pour les approches d'IMS. Une ontologie a déjà été proposée dans la communauté de l'IM (Mirbel, 2007) avec l'objectif d'atteindre une base sémantique pour établir les concepts centraux nécessaires à la description de la connaissance méthodologique. Cependant, cette ontologie est de trop haut niveau pour permettre d'établir une définition commune d'un composant de méthode utilisable par les approches d'IMS. Une autre base sémantique pour l'ingénierie des méthodes a été proposée dans (Niknafs, 2007) mais les concepts utilisés dans leur ontologie sont trop restrictifs

pour couvrir la diversité des blocs de construction modulaire de méthode ainsi que les différents niveaux introduits par l'ensemble des approches d'IMS. Cela est dû au fait que l'objectif principal de leurs travaux est d'améliorer l'approche proposée dans (Ralyté, 2003) et non de fournir une base commune pour le domaine de l'IMS.

L'ontologie OMCM est construite de manière ascendante à partir de la sémantique des concepts utilisés dans les métamodèles de composant de méthode existants. En effet, les approches d'IMS (Brinkkemper, 1998), (Ralyté, 2001b) et (Ralyté, 2001c), (Wistrand, 2004), (Firesmith, 2002) et (Guzélian, 2007a) promeuvent différents blocs de construction modulaire de méthode. Cependant, ces approches s'accordent toutes sur une définition commune du concept de méthode (Seligmann, 1989), (Plihon, 1995). Dans cette définition, une méthode est décrite par les cinq aspects inter reliés suivants : paradigme, produit, processus, organisationnel et outil. Le cœur de l'ingénierie des méthodes est formé de cette compréhension commune de ces concepts constituant ou pouvant constituer la description d'une méthode. En conséquence de cela, nous avons choisi de bâtir l'ontologie OMCM autour de ces cinq aspects fondamentaux de la définition d'une méthode. Les concepts de composants de méthode issus de différentes approches telles que les fragments de méthode, les chunks, les composants de méthode, les composants OPF et les services SO2M pourront donc être corrélés sémantiquement à l'aide l'ontologie.

La section 4.2 détaille l'ontologie OMCM alors que l'usage de cette ontologie dans une architecture orientée service est l'objet de la section 4.3.

4.2. L'ontologie OMCM

Une introduction aux ontologies est effectuée à la section 4.2.1 afin de positionner notre proposition. Les concepts de l'ontologie OMCM sont expliqués à la section 4.2.2. La section 4.2.3 illustre comment l'ontologie OMCM est utilisée pour représenter la sémantiques des cinq approches d'IMS présentées au chapitre 2 (l'état de l'art).

4.2.1 Les ontologies

Le terme ontologie vient du grec "onto" signifiant "étant", le participe présent de "être". D'un point de vue philosophique, ce terme signifie l'étude de l'être en tant qu'être (Aristote, 2008). Une ontologie est l'étude des propriétés générales de tout l'existant. Le terme ontologie a été repris dans le domaine de l'informatique par Gruber en 1993 (Gruber, 1993) pour proposer la première définition d'une ontologie dans le contexte de l'informatique.

Gruber (Gruber, 2009) définit désormais une ontologie comme suit.

Dans le contexte de l'informatique et des sciences de l'information, une ontologie définit un ensemble de primitives de représentation utilisées pour modéliser un domaine de connaissance ou de discours.

Les primitives de représentation sont généralement des classes (ou des ensembles), des attributs (ou des propriétés) et des relations (des relations entre différents membres de classe). Les définitions des primitives doivent inclure les informations concernant leur signification et les contraintes pour leur application logique et cohérente.

Dans le cadre de l'informatique, les ontologies ont d'abord été utilisées dans le domaine de l'intelligence artificielle pour les systèmes de connaissance. Elles sont utilisées dans le domaine du web sémantique pour définir un vocabulaire d'échange standardisé entre plusieurs systèmes, pour construire des bases de connaissances, construire des fonctionnalités de requêtage, ou encore faciliter l'interopérabilité entre des systèmes ou des bases de données.

Du fait de ces multiples cas d'application, il existe des typologies variées d'ontologies. Nous présentons dans cette section la typologie issue de (Psyche, 2003), également reprise dans les travaux de (Guzélian, 2007b) et (Mellal, 2007). Cette typologie propose une classification des ontologies par rapport à leur objet de conceptualisation. Elle recense six types d'ontologie différents: de représentation des connaissances, de haut niveau, générique, de domaine, de tâche et d'application.

- Les *ontologies de représentation des connaissances* regroupent les concepts impliqués dans la formalisation des connaissances.
- Les *ontologies de haut niveau* sont, comme leur nom l'indique, les ontologies définies à un haut niveau d'abstraction. Ce sont des ontologies générales, les concepts constituant une ontologie de haut niveau sont détachés d'un domaine ou d'un problème spécifique. Ils décrivent des choses très générales comme les entités, les événements, les états, les processus, les actions, le temps, l'espace, les relations et les propriétés.
- Les *ontologies génériques* ou "core ontologies" sont des méta-ontologies, elles décrivent cependant des concepts moins abstraits que les ontologies de haut niveau. Elles véhiculent néanmoins des connaissances génériques pouvant être utilisées dans plusieurs domaines. Elles permettent également de définir des standards pour l'interopérabilité de plusieurs systèmes.
- Les *ontologies de domaine* décrivent un vocabulaire spécifique à un domaine en particulier. Elles sont liées à leur domaine d'application, par exemple le domaine médical, le domaine du tourisme etc.
- Les *ontologies de tâches* regroupent l'ensemble des tâches d'un domaine spécifique. Elles permettent de fournir un vocabulaire en rapport avec une tâche et de décrire de manière générique la résolution d'un type de problème.
- Les *ontologies d'application* sont une combinaison d'ontologies de domaine et d'ontologies de tâches.

Concernant les langages de représentation des ontologies, il en existe une multitude. C'est pourquoi nous nous intéressons dans cette thèse uniquement au langage OWL (W3C/OWL, 2004), standard utilisé dans le domaine du web sémantique et défini par le W3C (W3C, 2012). Ce langage a pour origine le langage RDF (W3C/RDF, 2004) dont il est une extension. Il est également basé sur le langage de représentation XML (W3C/XML, 2008). Ceci lui permet d'être facilement opérationnalisable. OWL propose de représenter une ontologie en termes de concepts, de relations entre ces concepts et d'instances (Guzélian, 2007b).

4.2.2 Description des concepts de l'ontologie OMCM

Dans cette section, les différents concepts et leurs relations de l'ontologie OMCM sont présentés. Comme nous l'avons énoncé précédemment, les approches d'IMS actuelles ont chacune des blocs de construction modulaire de méthode différents. Cependant, elles s'accordent sur une définition commune de ce qu'est une méthode (Seligmann, 1989), (Plihon, 1995). D'après cette définition, une méthode est constituée d'un ensemble de composants modulaires. Notre ontologie illustrée à la figure 4.1 est basée sur cette définition.

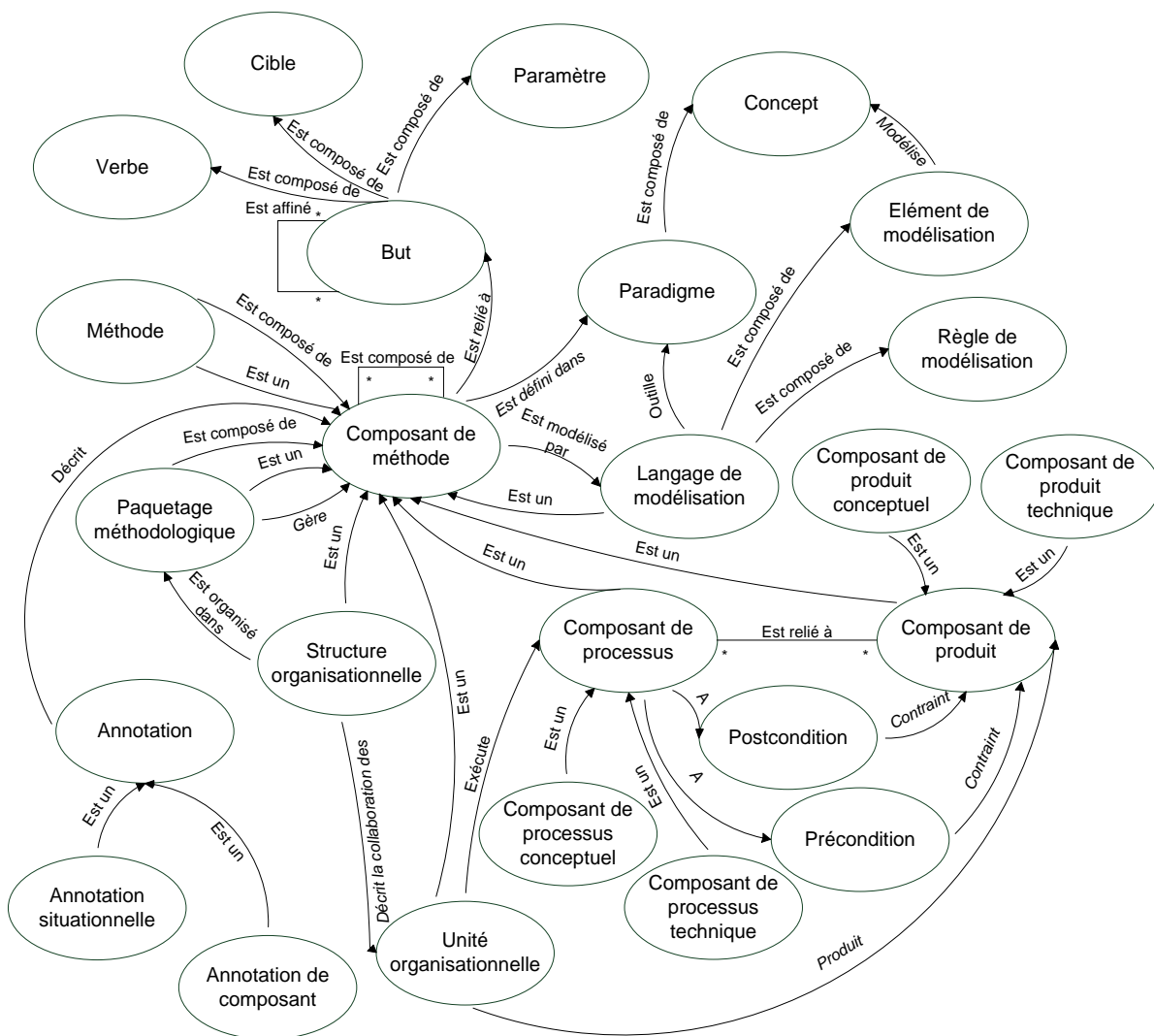


Figure 4.1 : Ontologie de métamodèles composant de méthode

Le terme méthode provient du grec "methodos" signifiant moyen d'investigation (Brinkkemper, 1996). D'après Harmsen, une méthode est un ensemble de procédures, de techniques, de descriptions de produit et d'outils pour un support efficace et efficient des processus d'ingénierie (Harmsen, 1997). Dans le contexte de l'IMS, les blocs de construction modulaire de méthode (i.e. *Composant de méthode*) sont assemblés afin de former une méthode adaptée à la situation dans laquelle elle sera appliquée. Cette décomposition d'une méthode en différentes parties qui sont les *Composants de méthode* est fondamentale pour garantir la flexibilité, l'adaptabilité, l'utilisation et la réutilisation des méthodes (Rolland, 2005). Une *Méthode* est ainsi formée d'une composition de *Composants de méthode* et elle sera elle-même perçue comme un *Composant de méthode*.

Un *Paquetage méthodologique* est une partie réutilisable d'une méthode construite autour d'un thème particulier de la méthode. Il définit un ensemble de *Composants de méthode* répondant à un type spécifique de projet. Il est également considéré comme une partie préconfigurée d'une méthode. De plus, il peut incorporer une dimension temporelle pour organiser les différents *Composants de*

méthode qui le constituent (Wistrand, 2004), (Karlsson, 2004), (Gonzalez-Perez, 2007). Par exemple la phase de gestion du cycle de développement d'une méthode situationnelle agile est un *Paquetage méthodologique*. Cette phase est organisée temporellement en plusieurs étapes (affinement des besoins, planification, traçabilité de l'exécution et rétrospective), chacune pouvant regrouper un assemblage de *Composants de méthode* préconfiguré.

Un *Composant de méthode* représente une unité de description de méthode selon les cinq aspects de Seligmann (Seligmann, 1989) avec les caractéristiques de modularité et de réutilisabilité. Cette unité est définie comme une pièce cohérente d'une méthode de développement de SI (Brinkkemper, 1996). Dans l'objectif de gérer la complexité des définitions de méthode, les approches d'IMS ont mis l'accent sur cette vision modulaire. Cette dernière est introduite par le concept de *Composant de méthode*. Ajouté à cela, les *Composants de méthode* peuvent être définis à des niveaux de granularité différents. En d'autres termes, les *Composant de méthode* peuvent être composés avec d'autres *Composants de méthode*. Il en va de même pour le concept de *Paquetage méthodologique* que nous avons présenté ci-dessus. Cela signifie qu'un *Paquetage méthodologique* est considéré comme un *Composant de méthode* tout en étant une composition de *Composants de méthode*. Les autres spécialisations possibles du concept de *Composant de méthode* correspondent aux différents aspects de la définition d'une méthode telle qu'elle a été énoncée par Seligmann et étendue par Rolland (Seligmann, 1989), (Plihon, 1995) : paradigme, produit, processus, organisationnel et outil. L'aspect paradigme est la philosophie utilisée dans un *Composant de méthode*. Cet aspect correspond dans notre ontologie au concept de *Paradigme* et est concrétisé en pratique par un *Langage de modélisation*. L'aspect produit, quant à lui, décrit les différents éléments de construction ainsi que leurs modèles résultant de l'application des méthodes. Cet aspect est défini au travers du concept de *Composant de produit* et plus précisément à l'aide du concept de *Composant de produit conceptuel*. En correspondance avec la définition de l'aspect produit, l'aspect processus exprime comment exécuter une méthode ou comment un produit évolue durant l'application d'une méthode. Cet aspect est identifié dans l'ontologie par le concept de *Composant de processus* et, plus particulièrement, par le concept de *Composant de processus conceptuel*. L'aspect organisationnel, pour sa part, spécifie comment organiser et exécuter un processus méthodologique dans une organisation. Il est représenté dans l'ontologie à la fois par les concepts de *Structure organisationnelle* et d'*Unité Organisationnelle*. Enfin, l'aspect outil présente les moyens utilisés pour outiller une méthode dans son exécution. Cet aspect est relié aux concepts de *Composant de processus technique* et de *Composant de produit technique*. L'ensemble des approches d'IMS partage la même définition d'une méthode au travers de ces cinq aspects, mais possède toutes des définitions différentes du concept de *Composant de méthode*. Avec l'objectif de tenir compte de cette diversité, l'ensemble des spécialisations possibles du concept de *Composant de méthode* sont inclusives pour les concepts de : *Méthode*, *Paquetage méthodologique*, *Composant de produit*, *Composant de processus*, *Langage de modélisation*, *Structure organisationnelle* et *Unité*

organisationnelle. En effet, le concept de chunk (Ralyté, 2001b) et (Ralyté, 2001c) est représenté (sa sémantique) par un *Composant de méthode* spécialisé en un *Composant de produit* et un *Composant de processus* afin de prendre en compte la partie processus et la partie produit.

L'objectif principal d'un *Composant de méthode* est d'introduire un élément de la définition d'une méthode qui soit modulaire et réutilisable. Dans le but d'augmenter cette réutilisabilité, il existe un mécanisme pour extraire et regrouper les aspects clefs contenus dans un *Composant de méthode*. Ce regroupement concis d'information sur un *Composant de méthode* est appelé une *Annotation*. Elle est utilisée en pratique pour la recherche et la récupération des *Composant de méthode* dans la base de stockage. Afin de réaliser ces activités, deux types d'information sont nécessaires : (a) les informations concernant les situations dans lesquelles un *Composant de méthode* peut être réutilisé et (b) les informations permettant de caractériser et de résumer le contenu d'un *Composant de méthode*. Respectivement, le premier type d'information est géré par le concept d'*Annotation situationnelle* et le second type est porté par le concept d'*Annotation de composant*. Considérons par exemple, un *Composant de méthode* issu de la méthode RUP (Kruchten, 1998), la situation favorable de réutilisation de ce *Composant de méthode* est : "un projet ayant un cycle de vie en cascade". Par conséquent, l'*Annotation situationnelle* peut se décrire par le texte suivant "définition des besoins d'une solution dans un projet ayant un cycle de vie en cascade". Le résumé de ce *Composant de méthode* peut, dans ce cas, se décrire par une structure avec les valeurs suivantes : "définition d'un diagramme de cas d'utilisation UML d'une solution (entrée : description du problème, sortie : diagramme de cas d'utilisation)" et représente une *Annotation de composant*.

Un *Composant de méthode* permet d'accomplir un *But* précis. Un *But* est une phrase exprimant ce qui est désiré (Jackson, 1995). En d'autres termes, il représente un état à atteindre ou à maintenir. L'approche linguistique proposée dans (Fillmore, 1968) et son extension proposée dans (Dik, 1989) sont basées sur une grammaire de cas. Cela signifie qu'elles reconnaissent un *But* comme une combinaison d'un *Verbe*, d'une *Cible* et d'un ensemble de *Paramètres*. Le verbe d'un but est l'élément central de la phrase constituant celui-ci. Il décrit l'action qui doit être exécutée. La cible, quant à elle, est le sujet du but, elle est le sujet sur lequel porte l'action du verbe. Elle peut décrire l'expression du résultat de l'accomplissement du but ou une entité existante qui sera altérée par son accomplissement. Ajouté à cela, les paramètres sont un ensemble d'informations complémentaires exprimées dans la phrase constituante du but. En pratique, chaque paramètre joue un rôle sémantique en correspondance avec le verbe. Les buts peuvent être définis à des niveaux de granularité différents. En effet, l'accomplissement d'un but complexe peut requérir l'accomplissement de sous buts. Nous définissons ici le fait qu'un but peut être affiné en un ensemble de sous buts. Ceci est exprimé dans notre ontologie par une relation de composition récursive sur le concept de *But*. Si l'on considère le *Composant de méthode* "test d'un incrément logiciel", son objectif (*But*) est "écrire les tests unitaires à l'aide du

schéma de parcours de l'algorithme du logiciel". Celui-ci se décompose de la manière suivante : *Verbe* = "écrire", *Cible* = "tests unitaires" et *Paramètre* = "à l'aide du schéma de parcours de l'algorithme du logiciel".

Un *Paradigme* de modélisation est la vision cohérente d'un monde basé sur une philosophie spécifique. Un *Paradigme* décrit un ensemble de concepts et leurs interactions, le tout ne pouvant être recoupé avec la définition d'un autre paradigme. Dans l'ontologie OMCM, les concepts utilisés dans l'expression d'un paradigme sont introduits par l'élément *Concept* du modèle. Un *Concept* aide à la représentation des éléments de construction et de leurs relations utiles à la description d'un *Paradigme*. Un *Langage de modélisation* est ensuite utilisé pour modéliser le monde en accord avec l'ensemble des concepts disponibles dans un *Paradigme*. Cela revient à le considérer comme un outil pour produire des modèles. Un *Langage de modélisation* peut être lui même défini comme un *Composant de méthode*, c'est-à-dire comme un ensemble d'*Éléments de modélisation* défini en correspondance avec un ensemble de *Règles de modélisation*. Un *Élément de modélisation* est un objet de représentation textuelle ou graphique d'un *Concept* issu d'un *Paradigme*, tandis qu'une *Règle de modélisation* est un axiome de modélisation qui doit obligatoirement être respecté par un *Élément de modélisation* ou un ensemble d'*Éléments de modélisation*. Par exemple, l'orientation objet est un *Paradigme* de description d'une solution au travers d'objets et de leurs relations les uns envers les autres. Dans le paradigme objet (*Paradigme*), le langage UML peut être considéré comme un *Langage de modélisation* de ce paradigme. Le concept d'agrégation entre deux classes est représenté graphiquement en UML par l'*Élément de modélisation* "un losange avec un trait simple". "Une agrégation est une relation asymétrique, dans cette relation les objets de la classe agrégat (dite composite) sont composés d'un ou plusieurs objets de la classe composante. Graphiquement, L'ancrage du losange doit se faire sur la classe composite alors que l'ancrage du trait simple doit permettre d'identifier la classe composante" est une règle d'usage de l'*Élément de modélisation* que l'on représente par une *Règle de modélisation*.

Au sein des approches d'IMS, le *Composant de processus* est l'un des concepts principaux. Il représente un élément abstrait ayant l'objectif de décrire un processus permettant de réaliser le *But* d'un *Composant de méthode* (Guzélian, 2007a). En d'autres termes, il représente la masse de travail qui doit être achevée pour obtenir le résultat désiré (Gonzalez-Perez, 2007). Ou, plus précisément, il peut être considéré comme l'ensemble des actions nécessaires à la transformation d'un produit (*Composant de produit*) en cours de développement (Rolland, 1998), à partir d'un produit source jusqu'à l'obtention du produit ciblé (Wistrand, 2004). Dans ce lien de spécialisation existant entre le concept de *Composant de processus* et le concept de *Composant de méthode*, un *Composant de processus* lui-même peut être défini à des niveaux de granularité variés. Il peut, par exemple, décrire les stratégies de haut niveau d'un projet ou encore un ensemble de procédures de développement.

La partie modélisation de la structure de processus est représentée par le concept de *Composant de processus conceptuel*. Ce concept comprend un ensemble de descriptions du processus et de modèles. Son implémentation est alors supportée par le concept de *Composant de processus technique*. Ce dernier représente une automatisation du concept de *Composant de processus* par un outil opérationnel. Comme présenté dans (Guzélian, 2007a), deux autres concepts sont reliés au *Composant de processus* : une *Précondition* et une *Postcondition*. Le concept de *Précondition* définit la situation initiale requise pour l'application d'un *Composant de processus*. À l'opposé de cela, le concept de *Postcondition* définit la situation finale résultante de l'application d'un *Composant de processus*. Ce concept est une restriction contraignant les instances de *Composant de produit* en résultat de l'exécution du *Composant de processus*. En d'autres termes, une *Postcondition* définit l'état souhaité des produits en sortie d'un *Composant de processus*. Dans le cadre d'une méthode agile, la planification d'une itération de développement nécessite un processus méthodologique de "calcul des estimations des besoins, de découpage en tâches et d'ordonnancement de celles-ci dans un planning" qui est représenté par un *Composant de processus conceptuel*. Par contre, le support outillé de ce processus est considéré comme un *Composant de processus technique*. Ces deux *Composants de processus* sont conditionnés par une *Précondition* exprimant "l'existence d'une liste de besoins à développer". Leur *Postcondition* correspond à "l'obtention d'un planning de tâches".

Un autre concept clef des approches d'IMS est le *Composant de produit*. C'est un élément abstrait décrivant l'aspect produit des méthodologies (Gonzalez-Perez, 2007) tout en étant conforme à la définition des paradigmes utilisés par celles-ci. Un *Composant de produit* modélise l'ensemble des artefacts utilisés ou produits par l'exécution d'un *Composant de processus* (Brinkkemper, 1996), (Rolland, 1998), (Guzélian, 2007a), (Gonzalez-Perez, 2007). Les modèles de ces artefacts sont définis dans notre ontologie à l'aide du concept de *Composant de produit conceptuel* tandis que leurs instances et leurs implémentations sont supportées par le concept de *Composant de produit technique*. Reprenons le *Composant de processus* associé à "la planification d'une itération de développement", le modèle de "liste des besoins" est un exemple de *Composant de produit conceptuel*. Le schéma XSD permettant de représenter la liste des besoins sous forme d'un document XML est considéré comme un *Composant de produit technique*.

L'aspect organisationnel que nous avons présenté précédemment est représenté dans l'ontologie OMCM par les concepts de *Structure organisationnelle* et d'*Unité organisationnelle*. Une *Unité organisationnelle* peut être définie comme une ressource, un rôle porté par un acteur, ou la description d'un ensemble d'acteurs (une équipe ou un plus grand groupe) intervenant dans l'exécution d'un *Composant de processus* dans l'objectif de produire un produit décrit par un *Composant de produit*. Le concept de *Structure organisationnelle* est pour sa part utilisé pour modéliser la collaboration entre plusieurs *Unités organisationnelles*. Il identifie l'ensemble des interactions entre les *Unités*

organisationnelles afin d'accomplir un projet, une mission professionnelle dans une entreprise (OPF, 2012). Une *Structure organisationnelle* peut-être reliée à un ensemble de *Paquetages méthodologiques* et être également organisée temporellement à l'intérieur de ces derniers. Enfin, le concept de *Structure organisationnelle* peut aussi être considéré comme un *Composant de méthode* grâce au lien d'héritage qui les relie. Par exemple, les rôles d'acteur comme celui de client, chef de projet et développeur dans une méthode peuvent être considérés comme des exemples d'*Unité organisationnelle*. De plus, une équipe de développement constituée d'un chef de projet et d'un ensemble de développeurs est également une *Unité organisationnelle*. Le type de collaboration "maîtrise d'œuvre/ maîtrise d'ouvrage" entre le client et l'équipe de développement est considérée comme une *Structure organisationnelle*.

Nous avons présenté dans cette section l'ensemble des concepts de l'ontologie OMCM constituant le cœur des différents composants de méthode des approches d'IMS. Dans la section suivante de ce chapitre, nous allons illustrer comment ces concepts sont utilisés pour définir les différents éléments des composants de méthode des cinq approches d'IMS les plus connues.

4.2.3 Correspondance entre l'ontologie OMCM et les métamodèles de composant de méthode

Nous présentons ici comment chacune des approches d'IMS peut être définie sémantiquement par rapport à l'ontologie OMCM que nous avons présentée dans la section précédente. Nous proposons, à l'instar de l'état de l'art réalisé dans le chapitre 2 de cette thèse, d'illustrer la couverture de cette ontologie sur les cinq approches d'IMS les plus citées dans la littérature (Brinkkemper, 1996), (Rolland, 1998), (Wistrand, 2004), (Firesmith, 2002) et (Guzélian, 2007a). Quatre d'entre elles sont basées sur des composants de méthode et une approche est définie au travers du paradigme de service. Nous souhaitons montrer que notre ontologie peut constituer un socle commun pour le domaine et les approches d'IMS. Pour des raisons de lisibilité, nous utilisons le terme "élément" pour désigner les concepts des métamodèles de composant de méthode et nous utilisons le terme "concept" pour désigner les concepts de l'ontologie OMCM.

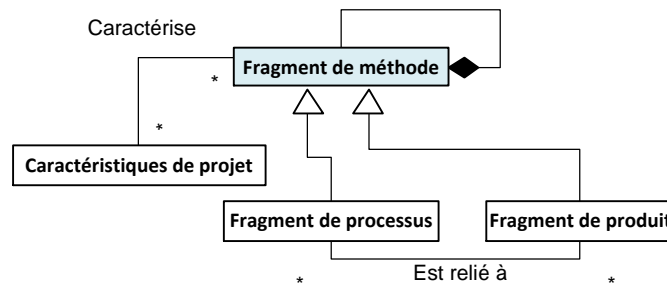
4.2.3.1. Fragments de méthode

Dans (Brinkkemper, 1996), (Brinkkemper, 1998), (Brinkkemper, 1999), (Harmsen, 1994) et (Harmsen, 1997), les auteurs proposent le concept de fragments de méthode. D'après l'analyse des approches d'IMS publiée dans (Agerfalk, 2007), ce concept de construction modulaire de méthode est l'un des premiers qui soit apparu dans le domaine de l'IM.

Selon (Brinkkemper, 1996), un fragment de méthode peut être défini comme un bloc de construction standardisé basé sur une partie cohérente d'une méthode (Brinkkemper, 1996). C'est un élément abstrait défini dans un des cinq niveaux de granularité disponibles: méthode, étape, modèle, diagramme ou concept (Brinkkemper, 1999). Au niveau de notre ontologie OMCM, le concept de

fragment de méthode correspond au concept de *Composant de méthode* et les différentes granularités d'un fragment de méthode sont représentées au travers du lien de décomposition récursif d'un *Composant de méthode*.

La figure suivante présente le modèle de la structure d'un fragment de méthode ainsi que la spécialisation des concepts de ce modèle depuis l'ontologie OMCM.



Partie A : structure d'un fragment de méthode

Partie B : correspondance avec l'ontologie OMCM

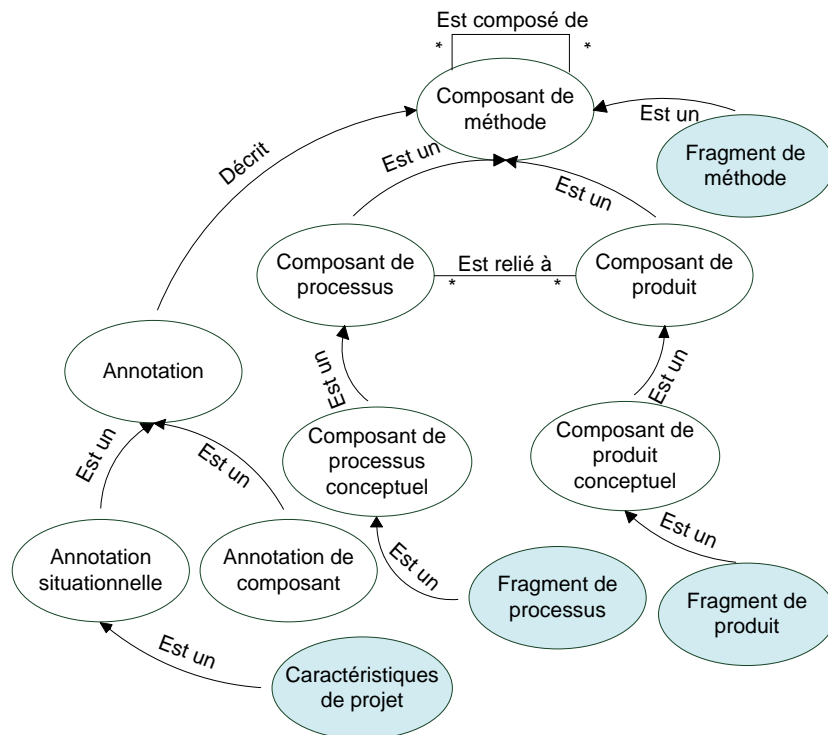


Figure 4.2 : Structure d'un fragment de méthode et correspondance avec l'ontologie OMCM

Comme représenté à la figure 4.2, un "Fragment de méthode" peut-être spécialisé soit en un "Fragment de processus" soit en un "Fragment de produit". Étant donné qu'un "Fragment de produit" est destiné à modéliser la structure d'un produit d'une méthode et qu'un "Fragment de processus" est une modélisation d'un processus de développement (Brinkkemper, 1996), ces deux derniers éléments

peuvent être respectivement définis dans l'ontologie OMCM comme des spécialisations des concepts de *Composant de Produit conceptuel* et de *Composant de processus conceptuel*.

Néanmoins, dans les "Fragments de méthode", les deux spécialisations produit et processus sont exclusives contrairement à ceux de l'ontologie. Par conséquent, cette spécialisation peut être vue comme un cas particulier des concepts de l'ontologie OMCM. En effet, la spécialisation d'un fragment en fragments de processus et de produits est exclusive.

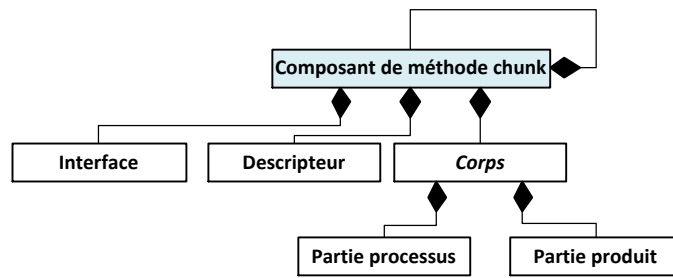
Concernant la récupération et l'usage de "Fragments de méthode", ces mécanismes sont fournis dans l'approche de Brinkkemper par un ensemble de "Caractéristiques de projet" attachés à chaque fragment de méthode. Ces caractéristiques de situation de projet correspondent à notre concept d'*Annotation situationnelle* tel qu'il est défini dans l'ontologie OMCM.

L'approche des fragments de méthode a posé les bases de l'ingénierie des méthodes situationnelles en proposant une réorganisation des méthodes sous la forme de relations entre des blocs de construction modulaire décrivant respectivement les aspects processus et produit d'une méthode. La réutilisation de ces blocs est favorisée par la description de la situation de réutilisation avec des facteurs de contingence.

4.2.3.2. Composants de méthode chunks

L'approche des composants de méthode chunk a été proposée dans (Rolland, 1996), (Rolland, 1998), (Ralyté, 2001b), (Ralyté, 2001c), (Ralyté, 2003), (Mirbel, 2006a). Un composant de méthode chunk est organisé en deux niveaux de connaissances : un niveau connaissance méthodologique et un niveau de méta connaissance (Rolland, 1996). Le niveau méthodologique d'un chunk peut être défini comme une composition d'une partie processus d'une méthode et de la partie produit qu'il lui est associé.

La figure suivante présente le modèle de la structure d'un chunk ainsi que la spécialisation des concepts de ce modèle depuis l'ontologie OMCM.



Partie A : structure d'un chunk

Partie B : correspondance avec l'ontologie OMCM

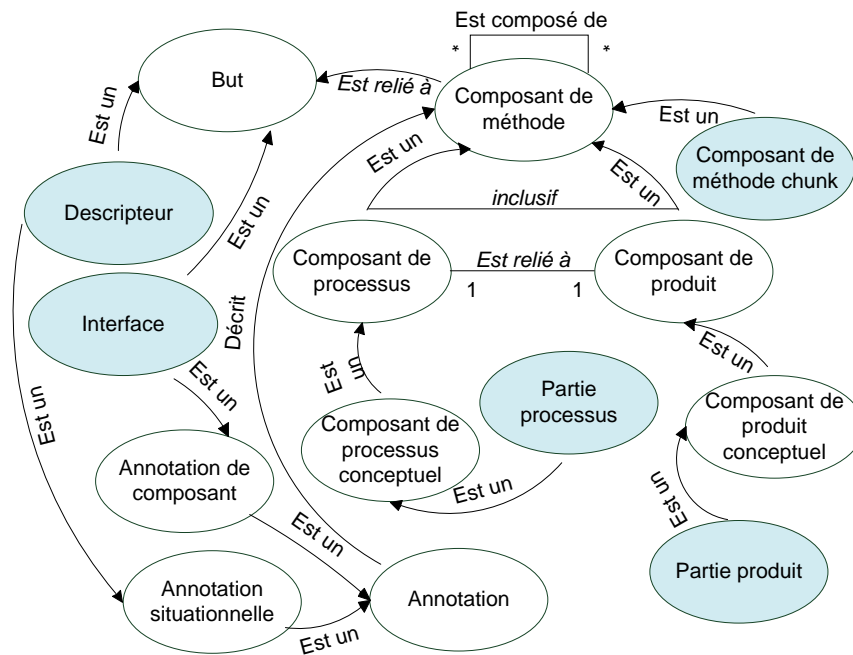


Figure 4.3 : Structure d'un composant de méthode chunk et correspondance avec l'ontologie OMCM

Un "Chunk" est donc caractérisé par une relation de spécialisation inclusive et simultanée du concept de *Composant de méthode* en concepts de *Composant de processus conceptuel* et de *Composant de produit conceptuel*. Par conséquent, les "Partie processus" et "Partie produit" d'un "Chunk" sont définies respectivement dans l'ontologie comme une spécialisation des concepts de *Composant de processus conceptuel* et de *Composant de produit conceptuel* (voir figure 4.3).

Cependant, la relation entre ces deux concepts pour l'approche chunk est un cas particulier de l'ontologie. En effet, la relation entre la "Partie processus" et la "Partie produit" d'un "Chunk" est plus spécifique que la relation entre un *Composant de processus* et un *Composant de produit* car c'est une relation ayant une cardinalité de un à un pour chacune de ses extrémités au lieu d'une cardinalité de n à n.

Concernant l'élément "Corps" introduit dans l'approche chunk, il est utilisé en tant qu'élément abstrait afin d'encapsuler les "Partie processus" et "Partie produit" d'un "Chunk". Étant donné la nature abstraite de cet élément, il n'apporte pas de signification sémantique structurelle à l'élément "Composant de méthode chunk". Nous ne prendrons donc pas en compte cet élément dans la modélisation d'un chunk par rapport à l'ontologie OMCM.

L'interface ("Interface") définit un ensemble d'informations sur le "Chunk" ainsi que son but. La correspondance entre cet élément "Interface" et l'ontologie est réalisée par un double lien d'héritage avec les concepts d'*Annotation de composant* et de *But*. D'une manière similaire, le niveau de méta connaissance d'un "Chunk" est représenté dans celui ci par l'élément "Descripteur" et ce dernier peut être défini dans l'ontologie par un double lien d'héritage avec le concept d'*Annotation situationnelle* et de *But*.

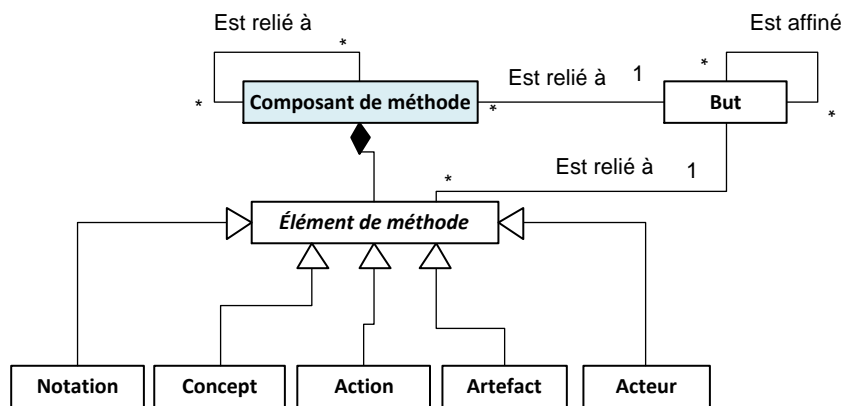
L'objectif du concept de "Descripteur" est de décrire l'ensemble des aspects situationnels lié à l'usage des composants de la méthode chunks, afin de favoriser leur processus de réutilisation. Un descripteur contient la définition du but réalisé par un "Chunk" ainsi que l'ensemble des paramètres caractérisant la situation de réutilisation de ce "Chunk".

Les composants de méthode chunks s'articulent autour de la relation entre l'aspect processus et l'aspect produit en encapsulant ces deux aspects au sein d'un même composant. Cette approche préconise une décomposition dirigée par les intentions.

4.2.3.3. Composants de méthode

L'élément "Composant de méthode" a pour objectif de représenter une partie autonome d'une méthode d'ingénierie des SI (Wistrand, 2004). Les dernières contributions apportées à cet élément ont été réalisées dans (Agerfalk, 2003), (Wistrand, 2004), (Karlsson, 2005 et, (Karlsson, 2006).

La figure suivante présente le modèle de la structure d'un composant de méthode selon (Wistrand, 2004), (Karlsson, 2004) ainsi que la spécialisation des concepts de ce modèle depuis l'ontologie OMCM.



Partie A : structure d'un composant de méthode

Partie B : correspondance avec l'ontologie OMCM

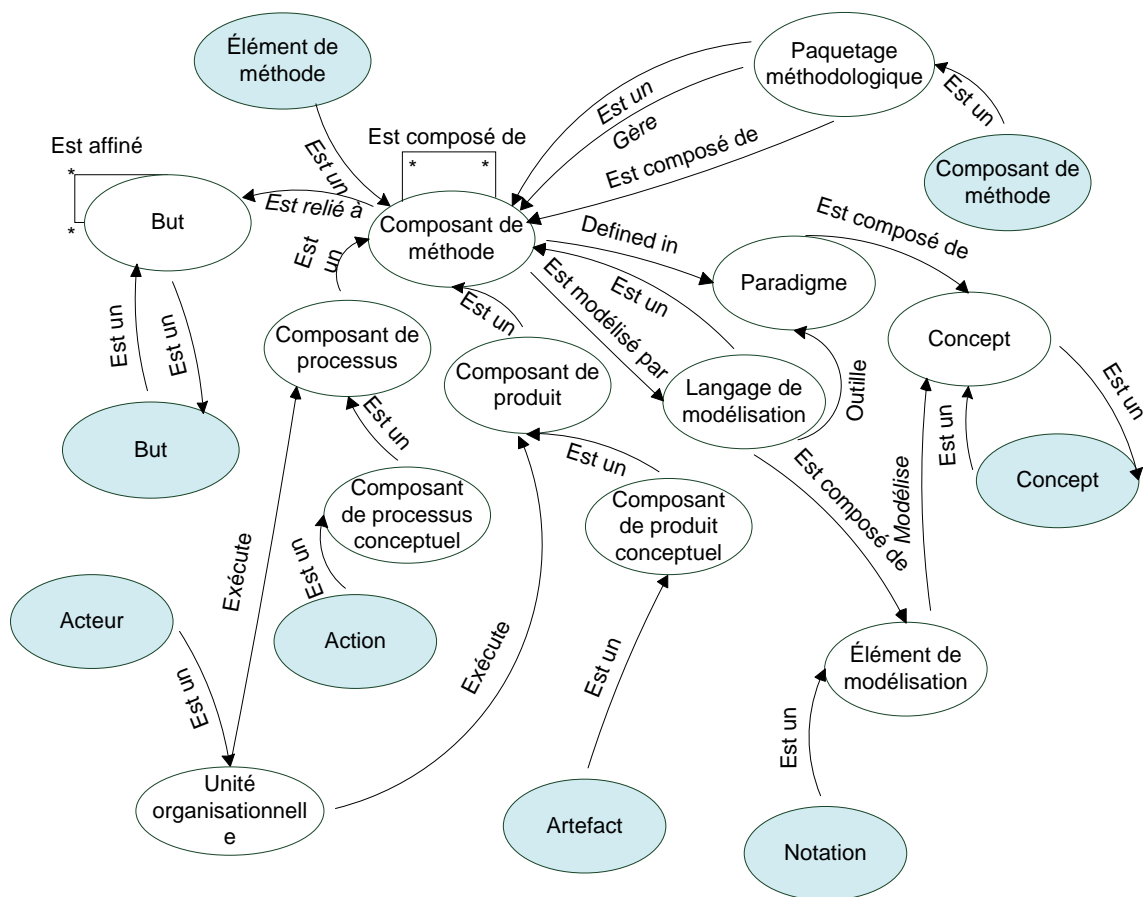


Figure 4.4 : Structure d'un composant de méthode et correspondance avec l'ontologie OMCM

Un "Composant de méthode" est destiné à être utilisé dans un cas spécifique du domaine de l'ingénierie des méthodes : la configuration de méthode. Chaque composant doit adresser un aspect d'un problème et est la partie la plus réduite d'une méthode qui soit utile en pratique (Agerfalk, 2007).

C'est pour ces raisons que nous avons choisi d'effectuer une correspondance entre l'élément "Composant de méthode" et le concept de *Paquetage méthodologique* de l'ontologie (voir figure 4.4).

Un "Composant de méthode" correspond à l'assemblage de plusieurs "Éléments de méthode" : une "Action", un "Artefact", "l'Acteur", un "Concept" et une "Notation". Cet élément "Élément de méthode" peut être défini dans l'ontologie OMCM comme une spécialisation du concept de *Composant de méthode*.

Une "Action" est définie dans l'approche des "Composants de méthode" comme un ensemble de tâches à exécuter durant l'application d'un composant. Étant donné que les éléments de type "Action" sont présentés comme les constituants centraux des modèles de processus d'une méthode dans (Wistrand, 2004), l'élément "Action" peut être défini comme une spécialisation du concept de *Composant de processus conceptuel* de l'ontologie.

Dans (Wistrand, 2004), les résultats d'exécution des éléments de type "Action" sont représentés par des "Artefacts" dans le modèle de produit. Le concept "d'Artefact" peut donc être mis en correspondance avec le concept de *Composant de produit conceptuel* par un lien d'héritage.

De plus, les actions sont exécutées par des membres, participant au projet, ayant chacun des rôles différents (Wistrand, 2004). Ceci implique que le concept "d'Acteur" peut être associé au concept d'*Unité organisationnelle* de l'ontologie.

Un ensemble de "Concepts" est alors utilisé pour décrire le domaine d'un problème adressé par un "Composant de méthode", ces éléments sont représentés dans des modèles par des "Notations" (Wistrand, 2004). Ces deux éléments ("Concept" et "Notation") correspondent respectivement à *Concept* et *Élément de modélisation* de l'ontologie OMCM.

Les "Composants de méthode" et les "Éléments de méthode" sont tout deux liés à des "Buts" pouvant être affinés en un ensemble de sous buts. Nous pouvons définir une correspondance parfaite entre cet élément "But" de l'approche des composants de méthode et le concept de *But* présenté dans l'ontologie OMCM.

Cette approche regroupe les différents éléments composant une méthode en paquetages méthodologiques. Elle intègre dans le composant l'aspect paradigme par le langage de modélisation qu'il lui est associé. De plus, elle apporte l'aspect organisationnel en incorporant la dimension acteur dans la modélisation d'un composant de méthode.

4.2.3.4. Composants de méthode OPF

Basée sur le standard international ISO/IEC 24744 (ISO/IEC 24744, 2010), l'approche OPF a été proposée dans (Firesmith, 2002), (Henderson-Sellers, 2002), (Gonzalez-Perez, 2007) et (OPF, 2012).

La figure 4.5 présente le modèle de la structure d'un composant OPF ainsi que la spécialisation des concepts de ce modèle par rapport à l'ontologie OMCM.

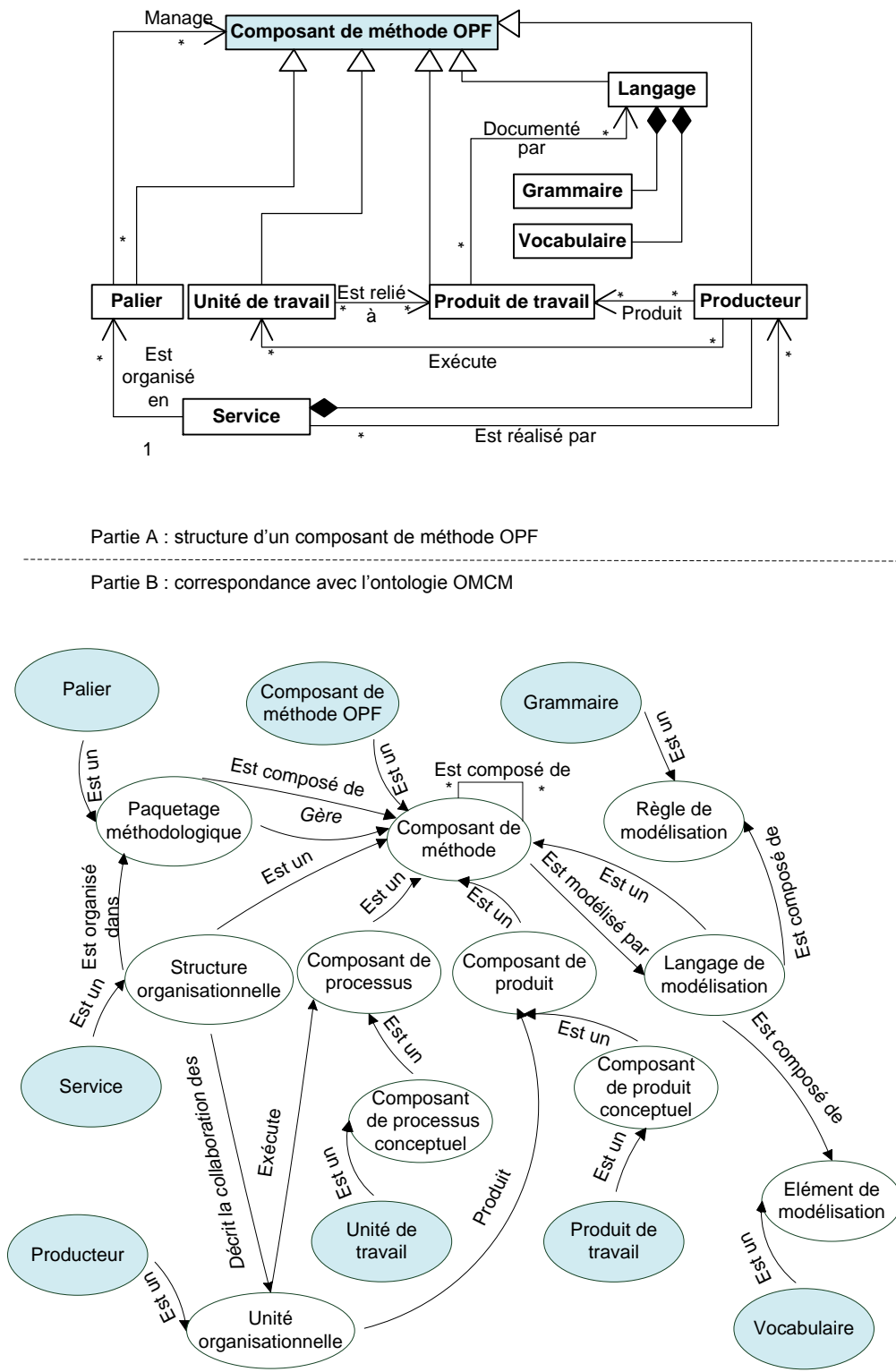


Figure 4.5 : Structure d'un composant de méthode OPF et correspondance avec l'ontologie OMCM

Les dernières contributions apportées à cette approche peuvent être consultées dans (OPF, 2012). En accord avec ce standard ISO, chaque composant de méthode OPF est généré à partir d'un élément d'un métamodèle sous-jacent (Agerfalk, 2007). Un "Composant de méthode OPF" est défini comme un élément abstrait duquel tous les autres éléments de construction d'une méthode sont dérivés (OPF, 2012). Il peut donc être défini comme une spécialisation du concept de *Composant de méthode* de l'ontologie OMCM (voir figure 4.5).

L'approche OPF est dirigée par la décomposition du processus d'une méthode en différentes "Unités de travail" exécutées par des "Producteurs", plus connus comme des rôles d'acteurs et des équipes. Ces deux derniers éléments correspondent respectivement aux concepts de *Composant de processus conceptuel* et d'*Unité organisationnelle* de l'ontologie.

Pour délivrer le système final, un ou plusieurs "Produits de travail" sont alors utilisés ou créés par des "Unités de travail" (Gonzalez-Perez, 2007). Cet aspect produit de la méthode est défini dans l'approche OPF par l'élément "Produit de travail" qui peut être défini comme une spécialisation du concept de *Composant de produit conceptuel* de l'ontologie.

Les "Produits de travail" sont ensuite documentés grâce à un "Langage" constitué d'un "Vocabulaire" et d'un ensemble de règles grammaticales ("Grammaire") (OPF, 2012). Ces derniers éléments peuvent être mis en correspondance respectivement avec nos concepts de *Langage de modélisation*, *Élément de modélisation* et *Règle de modélisation*.

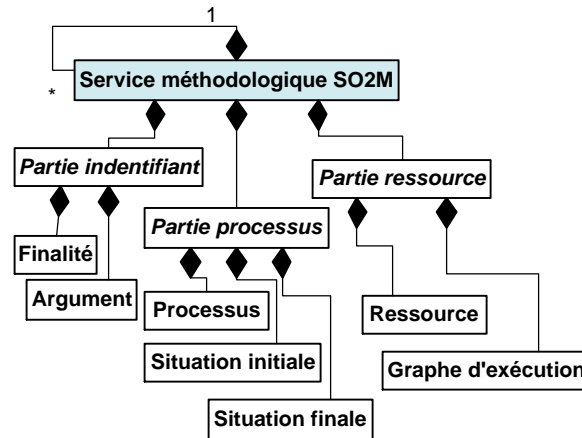
L'ensemble de ces composants de méthode OPF que nous venons de présenter sont utilisés lors d'une étape de la démarche d'une méthode et sont exécutées par une organisation collaborative spécifique de producteurs appelée un "Service" (OPF, 2012). Un "Palier" modélise une planification prévue de l'accomplissement d'un ensemble "d'Unités de travail" temporellement cohérentes (OPF, 2012). Un "Palier" peut alors être défini comme une spécialisation du concept de *Paquetage méthodologique* de l'ontologie OMCM alors que l'élément "Service" peut être défini comme une spécialisation du concept de *Structure organisationnelle*.

Cette approche constitue un premiers pas vers la prise en compte de l'aspect outil dans les méthodes situationnelles. En effet, cette approche est définie sur un paradigme objet étendu aux "clabjects". Un "clabject" est un élément qui est considéré à la fois comme un objet car il est l'instance d'une classe représentant un métatype et une classe car elle permet de décrire la structure et le comportement de façon intentionnelle d'instances. Néanmoins, les outils produits dans ce type de paradigme restent fortement dépendants de la plateforme et sont peu interopérables. L'approche OPF poursuit l'intégration de l'aspect organisationnel en représentant l'organisation dans laquelle un composant de méthode est réutilisable.

4.2.3.5. Services méthodologiques SO2M

Introduite dans (Guzélian, 2007a) et (Guzélian, 2007b), l'approche SO2M est le premier pas réalisé dans le domaine de l'IM pour l'application du paradigme orienté service (Papazoglou, 2003), (Papazoglou, 2007) sur une approche d'IMS.

La figure suivante présente le modèle de la structure d'un service méthodologique SO2M ainsi que la spécialisation des concepts de ce modèle depuis l'ontologie OMCM.



Partie A : structure d'un service méthodologique SO2M

Partie B : correspondance avec l'ontologie OMCM

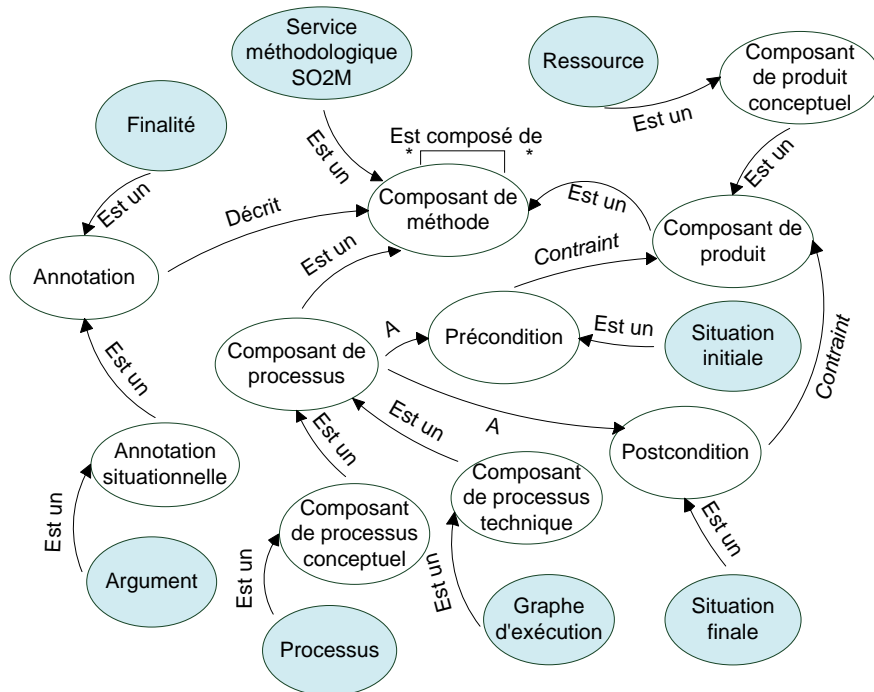


Figure 4.6 : Structure d'un service méthodologique SO2M et correspondance avec l'ontologie OMCM

Un "Service méthodologique SO2M" est une unité réutilisable contenant un ou plusieurs composants de méthode dans l'objectif de résoudre un problème lié au développement d'un SI (Guzélian, 2007a). Il peut être mis en correspondance avec le concept de *Composant de méthode* de l'ontologie OMCM et exploiter les liens de spécialisation inclusifs de celui-ci (voir figure 4.6).

Un service méthodologique de cette approche est constitué de trois parties abstraites : une "Partie identifiant", une "Partie processus" et une "Partie ressource". Étant donné le caractère abstrait de ces conteneurs nous n'établirons pas de correspondance entre ces concepts et l'ontologie OMCM.

La "Partie identifiant" a pour but de modéliser la connaissance contextuelle liée à la réutilisation du service méthodologique par la définition de sa "Finalité" et d'un "Argument". Une "Finalité" est la description du problème qu'adresse un service méthodologique, elle est structurée par un but, un moyen et un contexte (Guzélian, 2007a). Étant donné qu'une "Finalité" contient à la fois des informations situationnelles et structurelles concernant un service méthodologique, l'élément "Finalité" peut être défini dans l'ontologie OMCM en tant que spécialisation du concept d'*Annotation*. L'élément "Argument" introduit par l'approche SO2M caractérise la situation de réutilisation d'un service méthodologique. Cette situation est définie par le biais d'une liste d'argument pro (des avantages) et des arguments contre (des désavantages) sur le contexte d'utilisation service. Nous pouvons donc établir une correspondance entre l'élément "Argument" et le concept d'*Annotation situationnelle* de l'ontologie.

La partie processus est composée de la "Situation initiale" et de la "Situation finale" du processus méthodologique, ainsi que de la description de la structure du "Processus" lui-même. Ces trois éléments peuvent être respectivement mis en correspondance avec les concepts suivants de l'ontologie : *Précondition*, *Postcondition* et *Composant de processus conceptuel*.

La "Partie ressource" de cette approche définit l'implémentation du processus méthodologique par un "Graphe d'exécution" pouvant être défini dans l'ontologie comme une spécialisation du concept de *Composant de processus technique*. Cette partie d'un service méthodologique SO2M définit également l'ensemble des descriptions de toutes les "Ressources" consommées et délivrées par le processus de la méthode qu'il représente. Nous pouvons établir une correspondance entre cet élément "Ressource" et le concept de *Composant de produit conceptuel*.

Cette approche fournit une couverture de l'aspect outil des composants de méthode sous la forme de services méthodologiques. Elle propose d'outiller une méthode situationnelle par un graphe d'exécution constitué de services méthodologiques. La description des composants est dirigée par les intentions afin de favoriser leur réutilisation.

4.2.4 Synthèse

Cette section du chapitre montre que chaque concept existant dans l'ontologie OMCM peut être mis en correspondance avec l'ensemble des concepts issus des cinq approches d'IMS étudiées. Inversement, l'ensemble des concepts portés par les composants de méthode des approches d'IMS peuvent être spécifiés à partir des concepts de l'ontologie. Ces deux propriétés montrent ainsi la pertinence de l'ontologie OMCM pour positionner sémantiquement les métamodèles de composant de méthode.

De plus, la correspondance établie entre les approches d'IMS étudiées et l'ontologie montre que chacune de ces approches partage des concepts communs avec les autres mais inclut également de nouveaux concepts caractérisant des éléments de construction de méthode qui ne sont pas représentés dans les autres approches. Cette ontologie représente une base sémantique pour la compréhension des différences entre les différentes approches. Cela permet donc de réduire le problème de compréhension induit par la diversité des approches d'IMS. Elle montre également les différences entre les approches et justifie la diversité des approches d'IMS par le fait que chacune d'entre elles s'intéressent à des éléments rattachés à des aspects de la définition d'une méthode qui ne sont pas pris en compte par les autres approches.

Comme nous venons de le voir à la section précédente, l'ontologie OMCM permet de couvrir la sémantique des éléments des métamodèles de composant de méthode. On peut également remarquer qu'aucun concept de l'ontologie OMCM est inutile.

Nous allons présenter dans la section suivante de ce chapitre un exemple de scénario d'utilisation de cette ontologie dans le cadre de l'utilisation de techniques de recherche sémantique sur un annuaire de services méthodologiques.

4.3. Utilisation de l'ontologie OMCM dans une architecture orientée service

Cette section présente l'utilisation de l'ontologie OMCM dans le cadre du paradigme des AOS appliqué à l'IMS. L'ontologie OMCM apportant une base sémantique commune au domaine de l'IMS, son utilisation a pour objectif d'apporter une solution standard à la description des services méthodologiques ou des composants de méthode utile lors de leur publication et leur recherche dans un annuaire. En effet, ce type d'ontologie, nommé générique de par sa nature abstraite, permet d'établir une interopérabilité sémantique entre les descripteurs des composants de méthode, peu importe l'approche dont ils sont issus. En s'inspirant des approches de recherche sémantique de web services tel que l'approche YASA (Chabeb, 2008) et (Chabeb, 2010), nous proposons d'utiliser l'ontologie OMCM conjointement aux standards de descriptions des services de l'architecture AOS afin de produire des descripteurs de service méthodologique annotés sémantiquement. Ces annotations construites à partir de l'ontologie constituent la partie sémantique du descripteur service

méthodologique. Elles contiennent des informations relatives au composant de méthode associé au service méthodologique.

Nous présentons, dans un premier temps, dans cette section l'architecture orientée service. Dans un deuxième temps, nous décrivons les principes de fonction de l'approche de recherche sémantique de web service sur un annuaire nommée YASA. Enfin, dans un troisième temps nous proposons une utilisation de l'ontologie OMCM pour la construction de descripteurs de services méthodologiques dans le cadre de leur publication et de leur recherche sur un annuaire de services méthodologiques.

4.3.1 Les architectures orientées services (AOS)

L'architecture orientée service fournit un moyen logique de concevoir un système applicatif afin de fournir des services à d'autres applications. Ces dernières peuvent être destinées à des utilisateurs finaux ou distribuées sur un réseau au moyen d'interfaces publiques publiées et permettant leurs recherches (Papazoglou, 2003), (Papazoglou, 2007).

Cette architecture a été élaborée pour construire des plateformes logicielles distribuées capables de garantir une intégration des applications tout en répondant aux critères de flexibilité et d'agilité des entreprises actuelles.

L'AOS met en avant un principe de couplage faible et une intégration automatique des services avec pour objectif la construction d'applications distribuées sur le Web. C'est un modèle d'architecture définissant un système comme un ensemble d'agents logiciels distribués fonctionnant de concert pour la réalisation d'une fonctionnalité globale préalablement établie (Heather, 2001). L'AOS est une philosophie de conception d'applications logicielles indépendantes des plateformes techniques ou de technologies spécifiques. Cette architecture repose sur un ensemble de standards pour la construction d'applications distribuées tout en garantissant une gestion des transactions, la mise en place de politique de sécurité et une indépendance vis-à-vis des technologies et des protocoles utilisés.

D'après (Papazoglou, 2007), un service peut être défini comme un couple interface-implémentation formant un élément autonome invocable par son interface et dont l'implémentation est perçue comme une "boîte-noire" par les éléments externes. L'implémentation d'un service est une solution technique à une fonctionnalité désirée. Elle est encapsulée derrière une interface invocable et indépendante de la plateforme utilisée pour réaliser ou exécuter l'implémentation. Cette opacité concernant l'implémentation du service permet de garantir une indépendance entre le composant invoquant le service et l'implémentation du dit service. Cela signifie que les composants logiciels externes invoquant le service ne connaissent ni l'algorithme ni le moyen technique utilisé pour implémenter le service. Ces caractéristiques du concept de service de l'AOS en font l'élément de base constituant l'architecture AOS. En effet, dans l'AOS, tout est service, chaque fonctionnalité d'un système est implémentée par un service.

L'architecture découlant de ces principes est la suivante.

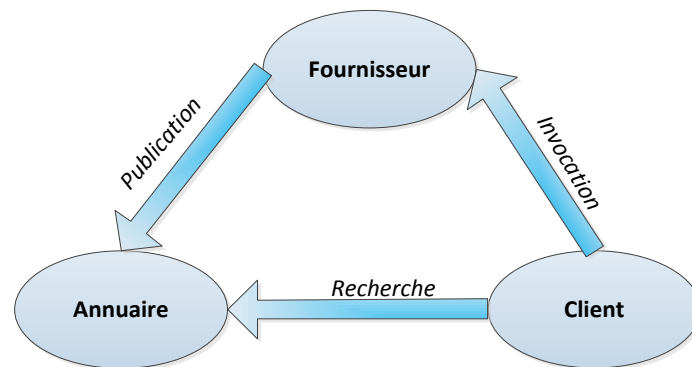


Figure 4.7 : Architecture orientée service

Comme le présente la figure 4.7, l'architecture AOS s'organise autour de trois rôles clés : le fournisseur de service, l'annuaire de services et le client.

Un fournisseur de service est une entité, une personne ou une organisation, proposant un agent approprié pour l'implémentation d'un service métier particulier (W3C/SOA, 2004). Il réalise également une interface pour l'invocation du service qu'il propose. Cette interface prend la forme d'un descripteur standard que le fournisseur publiera sur un annuaire de service pour rendre son service accessible au plus grand nombre.

Un client est une entité, une personne ou une organisation, souhaitant faire usage d'un service dont l'implémentation est proposée par un fournisseur de service (W3C/SOA, 2004). Lorsqu'il doit répondre à un besoin, un client effectue une recherche dans un annuaire de services. Une fois trouvé un service correspondant à ces attentes, il extrait le descripteur de ce service de l'annuaire. Ce descripteur contient toutes les données nécessaires pour le client pour réaliser l'invocation du service fournit par le fournisseur.

Un annuaire de service fournit les facilités de publication et de stockage des descripteurs de service pour les fournisseurs. Il fournit également les facilités de recherche de service et d'extraction des descripteurs pour les clients.

Les avantages tirés de l'application de cette architecture et son fonctionnement repose sur trois principaux standards. Le standard "Simple Object Access Protocol" (SOAP) (W3C/SOAP, 2007) est un protocole de communication léger permettant une communication avec et entre les services. Dans l'AOS, SOAP est un vecteur de communication faiblement couplé entre un client et un fournisseur. De par sa nature, ce standard est indépendant des langages utilisés pour le développement des services et de leurs plateformes d'exécution. SOAP définit un protocole pour l'encapsulation de données ou l'invocation d'opération à distance. Il ne définit pas un environnement d'exécution, les enveloppes des messages SOAP reçus sont ouvertes et les données extraites du corps du message sont alors

transformées selon le protocole natif de la plateforme. La requête est alors déléguée à l'implémentation du service invoqué (Papazoglou, 2007).

Afin de faciliter la recherche et l'invocation des services, le standard de "Web Services Description Language" (WSDL) (W3C/WSDL, 2001) a été défini. Il fournit un langage de description des services indépendamment des spécificités de leurs implémentations. Il permet de définir une interface autonome d'un service pour son invocation.

Le standard UDDI, quant à lui, propose une façon de définir un annuaire de services. Il régit la manière de publier et de rechercher des services. Ces deux fonctionnalités sont essentielles au fonctionnement de l'AOS. La publication est régie par un schéma de description extensible des services et des fournisseurs. Les fonctionnalités de recherche permettent aux clients de récupérer des informations sur les services ainsi que sur leurs fournisseurs et de rechercher des services répondant à leurs exigences. Les annuaires conformes à la norme UDDI maintiennent un index des services publiés ainsi qu'un index des fournisseurs. Ils permettent d'établir une relation de confiance entre les clients et les fournisseurs.

4.3.2 YASA : une technique de recherche de service sémantique

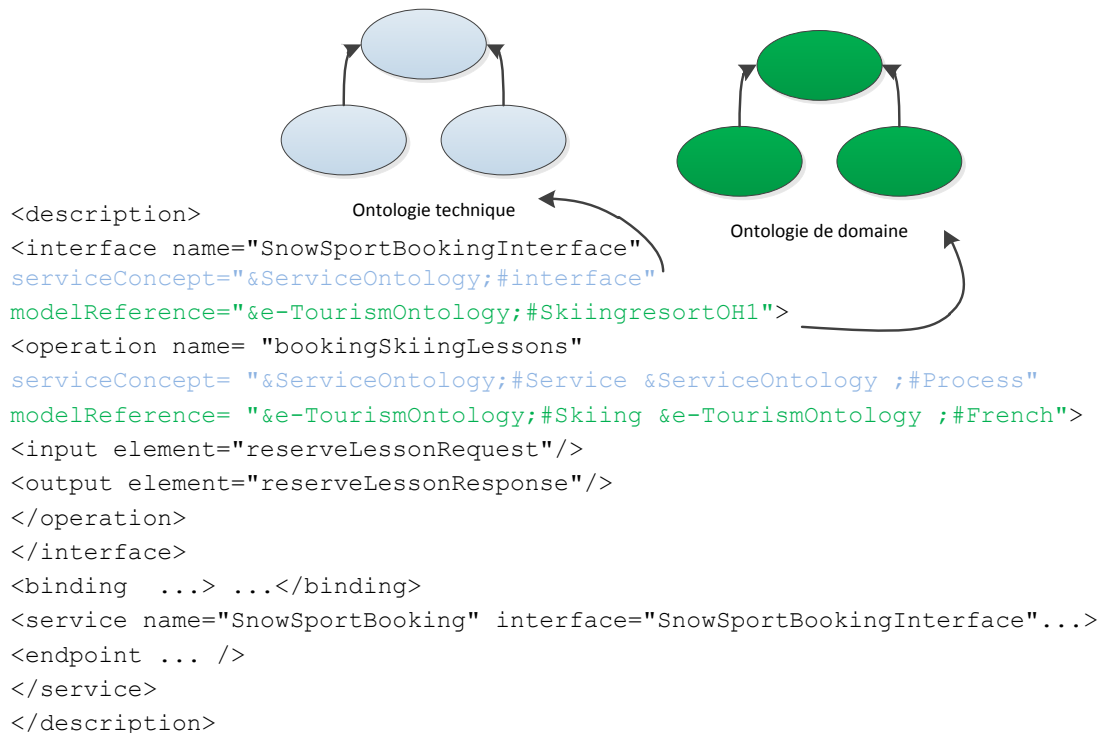
YetAnotherSemanticAnotation (YASA) est une approche de recherche sémantique de service. Elle a été proposée par Chabeb et Tata dans (Chabeb, 2008), elle s'inscrit dans le domaine des AOS et plus particulièrement au niveau des techniques de recherche sémantique de service web. En effet, cette approche se base sur l'introduction de données sémantiques dans les descripteurs techniques de service tels qu'un descripteur WSDL par exemple (W3C/WSDL, 2001). Ces ajouts de données sont introduits sous la forme d'annotations dans le descripteur technique d'un service web. Ils permettent de faciliter la découverte et la recherche des services stockés dans un annuaire et ainsi d'améliorer la pertinence des résultats retournés lorsque qu'une requête est exécutée.

Actuellement, l'approche YASA fait partie des meilleures approches de technique de recherche de services web sémantiques. La pertinence d'une technique de recherche de composants qu'ils soient logiciel ou service, se mesure d'une manière générale par deux indicateurs : la précision et le rappel. Lorsqu'une requête est exécutée, la précision est la mesure du nombre de résultats corrects à la requête sur l'ensemble des résultats obtenus. Le rappel, quant à lui, est la mesure du nombre de résultats corrects à la requête sur l'ensemble des résultats corrects contenus dans l'annuaire. En d'autres termes, un haut taux de précision permet d'affirmer que l'ensemble des résultats à une requête ne contient que peu de résultats erronés et un haut taux de rappel assure que peu de résultats corrects ont été "oubliés" dans l'annuaire de service. La difficulté pour les approches de recherche de composants est de fournir un fort taux pour ces deux mesures. Un fort taux dans une des deux mesures à tendance à être réalisé au détriment de l'autre et inversement.

Les approches de recherche sémantique de service ont vu le jour avec l'objectif d'apporter une dimension sémantique dans les techniques de recherche de services. En effet, les anciennes techniques de recherche sont basées sur une correspondance logique entre les services au niveau de leurs opérations, de leurs entrées, et leurs sorties. La dimension sémantique permet d'apporter du sens au descripteur d'un service. Elle permet également de rechercher les services par rapport aux fonctionnalités qu'ils rendent est donc d'augmenter la précision et le rappel par rapport aux approches classiques. Avec le développement de ce type d'approche sémantique, une multitude de descripteurs de services et une multitude d'implémentations d'annuaires spécifiques à ces descripteurs et aux types d'ontologies utilisées sont apparus. D'après (Chabeb, 2008) ces descripteurs destinés à résumer la sémantique d'un service sont trop fermés et trop restrictifs. Les descripteurs basés sur le langage OWL-S et WSMO ne peuvent utiliser que des ontologies décrites avec les langages OWL et WSML pour leurs ajouts sémantiques. De plus, ces deux langages ne disposent que d'un ensemble limité de concepts difficilement extensible pour effectuer les ajouts sémantiques sur les descripteurs de services. En ce qui concerne le langage SAWSDL, malgré le fait qu'il soit moins restrictif que les deux derniers langages au niveau des supports pour les ajouts sémantiques, il reste cependant limité dans ses possibilités d'annotations sémantiques sur un ensemble trop restreint d'éléments de descripteurs de services. L'approche YASA propose donc un nouveau moyen plus flexible pour l'ajout de données sémantiques à un descripteur de services, ainsi qu'une technique de recherche associée à ce nouveau descripteur.

La première contribution apportée par l'approche YASA se présente sous la forme d'une ontologie de descripteurs de service web. En d'autres termes, cette ontologie, qualifiée d'ontologie technique, agrège l'ensemble des concepts existants dans les différents langages de descripteur de service actuels tels que : WDSL, OWL-S, WSMO et SAWSDL. La construction de cette ontologie par la correspondance entre les différents concepts issus de ces langages a été effectuée par des calculs de similarité et ont été présentés dans (Chabeb, 2010). L'ontologie technique de service de l'approche YASA est utilisée dans un nouveau langage de description de service nommé YASA4WSDL. Ce nouveau langage est basé sur le langage SAWSDL (W3C/SAWSDL, 2007), lui-même basé sur le langage WSDL, et en propose une extension afin d'augmenter son expressivité. En effet, le langage SAWSDL propose la possibilité d'annoter les concepts d'un descripteur de service par des données sémantiques. Cependant, il n'est pas possible de préciser la nature de ces données. C'est ici qu'intervient l'ontologie technique de service dans le langage YASA4WSDL qui permet de préciser la nature d'une donnée sémantique en référence à l'ontologie technique lorsque qu'elle est ajoutée dans une annotation.

Nous appliquons l'approche YASA sur l'exemple de réservation de cours de ski (ou de snowboard) que nous avons réalisé. Le descripteur de ce service est écrit selon le langage YASA4WSDL et permet d'annoter n'importe quel concept d'un descripteur de service (voir figure 4.8).



Légende :

- **annotations en bleu** : concepts issus de l'ontologie technique.
- **annotations en vert**: concepts issus d'une l'ontologie de domaine.

Figure 4.8 : Exemple de descripteur YASA4WSDL de service de réservation de cours de ski

Comme nous pouvons le constater à la figure 4.8, une annotation de l'approche YASA se compose de deux parties. La première partie correspond à l'attribut *serviceconcept* qui peut être ajouté sur n'importe quel élément entrant dans la structure d'un descripteur WSDL. Cet attribut permet de définir une liste de références à des concepts de l'ontologie technique de YASA pour l'élément du descripteur WSDL auquel il est attaché. La deuxième partie est inspirée du langage SAWSDL et se présente sous la forme de l'attribut *modelreference*. Il correspond à la liste des données sémantiques respectivement affectées aux concepts de l'ontologie technique de l'attribut *serviceconcept*. De la même manière que pour l'attribut *serviceconcept*, l'attribut *modelreference* peut être utilisé sur n'importe quel élément d'un descripteur WSDL. Ces deux attributs doivent obligatoirement être utilisés conjointement pour former une annotation valide. Ils permettent ainsi d'ajouter des données sémantiques en référence à des concepts d'une ontologie technique sur n'importe quel élément entrant dans la composition d'un descripteur de service. C'est précisément ce point qui constitue le point fort de l'approche YASA par rapport aux autres approches de recherche sémantique de services.

D'un point de vue de l'utilisateur, lorsqu'il souhaite rechercher des services, il effectue une requête qu'il envoie à l'annuaire de services. Cette requête est construite en analogie avec la structure d'un descripteur WSDL. Cela signifie que l'utilisateur doit décrire l'interface du service dont il a besoin, ainsi que ses opérations et les entrées/sorties de ces dernières. Il peut ensuite annoter chacun de ces éléments constituant la requête en utilisant la même méthode que pour annoter un descripteur. Il peut ensuite ajouter des données sémantiques sur le comportement du service qu'il recherche à l'aide des deux attributs *serviceconcept* et *modelreference* sur chaque élément de la requête. La figure 4.9 montre un exemple de requête de l'approche YASA formulée par un utilisateur. Dans cet exemple, l'utilisateur souhaite rechercher un service de réservation de cours de ski.

```
<description>
...
<interface name="Interfacel"
serviceConcept="&ServiceOntology;#interface"
modelReference="&e-TourismOntology;#SkiingresortOH1">
<operation name= "Operation1"
serviceConcept= "&ServiceOntology;#Service"
modelReference= "&e-TourismOntology;#Skiing">
...
</operation>
</description>
```

Légende :

- **annotations en bleu** : concepts issus de l'ontologie technique.
- **annotations en vert**: concepts issus d'une l'ontologie de domaine.

Figure 4.9 : Exemple de requête YASA de recherche de service de réservation de cours de ski

Une autre contribution apportée par l'approche YASA est la technique de recherche en elle-même (Chabeb, 2010). C'est elle qui va calculer l'appariement entre la requête d'un utilisateur et les annotations des descripteurs de service lors de la recherche. Afin de mener à bien cette étape, l'ensemble des annotations de chaque descripteur de service est traduit en un ensemble de triplets RDF (W3C/RDF, 2004). Une étape de présélection est d'abord effectuée à l'aide d'une requête SPARQL (W3C/SPARQL, 2008) sur l'ensemble des descripteurs de services stockés en RDF. Cette présélection extrait les services dont le descripteur a une structure en correspondance avec la structure de la requête de l'utilisateur. Puis, l'appariement sémantique entre la requête et les descripteurs des services présélectionnés est réalisé en plusieurs opérations. La première consiste à effectuer un appariement élémentaire entre les concepts utilisés dans les annotations de la requête avec ceux du descripteur de service. Le processus d'appariement est effectué comme suit : pour chaque élément de la requête, du plus générique au plus spécifique (interface, opération, entrée, sortie), les concepts de l'ontologie technique utilisée pour annoter l'élément dans l'attribut *serviceconcept* sont comparés à ceux utilisés dans le même élément du descripteur d'un service. Pour chaque couple de concepts identiques entre la requête et le descripteur de service, les données des ontologies de domaine affectées à ces concepts de

l'ontologie technique sont comparées par un appariement élémentaire. Le degré d'appariement entre deux concepts est défini dans l'approche YASA par cet ensemble de valeurs du plus fort appariement au plus faible: {exact, subsume, est subsumé par, possède la même classe, non classifié, echec}.

Les définitions suivantes de chacune de ces valeurs sont issues de (Chabeb, 2010).

- *Exact: le concept de la requête et celui du descripteur sont équivalents.*
- *Subsume : le concept utilisé dans le descripteur est un sous concept de celui exprimé dans la requête.*
- *Subsumé par : le concept issu de la requête est un sous concept de celui utilisé dans le descripteur.*
- *Possède la même classe: les deux concepts issus de la requête et du descripteur de service sont tous deux des sous concepts d'un même concept parent.*
- *Non classifié: au moins un des deux concepts comparés n'est pas classifié.*
- *Echec : aucune relation ne peut être établie entre le concept de la requête et celui du descripteur.*

En fonction du degré d'appariement et de l'élément de la requête ou du descripteur (interface, opération, entrée, sortie), une valeur d'appariement est affectée aux couples de concepts comparés. Pour la liste des résultats retournés à l'utilisateur, les services sont classés par leur valeur d'appariement minimum obtenu pour la comparaison d'un concept de leur descripteur avec son concept correspondant dans la requête, puis par la valeur moyenne d'appariement de tous leurs concepts.

En reprenant l'exemple de requête de la figure 4.9, une présélection de service est effectuée par l'exécution de la requête SPARQL (W3C/SPARQL, 2008) suivante sur les descripteurs stockés dans l'annuaire sous la forme d'ensembles de triplets RDF. Cette requête en SPARQL sélectionne les services de l'annuaire ayant un descripteur présentant la même structure que la requête formulée par l'utilisateur.

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
SELECT ?service
WHERE {
  ?service ?r1 ?interface. FILTER
  regex(xsd:string(?r1), "hasInterface") .
  ?interface ?r2 ?op1. FILTER
  regex(xsd:string(?r2), "hasOperation")
  ...
}
```

Figure 4.10 : Exemple de requête de présélection en SPARQL

La requête SPARQL présentée à la figure 4.10 présélectionne dans l'annuaire l'ensemble des services ayant une interface et une opération. Cette étape va, par exemple, présélectionner dans l'annuaire le service de réservation de cours de ski présenté à la figure 4.8 et celui de réservation de cours de snowboard présenté à la figure 4.11.

```
<description>
<interface name="SnowSportBookingInterface"
serviceConcept="&ServiceOntology;#interface"
modelReference="&e-TourismOntology;#SkiingresortOH1">
<operation name="bookingSnowboardingLessons"
serviceConcept="&ServiceOntology;#Service &ServiceOntology ;#Process"
modelReference="&e-TourismOntology;#Sknowboarding&e-TourismOntology ;#French">
<input element="reserveLessonRequest"/>
<output element="reserveLessonResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="SnowSportBooking" interface="SnowSportBookingInterface"...>
<endpoint ... />
</service>
</description>
```

Légende :

- **annotations en bleu** : concepts issus de l'ontologie technique.
- **annotations en vert**: concepts issus d'une l'ontologie de domaine.

Figure 4.11 : Exemple de descripteur YASA4WSDL d'un service de réservation de cours de snowboard

La figure suivante présente les descripteurs des services de réservation de cours de ski et de snowboard ainsi que la requête de l'utilisateur au format RDF.

Descripteur sémantique du service de réservation de cours de ski au format RDF :

```
<description::42a, hasInterface, interface::SnowSportBookingInterface>
<description::42a/interface::SnowSportBookingInterface, hasInterface,
http://crinfo.univ-paris1.fr/e-TourismOntology#SkiingResortOH1>
<description::42a/interface::SnowSportBookingInterface, hasOperation,
description::42a/interface::SnowSportBookingInterface/operation::bookingSkiingLessons>
<description::42a/interface::SnowSportBookingInterface/operation::bookingSkiingLessons, hasService, http://crinfo.univ-paris1.fr/e-TourismOntology#Skiing>
<description::42a/interface::SnowSportBookingInterface/operation::bookingSkiingLessons, hasProcess, http://crinfo.univ-paris1.fr/e-TourismOntology#French>
```

Descripteur sémantique du service de réservation de cours de snowbord au format RDF :

```
<description::42b, hasInterface, interface::SnowSportBookingInterface>
```

```

<description::42b/interface::SnowSportBookingInterface, hasInterface,
http://crinfo.univ-parisl.fr/e-TourismOntology#SkiingResortOH1>
<description::42b/interface::SnowSportBookingInterface, hasOperation,
description::42a/interface::SnowSportBookingInterface/operation::bookingSnowboardin
gLessons>
<description::42b/interface::SnowSportBookingInterface/operation::bookingSnowboardi
ngLessons, hasService, http://crinfo.univ-parisl.fr/e-TourismOntology#Snowboarding>
<description::42b/interface::SnowSportBookingInterface/operation::bookingSnowboardi
ngLessons, hasProcess, http://crinfo.univ-parisl.fr/e-TourismOntology#French>

```

Requête de l'utilisateur au format RDF :

```

<description::42r, hasInterface, interface::Interfacel>
<description::42r/interface::Interfacel, hasInterface, http://crinfo.univ-
parisl.fr/e-TourismOntology#SkiingResortOH1>
<description::42r/interface::Interfacel, hasOperation,
description::42a/interface::Interfacel/operation::Operationl>
<description::42a/interface::Interfacel/operation::Operationl, hasService,
http://crinfo.univ-parisl.fr/e-TourismOntology#Skiing>

```

Figure 4.12 : Descripteurs RDF des exemples de services et de requête utilisateur

Comme présenté à la figure 4.12 les descripteurs RDF des deux services présélectionnés correspondent à la requête SPARQL de présélection. En effet, ils disposent chacun de deux triplets RDF ayant respectivement les prédicats "hasInterface" et "hasOperation". La requête de l'utilisateur est également transformée en un ensemble de triplets RDF afin de faciliter sa comparaison avec les descripteurs des services présélectionnés.

Pour chaque service présélectionné dans l'annuaire, les concepts utilisés pour l'annoter sont comparés deux à deux avec ceux utilisés dans la requête. En effet, pour chaque concept de l'ontologie technique identique entre le descripteur et la requête, les concepts issus d'ontologies de domaine qui leur sont associés sont comparés. Cette comparaison est réalisée à l'aide de l'échelle des degrés d'appariement (exact, subsume, est subsumé par, possède la même classe, non classifié, échec).

En fonction du degré d'appariement entre les annotations du descripteur et les annotations de la requête et de l'élément du descripteur qui est annoté (interface : <Interface>...</Interface>, opération : <Operation>...</Operation>, entrée : <Input>...</Input> et sortie : <Output>...</Output>), une valeur d'appariement est affectée pour chaque couple de concepts. La table de correspondance suivante est proposée dans l'approche YASA (Chabeb, 2010) :

| Degrés d'appariement | Éléments "Interface" / "Operation" / "Output" | Éléments "Input" |
|----------------------|---|------------------|
| Exacte | 5 | 5 |
| Subsume | 4 | 3 |
| Subsumé par | 3 | 4 |

| | | |
|------------------------|---|---|
| Possède la même classe | 2 | 2 |
| Non classifié | 1 | 1 |
| Échec | 0 | 0 |

Table 4.1 : Table de correspondance "degré d'appariement - valeur" de l'approche YASA (Chabeb, 2010)

Cette table (voir table 4.1) permet d'affecter une valeur d'appariement à chaque couple de concepts comparés. Les couples appariés sont entre un descripteur et la requête, les concepts issus d'ontologie de domaine étant rattachés au même concept de l'ontologie technique. La valeur d'appariement de chaque couple leur est affectée en fonction de la table ci-dessus par rapport à leur degré d'appariement et la position de l'annotation contenant ces concepts dans le descripteur.

L'ensemble des valeurs d'appariement d'un descripteur est ensuite agrégé par un algorithme de calcul de la valeur moyenne d'appariement (voir figure 4.13). Le score final d'appariement d'un descripteur avec une requête est ainsi formé de la valeur moyenne d'appariement du descripteur et de la valeur d'appariement la plus faible obtenue par le descripteur pour l'un de ses concepts.

Après avoir calculé l'appariement de tous les descripteurs présélectionnés sur l'annuaire avec la requête. Les descripteurs sont classés en fonction de leur valeur minimum et moyenne d'appariement.

Par exemple, le descripteur du service de réservation de cours de ski (voir figure 4.8) possède exactement la même annotation dans l'élément XML *interface* pour le concept *interface* de l'ontologie technique que dans la requête formulée par l'utilisateur. La valeur d'appariement 5 est donc affectée à l'élément *interface*. De la même manière, pour l'élément *operation* les annotations du descripteur et de la requête ont un degré d'appariement exact et la valeur 5 est donc affectée à cet élément. La requête ne dispose d'annotations que sur les éléments *interface* et *operation* la valeur minimum d'appariement et la valeur moyenne d'appariement sont de 5 pour ce service.

Concernant le deuxième service donné en exemple (voir figure 4.11), la différence avec le premier service se situe au niveau de l'opération proposée. Dans ce cas, il s'agit de cours de snowboard. Le degré d'appariement entre le concept de l'ontologie de tourisme utilisé dans le descripteur et celui de la requête est de "possède la même classe" car le ski et le snowboard sont tout les deux des "sports d'hiver". Conformément à la table 4.1, la valeur de 2 est donc affectée à l'élément *operation* du descripteur du deuxième service. La valeur minimum d'appariement de ce service est de 2 et sa valeur moyenne est donc de 3.5. L'algorithme suivant permet d'effectuer ces calculs.

```

InterfaceMD=SemanticMatching(rqtInterface,advInterface);
if (InterfaceMD > a){
loop{
OperationMD=SemanticMatching(rqtOperation,advOperation);
if (OperationMD > b)
loop{
InputsMD=SemanticMatching(rqtInputs,advInputs);
OutputsMD=SemanticMatching(rqtOutputs,advOutputs);
}end loop
GlobalOperationMD=(Average(InputsMD,OutputsMD,OperationMD));
OperationsMD= OperationsMD+ GlobalOperationMD
}end loop
return YasaMD=Average(InterfaceMD, (OperationsMD/nbrRequestedOperations));
} else return 0

```

Figure 4.13 : Algorithme de calcul de l'approche YASA pour la valeur moyenne d'appariement

L'algorithme présenté à la figure 4.13 est proposé dans l'approche YASA par (Chabeb, 2010). Il décrit par du pseudo code comment calculer la valeur moyenne d'appariement entre un descripteur et une requête. Après avoir calculé la valeur élémentaire d'appariement de l'interface du descripteur par rapport à la requête et si elle dépasse un seuil a , la valeur de chaque opération est calculée. Pour chaque opération ayant une valeur d'appariement supérieure à un seuil b , l'appariement des entrées et des sorties est calculé. Pour chaque opération, une valeur moyenne est calculée entre les valeurs d'appariement de l'opération, de ses entrées et de ses sorties. Puis, pour l'ensemble des opérations une valeur moyenne est calculée à partir de ces valeurs agrégées de chaque opération, entrées et sorties. La valeur d'appariement moyenne des opérations est ensuite agrégée à la valeur de l'interface par un dernier calcul de moyenne entre les deux valeurs. L'application de cet algorithme sur nos deux descripteurs de service (cours de ski et cours de snowboard) permet de calculer les valeurs d'appariement présentées dans la table suivante.

| | Élément "Interface" | Éléments "Operation" | Minimum | Moyenne | (Minimum, Moyenne) |
|---------------------------------------|------------------------|-------------------------|---------|---------|-----------------------|
| Descripteur cours de ski | 5 | 5 | 5 | 5 | (5, 5) |
| Descripteur cours de snowboard | 5 | 2 | 2 | 3.5 | (2, 3.5) |

Table 4.2 : Exemple de calcul du couple de valeurs minimum - moyenne

Le service de réservation de cours de ski est donc retourné en premier choix à l'utilisateur. Le service de réservation de cours de snowboard pourra néanmoins être proposé à l'utilisateur en second choix, si par exemple les réservations sont complètes pour les cours de ski.

4.3.3 Publication et recherche dans l'annuaire de service méthodologique

Traditionnellement, dans les architectures orientées service nous distinguons deux processus différents :

- La publication des services dans l'annuaire.

- La recherche de services.

Nous allons détailler ces deux activités dans le cas des services méthodologiques.

4.3.3.1. Publication

Le processus de publication est destiné à favoriser l'usage et la réutilisation des services méthodologiques et des lignes de méthodes et il est présenté à la figure 4.14. Il préconise la représentation de ces derniers par un élément offrant un résumé des buts qu'ils réalisent, de leur comportement, de leur structure et de leur situation de réutilisation. Il correspond à l'essence d'un service méthodologique ou d'une ligne de méthode. Il prend la forme d'un descripteur technique et doit avoir des facilités d'opérationnalisation.

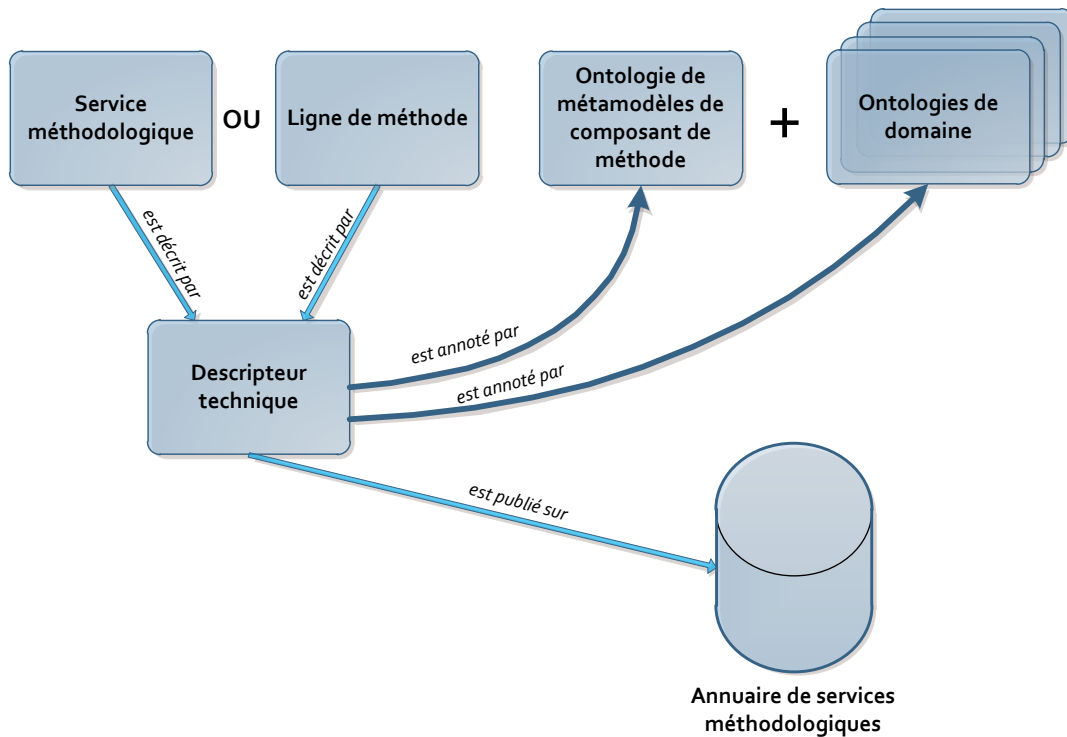


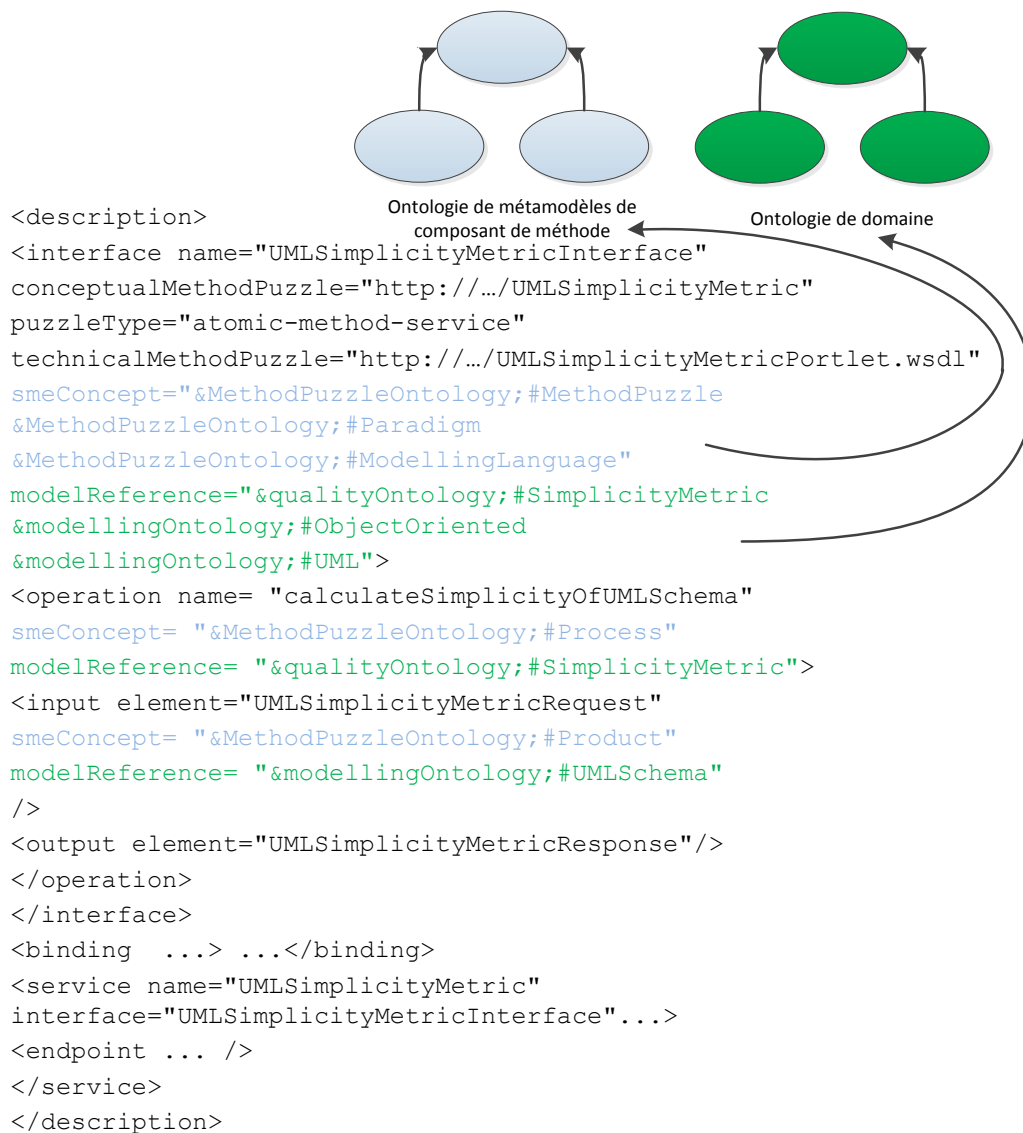
Figure 4.14 : Processus de publication d'un descripteur de SM dans un annuaire MaaS

Les descripteurs techniques, une fois réalisés, sont alors publiés sur un annuaire de services méthodologiques où ils seront stockés. L'étape de publication permet d'exposer les services méthodologiques ou les lignes de méthodes produites au plus grand nombre par l'accessibilité de l'annuaire sur un intranet ou internet.

Les activités de ce processus (voir figure 4.14) sont destinées à être réalisées par un fournisseur de services méthodologiques et de lignes de méthodes avec l'objectif d'offrir une visibilité et une accessibilité sur internet à ces réalisations. Cette grande accessibilité favorise la réutilisation des services méthodologiques et des lignes de méthodes produites par les fournisseurs.

Etant donné qu'un descripteur WSDL contient uniquement des données logiques nécessaires à son invocation et son exécution; dans le but de faciliter la recherche des services méthodologiques, nous proposons d'incorporer des données sémantiques au composant de méthode implémenté par le service. Pour l'implémentation de ce principe en pratique nous avons choisi de baser notre descripteur MaaS sur celui de l'approche YASA, en l'adaptant au domaine de l'IMS. Le descripteur technique de service méthodologique, à l'instar de l'approche YASA, est donc basé sur un descripteur de service web WSDL étendu. De la même manière que pour un descripteur YASA, les éléments principaux d'un descripteur WSDL peuvent être annotés avec des données sémantiques par rapport à des concepts de l'ontologie OMCM (voir figure 4.1). Cela signifie que les éléments interface, opération, entrée et sortie d'un descripteur WSDL d'un service méthodologique peuvent contenir des données sémantiques supplémentaires stockées dans les attributs : *smeconcept* et *modelreference*. (Le schéma XSD de la définition de la structure d'annotation du WSDL est réalisée à la manière de celle utilisée dans le langage SAWSDL (W3C/SAWSDL, 2007) et est disponible dans l'annexe B).

La figure suivante présente un exemple de descripteur de service méthodologique dont certains éléments ont été annotés par les attributs *smeconcept* et *modelreference* contenant respectivement des concepts de l'ontologie OMCM et des concepts d'ontologie de domaine.



Légende :

- **annotations en bleu** : concepts issus de l'ontologie de métamodèles de composant de méthode.
- **annotations en vert**: concepts issus d'une l'ontologie de domaine.

Figure 4.15 : Exemple de descripteur de service méthodologique

La figure 4.15 présente un exemple de descripteur de SM basé sur le calcul de la métrique de simplicité d'un schéma UML proposée par (Si-Said Cherfi, 2002). L'attribut *smeconcept* contient pour un élément donné, une liste des concepts issus de l'ontologie OMCM. L'attribut *modelreference* contient, quant à lui, la liste des concepts sémantiques des ontologies de domaine servant à décrire le composant de méthode supporté par le service méthodologique et ses situations de réutilisation. Comme présenté à la figure 4.15, les concepts présents dans l'attribut *modelreference* sont issus d'un modèle sémantique donné en référence.

L'ordonnement des concepts dans les deux attributs est primordial. Etant donné qu'un concept d'une ontologie de domaine annote un élément du descripteur par rapport à l'ontologie OMCM, l'ordre des concepts dans l'attribut *modelreference* doit donc être réalisé respectivement par rapport à l'ordre des concepts utilisé pour l'attribut *smeconcept*.

Nous proposons également d'ajouter un attribut *conceptualMethodPuzzle* à l'élément *interface* du descripteur WSDL afin de référencer par une URL servant d'identifiant unique, le composant conceptuel supporté par le SM.

Concernant la publication des descripteurs des services méthodologiques et des lignes de méthodes dans l'annuaire, nous proposons d'étendre les fonctionnalités d'un annuaire UDDI (OASIS/UDDI, 2002) afin de prendre en compte les spécificités des descripteurs de services méthodologiques.

Voici un exemple de message contenant une requête de publication sur un annuaire UDDI.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <save_service generic="2.0" xmlns="urn:uddie-org:api_v2">
      <authInfo>***</authInfo>
      <businessService businessKey="****" serviceKey="">
        <name>***</name>
        <description>***</description>
        <bindingTemplates>
          <bindingTemplate bindingKey="">
            <accessPoint URLType="http">***</accessPoint>
            <tModelInstanceDetails>
              <tModelInstanceInfo tModelKey="****">
                <instanceDetails>
                  <overviewDoc>
                    <overviewURL>***</overviewURL>
                  </overviewDoc>
                </instanceDetails>
              </tModelInstanceInfo>
            </tModelInstanceDetails>
          </bindingTemplate>
        </bindingTemplates>
      </businessService>
    </save_service>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 4.16 : Message SOAP de publication d'un descripteur WSDL

La figure 4.16 représente un message SOAP (W3C/SOAP, 2007) envoyé à un annuaire pour la publication d'un service web classique. Nous proposons de rajouter une fonction de publication *save_methodservice* sur un annuaire UDDI. Cette fonction a pour objectif, en plus des fonctionnalités classiques de publication de service, d'extraire les données sémantiques du descripteur de service pour les stocker dans l'annuaire sous la forme de triplets RDF. Cela permet au descripteur du service méthodologique publié d'être apparié par l'algorithme de recherche sémantique avec de futures requêtes.

Voici une figure représentant un exemple d'utilisation de cette fonctionnalité de publication étendue pour les services méthodologiques.

```

<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <save_methodservice generic="2.0" xmlns="urn:uddie-org:api_v2">
      <authInfo>authToken:LK1DA768-42ZI-12CC-6G82-DL745DPZ0X31</authInfo>
      <businessService businessKey="MG59Q4J9-429Q-22FF-B7B1-BFF8D9111D44"
        serviceKey="">
        <name>UMLSimplicityMetric</name>
        <description>Calculates the simplicity quality metric on an UML
          schema</description>
        <bindingTemplates>
          <bindingTemplate bindingKey="">
            <accessPoint URLType="http">http://***/UMLSimplicityMetric
            </accessPoint>
            <tModelInstanceDetails>
              <tModelInstanceInfo tModelKey="uuid:LD8F4KS9-M354-0274-D600-
                94OC63J8FS54">
                <instanceDetails>
                  <overviewDoc>
                    <overviewURL>http://***/UMLSimplicityMetric.wsdl
                    </overviewURL>
                  </overviewDoc>
                </instanceDetails>
              </tModelInstanceInfo>
            </tModelInstanceDetails>
          </bindingTemplate>
        </bindingTemplates>
      </businessService>
    </save_methodservice>
  </soapenv:Body>
</soapenv:Envelope>

```

Figure 4.17 : Exemple de Message SOAP de publication d'un descripteur de SM

La figure 4.17 montre le message SOAP étendu d'un fournisseur de service méthodologique qui sera envoyé à l'annuaire UDDI étendu pour la publication de son service. La couleur bleue du code sur cette figure met en avant les éléments de l'exemple de la fonction de publication *save_methodservice* qui est appelée.

4.3.3.2. Recherche

Le processus de recherche de services méthodologiques ou de lignes de méthode, présenté sur la figure 4.18 ci-dessous, est initié par un acteur ayant pour rôle "utilisateur".

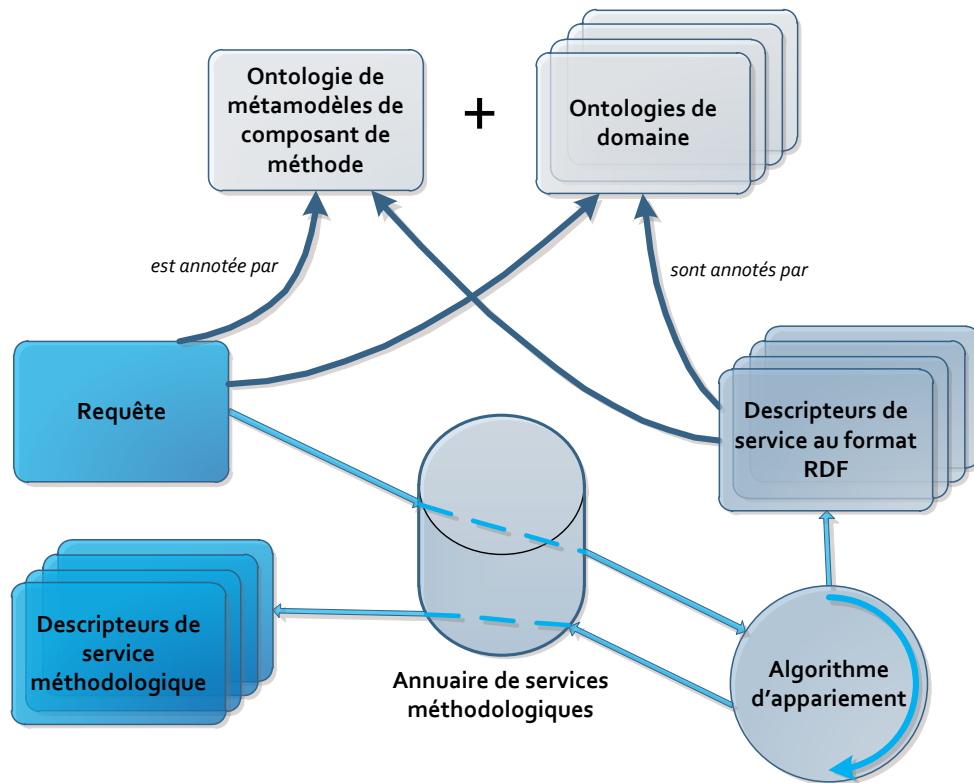


Figure 4.18 : Processus de requête de SM dans un annuaire MaaS

L'utilisateur commence par émettre une requête exprimant les exigences concernant le ou les services qu'il recherche. Ces besoins sont la caractérisation des facteurs de contingence de son projet et de ses préférences utilisateurs.

La requête à envoyer à l'annuaire de services méthodologiques prend la même structure qu'une requête de l'approche YASA. Voici sur la figure suivante, un exemple de requête de service méthodologique sur un annuaire.

```

<description>
<interface name="interfacel"
smeConcept="&MethodPuzzleOntology;#Paradigm
&MethodPuzzleOntology;#ModellingLanguage"
modelReference="modellingOntology;#ObjectOriented
&modellingOntology;#UML">
<operation name= "operation1"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&qualityOntology;#SimplicityMetric">
<input element="inputRequestOp1"
smeConcept= "&MethodPuzzleOntology;#Product"
modelReference= "&modellingOntology;#UMLSchema"
/>
<output element="outputResponseOp1"/>
</operation>

```



```

</interface>
...
<servicename="service1" interface=" interface1"...>
<endpoint ... />
</service>
</description>

```

Légende :

- **annotations en bleu** : concepts issus de l'ontologie de métamodèles de composant de méthode.
- **annotations en vert**: concepts issus d'une l'ontologie de domaine.

Figure 4.19 : Exemple de requête de SM dans l'approche MaaS

Comme le montre la figure 4.19, une requête est construite sur le squelette d'un descripteur WSDL simplifié. La requête doit, dans un premier temps, déclarer l'interface du service désiré puis contenir une description des opérations souhaitées pour ce service avec leurs entrées / sorties. De la même manière qu'un fournisseur de service annote les descripteurs de services méthodologiques, un utilisateur peut annoter avec des données sémantiques les quatre types d'élément constituant la structure de sa requête. Il peut ajouter sur chacun de ces éléments les deux attributs *smeconcept* et *modelreference*. Puis, dans l'objectif de décrire la situation de son projet et les caractéristiques du service méthodologique qu'il recherche, il peut leur affecter respectivement des concepts de l'ontologie OMCM présentée dans ce chapitre et des concepts d'ontologie de domaine.

Une fois que l'utilisateur a construit sa requête, elle est envoyée à l'annuaire, encapsulée dans un message SOAP. D'une manière similaire à l'activité de publication sur l'annuaire, l'annuaire UDDI doit être étendu par une fonctionnalité de recherche supplémentaire liée au domaine de l'IMS. Sur la figure 4.20 ci-dessous, nous pouvons constater la structure du message SOAP encapsulant une requête classique de service web.

```

<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <find_service businessKey="***" generic="2.0" xmlns="urn:uddi-org:api_v2">
      <findQualifiers>
        <findQualifier>***</findQualifier>
      </findQualifiers>
      <name>***</name>
      <categoryBag>
        <keyedReference tModelKey="***" keyName="***" keyValue="***" />
      </categoryBag>
      <tModelBag>
        <tModelKey>***</tModelKey>
      </tModelBag>
    </find_service>
  </soapenv:Body>
</soapenv:Envelope>

```

Figure 4.20 : Message SOAP de requête de service web sur un annuaire UDDI

La figure 4.21 présente, quant à elle, le message SOAP envoyé à l'annuaire de services méthodologiques. La méthode de recherche étendue nommée "find_methodservice" est utilisée à la place de la méthode "find_service" du message classique. Cette fonctionnalité étendue a pour objectif d'interpréter la requête de l'utilisateur pour en extraire les annotations. Puis, en vue d'une opérationnalisation de la requête, une transformation des annotations en un ensemble de triplets RDF est effectuée.

```

<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<find_methodservice businessKey="" generic="2.0"
xmlns="urn:uddi-org:api_v2">
<description>
<interface name="interface1"
smeConcept="&MethodPuzzleOntology;#Paradigm
&MethodPuzzleOntology;#ModellingLanguage"
modelReference="modellingOntology;#ObjectOriented
&modellingOntology;#UML">
<operation name="operation1"
smeConcept=" &MethodPuzzleOntology;#Process"
modelReference=" &qualityOntology;#SimplicityMetric">
<input element="inputRequestOp1"
smeConcept=" &MethodPuzzleOntology;#Product"
modelReference=" &modellingOntology;#UMLSchema"/>
<output element="outputResponseOp1"/>
</operation>
</interface>
...
<service name="service1" interface=" interface1"...>
<endpoint ... />
</service>
</description>
</find_methodservice>
</soapenv:Body>
</soapenv:Envelope>

```

Légende :

- annotations en rouge : éléments de la requête find_methodservice envoyée à l'annuaire
- annotations en bleu : concepts issus de l'ontologie de métamodèles composant de méthode.
- annotations en vert: concepts issus d'une l'ontologie de domaine.

Figure 4.21 : Exemple de message SOAP de requête de SM

L'algorithme d'appariement est ensuite lancé sur l'ensemble des triplets RDF issus de la requête et les triplets RDF issus des descripteurs de services méthodologiques contenus dans l'annuaire. L'algorithme utilisé ici, est celui de l'approche YASA que nous avons présenté à la section 4.4.4 de ce chapitre. A l'issue de son exécution, un ensemble de services est extrait de l'annuaire et est proposé à l'utilisateur. La figure ci-dessous présente un exemple de réponse SOAP de l'annuaire de service méthodologique correspondant à la requête définie à la figure 4.19.

```

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <methodserviceList generic="2.0"
operator="jUDDI.org" xmlns="urn:uddi-org:api_v2">
      <methodserviceInfos>
        <methodserviceInfo businessKey="MG59Q4J9-429Q-22FF-B7B1-BFF8D9111D44"
serviceKey="987FLK34-02J8-22EE-0D5F-LD8Q745DBV78">
          <name>UMLSimplicityMetric</name>
          <conceptualDescriptor>http://.../UMLSimplicityMetric
</conceptualDescriptor>
          <technicalDescriptor>http://***/UMLSimplicityMetric.wsdl
</technicalDescriptor>
          <matching min="5" average="5">
            <interface name="UMLSimplicityMetricInterface" value="5">
              <operation name="calculateSimplicityOfUMLSchema" value="5">
                <input name="UMLSimplicityMetricRequest" value="5"/>
              </operation>
            </interface>
          </matching>
        </methodserviceInfo>
      </methodserviceInfos>
    </methodserviceList>
  </soapenv:Body>
</soapenv:Envelope>

```

Figure 4.22 : Exemple réponse SOAP à la recherche de SM

L'envoi de la liste des résultats contenant les services méthodologiques sélectionnés à l'utilisateur est réalisé par l'annuaire au moyen d'un message SOAP classique contenant la liste des services. La figure 4.22 reprend la structure de ce type de réponse envoyée aux utilisateurs. Concernant cette fonctionnalité de l'annuaire de services méthodologiques, seule l'adaptation de la construction de la liste des résultats au contexte de l'IMS est requise (la partie du code en bleu sur la figure). Les fonctionnalités d'envoi de réponse d'un annuaire UDDI peuvent être utilisées telles qu'elles sont définies pour l'envoi des résultats de la sélection des services méthodologiques.

4.3.3.3. Une architecture de méthode orientée service

L'approche MaaS s'inscrit dans le cadre du paradigme de l'AOS appliqué à l'ingénierie des méthodes. Elle est basée sur une extension des principes de fonctionnement d'une architecture orientée service adaptés au domaine de l'ingénierie des méthodes situationnelles. Comme nous pouvons le constater aux sections 4.4.5.1 et 4.4.5.2 de ce chapitre, les processus de publication et de recherche de services méthodologiques présentent des extensions de l'AOS nécessaires à son application dans le domaine de l'IMS.

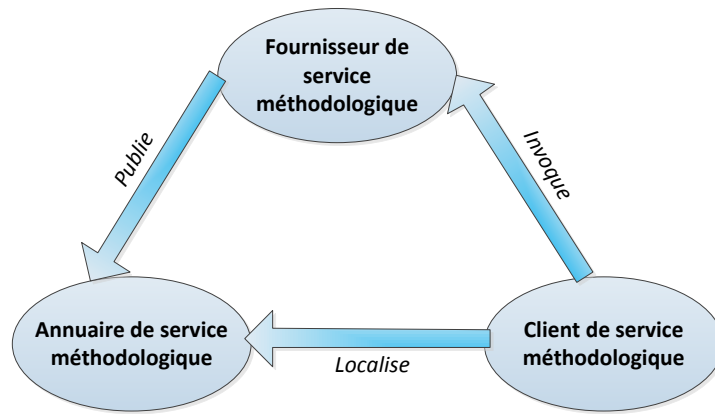


Figure 4.23 : L'architecture MOS

Nous proposons dans l'approche MaaS d'étendre l'architecture orientée service par les spécificités du domaine de l'IMS pour former une architecture de méthode orientée service (MOS) comme le montre la figure 4.23.

Au sein de cette nouvelle architecture, un fournisseur de méthode développe des services méthodologiques, des lignes méthodes et leur descripteur en accord avec le processus que nous avons présenté dans ce chapitre. Il les publie ensuite sur un annuaire de service étendu en tenant compte des spécificités des services méthodologiques par rapport aux services web classique. L'objectif est d'étendre les standards existants, afin d'adapter l'AOS au domaine de l'IMS afin de garantir une interopérabilité entre les services méthodologiques basés sur des composants de méthode issus d'approche d'IMS différentes.

Un utilisateur de méthodes peut alors accéder à l'annuaire afin de rechercher des services méthodologique correspondant à ces exigences. L'utilisateur peut alors sélectionner un ou plusieurs services méthodologiques pour les incorporer à ses méthodes. Il peut également sélectionner des lignes de méthodes complètes et les configurer en fonction des facteurs de contingence de ses projets.

Une fois les services méthodologiques et/ou les lignes de méthodes sélectionnés, un utilisateur peut les invoquer afin de profiter du support outillé qu'ils offrent et exécuter le processus du composant de méthode qu'ils contiennent.

4.3.4 Exemple de recherche de service méthodologique

Prenons par exemple trois services méthodologiques proposant de calculer des métriques de qualité de diagrammes de classes UML tel que proposé dans (Si-Said Cherfi, 2002) : un SM supportant le calcul de la métrique d'efficacité des liens d'héritage, un SM supportant le calcul de la métrique d'efficacité des liens d'agrégation et une ligne de méthode regroupant l'ensemble des métriques. Les trois figures suivantes contiennent des exemples de descripteurs de ces trois services méthodologiques de calcul de métriques de qualité dans un schéma UML.

```

<description>
<interface name="UMLInheritanceEffectivenessMetricInterface"
conceptualMethodPuzzle="http://.../UMLInheritanceEffectivenessMetric"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../
UMLInheritanceEffectivenessMetricPortlet.wsdl"
smeConcept="&MethodPuzzleOntology;#MethodPuzzle
&MethodPuzzleOntology;#Paradigm
&MethodPuzzleOntology;#ModellingLanguage"
modelReference="&qualityOntology;#InheritanceEffectivenessMetric
&modellingOntology;#ObjectOriented
&modellingOntology;#UML">
<operation name= "calculateInheritanceEffectivenessOfUMLSchema"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&qualityOntology;#InheritanceEffectivenessMetric">
<input element="UMLInheritanceEffectivenessMetricRequest"
smeConcept= "&MethodPuzzleOntology;#Product"
modelReference= "&modellingOntology;#UMLSchema"/>
<output element="UMLInheritanceEffectivenessMetricResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="UMLInheritanceEffectivenessMetric"
interface="UMLInheritanceEffectivenessMetricInterface"...>
<endpoint ... />
</service>
</description>

```

Légende :

- **annotations en bleu** : concepts issus de l'ontologie de métamodèles de composant de méthode.
- **annotations en vert**: concepts issus d'une l'ontologie de domaine.

Figure 4.24 : Exemple de descripteur de SM de calcul d'efficacité des liens d'héritage

```

<description>
<interface name="UMLAggregationEffectivenessMetricInterface"
conceptualMethodPuzzle="http://.../UMLAggregationEffectivenessMetric"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../
UMLAggregationEffectivenessMetricPortlet.wsdl"
smeConcept="&MethodPuzzleOntology;#MethodPuzzle
&MethodPuzzleOntology;#Paradigm
&MethodPuzzleOntology;#ModellingLanguage"
modelReference="&qualityOntology;#AggregationEffectivenessMetric
&modellingOntology;#ObjectOriented
&modellingOntology;#UML">
<operation name="calculateAggregationEffectivenessOfUMLSchema"
smeConcept=" &MethodPuzzleOntology;#Process"
modelReference=" &qualityOntology;#AggregationEffectivenessMetric">
<input element="UMLAggregationEffectivenessMetricRequest"
smeConcept=" &MethodPuzzleOntology;#Product"
modelReference=" &modellingOntology;#UMLSchema"/>
<output element="UMLAggregationEffectivenessMetricResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="UMLAggregationEffectivenessMetric"
interface="UMLAggregationEffectivenessMetricInterface"...>
<endpoint ... />
</service>
</description>

```

Légende :

- **annotations en bleu** : concepts issus de l'ontologie de métamodèles de composant de méthode.
- **annotations en vert**: concepts issus d'une l'ontologie de domaine.

Figure 4.25 : Exemple de descripteur de SM de calcul d'efficacité des liens d'agrégation

```

<description>
<interface name="UMLMetricsInterface"
conceptualMethodPuzzle="http://.../UMLMetrics"
puzzleType="method-line"
technicalMethodPuzzle="http://.../UMLMetricsMethodLine.xml"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#Paradigm
&MethodPuzzleOntology;#ModellingLanguage"
modelReference="&qualityOntology;#UMLMetrics
&modellingOntology;#ObjectOriented
&modellingOntology;#UML">
<operation name= "calculateUMLMetrics"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&qualityOntology;#UMLMetrics">
<input element="UMLMetricsRequest"
smeConcept= "&MethodPuzzleOntology;#Product"
modelReference= "&modellingOntology;#UMLSchema"/>
<output element="UMLMetricsResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="UMLMetrics" interface="UMLMetricsInterface"...>
<endpoint ... />
</service>
</description>

```

Légende :

- **annotations en bleu** : concepts issus de l'ontologie de métamodèles de composant de méthode.
- **annotations en vert**: concepts issus d'une l'ontologie de domaine.

Figure 4.26 : Exemple de descripteur de SM de calcul de métriques de qualité des diagrammes de classes UML

Les trois figures 4.24, 4.25 et 4.26 présentent les descripteurs de ces trois SM qui sont alors publiés dans l'annuaire de services méthodologiques. Un utilisateur peut ensuite formuler une requête pour rechercher par exemple, un SM de calcul de l'efficacité des liens d'héritage pour vérifier si son diagramme de classe UML nécessite des améliorations dans la gestion des hiérarchies de classes. Cette requête prend la forme présentée à la figure 4.27.

```

<description>
<interface name="interfacel"
smeConcept="&MethodPuzzleOntology;#Paradigm
&MethodPuzzleOntology;#ModellingLanguage"
modelReference="modellingOntology;#ObjectOriented
&modellingOntology;#UML">
<operation name= "operation1"
smeConcept= "&MethodPuzzleOntology;#Process"

```

```

modelReference= "&qualityOntology;#InheritanceEffectivenessMetric">
<input element="inputRequestOp1"
smeConcept= "&MethodPuzzleOntology;#Product"
modelReference= "&modellingOntology;#UMLSchema"
/>
<output element="outputResponseOp1"/>
</operation>
</interface>
...
<servicename="service1" interface=" interface1"...>
<endpoint ... />
</service>
</description>

```

Légende :

- **annotations en bleu** : concepts issus de l'ontologie de métamodèles de composant de méthode.
- **annotations en vert**: concepts issus d'une l'ontologie de domaine.

Figure 4.27 : Exemple de requête de recherche de SM de calcul d'efficacité des liens d'héritage

Après l'étape de présélection des descripteurs de SM dans l'annuaire, conformément à la table d'affectation des valeurs pour l'évaluation de l'appariement élémentaire (voir table 4.1), toutes les annotations de la requête de l'utilisateur sont couplées avec celles présentes dans les descripteurs. Ensuite conformément à l'algorithme présenté sur la figure 4.13 la valeur moyenne d'appariement de chaque descripteur est calculée comme présenté à la table suivante.

| | Élément "Interface" | Éléments "Operation" | Éléments "Input" | Minimum | Moyenne | (Minimum, Moyenne) |
|---|------------------------|-------------------------|---------------------|---------|---------|-----------------------|
| MS de qualité des liens d'héritages | 5 | 5 | 5 | 5 | 5 | (5, 5) |
| Ligne de méthode de qualité des diagrammes UML | 5 | 3 | 5 | 3 | 4.5 | (3, 4.5) |
| MS de qualité des liens d'agrégation | 5 | 2 | 5 | 2 | 4.25 | (2, 4.25) |

Table 4.3 : Calcul du couple de valeurs minimum - moyenne pour les trois exemples de service

Par rapport à l'annotation utilisée dans la requête sur l'élément *interface* avec les concepts de *paradigm* et *modellinglanguage*, les trois services possèdent des annotations identiques pour ces concepts sur cet élément de leurs descripteurs. Par conséquent, l'appariement élémentaire des annotations des descripteurs de service avec l'annotation donnée par l'utilisateur dans la requête est "exact". Une valeur d'appariement de 5 est donc affectée à l'interface de ces trois services (voir table 4.3). Il en va de même pour l'élément "input" de la requête et des descripteurs des trois services qui sont identiques. La

différence entre les descripteurs de ces services se situe sur l'élément *operation*. Le service de calcul de la métrique d'efficacité des liens d'héritage a un degré d'appariement "exact" pour cet élément. Concernant la ligne de méthode de qualité des diagrammes de classes UML, comme elle englobe l'ensemble des métriques, le concept de l'ontologie de domaine utilisé dans la requête est par exemple subsumé par celui utilisé dans le descripteur car la ligne de méthode rend un service plus générique que celui recherché par l'utilisateur. Une valeur d'appariement de 3 est donc affectée à la ligne de méthode pour l'élément *operation*. Enfin, le service de calcul de la métrique d'efficacité des liens d'agrégation a un degré d'appariement de "possède la même classe" car les métriques appartiennent à la même famille de métrique de qualité dans l'ontologie de domaine de la qualité des schémas conceptuels. Une valeur d'appariement de 2 est affectée à l'élément *operation* de ce service.

Les résultats retournés à l'utilisateur pour sa requête dans cet exemple sont constitués de la liste des trois services dans l'ordre suivant :

- Premièrement, le service de calcul de la métrique d'efficacité des liens d'héritage car il correspond exactement à la requête formulée par l'utilisateur.
- Ensuite, la ligne de méthode des métriques de qualité des diagrammes de classe UML car elle englobe la métrique de qualité que recherche l'utilisateur.
- Enfin le service de calcul de la métrique d'efficacité des liens d'agrégation car il ne satisfait pas la demande de l'utilisateur mais, comme il propose le calcul d'une métrique de la même famille que celle recherchée par l'utilisateur, ce service est susceptible de l'intéresser.

4.4. Conclusion

Dans ce chapitre nous avons présenté une ontologie de métamodèles de composant de méthode en fonction des cinq aspects de la définition d'une méthode de Seligmann (Seligmann, 1989) (paradigme, produit, processus, organisationnel et outil). L'ensemble de la communauté du domaine de l'IMS s'accordant sur cette définition, ajouté au fait que l'ensemble des concepts des métamodèles de composant des approches d'IMS peuvent être corrélés sémantiquement à partir de l'ontologie et que tous les éléments identifiés dans l'ontologie trouvent au moins une spécialisation dans les approche d'IMS font que cette ontologie définit une base commune pour le domaine d'IMS.

L'ontologie OMCM est utilisée dans l'approche MaaS pour définir les descripteurs des services méthodologiques (son implémentation dans le langage de définition d'ontologie OWL est disponible à l'annexe A). En effet, les SM sont les éléments de base de notre approche et nous proposons de construire leurs descripteurs par un ensemble de couples de concepts constitués des concepts identifiés dans l'ontologie OMCM liés à des concepts provenant d'ontologies de domaine. Ce principe permet d'extraire des informations contenues dans un composant de méthode conceptuel en les indexant par

rapport à notre ontologie dans le descripteur du SM. Cela permet d'avoir une manière de décrire les composants de méthode commune à toutes les approches et, par conséquent, de rechercher ou de publier sur un annuaire des SM ou des composants de méthode indépendamment de l'approche dont ils sont issus. Dans le cadre de l'application du paradigme de service au domaine de l'IMS, nous avons choisi d'implémenter les descripteurs de SM sous la forme d'annotation sémantique dans le descripteur technique du SM. Le descripteur, dans notre cas, est conforme au standard WSDL car la technique des services web SOAP est choisie comme environnement d'implémentation. Une autre implémentation est envisageable et dans ce cas l'exploitation de l'ontologie est la même mais intégrée à un langage de programmation différent. Cette manière commune de décrire, publier, requêter et rechercher des composants nous permet d'accueillir dans une base commune de services méthodologiques (un annuaire de SM) des composants issus d'approches différentes.

L'utilisation d'une ontologie de métamodèles de composant de méthode pour la description des SM permet d'utiliser les dernières techniques de recherche sémantique de services telle que l'approche YASA (Chabeb, 2008), (Chabeb, 2010) pour fournir aux utilisateurs un système de publication et de recherche performant et efficace sur l'annuaire de SM. En effet, l'approche YASA permet d'avoir une implémentation indépendante du langage de description, de l'extension sémantique et du type d'ontologie.

La solution présentée dans ce chapitre permet de satisfaire l'objectif de construction d'un socle commun au domaine de l'IMS à travers un annuaire ayant des fonctionnalités de publication et de recherche sémantique indispensable à l'orientation service.

Chapitre 5. Métamodèles de service méthodologique

5.1. Introduction

Après avoir décrit dans le chapitre précédent le fonctionnement de l'annuaire de services méthodologiques de l'approche MaaS, nous proposons de nous focaliser sur l'élément central de cette approche, le service méthodologique. Dans ce chapitre, nous décrivons les deux types de services méthodologiques à travers leur métamodèle et leur opérationnalisation : le service méthodologique atomique (SMA) et la ligne de méthode (LM). Nous concluons ce chapitre en définissant le processus d'utilisation des services méthodologiques en entreprise selon plusieurs niveaux de maturité pour une meilleure intégration.

Les services méthodologiques ont une orientation axée sur l'aspect outil des méthodes, ce qui les amène à être considérés comme des *composants de méthode techniques*. Ils sont destinés à outiller les méthodes en fournissant un module logiciel et standard pour l'implémentation des *composants de méthode dits conceptuels*. Les services méthodologiques représentent notre réponse au manque de support outillé dans les méthodes issues d'approches d'IMS. Ce sont des composants exécutables et standardisés. Le concept de service méthodologique nous permet de dériver deux nouveaux concepts : les services méthodologiques atomiques et les lignes de méthode.

Les services méthodologiques atomiques (SMA) sont une implémentation technique d'un composant de méthode conceptuel sous la forme d'un service ou d'une composition statique de services. La réalisation de ce type de composant exécutable a nécessité d'outrepasser un certain nombre de verrous technologiques. En effet, un service atomique n'est pas un simple service classique, c'est un *Composant de processus technique*. En d'autres termes, il représente une implémentation possible d'un *Composant de processus conceptuel*. Ces entrées/sorties sont, quant à elles, des *Composants de produit techniques*, qui sont elles mêmes une implémentation de *Composants de produit conceptuels*.

Les lignes de méthode (LM) permettent de représenter le concept de variabilité dans le processus d'une méthode. En effet, elles modélisent les différents points de variation articulant les services méthodologiques atomiques entrant dans la composition d'un *Paquetage méthodologique* ou d'une *Méthode*. Elles permettent de configurer une méthode au début ou pendant le déroulement d'un projet en modélisant les différentes alternatives de choix présentes dans celle-ci. Les lignes de méthode sont une proposition de solution aux problèmes des entreprises pour la configuration et l'adaptation de leurs méthodes.

Dans l'objectif de favoriser l'utilisation des services méthodologiques dans une entreprise, nous proposons d'adapter l'approche CMMI (CMMI, 2006) aux approches d'IMS. En effet, elle introduit plusieurs niveaux de maturité des processus pour leur intégration incrémentale dans une entreprise.

Nous proposons plusieurs niveaux d'application de la démarche MaaS en correspondance avec les cinq niveaux de maturité de l'approche CMMI.

Ce chapitre s'organise en trois parties. Dans un premier temps, nous présentons les SMA et leurs enjeux. Puis, nous définissons les LM ainsi que le processus permettant de les créer et de les configurer. Enfin, nous proposons un processus progressif pour l'utilisation des services méthodologiques et l'intégration de l'approche MaaS en général dans une entreprise.

5.2. Support outillé pour les composants de méthode

Avant de présenter le métamodèle de service méthodologique atomique, nous expliquons quels sont les verrous technologiques qu'il faut lever pour pouvoir réaliser une architecture orientée service d'une méthode jusqu'à son opérationnalisation à travers un outil.

5.2.1 Verrous technologiques liés à l'architecture orientée service d'une méthode

L'objectif principal de notre solution est l'amélioration de l'usage de l'IMS dans les entreprises tel que nous l'avons défini dans la problématique de cette thèse. Notre hypothèse pour réaliser cet objectif est d'appliquer le paradigme des AOS au domaine de l'IMS. Le concept de service méthodologique est le résultat de cette application au niveau des composants de méthode.

Comme nous l'avons montré dans notre état de l'art présenté au chapitre 2 de cette thèse, l'aspect outil n'est pas pris en compte lors de l'adaptation des méthodes par les approches d'IMS actuelles. De plus, les approches, elles-mêmes, souffrent d'un manque d'outil de support nécessaire à leur application. Nous proposons, pour combler ce manque, une approche d'IMS axée sur le support outillé des méthodes afin de faciliter leur usage dans les entreprises. Pour ce faire, nous nous intéressons donc à l'élément central des approches d'IMS, le composant de méthode. Par analogie aux services issus du paradigme des AOS, nous proposons de définir un type de service adapté au domaine de l'ingénierie des méthodes. Ce nouveau type de service, que nous appelons "service méthodologique", est défini comme le support outillé d'un composant de méthode. Un service méthodologique est un composant "technique" qui représente l'aspect outil d'un composant de méthode conceptuel d'une approche d'IMS prenant en compte à minima l'aspect processus d'une méthode. Les composants de méthode peuvent être définis en fonction des approches d'IMS à des niveaux de granularité différents. Nous nous intéressons, dans un premier temps, aux composants de méthode ayant une granularité basse avec les SMA. Nous présentons dans la section 5.3 de ce chapitre les lignes de méthode qui sont associées aux composants de méthode à haute granularité.

La réalisation de l'application du paradigme des AOS au domaine de l'IMS pour la définition des SMA nécessite au préalable de résoudre plusieurs verrous technologiques (voir la figure 5.1). Ils sont liés à

des spécificités indispensables au domaine de l'IMS et la transcription de ces spécificités dans le paradigme des AOS.

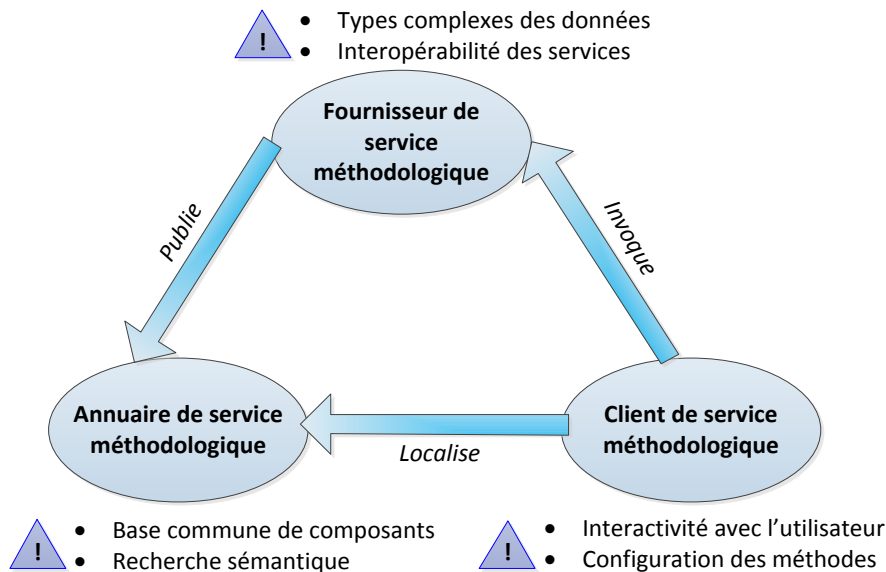


Figure 5.1 : Verrous technologiques liés à l'application du paradigme des AOS au domaine de l'IMS

La figure 5.1 présente du point de vue des acteurs d'une architecture MOS, les différents verrous technologiques à lever pour définir un métamodèle de SMA opérationnel. Les six sous-sections suivantes détaillent chacun des six verrous et les solutions que nous proposons pour y répondre.

5.2.1.1. Verrou : interopérabilité des services méthodologiques

En commençant par la production des SMA, un fournisseur de méthode et de services méthodologiques est confronté aux problèmes de réutilisation des services qu'il produit et par conséquent, au problème d'interopérabilité avec les services existants. En effet, les composants de méthode issus des approches existantes souffrent de ces problèmes d'interopérabilité. La création d'un support outillé pour les composants de méthode ne doit pas souffrir de cette lacune. L'application du paradigme des AOS à la construction des SMA est une solution pour fournir une interopérabilité aux outils de support de l'IMS. Cependant, les adaptations nécessaires à la transposition du paradigme des AOS au domaine de l'IMS ne doivent donc pas être contraires au principe d'interopérabilité entre les SMA. Pour cela, nous proposons d'utiliser les standards des solutions des AOS pour la construction des SMA. Nous proposons également que les adaptations nécessaires aux spécificités du domaine de l'IMS soient réalisées sous la forme d'extension des standards des AOS existants. Les SMA devront être basés sur le modèle des services de l'architecture AOS. Cela signifie, comme nous l'avons présenté au chapitre 4 de cette thèse, qu'ils doivent être décrits par un descripteur conforme au standard WSDL (W3C/WSDL, 2001) et publiés dans un annuaire basé sur le standard UDDI (OASIS/UDDI, 2002). La communication entre les différents composants et acteurs de cette

architecture orientée méthode doit être réalisée au moyen de messages conformes au standard SOAP (W3C/SOAP, 2007).

5.2.1.2. Verrou : types de données complexes

La première adaptation des standards des AOS existants concerne le fournisseur de services méthodologiques et la représentation des données d'entrée et de sortie des SMA. En effet, les données consommées ou produites par les composants de méthode sont des produits complexes à part entière. Les produits manipulés dans une approche d'IMS ont une représentation à plusieurs niveaux comme le montre la figure 5.2.

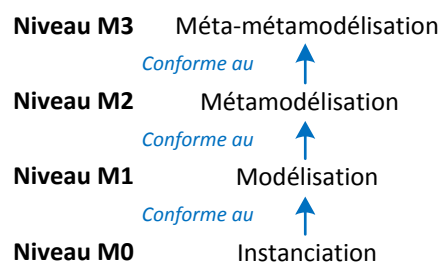


Figure 5.2 : Les différents niveaux de modélisation

D'un point de vue conceptuel, les produits sont représentés par des schémas conformes à un métamodèle. Ce métamodèle est, lui-même, un langage de modélisation d'un paradigme donné. Un schéma peut également être instancié et utilisé à un niveau opérationnel (voir figure 5.2).

Cette représentation en multi-niveaux de modélisation des données d'entrée et de sortie des composants de méthode n'a pas de correspondance établie dans une architecture AOS. Les types de données classiques manipulées par les services ne permettent pas de prendre en compte ces différents niveaux de représentation. Nous proposons pour notre solution d'utiliser un standard existant basé sur une représentation des produits à plusieurs niveaux de modélisation : le standard XMI (OMG/XMI, 2011).

XMI est un standard défini par l'OMG, il se base sur le langage XML pour la représentation de métadonnées en conformité avec un métamodèle donné. Le Meta-Object Facility ou MOF (OMG/MOF, 2006), un autre standard du groupe OMG permettant la représentation de métamodèles, est utilisé pour la définition du métamodèle utilisé dans un document XMI. Le standard XMI est utilisé pour l'échange de modèles. Actuellement, son utilisation est le plus fréquemment cantonnée à des schémas représentés dans le langage de modélisation UML (OMG/UML, 2011), même si XMI permet la représentation de n'importe quel modèle dont le métamodèle a été préalablement modélisé avec MOF.

La figure suivante présente la correspondance entre les différents niveaux de modélisation et leurs correspondances en MOF/XMI.

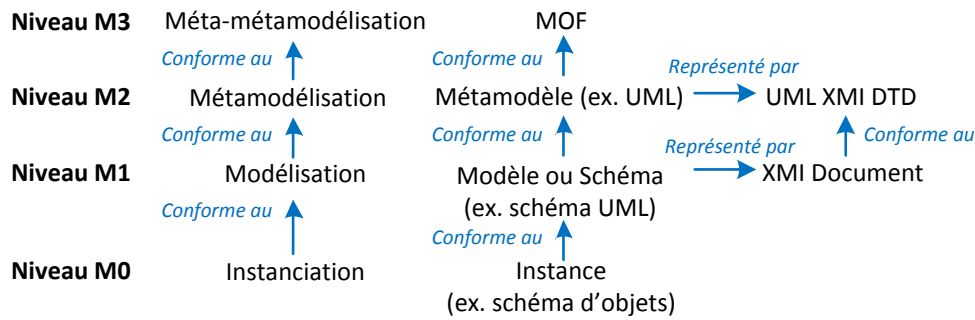


Figure 5.3 : Le standard XMI appliqué aux différents niveaux de modélisation.

A la figure 5.3 sont représentés les différents niveaux de modélisation utilisés dans les approches d'IMS avec leurs correspondances d'un point de vue de l'implémentation technique avec les standards XMI et MOF. Le niveau de modélisation le plus haut est le niveau M3 de méta-métamodélisation, c'est un niveau auto-descriptif permettant de modéliser, au niveau inférieur, dit M2, des paradigmes sous la forme de métamodèles. D'un point de vue technique, le niveau M3 est pris en compte par MOF. Le niveau M2 supporte la définition des métamodèles, au niveau de l'implémentation cela se traduit dans notre solution par la définition du métamodèle en MOF qui est utilisé pour la réalisation de schémas XMI. En d'autres termes, le niveau M2 prendra en compte la définition de la DTD d'un langage de modélisation utilisé dans un document XMI. Prenons un langage de modélisation dans le paradigme orienté objet comme le langage UML, le niveau M2 est, dans cet exemple, le métamodèle du langage UML et son implémentation technique est la définition en MOF de ce métamodèle UML servant de DTD pour la réalisation de document XMI représentant des modèles UML. Les modèles construits conformément à ces métamodèles sont au niveau M1, le niveau des modèles ou des schémas. Ils sont vus comme une représentation du monde réel au travers du prisme d'un métamodèle. D'un point de vue technique, ce niveau est pris en compte par les documents XMI permettant de représenter et de partager des modèles. Le dernier niveau, le niveau M0, est le niveau des instances. Pour l'implémentation de notre solution, cela représente l'instanciation d'un schéma contenu dans un document en objets opérationnels. La manipulation de ces objets par un algorithme permet d'effectuer les opérations et les transformations impliquées par l'application d'un processus d'un composant de méthode sur un modèle.

Prenons, par exemple un SMA simple dont l'objectif est d'améliorer les associations d'un diagramme de classes dont l'une des cardinalités est 0 - n (voir les figures ci-dessous).

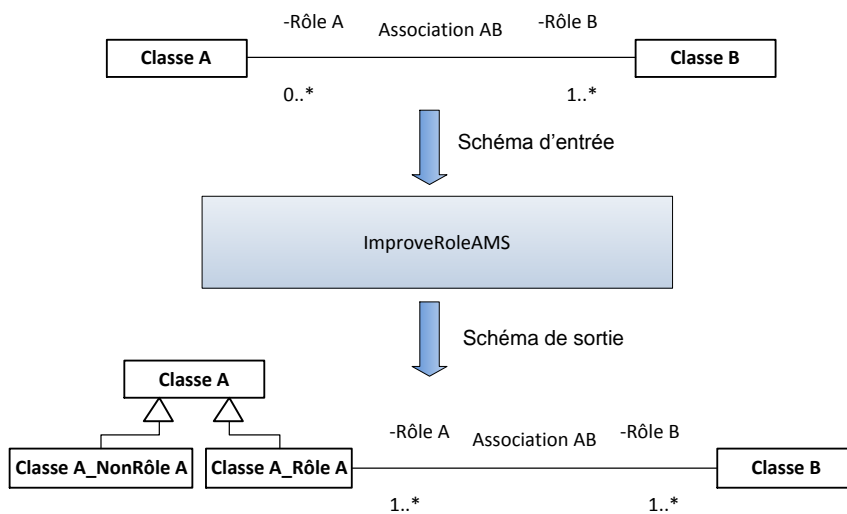


Figure 5.4 : Exemple de schéma conceptuel en entrée et en sortie d'un SMA

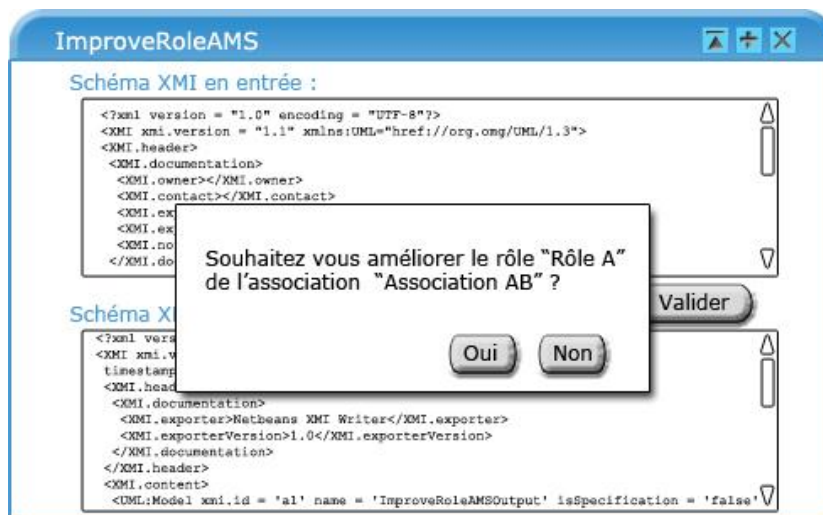


Figure 5.5 : Maquette de l'interface du SMA ImproveRoleAMS

Dans l'exemple des figure 5.4 et 5.5, le SMA nommé *ImproveRoleAMS* prend un schéma UML en entrée (n'importe quel diagramme de classe) et teste la présence d'associations avec au moins un rôle de cardinalité 0-n. Pour chaque association détectée, le SMA demande à l'utilisateur s'il souhaite améliorer le rôle de l'association. Lorsque cela est le cas, le SMA crée alors deux classes spécialisées de classe porteuse de ce rôle, l'une sans le rôle et l'autre avec le rôle dans l'association mais avec cette fois une cardinalité 1-n. Sur l'exemple présenté à la figure 5.4, c'est la classe *Classe A* porteuse du rôle *Rôle A* dans l'association *Association AB* qui possède une cardinalité 0-n pour ce rôle dans cette association. Cette classe est donc transformée par le SMA pour donner dans le schéma en sortie une classe *Classe A* avec ses deux classes spécialisées *Classe A_NonRôleA* et *Classe A_Rôle A*. La *Classe A_NonRôleA* correspond à la cardinalité 0 et donc n'intervient pas dans l'association avec la classe

Classe B. La Classe A_Rôle A est désormais la classe intervenant dans l'Association AB avec le Rôle A. Elle possède donc désormais une cardinalité 1-n pour ce rôle dans cette association.

Le SMA de cet exemple prend en entrée et délivre en sortie un schéma UML qui est par conséquent décrit sous la forme d'un document XMI. Les deux figures suivantes représentent les documents XMI correspondants au schéma UML de cet exemple de SMA.

```

<XMI xmi.version = "1.2" xmlns:UML="'org.omg.xmi.namespace.UML">
<XMI.header>
...
</XMI.header>
<XMI.content>
<UML:Model xmi.id="ImproveRoleAMSInput">
<UML:Namespace.ownedElement>
<UML:Model xmi.id="UMLModel.2" name="InputSchema" ...>
<UML:Namespace.ownedElement>
<UML:Class xmi.id="UMLClass.3" name="Classe A" visibility="public"
isSpecification="false" namespace="UMLModel.2" isRoot="false" isLeaf="false"
isAbstract="false" participant="UMLAssociationEnd.6" isActive="false"/>
<UML:Class xmi.id="UMLClass.4" name="Classe B" .../>
<UML:Association xmi.id="UMLAssociation.5" name="Association AB" ...>
<UML:Association.connection>
<UML:AssociationEnd xmi.id="UMLAssociationEnd.6" name="Rôle A"
visibility="public" isSpecification="false" isNavigable="false"
ordering="unordered" aggregation="none" targetScope="instance"
changeability="changeable" association="UMLAssociation.5"
participant="UMLClass.3">
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity xmi.id="X.8">
<UML:Multiplicity.range>
<UML:MultiplicityRange xmi.id="X.9" lower="0"upper="-1"
multiplicity="X.8"/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id="UMLAssociationEnd.7" name="Rôle B" ...>
...
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
</UML:Namespace.ownedElement>
</UML:Model>
</UML:Namespace.ownedElement>
</UML:Model>
</XMI.content>
</XMI>

```

Figure 5.6 : Extrait de l'exemple de document XMI d'entrée du SMA ImproveRoleAMS

La définition de la Classe A apparaît en bleu à la figure 5.6 du document XMI en entrée du service *ImproveRoleAMS* et en vert la définition de son rôle dans l'association *Association AB* avec une cardinalité 0-n par le rôle *Role A*. Une association est composée de deux rôles (*AssociationEnd* en XMI) exprimant la participation d'une classe à l'association (dans cet exemple le rôle *Role A* associé à la classe *Classe A*).

```

<XMI xmi.version = '1.2' xmlns:UML = 'org.omg.xmi.namespace.UML' ...>
<XMI.header>
...
</XMI.header>

```

```

<XMI.content>
<UML:Model xmi.id = 'a1' name = 'ImproveRoleAMSOutput' ...>
<UML:Namespace.ownedElement>
<UML:Class xmi.id = 'a2' name = 'Classe A_NonRôle A' ...>
<UML:GeneralizableElement.generalization>
<UML:Generalization xmi.idref = 'a3'/>
</UML:GeneralizableElement.generalization>
</UML:Class>
<UML:Class xmi.id = 'a4' name = 'Classe A_Rôle ...>
<UML:GeneralizableElement.generalization>
<UML:Generalization xmi.idref = 'a5'/>
</UML:GeneralizableElement.generalization>
</UML:Class>
<UML:Association xmi.id = 'a6' name = 'Association AB' ...>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = 'a7' name = 'Rôle A'...>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity xmi.id = 'a8'>
<UML:Multiplicity.range>
<UML:MultiplicityRange xmi.id = 'a9' lower = '1' upper = '-1'/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Class xmi.idref = 'a4'/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = 'a10' name = 'Rôle B' ...>
...
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<UML:Class xmi.id = 'a13' name = 'Classe B' .../>
<UML:Class xmi.id = 'a14' name = 'Classe A' .../>
</UML:Namespace.ownedElement>
</UML:Model>
<UML:Generalization xmi.id = 'a3' isSpecification = 'false'>
<UML:Generalization.child>
<UML:Class xmi.idref = 'a2'/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:Class xmi.idref = 'a14'/>
</UML:Generalization.parent>
</UML:Generalization>
<UML:Generalization xmi.id = 'a5' isSpecification = 'false'>
<UML:Generalization.child>
<UML:Class xmi.idref = 'a4'/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:Class xmi.idref = 'a14'/>
</UML:Generalization.parent>
</UML:Generalization>
</XMI.content>
</XMI>

```

Figure 5.7 : Extrait du document XMI en sortie du SMA *ImproveRoleAMS*

La figure 5.7 représente un exemple de document XMI en sortie du SMA *ImproveRoleAMS* après la transformation de la *Classe A* et de son rôle dans l'*Association AB*. Nous mentionnons en bleu sur le document que les définitions des deux classes *Classe A_NonRôleA* et *Classe A_Rôle A* ont été ajoutées et que la cardinalité pour le *Rôle A* (en vert sur le schéma) est maintenant de 1-n. Il est également fait référence dans la définition de cette terminaison d'association que désormais c'est la classe *Classe A_Rôle A* qui est porteuse du rôle *Rôle A* dans l'association (en bleu sur le schéma). Les définitions des

liens de spécialisation pour la *Classe A* et ses deux classes filles *Classe A_NonRôleA* et *Classe A_RôleA* ont été ajoutées au schéma XMI (en rouge à la figure 5.7).

L'utilisation des standards XMI et MOF, pour l'opérationnalisation des échanges de données complexes et du standard WSDL pour la description des services, impose d'importer le schéma XSD (W3C/XSD, 2012) des documents XMI dans le descripteur des SMA manipulant des documents XMI en entrée ou en sortie. La figure suivante correspond au descripteur de service WSDL *ImproveRoleAMS*. IL est à noter que le schéma XSD des documents XMI est importé au niveau des messages d'entrée et de sortie du descripteur.

```

<description
targetNamespace="http://crinfo.univ-paris1.fr/schema/ImproveRoleAMS.wsdl"
name="ImproveRoleAMS"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:tns="http://ImproveRoleAMS/"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/ImproveRoleAMS.xsd"
  xmlns:xmldocument="http://schema.omg.org/spec/XMI/2.1"
  ...>
<types>
<xsd:schema
targetNamespace="http://crinfo.univ-paris1.fr/schema/ImproveRoleAMS.xsd">
<xsd:import namespace="http://schema.omg.org/spec/XMI/2.1"
  schemaLocation="http://www.omg.org/spec/XMI/20071213/XMI.xsd"/>
<xsd:element name="ImproveRoleAMSRequest">
<xsd:complexType>
<xsd:sequence>
<element ref="xmldocument:XMI" minOccurs="1" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="ImproveRoleAMSResponse">
<xsd:complexType>
<xsd:sequence>
<element ref="xmldocument:XMI" minOccurs="1" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</types>
...
<operation name="ImproveRole">
  <input messageLabel="BusinessStudyInputMessage"
    element="xsdio:ImproveRoleAMSRequest" />
  <output messageLabel="BusinessStudyOutputMessage"
    element="xsdio:xsdio:ImproveRoleAMSResponse" />
</operation>
...

```

Figure 5.8 : Exemple de descripteur WSDL pour le SMA *ImproveRoleAMS*

L'import de schéma XSD (en rouge à la figure 5.8) permet de définir les types des messages d'entrée et de sortie du SMA (représentés respectivement en bleu et en vert à la figure 5.8) comme un élément de type XMI qui contient un schéma UML au format XMI. Cette technique permet de représenter des types complexes au niveau des services et ainsi de pouvoir opérationnaliser la transmission de données complexes entre les services.

5.2.1.3. Verrou : base commune de composants

Dans l'architecture AOS, l'annuaire fournit les fonctionnalités nécessaires à la publication de descripteur de service et à leur recherche. Il centralise ainsi de nombreux descripteurs de services produits par une multitude de fournisseurs de services. Dans le domaine de l'IMS, cette centralisation s'applique aux SMA. Leurs descripteurs doivent fournir une technique de représentation des composants de méthode qu'ils soutiennent indépendamment de l'approche d'IMS dont ils sont issus. Cela permet d'obtenir un annuaire de SMA unique les centralisant afin de faciliter leur recherche et leur réutilisation.

Nous proposons comme solution d'effectuer une extension du format WSDL pour inclure dans le descripteur du SMA des annotations sémantiques sur le composant de méthode qu'il implémente. Nous proposons d'utiliser l'ontologie de métamodèles de composant de méthode ainsi que la technique d'annotation proposée par l'approche YASA que nous avons présentée au chapitre 4 pour effectuer ces annotations. La figure 5.9 regroupe l'ensemble des modifications apportées au schéma XSD de définition d'un descripteur WSDL pour inclure le mécanisme d'annotation du standard WSDL, tel que nous l'avons présenté au chapitre 4.

```
<xsd:simpleType name="listOfAnyURI">
<xsd:list itemType="xsd:anyURI"/>
</xsd:simpleType>

<xsd:attribute name="smeConcept" type="listOfAnyURI" />
<xsd:attribute name="modelReference" type="listOfAnyURI" />
```

Figure 5.9 : Définition des annotations sémantiques pour un descripteur WSDL

Les éléments XML constituant un descripteur WSDL sont définis dans le schéma XSD du langage WSDL (pour la version 2.0). Il est alors facile d'étendre un descripteur WSDL en définissant un schéma XSD uniquement pour définir la structure des annotations, puis de l'importer dans le descripteur afin de pouvoir les utiliser. Ce mécanisme a été utilisé dans le langage SAWSDL (W3C/SAWSDL, 2007) afin d'apporter des annotations au langage WSDL, Il a été ensuite repris et étendu dans l'approche YASA afin de rajouter une annotation supplémentaire. Par conséquent, nous proposons de définir un schéma XSD de définition des annotations de descripteur des services méthodologiques inspiré de ceux utilisés dans SAWSDL et YASA. Comme présenté à la figure 5.9, deux attributs sont définis : *smeConcept* et *modelReference* (en vert à la figure 5.9). Ces deux attributs servent de conteneur pour les annotations et sont utilisables sur n'importe quel élément extensible et pertinent d'un descripteur WSDL, c'est-à-dire les éléments : *Interface*, *Operation*, *Input* et *Output*. Ils contiennent pour l'attribut *smeConcept* une liste d'URI faisant référence à des concepts de notre ontologie OMCM et pour l'attribut *modelReference* une liste d'URI pointant vers des concepts d'ontologie de domaine. Cela est réalisable par le biais de la définition d'un nouveau type d'attribut *listOfAnyURI* dans le schéma XSD (en bleu à la figure 5.9) qui est défini comme une liste d'éléments URI.

Par définition, un descripteur WSDL est une représentation logique d'un service dans une architecture AOS. Cette extension apportée au standard WSDL permet de tenir compte de la dimension "méthode" dans la description technique d'un SMA et par conséquent, d'injecter des données sémantiques du composant de méthode relatif au SMA dans son descripteur.

5.2.1.4. Verrou : recherche sémantique

Toujours concernant l'annuaire de services méthodologiques dans l'architecture MOS, les SMA ne sont pas des services simples à l'image de ceux utilisés dans l'architecture AOS. Un SMA est l'implémentation d'un composant de méthode d'une approche d'IMS. Par conséquent, de la même manière qu'un SMA ne peut pas être uniquement décrit de manière logique pour être réutilisé, la recherche des SMA ne peut pas être réalisée uniquement sur des critères logiques.

En effet, les critères des utilisateurs recherchant des SMA sur l'annuaire de services méthodologiques correspondent à leurs besoins en termes méthodologiques vis-à-vis de leur contexte de projet et des méthodes qu'ils utilisent. Les exigences de ces utilisateurs sont des critères sémantiques décrivant les caractéristiques des composants de méthode dont ils ont besoin. Par conséquent, un algorithme de recherche classique basé sur des critères logiques uniquement tels que ceux présents dans un annuaire UDDI issu d'une architecture AOS ne peut répondre à ces besoins de recherche propres au domaine de l'IMS.

Nous proposons donc d'étendre le standard UDDI pour la publication et la recherche de SMA afin de prendre en compte les exigences spécifiques des utilisateurs au domaine de l'IMS. Comme nous l'avons présenté au chapitre 4 de cette thèse, nous proposons d'adapter l'une des approches de recherche de services web sémantiques au contexte du domaine de l'IMS. Pour cela, nous avons choisi d'utiliser notre ontologie de métamodèles de composant de méthode conjointement à l'approche YASA. Cette adaptation permet aux utilisateurs de rechercher les SMA sur un annuaire de services méthodologiques en exprimant dans une requête leurs exigences vis-à-vis des SMA sous la forme d'annotations sémantiques. L'ontologie OMCM est ensuite utilisée par l'algorithme d'appariement de l'approche YASA pour mettre en correspondance la requête d'un utilisateur avec les descripteurs de services méthodologiques présents dans l'annuaire.

5.2.1.5. Verrou : interactivité avec l'utilisateur

Du point de vue du client dans l'architecture AOS, les services sont des éléments métiers qui ne sont pas utilisables directement par les utilisateurs finaux. En effet, l'interface graphique et la gestion de l'interaction avec les utilisateurs finaux sont laissées à la charge du client. Aucun mécanisme standard n'est proposé dans l'architecture AOS initiale pour gérer les interactions homme-machine. Dans le contexte d'une application du paradigme des AOS au domaine de l'IMS, cette vision des services uniquement orientée métier est un frein à leur usage par les utilisateurs finaux et par conséquent, un

frein à la réutilisation des composants de méthode qu'ils implémenteront. Un composant de méthode est par définition un élément autonome d'une méthode et son support outillé se doit de l'être également. Un SMA se doit donc d'être directement prêt à l'usage. Pour produire des SMA ayant cette propriété, l'application des principes de l'architecture AOS ne suffit pas. En effet, les services développés doivent être des services applicatifs utilisables directement par des utilisateurs finaux. C'est pour cette raison que nous proposons également de repenser la plateforme cliente pour l'usage des méthodes sous la forme d'un portail méthodologique, puis d'appliquer le standard WSRP (OASIS/WSRP, 2008) défini par le groupe OASIS (OASIS, 2012) en plus des standards des AOS pour la réalisation des SMA. Ce standard permet la définition de portlets, ceux-ci étant une encapsulation d'une couche de présentation et d'interactions avec les utilisateurs avec un service métier comme le montre la figure 5.10.

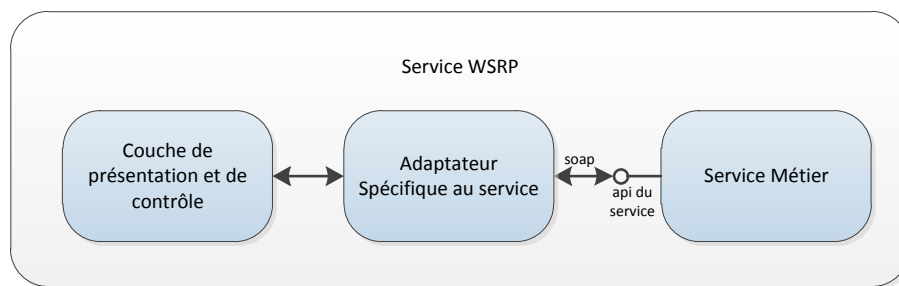


Figure 5.10 : Structure interne d'un portlet

Un portlet appelé également service WSRP est décrit par un descripteur WSDL standard. Il peut donc être publié dans un annuaire de services d'une architecture AOS. Dans le cadre de la réalisation d'un SMA, le descripteur WSDL étendu au domaine de l'IMS tel que nous l'avons présenté précédemment sera utilisé.

La couche de présentation d'un portlet regroupe toute la partie interaction avec l'utilisateur ainsi que les fonctionnalités de saisie et de présentation de données. Si l'implémentation de cette couche peut être réalisée au moyen de n'importe quel langage, elle se doit cependant de générer l'interface homme-machine conformément aux standards interprétables par tous les navigateurs internet, puisque les portlets sont exécutés dans un portail web. Nous utilisons donc le standard XHTML (W3C/XHTML, 2002) et CSS (W3C/CSS, 2011) pour la présentation, puis Javascript (ECMA, 2011), DOM (W3C/DOM, 2004) et XML (W3C/XML, 2008) pour la gestion de l'interactivité de l'interface avec l'utilisateur.

La partie "contrôle" d'un portlet doit, quant à elle, effectuer le traitement des événements résultant des actions d'un utilisateur sur l'interface. Elle a pour fonction d'invoquer le traitement applicatif en fonction de l'événement reçu. De plus, elle a pour rôle de formater les données provenant de la partie

présentation du portlet pour quelles soient en concordance avec les types de données attendus par le traitement applicatif.

Le traitement applicatif est réalisé par un service métier accessible à travers un adaptateur spécifique au service. Celui-ci prend en charge la communication avec le service métier en fonction de son implémentation.

Concernant la couche métier du portlet, elle est implémentée par un service web classique conformément aux standards de l'architecture AOS. Elle peut également être plus complexe et être implémentée par une composition de services conformément au standard BPEL (OASIS/BPEL, 2007) d'orchestration de services.

Les portlets choisis par les utilisateurs sont invoqués dans un portail comme le montre la figure suivante.

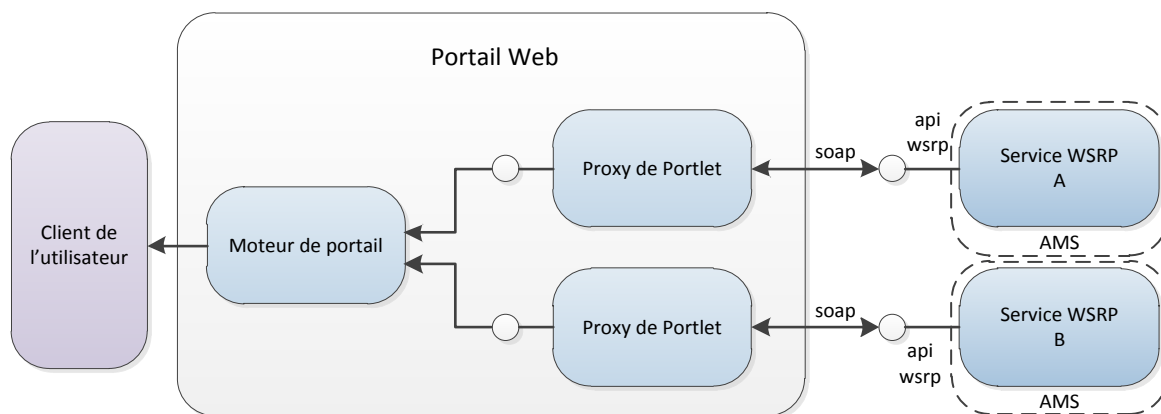


Figure 5.11 : Structure d'un portail pour l'agrégation des portlets

Comme présenté à la figure 5.11, le moteur de portail permet d'invoquer un portlet à l'aide d'un proxy de portlet qui est un conteneur générique de portlet WSRP. Ce moteur permet ainsi d'agréger des portlets dans le portail et de générer la présentation globale à afficher à l'utilisateur. L'implémentation du moteur de portail est spécifique à chaque type d'environnement d'exécution de portail (LifeRay (Liferay, 2012), OpenPortal (Open Portal, 2012)...). Le proxy de portlet, est quant à lui, un module générique permettant d'invoquer un service WSRP et de transmettre la réponse de ce service au moteur de portail. Cette généricité est possible car les services WSRP sont des services interactifs, ils encapsulent les couches de présentation, de contrôle et de métier dans le même composant logiciel. Les services WSRP possèdent donc tous la même interface d'invocation.

Les portlets peuvent communiquer entre eux par un mécanisme événementiel décrit dans la JSR 286 (ORACLE, 2008). Une application peut ainsi être constituée par une agrégation de portlets dans un portail (voir figure 5.11).

Notre solution se base sur ces principes de fonctionnement d'un portail et nous proposons de constituer une plateforme orientée méthode et dédiée à l'invocation de SMA dans un portail. L'implémentation de la plateforme est ainsi constituée d'une agrégation de portlets fournissant aux utilisateurs les fonctionnalités de recherche puis d'invocation et d'exécution des services méthodologiques. Chaque SMA étant lui-même implémenté comme un service interactif par un portlet WSRP. Le mécanisme d'invocation événementiel entre portlets est utilisé pour transmettre les paramètres d'entrée et de sortie entre un SMA implémenté comme un portlet et la plateforme MaaS elle-même implémentée comme un portlet. Nous présentons en détail cette plateforme dans le chapitre 7.

5.2.1.6. Verrou : configuration de méthodes

Toujours du point de vue du client, les besoins des utilisateurs de méthode vont au delà de l'exécution de services méthodologiques atomiques. Afin de mener à bien leurs projets, notre solution doit leur fournir les fonctionnalités nécessaires à l'exécution d'une méthode entière. Contrairement à la position prise par la plupart des approches d'IMS, l'enquête auprès avec des industriels, que nous avons présentée au chapitre 3, nous a montré que les utilisateurs préféraient configurer des méthodes pour satisfaire les exigences spécifiques des projets plutôt que de composer une nouvelle méthode par un assemblage de composants à chaque projet. Etant donné le caractère "agile" de nombreux projets actuels d'ingénierie de SI, cette configuration méthodologique doit également pouvoir survenir à n'importe quelle étape d'un projet.

Nous proposons d'intégrer la notion de variabilité telle que définie dans (Bühne, 2004) dans les méthodes pour former des lignes de méthode configurables. D'un point de vue technique, l'implémentation consiste en premier lieu en la définition d'un langage pour la représentation technique des lignes de méthode. Ensuite, il y a la réalisation d'un moteur pour l'exécution des lignes de méthode, c'est-à-dire l'orchestration de l'exécution des SMA constituant la ligne. Etant donné que les SMA sont des services WSRP et non des services des AOS classiques leur orchestration doit être réalisée dans un portail. Pour cela, nous proposons de nous inspirer du fonctionnement du standard BPEL (OASIS/BPEL, 2007) pour la définition d'un langage basé sur XML, adapté aux spécificités de la définition de lignes de méthode, ainsi que la définition d'un moteur d'exécution de ligne de méthode gérant l'orchestration des SMA dans un portail. Nous détaillons de manière plus ample cette solution dans la section 5.3 de ce chapitre.

Après avoir expliquées les solutions techniques qui sont envisagées pour lever les six verrous technologiques, nous allons expliquer, dans la section suivante, le métamodèle de service atomique.

5.2.2 Le métamodèle de service méthodologique atomique

Le métamodèle de service méthodologique atomique présente la structure générique d'un outil apportant l'aspect outil qui fait actuellement défaut aux méthodes issues des approches d'IMS. Chaque composant de méthode ainsi créé par une approche d'IMS peut être outillé par un service méthodologique.

Par conséquent, le métamodèle de SMA tel que présenté à la figure 5.12 peut être relié aux approches d'IMS en tant que modèle de support outillé des composants de méthode. Il est donc défini comme une spécialisation de l'ontologie OMCM présenté au chapitre 4 de cette thèse.

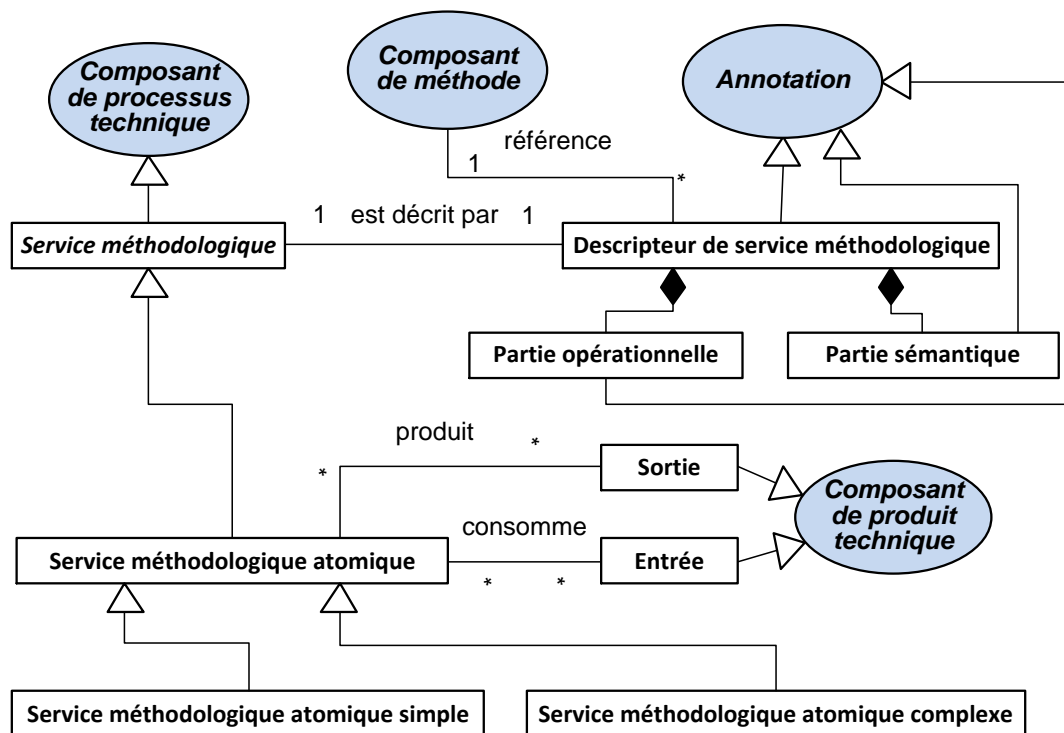


Figure 5.12 : Métamodèle de service méthodologique atomique

Comme présenté à la figure 5.12, un service méthodologique est un concept abstrait héritant du concept de *Composant de processus technique* (en bleu à la figure 5.12) de l'ontologie OMCM. Chacun d'entre eux est décrit par un descripteur de service méthodologique.

Ce descripteur synthétise à la fois la partie opérationnelle du service méthodologique et la partie sémantique du composant de méthode associé au service. Le descripteur de service méthodologique ainsi que les parties opérationnelle et sémantique qui le composent peuvent être définies comme des spécialisations du concept d'*Annotation* (en bleu à la figure 5.12) de l'ontologie OMCM.

En effet, la partie opérationnelle du descripteur décrit la partie *Composant de processus technique* et *Composant de produit technique* d'un composant de méthode. En d'autres termes, elle décrit

l'ensemble des données nécessaires à l'invocation et l'exécution de l'outil associé au composant de méthode : le service méthodologique.

La partie sémantique du descripteur de SMA, quant à elle, résume la vision conceptuelle d'un composant de méthode à l'aide d'annotation sémantique incluse dans le descripteur. Celles-ci sont exploitées lors de la publication et la recherche des services méthodologiques. De plus, l'association d'un SMA à un composant de méthode conceptuel est réalisée dans le descripteur de service méthodologique par une référence à son *Composant de méthode*.

Un service méthodologique peut être spécialisé de manière plus concrète par un service méthodologique atomique. Ce dernier est défini comme un outil de support : la partie technique d'un composant de méthode. En d'autres termes, c'est l'implémentation du processus d'un composant de méthode. Comme indiqué par le mot "atomique" présent dans sa désignation, un SMA est le support d'un composant de méthode autonome de granularité faible. L'implémentation d'un SMA peut être simple ou complexe. Un service méthodologique atomique simple correspond à un type d'implémentation sous la forme d'un service atomique et autonome. Tandis qu'un service méthodologique atomique complexe correspond à une implémentation d'un service méthodologique sous la forme d'une composition opérationnelle de services.

Un service méthodologique atomique simple nécessite le développement d'un service interactif utilisant un seul service métier alors que plusieurs services métiers sont utilisés par un complexe.

Chaque SMA consomme des produits en entrée du processus du composant de méthode qu'il implémente et délivre des produits en sortie. Ces "entrées" et "sorties" sont une représentation opérationnelle des données complexes manipulées par le composant de méthode. Du point de vue de l'ontologie OMCM, ce sont des *Composants de produit technique* (en bleu à la figure 5.12). Comme nous l'avons énoncé à la section 5.2.1.2 de ce chapitre les données manipulées par les SMA ont plusieurs niveaux de représentation qui doivent être pris en compte par le SMA.

Un SMA simple manipulant des données complexes peut, par exemple, être un SMA d'objectification d'association entre deux classes avec une cardinalité 1-n à chaque terminaison. La figure ci-dessous présente l'exemple du composant de méthode conceptuel associée au SMA *ObjectifyAMS*.

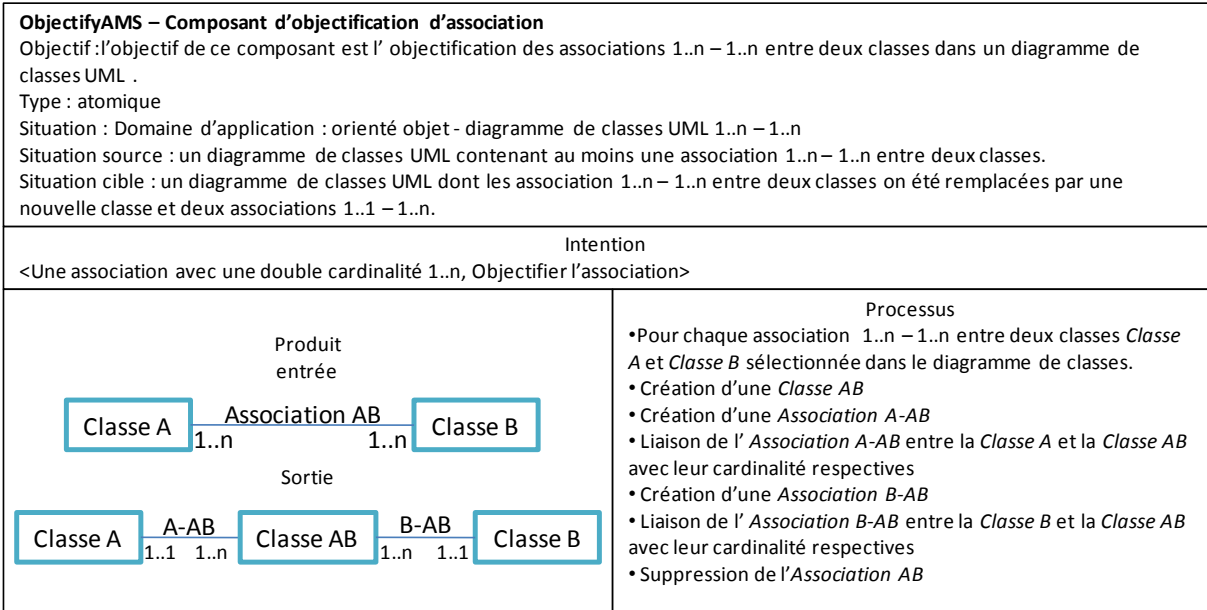


Figure 5.13 : Exemple de composant de méthode d’objectification d’association

Comme présenté à la figure 5.13, le composant de méthode d’objectification propose de rechercher toutes les associations 1..n - 1..n entre deux classes et pour celles sélectionnées par l'utilisateur, de les transformer en une classe représentant l'association avec deux associations de type 1..1 - 1..n reliant cette nouvelle classe aux deux classes intervenant dans l'ancienne association. Dans cet exemple, une *Classe A* et une *Classe B* sont reliées par une *Association AB* avec deux cardinalités 1..n. L'Association est donc transformée en *Classe AB*. Cette nouvelle classe est reliée à la *Classe A* par une nouvelle association *A-AB* avec des cardinalités 1..1 et 1..n. Il en va de même pour la *Classe B* qui est reliée par une nouvelle association *B-AB* avec des cardinalités 1..1 et 1..n à cette nouvelle *Classe AB*. Les figures suivantes présentent le schéma de fonctionnement du SMA simple implémentant ce composant conceptuel et la maquette du portlet implémentant ce SMA.

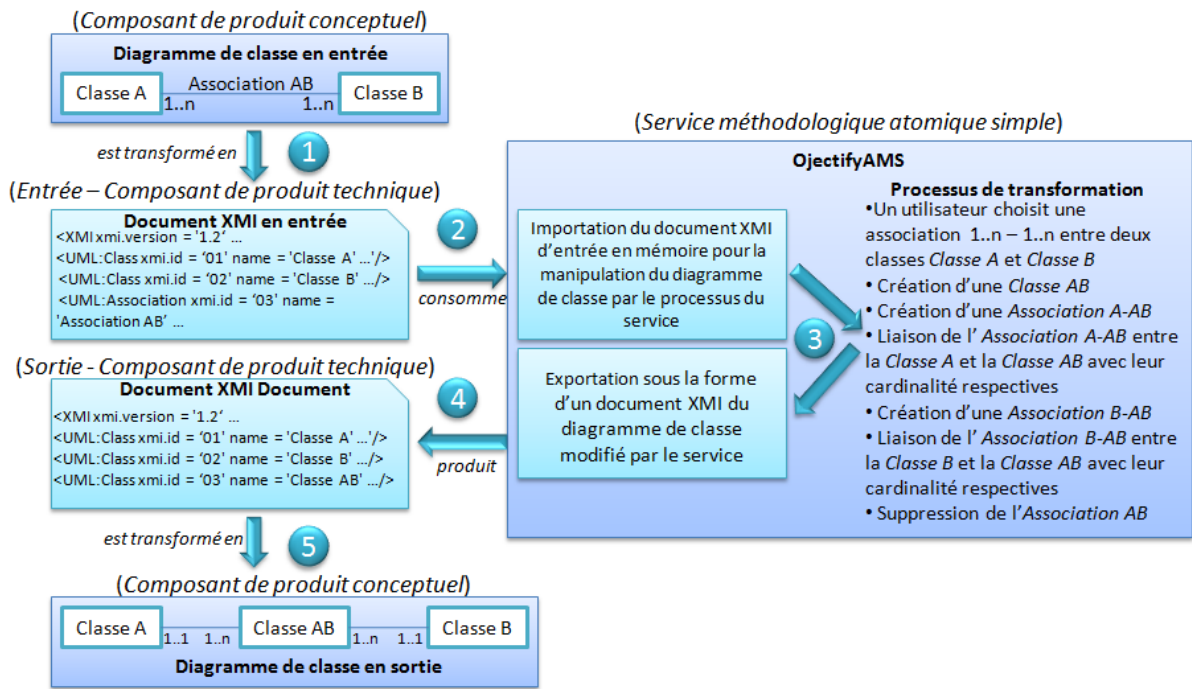


Figure 5.14 : Principe de fonctionnement du SMA utilisant le composant d'objectification

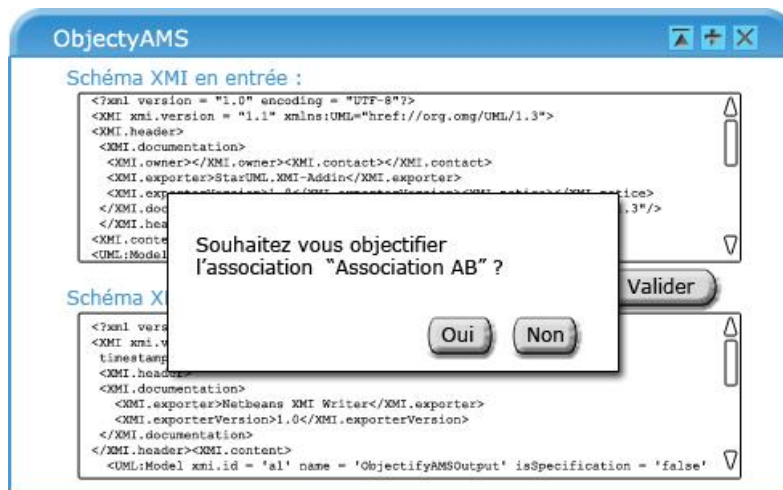


Figure 5.15 : Maquette du portlet implémentant le SMA ObjectifyAMS

Le composant conceptuel d'objectification d'association s'appliquant sur des diagrammes de classes UML, le SMA le prenant en compte doit, par conséquent, traiter avec ces types de données complexes en entrée et en sortie. Pour cela, le diagramme en entrée sur lequel va travailler le SMA est exporté au format XMI (point 1 de la figure 5.14) avant d'être passé en paramètre après l'invocation du SMA par un utilisateur (point 2 de la figure 5.14 et champ supérieur de la figure 5.15). Le SMA charge ensuite en mémoire le document XMI donné en entrée, il peut par exemple convertir le document XMI en une série d'objets plus facile à manipuler. Le SMA exécute ensuite l'algorithme implémentant le processus du composant conceptuel (point 3 de la figure 5.14 et figure 5.15) permettant de détecter les associations pouvant être objectifiées et de demander à l'utilisateur de sélectionner celles qu'il souhaite

transformer. Il procède ensuite aux transformations sur le diagramme de classes, puis il exporte le diagramme de classes modifié qu'il a en mémoire sous la forme d'un document XMI qu'il affiche en réponse à l'utilisateur (point 4 de la figure 5.14 et champ inférieur de la figure 5.15). Ce document XMI résultat du SMA peut à nouveau être chargé par l'utilisateur dans un outil de modélisation afin de visualiser le nouveau diagramme de classe ainsi amélioré. Le descripteur complet du SMA de cet exemple est disponible à l'annexe C.

Le métamodèle de SMA présenté dans cette section propose une solution à l'intégration de l'aspect outil à un composant de méthode. En effet, dans cette solution un service méthodologique est une spécialisation du concept *Composant de processus technique* de l'ontologie OMCM que nous avons présenté au chapitre 4. Cette partie "technique" d'un composant de méthode est liée à un composant dit conceptuel au travers de son descripteur sémantique. Ce métamodèle de SMA est applicable à toutes les approches d'IMS. Il permet de réaliser une implémentation interopérable de l'aspect outil de leurs composants de méthode.

L'annuaire étendu de services méthodologiques expliqué au chapitre 4 et combiné avec le métamodèle de SMA permet de rechercher des *Composant de processus technique* en fonction des caractéristiques sémantiques extraites du *Composant de processus conceptuel* et du *Composant de produit conceptuel* associés.

Nous avons expérimenté la faisabilité du service méthodologique à l'aide d'une solution de type portail web conforme à la norme JSR286 (ORACLE, 2008). Celle-ci préconise la technique des portlets pour les services interactifs et donc par conséquent pour les SMA. L'architecture du portlet inclut une partie présentation/contrôle et une partie service métier conforme au service web de type SOAP. Ce dernier représente l'implémentation du processus conceptuel du composant de méthode.

Enfin, la technologie WSRP intégrée au portail conforme à la norme JSR286 permet d'invoquer à distance un portlet (cette solution technique est détaillée au chapitre 7).

5.3. Les lignes de méthode

Après avoir défini le concept de service méthodologique atomique, nous allons nous intéresser au service méthodologique de granularité plus importante que l'on appelle des lignes de méthode. L'objectif de ce concept est de représenter au premier niveau la variabilité qui existe dans la démarche d'une méthode.

La section 5.3.1 effectue un rapide panorama des différentes significations du concept de variabilité et positionne notre travail dans celui-ci.

Le métamodèle de ligne de méthode est décrit à la section 5.3.2 alors que la démarche de construction d'une ligne de méthode est détaillée à la section 5.3.3 avec un exemple sur l'itération de

développement selon les méthodes agiles (Extreme Programming (Wells, 2009) et Scrum (Schwaber, 2011)).

La spécification des guides de configuration est expliquée et illustrée sur l'exemple à la section 5.3.4.

Enfin, la spécification des aspects produits d'une ligne de méthodes fait l'objet de la section 5.3.5.

5.3.1 La variabilité des processus

D'une manière générale, la variabilité est définie par (VanGurp, 2000) comme la capacité d'un système à être changé, personnalisé et configuré selon un contexte spécifique.

Dans la littérature, ce concept de variabilité a été appliqué à une multitude de domaines et de contextes. D'après (Bennasri, 2005), il en ressort deux classifications pertinentes des types de variabilités dans le domaine de l'ingénierie des SI, celle de (Bachmann, 2001) (voir figure 5.16 ci-dessous) orientée sur l'ensemble des éléments d'un SI et celle de (Halmans, 2003) (voir figure 5.17) faisant une distinction entre une variabilité dite "essentielle" et une variabilité dite "technique".

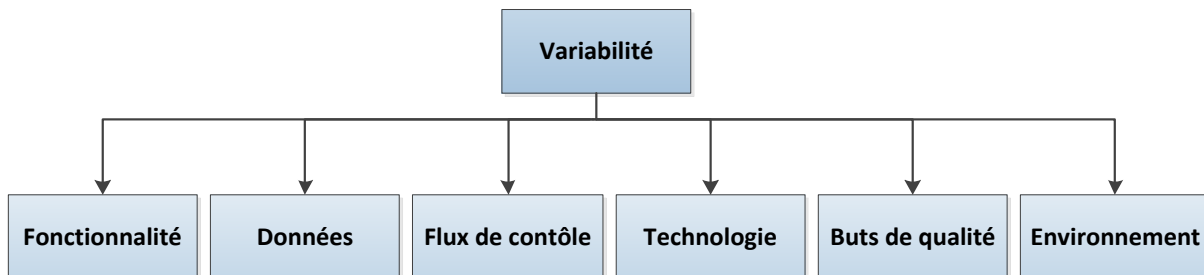


Figure 5.16 : Les catégories de variabilité selon (Bachmann, 2001)

Comme le montre la figure 5.16, Bachmann et ses collègues proposent une distinction de la variabilité en six catégories :

- **Variabilité dans les fonctions :** l'application de nombreuses fonctionnalités peuvent conduire à l'obtention d'un résultat équivalent tout en employant des manières différentes dans la réalisation du résultat. Par exemple le parcours d'un arbre peut être réalisé selon deux variantes : en partant de la branche la plus à droite du nœud racine ou en partant de la branche la plus à gauche.
- **Variabilité dans les données :** une structure de données peut varier d'un système à l'autre. Par exemple, la structure employée pour stocker les données relatives aux clients peut être différente dans une application dédiée au service marketing de celle dédiée au service de livraison.
- **Variabilité dans le flux de contrôle :** les modes d'interaction entre les différents composants d'un système varient.

- **Variabilité dans la technologie** : ces variations concernent les changements de plateforme tels que le matériel, le système d'exploitation, le langage de programmation ou encore l'interface homme machine.
- **Variabilité dans les buts de qualité** : les exigences vis-à-vis des besoins non fonctionnels peuvent varier d'un système à l'autre. Par exemple, pour la même application entre la version distante et la version mobile les critères de sécurité peuvent varier.
- **Variabilité dans l'environnement du système** : un domaine spécifique peut imposer à un système des exigences qui lui sont propres. Par exemple, le lancement d'une fonctionnalité dans un premier système peut être manuel et être automatisé dans un deuxième système.

Une autre classification de la variabilité a été proposée dans (Halmans, 2003). Cette classification met l'accent sur l'opposition entre le point de vue des utilisateurs (utilisation d'un système), représenté par le concept de variabilité essentielle, et le point de vue des développeurs du système, associé au concept de variabilité technique (voir la figure ci-dessous).

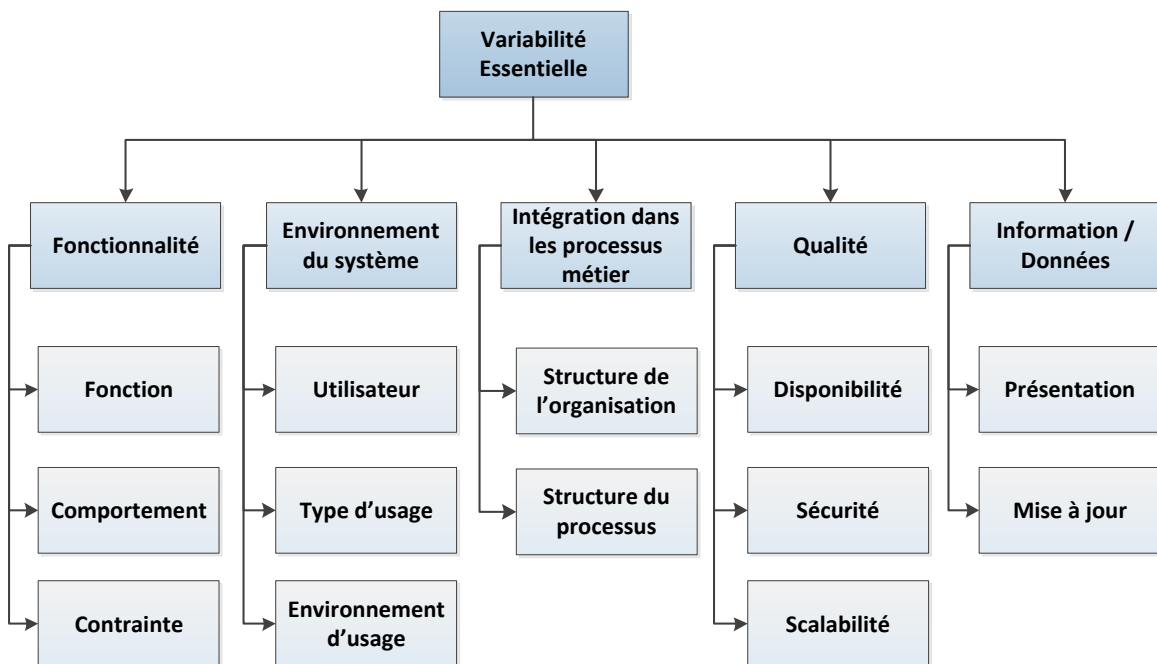


Figure 5.17 : Les catégories de variabilité essentielle selon (Halmans, 2003).

La variabilité essentielle regroupe en cinq catégories (voir figure 5.17) les aspects propres à l'utilisation d'un système :

- **Variabilité dans les fonctionnalités** : deux applications d'une même famille peuvent proposer aux utilisateurs des fonctionnalités différentes. Inversement, plusieurs fonctionnalités

peuvent obtenir le même résultat en utilisant des manières différentes pour le produire. Les contraintes associées aux fonctionnalités peuvent également varier.

- **Variabilité dans l'environnement du système** : ce type de variabilité regroupe l'ensemble des changements possibles dans l'environnement d'usage d'une application indépendamment des aspects techniques. Elle comprend les variations concernant les utilisateurs et les variations d'usage des utilisateurs sur le système. Par exemple, le nombre d'utilisateurs d'une application peut varier ou encore certaines fonctionnalités du système peuvent être invoquées manuellement ou automatiquement. L'environnement d'usage peut également varier. Par exemple, une application peut être utilisée de manière autonome ou être intégrée à une autre application.
- **Variabilité dans l'intégration dans les processus métier** : les variations concernant les processus métiers s'organisent en deux points de vue. Le premier regroupe les variations dans la structure de l'organisation, par exemple, une application adaptable aux différences de structure hiérarchique des organisations. Les rôles et les responsabilités des acteurs de l'organisation ainsi que les tâches qui leur sont affectées peuvent également varier. Le deuxième point de vue est au niveau des processus métier de l'organisation. Deux organisations ayant la même structure et remplissant des buts similaires peuvent avoir une structure et un ordonnancement de leurs processus métiers totalement différents.
- **Variabilité dans la qualité** : cette notion de variabilité rassemble toutes les variations possibles des besoins non fonctionnels d'une application comme la sécurité, la disponibilité de l'application ou encore sa robustesse.
- **Variabilité dans l'information et les données** : les mêmes données peuvent être présentées de manières différentes en fonction du système où elles sont utilisées ou des utilisateurs auxquelles elles sont présentées. La structure de ces données peut également varier d'une application à l'autre. De plus, le taux d'actualisation des données peut changer d'une application à l'autre. Par exemple, une application de gestion de température peut, en fonction de son utilisation, actualiser la mesure de la température toutes les trente minutes pour une maison et en temps réel lorsque la chaîne du froid ne doit pas être rompue dans le domaine alimentaire.

La figure suivante présente les trois catégories de variabilité technique.

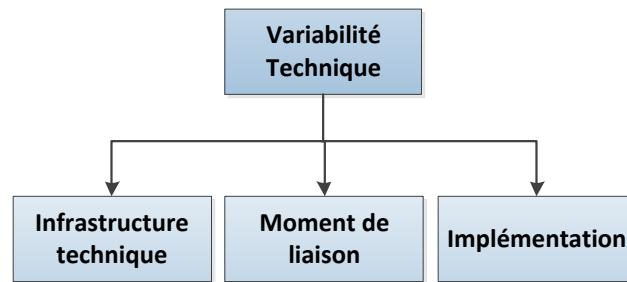


Figure 5.18 : Les catégories de variabilité technique selon (Halmans, 2003).

La variabilité technique regroupe en trois catégories l'ensemble des variations dans le développement d'un système (voir figure 5.18) :

- **Variabilité de l'infrastructure technique :** l'infrastructure technique dans laquelle une application s'exécute est susceptible de changer d'une organisation à l'autre. Cette catégorie de variabilité correspond à la flexibilité d'une famille de produits applicatifs à des environnements techniques différents. Par l'exemple, l'adaptation d'une application à des configurations de réseau et de débits différents, ou encore, sa robustesse vis-à-vis de multiples composants physiques.
- **Variabilité dans le moment de liaison :** le moment de liaison correspond à l'instant de changement dans la mise en place de la variation dans le processus d'ingénierie. Cela correspond au moment où une adaptation à un changement peut être effectuée dans un processus.
- **Variabilité dans l'implémentation :** il existe en pratique une multitude de techniques pour réaliser la variabilité dans un système. Les variations peuvent être par exemple réalisées au moyen d'un paramétrage prédéfini, par des instanciations de patrons ou encore des spécialisations de modèles d'implémentation.

La variabilité pertinente à représenter dans les méthodes s'apparente, du point de vue de l'utilisateur, à une variabilité essentielle dans les fonctionnalités et, du point de vue du système, à une variabilité technique dans le moment de liaison et dans l'implémentation selon (Halmans, 2003).

Après avoir ciblé le type de variabilité utile pour les méthodes, nous allons nous intéresser à la modélisation de cette variabilité. Dans la littérature, plusieurs définitions de la variabilité existent. Elles convergent toutes vers une expression de la variabilité sous la forme de points de variation et de variantes.

D'une manière générale, (Czarnecki, 2000) définit les points de variation et les variantes comme suit : *Un point de variation désigne un endroit dans le système où il existe une variation. C'est à cet endroit*

que des choix doivent être faits afin d'identifier les variantes à utiliser. Chaque variante est une manière de réalisation de la variabilité.

La figure suivante présente le métamodèle de cette expression de la variabilité.

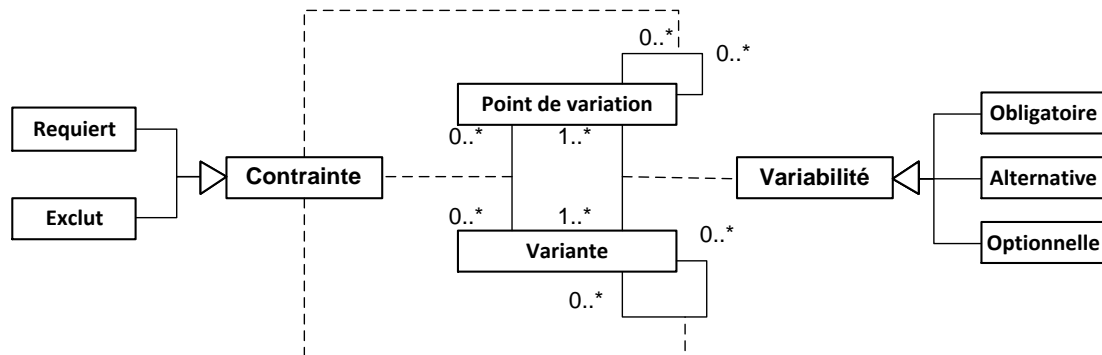


Figure 5.19 : Métamodèle de la variabilité (Bühne, 2004)

Un métamodèle de la variabilité a été proposé en 2004 par (Bühne, 2004) (voir figure 5.19). Ce métamodèle s'inscrit dans la continuité des travaux dans le domaine de la variabilité. Il permet la modélisation de la variabilité pour la définition d'une ligne d'éléments, par exemple des lignes de produits ou de processus. Il définit la variabilité comme une relation entre des points de variation et des variantes. Un point de variation représente la localisation et la raison des variations. Les variantes correspondent, quant à elles, aux différentes possibilités de variations. La relation entre les points de variation et les variantes est appelée variabilité.

Cette relation peut être de différents types :

- Un lien de variabilité de type *obligatoire* entre une variante et un point de variation implique que cette variante doit obligatoirement faire partie intégrante des variantes sélectionnées pour ce point de variation. Par exemple, un point de variation P1 avec deux variantes obligatoires V1 et V2 implique que ces deux variantes soient toujours sélectionnées dans la réalisation des fonctionnalités attachées à ce point.
- Une variabilité qualifiée d'*alternative* implique qu'une des variantes doit être sélectionnée parmi celles reliées par ce type de lien au point de variation. Par exemple, un point de variation P2 avec une variante obligatoire V3 et deux variantes alternatives V4 et V5 implique de sélectionner de manière exclusive V4 ou V5 en plus de la variante V3 qui est obligatoire.
- Une variabilité de type *optionnelle* implique que la variante reliée à un point de variation par ce type de variabilité est optionnelle, c'est-à-dire qu'elle peut être sélectionnée ou écartée de ce point de variation. Un point de variation P3 avec deux variantes optionnelles V6 et V7

implique par exemple comme possibilités de sélection : soit V6, soit V7, soit V6 et V7, soit aucune des deux.

Un autre type de relation existe dans ce métamodèle (voir figure 5.19) : les contraintes. Cette relation peut s'exprimer entre des points de variation et des variantes, entre les points de variation eux même ou entre les variantes elles mêmes. La sémantique de ces contraintes est de deux types : *requiert* et *exclut*. Une contrainte de type *requiert* implique qu'une variante nécessite qu'une autre variante soit également sélectionnée, ou qu'une variante nécessite qu'une décision ai été prise à un autre point de variation, ou qu'un point de variation requiert un autre point de variation pour pouvoir être réalisé. La contrainte *exclut* implique qu'une variante ne peut être sélectionnée conjointement à une autre variante, point de variation ou qu'aucune variante de deux points de variations ne peuvent être sélectionnées conjointement.

5.3.2 Le métamodèle de ligne de méthode

Par analogie à l'application de la variabilité dans le domaine de l'ingénierie des SI pour former des lignes de produits logiciels, nous proposons d'appliquer les principes de la variabilité au domaine de l'IIMS pour la formation de lignes de méthode.

Les concepts de ligne de méthode et d'une manière plus générale de famille de méthodes ont été introduits par (Kornysheva, 2011). Une famille de méthode est définie comme un ensemble de composants de méthode sémantiquement proches. Une famille peut être vue comme un regroupement de lignes de méthode traitant d'un domaine particulier. Une ligne de méthode est définie par (Kornysheva, 2011) comme une sélection de composants de méthode d'une même famille pour former une méthode configurée pour une situation de projet spécifique.

Le concept de famille de méthode diffère donc de celui de méthode dans le sens où une famille est un regroupement de composants de méthode autour d'un thème d'un niveau d'abstraction supérieur à la méthode. Contrairement à une méthode, la connaissance méthodologie contenue dans une famille a une étendue trop vaste pour être directement utilisable. Il est nécessaire d'effectuer une étape de sélection et/ou de configuration de la famille pour obtenir une connaissance méthodologique applicable dans le contexte d'une entreprise : une ligne de méthode. Le concept de ligne de méthode introduit dans (Kornysheva, 2011) est défini uniquement au niveau conceptuel (c'est-à-dire que ce modèle n'est pas interprétable par un outil d'exécution).

Cette stratégie descendante à partir d'une famille permet d'obtenir une ligne de méthode, il existe également une stratégie ascendante pour la production d'une ligne de méthode comme le présente la figure ci-dessous :

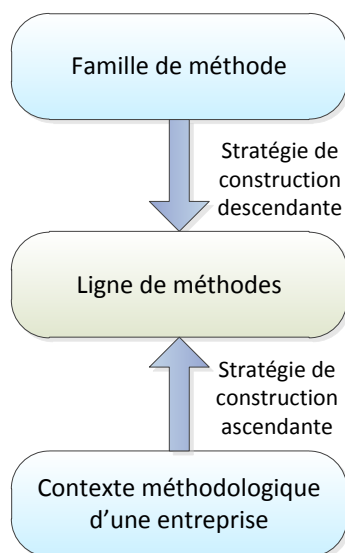


Figure 5.20 : Construction descendante ou ascendante d'une ligne de méthode

Nous proposons, dans cette thèse, d'utiliser les lignes de méthode comme étant une sous-sélection de composants de méthode d'une famille de méthodes d'un domaine particulier adaptée au contexte d'une entreprise. Par conséquent, nous nous focalisons sur la construction d'une ligne de méthode d'une manière ascendante à partir du contexte méthodologique d'une entreprise (voir figure 5.20), c'est à dire des méthodes déjà utilisées par celle-ci et du contexte de ses projets. Une ligne de méthode produite par cette démarche est une mise à plat des processus méthodologiques de l'entreprise en capturant leur variabilité. De plus, nous proposons de nous intéresser à l'aspect outil des lignes de méthode et d'établir un modèle opérationnel de ligne de méthode basé sur l'ontologie OMCM que nous avons présentée au chapitre 4 et le métamodèle de service méthodologique atomique que nous avons présenté à la section 5.2.2 de ce chapitre.

La stratégie descendante à partir d'une famille de méthode peut être utilisée dans notre approche pour sélectionner des composants de méthode pertinents pour l'entreprise et les réorganiser dans un processus à plat exécutable et à forte variabilité.

Au niveau de l'opérationnalisation des lignes de méthode, l'architecture AOS est centrée sur la réutilisation des composants logiciels métiers d'un SI et ne prend pas en compte la réutilisation des interfaces et des interactions avec les utilisateurs du SI. En conséquence, le langage d'orchestration BPEL (OASIS/BPEL, 2007) des services pour l'exécution des processus techniques ne tient pas compte des interactions avec les utilisateurs. Il permet d'orchestrer uniquement des services de bas niveau dit de niveau métier. Face à cette lacune, des extensions du langage BPEL comme le langage BPEL4People (SAP, 2005) ont vu le jour. Cette extension permet d'intégrer des activités exécutées exclusivement par des humains dans un processus BPEL. Par exemple, dans un processus bancaire d'attribution de prêt, la décision d'attribution du prêt ne sera pas prise automatiquement pour les

dossiers client présentant une solvabilité proche de la limite fixée dans le système. Ces dossiers sont redirigés vers des agents bancaires pour être traités manuellement. Concernant notre approche nous proposons d'établir une orchestration à mi-chemin entre celle du langage BPEL et celle de BPEL4People. Autrement dit, proposer un langage permettant une orchestration de services interactifs, directement utilisables par des utilisateurs finaux. En effet, l'exécution d'une méthode nécessite de nombreuses interventions humaines dans le processus, que ce soit pour la réalisation d'activités ou pour la prise de décision. Nous proposons donc d'effectuer une orchestration de services interactifs comportant une partie d'expertise méthodologique ainsi qu'une partie gérant les interactions et la couche présentation pour les utilisateurs, le tout formant un support outillé pour un composant de méthode, puis une orchestration de services interactifs formant un outil configurable pour l'opérationnalisation d'une ligne de méthode.

Un langage de représentation de la ligne de méthode s'apparente au niveau opérationnel à l'orchestration de services méthodologiques. Néanmoins, il faut maintenant introduire dans ce langage le concept de variabilité.

La définition de la variabilité telle que nous l'avons présentée à la section 5.3.1 de ce chapitre repose sur les concepts de points de variation et de variantes. Etant donné qu'une variante représente une possibilité de variation d'un système, une variante est alors représentée par un service méthodologique atomique dans notre approche. En effet, un SMA est le support outillé d'une partie de processus d'une méthode, il correspond à l'aspect outil d'un composant de méthode. Dans le cadre de la construction d'une ligne de méthode, c'est-à-dire une méthode configurable et outillée, le service méthodologique atomique représente une possibilité de variation dans le processus de la méthode ainsi que dans l'outil qui lui est associé.

Appliqué aux méthodes situationnelles, le concept de point de variation représente un endroit dans le processus méthodologique avec plusieurs variations. D'après le métamodèle de la variabilité de (Bühne, 2004) que nous avons présenté à la section précédente, la relation entre les points de variation et les variantes peut être caractérisée en fonction de trois notions : l'obligation, l'alternative ou l'option.

Les points de variations dans le processus d'une méthode doivent regrouper les prises de décision des utilisateurs sur le choix des variantes. Nous proposons donc de séparer les points de variation en deux catégories :

- La première catégorie, que nous nommons "point de variation", regroupe les points de choix où une sélection parmi plusieurs variantes doit être effectuée. Cette catégorie représente en un seul concept les liens de variabilité alternatif et optionnel présentés à la figure 5.19. En effet,

la distinction entre ces deux variabilités est réalisée par le couple de cardinalités min et max défini sur le point de variation.

- La deuxième catégorie regroupe les flux qui ne présentent pas de prises de décision par les utilisateurs, autrement dit des flux obligatoires de variantes ou de points de variation à exécuter dans le processus. Cette catégorie correspond dans la décomposition du processus d'une méthode à des agrégations de services méthodologiques atomiques. En d'autres termes, un enchaînement logique de services ou de points de variation dans lequel tous les éléments doivent être exécutés. De la même manière que pour la définition du langage BPEL (OASIS/BPEL, 2007) qui est un langage d'orchestration de services au niveau technique dans l'architecture AOS, nous proposons de décomposer le concept d'agrégation entre les éléments de ligne de méthodes en trois sous-catégories. En effet, une agrégation peut être une séquence d'éléments, des éléments pouvant être exécutés en parallèle ou de manière itérative. Par exemple, une séquence de SMA représente le fait qu'ils doivent tous être exécutés les uns après les autres, tandis qu'une agrégation de services en parallèle signifie que tous les services doivent être exécutés mais qu'ils peuvent l'être simultanément. Une itération est, par exemple, une boucle d'exécution de services ou de points de variation qui est répétée jusqu'à ce qu'une condition soit satisfaite.

Nous souhaitons mettre en avant les choix dans une ligne de méthode et laisser les utilisateurs prendre les décisions finales. Une ligne de méthode retranscrit une partie des contraintes existantes entre les éléments la constituant dans sa structure elle-même par l'organisation des différents types de point de variation et de variante. Concernant les contraintes pouvant restreindre le processus de décision pour l'exécution des variantes, nous proposons de les intégrer sous la forme de guide de configuration attaché aux points de variation. Par exemple, dans une ligne de méthode regroupant des méthodes agiles avec des méthodes ayant un cycle de vie en cascade comme RUP (Kruchten, 1998), il est déconseillé d'utiliser un composant de méthode basé sur le cycle de vie agile avec un composant de définition des besoins issu de RUP.

La figure ci-dessous présente le métamodèle conceptuel d'une ligne de méthode.

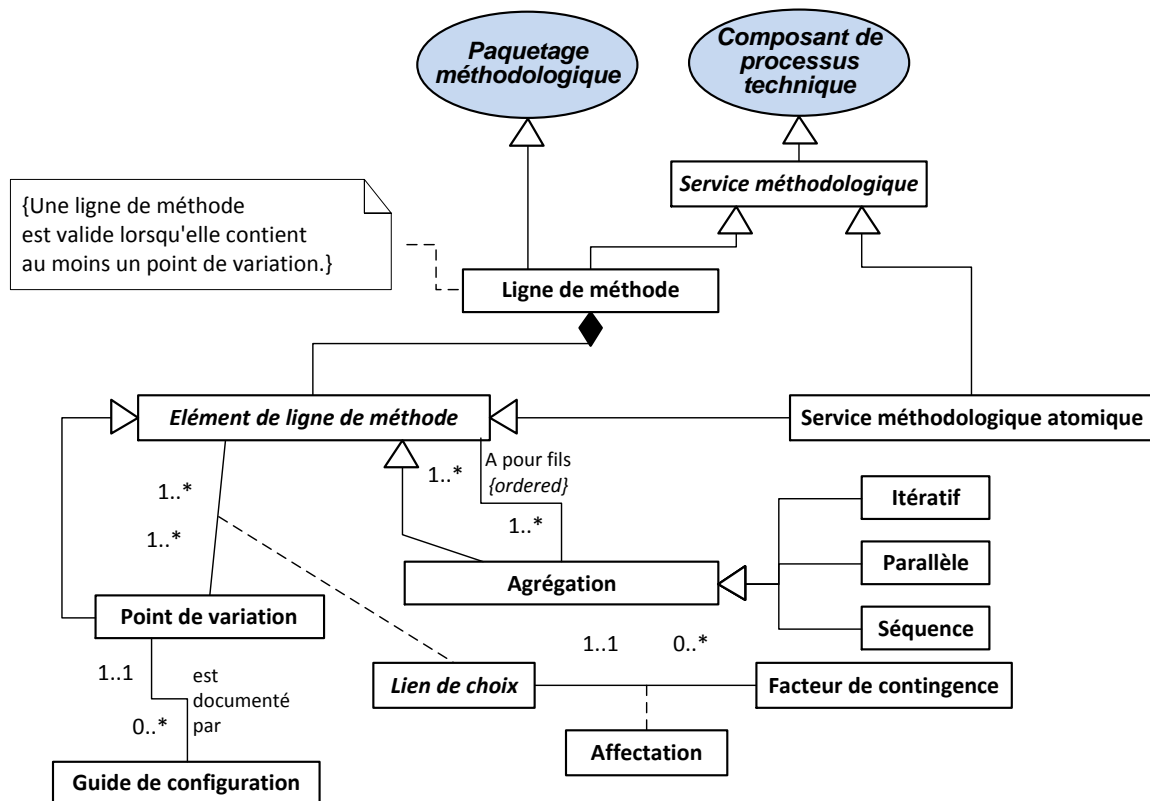


Figure 5.21 : Métamodèle de ligne de méthode : partie processus

Le métamodèle de ligne de méthode que nous proposons afin de prendre en compte la variabilité dans les méthodes situationnelles est présenté à la figure 5.21 en utilisant le langage de modélisation UML.

Ce métamodèle est une spécialisation de l'ontologie OMCM que nous avons proposé au chapitre 4 de cette thèse. En effet, le concept de *ligne de méthode* peut être défini comme une spécialisation du concept de *Paquetage méthodologique*, elle permet à une entreprise de capitaliser l'ensemble des adaptations de ses méthodes en une méthode configurable par une organisation temporelle et configurable de composants de méthode articulés autour d'un même thème d'application. Une *ligne de méthode* est une sélection de composants d'une famille de méthodes adaptés au contexte d'une entreprise orchestrés dans un processus méthodologique comportant au minimum un point de variation, c'est-à-dire un paquetage de méthode ou une méthode comportant au moins un choix à effectuer entre deux éléments de ligne de méthode par un utilisateur. Une ligne peut être construite directement comme un sous-ensemble d'une famille lorsque celle-ci existe. Dans le cas contraire, la construction d'une ligne est réalisée par la formalisation des méthodes et de leurs variations utilisées en pratique dans une entreprise.

Il est à noter qu'une ligne de méthode est un concept fini qui n'est pas décrit de manière récursive. En effet, une ligne de méthode fusionne les processus méthodologiques à un seul niveau de granularité (mise à plat des processus). Il en va de même pour la partie produit de la ligne de méthode.

A l'instar des processus métier dans les entreprises, la complexité du processus d'une méthode peut se représenter par un ensemble de lignes de méthode.

Du point de vue de l'approche MaaS, une ligne de méthode est vue comme un service méthodologique qui est une spécialisation du concept de *Composant de processus technique*. En d'autres termes, une ligne de méthode est l'implémentation de l'aspect outil d'une méthode ayant un processus variable.

Une ligne de méthode est constituée d'une agrégation d'*éléments de ligne de méthode*. Ce concept abstrait regroupe tous les éléments entrants dans la composition d'une ligne de méthode tels que : les *SMA*, les *points de variation* et les *agrégations*.

Un *SMA* est le composant de base utilisé dans une ligne de méthode. C'est l'implémentation de l'aspect outil d'un composant de méthode. Une nouvelle méthode situationnelle est créée à partir du processus de configuration d'une ligne de méthode. Les décisions prises sur l'ensemble des points de variations permettent de sélectionner un ensemble de *SMA* adaptés à un contexte de projet. L'agrégation puis l'exécution des *SMA* sélectionnés dans la ligne méthode forment ainsi un outil composite pour l'application en pratique à la méthode situationnelle nouvellement créée.

Le concept d'*agrégation* regroupe un ensemble d'éléments de ligne de méthode reliés par un lien de composition. Une agrégation de plusieurs éléments implique qu'ils sont obligatoires. Cela signifie qu'ils doivent tous être exécutés dans le déroulement du processus de la méthode. Le lien de composition entre plusieurs éléments de ligne de méthode formé par un point d'agrégation peut être caractérisé par son type : *séquence*, *parallèle* ou *itératif*.

Une *agrégation* de type *séquence* entre plusieurs *éléments de ligne de méthode* agrégés implique que les éléments constituant l'agrégation doivent être exécutés dans un ordre précis. Par exemple, dans une méthode d'ingénierie de logiciels classique ayant dans sa *ligne de méthode* une *agrégation* en *séquence* de *SMA* supportant les activités d'analyse, de développement et de tests : le *SMA* outillant le développement devra être exécuté après la fin du *SMA* outillant les activités et avant celui outillant la phase de tests.

Une *agrégation* de type *parallèle* entre un *élément de ligne de méthode* et une *agrégation* implique que cet élément peut être exécuté simultanément aux autres éléments de l'agrégation. Par exemple, dans une *ligne de méthode* comportant une *agrégation* de type *parallèle* avec les *SMA* de support aux activités de tests unitaires et de tests d'acceptation, ces deux *SMA* peuvent être exécutés en même temps. L'exécution de la *ligne* reprendra lorsque l'exécution des deux *SMA* sera terminée.

Une *agrégation* de type *itérative* entre un ou plusieurs éléments implique que ce groupe d'*éléments de ligne de méthode* peut être exécuté une ou plusieurs fois. Par exemple, pour une *ligne de méthode* concernant le début d'un projet de développement de logiciel, une *itération* de deux *SMA* : définition

des besoins et affinement des besoins implique que ces deux *SMA* peuvent être exécutés plusieurs fois jusqu'à ce que le niveau de détail des besoins soit suffisant pour le projet.

Les points de décision dans une ligne de méthode sont portés par les *points de variations*. Ils relient des *éléments de ligne de méthode* par des *liens de choix*. La cardinalité d'un *point de variation* caractérise le nombre de possibilité de choix parmi toutes ses alternatives. Cette cardinalité s'exprime en un nombre d'éléments à sélectionner au minimum et au maximum parmi l'ensemble des éléments constituant les alternatives du point de variation. Cela signifie que pour un *point de variation* constitué d'un ensemble de n *éléments de ligne de méthode* reliés par un lien de choix avec une cardinalité de (i, j) avec $i \in \{0, 1, 2, \dots, n\}$ et $j \in \{1, 2, \dots, n\}$ tel que $i < j$ implique un choix d'une sélection entre i et j éléments parmi les n reliés par le lien de choix. Par exemple, un lien de choix reliant trois *SMA* avec une cardinalité de $(0,1)$ est une variabilité de type alternative telle que nous l'avons présenté à la section 5.3.1 de ce chapitre, c'est-à-dire qu'un utilisateur devra faire un choix exclusif entre ces trois alternatives. Une variabilité de type optionnelle pour ce même lien de choix serait caractérisée par exemple par une cardinalité de $(1,3)$. Par exemple, un point de variation proposant un choix exclusif en plusieurs types de modélisation statique d'une solution, comme Entité-Relation ou UML, a une cardinalité de choix minimum de 1 et maximum de 1 élément parmi les 2 éléments disponibles, c'est-à-dire qu'un utilisateur devra choisir l'une des deux alternatives.

Afin de guider les utilisateurs dans leur choix au niveau d'un point de variation, nous proposons d'identifier les caractéristiques discriminantes (nommées facteurs de contingence) entre les éléments de ligne de méthode constituant les différentes alternatives du point de variation. Pour un point de variation, toutes ses alternatives sont évaluées par un même ensemble de facteurs de contingence. Le concept d'affectation représente la valeur d'un facteur de contingence pour un élément de ligne de méthode relié à un point de variation. Un guide de configuration synthétise de manière informelle les valeurs d'affectations à satisfaire pour le choix d'une variante. La taille des projets, la disponibilité du client, la taille de l'équipe de développement et la versatilité de l'environnement métier sont des exemples de facteurs de contingence s'appliquant sur des composants de gestion du cycle de vie d'un projet de développement de SI. Une recommandation privilégiant une gestion de cycle de vie en spirale si l'environnement métier est sujet à beaucoup de changements, que la disponibilité du client est grande pour un projet et que l'équipe du projet est de taille moyenne ou réduite est un exemple de guide lié à ces facteurs.

Un guide de configuration peut également être une recommandation concernant le processus de la ligne de méthode, comme par exemple une recommandation sur la précédence d'un élément sur un autre ou encore l'incompatibilité entre deux éléments. Ce qui est modélisé, en règle générale, par des contraintes. Il est par exemple déconseillé d'utiliser sur le même produit, et l'un à la suite de l'autre, deux composants ayant des paradigmes et des langages de modélisation différents.

La fusion des processus méthodologiques et des bonnes pratiques des méthodes Scrum (Schwaber, 2011) et XP (Wells, 2009) autour du thème de la gestion d'une itération de développement dans un projet agile (GIDPA) est un exemple de ligne de méthode. L'usage du métamodèle de ligne méthode pour formaliser cet exemple est détaillé à la section 5.3.3.2 et synthétisée à la figure 5.34.

5.3.3 Construction d'une ligne de méthode

Une ligne de méthode est produite à partir d'un ensemble de méthodes utilisées dans une entreprise ou une famille de méthodes. Son objectif est d'externaliser dans un seul modèle les variations des adaptations de méthode au contexte d'une entreprise. Le processus de construction d'une ligne de méthode se déroule en cinq étapes comme le présente la figure suivante.

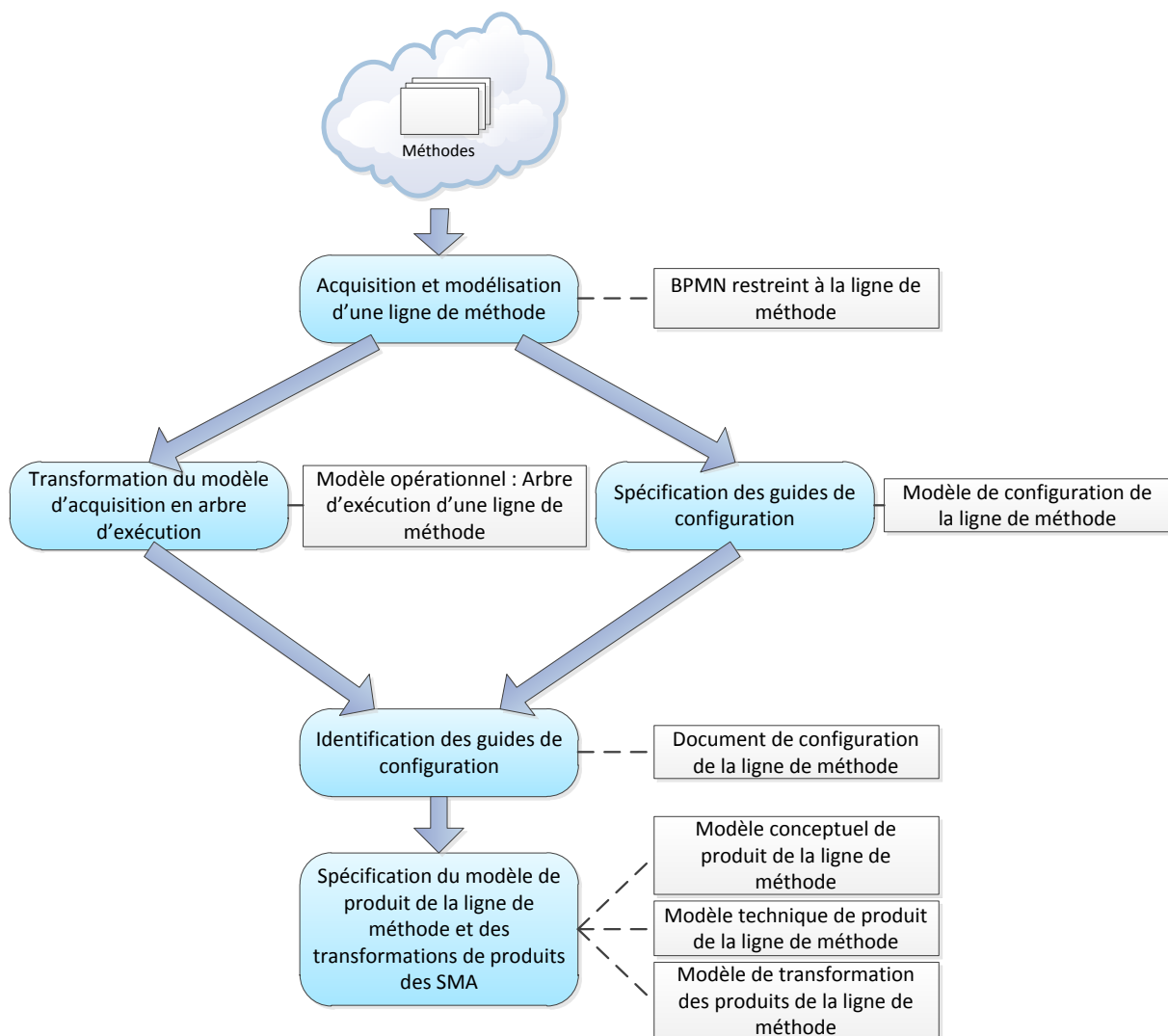


Figure 5.22 : Processus de construction d'une ligne de méthode

La première étape de ce processus consiste à l'acquisition du processus de la ligne de méthode adapté à une entreprise à partir de ses méthodes (voir figure 5.22). Cette étape, décrite aux sections 5.3.3.1 et

5.3.3.2 de ce chapitre, permet de produire une modélisation BPMN (OMG/BPMN, 2011) du processus de la ligne de méthode.

Les deux étapes suivantes (la seconde et la troisième) du processus de construction d'une ligne de méthode peuvent être exécutées en parallèle.

La seconde étape du processus de construction d'une ligne est la transformation du modèle d'acquisition en un modèle opérationnel prenant la forme d'un arbre d'exécution du processus de la ligne. La structure hiérarchique de ce modèle opérationnel est décrite dans un document XML. Cette étape de transformation est présentée à la section 5.3.3.3.

La troisième étape de la production d'une ligne est présentée à la section 5.3.4. Elle consiste à l'indentification des guides de configuration de la ligne de méthode afin d'aider un utilisateur dans ses décisions concernant la configuration du processus de la ligne. Pour chaque point de variation du processus BPMN, des guides de configuration sont déterminés en fonction des facteurs de contingence identifiés sur chacune de ces variantes. L'ensemble des facteurs et des guides est ensuite reporté soit directement en tant que notes dans le modèle de processus BPMN soit dans un document annexe de configuration de la ligne.

La quatrième étape du processus consiste en la spécification d'un modèle de configuration opérationnel de la ligne de méthode au format XML. Pour cela, les facteurs de contingence et les guides de configuration identifiés à l'étape précédente sont liés respectivement aux SMA et aux points de variation du modèle opérationnel du processus de la ligne, c'est dire l'arbre d'exécution de la ligne de méthode.

La cinquième étape de ce processus concerne la spécification d'un modèle de transformation des produits en entrée et en sortie de la ligne et de ses SMA. Cette étape commence par la production d'un modèle de produit conceptuel global de la ligne de méthode. Ce modèle est ensuite transformé en un modèle de produit opérationnel au format XSD. Enfin, un modèle de transformation est établi à partir du modèle de produit XSD de la ligne de méthode et de chacun des modèles de produit XSD des SMA. Pour chaque SMA, une correspondance de ses entrées et ses sorties avec le modèle de produit global sont décrites par des règles de transformation. Ces règles sont ensuite regroupées dans le modèle de transformation des entrées et des sorties qui prend la forme d'un document XML. Cette étape est présentée à la section 5.3.5 de ce chapitre.

Une ligne de méthode étant un service méthodologique, elle se doit d'être décrite par un descripteur de service méthodologique (voir chapitre 4) afin de pouvoir être publiée et recherchée dans l'annuaire de services méthodologiques.

5.3.3.1. *Le standard BPMN adapté à la modélisation d'une ligne de méthode*

Le standard Business Process Modeling Notation (BPMN) est un standard de modélisation des processus métiers dans les entreprises. C'est un standard proposé par l'OMG dont la première version a été publiée en 2006. La version actuelle de BPMN est la version 2.0 (OMG/BPMN, 2011). Ce standard de modélisation est une norme de représentation graphique des différentes étapes ou activités d'un processus. Il fournit les moyens de représenter le flux d'un processus du début jusqu'à sa fin. Il permet donc de représenter les séquences et les variations des activités d'un processus ainsi que les flux de messages entre ces activités (OMG/BPMN, 2011). A la différence des autres langages de modélisation des processus, le langage BPMN est orienté sur l'opérationnalisation des processus métier. En effet, il peut être utilisé conjointement au standard BPEL (OASIS/BPEL, 2007) pour opérationnaliser un processus.

Une ligne de méthode est un processus méthodologique déployé dans une entreprise pour le développement des SI. Nous avons, en conséquence, choisi d'utiliser le standard de modélisation BPMN pour modéliser les processus des lignes de méthode des entreprises. C'est un standard largement utilisé par les entreprises, celles-ci sont donc familiarisées avec son utilisation et cela permet de faciliter l'usage des lignes de méthode en réduisant les efforts de modélisation ou de formation pour l'application du métamodèle de ligne de méthode.

Ce standard dispose de suffisamment d'éléments de modélisation pour pouvoir représenter tous les types de point de variation introduit dans le métamodèle de la variabilité présenté à la figure 5.19 de la section 5.3.1 de ce chapitre ce qui n'est pas le cas de tous les langages de modélisation des processus. Par exemple, sur un diagramme d'activité UML (OMG/UML, 2011), il n'est pas possible de représenter un point de variation représentant une variabilité optionnelle entre plusieurs alternatives, entre zéro et n alternatives peuvent donc être sélectionnées. Dans un diagramme d'activité, seuls les points de variation représentant des variabilités obligatoires ou alternatives sont disponibles.

L'utilisation de BPMN, de par son caractère standard, permet aux approches d'IMS existantes de créer facilement des règles de transformation des modèles de processus qu'elles utilisent pour représenter les processus des méthodes situationnelles vers une représentation BPMN. Cela permet d'utiliser n'importe quelle approche d'IMS pour créer des composants de méthode, des *Paquetages méthodologiques* ou des méthodes situationnelles et de les assembler dans une ligne de méthode modélisée avec BPMN. A l'inverse, il est possible de partir d'une famille de méthode existante d'en sélectionner une ligne de méthode comme un ensemble de composants de méthode et de la modéliser en BPMN afin de l'opérationnaliser.

Le processus d'une ligne de méthode étant moins complexe qu'un processus métier, seul un sous-ensemble des éléments de modélisation du standard BPMN sera utilisé. La modélisation d'une ligne de méthode sera donc effectuée avec les éléments de modélisation de BPMN correspondant aux différents

concepts identifiés sur notre métamodèle de ligne de méthode (voir figure 5.21 de la section 5.3.2 de ce chapitre). En d'autres termes, seuls les éléments de modélisation de BPMN correspondant aux spécificités de l'application de la variabilité aux méthodes d'IMS sont utilisés pour représenter la ligne de méthode. Ces spécificités doivent également apparaître de manière claire sur le modèle du processus de la ligne de méthode afin d'être aisément opérationnalisés. C'est pourquoi nous n'utilisons qu'une partie des concepts disponibles dans BPMN et que nous les stéréotypons de manière à ce qu'ils soient conformes aux concepts de notre métamodèle de ligne de méthode (point de variation, lien de choix, séquence, parallèle, itératif, SMA, guide de configuration, facteur de contingence et affectation). Il est à noter que nous ne proposons pas une extension de BPMN, mais une restriction d'utilisation de ses éléments de modélisation à certains stéréotypes prédéfinis. La suite de cette section présente l'ensemble des stéréotypes disponibles pour la modélisation du processus d'une ligne de méthode.

Le début et la fin du processus

La figure 5.23, présente les éléments modélisant le début d'un processus d'une ligne de méthode et celui modélisant sa fin. Toute ligne de méthode doit avoir un élément de commencement et un élément de fin.

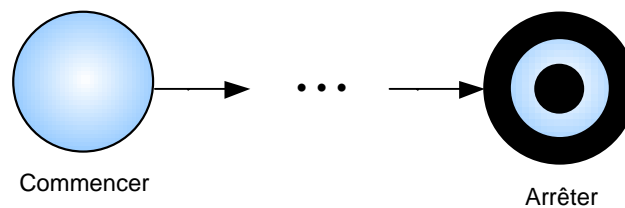


Figure 5.23 : Éléments de début et fin d'un processus BPMN

De plus, tous les chemins d'un processus d'une ligne de méthode doivent avoir comme premier élément "commencer" et comme dernier élément "arrêter".

Les alternatives : composants de méthode / SMA

Un composant de méthode ou SMA est représenté dans le modèle BPMN de la ligne de méthode par un élément "activité" du standard BPMN (voir figure 5.24).

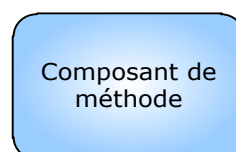


Figure 5.24 : Élément représentant un composant de méthode dans une ligne de méthode

Les alternatives d'un point de variation peuvent être des SMA ou d'autres points de variation tels que nous l'avons défini dans notre métamodèle de ligne de méthode (voir figure 5.21).

Les points d'agrégation séquentiels

Une séquence entre deux éléments de ligne de méthode est représentée par un élément de flux séquentiel du standard BPMN comme présenté à la figure suivante.

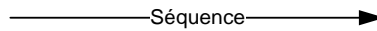


Figure 5.25 : Élément représentant une séquence entre deux éléments

Un processus BPMN étant un flux, plusieurs éléments de séquence consécutifs sont regroupés un seul point d'agrégation séquentiel du métamodèle de ligne de méthode.

La figure ci-dessous présente un exemple de flux séquentiel entre trois SMA.

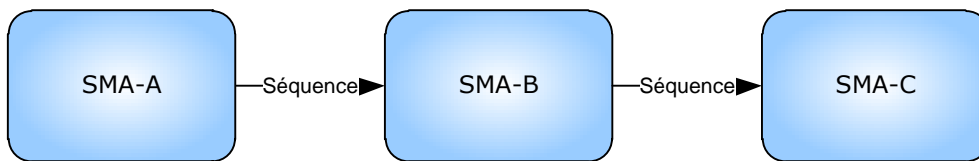


Figure 5.26 : Enchaînement de flux séquentiels

Comme le montre l'exemple présenté à la figure 5.26, les deux séquences consécutives représentent le point d'agrégation séquentiel entre les trois SMA-A, SMA-B et SMA-C.

Les points d'agrégation parallèles

Un point d'agrégation de type parallèle est représenté dans le standard BPMN par deux éléments : un élément "AND" de type "split" et un élément "AND" de type "join" (voir figure 5.27).

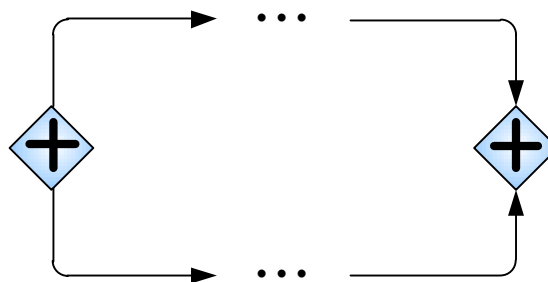


Figure 5.27 : Éléments représentant un point d'agrégation parallèle

Le premier élément de type "split" modélise la séparation du flux du processus en n flux parallèles (minimum 2). Les différents chemins parallèles du processus se rejoignent ensuite en un élément de type "join" permettant une synchronisation des flux. Il est à noter que l'ensemble des flux parallèles émis à partir d'un élément de type "split" doit obligatoirement se rejoindre directement ou indirectement sur le même élément de type "join". De plus, comme tout point d'agrégation, tous ces flux parallèles doivent être exécutés.

Les points d'agrégation itératifs

La figure 5.28 présente l'élément de modélisation d'un point d'agrégation itératif dans le processus d'une ligne de méthode. Du point de vue du standard BPMN, c'est une condition avec une séquence pointant sur un élément antérieur dans le processus.

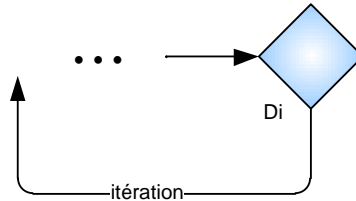


Figure 5.28 : Élément représentant un point d'agrégation itératif

Tel que défini dans notre métamodèle de ligne de méthode, le point d'agrégation itératif s'exécute au moins une fois avant de tester la condition de continuation de la boucle. C'est pourquoi la condition est placée à la fin du flux de cette boucle dans ce stéréotype d'élément de modélisation BPMN. Il est recommandé d'attacher l'expression de la condition directement sur l'élément de condition ou, pour plus de lisibilité du modèle, d'identifier le point de condition par un identifiant unique afin d'indexer la condition dans un document annexe.

Les liens de choix

Les liens de choix entre des points de variation et des éléments de ligne de méthode sont modélisés par un flux BPMN stéréotypé (voir figure 5.29).

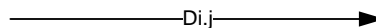


Figure 5.29 : Élément de modélisation représentant le lien de choix

Afin d'attacher plus facilement des facteurs de contingence et leurs affections aux liens de choix, ils doivent être identifiés par un identifiant unique et relatif à leur point de variation parent. Par exemple un point de variation "D1" avec deux liens de choix : ces derniers auront une identification relative à leur parent "D1" par exemple "D1.1" pour le premier lien de choix et "D1.2" pour le deuxième.

Les points de variations

Un point de variation tel que nous l'avons défini dans le métamodèle de ligne de méthode, peut exprimer une variabilité alternative ou optionnelle. Il n'existe pas d'élément de modélisation dans le standard BPMN permettant à la fois d'exprimer une alternative ou une option. En revanche, il existe des éléments différents permettant de représenter ces deux concepts (voir les trois figures suivantes).

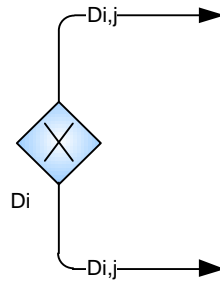


Figure 5.30 : Élément représentant un point de variation avec une variabilité alternative au niveau processus

Un point de variation représentant une variabilité de type alternative est modélisé dans le processus de la ligne de méthode par un élément BPMN de type "XOR" ou "OU EXCLUSIF" (voir figure 5.30). Nous recommandons d'utiliser ce stéréotype pour modéliser une décision à prendre par un utilisateur au niveau du processus de la ligne de méthode, par exemple un choix entre plusieurs composants de méthode/SMA.

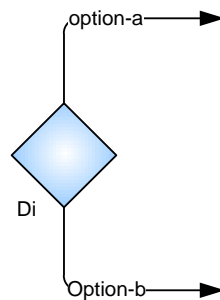


Figure 5.31 : Élément représentant un point de variation avec une variabilité alternative au niveau produit

La figure 5.31 présente, quant à elle, une condition au niveau du produit, c'est-à-dire un point de variation de type alternatif portant une condition sur un produit et proposant diverses alternatives en fonction du résultat de la condition. Pour cet élément, la condition doit être exprimée sur l'élément de modélisation ou alors être identifiée par un identifiant unique afin d'indexer la condition dans un document annexe pour des raisons de lisibilité du modèle. De plus, chaque lien de choix en sortie de l'élément porteur de la condition doit être nommé en fonction du résultat de la condition attendue pour emprunter ce chemin.

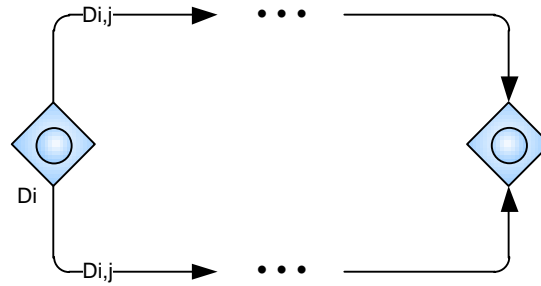


Figure 5.32 : Élément représentant un point de variation avec une variabilité optionnelle

Enfin, un point de variation représentant une variabilité optionnelle est modélisée dans le processus de la ligne de méthode par un élément BPMN de type "OR" ou "OU" (voir figure 5.32). Cet élément se présente en deux parties une partie "split" et une partie "join". La partie "split" modélise le début de l'option avec l'ensemble des alternatives possibles. La partie "join", quant à elle, permet de synchroniser tous les flux optionnels initiés par la partie "split" pour reprendre en sortie le flux du processus de la ligne de méthode.

Chaque point de variation dans le processus de la ligne de méthode doit être identifié par un identifiant unique afin de pouvoir leur attacher facilement des guides de configuration. De même, les liens de choix reliant chaque point de variation à ces alternatives (éléments de ligne de méthode dans une ligne de méthode) doivent être identifiés par un identifiant unique et relatif à leur point de variation parent dans le but de leur adjoindre des facteurs de contingence.

Les facteurs de contingence, leurs affectations et les guides de configuration

Concernant la modélisation des guides de configuration, nous proposons de les ajouter aux nœuds de décisions par des éléments "notes" ou "commentaires" du standard BPMN (voir figure 5.33). Pour des raisons de lisibilité du modèle de processus d'une ligne de méthode, il est également possible de définir les guides de configuration en les reliant à leur point de variation par son identifiant unique.

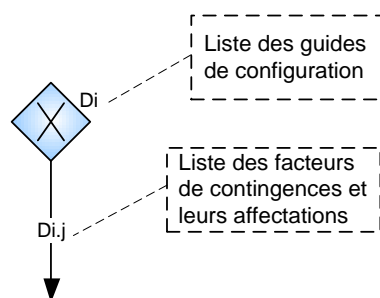


Figure 5.33 : Facteurs de contingence, affectations et guides de configuration attachés aux éléments

De la même manière que pour les guides de configuration, les facteurs de contingence et leurs affectations sont reliés aux liens de choix dans le modèle du processus par des éléments de type "notes" ou indexé par l'identifiant du lien de choix dans un document annexe au modèle.

Les éléments et des règles de modélisation associées à ceux-ci dans cette section permettent de modéliser facilement la variabilité des méthodes d'une entreprise. Nous présentons à la section suivante la démarche utilisée pour acquérir le processus de la ligne et produire un modèle à partir de ces éléments de modélisation.

5.3.3.2. Processus de modélisation d'une ligne de méthode

Nous proposons de définir une ligne de méthode comme la modélisation des variantes et points de variation d'un processus méthodologique d'une entreprise autour d'un même thème. En effet, une ligne de méthode regroupe plusieurs méthodes et leurs variations utilisées dans une entreprise. Elle met en avant les choix que doivent faire les ingénieurs méthodes entre les variantes des méthodes de l'entreprise lors du déroulement d'un projet. A l'image des processus métier, les lignes de méthode regroupent dans un même processus les variantes de méthode autour d'un thème, par exemple le lancement d'un projet ou encore le développement d'un incrément logiciel. Cela permet la représentation de la méthodologie générale d'une entreprise sous la forme de paquetages méthodologiques autonomes évoluant en séquence ou en parallèle. A chaque ligne de méthode correspond ainsi un paquetage de la méthode d'une entreprise.

La construction d'une ligne de méthode peut être réalisée de deux manières différentes : à partir d'une famille de méthode ou à partir de la modélisation du processus méthodologique existant dans une entreprise.

Une famille de méthode regroupe un ensemble de composants de méthode sémantiquement proches. Une ligne de méthode peut être produite à partir d'une famille de méthode, pour une entreprise donnée, par la sélection des composants de méthode de la famille applicables à son contexte. Il résulte de cette sélection un processus méthodologique centré sur la variabilité entre les composants de méthode.

Inversement, pour produire une ligne de méthode à partir des méthodes existantes dans une entreprise, il est possible d'utiliser n'importe quelle approche d'IMS pour décomposer le processus méthodologique en composants de méthode. Pour cela, les principes de décomposition associés à l'approche d'IMS choisie sont utilisés pour identifier des composants dans la méthodologie de l'entreprise et modéliser la variabilité entre les composants identifiés dans le processus d'une ligne de méthode. Il est également possible de produire une ligne de méthode par la modélisation des processus méthodes de l'entreprise à l'aide du standard BPMN et de faire correspondre à chaque activité un composant de méthode et un SMA pour l'outiller.

A l'issue de la démarche de modélisation d'une ligne de méthode, le processus permet d'identifier les composants de méthode nécessaires à l'entreprise et, par conséquent, de rechercher pour chacun d'entre eux, un SMA, l'outiller dans l'annuaire de services méthodologiques ou de développer un SMA si aucun SMA existant ne répond aux besoins du composant de méthode.

Voici un exemple de ligne de méthode de gestion d'une itération de développement dans un projet agile (GIDPA), illustré à la figure suivante.

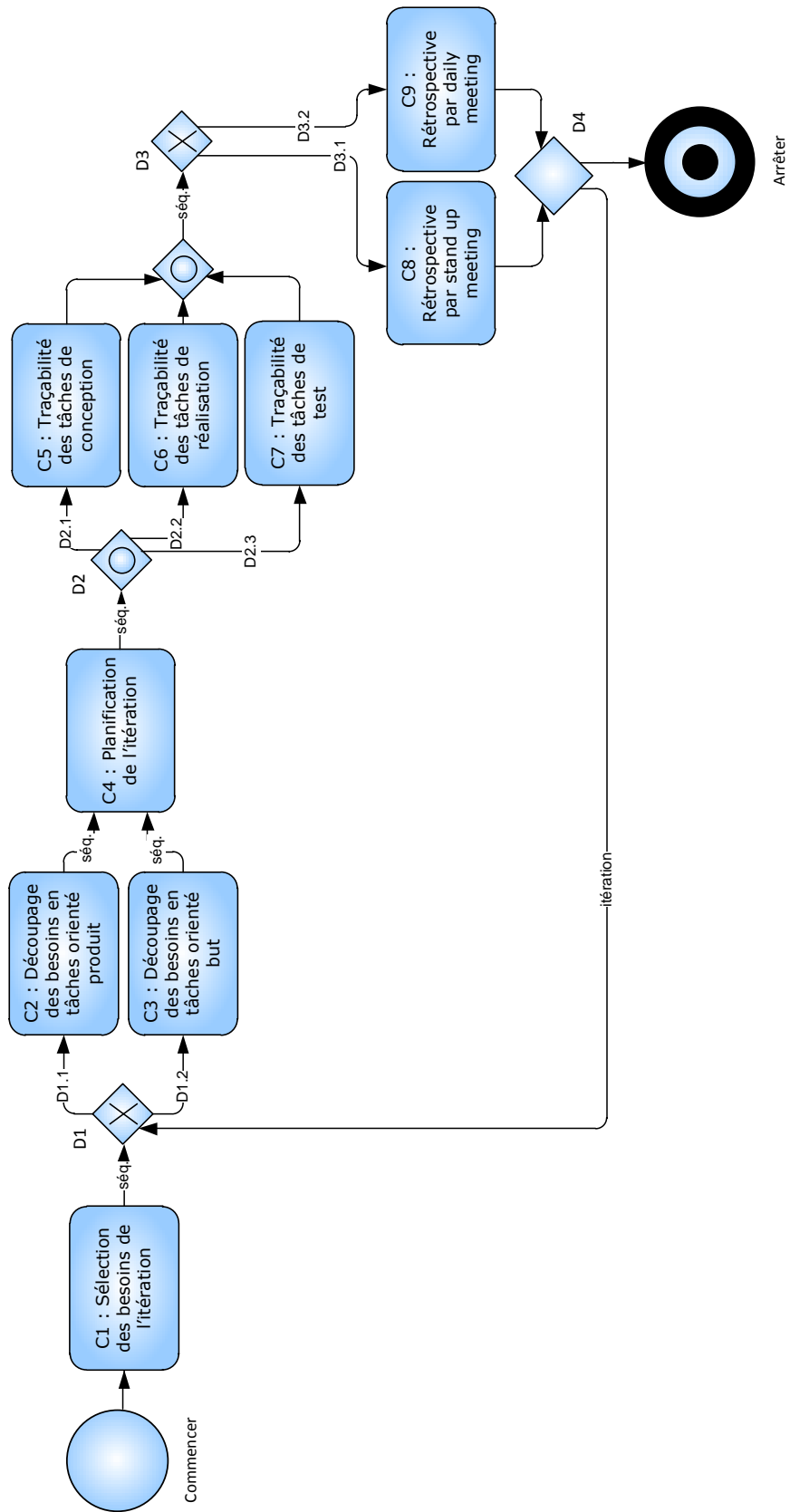


Figure 5.34 : Processus d'une ligne de méthode d'exécution d'une itération de développement

La figure 5.34 présente par exemple la modélisation du processus de la ligne de méthode GIDPA. Cette ligne regroupe les bonnes pratiques des méthodes agiles XP (Wells, 2009) et Scrum (Schwaber, 2011) concernant le déroulement d'une itération. Ces deux méthodes agiles sont utilisées dans l'entreprise, et nous avons représenté dans ce processus les différentes bonnes pratiques de ces deux méthodes pour l'exécution d'une itération de développement.

La modélisation du processus de cette ligne est le résultat de la modélisation conjointe du processus des deux méthodes agiles. En effet, du point de vue de la gestion d'un projet agile (XP ou Scrum), une itération se décompose en cinq groupes d'activités reliées par des liens de séquence.

La première étape concerne la sélection d'un sous-ensemble de besoins à implémenter dans l'itération parmi la liste des besoins. Cette activité est réalisée de la même manière dans la méthode XP ou dans Scrum. Les besoins sont sélectionnés parmi les plus prioritaires et ceux apportant la plus forte valeur ajoutée pour le client (Wells, 2009), (Schwaber, 2011). Cela est représenté par le composant C1 dans le processus de la ligne de méthode. Ce composant peut être implémenté par un SMA fournissant une interface permettant d'établir une sélection de besoins parmi la liste complète.

Ensuite, dans une deuxième étape, les besoins de haut niveau à implémenter dans l'itération sont affinés jusqu'à une granularité assez fine pour pouvoir établir des tâches pour les développer. Cette activité est réalisée d'une manière différente en fonction de la méthode agile utilisée. En effet, XP préconise de décomposer les besoins en sous-besoins plus simples à développer jusqu'à pouvoir leur faire correspondre des tâches (Wells, 2009). Il s'agit donc d'une décomposition axée sur le produit. De son côté la méthode Scrum préconise de définir le but de l'itération en fonction des besoins de haut niveau qui ont été sélectionnés. La décomposition des besoins en tâches est réalisée en présence du client et pour chaque besoin l'équipe s'accorde sur leurs conditions de satisfaction, c'est-à-dire la post condition de l'implémentation du besoin au niveau de tous les produits (Schwaber, 2011). Cette méthode est plus formelle que XP et est orientée sur les buts pour la décomposition des besoins en tâches. Cette différence se traduit sur le modèle de la ligne par deux composants distincts C2 et C3 issus respectivement de XP et de Scrum. Ces deux variantes d'identification des tâches à partir des besoins sont exclusives. En effet, l'utilisateur appliquera soit la variante dirigée par le produit ou la variante dirigée par les buts. Le choix entre ces deux composants est donc décrit par un élément "ou exclusif" dans le processus modélisé avec BPMN. Les SMA supportant ces deux composants doivent fournir une interface de saisie des tâches relatives aux besoins.

Après avoir établi une liste de tâches à effectuer dans l'itération, il convient de fournir une estimation de ces tâches, puis de les affecter à des acteurs et de les répartir dans le planning de l'itération. Les estimations des tâches sont ajustées en fonction de l'indice de vitesse du projet de l'itération précédente (indice de productivité). Cette étape est réalisée de manière similaire dans les deux méthodes. Elle est donc modélisée dans la ligne de méthode par un composant C4 de planification de

l'itération. L'outillage de ce composant peut être réalisé par un SMA fournissant les facilités pour la création d'un planning de l'itération ordonnant les tâches et la répartition des acteurs pour les réaliser.

Dans un quatrième temps vient l'exécution des tâches en elles-mêmes. Dans un projet de développement de SI, ces tâches peuvent être des tâches de conception, de réalisation ou de test. Du point de vue de la gestion du projet il est important d'avoir une traçabilité de ces tâches et c'est ce point qui est pris en charge par les trois composants de méthode C5, C6 et C7. Les outils associés à ces composants doivent permettre de tracer l'état d'une tâche de "à faire" à "fini" et d'avoir une interface de saisie des obstacles rencontrés et des rapports d'exécution en fonction des spécificités du type de la tâche exécutée. Ces composants sont communs aux méthodes XP et Scrum qui préconisent une gestion du planning journalière. L'utilisation de ces trois composants dépend de la planification de l'itération et des tâches à exécuter dans une journée. En conséquence, les composants C5, C6 et C7 sont reliés dans le modèle de processus de la ligne à un point de variation optionnel ou encore un élément de type "OU".

Dans un dernier temps, la boucle journalière d'une itération se termine le matin de la journée suivante par une rétrospective de la journée écoulée. La manière de régir cette réunion diffère entre les méthodes XP et Scrum. Pour la première, c'est une réunion plutôt informelle ou en fonction des traces des exécutions des tâches. L'équipe fait le point sur l'avancement journalier du projet. C'est une réunion à caractère spontanée pouvant être réalisée devant une machine pour pouvoir visualiser rapidement le code développé dans la journée. Cette réunion permet également de réfléchir en équipe à des solutions pour les tâches ayant rencontrées des obstacles (Wells, 2009). Pour la méthode Scrum, c'est une réunion plus formelle destinée à des équipes plus conséquentes où chaque acteur fait le point sur la journée précédente en répondant à ces trois questions : "qu'est ce qui a été fait?", "qu'est ce qui est prévu de faire?", "quels sont les obstacles rencontrés?". Cela permet d'évaluer l'avancement du projet au jour le jour et d'identifier les obstacles pouvant être la cause de retards. Contrairement à XP, les solutions ne sont pas directement proposées dans cette réunion mais les acteurs ayant des idées pour résoudre les différents obstacles sont identifiés afin de poursuivre cette réflexion ultérieurement après la réunion journalière. Un indicateur appelé "burndown chart" est actualisé au cours de cette réunion à partir des traces d'exécution des tâches. Cet indicateur permet de visualiser la progression de l'implémentation des besoins au cours du temps. Pour XP et Scrum, ces deux réunions ont pour objectifs d'identifier les obstacles dans la réalisation des tâches afin de pouvoir ajuster la planification et d'identifier des erreurs dans le développement afin de créer des tâches de correction de ces erreurs. Le composant C8 correspond à la version issue de XP de cette rétrospective journalière tandis que la version Scrum est représentée par le composant C9. Le choix entre ces deux composants est exclusif et modélisé par l'élément "OU Exclusif" D3 dans la ligne. Le choix entre les deux composants est axé sur

le degré de formalisme souhaité pour ce type de réunion. L'implémentation de ces composants doit, quant à elle, fournir une fonctionnalité de saisie du rapport de la réunion.

Le processus de la ligne de méthode effectue ensuite une boucle journalière modélisée par l'élément itératif D4 et ce jusqu'à la fin du délai imparti pour l'itération.

Ce processus décrit la gestion d'une itération et les différentes variantes spécifiques à Scrum et XP. Il intègre les variations de ces deux méthodes concernant les activités de décomposition des besoins en tâches, les activités de traçabilité et les activités de rétrospectives d'une itération de développement. Nous proposons dans la section suivante la démarche pour produire un modèle de processus exécutable à partir de ce type de modèle d'acquisition (BPMN).

5.3.3.3. Processus de construction du modèle opérationnel d'une ligne de méthode

La construction d'une ligne de méthode est réalisable à partir du modèle BPMN de son processus, c'est-à-dire à partir du modèle d'acquisition du processus de la ligne. Le processus de construction de la ligne a pour objectif la production d'un modèle opérationnel de la ligne de méthode. Une fois construit, ce support technique de la ligne peut ensuite être exécuté sur la plateforme d'exécution de l'approche MaaS.

Afin d'outiller le processus d'une ligne de méthode, nous proposons, par analogie au standard BPEL d'exécution de processus (OASIS/BPEL, 2007), de transformer les modèles d'acquisition des lignes de méthode en arbres d'exécution des SMA constituant la ligne de méthode et de les stocker au format XML (W3C/XML, 2008) afin de pouvoir faciliter leur opérationnalisation.

L'algorithme de transformation d'un modèle de processus BPMN d'une ligne en un modèle opérationnel XML consiste à parcourir le processus de la ligne de méthode sur le modèle BPMN en partant de l'élément *commencer* et ce jusqu'à l'élément *arrêter*. La structure de l'arbre XML opérationnalisant une ligne est ainsi construite de manière incrémentale élément par élément. Pour chaque élément de processus rencontré dans le parcours du processus, il est nécessaire d'appliquer une règle de transformation correspondant à son stéréotype. Il existe, par conséquent, autant de règles de transformation que de stéréotypes d'élément de modélisation utilisés pour produire un modèle de processus BPMN d'une ligne de méthode. L'ensemble de l'arbre d'exécution est ensuite rattaché à un nœud racine représentant la ligne de méthode. Les sous-sections suivantes présentent pour chaque élément de modélisation sa règle de transformation (le schéma XML permettant de valider les modèles de lignes est présenté dans l'annexe D).

Les alternatives : composants de méthode / SMA

Les activités du modèle BPMN modélisant le processus d'une ligne de méthode représentent les services méthodologiques composant la ligne (voir la figure suivante).

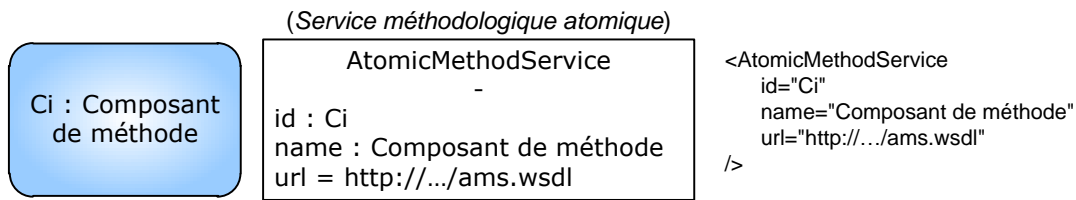


Figure 5.35 : Transformation d'une activité en élément "AtomicMethodService"

Elles sont transposées dans le modèle opérationnel sous la forme d'éléments *AtomicMethodService* dans le document XML représentant la ligne (voir figure 5.35). Cet élément représente un SMA exécutable de la ligne de méthode, c'est-à-dire le concept de "Service méthodologique atomique" du métamodèle de ligne de méthode (voir figure 5.21). L'élément XML *AtomicMethodService* contient par conséquent tous les éléments nécessaires à son invocation, c'est-à-dire : son identifiant (*id*) unique dans le modèle opérationnel, le nom du SMA porté par l'attribut "name" et l'attribut "url" représente l'adresse du descripteur technique du SMA permettant de l'invoquer. Le modèle opérationnel de ligne de méthode est un arbre décisionnel représentant le processus de sélection des éléments exécutables de type SMA : les éléments *AtomicMethodService*. Par conséquent, ils représentent toujours les feuilles de cet arbre.

Les points d'agrégation séquentiels

Une séquence représente un enchaînement obligatoire entre deux blocs de processus dans le modèle BPMN d'une ligne de méthode (voir la figure ci-dessous).

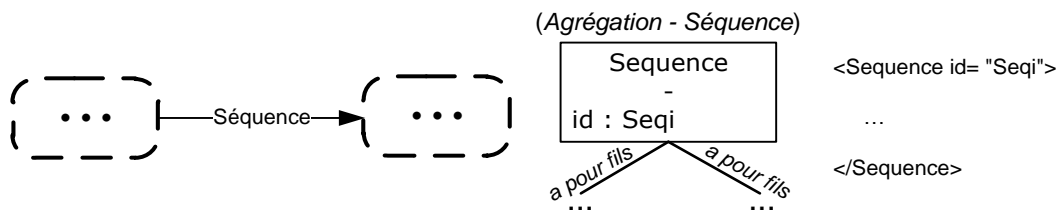


Figure 5.36 : Transformation d'une séquence entre deux blocs de processus en élément "Sequence"

Une séquence ne porte donc pas uniquement sur l'élément la précédant ou l'élément la suivant mais sur un regroupement d'éléments la précédant et sur un regroupement d'éléments la suivant (voir figure 5.36). La transformation d'un concept "Séquence" du BPMN qui est binaire à un concept "Séquence" du métamodèle de ligne de méthode (voir figure 5.21) qui est n-aire implique la fusion d'une chaîne de séquence BPMN binaire dans une séquence n-aire du métamodèle de ligne de méthode. Cela se traduit par la règle de transformation entre le modèle de processus et le modèle opérationnel suivante : une séquence entre deux blocs de processus est représentée par l'ajout d'un nœud *Sequence* en tant que parent du sous-arbre représentant le bloc de processus précédant la séquence dans le processus et celui représentant le bloc de processus suivant. Cependant, si le nœud parent de ce sous-arbre est déjà un

nœud agrégatif représentant une séquence (*Sequence*) ou une itération (*RepeatUntil*) aucun nœud *Sequence* n'est créé mais en contrepartie une nouvelle branche est ajoutée à ce nœud d'agrégation parent pour le relier avec le sous-arbre du bloc de processus suivant l'élément séquence dans le modèle de processus. Un nœud *Sequence* doit obligatoirement comporter au minimum deux éléments fils dans le modèle opérationnel.

La figure suivante présente un exemple de transformation d'une séquence entre trois SMA.

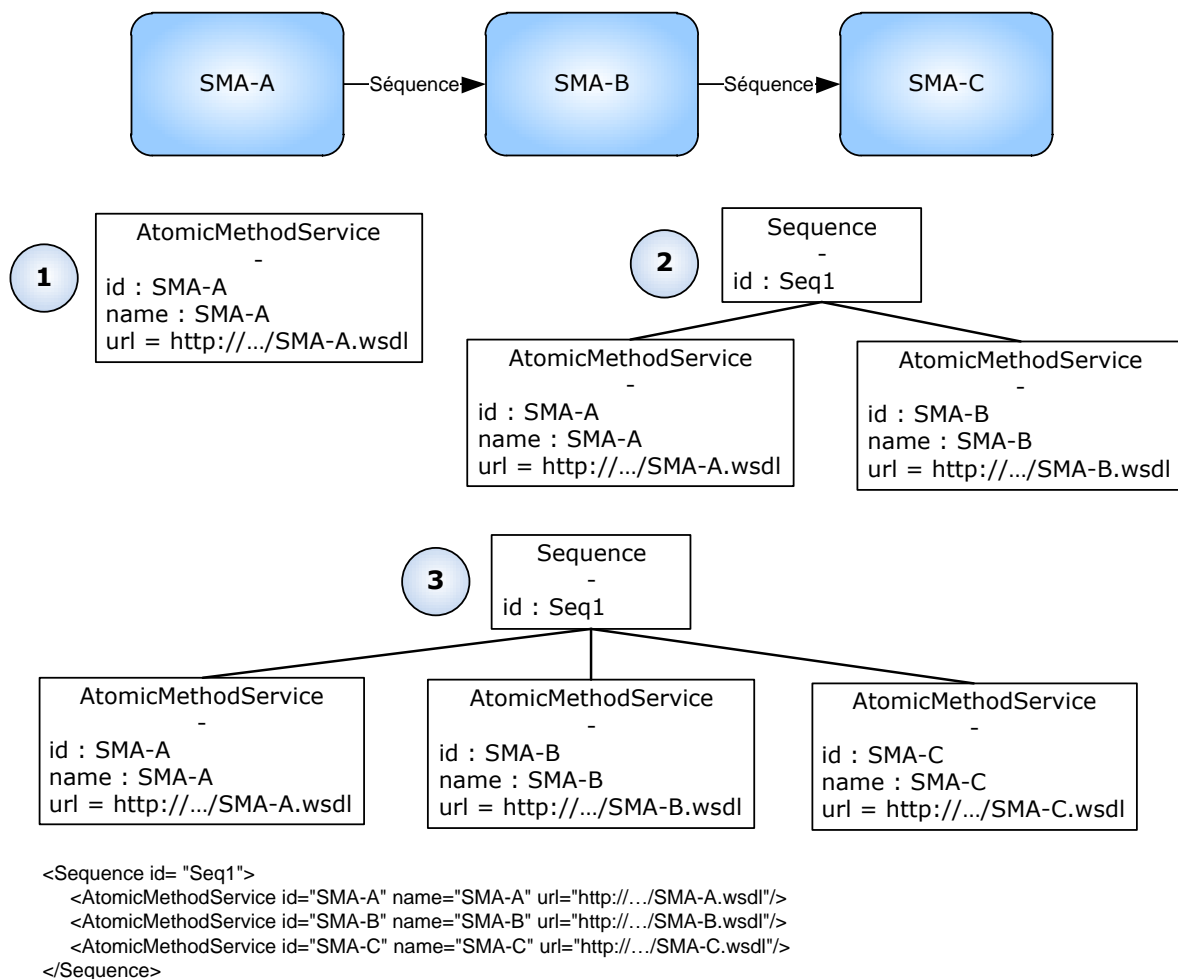


Figure 5.37 : Exemple de transformation de deux séquences entre trois SMA

La figure 5.37 présente un exemple de transformation de deux séquences entre trois SMA du modèle de processus vers le modèle opérationnel. La première étape consiste en la transformation du premier SMA vers le modèle opérationnel. Viens ensuite la transformation de la première séquence, l'élément la précédant est le SMA-A et l'élément la suivant est le SMA-B. Comme la feuille de l'arbre représentant le SMA-A n'a pas de nœud parent, un nœud *Sequence* englobant les feuilles SMA-A et SMA-B est créé. La dernière étape de la transformation dans cet exemple consiste en la transformation de la deuxième séquence de ce processus reliant le SMA-B avec le SMA-C. Etant donné que le nœud parent de la feuille SMA-B de l'arbre est un nœud *Sequence* (Seq1) qui est un type de nœud agrégatif

séquentiel, une branche est alors ajoutée à l'arbre pour relier ce nœud Seq1 avec la feuille de le SMA-C.

Les points d'agrégation parallèles

Un élément de modélisation de flux s'exécutant en parallèle dans un modèle de processus d'une ligne de méthode est constitué de deux parties : une partie "split" pour la séparation des flux et une partie "join" pour leur synchronisation (voir la figure ci-dessous).

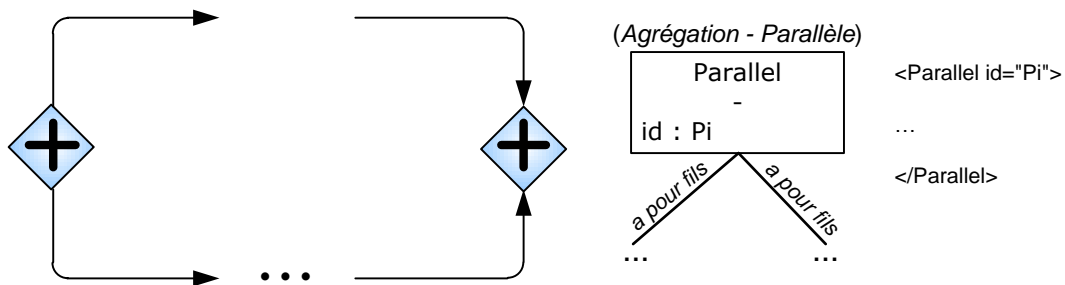


Figure 5.38 : Transformation d'un parallélisme entre plusieurs flux en un élément "Parallel"

La transformation d'un élément de modélisation de flux parallèles dans une ligne se traduit par la création d'un nœud *Parallel* dans l'arbre (voir figure 5.38). Il représente le concept d'une "Agrégation" de type "Parallèle" dans le métamodèle de ligne de méthode (voir figure 5.21). Ce nœud possède une branche pour chaque flux de processus évoluant en parallèle entre les éléments de modélisation "split" et "join". Chacune de ces branches est ensuite reliée à l'un des sous-arbres représentant un flux de processus contenu entre les deux éléments "split" et "join". De plus, un nœud *Parallel* doit obligatoirement avoir au minimum deux éléments fils qui lui sont rattachés.

Les points d'agrégation itératifs

Un point d'agrégation itératif dans le processus d'une ligne de méthode représente un fragment de processus pouvant être exécuté plusieurs fois avec un minimum d'une fois (voir la figure suivante).

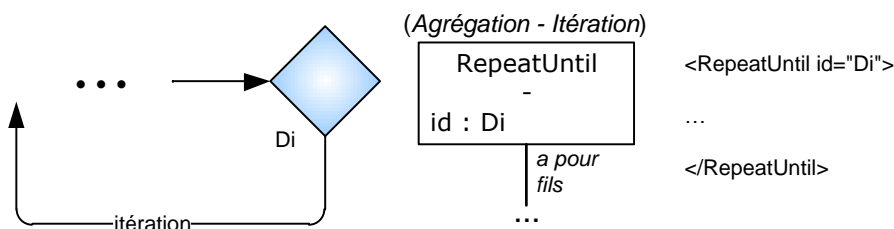


Figure 5.39 : Élément représentant un point d'agrégation itératif

La transformation de cet élément vers le modèle opérationnel de ligne de méthode est réalisé de manière simple par la création d'un nœud parent *RepeatUntil* englobant le sous-arbre représentant le fragment de processus exécutable plusieurs fois (voir figure 5.39). La condition d'arrêt de l'itération

est, quant à elle, stockée dans un guide de configuration dans un document séparé attaché au modèle opérationnel de la ligne (voir section 5.3.4 de ce chapitre). Le nœud *RepeatUntil* représente le concept d'une "Agrégation" de type "Itération" dans le métamodèle de ligne de méthode (voir figure 5.21).

Les points de variation

Les points de variation peuvent être modélisés de trois manières différentes dans le modèle de processus d'une ligne de méthode en fonction de leur type de variabilité : alternative de niveau processus ou produit et option (voir la figure ci-dessous).

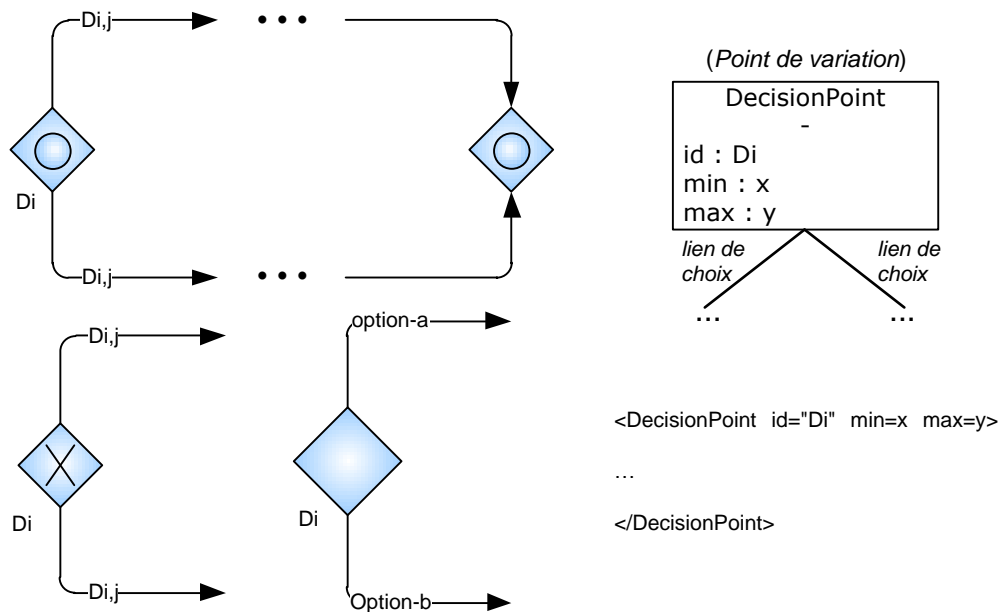


Figure 5.40 : Transformation des différents types de points de variation en élément "DecisionPoint"

Comme présenté à la figure 5.40, les trois types de points de variation sont transformés en un seul nœud nommé *DecisionPoint* dans le modèle opérationnel et pouvant les représenter tous les trois. En effet, un nœud *DecisionPoint* dispose de deux attributs min et max permettant de déterminer le nombre minimum et maximum de branches sélectionnables pour ce nœud dans l'exécution de l'arbre, de la même manière que le concept de "Point de variation" du métamodèle de ligne de méthode (voir figure 5.21) qu'il représente. Un point de variation de type alternatif a ainsi pour cardinalité minimum et maximum une seule branche sélectionnable. De son côté, un point de variation de type optionnel a une cardinalité minimum de un et une cardinalité maximum égale au nombre de branches attachées à son nœud *DecisionPoint*. Ce type de nœud proposant un choix entre plusieurs alternatives doit donc avoir un minimum de deux branches, sauf dans un cas particulier où un des liens de choix du point de variation dans le modèle de processus est relié à l'élément "arrêter" du processus. Dans ce cas précis le nœud *DecisionPoint* peut avoir une cardinalité minimum de 0 et ne comporter au minimum qu'une seule branche. De la même manière que pour les points d'agrégation itératifs, les guides de

configuration liés à un point de variation ainsi que les facteurs de contingence avec leurs affectations correspondant aux liens de choix de ce point de variation sont détaillés dans un document externe attaché au modèle opérationnel de la ligne de méthode (voir section 5.3.4 de ce chapitre).

La ligne de méthode

Une ligne de méthode, pour être opérationnelle, nécessite un arbre d'exécution de son processus, un document de configuration pour guider les utilisateurs dans leur choix et une liste des attributions et transformations de produit entre les SMA au sein du processus de la ligne de méthode (voir la figure suivante).

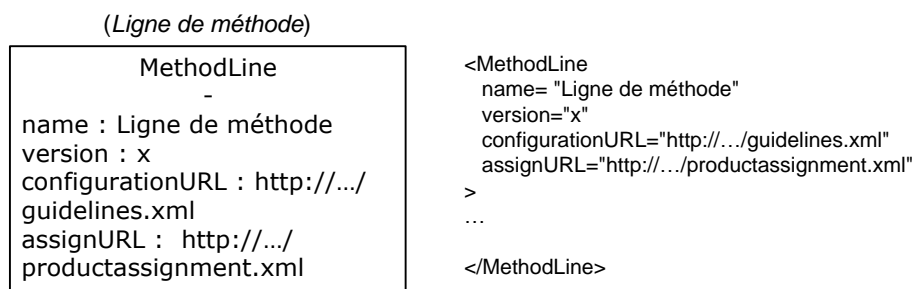


Figure 5.41 : Élément racine de l'arbre d'exécution représentant une ligne de méthode

Le nœud de l'arbre d'exécution contenant toutes les informations nécessaires à l'opérationnalisation d'une ligne est le nœud *MethodLine* qui est la racine de tout arbre d'exécution représentant une ligne de méthode (voir figure 5.41). Ce nœud représente le concept principal de "Ligne de méthode" du métamodèle de ligne de méthode présenté à la figure 5.21.

Exemple de modèle opérationnel de ligne de méthode

Nous présentons dans cette sous-section un exemple de construction d'un modèle de ligne de méthode. Pour ce faire, nous utilisons l'exemple de modèle d'acquisition de la ligne de méthode GIDPA présenté à la figure 5.34. Nous produisons ensuite le modèle de cette ligne de méthode par l'application du processus de transformation présenté à la section 5.3.3.3 de ce chapitre sur le modèle d'acquisition de la ligne. L'arbre d'exécution de cette ligne de méthode est présenté au format XML à la figure 5.43 et par une représentation graphique à la figure 5.42.

Le premier élément du processus de la ligne de méthode est le SMA C1 de sélection des besoins de l'itération. La première étape de transformation du processus consiste en la création du nœud racine *MethodLine* de l'arbre d'exécution de la ligne. Une feuille *AtomicMethodService* représentant le SMA C1 est ensuite ajoutée en tant que fils au nœud racine.

La deuxième étape de transformation est initiée par l'élément séquence entre le SMA C1 et le point d'agrégation itératif D4 dans le processus de la ligne. Cela a pour conséquence l'ajout d'un nœud *Sequence* dans l'arbre d'exécution (identifié comme Seq1) en tant que parent de la feuille

AtomicMethodService représentant C1. Un nœud *RepeatUntil* associé à l'itération D4 est également ajouté en tant que fils du nœud *Sequence* Seq1. Le fils du nœud racine de l'arbre est donc désormais le nœud *Sequence* Seq1.

L'élément suivant dans le processus de la ligne est le point de variation D1 et porte sur les SMA C2 et C3. C'est également le premier élément contenu dans l'itération D4. La troisième transformation consiste donc en la création du nœud *DecisionPoint* D1 en tant que premier fils du nœud *RepeatUntil* D4 dans l'arbre. Les feuilles de type *AtomicMethodService* représentant les SMA C2 et C3 sont ensuite ajoutées à l'arbre en tant que fils du nœud *DecisionPoint* D1. Etant donné que le point de variation D1 est de type XOR, les cardinalités du nœud *DecisionPoint* le représentant sont de 1 pour la cardinalité min et la cardinalité max.

La quatrième étape de la transformation du processus de la ligne en arbre exécutable concerne la séquence portant sur le bloc de processus formé par les éléments D1, C2 et C3 en les reliant avec le SMA C4. Etant donné que le nœud D1 a pour parent un nœud de type *RepeatUntil* dans l'arbre. Il n'est pas nécessaire de créer un nœud *Sequence*, une feuille *AtomicMethodService* représentant le SMA C4 est directement ajoutée en tant que second fils du nœud *RepeatUntil* D4.

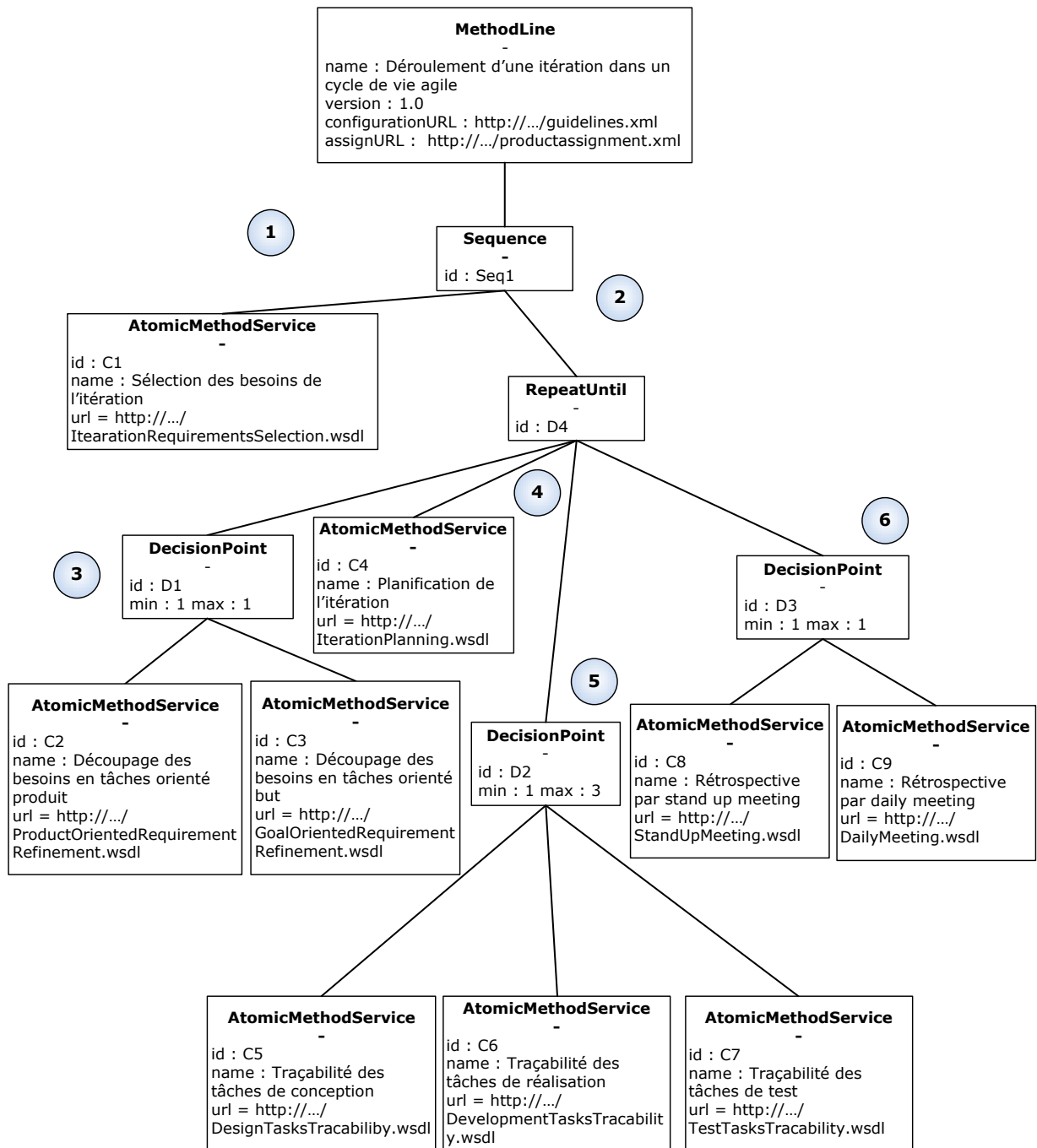


Figure 5.42 : Exemple de représentation graphique de l'arbre d'exécution de la ligne de méthode
GIDPA

```
<?xml version="1.0" encoding="UTF-8"?>
<ml:MethodLine xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:ml='http://crinfo.univ-paris1.fr/schema/MethodLine'
xsi:schemaLocation='http://crinfo.univ-paris1.fr/schema/MethodLine MethodLine.xsd'
name="Déroulement d'une itération dans un cycle de vie agile"
version="1.0"
configurationUrl="http://.../guidelines.xml"
assignUrl="http://.../productassignment.xml">
  <ml:Sequenceid="Seq1">
```

```

<ml:AtomicMethodServiceid="C1"
name="Sélection des besoins de l'itération"
url="http://.../IterationRequirementsSelection.wsdl"/>
<ml:RepeatUntilid="D4">
  <ml:DecisionPointid="D1"min="1" max="1">
    <ml:AtomicMethodServiceid="C2"
name="Découpage des besoins en tâches orienté produit"
url="http://.../ProductOrientedRequirementRefinement.wsdl"/>
    <ml:AtomicMethodServiceid="C3"
name="Découpage des besoins en tâches orienté but"
url="http://.../GoalOrientedRequirementRefinement.wsdl"/>
  </ml:DecisionPoint>
  <ml:AtomicMethodServiceid="C4"
name="Planification de l'itération"
url="http://.../IterationPlanning.wsdl"/>
  <ml:DecisionPointid="D2" min="1" max="3">
    <ml:AtomicMethodServiceid="C5"
name="Traçabilité des tâches de conception"
url="http://.../DesignTasksTracabiliby.wsdl"/>
    <ml:AtomicMethodServiceid="C6"
name="Traçabilité des tâches de réalisation"
url="http://.../DevelopmentTasksTracability.wsdl"/>
    <ml:AtomicMethodServiceid="C7"
name="Traçabilité des tâches de test"
url="http://.../TestTasksTracability.wsdl"/>
  </ml:DecisionPoint>
  <ml:DecisionPointid="D3" min="1" max="1">
    <ml:AtomicMethodServiceid="C8"
name="Rétrospective par stand up meeting"
url="http://.../StandUpMeeting.wsdl"/>
    <ml:AtomicMethodServiceid="C9"
name="Rétrospective par daily meeting"
url="http://.../DailyMeeting.wsdl"/>
  </ml:DecisionPoint>
</ml:RepeatUntil>
</ml:Sequence>
</ml:MethodLine>

```

Figure 5.43 : Exemple d'arbre d'exécution d'une ligne de méthode

De la même manière que la séquence reliant le SMA C4 et le point de variation D2, il suffit d'ajouter un élément *DecisionPoint* D2 en tant que troisième fils du nœud *RepeatUntil* D4. Le point de variation étant de type OU et portant sur trois éléments de type SMA (C5, C6 et C7), ses cardinalités sont de 1 pour le min et de 3 pour le max. Les feuilles *AtomicMethodService* C5, C6 et C7 sont ensuite ajoutées respectivement en tant que fils du nœud D2. Cela constitue la cinquième transformation effectuée pour produire l'arbre d'exécution de la ligne.

La sixième et dernière transformation est initiée par la séquence reliant le point de variation D3 avec le reste des éléments précédents de l'itération D4. De la même manière que pour les deux séquences précédentes, le nœud *DecisionPoint* représentant D3 est directement ajouté en tant que quatrième fils du nœud D4. L'élément D3 représentant un choix de type XOR entre les deux SMA C8 et C9, les

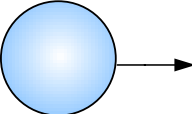
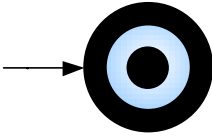
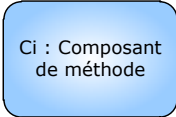
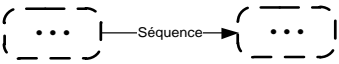
cardinalités min et max du nœud *DecisionPoint* D3 sont par conséquent de valeur 1. De plus, le nœud D3 a donc pour fils les feuilles *AtomicMethodService* représentant les SMA C8 et C9.

Il résulte de ces six transformations le modèle de la ligne de méthode au format XML (voir figure 5.43). Ce modèle opérationnel représente l'arbre d'exécution de la ligne du déroulement d'une itération dans un cycle de vie agile.

Nous venons de décrire la transformation BPMN en un modèle opérationnel sous la forme d'un arbre d'exécution du processus de la ligne de méthode. Afin de faciliter les décisions des utilisateurs dans la configuration de son processus et de son exécution, nous présentons à la section suivante la démarche de spécification des guides de configuration d'une ligne de méthode.

5.3.3.4. Récapitulatif des éléments de modélisation du processus d'une ligne de méthode

Le tableau suivant présente un récapitulatif global des éléments de modélisation du processus d'une ligne de méthode présentés aux sections 5.3.3.1 et 5.3.3.3 de ce chapitre.

| Élément | Modèle d'acquisition | Arbre d'exécution | Modèle opérationnel XML |
|--------------------|--|---|---|
| Début du processus |  Commencer | (Ligne de méthode) MethodLine - name : Ligne de méthode version : x configurationURL : http://.../ guidelines.xml assignURL : http://.../ productassignment.xml | <MethodLine name= "Ligne de méthode" version="x" configurationURL="http://.../guidelines.xml" assignURL="http://.../productassignment.xml" > ... </MethodLine> |
| Fin du processus |  Arrêter | Non représenté | Non représenté |
| SMA |  Ci : Composant de méthode | (Service méthodologique atomique) AtomicMethodService - id : Ci name : Composant de méthode url = http://.../ams.wsdl | <AtomicMethodService id="Ci" name="Composant de méthode" url="http://.../ams.wsdl" > </> |
| Séquence |  | (Agrégation - Séquence) Sequence - id : Seqi a pour fils ... a pour fils ... | <Sequence id= "Seqi"> ... </Sequence> |

| | | | |
|---|--|--|---|
| Agrégation parallèle | | <p>(Agrégation - Parallèle)</p> <pre> Parallel id : Pi a pour fils a pour fils ... </pre> | <pre> <Parallel id="Pi"> ... </Parallel> </pre> |
| Agrégation itérative | | <p>(Agrégation - Itération)</p> <pre> RepeatUntil id : Di a pour fils ... </pre> | <pre> <RepeatUntil id="Di"> ... </RepeatUntil> </pre> |
| Variabilité alternative de niveau processus | | <p>(Point de variation)</p> <pre> DecisionPoint id : Di min : x max : y lien de choix lien de choix ... </pre> | <pre> <DecisionPoint id="Di" min=x max=y> ... </DecisionPoint> </pre> |
| Variabilité alternative de niveau produit | | <p>(Point de variation)</p> <pre> DecisionPoint id : Di min : x max : y lien de choix lien de choix ... </pre> | <pre> <DecisionPoint id="Di" min=x max=y> ... </DecisionPoint> </pre> |
| Variabilité optionnelle | | <p>(Point de variation)</p> <pre> DecisionPoint id : Di min : x max : y lien de choix lien de choix ... </pre> | <pre> <DecisionPoint id="Di" min=x max=y> ... </DecisionPoint> </pre> |

Table 5.1 : Tableau récapitulatif des éléments de modélisation du processus d'une ligne de méthode

Le tableau présente l'ensemble des éléments de modélisation pour la représentation d'un processus d'une ligne de méthode pour le modèle d'acquisition et l'arbre d'exécution d'une ligne (voir table 5.1). Il présente également les éléments XML du modèle opérationnel de l'arbre d'exécution d'une ligne de méthode.

5.3.4 Spécification des guides de configuration

5.3.4.1. Une spécification de la configuration d'une ligne de méthode en quatre étapes

Cette étape permet d'identifier et de spécifier les guides de configuration des points de variation de la ligne de méthode sous la forme d'heuristiques afin de guider les choix de l'utilisateur dans l'exécution du processus de la ligne.

- La première étape de cette démarche est d'identifier les facteurs de contingence affectés aux liens de choix d'un point de variation.
- La deuxième étape de cette démarche de configuration est de déterminer pour chacun de ces liens, les valeurs d'affectation de leurs facteurs de contingence.
- La troisième étape consiste à identifier pour le point de variation un ensemble d'heuristiques ou de bonnes pratiques, appelées guides de configuration, à partir de l'ensemble de ses liens de choix et de leurs facteurs de contingence avec leurs valeurs associées. Il est également possible d'identifier des contraintes concernant les choix ou l'exécution du processus et de les affecter sous la forme de guides de configuration à un point de variation.
- Enfin, la quatrième étape est de représenter les facteurs de contingence et les guides de configuration en un document de configuration de la ligne de méthode au format XML.

Identification des facteurs de contingence

Les facteurs de contingence permettent de caractériser la situation de choix d'un point de variation. Ils correspondent aux facteurs discriminant les différentes situations d'application des alternatives d'un point de variation. Utilisé dans le domaine de l'ingénierie des méthodes, les facteurs de contingence servent à caractériser la situation de réutilisation méthodologique liée au point de variation. Dans ce cadre d'utilisation, l'identification des facteurs de contingence d'une ligne de méthode est réalisée de manière empirique par un ingénieur méthode lors de sa phase d'acquisition. Dans l'objectif d'aider à la découverte des facteurs de contingence, il est recommandé dans un premier temps de comparer les situations d'utilisation alternatives au travers des cinq aspects méthodologiques de la définition d'une méthode (Seligmann, 1989), c'est-à-dire les aspects : paradigme, produit, processus, organisationnel et outil. Pour cela, un ingénieur méthode peut utiliser l'ontologie OMCM que nous avons proposée dans le chapitre 4 de cette thèse (voir figure 4.1). Dans un deuxième temps, il est nécessaire d'axer la comparaison sur les situations de réutilisation des alternatives d'un point de variation, c'est-à-dire sur les facteurs externes à la méthodologie influençant le choix. Ces facteurs peuvent par exemple être liés au contexte du projet ou encore au métier de l'entreprise.

Attribution des valeurs d'affectation des facteurs de contingence

Une fois les facteurs de contingence affectés aux liens de choix reliant un point de variation à ses alternatives, une valeur doit être affectée pour chaque facteur de contingence en fonction de l'alternative et de l'aspect qu'il caractérise. Ces valeurs sont déterminées par l'ingénieur méthode de manière empirique. L'ensemble des facteurs de contingence et des valeurs d'affectation d'un lien de choix permettent de décrire sa situation de choix. Cela permet de le différencier des autres liens du même point de variation et ainsi, aider les utilisateurs d'une ligne de méthode à prendre leurs décisions.

Identification des guides de configuration

Après avoir identifié l'ensemble des facteurs de contingence et leurs valeurs d'affectation, un ingénieur méthode peut ensuite passer à l'étape de rédaction des guides de configuration de la ligne de méthode. Un guide de configuration est une heuristique, une bonne pratique, une condition ou une contrainte sous la forme d'un texte de quelques phrases avec l'objectif de guider un utilisateur dans sa prise de décision entre les différentes alternatives d'un point de variation. Ces guides peuvent être déterminés de plusieurs manières pour un point de variation :

- soit à partir d'une analyse des alternatives en comparant les valeurs affectées aux facteurs de contingence,
- soit de manière empirique par un ingénieur méthode,
- soit être l'expression d'une condition sur un produit ou d'un point d'agrégation itératif,
- ou encore en identifiant les contraintes d'exécution entre les points de variation et alternatives du processus de la ligne de méthode.

Une fois le processus de configuration terminé, les guides de configuration ainsi que les facteurs de contingence avec leurs valeurs d'affectations peuvent être reportés sur le modèle d'acquisition de la ligne de méthode sous la forme de notes et de commentaires. Pour des raisons de lisibilité de schéma du modèle d'acquisition de la ligne de méthode, il est également possible de les regrouper dans un document annexe.

Spécification du document de configuration

Cette étape nécessite d'avoir au préalable réalisé l'arbre d'exécution et identifié les guides de configuration de la ligne de méthode. Il est ensuite possible d'établir le document de configuration de la ligne au format XML. Pour chaque point de variation de l'arbre de décision, l'ensemble de ses guides de configuration ainsi que les facteurs de contingence de ses variantes sont représentés dans le document. Le détail de la réalisation de cette étape est présenté à la section 5.3.4.3.

5.3.4.2. Exemple de configuration d'une ligne de méthode

Nous proposons, dans cet exemple, d'identifier les facteurs de contingence de la ligne de méthode GIDPA présentée par son modèle d'acquisition BPMN dans ce chapitre à la figure 5.34.

Le premier point de variation rencontré dans l'exécution du processus de la ligne est le point de variation D1 de type XOR. Il propose un choix entre deux alternatives D1.1 et D1.2, conduisant respectivement à l'exécution des SMA C2 et C3. Ces SMA proposent tous deux une méthodologie pour affiner des besoins. Leur principale différence se situe au niveau de la démarche qu'ils proposent pour décomposer un besoin. Le SMA C2 est issu de la méthode XP et propose une décomposition des besoins en fonction du produit à réaliser. Le SMA C3 issu de la méthode Scrum propose, quant à lui,

une décomposition sous la forme de buts et de sous-but à atteindre dans l'itération courante. Nous pouvons, par conséquent, définir un facteur de contingence nommé "processus de décomposition" pour les liens de choix D1.1 et D1.2 avec pour affectation les valeurs respectives de "orienté produit" et "orienté but". Au niveau des guides de configuration il est possible d'identifier un guide G1 comme suit : Il est recommandé d'exécuter le composant C2 si "le client et l'équipe projet ont des facilités à exprimer les besoins en termes de produit à réaliser" et d'exécuter le composant C3 si "ils ont des facilités à exprimer les besoins en termes de buts à atteindre pour cette itération".

Le point de variation suivant rencontré dans l'exécution de la ligne est le point D2 de type OR. Il propose un choix optionnel entre trois SMA (C5, C6 et C7) permettant de tracer l'exécution de trois types de tâches : conception, réalisation et test. Le choix entre ces alternatives est réalisé en fonction du type des tâches à exécuter dans le planning. Un facteur de contingence pour les liens de choix D2.1, D2.2 et D2.3 peut par exemple être le "type de tâche à exécuter" et avoir pour valeur d'affectation pour les liens de choix respectivement "conception", "réalisation" et "test". Par conséquent, le guide de configuration de ce point de variation est : "Il est nécessaire d'utiliser pour chaque tâche du planning en cours le SMA de traçabilité correspondant au type de la tâche exécutée".

Le troisième point de variation de la ligne est D3. Il permet de choisir entre deux manières d'effectuer une réunion de rétrospective journalière dans une itération réalisée par les SMA C8 et C9. Le SMA C8 propose une réunion informelle et est destiné à des équipes de taille réduite. Le SMA propose, quant à lui, un type de réunion plus formel pour des équipes plus importantes. Une autre différence concerne les métriques de mesure de productivité utilisées. Les deux SMA utilisent la métrique de vélocité du projet mais le SMA C9 propose d'utiliser également la métrique de burndown chart afin d'établir une trace des besoins implémentés depuis le début du projet afin de visualiser la productivité de l'équipe sur l'ensemble du projet. Nous pouvons donc identifier trois facteurs de contingence pour ce point de variation : "niveau de formalisme" (aspect processus), "métriques utilisées" (aspects processus et produit) et "taille de l'équipe" (aspect de réutilisation). La branche D3.1 reliée au SMA C8 a pour valeurs d'affectation respectivement "faible", "vélocité du projet" et "petite" et la branche D3.2 reliée au SMA C9 a pour valeurs d'affectation respectivement "élevé", "vélocité du projet, burndown chart" et "grande". Nous pouvons ensuite identifier deux guides de configuration à partir de ces facteurs : "Il est recommandé aux petites équipes préférant effectuer des réunions journalières informelles de rétrospective d'appliquer le SMA C8", "Il est recommandé aux équipes conséquentes souhaitant avoir des réunions journalières de rétrospective plus formelles et organisées avec une vision de l'itération élargie à l'ensemble du projet d'appliquer le SMA C9".

La dernière étape du processus de configuration de cette ligne concerne le point d'agrégation itératif D4. Sa condition de continuation de boucle doit être saisie sous la forme du guide de configuration

suivant : "A l'issue de la journée si l'itération en cours n'est pas arrivée à son terme, prévoir une journée supplémentaire pour effectuer l'itération".

5.3.4.3. *Modèle de configuration d'une ligne de méthode*

Après avoir effectué l'acquisition des facteurs de contingence, de leurs valeurs d'affectation et des guides de configuration d'une ligne de méthode, il est nécessaire de produire le modèle de configuration de la ligne. C'est un modèle opérationnel de guide de configuration attaché au modèle de la ligne de méthode. Ce modèle propose pour chaque point de variation ou point d'agrégation itératif de la ligne de méthode, ses guides de configuration, facteurs de contingence et affectations au format XML.

Voici l'ensemble des règles de transformation permettant d'obtenir le modèle de configuration d'une ligne à partir des résultats de son processus de configuration :

```
<ConfigurationModel name="...">
    ...
</ConfigurationModel>
```

Figure 5.44 : Élément racine du document XML du modèle de configuration

L'élément racine du document XML représentant le modèle de configuration est nommé *ConfigurationModel* et possède un attribut "name" (voir figure 5.44).

Pour chacun des points de variation et points d'agrégation itératif un élément *Decision* est ajouté comme fils du nœud racine *ConfigurationModel* du document (voir figure 5.45).

```
<Decision targetId="...">
    <ConfigurationGuidelineList>
        ...
    </ConfigurationGuidelineList>
    <ContingencyFactorList>
        ...
    </ContingencyFactorList>
</Decision>
```

Figure 5.45 : Élément "Decision" du modèle de configuration d'une ligne de méthode

Comme présenté à la figure 5.45, chaque élément *Decision* a pour attribut "targetId" étant une chaîne de caractères faisant référence à l'id de l'élément représentant le point de variation ou le point d'agrégation itératif dans l'arbre d'exécution de la ligne de méthode. Ce nœud *Decision* possède deux éléments fils : un élément *ConfigurationGuidelineList* regroupant la liste des guides de configuration du point et un élément *ContingencyFactorList* regroupant la liste des facteurs de contingence des liens de choix du point ainsi que leurs valeurs d'affectation.

Les guides de configuration du point de variation ou du point d'agrégation itératif sont ensuite ajoutés comme fils de l'élément *ConfigurationGuidelineList* de l'élément *Decision* le représentant.

Ces guides sont représentés par l'élément *ConfigurationGuideline* du modèle de configuration (voir figure 5.46) qui est également pourvu d'un attribut "name" permettant de l'identifier.

```
<ConfigurationGuideline name="...">
    ...
</ConfigurationGuideline>
```

Figure 5.46 : Élément "ConfigurationGuideLine" du modèle de configuration d'une ligne de méthode

Le guide en lui-même est saisi de manière textuelle entre les balises de début et de fin de l'élément *ConfigurationGuideline*.

Les facteurs de contingence sont décrits dans le modèle de configuration par l'élément *ContingencyFactor*(voir la figure ci-dessous).

```
<ContingencyFactor name="..." targetId="...">
    ...
</ContingencyFactor>
```

Figure 5.47 : Élément "Decision" du modèle de configuration d'une ligne de méthode

Les facteurs de contingence sont exprimés dans le modèle de configuration par l'élément *ContingencyFactor*. Cet élément possède un attribut "name" représentant son nom et un attribut "targetId" représentant l'identifiant du nœud ou de la feuille dans l'arbre d'exécution ciblé par ce facteur de contingence pour le point de variation ou du point d'agrégation itératif donné. La valeur d'affectation pour ce facteur de contingence est saisie entre les balises de début et de fin de cet élément (voir figure 5.47).

Le schéma XSD de validation du modèle de configuration au format XML est disponible dans l'annexe E.

5.3.4.4. Exemple de modèle de configuration d'une ligne de méthode

Nous utilisons pour cet exemple les résultats du processus de configuration (voir section 5.3.4.2 de ce chapitre) de la ligne de méthode GIDPA (voir figure 5.34). L'ensemble des facteurs de contingence avec leurs valeurs dans cet exemple est transformé en éléments XML. Il en est de même pour les guides de configuration identifiés pour cette ligne de méthode. Il résulte de l'ensemble de ces transformations le modèle de configuration de la ligne de méthode du déroulement d'une itération dans un cycle de vie agile au format XML tel que présenté à la figure 5.48.

```
<?xml version="1.0" encoding="UTF-8"?>
<cm:ConfigurationModel xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:cm='http://crinfo.univ-parisl.fr/schema/ConfigurationModel'
xsi:schemaLocation='http://crinfo.univ-parisl.fr/schema/ConfigurationModel
ConfigurationModel.xsd' name="Modèle de configuration de la ligne de méthode de
déroulement d'une itération dans un cycle de vie agile">
<cm:Decision targetId="D1">
<cm:ConfigurationGuidelineList>
```

```

<cm:ConfigurationGuideline name="G1">Il est recommandé d'exécuter le composant C2
si le client et l'équipe projet ont des facilitées à exprimer les besoins en termes
de produit à réaliser et d'exécuter le composant C3 si ils ont des facilitées à
exprimer les besoins en termes de buts à atteindre pour cette
itération</cm:ConfigurationGuideline>
</cm:ConfigurationGuidelineList>
<cm:ContingencyFactorList>
<cm:ContingencyFactor name="processus de décomposition" targetId="C2">orienté
produit</cm:ContingencyFactor>
<cm:ContingencyFactor name="processus de décomposition" targetId="C3">orienté
but</cm:ContingencyFactor>
</cm:ContingencyFactorList>
</cm:Decision>
<cm:Decision targetId="D2">
<cm:ConfigurationGuidelineList>
<cm:ConfigurationGuideline name="G2">Il est nécessaire d'utiliser pour chaque tâche
du planning en cours le SMA de traçabilité correspondant au type de la tâche
exécutée</cm:ConfigurationGuideline>
</cm:ConfigurationGuidelineList>
<cm:ContingencyFactorList>
<cm:ContingencyFactor name="type de tâche à exécuter"
targetId="C5">conception</cm:ContingencyFactor>
<cm:ContingencyFactor name="type de tâche à exécuter"
targetId="C6">réalisation</cm:ContingencyFactor>
<cm:ContingencyFactor name="type de tâche à exécuter"
targetId="C7">test</cm:ContingencyFactor>
</cm:ContingencyFactorList>
</cm:Decision>
<cm:Decision targetId="D3">
<cm:ConfigurationGuidelineList>
<cm:ConfigurationGuideline name="G3">Il est recommandé aux petites équipes
préférant effectuer des réunions journalières informelles de rétrospective
d'appliquer le SMA C8</cm:ConfigurationGuideline>
<cm:ConfigurationGuideline name="G4">Il est recommandé aux équipes conséquentes
souhaitant avoir des réunions journalières de rétrospective plus formelle et
organisée avec une vision de l'itération élargie l'ensemble du projet d'appliquer
le SMA C9</cm:ConfigurationGuideline>
</cm:ConfigurationGuidelineList>
<cm:ContingencyFactorList>
<cm:ContingencyFactor name="niveau de formalisme"
targetId="C8">faible</cm:ContingencyFactor>
<cm:ContingencyFactor name="niveau de formalisme"
targetId="C9">élevé</cm:ContingencyFactor>
<cm:ContingencyFactor name="métriques utilisées" targetId="C8">vélocité du
projet</cm:ContingencyFactor>
<cm:ContingencyFactor name="métriques utilisées" targetId="C9">vélocité du projet,
burndown chart </cm:ContingencyFactor>
<cm:ContingencyFactor name="taille de l'équipe"
targetId="C8">petite</cm:ContingencyFactor>
<cm:ContingencyFactor name="taille de l'équipe"
targetId="C9">grande</cm:ContingencyFactor>
</cm:ContingencyFactorList>
</cm:Decision>
<cm:Decision targetId="D4">
<cm:ConfigurationGuidelineList>
<cm:ConfigurationGuideline name="G5">A l'issue de la journée effectuée si
l'itération en cours n'est pas arrivée à son terme, effectuez une journée
supplémentaire de l'itération</cm:ConfigurationGuideline>
</cm:ConfigurationGuidelineList>
<cm:ContingencyFactorList/>
</cm:Decision>
</cm:ConfigurationModel>

```

Figure 5.48 : Exemple de modèle de configuration de la ligne de méthode GIDPA

Les points de décisions identifiés à la section 5.3.4.2 sont écrits en couleur bleue à la figure 5.48, alors que les guides méthodologiques sont en couleur orange et les facteurs de contingence avec leurs valeurs de couleur verte.

5.3.5 Spécification des aspects produits de la ligne de méthode

La partie produit du métamodèle de ligne de méthode est composée d'une part du modèle de produit global à la ligne et d'autre part de la spécification des transformations de produits de type entrée et de type sortie des SMA inclus dans la ligne.

La figure suivante présente la partie produit du métamodèle d'une ligne méthode.

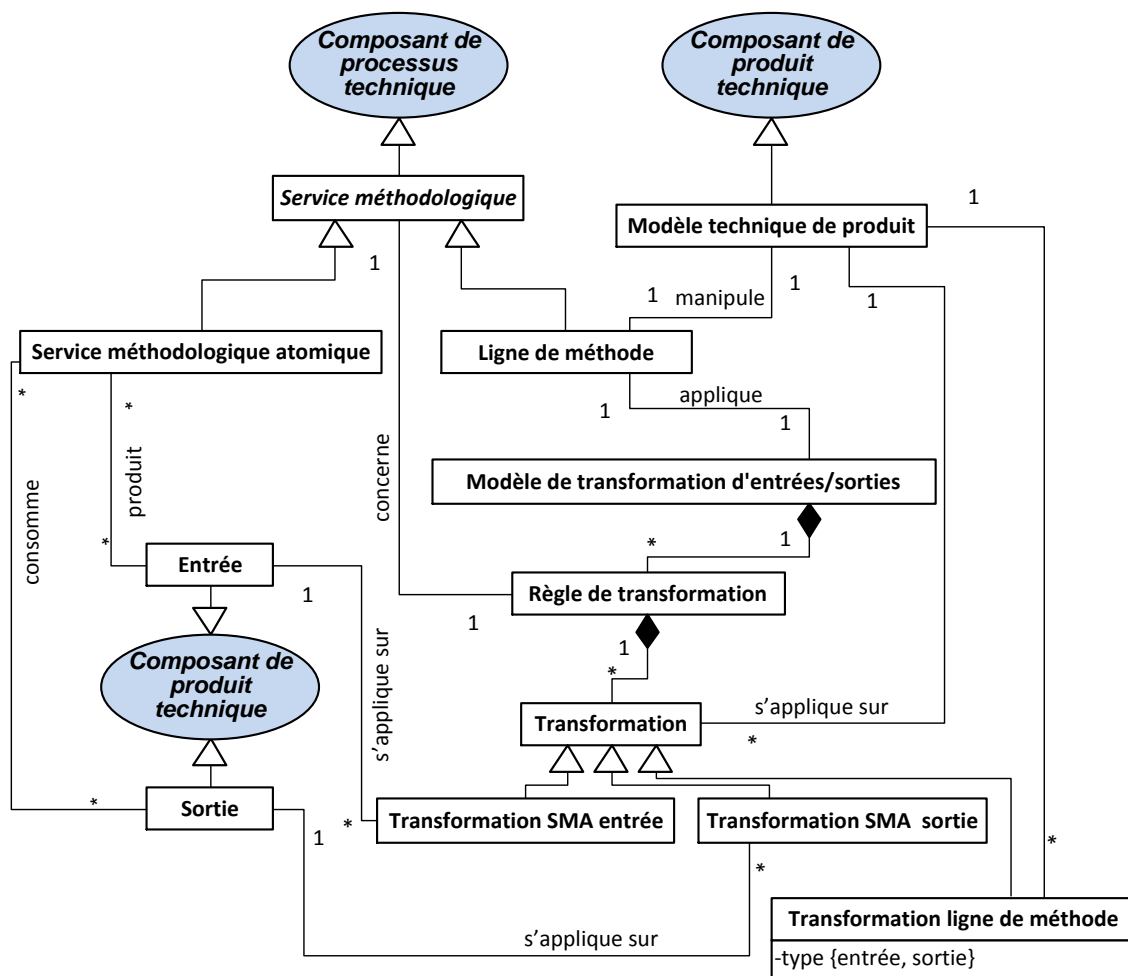


Figure 5.49 : Métamodèle de ligne de méthode : partie produit

La description opérationnelle du produit global d'une ligne de méthode est représentée par le *Modèle technique de produit* de la ligne. Ce métamodèle agrège l'ensemble des modèles de produit des SMA en une description globale du produit manipulé par la ligne (voir figure.5.49).

Le *Modèle technique de produit* de la ligne est ensuite complété par le *Modèle de transformation d'entrées/sorties* qui comprend toutes les transformations à effectuer sur les produits en entrée et en

sortie des SMA pour les faire correspondre au modèle de produit global de la ligne. Ce modèle de transformation est constitué d'un ensemble de *Règles de transformation* portant chacune sur un SMA. Une règle étant elle-même constituée de *Transformations* correspondant chacune à une transformation à effectuer entre le produit global et une *Entrée* du SMA (*Transformation SMA entrée*) ou une transformation à effectuer entre un produit en *Sortie* du SMA pour l'intégrer dans le produit global de la ligne (*Transformation SMA sortie*). Il est à noter qu'une ligne de méthode est un service méthodologique qui doit comporter de la même manière qu'un SMA un produit en entrée et un produit en sortie. Ces deux produits sont inclus dans le *Modèle technique de produit* de la ligne et sont soumis aussi à une règle de transformation (*Transformation ligne de méthode*) pour intégrer le produit en entrée de la ligne dans le produit global et dériver le produit de sortie à partir du produit global.

La réalisation du *Modèle de transformation d'entrées/sorties* se déroule en trois étapes. La première étape consiste à définir le modèle conceptuel de produit global de la ligne de méthode. Ensuite, la deuxième étape permet de réaliser le *Modèle technique de produit* de la ligne à partir de ce modèle conceptuel. Enfin, la troisième étape consiste à définir le *Modèle de transformation d'entrées/sorties* à partir de *Modèle technique de produit* de la ligne et des *Technical Product Puzzle* de ses SMA.

5.3.5.1. Un modèle de produit global à la ligne de méthode

La première étape consiste à spécifier le modèle de produit global de la ligne de méthode par une agrégation de l'aspect produit pris en compte par chaque SMA de la ligne. Une fois établi, ce modèle de produit fournit une vue d'ensemble du produit de la ligne de méthode et de ses entrées/sorties. Il permet également d'identifier la partie du produit global de la ligne manipulé par chaque SMA et ainsi de caractériser leurs entrées et leurs sorties par rapport à ce modèle global. Le fait de caractériser ainsi les entrées/sorties de chaque SMA en fonction du modèle de produit de la ligne de méthode diminue la complexité du modèle de transformation. En effet, l'identification des règles de transformation est réalisée pour chaque SMA et concerne les transformations de produit du modèle de la ligne vers le format de ses entrées qui lui sont spécifiques et, inversement, les transformations de ses sorties spécifiques vers le modèle de produit global. Cela simplifie le modèle de transformation car seul un groupe de transformations est défini par SMA vers le modèle global de produit et permet d'éviter une explosion combinatoire liée à l'exploration de chaque SMA de ses SMA suivants et la définition d'un groupe de transformations pour chaque lien possible identifié.

Le formalisme et le langage de modélisation utilisés pour effectuer l'acquisition du modèle de produit conceptuel d'une ligne de méthode sont laissés au choix de l'ingénieur méthode. L'acquisition de ce modèle étant fortement lié aux différents paradigmes entrant en jeu dans la ligne de méthode, la représentation du modèle est réalisée selon le choix de l'ingénieur méthode dans l'un des langages de modélisation existant pour ces paradigmes.

Une fois l'acquisition du modèle de produit de la ligne de méthode effectué, nous proposons de l'implémenter sous la forme d'un schéma XML (XSD) permettant de définir la structure d'une implémentation du produit de la ligne sous la forme d'un document XML. En d'autres termes, les instances des concepts du modèle de produit sont intégrées à un document XML. Ce document XML est lui-même conforme à la définition de sa structure dans un schéma XSD représentant le modèle de produit.

Les règles de transformation du modèle de produit global de la ligne de méthode en schéma XSD vont varier en fonction du paradigme et du langage de modélisation utilisé pour modéliser le produit. Il existe cependant dans le domaine de l'ingénierie dirigée par les modèles de nombreux outils et approches de transformation de modèles vers le format de représentation XSD. Par exemple, pour le langage UML, (Routledge, 2002) proposent un ensemble de ces règles. Des outils comme (IBM, 2012) existent et proposent une automatisation de ce type de transformation. Il en va de même pour le langage de modélisation E/R (Pin-shan Chen, 1976) avec l'approche de transformation proposée par (Liu, 2006) afin de produire un schéma XSD implémentant un modèle E/R.

5.3.5.2. Exemple de modèle de produit d'une ligne de méthode

Dans cette section, nous développons l'exemple du modèle de produit global de la ligne de méthode du déroulement d'une itération dans un cycle de vie agile dont nous avons présenté le processus à la figure 5.34 et les modèles d'exécution et de configuration respectivement sur les figures 5.43 et 5.48. Conformément à la démarche proposée dans la section 5.3.5.1, les modèles de produit de l'ensemble des SMA intervenants dans la ligne de méthode doivent être agrégés en un modèle de produit global pour cette ligne. La figure 5.50 présente ce modèle de produit modélisé avec le langage UML. Nous décrivons uniquement les variables instances et les liens entre les classes.

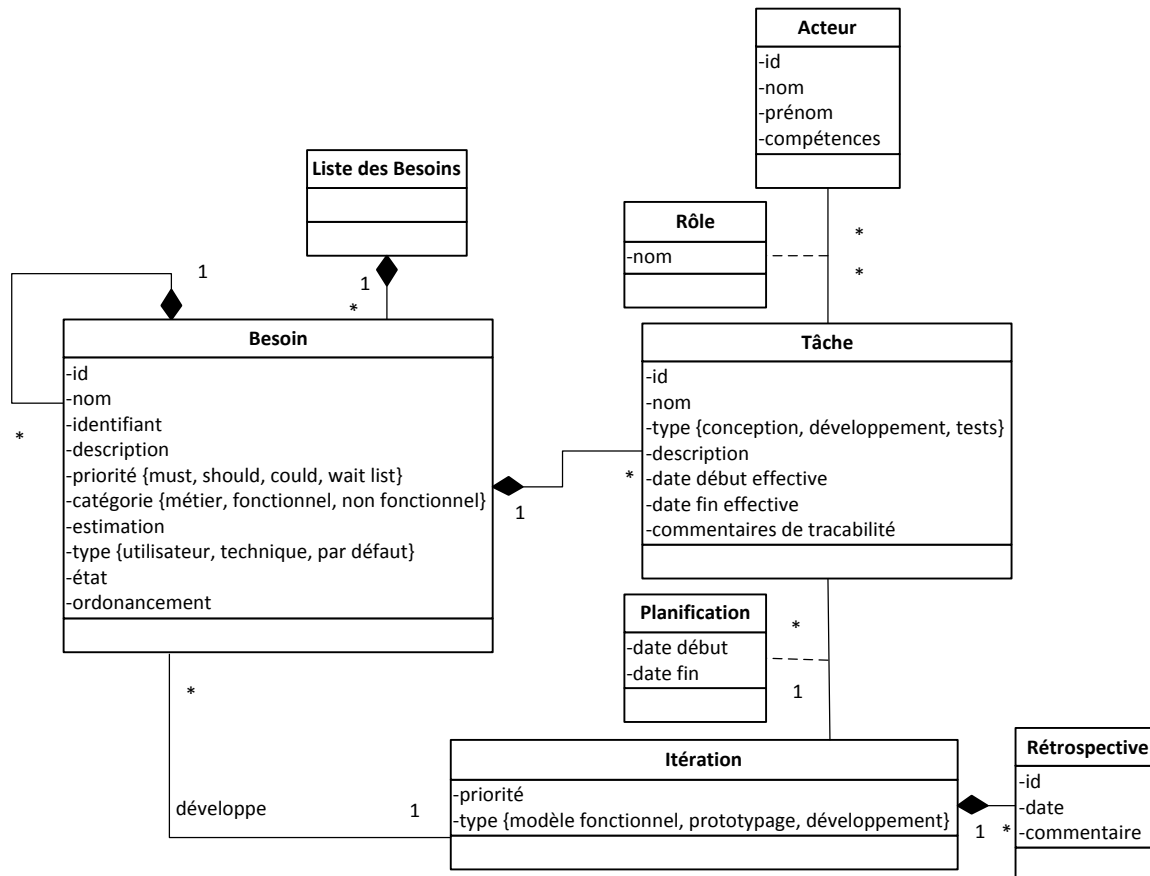


Figure 5.50 : Modèle de produit de la ligne du de méthode GIDPA

La ligne de méthode GIDPA prend en entrée la liste des besoins restant à implémenter dans le projet. Cela est représenté dans le modèle de produit de la ligne (voir figure 5.50) par les classes *Liste des Besoins* et *Besoin*. Ces deux classes sont reliées par un lien de composition représentant le fait que la classe *Liste des Besoins* regroupe un ensemble de *Besoin*.

Le premier SMA de la ligne propose d'effectuer une sélection d'un sous-ensemble parmi les éléments de la liste des besoins pour les affecter à l'itération courante et les développer dans un incrément logiciel. Cela se traduit dans le modèle de la ligne par la classe *Itération* et une association entre celle-ci et la classe *Besoin* pour décrire la sélection de besoins à développer dans l'itération.

Ensuite, la ligne de méthode préconise d'affiner ces besoins jusqu'à obtenir des besoins de granularité assez fine pour que leur développement soit décomposable en une série de tâches. Ces aspects sont modélisés dans le modèle de produit de la ligne respectivement par un premier lien de composition réflexif de la classe *Besoin* sur elle-même pour tracer le fait qu'un besoin peut être affiné puis par une classe *Tâche* avec un second lien de composition entre cette dernière et la classe *Besoin* pour représenter la décomposition du processus de développement d'un besoin en une série de tâches.

Dans la continuité du processus de la ligne de méthode, un planning de réalisation est établi pour l'ensemble des tâches à effectuer dans l'itération. Nous retrouvons cela sur le modèle de produit de la ligne sous la forme de la classe association *Planification* entre les classes *Itération* et *Tâche*. Puis, pour chaque tâche de l'itération un ou plusieurs acteurs leur sont affectés et cela se traduit dans le modèle par la classe *Acteur* et la classe association *Rôle* reliant les classes *Acteur* et *Tâche*. La traçabilité de l'exécution des tâches est, quant à elle, réalisée par les attributs de la classe *Tâche* suivants : *date début effective*, *date fin effective* et *commentaires de traçabilité*.

A l'issue de chaque journée de l'itération, une rétrospective est réalisée par l'équipe du projet sur l'état des tâches réalisées et celles en cours. Cela est représenté dans le modèle de produit de la ligne par la classe *Rétrospective* et le lien de composition la reliant à la classe *Itération*.

Après avoir fait l'acquisition du modèle global de la ligne de méthode, celui-ci doit être transformé en modèle XSD qui, correspond au niveau technique et opérationnel de ce modèle. Pour cet exemple, le schéma XSD présenté à la figure 5.51 est le résultat de la transformation en XSD du modèle de produit de la ligne modélisé en UML présenté à la figure 5.50.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://crinfo.univ-parisl.fr/schema/ML-PM-Iteration"
  xmlns:tns="http://crinfo.univ-parisl.fr/schema/ML-PM-Iteration"
  elementFormDefault="qualified"
  >

  <xsd:element name="ModeleDeProduit">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:ListeDesBesoins" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="tns:Iteration" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="tns:Acteur" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="tns:Role" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="ListeDesBesoins">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:Besoin" minOccurs="1" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Besoin" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:Besoin" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="tns:Tache" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="id" use="required" type="xsd:string" />
      <xsd:attribute name="nom" use="required" type="xsd:string" />
      <xsd:attribute name="etat" use="required" type="xsd:string" />
      <xsd:attribute name="description" use="optional" type="xsd:string" />
      <xsd:attribute name="priorite" use="optional" type="xsd:string" />
      <xsd:attribute name="categorie" use="optional" type="xsd:string" />
      <xsd:attribute name="type" use="optional" type="xsd:string" />
      <xsd:attribute name="estimation" use="optional" type="xsd:string" />
      <xsd:attribute name="ordonancement" use="optional" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Tache">
```

```

<xsd:complexType>
<xsd:attribute name="id" use="required" type="xsd:string" />
<xsd:attribute name="nom" use="required" type="xsd:string" />
<xsd:attribute name="dateDebutEffective" use="optional" type="xsd:dateTime" />
<xsd:attribute name="dateFinEffective" use="optional" type="xsd:dateTime" />
<xsd:attribute name="type" use="optional" type="xsd:string" />
<xsd:attribute name="description" use="optional" type="xsd:string" />
<xsd:attribute name="commentairesDeTracabilite" use="optional" type="xsd:string" />
</xsd:complexType>
</xsd:element>

<xsd:element name="Iteration">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="tns:Developpe" minOccurs="0" maxOccurs="unbounded" />
<xsd:element ref="tns:Planification" minOccurs="0" maxOccurs="unbounded" />
<xsd:element ref="tns:Retrospective" minOccurs="0" maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="type" use="optional" type="xsd:string" />
<xsd:attribute name="priorite" use="optional" type="xsd:string" />
</xsd:complexType>
</xsd:element>

<xsd:element name="Developpe">
<xsd:complexType>
<xsd:attribute name="idBesoin" use="required" type="xsd:string" />
</xsd:complexType>
</xsd:element>

<xsd:element name="Planification">
<xsd:complexType>
<xsd:attribute name="idTache" use="required" type="xsd:string" />
<xsd:attribute name="dateDebut" use="required" type="xsd:dateTime" />
<xsd:attribute name="dateFin" use="required" type="xsd:dateTime" />
</xsd:complexType>
</xsd:element>

<xsd:element name="Retrospective">
<xsd:complexType>
<xsd:attribute name="id" use="required" type="xsd:string" />
<xsd:attribute name="date" use="required" type="xsd:dateTime" />
<xsd:attribute name="commentaire" use="required" type="xsd:string" />
</xsd:complexType>
</xsd:element>

<xsd:element name="Acteur">
<xsd:complexType>
<xsd:attribute name="id" use="required" type="xsd:string" />
<xsd:attribute name="nom" use="required" type="xsd:string" />
<xsd:attribute name="prenom" use="required" type="xsd:string" />
<xsd:attribute name="competences" use="optional" type="xsd:string" />
</xsd:complexType>
</xsd:element>

<xsd:element name="Role">
<xsd:complexType>
<xsd:attribute name="idActeur" use="required" type="xsd:string" />
<xsd:attribute name="idTache" use="required" type="xsd:string" />
<xsd:attribute name="nom" use="required" type="xsd:string" />
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

Figure 5.51 : Modèle de produit de la ligne de méthode au format XSD

Comme présenté à la figure 5.51, les transformations du modèle de produit au format UML en schéma XSD sont simples (les éléments sont marqués en bleu et les références à d'autres éléments en vert). A chaque classe du modèle UML correspond la définition d'un *element* complexe dans le schéma XSD.

Par conséquent pour les classes et leurs attributs *Liste des Besoins*, *Besoin*, *Itération*, *Tâche*, *Acteur* et *Rétrospective* nous trouvons respectivement la définition des éléments suivants : *ListeDesBesoins*,

Besoin, Iteration, Tache, Acteur et Retropective. Pour des raisons d'implémentation, les accents et les espace présents dans les noms des classes ont été retirés et les noms des attributs simplifiés.

Il en va de même pour les classes associations *Rôle* et *Planification* qui sont transformée en éléments *Role* et *Planification*. Comme il ne peut y avoir plus d'une itération dans cette ligne de méthode l'élément *Iteration* contient directement la définition d'une liste d'éléments *Planification* dans sa déclaration et l'élément *Planification* contient lui-même un attribut *idTache* faisant référence à l'attribut d'identification *id* de l'élément *Tache* qu'il planifie. Cela est différent pour l'élément *Role* car plusieurs éléments *Acteur* peuvent être affectés à plusieurs éléments *Tache* et inversement. L'élément *Role* contient par conséquent deux attributs *idActeur* et *idTache* faisant respectivement référence aux attributs d'identification *id* des éléments *Acteur* et *Tache*.

De la même manière, l'association entre les classes *Itération* et *Besoin* du modèle est représentée dans le schéma XSD par un élément *Developpe* ayant une référence vers un élément *Besoin*. La référence entre l'élément *Iteration* et l'élément *Developpe* est, quant à elle, directement contenue dans la définition de l'élément *Iteration* car il ne peut exister qu'une instance de l'élément itération dans cette ligne.

Les liens de composition présents dans le modèle UML sont transformés dans le schéma XSD en liens de séquence (*sequence*) dans la définition de l'élément agrégat faisant référence à l'élément composant. Par conséquent, l'élément *ListeDesBesoins* contient une référence vers l'élément *Besoin*, l'élément *Besoin* possède, quant à lui, des références vers l'élément *Tache* et lui-même, enfin l'élément *Iteration* a une référence vers l'élément *Retrospective*.

Le modèle de produit dans son ensemble est lui-même encapsulé dans l'élément racine *ModeleDeProduit* qui contiendra une instance des éléments *ListeDesBesoins* et *Iteration*, puis des instances d'éléments *Acteur* et *Role*.

Une instance de ce modèle de produit opérationnel en sortie de cette ligne de méthode pourrait être par exemple du type de celui présenté à la figure 5.52.

```
<?xml version="1.0" encoding="UTF-8"?>
<pm:ModeleDeProduit
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:pm='http://crinfo.univ-paris1.fr/schema/ML-PM-Iteration'
xsi:schemaLocation='http://crinfo.univ-paris1.fr/schema/ML-PM-Iteration ML-PM-Iteration.xsd' >
<pm>ListeDesBesoins>
<pm:Besoin id="B1" nom="Rechercher un ancêtre" etat="à développer">
<pm:Tache id="T1" nom="API de manipulation de la BDD généalogique" />
<pm:Tache id="T2" nom="Interface graphique de l'application" />
</pm:Besoin>
</pm>ListeDesBesoins>
<pm:Iteration>
<pm:Developpe idBesoin="B1"/>
<pm:Planification idTache="T1"
dateDebut="2012-04-02T08:00:00" dateFin="2012-04-02T18:00:00"/>
<pm:Planification idTache="T2"
dateDebut="2012-04-03T08:00:00" dateFin="2012-04-03T18:00:00"/>
<pm:Retrospective id="R1" date="2012-04-03T08:00:00"
commentaire="La recherche des ancêtres rencontre des difficultés d'implémentation">
```

```

</pm:Iteration>
<pm:Acteur id="A1" nom="Ward" prenom="Charles Dexter" />
<pm:Role idActeur="A1" idTache="T1" nom="développeur"/>
<pm:Role idActeur="A1" idTache="T2" nom="développeur"/>
</pm:ModeleDeProduit>

```

Figure 5.52 : Exemple d'instance de produit en sortie d'une ligne de méthode après son exécution

La figure 5.52 présente l'exemple de produit en sortie de l'exécution de la ligne de méthode de déroulement d'un projet agile pour une itération d'un projet de développement d'une application de gestion d'arbre généalogique. La fonctionnalité à implémenter dans cette itération a pour but la recherche d'ancêtres et son développement est confié au membre de l'équipe du projet nommé Charles Dexter Ward. Cette fonctionnalité est découpée en deux tâches: la réalisation d'un composant logiciel de recherche et la réalisation d'une interface graphique. Enfin, le rapport de rétrospective indique que le développeur rencontre des difficultés avec l'implémentation du module de recherche.

5.3.5.3. *Modèle de transformation des entrées/sorties*

Le modèle de transformation des entrées/sorties d'une ligne de méthode est basé sur le modèle de produit global de la ligne et des produits manipulés par les SMA composant la ligne. Ce modèle de transformation a pour vocation de rassembler toutes les transformations de produits à effectuer entre le modèle de la ligne et les produits spécifiques en entrée de chaque SMA et inversement entre les produits spécifique en sortie de chaque SMA et le modèle global de la ligne. Ce modèle de transformation utilise donc le modèle de produit de la ligne comme une base produit commune pour tous les SMA de la ligne et fournit les règles de transformation pour effectuer une correspondance entre les produits spécifiques des SMA et le produit global de la ligne.

A l'instar du standard BPEL (OASIS/BPEL, 2007) et de ses activités d'assignation de variables entre les services d'une orchestration, nous proposons de définir ce modèle de transformation comme un schéma XML contenant l'ensemble des transformations de produits à effectuer dans le flux d'exécution d'une ligne de méthode.

L'élément racine de ce modèle XML se nomme *AssignmentModel* et a pour fils un élément *MethodLine* et au minimum un élément *Assign* (voir figure 5.53)

```

<AssignmentModel
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:='http://crinfo.univ-paris1.fr/schema/AssignmentModel'
xsi:schemaLocation='http://crinfo.univ-paris1.fr/schema/AssignmentModel AssignmentModel.xsd
http://www.w3.org/1999/XMLSchema http://www.w3.org/1999/XMLSchema'
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns: ... ="..."
xmlns: ... ="..."
...
>
<am:ProductModels>
  <xsd:schema targetNamespace="...">
    <xsd:import namespace="..." schemaLocation="... .xsd"/>
    <xsd:import namespace="..." schemaLocation="... .xsd"/>
    ...
  </xsd:schema>
</am:ProductModels>

```

```

<MethodLine name="..."> ... </MethodLine>
<Assign parentId="..." leafId="...">... </Assign>
<Assign parentId="..." leafId="...">... </Assign>
...
</AssignmentModel>

```

Figure 5.53 : Élément AssignmentModel

L'élément *AssignmentModel* représente le modèle de transformation de la ligne au complet, c'est pourquoi il est l'élément racine du document XML relatif à ce modèle. A la figure 5.53 nous pouvons retrouver en vert la référence du schéma XSD de définition du modèle de transformation (voir l'annexe F pour la définition complète de ce schéma XSD). Toujours à la figure 5.53, mais représentées cette fois en bleu, viennent les définitions des espaces de nommage et les références de tous les modèles de produit utilisés dans ce document, c'est-à-dire du schéma XSD du modèle de produit global de la ligne de méthode et des schémas XSD des produits en entrée et en sortie des SMA. Ces derniers sont également ceux utilisés dans les descripteurs WSDL des SMA pour décrire leurs produits en entrée et en sortie. L'ensemble des modèles utilisés par les règles de transformation est réuni sous l'élément *ProductModels* afin de réaliser les importations de ceux-ci au niveau technique.

L'élément *AssignmentModel* a obligatoirement un élément fils de type *MethodLine* et cet élément permet de considérer la ligne de méthode comme un ensemble et d'identifier les transformations à effectuer sur ses données en entrée et d'identifier les données produit qu'elle doit restituer en sortie (voir figure 5.54).

```

<MethodLine name="...">
<Input>
...
</Input>
<Output>
...
</Output>
</MethodLine>

```

Figure 5.54 : Élément MethodLine

Comme présenté à la figure 5.54, l'élément *MethodLine* possède un élément Input contenant l'ensemble des transformations à effectuer sur les données d'entrée de la ligne et un élément Output rassemblant l'ensemble des transformations à effectuer sur l'instance du modèle de produit de la ligne de méthode afin d'extraire les données à restituer en sortie de l'exécution de cette ligne. L'élément *MethodLine* possède également un attribut *name* représentant le nom de la ligne afin d'identifier plus facilement sur quelle ligne s'applique ces transformations.

Les éléments de type *Assign* possèdent la même structure que l'élément *MethodLine* comme nous pouvons le constater à la figure 5.55.

```

<Assign parentId="..." leafId="...">
<Input>
...
</Input>
<Output>
...
</Output>
</Assign>

```


Figure 5.55 : Élément Assign

Ces éléments de type *Assign* servent à identifier les transformations de produit à effectuer entre un nœud (décisionnel ou agrégatif) de l'arbre d'exécution de la ligne de méthode et un de ses SMA, feuilles de ce nœud. Comme présenté à la figure 5.55, un élément *Assign* a un attribut *parentId* étant la référence de l'identifiant unique du nœud parent dans l'arbre d'exécution de la ligne et un attribut *leafId* étant la référence de l'identifiant unique du SMA sur lequel doivent s'opérer les transformations de produit. L'élément *Assign* possède deux éléments fils : un *Input* et un *Output*. L'élément *Input* contient les règles de transformation du produit de la ligne de méthode vers le produit en entrée du SMA et inversement pour l'élément *Output* pour le produit en sortie du SMA et le produit de la ligne.

Les deux éléments *Input* et *Output* permettent de regrouper les éléments de type *Transform* à exécuter pour les produits respectivement d'entrée et de sortie (voir figure 5.56).

```
<Input>
<Transform source="..." target="..." xsltRef="http://.../Rule.xslt"/>
<Transform source="..." target="..." xsltRef="http://.../Rule.xslt"/>
...
</Input>
<Output>
<Transform source="..." target="..." xsltRef="http://.../Rule.xslt"/>
<Transform source="..." target="..." xsltRef="http://.../Rule.xslt"/>
...
</Output>
```

Figure 5.56 : Éléments Input, Output et Transform

Chaque élément *Transform* est constitué de trois attributs : *source*, *target* et *xsltRef*. Les attributs *source* et *target* contiennent la référence d'un type de donnée d'un produit défini dans un schéma XSD. L'attribut *xsltRef* contient, quant à lui, une adresse URL pointant vers la règle de transformation à appliquer sur le produit défini par l'attribut *source* pour obtenir le produit défini par l'attribut *target*. Le moteur d'exécution des lignes de méthode de la plateforme se chargera de manipuler les instances des produits et de leur appliquer les règles de transformation. Ces règles sont stockées sous la forme de documents codés avec le langage XSLT qui est un langage de transformation de documents XML. Les références de produits utilisées dans les attributs de l'élément *Transform* vont par conséquent varier en fonction de l'endroit où la règle de transformation est utilisée.

Pour un élément *MethodLine* et *Input*, l'attribut *source* contient une référence d'un type de produit en entrée de la ligne définie dans le schéma XSD des paramètres d'entrées de son descripteur WSDL et l'attribut *target* contient une référence d'un type du schéma XSD du modèle de produit de la ligne. Dans l'élément *Output* d'un élément *MethodLine*, l'attribut *source* d'un élément *Transform* contient la référence du type du schéma XSD du modèle de produit de la ligne à donner en sortie de la ligne. Ce dernier est identifié dans l'attribut *target* par une référence d'un produit défini dans le schéma XSD des paramètres de sortie du descripteur WSDL de la ligne de méthode.

Concernant les éléments *Transform* utilisés dans un élément *Assign* et *Input*, l'attribut *source* correspond à une référence du schéma XSD du modèle de produit de la ligne à transformer pour la transmettre dans le produit identifié dans l'attribut *target* par une référence d'un produit d'entrée du SMA dans la définition XSD de ses paramètres d'entrées dans son descripteur WSDL. Inversement pour les éléments *Transform* utilisés dans un élément *Assign* et *Output*, leur attribut *source* contient la référence d'un produit en sortie du SMA défini dans son descripteur par un schéma XSD et l'attribut *target* la référence d'un fragment du produit du modèle global de la ligne dans lequel intégrer les données en sortie du SMA.

Le moteur d'exécution de la ligne de méthode doit se charger de stocker en mémoire l'instance du produit de la ligne de méthode durant son exécution. Lorsqu'une règle de transformation a pour source un fragment de cette instance de produit, le moteur sélectionne l'ensemble des éléments du produit du type déclaré dans la source de la règle de transformation. Il applique ensuite la règle sur cette sélection et la transmet au SMA dans le type déclaré dans la cible de la règle. Inversement, lorsque que le moteur d'exécution récupère les produits en sortie d'un SMA, il les réintègre dans l'instance du produit global de la ligne en appliquant la règle de transformation associée.

Les règles de transformation peuvent être simples à la manière de celles utilisées dans le standard BPEL (OASIS/BPEL, 2007) dans les activités de type "assign", par exemple un simple passage de paramètres ou alors une sous-sélection dans la liste d'éléments du produit. Des règles de transformation génériques peuvent alors être développées en XSLT et réutilisées pour toutes ces transformations simples. Mais, de par la nature des produits manipulés dans une ligne de méthode, les règles peuvent être complexes. En effet, un produit d'une ligne peut avoir une représentation liée à plusieurs niveaux d'abstraction : instance, modèle, métamodèle, métamétamodèle (voir section 5.2.1.2 de ce chapitre) et le passage d'un produit en entrée d'un SMA peut nécessiter des transformations impliquant un changement de paradigme ou lorsque le produit global de la ligne et celui d'un SMA sont conformes à des métamodèles différents. Une règle de transformation spécifique à chaque changement de cet ordre devra être développée sous la forme d'un document XSLT.

5.3.5.4. Exemple de modèle de transformation des entrées/sorties

Nous présentons dans cette section le modèle de transformation des entrées/sorties de la ligne de méthode GIDPA que nous avons donné en exemple dans la section 5.3.3.2 de ce chapitre. La figure 5.57 reproduit l'implémentation de ce modèle de transformation pour cette ligne qui se présente sous la forme d'un document XML. Ce document étant très volumineux et donc pour des raisons de lisibilité, seules les définitions des entrées/sorties des SMA C1, C2 et de la ligne de méthode ont été représentées sur cette figure (la version complète de ce document est disponible à l'annexe G).

```
<?xml version="1.0" encoding="UTF-8"?>
<am:AssignmentModel xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:am='http://crinfo.univ-paris1.fr/schema/AssignmentModel'
```

```

xsi:schemaLocation='http://crinfo.univ-paris1.fr/schema/AssignmentModel      AssignmentModel.xsd
http://www.w3.org/1999/XMLSchema http://www.w3.org/1999/XMLSchema'
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:mlpm="http://crinfo.univ-paris1.fr/schema/ML-PM-Iteration"
xmlns:c1="http://crinfo.univ-paris1.fr/schema/IterationRequirementsSelection"
xmlns:c2="http://crinfo.univ-paris1.fr/schema/ProductOrientedRequirementRefinement"
xmlns:c3="http://crinfo.univ-paris1.fr/schema/GoalOrientedRequirementRefinement"
xmlns:c4="http://crinfo.univ-paris1.fr/schema/IterationPlanning"
xmlns:c5="http://crinfo.univ-paris1.fr/schema/DesignTasksTracabiliby"
xmlns:c6="http://crinfo.univ-paris1.fr/schema/DevelopmentTasksTracability"
xmlns:c7="http://crinfo.univ-paris1.fr/schema/TestTasksTracability"
xmlns:c8="http://crinfo.univ-paris1.fr/schema/StandUpMeeting"
xmlns:c9="http://crinfo.univ-paris1.fr/schema/DailyMeeting"
>
<am:ProductModels>
  <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/AM-Iteration.xsd">
    <xsd:import namespace=http://crinfo.univ-paris1.fr/schema/ML-PM-Iteration
      schemaLocation="http://crinfo.univ-paris1.fr/schema/ML-PM-Iteration.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/IterationRequirementsSelection"
      schemaLocation="http://crinfo.univ-paris1.fr/schema/IterationRequirementsSelection.xsd"/>
    <xsd:import namespace="http://crinfo.univ-
      paris1.fr/schema/ProductOrientedRequirementRefinement" schemaLocation="http://crinfo.univ-
      paris1.fr/schema/ProductOrientedRequirementRefinement.xsd"/>
    ...
  </xsd:schema>
</am:ProductModels>
<am:MethodLine name="Déroulement d'une itération dans un cycle de vie agile">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="mlpm:ModeleDeProduit"
      xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="mlpm:ModeleDeProduit" target="mlpm: ModeleDeProduit"
      xsltRef="http://.../copyAll.xslt"/>
  </am:Output>
</am:MethodLine>

  <am:Assign parentId="Seq1" leafId="C1">
    <am:Input>
      <am:Transform source="mlpm:ListeDesBesoins" target="c1:ListeDesBesoins"
        xsltRef="http://.../copyAll.xslt"/>
    </am:Input>
    <am:Output>
      <am:Transform source="c1:Iteration" target="mlpm:Iteration"
        xsltRef="http://.../copyAll.xslt"/>
    </am:Output>
  </am:Assign>

  <am:Assign parentId="D1" leafId="C2">
    <am:Input>
      <am:Transform source="mlpm:ListeDesBesoins" target="c2:ListeDesBesoins"
        xsltRef="http://.../copyAll.xslt"/>
      <am:Transform source="mlpm:Iteration" target="c2:Iteration"
        xsltRef="http://.../copyAll.xslt"/>
    </am:Input>
    <am:Output>
      <am:Transform source="c2:ListeDesBesoins" target="mlpm:ListeDesBesoins"
        xsltRef="http://.../replaceAll.xslt"/>
    </am:Output>
  </am:Assign>

  ...
</am:AssignmentModel>

```

Figure 5.57 : Exemple de modèle de transformation des entrées et des sorties d'une ligne de méthode

A partir du modèle de produit de cette ligne de méthode (voir section 5.3.5.2 de ce chapitre), de l'arbre d'exécution de la ligne (voir section 5.3.3.3 de ce chapitre) et des SMA intervenants dans la ligne, il est possible d'établir le modèle de transformation des entrées/sorties de cette ligne de méthode donnée en exemple. Comme présenté en bleu à la figure 5.57, la première étape dans la production de ce modèle

consiste en la définition des espaces de nommage correspondant aux schémas XSD implémentant les modèles de produit de la ligne de méthode et de ses SMA. Il est nécessaire d'importer à cet endroit les modèles un par un pour pouvoir faire référence ultérieurement dans le document aux différents produits de la ligne ou des SMA.

Après la définition de l'élément racine du modèle, vient la définition de l'élément représentant la ligne de méthode. Dans cet exemple, nous avons identifié une règle de transformation pour le produit en entrée de la ligne et une règle pour le produit en sortie. Ces deux règles de transformation sont représentées en vert à la figure 5.57. La première règle signifie qu'un utilisateur doit fournir une liste des besoins en entrée de la ligne conforme à sa définition dans le modèle de produit de la ligne. Cette liste est ensuite copiée entièrement dans l'instance, pour l'instant vide, du modèle de produit global de la ligne. Cette copie est réalisée à l'aide d'un document XLST contenant le code de transformation prenant le produit source et cible au format XML et copie la source en temps qu'élément fils de la cible. Concernant la deuxième règle de transformation, l'instance complète du produit global de la ligne est restituée en sortie à l'utilisateur.

Dans cet exemple de ligne de méthode, tous les SMA manipulent des produits définis dans le même paradigme. Ces produits sont des instances de fragment du modèle de produit de la ligne. En effet, le modèle de produit associé aux SMA de cette ligne est un fragment du modèle global de produit de la ligne.

Le premier SMA de cette ligne est le SMA nommé C1, il permet la sélection des besoins à implémenter dans l'itération. Il possède deux règles de transformation représentées en rouge sur le premier élément *Assign* de la figure 5.57. Il prendra en entrée la liste des besoins dans son état courant à cet instant de l'exécution du processus dans l'instance du produit de la ligne de méthode. Cet SMA sert à construire la partie concernant l'itération dans le modèle de produit. Les données de l'itération sont donc copiées dans l'instance du produit en sortie de l'exécution du SMA.

Le deuxième SMA nommé C2 fournit une méthode de découpage des besoins en tâches. Il prendra donc en entrée deux fragments de l'instance du produit de la ligne : la partie contenant la liste des besoins et la partie contenant l'itération. Il a en effet besoin de connaître les besoins qui seront implémentés dans cette itération car c'est ceux-ci qui doivent être affinés et découpés en tâches. Il restituera en sortie la liste des besoins enrichie avec un affinement des besoins traités dans l'itération ainsi que le découpage des besoins en tâches. La liste des besoins en mémoire dans le produit global de la ligne est donc remplacée par cette nouvelle liste des besoins modifiée à l'aide d'un document XSLT contenant le code nécessaire pour le remplacement d'un élément cible par un élément source. Le SMA C3 étant similaire à C2 il possèdera également deux règles de transformation identiques à celles de C2.

De la même manière, les règles de transformation de produit des autres SMA de la ligne de méthode sont définies dans ce modèle.

Le SMA C4 proposant un outil pour aider à la planification de l'itération prend en entrée deux fragments de produit : la liste des besoins (pour accéder à ceux à traiter et aux tâches), les données relatives à l'itération et un ensemble d'acteurs (qui peut être vide). Il donne en sortie une itération enrichie avec les éléments de planification des tâches, un ensemble d'éléments de type Acteur mis à jour et leurs rôles concernant les tâches à exécuter. Ces deux derniers ensembles de produit sont à ajouter au produit global de la ligne.

Les SMA C5, C6 et C7 manipulent les mêmes produits, ils ont par conséquent des transformations de produits similaires. Ils prennent en entrée l'ensemble des tâches de l'itération et la partie du produit concernant l'itération pour avoir les données de planification et pouvoir identifier les tâches du jour à exécuter. Il restitue en sortie un ensemble de tâches dont l'état à été mis à jour et la trace effectuée.

Les deux derniers SMA de la ligne nommés C8 et C9 traitent de deux manières différentes la rétrospective journalière. Ils manipulent cependant les mêmes types de produit et ont donc des règles de transformation similaires. Ils ont besoin en entrée des données de l'itération et d'un ensemble de tâches pour effectuer l'activité de rétrospective journalière en fonction de la trace d'exécution des tâches qui étaient planifiées dans la journée. Il donne en sortie les données relatives à l'itération contenant en plus les données de la rétrospective effectuée.

Ce modèle de transformation des entrées/sorties est le troisième modèle entrant dans le cadre de l'opérationnalisation des lignes de méthode et il décrit dans cet exemple l'enchaînement de produits et de leurs transformations dans le flux d'exécution de la ligne de méthode du déroulement d'une itération de développement d'un projet agile.

Nous venons de décrire le processus complet de spécification d'une ligne de méthode. La spécification est suffisamment détaillée, d'une part, au niveau du processus de configuration et, d'autre part, au niveau de la gestion des flux de produit pour être exécutable à travers un module d'exécution de la plateforme MaaS. Les deux types de service méthodologique (SMA et LM) sont définis à partir de métamodèles différents et exécutables dans la plateforme MaaS qui est détaillée au chapitre 7.

Nous continuons ce chapitre par le processus progressif d'application de l'approche MaaS.

5.4. Un processus d'IMS progressif

La démarche d'une approche d'IMS lorsqu'elle est appliquée dans une entreprise peut être considérée comme un processus de gestion de méthodes de développement de SI appliquées dans les projets. Ce processus peut donc être exécuté et géré dans une entreprise comme n'importe lequel de ses processus, comme par exemple le processus de gestion de la qualité. Une démarche d'IMS doit, par conséquent,

être soumise aux mêmes règles de qualité et de maturité que le reste des processus utilisés par l'entreprise. De plus, une approche d'IMS étant dotée d'un processus impliquant des modifications importantes dans l'environnement d'une entreprise, son adoption complète ne peut être réalisée en un seul temps. Elle nécessite une intégration progressive dans l'entreprise avec les autres processus existants.

Il existe une multitude de bonnes pratiques pour la gestion de la qualité des processus d'une entreprise. Une partie de ces pratiques ont été regroupées dans la méthode Capability Maturity Model Integration (CMMI) (CMMI, 2006). L'utilisation de cette méthode dans le domaine de l'ingénierie des méthodes a été proposée par (Henderson-Sellers, 2004) pour être appliquée à l'aspect processus des composants de méthode afin d'en améliorer la qualité. Nous proposons de continuer sur cette voie et d'appliquer la méthode CMMI à la démarche d'une approche d'IMS afin d'en améliorer la qualité et que son introduction dans une entreprise soit progressive. En effet, la méthode CMMI propose un cadre de référence pour caractériser les processus d'une entreprise en cinq niveaux de capacité et de maturité :

- **Niveau 0 - incomplet** : à ce niveau de capacité le processus n'est pas ou n'est que partiellement exécuté dans l'entreprise et les buts du domaine d'application du processus ne sont pas satisfaits (CMMI, 2006). D'un point de vue d'une démarche d'ingénierie des méthodes, cela correspond à une absence de démarche et donc à des développements de SI dans l'entreprise sans méthodologie ou par l'utilisation de méthode rigides, comme par exemple le Rational Unified Process (RUP) (Kruchten, 1998).
- **Niveau 1 - initial** : ce niveau de capacité et de maturité représente un processus exécuté dans l'entreprise accomplissant des objectifs de travail et de produits à produire. Les buts spécifiques au domaine d'application sont satisfaits. Cependant, de nombreuses améliorations sont à apporter au processus. En effet, à ce niveau, le succès du processus repose entièrement sur les personnes participant à sa mise en œuvre dans l'entreprise. Le processus est exécuté de manière ad hoc et chaotique. Il n'est pas formalisé, les cas d'incidents sont traités en urgence et par conséquent la répétition d'un succès n'est pas garantie. Ce type de processus nécessite également une optimisation de ses coûts et de ses ressources (CMMI, 2006). Dans le cadre du domaine de l'IMS, ce niveau correspond à l'adaptation de manière ad hoc de méthodes utilisées dans l'entreprise à leurs contextes de projet. Le succès de ces adaptations dépend entièrement de la compétence des personnes les réalisant et la connaissance de ces adaptations n'est pas capitalisée.
- **Niveau 2 - discipliné (ou "managed")** : un processus de niveau 2 est un processus qui est planifié, exécuté et contrôlé afin de satisfaire la politique de l'entreprise. En d'autres termes, le processus fait l'objet d'un suivi et de vérifications dans son exécution. Les produits de ce type de processus sont contrôlés et réalisés par les personnes disposant des compétences requises et

des ressources suffisantes (CMMI, 2006). Concernant une démarche d'IMS, ce niveau se traduit par l'identification de modules ou composants de méthode utilisables dans le cadre de l'adaptation des méthodes de l'entreprise aux situations de ses projets. De plus, la planification de cette phase d'adaptation des méthodes est intégrée dans le processus de gestion des projets et est confiée à un acteur expert, c'est-à-dire un ingénieur méthode. A ce niveau de maturité, un ingénieur méthode peut se servir d'un annuaire de services méthodologiques pour rechercher des SMA et les utiliser pour fournir un support outillé ponctuel aux méthodes et à leurs adaptations.

- **Niveau 3 - ajusté (ou "defined")** : à ce niveau, un processus est caractérisé et maîtrisé. Il est ajusté au contexte de l'entreprise et à ses projets. Contrairement au niveau 2 où les standards et la description du processus peuvent varier d'une instance à l'autre, au niveau 3 sa description est maintenue à jour et est standardisée pour l'ensemble des instances de ce processus. La description de ce type de processus est également formalisée d'une manière plus rigoureuse qu'au niveau inférieur, c'est-à-dire que ses entrées, sorties, activités, rôles et métriques de contrôles sont clairement identifiées et définies (CMMI, 2006). Du point de vue d'une démarche d'IMS, celle-ci est intégrée à l'entreprise avec ses autres processus à ce niveau. Elle est clairement identifiée et s'adapte au contexte de l'entreprise par la définition des lignes de méthode de l'entreprise. Pour cela, la démarche permet de répertorier et raisonner sur les adaptations apportées aux méthodes par l'utilisation de la variabilité dans les méthodes pour former des lignes. L'ensemble des lignes de méthode d'une entreprise permet de répertorier les méthodologies utilisées et les variations nécessaires à leurs adaptations aux différents contextes projets rencontrés. Ces lignes sont prises en charge par une plateforme d'exécution et fournissent un support outillé complet aux méthodes de l'entreprise. Ce support outillé s'adaptant aux différentes variations est fourni de par la composition d'une ligne comme une orchestration d'outils élémentaires et indépendants : les SMA. Ce niveau de maturité comprend donc l'identification et la formalisation des lignes de méthode et des SMA de l'entreprise.
- **Niveau 4 - géré quantitativement (ou "quantitatively managed")** : un processus à ce niveau de maturité doit répondre à des objectifs quantitatifs de qualité et de performance. En d'autres termes, la qualité et la performance du processus sont gérées sous la forme de statistiques dans le cycle de vie du processus. L'identification de ces objectifs quantitatifs est basée sur les besoins de l'organisation, du client ou des utilisateurs finaux. La différence entre les niveaux 3 et 4 réside dans la possibilité d'établir des prévisions concernant la performance d'un processus grâce à l'utilisation de ces métriques de qualité et de leurs analyses statistiques (CMMI, 2006). A ce niveau de maturité, la démarche d'IMS est vue par l'entreprise comme un

de ses processus et le gère comme n'importe lequel de ses projets. Par conséquent, le processus d'IM de l'entreprise doit être soumis aux mêmes métriques de performance que les autres processus ou projets de l'entreprise, c'est-à-dire en termes de délai, d'efficacité et d'efficience de la production et de la configuration des lignes de méthode. Il en va de même concernant la pertinence de la sélection des SMA, de leur qualité ou de leur délai de production. Etant donné que la démarche d'IMS de l'entreprise impacte d'autres processus de l'entreprise comme ses méthodes de développement de SI, les méthodes configurées par la démarche doivent également être soumises à un processus d'évaluation afin de mesurer l'efficacité des configurations effectuées. Cela permet d'établir des statistiques sur les configurations des lignes de méthode, de mesurer le taux d'application des méthodes et le respect de la méthodologie. Ensuite, il est possible d'utiliser ses statistiques pour aider les acteurs de l'entreprise à mieux choisir leurs configurations et ainsi augmenter la qualité de leurs processus méthodologiques.

- **Niveau 5 - en optimisation (ou "optimizing")** : le dernier niveau de maturité caractérise un processus continuellement optimisé à partir des analyses quantitatives de ses métriques de qualité et de performance. L'entreprise utilise alors une approche quantitative pour la compréhension des variations des résultats de l'exécution du processus. L'organisation s'inscrit dans une boucle itérative et incrémentale d'optimisation du processus continue. A ce niveau, l'optimisation des processus est réalisée à partir de l'analyse des données de leurs multiples exécutions. Cette optimisation des processus est également formalisée dans la démarche de gestion de l'entreprise (CMMI, 2006). Du point de vue de la démarche d'IMS, cela correspond à l'optimisation de la production des lignes de méthode et des SMA. Mais, cela correspond également à l'optimisation de la variabilité des lignes de méthode, des SMA utilisés et des guides de configuration des lignes de méthode afin d'augmenter la qualité des lignes et de les faire évoluer avec l'entreprise.

Ce tronc commun utilisé par les différentes déclinaisons de la méthode CMMI ("for acquisition", "for development" ou "for services") appliqué au domaine de l'IMS permet d'évaluer le processus d'ingénierie des méthodes d'une entreprise. En fonction du niveau de maturité mesuré, il est possible de proposer une solution basée sur son niveau de maturité pour l'intégration de la démarche d'une approche d'IMS dans l'entreprise.

5.5. Conclusion

Nous avons présenté dans ce chapitre un métamodèle de composant de méthode technique, le SMA, pour le support outillé des composants conceptuels et un métamodèle de ligne de méthode afin

d'apporter de la variabilité dans les méthodes. Tous deux étant perçus comme des services méthodologiques, le premier représentant l'aspect atomique et le deuxième l'aspect composite.

Le métamodèle de SMA permet de décrire l'aspect outil d'un composant de méthode. Il est l'implémentation d'un outil d'un composant conceptuel. Cette version "technique" d'un composant de méthode implémente le processus d'un composant dans un service "boîte noire". Il applique ce processus sur le produit qu'il reçoit en entrée pour fournir le produit de sortie. L'implémentation peut prendre la forme d'un seul service (SMA simple) ou d'une composition de services (SMA complexe). Le ou les services implémentant le composant conceptuel étant de type métier, ils sont alors encapsulés dans un service interactif afin d'être directement utilisables par un utilisateur. C'est ce service interactif ajouté à son descripteur technique et au descripteur sémantique du composant implémenté qui forme le concept de SMA. Ces SMA conservent toutes les propriétés des services, ils sont standardisés, interopérables et faiblement couplés. De plus, ils sont indépendants de l'approche utilisée pour créer le composant conceptuel qu'ils implémentent. Ils peuvent être utilisés de manière autonome ou être utilisés comme éléments de base dans la construction d'une ligne de méthode.

Les lignes de méthode apportent la vision des *Packages méthodologiques* d'une entreprise. Il est alors possible pour une entreprise de représenter dans une ligne l'adaptation de ses méthodes aux contextes projets autour d'un thème. A la manière de la modélisation des processus métiers dans une entreprise, les lignes de méthode regroupent les méthodes de l'entreprise par thème et permettent d'introduire la variabilité que nous retrouvons dans les adaptations de méthodes effectuées en pratique. De plus, l'opérationnalisation des lignes de méthode fournit un support outillé à la configuration des méthodes dans une entreprise. Cette opérationnalisation se traduit par trois modèles opérationnels fournissant toutes les facilités pour l'exécution d'une ligne : l'arbre d'exécution d'une ligne, le modèle de configuration et le modèle de transformation des entrées/sorties. L'arbre d'exécution décrit le processus de la ligne de méthode en mettant en avant les points de variation de la ligne, c'est-à-dire les instants où les utilisateurs de la ligne doivent faire des choix dans le processus méthodologique. Ils sont alors aidés dans leur tâche décisionnelle par le modèle de configuration de la ligne. En effet ce dernier regroupe tous les guides de configuration (heuristiques ou contraintes d'exécution) aidant les utilisateurs à faire leurs choix. Ils sont également aidés par les facteurs de contingence liés aux différentes options de choix qui leurs sont présentées. Ces facteurs permettent de décrire les éléments discriminants entre les différentes alternatives s'offrant à un utilisateur. Le flux de produit dans le processus de la ligne de méthode est, quant à lui, géré par le modèle de transformation des entrées/sorties. Ce dernier modèle fournit les règles de transformation des produits en entrées et en sorties de chaque SMA de la ligne vers le produit global de la ligne de méthode. Il permet alors, lors de l'exécution du processus de la ligne, de transmettre les produits en entrées d'un SMA et de récupérer le produit en sortie de ce SMA pour les transmettre en entrée du SMA suivant dans le flux

d'exécution. Ces modèles sont implémentés à l'aide de document XML, facilitant ainsi leur opérationnalisation. Etant définie comme un *Paquetage méthodologique*, une ligne de méthode est conforme aux aspects de la définition d'une méthode selon Seligmann (Seligmann, 1989), c'est-à-dire qu'elle possède :

- un aspect paradigme : le concept de variabilité appliqué aux méthodes,
- un aspect processus : le concept de ligne de méthode représentant un processus méthodologique configurable,
- un aspect produit : le modèle de produit global de la ligne,
- un aspect outil : le support outillé de la ligne au travers de ses modèles opérationnels et de sa plateforme d'exécution.
- un aspect organisationnel peut aisément être ajouté en intégrant dans son modèle d'acquisition du processus les aspects organisationnels de la modélisation avec le standard BPMN.

Une ligne de méthode étant un service méthodologique elle possède elle-même un descripteur pouvant être publié dans un annuaire.

Afin de favoriser l'utilisation des SMA et des lignes de méthode dans les entreprises, nous avons proposé dans ce chapitre une architecture de plateforme d'exécution des services méthodologiques. Cette architecture est constituée de deux composants logiciels, d'un composant générique ayant pour objectif la gestion des interactions avec les utilisateurs et fournissant un moteur d'exécution des lignes, puis d'un composant spécifique gérant l'invocation des SMA.

Toujours dans l'optique de favoriser l'usage de l'approche MaaS dans les entreprises, nous avons proposé une adaptation des cinq niveaux de capacité et de maturité des processus CMMI à une démarche d'IMS. Cela permet ainsi une intégration progressive d'une démarche d'IMS dans une entreprise et d'augmenter la qualité de cette démarche tout en minimisant la résistance aux changements des utilisateurs concernés.

Les solutions proposées dans ce chapitre répondent à l'objectif de bénéficier d'un support outillé standardisé et interopérable pour les composants et les méthodes situationnelles construits à l'aide d'une approche d'IMS. Elles répondent également à l'objectif du besoin de configuration des méthodes dans les entreprises. L'application de la démarche de qualité CMMI à une démarche d'IMS permet de répondre à l'objectif d'une introduction progressive des approches d'IMS dans les entreprises.

Chapitre 6. Cas d'application

6.1. Introduction

Après avoir présenté notre approche dans les chapitres précédents, nous consacrons ce chapitre à la présentation d'un cas pratique d'application. Nous présentons, l'identification d'une ligne de méthode à partir des différentes étapes de lancement d'un projet utilisées dans les trois méthodes agiles répandues dans les entreprises : la Dynamic System Development Method (DSDM), l'Extreme Programming (XP) et Scrum. Nous décrivons comment le processus de cette ligne de méthode est modélisé à l'aide du langage BPMN en respectant les contraintes de modélisation imposées par notre approche. Ensuite, nous montrons comment un ingénieur méthode peut implémenter cette ligne de méthode en utilisant le modèle de représentation que nous avons proposé à la section 5.3.3.3 du chapitre 5. Nous détaillons ensuite l'ensemble des composants de méthode intervenant dans cette ligne et comment ils sont outillés par les services méthodologiques atomiques de notre approche. Enfin, nous présentons les transformations de produit nécessaires au passage des produits en sortie des services méthodologiques vers les entrées des services suivants dans la ligne de méthode.

La section 6.2 de ce chapitre présente une description du cas d'application « Lancement d'un projet agile » accompagné d'une description des étapes de lancement de projet des méthodes agiles DSDM, XP et Scrum. La section 6.3 présente le processus de la ligne de méthode au format BPMN et la section 6.4 son implémentation au format XML en utilisant l'approche MaaS. La section 6.5 décrit l'ensemble des composants de méthode intervenant dans la ligne de méthode à la fois d'un point de vue conceptuel en utilisant l'approche des composants de méthode chunk et technique sous la forme de services méthodologiques atomiques en utilisant l'approche MaaS. Enfin, la section 6.6 présente la spécification des transformations des entrées/sorties des services méthodologiques atomiques pour la ligne de méthode.

6.2. Cas d'application : une ligne de méthode d'un lancement de projet agile

Le cas d'application présenté dans ce chapitre traite de la construction d'une ligne de méthode depuis zéro en regroupant les étapes définies dans les trois méthodes agiles (DSDM, XP et Scrum) très connues et utilisées pour notre exemple, dans une même entreprise. Nous limitons notre exemple à la phase de lancement d'un projet agile. La construction de cette ligne de méthode est basée sur l'approche MaaS que nous proposons dans cette thèse. La ligne en elle-même est constituée de services SMA fournissant un support outillé aux composants de méthode constituant les trois méthodes agiles choisies. L'objectif de ce travail est de fournir une ligne de méthode organisant les bonnes pratiques des trois méthodes agiles DSDM, XP et Scrum spécifiques à la phase de lancement du projet dans un

seul processus supporté par un outil afin de faciliter leur application en fonction de la situation de projet par des non experts. Nous détaillons dans ce cas d'application les étapes de la construction de la ligne depuis la modélisation de son processus jusqu'aux détails des SMA la composant.

6.2.1 Les méthodes agiles : étape de lancement d'un projet

Cette sous-section regroupe la description des étapes de lancement de projet utilisées dans les méthodes agiles DSDM, XP et Scrum. Ces descriptions serviront de base pour la définition du processus d'une ligne de méthode. (Tous les concepts issus des trois méthodes agiles sont mentionnés entre guillemets "").

6.2.2 Dynamic System Development Method

La méthode DSDM a été créée dans le début des années 1990 par le DSDM Consortium (DSDM, 2009). C'est une méthode de gestion de projet de développement dite "agile". Elle est en effet basée sur un cycle de vie en spirale. Elle est centrée sur l'inclusion des utilisateurs finaux du système à développer dans les définitions des besoins et des tests. Cette méthode propose de réaliser des applications par prototypage et un développement itératif.

La figure suivante présente le cycle de vie de la méthode agile DSDM.

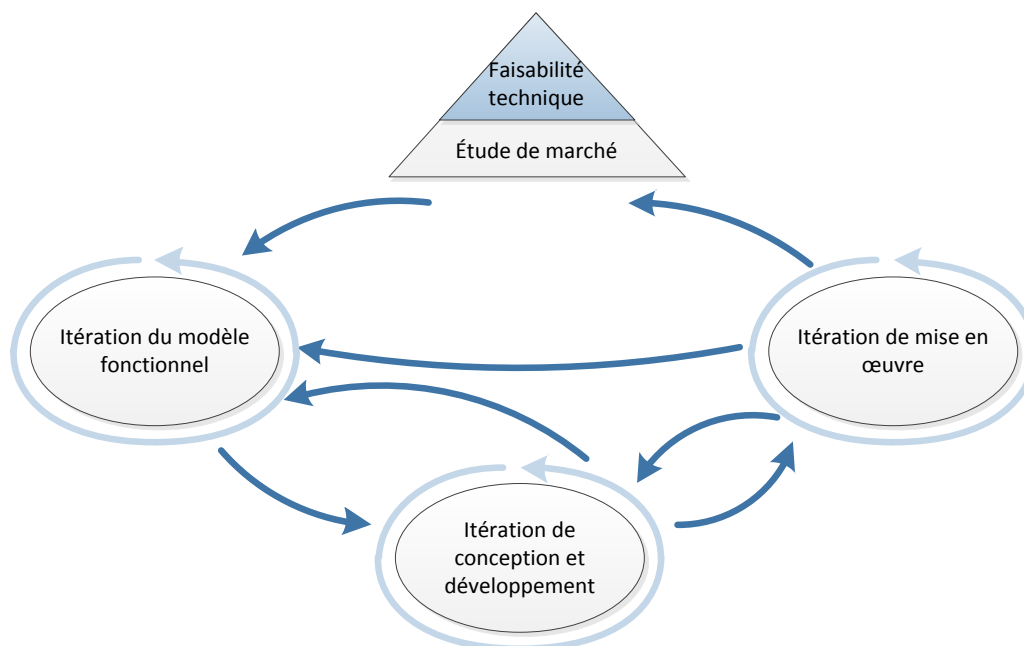


Figure 6.1 : Cycle de vie de la méthode agile DSDM

Le cycle de vie de la méthode démarre par une étape de lancement de projet, puis, il est organisé en trois spirales itératives de développement (voir figure 6.1). L'étape de lancement de projet ou "pré-projet" consiste en deux activités : une étude de marché et une étude de faisabilité technique du projet. Les trois cycles itératifs de développement concernent, quant à eux, la réalisation du modèle

fonctionnel de l'application à développer, le cycle de design et de développement puis le cycle d'implémentation.

Le cycle concernant le modèle fonctionnel a pour objectif la réalisation des modèles statiques de l'application à réaliser ainsi que le développement du premier prototype correspondant aux besoins fonctionnels identifiés par les utilisateurs et l'identification des besoins non fonctionnels de l'application à développer (DSDM, 2009).

Les itérations de conception et de développement permettent d'affiner les prototypes réalisés dans les cycles précédents afin qu'ils satisfassent l'ensemble des besoins non fonctionnels du projet (DSDM, 2009). Cette étape comprend également les activités de test du système.

Le dernier cycle de la méthode concerne le déploiement du système développé dans son environnement de production réel. Les objectifs de ce cycle sont de confronter les utilisateurs finaux avec le système, de les former, de déterminer les améliorations futures à apporter et de préparer la maintenance du système ainsi que les personnes impliquées dans cette maintenance (DSDM, 2009).

Ces trois cycles de développement ne sont pas exécutés en une séquence figée. Comme le montre la figure 6.1, il est possible de naviguer entre les trois cycles et d'exécuter de nouvelles itérations de cycles précédents si des changements surviennent en cours de projet ou afin d'affiner les produits de ces cycles.

Deux étapes d'étude sont effectuées avant l'exécution de ces trois cycles de développement. Elles permettent de valider ou d'invalider la faisabilité du projet d'un point de vue du marché ou technique.

L'étude de faisabilité se concentre sur la détermination de l'existence d'une solution réalisable au problème posé. Elle permet de détailler l'ensemble des exigences de haut niveau ainsi que les risques pour l'application ou le système à développer. Cette étape prend en compte également la réalisation d'une première planification dans les grandes lignes du projet ("plan directeur") et donc de déterminer les premières estimations de délais du projet. Il est préconisé dans la méthode de limiter cette première étude dans le temps, c'est-à-dire qu'elle soit de courte durée par rapport au reste du projet. DSDM préconise également de réunir les acteurs impliqués dans le développement du projet pour une première réunion ou atelier avec l'objectif de vérifier l'adéquation des acteurs clefs du projet sur la définition du problème à résoudre, les risques du projet, la composition de l'équipe projet et la répartition des rôles au sein de cette dernière. Les résultats de cette première étude sont consignés dans un rapport de faisabilité. Ce rapport regroupe les informations sur le problème à adresser avec les solutions alternatives qu'il est possible de développer. Ajouté à ce rapport, un "plan directeur" permet de réaliser la première planification du projet. Ce planning consiste à définir les deadlines et les jalons des différentes grandes phases du projet en fonction de la liste des besoins de haut niveau. Il est

également possible de développer au cours de cette phase des prototypes simples visant à tester la faisabilité des solutions.

L'étude métier de DSDM concerne l'identification de la problématique métier du projet et son adéquation avec la stratégie de l'entreprise. Cela passe par l'identification du domaine métier lié au projet, c'est-à-dire par une vue de haut niveau des processus métier, des utilisateurs, des informations des interfaces avec d'autres systèmes et des besoins métier qui seront pris en charge par la solution (DSDM, 2009). Cette étape doit également prendre en compte une vérification de l'applicabilité de la méthode DSDM au projet et fournir une estimation du coût du projet. Le recueil de toutes ces informations liées au domaine métier du projet permet d'affiner puis de prioriser la liste des besoins en collaboration avec les utilisateurs finaux du système. Une première ébauche des besoins non fonctionnels peut être réalisée au cours de cette étape. La planification peut ensuite être détaillée davantage en partant du "plan directeur" pour répartir les besoins sur les jalons.

6.2.3 Extreme Programming

La méthode XP a été créée par Kent Beck et Ward Cunningham en 1999 (Wells, 2009). Cette méthode de gestion de projet de développement de logiciel contient un recueil de règles et de bonnes pratiques de développement s'utilisant conjointement les unes aux autres. Ces bonnes pratiques sont concentrées autour de quatre thèmes principaux tels que : la planification, le design, le code source et les tests. XP a été créé en réponse aux problèmes rencontrés par les méthodes classiques dans des environnements où les exigences du client changent fréquemment. Les deux piliers revendiqués par la méthode XP sont l'introduction du client dans le processus de développement et la testabilité du code source. En effet, les relations contractuelles sont gommées et remplacées par des relations collaboratives entre le client et l'équipe de développement. Cette méthode requiert donc une forte disponibilité du client et de préférence une communication face à face avec l'équipe du projet. Au niveau des tests, l'XP nécessite la mise en place de tests du code source fonctionnels et automatisés. Le but premier de l'XP est de délivrer l'application dont le client a besoin quand il en a besoin.

La figure ci-dessous présente le cycle de vie de la méthode agile XP.

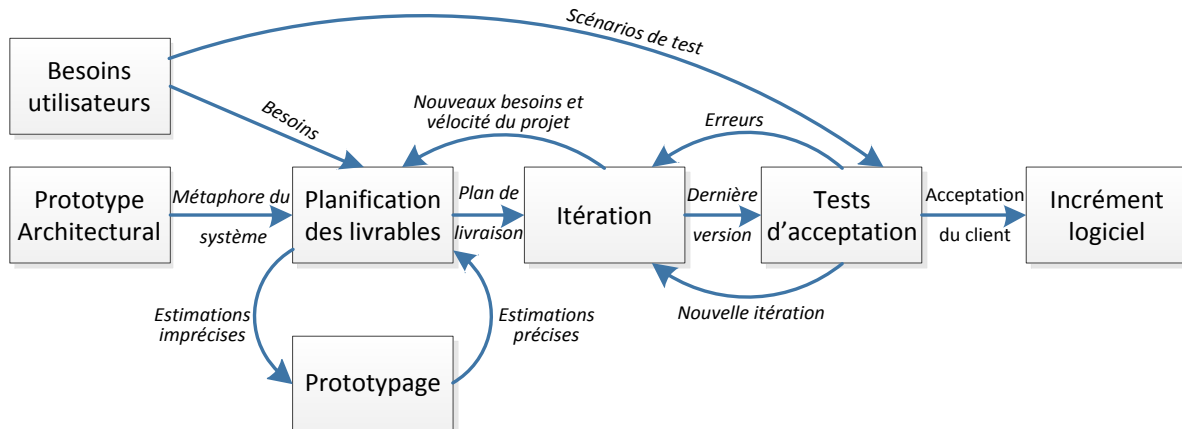


Figure 6.2 : Cycle de vie de la méthode agile XP

Comme présenté à la figure 6.2, le cycle de vie de l'XP s'organise autour d'un cycle de livraison de petits incréments au client. La première étape d'un projet XP est la définition des besoins du client par l'écriture de "besoin utilisateur" qui peuvent s'apparenter à des cas d'utilisation. Après avoir choisi un style d'architecture pour le système à développer, une première planification du projet peut être définie. Commence ensuite le cycle itératif de production des incréments, chaque cycle incorporant les activités de modélisation, développement et test. A la fin de chaque itération, en fonction du déroulement de son exécution, une activité de rétrospective est effectuée par l'équipe afin d'ajuster la définition des besoins et la planification du projet. Lorsque qu'un incrément est livrable, il est validé par une série de tests d'acceptation par le client. Il est ensuite déployé dans son environnement de production.

Le lancement d'un projet en XP commence par l'étape de définition des besoins à l'aide des "besoins utilisateur". Ce type de définition des exigences du client vis-à-vis du logiciel à développer reprend les principes des cas d'utilisation sans, toutefois, en reprendre la forme (Wells, 2009). Chaque "besoin utilisateur" est écrit par le client en trois phrases exprimant un besoin à réaliser dans le système à développer. A la différence des cas d'utilisation, les "besoins utilisateur" doivent contenir suffisamment de détails pour pouvoir établir une première estimation à faible risque de leur temps de développement. A partir de cette première estimation de temps, des recommandations de la méthode XP sont à appliquer : un "besoin utilisateur" doit avoir une durée d'implémentation d'une à trois semaines. Si l'implémentation d'un "besoin utilisateur" est estimé à plus de trois semaines, le "besoin utilisateur" est défini à un niveau trop élevé, il faut donc le séparer en plusieurs "besoins utilisateur". Au contraire, si l'implémentation d'un "besoin utilisateur" est estimé à moins d'une semaine, il est trop détaillé, il faut alors le rassembler avec d'autres "besoins utilisateur". Dans l'objectif de réaliser des estimations des "besoins utilisateur" plus précises, il est possible de réaliser des prototypes afin d'explorer les solutions possibles pour un "besoin utilisateur". La spécification détaillée des "besoins utilisateur" est réalisée ultérieurement, au cours d'une itération, lorsque que le "besoin utilisateur" aura

été sélectionné pour être implémenté. Dans le but de faciliter la planification générale du projet et des jalons, il est préconisé dans XP d'écrire les "besoins utilisateur" sur des fiches cartonnées afin de pouvoir manipuler rapidement et facilement leur ordonnancement. Un "besoin utilisateur" particulier est également défini au cours de cette étape de définition des besoins, il correspond à la métaphore du système à réaliser. En d'autres termes, c'est un "besoin utilisateur" expliquant comment le système fonctionne de manière compréhensible par tout le monde.

Un premier planning appelé "planning des livrables" peut ensuite être établi à partir de la liste des "besoins utilisateur" et des estimations de délai de celles-ci. Ce planning contient le découpage du projet en jalons. Dans cette étape de lancement du projet, la planification du premier "livrable" est détaillée par un découpage en itération. Les "besoins utilisateur" sont ensuite priorisés avec l'aide du client et une partie des "besoins utilisateur" est réparti sur les itérations du premier "livrable" en fonction de leurs priorités et de leurs estimations de délai. Ce premier planning sera modifié tout au long du déroulement du projet pour venir ajuster la répartition des "besoins utilisateur" sur les itérations en fonction de la productivité de l'équipe et pour planifier les "livrables" suivants au fur et à mesure de la livraison des incréments logiciels.

6.2.4 Scrum

La méthode Scrum (Schwaber, 2011) est une méthode de gestion de projet agile basée sur un processus itératif et incrémental (voir figure 6.3). Le terme "Scrum" provient du rugby dans lequel sa signification est la remise en jeu en équipe d'une balle hors jeu. Ce terme a été utilisé la première fois par Takeuchi et Nonaka en 1986 (Takeuchi, 1986). Comme beaucoup d'autres méthodes agiles, Scrum est basée sur un regroupement empirique de pratiques et de processus industriels sur la gestion du chaos de leur environnement de développement. Cette méthode introduit les concepts de flexibilité, d'adaptabilité et de productivité dans le processus de développement d'un logiciel.

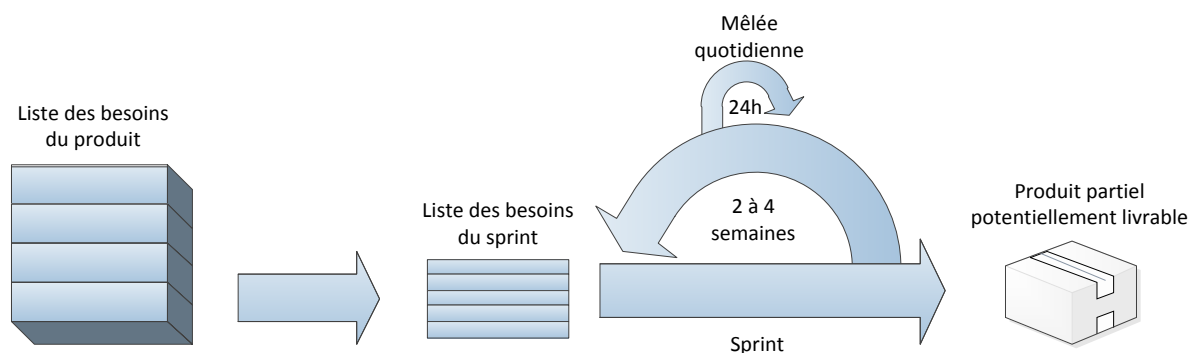


Figure 6.3 : Cycle de vie de la méthode agile Scrum

Le cycle de vie de cette méthode se résume en l'exécution d'une série de "sprint", c'est-à-dire une série d'itération ("sprint") regroupées en jalons afin de produire des incréments "livrables" au client. A chaque itération, l'équipe sélectionne un ensemble de besoins d'une liste des besoins priorisée appelée

"liste des besoins du produit" pour former un sous-ensemble nommé "liste des besoins du sprint". Dans cette liste des besoins de "sprint", les besoins sont affinés, puis découpés en tâches à réaliser par l'équipe du projet. L'ensemble des tâches est ensuite ordonnancé pour former un planning du "sprint". Ces tâches comprennent l'ensemble des tâches classiques de développement de logiciel comme la modélisation, le développement et les tests, pour chaque besoin à traiter dans ce "sprint". Tout au cours de l'exécution d'un "sprint", une "réunion journalière" est réalisée par le chef de projet afin de faire le point avec l'équipe sur les tâches en cours, celles achevées et celles à venir. Cette réunion régulière de courte durée (environ X minutes) permet à l'équipe de mieux piloter le projet au cours d'un "sprint". Enfin, à l'issue d'un "sprint", une étape rétrospective est effectuée afin d'intégrer les retours sur le "sprint" venant d'être exécuté dans le projet. Autrement, on effectue une mise à jour de la liste des besoins à traiter dans la "liste des besoins du produit" par les retours de l'équipe ou du client. Il est également possible à cette étape de modifier la planification globale du projet, si besoin est. Lorsque l'ensemble des "sprints" d'un "livrable" sont terminés et qu'un incrément logiciel est produit, il est alors livré au client et intégré dans son environnement de production.

Un projet Scrum commence par un "sprint" particulier nommé le "sprint 0". Dans ce "sprint", les tâches à effectuer au cours de son déroulement sont différentes des autres "sprints". Son objectif est d'établir le blacklog produit initial et une première planification du projet.

La "liste des besoins du produit" est une liste de "besoins utilisateur" ordonnés par ordre de priorité (Schwaber, 2011). Il existe une seule "liste des besoins du produit" pour un produit. Elle regroupe l'ensemble des besoins relatifs à ce produit. La "liste des besoins du produit" n'est pas une liste figée, l'ordonnancement des "besoins utilisateur" peut changer au cours du projet et des "besoins utilisateur" peuvent ajoutés ou supprimés de la "liste des besoins du produit" en fonction des retours de l'équipe et du client. Un "besoin utilisateur" permet d'identifier un besoin du client vis-à-vis du système à développer. Il est constitué principalement d'un titre, d'une description et d'une condition pour vérifier sa complétude (Schwaber, 2011). Son descripteur doit décrire un élément fonction apportant de la valeur pouvant être implémenté au cours d'un "sprint". Il est également intéressant d'estimer la taille de l'effort à fournir pour réaliser un "besoin utilisateur" ainsi que de tracer son état (par exemple : à faire, en cours, terminée). Dans la méthode Scrum, il existe trois types de "besoins utilisateur" différents (Schwaber, 2011). Les "besoins utilisateur" décrivent un comportement du système attendu par les utilisateurs. Les "besoins utilisateur techniques" décrivent des besoins internes à caractère technique au système à développer, ces "besoins utilisateur" sont connus de l'équipe de développement mais ils sont invisibles pour les utilisateurs. Les "besoins utilisateur de type défaut" servent à répertorier les anomalies (bug ou évolution du système) à traiter. L'estimation du temps d'implémentation d'un "besoin utilisateur" est réalisée en utilisant la technique du "poker de planification", c'est-à-dire que chaque membre de l'équipe va formuler une estimation simultanément pour un "besoin utilisateur" à

l'aide d'un jeu de carte. Il s'ensuit une discussion entre les membres de l'équipe pour ajuster cette estimation. La priorisation des "besoins utilisateur" dans la "liste des besoins du produit" est réalisée, quant à elle, par l'étude pour chaque "besoin utilisateur" des risques qu'il permet de réduire, de la réduction de l'incertitude dans la définition des besoins du client grâce aux retours rapides qu'il permettra d'obtenir, de la qualité qu'il apporte au produit fini et enfin, à la dépendance qu'il peut avoir avec d'autres "besoins utilisateur".

Une fois la "liste des besoins du produit" établie, la méthode Scrum préconise de réaliser une première planification globale du projet appelée "planification générale des livrables". Cette planification commence par l'identification des "livrables", de leur type, de leur condition de fin et de la date de leur livraison. La date de fin de "livrable" va dépendre du type du "livrable" et donc de la technique employée pour établir sa planification. En effet, il est possible de planifier les "livrables" en fonction du produit à livrer, autrement dit que le "livrable" est terminé lorsqu'un sous ensemble de "besoins utilisateur" de la "liste des besoins du produit" sont implémentés. L'estimation de délai de chaque "besoin utilisateur" permet ensuite de déterminer une date de fin de "livrable". Il est possible de planifier les "livrables" en fonction du budget ou encore de délivrer les "livrables" à dates fixes. Lorsque l'ordonnancement des "livrables" est établi, il ne reste plus qu'à fixer la durée des "sprints" dans le projet ou par "livrable" afin d'obtenir le nombre de "sprints" à effectuer par "livrable". La planification détaillée des "sprints" sera réalisée au début de chacun d'entre eux. La sélection des "besoins utilisateur" à développer durant un "sprint" sera également effectuée au début de celui-ci.

6.3. Modèle d'acquisition de la ligne de méthode d'un lancement de projet agile

En partant des trois méthodes agiles présentées dans ce chapitre, il est possible de créer une ligne de méthode les combinant. Nous nous intéresserons dans cette section à la création du processus de la ligne de méthode concernant l'étape de lancement d'un projet agile. L'objectif est d'aider l'entreprise à choisir l'application la plus optimale des bonnes pratiques des méthodes agiles Scrum (Schwaber, 2011), XP (Wells, 2009) et DSDM (DSDM, 2009) en fonction du contexte du projet.

6.3.1 Modélisation BPMN de la ligne de méthode de lancement de projet agile

En étudiant le détail des activités préconisées par les méthodes agiles DSDM, Scrum et XP lors du lancement d'un projet, nous pouvons clairement distinguer deux groupes d'activités, celles liées à la définition des besoins du client vis-à-vis du système à réaliser et celles traitant de la planification globale de haut niveau du projet. La méthode DSDM préconise en plus de ces deux groupes d'activités d'effectuer une étude technique et une étude de marché avant tout chose dans le projet. Nous pouvons donc introduire un troisième groupe d'activités d'étude dans la ligne de méthode.

Etant donné que les activités d'études de marché et technique sont uniquement proposées par la méthode DSDM, nous proposons de les rendre optionnelles et de laisser le choix aux utilisateurs de faire ses activités avant de passer à la définition des besoins du projet. Ce premier choix est intégré par le point de variation de type "ou exclusif" et nommé D1 dans le processus de la ligne de méthode (voir figure 6.4).

Si un utilisateur choisit de passer par la branche D1.1 du nœud D1, il devra choisir d'exécuter l'activité d'étude de marché (C1) ou d'étude technique (C2) ou les deux. Cela est représenté par le nœud D2 de type "ou" entre les deux activités (ou composants de méthode) C1 et C2. Par rapport à leur méthode d'origine (DSDM), les activités C1 et C2 sont un sous-processus de l'étape de pré-projet. Uniquement les recommandations et processus concernant l'étude de marché et l'étude technique sont conservés dans C1 et C2. Les parties de la méthode DSDM concernant la définition des besoins, ou encore la planification, seront reprises ultérieurement dans d'autres composants. Le processus de la ligne de méthode est ensuite synchronisé juste en sortie de C1 et C2. Autrement dit, l'exécution de la ligne de méthode est mise en attente du résultat produit par C1 et/ou C2 avant de continuer. En effet, la condition suivante, nommée D4, porte sur les produits en sortie de C1 et/ou C2. La condition de D4 concerne la décision prise dans les études de marché et technique. Est-ce que la décision des études est favorable à la poursuite du projet? Si le résultat d'une des deux études donne un avis défavorable à la poursuite du projet alors le composant C3 de clôture du projet est exécuté et le processus s'arrête. Si les résultats des études sont positifs, le processus de la ligne continue. La partie du processus concernant la définition des besoins est alors exécutée comme si l'utilisateur avait choisi la branche D1.2 de D1.

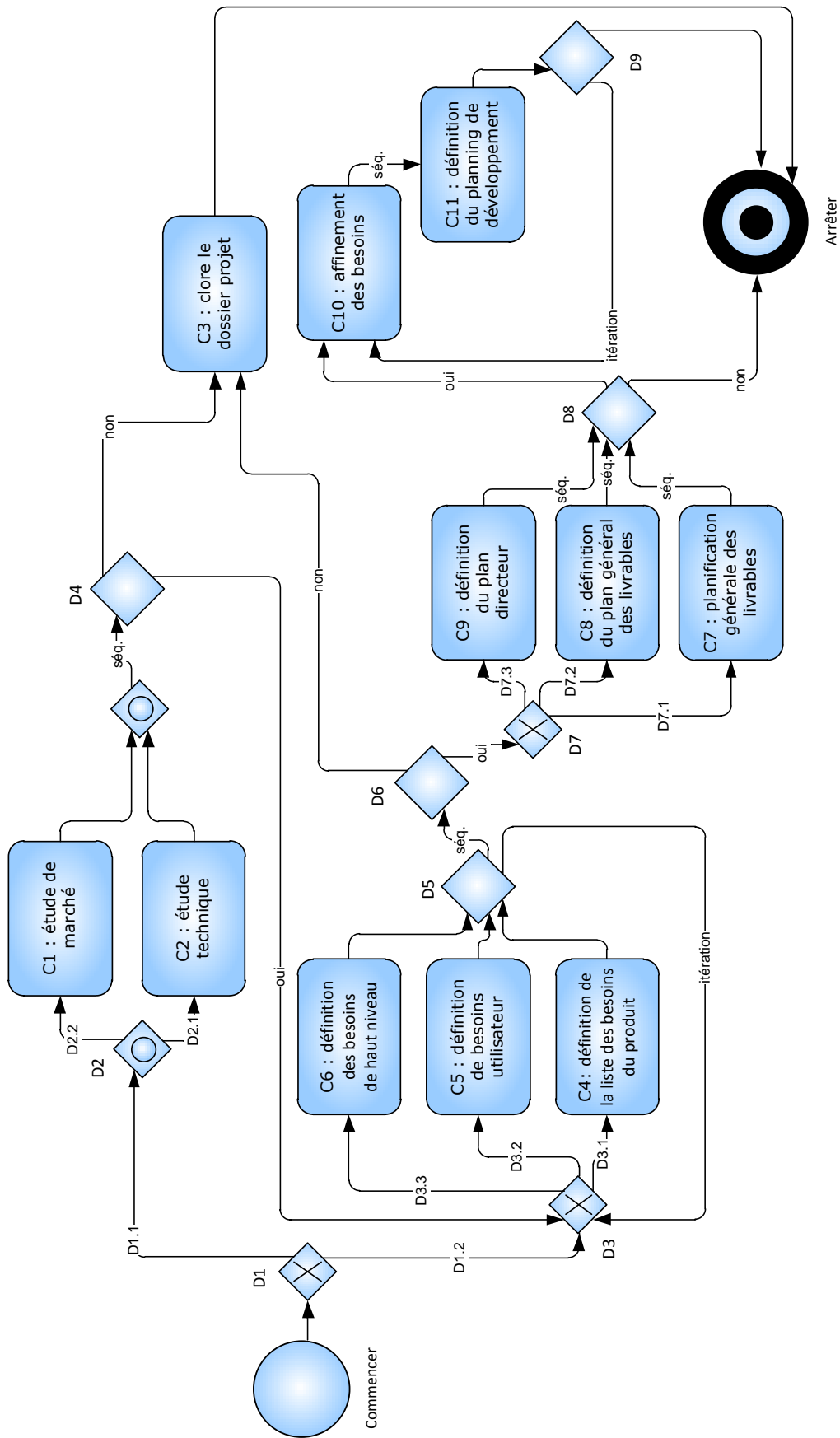


Figure 6.4 : Exemple de ligne de méthode de lancement de projet agile

Au niveau du nœud de décision D1, si un utilisateur choisit directement la branche D1.2, il passe directement à la définition des besoins du système à développer dans le projet. Chacune des trois méthodes agiles proposent une méthodologie différente pour définir les besoins (de haut niveau) du client. Nous proposons donc à l'utilisateur de choisir une méthode parmi les trois par un nœud de choix (D3) de type "ou exclusif" pour l'aider à établir une liste des besoins de haut niveau pour le projet. Le choix s'opère entre trois composants C4, C5 et C6. Le composant C4 est issu de la méthode Scrum. Il propose d'identifier les besoins du client sous la forme de "besoin utilisateur", d'estimer leur taille par la technique dite du "poker de planification", puis de prioriser l'ensemble des "besoins utilisateur" par rapport aux risques et aux incertitudes qu'ils permettent de réduire. Pour cela, le composant C5, issu de la méthode XP, propose de réaliser une liste des besoins par une itération de définition de besoins et d'estimation. Chaque besoin est identifié sous la forme d'un "besoin utilisateur" de façon similaire au composant C4. Le "besoin utilisateur" est ensuite estimé en temps de développement idéal. Si son temps estimé est trop important, le "besoin utilisateur" est alors découpé en plusieurs "besoins utilisateur". Inversement, s'il est trop petit, il est regroupé à d'autres "besoins utilisateur" pour en former un plus important. La méthode XP propose également un style moins formel pour la rédaction des "besoins utilisateur" que la méthode Scrum. Concernant le composant C6, issu de DSDM, il propose d'inclure les futurs utilisateurs finaux du système à développer directement dans le processus pour la définition et la priorisation d'une liste de besoins de haut niveau répondant à la problématique du projet. Après avoir exécuté un des trois composants C4, C5 ou C6, il est possible, si la liste des besoins n'est pas complète ou satisfaisante, de revenir au nœud D3 afin de refaire une étape de découverte de besoins. Cela est modélisé dans la ligne de méthode par le nœud D5 portant la condition de complétude sur la liste des besoins en résultat de l'exécution des composants C4, C5 ou C6. Une fois la liste des besoins établie, la méthode DSDM propose encore une fois de s'interroger sur la viabilité ou la faisabilité du projet avec l'ensemble des besoins tels qu'ils sont définis. Ceci est représenté dans la ligne par le nœud conditionnel D6. Si le projet n'est pas viable, le composant C3 de clôture est alors exécuté et le processus de la ligne s'arrête ensuite. Dans le cas contraire, le processus de la ligne de méthode passe à la partie de planification du projet.

De la même manière que pour la phase de définition des besoins, le nœud de choix D7 propose un choix exclusif entre les trois méthodes pour réaliser la planification du projet. Le composant C7 basé sur la méthode Scrum propose de planifier les "livrables" d'incrément logiciel sans pour autant fixer une répartition des besoins ("besoins utilisateur") sur l'ensemble des "livrables". Le composant C8 issu de XP propose, quant à lui, d'effectuer une estimation plus précise du temps de développement des besoins ("besoins utilisateur") et de répartir les besoins sans les ordonnancer sur un planning de "livrables". Le composant C9 dérivé de la méthode DSDM propose de définir l'ensemble des "livrables" par rapport à la liste des besoins et d'organiser ensuite les "livrables" dans un planning.

Après avoir réalisé le planning du projet, un choix s'offre à l'utilisateur de la ligne de méthode. Le nœud de choix D8 porte une condition sur la volonté de commencer un affinement des besoins de haut niveau du projet après avoir établi une première planification. Cette recommandation est issue de la méthode DSDM. L'utilisateur peut donc, s'il le désire, arrêter là le processus de ligne de méthode et terminer ici l'étape de lancement du projet pour ensuite lancer les cycles d'itération de développement. Au contraire, s'il désire affiner les besoins du projet en amont des itérations de développement, il exécute alors la séquence de composants C10 et C11 issus de DSDM. Le composant C10 permet d'affiner un besoin fonctionnel en divers sous-besoins fonctionnels et de réaliser une ébauche de liste des besoins non fonctionnels. Le composant C11 propose lui de revenir sur la planification du projet pour affecter des acteurs aux "livrables" et de caractériser par un type (modèle fonctionnel, prototypage ou développement) les itérations présentes dans les "livrables". L'utilisateur peut exécuter cette séquence de composants C10 et C11 jusqu'à ce que les niveaux de granularité des besoins et du planning soient satisfaisants (cela est représenté par le nœud d'itération D9). Le processus de la ligne de méthode de lancement d'un projet agile s'arrête à l'issue de cette boucle itérative.

Le processus de cette ligne intègre des composants compatibles entre eux. En effet, ils permettent de satisfaire des objectifs similaires à partir de modèle de produit similaires. Leurs variations se situent au niveau de la démarche de processus proposée pour atteindre ces objectifs.

Cependant, une ligne de méthode ne nécessite pas obligatoirement d'avoir des composants compatibles dans son processus, des guides de configuration peuvent être définis pour expliciter des contraintes d'usage entre composants. De plus, des règles de transformation de produit avancées peuvent être définies dans le modèle de transformation de produits pour chaque composant pour favoriser leur interopérabilité.

6.3.2 Guides de configuration pour la ligne de méthode de lancement de projet agile

Afin de faciliter les décisions des utilisateurs au niveau des différents points de variation d'une ligne de méthode, nous préconisons dans l'approche MaaS d'ajouter au modèle de processus d'une ligne de méthode, un ensemble de guides de configuration. Comme présenté aux sections 5.3.2 et 5.3.4 du chapitre 5 de cette thèse, les guides de configuration sont des heuristiques ou des contraintes attachées à des points de variation de la ligne de méthode. Les guides reposent sur un ensemble de facteurs de contingence attachés au couple variante et point de variations. Ces indicateurs caractérisent les facteurs discriminants entre les variantes d'un point de variation. Ces principes, reportés sur le cas d'application présenté dans ce chapitre, nous incitent à identifier les facteurs de contingence et les guides de configuration pour les nœuds D1, D2, D3 et D7 de la ligne de méthode (voir figure 6.4). Nous avons utilisé les facettes issues du cadre de référence de l'étude comparative des méthodes agiles que nous avons réalisée en 2008 (Iacovelli, 2008) ainsi que le cadre de réutilisation des composants de méthode proposé dans (Mirbel, 2006a) et (Mirbel, 2006b) afin d'identifier les facteurs discriminants de

réutilisation des différentes alternatives. En effet, les produits manipulés par les différents composants étant identiques et leurs processus assez proches, les différences entre les alternatives porteront sur leurs situations de réutilisation dans un contexte agile. Le cadre de réutilisation (Mirbel, 2006a) et (Mirbel, 2006b) regroupe sous la forme d'une ontologie un ensemble de critères concernant les aspects organisationnel, humain et techniques de la réutilisation de composants de méthode. Le cas d'application présenté dans ce chapitre traite de la gestion d'un lancement d'un projet agile. Par conséquent, nous utilisons pour identifier les facteurs de contingence du guide de configuration de la ligne de méthode les parties du cadre concernant la gestion de projet et les facteurs de contingence d'un projet. Ces deux parties sont issues de l'aspect organisationnel du cadre de réutilisation et sont respectivement présentées aux figures 6.5 et 6.6.

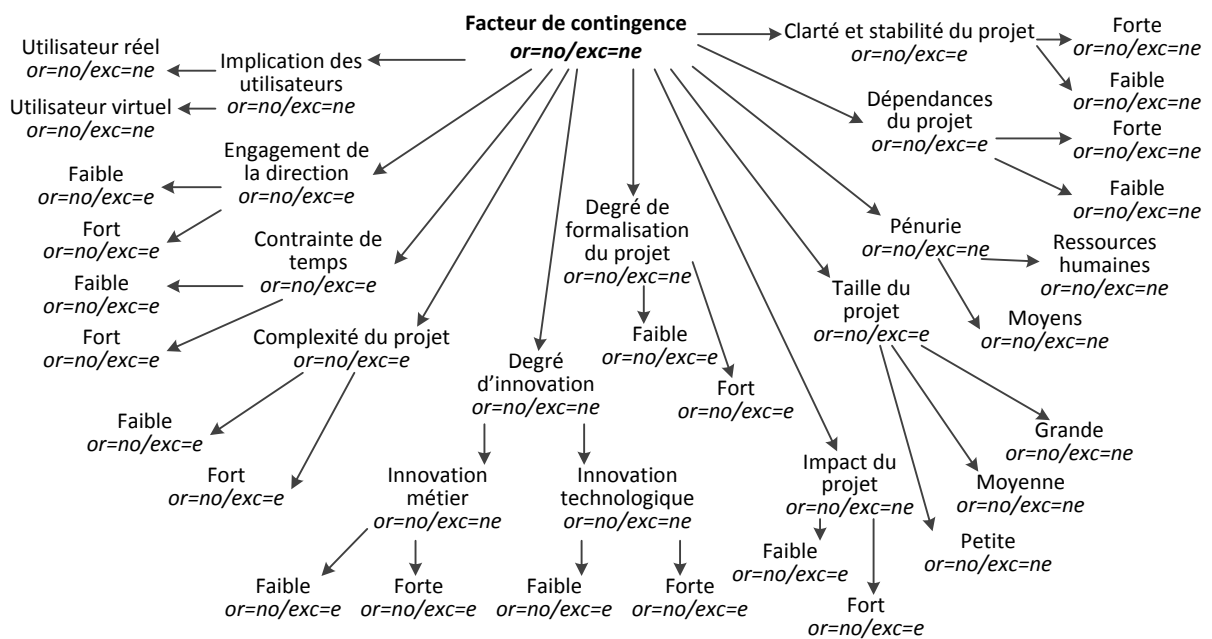


Figure 6.5 : Partie du cadre de réutilisation concernant les facteurs de contingence d'un projet (Mirbel, 2006b)

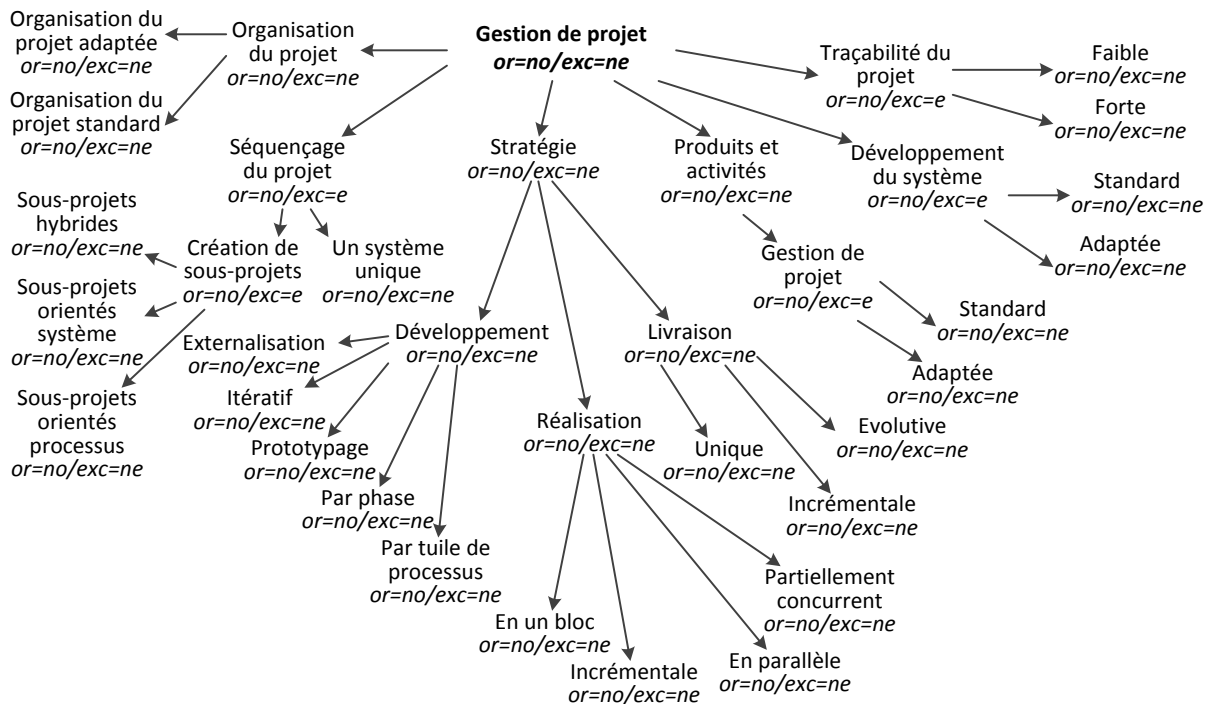


Figure 6.6 : Partie du cadre de réutilisation concernant la gestion d'un projet (Mirbel, 2006b)

Les figures 6.5 et 6.6 présentent sous la forme d'ontologie les critères de réutilisation discriminants pour la réutilisation d'un composant de méthode. Dans cette structure ontologique, les critères de réutilisation sont organisés hiérarchiquement. Les nœuds représentent les familles de critères et les feuilles les différents aspects d'un critère. Chaque famille de critères peut également être caractérisée par un complément d'information concernant les relations avec ses sous-familles ou sous-aspects. En effet, une relation entre une famille et ses sous-familles ou sous-aspects directs peut être ordonnée ou non, c'est-à-dire que l'ordonnement des sous-familles ou des sous-aspects peut être important. Il peut également exister une contrainte d'exclusion entre les sous-familles ou des sous-aspects d'une même famille parente.

Le cadre de réutilisation (Mirbel, 2006a), (Mirbel, 2006b) nous permet d'identifier une sélection de facteurs de contingence parmi l'ensemble des critères de réutilisation qu'il propose (voir figure 6.5 et 6.6) pour les nœuds décisionnels D1, D2, D3 et D7 de la ligne de méthode de ce cas d'application.

Les nœuds D4, D5, D6, D8 et D9 représentent des nœuds conditionnés sur des produits, les guides de configurations qui leurs sont attachés décrivent, en conséquence, les conditions et les contraintes portant sur les produits résultant de l'exécution des composants précédents (Voir l'annexe H pour l'implémentation du modèle de configuration adapté à l'arbre d'exécution de ce cas d'application).

6.3.2.1. Le point de variation D1

Le nœud de choix D1 est de type "ou exclusif", il propose un choix entre deux alternatives D2 et D3 qui sont elles-mêmes des points de variation. Le choix se situe sur l'exécution ou non des étapes d'étude sur la faisabilité du projet avant de passer à la définition des besoins du système à développer.

Pour établir les facteurs de contingence du point de décision D1, nous avons appliqué le cadre de réutilisation (Mirbel, 2006a), (Mirbel, 2006b) présenté aux figures 6.5 et 6.6, ce qui nous a permis de sélectionner les facteurs suivants :

- Taille du projet : ENUM {petite, moyenne, grande}
- Complexité du projet : ENUM {faible, moyenne, forte}
- Degré d'innovation du projet : ENUM {faible, moyen, fort}
- Degré de risques liés au projet : ENUM {faible, moyen, fort}

Le facteur de la taille du projet reflète l'effort à fournir par l'entreprise pour réaliser le projet ainsi que sa durée. La complexité représente le degré de difficulté auquel sera confronté l'équipe du projet dans sa réalisation. Le degré d'innovation permet de représenter la masse de nouvelles solutions conceptuelles ou techniques auxquelles l'équipe n'a jamais été confrontée et qu'elle devra mettre en œuvre dans le projet. Enfin, le degré de risque apparaît comme un regroupement des facteurs d'impact du projet et de clarté et stabilité du projet. Ce facteur représente le niveau de criticité du projet en terme économique pour l'entreprise ou le client.

Appliqués au lien D1.1 (étude de faisabilité), les facteurs ont pour valeur d'affectation :

- Taille du projet: grande
- Complexité du projet: forte
- Degré d'innovation du projet: fort
- Degré de risques liés au projet: faible

Appliqués au lien D1.2 (définition des besoins), les facteurs ont pour valeur d'affectation :

- Taille du projet : petite
- Complexité du projet : faible
- Degré d'innovation du projet: faible
- Degré de risques liés au projet: faible

Les guides de configuration du point de variation D1 sont :

- G1 : Il est recommandé de s'intéresser au type du projet et de choisir la branche D1.1 si c'est un grand projet ou un projet d'une complexité élevée.
- G2 : Il est recommandé de choisir la branche D1.1 si le projet est innovant.

- G3 : Il est recommandé de choisir la branche D1.1 si le projet est risqué en termes de besoins.

6.3.2.2. Le point de variation D2

Le point de variation D2 est de type "ou", il permet de choisir d'exécuter l'une ou les deux alternatives C1 et C2. Ce choix porte sur la réalisation d'une étude de marché ou d'une étude technique de faisabilité du projet.

Pour ces deux alternatives, nous avons sélectionné à partir du cadre de réutilisation les facteurs suivants :

- Degré d'innovation du projet : ENUM { faible, moyen, fort }
- Degré de risques liés au projet : ENUM { faible, moyen, fort }

De la même manière que pour le point de variation D1, le degré de risques liés au projet représente un regroupement des facteurs d'impact du projet et de clarté et stabilité du projet.

Appliqués au lien D2.1 (étude technique), les facteurs ont pour valeur d'affectation :

- Degré d'innovation du projet: fort
- Degré de risques liés au projet : faible

Appliqués au lien D2.2 (étude métier), les facteurs ont pour valeur d'affectation :

- Degré d'innovation du projet: faible
- Degré de risques liés au projet : fort

Les guides de configuration du point de variation D2 sont :

- G4 : Il est recommandé de choisir la branche D2.1 si le projet est innovant.
- G5 : Il est recommandé de choisir la branche D2.2 si le projet est risqué en termes de besoins.

6.3.2.3. Le point de variation D3

Le point de variation D3 est de type "ou exclusif". L'utilisateur devra à cet instant du processus choisir entre les trois alternatives C4, C5 et C6 proposant des méthodes différentes de définir une liste des besoins du client ou des utilisateurs.

Afin d'établir les facteurs du point de variation D3, nous avons utilisé le cadre de réutilisation pour déterminer les facteurs de contingence suivants :

- Taille du projet : ENUM { petite, moyenne, grande }
- Complexité du projet : ENUM { faible, moyenne, forte }
- Degré d'interaction avec les utilisateurs finaux du système : ENUM { faible, moyen, fort }
- Implication du client dans le processus de développement : ENUM { faible, moyenne, forte }

- Coût de mise en œuvre de l'alternative pour l'équipe : ENUM {1, 2, 3, 4, 5, 6} (1 =faible et 6=fort)

Le degré d'interaction est inspiré du facteur d'implication des utilisateurs du cadre de réutilisation avec en supplément l'objectif de mesurer le niveau ou degré d'implication des utilisateurs finaux dans le processus de définition des exigences et les tests du système à développer. Le facteur d'implication du client dans le processus de développement est inspiré par les facteurs : implication des utilisateurs et engagement de la direction. Néanmoins, celui-ci reflète la disponibilité du client dans les itérations de développement pour la priorisation des besoins, leurs définitions détaillées et les tests d'acceptation de chaque itération. Le coût de mise en œuvre représente le degré d'effort et de contrainte requis pour l'application du composant de méthode. Ce facteur correspond au coût humain de l'application d'une alternative. Il n'est pas issu du cadre de réutilisation qui est focalisé sur la caractérisation des projets et non sur l'application des méthodes.

Appliqués au lien D3.1 (méthode Scrum), les facteurs ont pour valeur d'affectation :

- Taille du projet : petite
- Complexité du projet : forte
- Degré d'interaction avec les utilisateurs finaux du système : faible
- Implication du client dans le processus de développement: moyenne
- Coût de mise en œuvre de l'alternative pour l'équipe: 2

Appliqués au lien D3.2 (méthode XP), les facteurs ont pour valeur d'affectation :

- Taille du projet : petite
- Complexité du projet : faible
- Degré d'interaction avec les utilisateurs finaux du système : faible
- Implication du client dans le processus de développement: forte
- Coût de mise en œuvre de l'alternative pour l'équipe: 6

Appliqués au lien D3.3 (méthode DSDM), les facteurs ont pour valeur d'affectation :

- Taille du projet : grande
- Complexité du projet : forte
- Degré d'interaction avec les utilisateurs finaux du système : fort
- Implication du client dans le processus de développement: moyenne
- Coût de mise en œuvre de l'alternative pour l'équipe: 4

Les guides de configuration du point de variation D3 sont :

- G6 : Il est recommandé de choisir la branche D3.1 si le projet est de petite taille mais avec un niveau de complexité important.
- G7 : Choisir la branche D3.1 impliquera d'avoir une forte implication du client pour la définition des besoins. Cependant, la mise en œuvre des interactions entre le client et l'équipe sera peu contraignante.
- G8 : Il est recommandé de choisir la branche D3.2 si le projet est de petite taille et de faible complexité.
- G9 : Choisir la branche D3.2 impliquera d'avoir une forte implication du client pour la définition des besoins et la mise en œuvre des interactions entre le client et l'équipe sera contraignante pour le client.
- G10 : Il est recommandé de choisir la branche D3.3 si le projet est de grande taille et complexe.
- G11 : Choisir la branche D3.3 impliquera d'avoir une forte implication des utilisateurs finaux pour la définition des besoins.

6.3.2.4. Le point de variation D4

Le point de variation D4 est de type "conditionnel", il vise à tester le résultat de l'exécution des composants C1 ou C2. Son objectif est de tester si les études technique et/ou de marché ont déterminé que le projet est valide / faisable. Ce point ne contient donc qu'un seul guide de configuration :

- G12 : Est-ce que le projet à été décrété valide dans les études techniques et de marché?

6.3.2.5. Le point de variation D5

Le point de variation D5 est le nœud "conditionnel" d'une itération. Il a pour objectif de tester si le produit constitué par l'ensemble des besoins du projet est complet. S'il ne l'est pas, l'utilisateur peut alors exécuter à nouveau un cycle de définition des besoins afin de découvrir de nouveaux besoins du client ou des utilisateurs. Dans le cas contraire, le processus de la ligne de méthode continue. Un seul guide de configuration est donc attaché à ce point de variation :

- G13 : Est-ce que la liste des besoins est complète?

6.3.2.6. Le point de variation D6

Le point de variation D6 est de type "conditionnel", il porte sur le produit "liste des besoins". Il a pour but de tester si, au vu de l'ensemble des besoins définis dans la liste des besoins, le projet est faisable. Il ne comporte alors qu'un seul guide de configuration :

- G14 : Au vu de l'ensemble des besoins définis dans la liste des besoins, le projet est-il viable et faisable?

6.3.2.7. *Le point de variation D7*

Le point de variation D7 est de type "ou exclusif", il propose à l'utilisateur de choisir entre trois alternatives C7, C8 et C9 proposant des méthodes différentes de réaliser une première planification générale du projet.

A partir du cadre de réutilisation nous avons retenu les facteurs suivants :

- Taille du projet : ENUM {petite, moyenne, grande}
- Complexité du projet : ENUM {faible, moyenne, forte}
- Type de planification : SET {au produit, au délai, au budget}
- Respect des délais : ENUM {facilement, difficilement}
- Longueur d'un cycle itératif: ENUM {court, long}

Le facteur type de planification est un regroupement construit autour des facteurs organisation du projet et séquençage du projet du cadre de réutilisation. Il permet de décrire la politique utilisée pour planifier les "livrables" du projet en se basant sur les différentes options disponibles dans les méthodes agiles. En effet, la planification peut par exemple être réalisée "au produit", c'est-à-dire que produit est découpé en "livrables" en fonction des besoins. Une estimation du temps de développement de l'ensemble des besoins d'un "livrable" donne sa date de livraison. A l'inverse, une planification "au délai" impose des dates de "livrable" fixes et les besoins sont ensuite répartis sur les "livrables" en fonction de leur priorité et estimations de temps. Une planification "au budget" fixe des coûts pour chaque "livrable" et les besoins sont répartis sur les "livrables" en fonction de leur estimation de coût de développement, une estimation de temps peut ensuite donner les dates des "livrables". Le facteur de respect des délais correspond au facteur contrainte de temps du cadre de réutilisation. Il décrit si la méthode est sensible ou non aux contraintes de temps. Ce facteur caractérise si les délais de livraison du projet sont facilement ou difficilement respectables en choisissant cette alternative.

Étant donné que cette ligne de méthode est construite exclusivement à partir de méthode agile ayant par conséquent des facteurs de réutilisation du cadre développement itératif et réalisation incrémentale, nous avons choisi pour différencier les alternatives de ce point de variation D7 de caractériser la durée de ce cycle itératif et incrémental. Le facteur de longueur d'un cycle itératif représente donc la durée des cycles itératifs de développement du projet: court pour une à deux semaines et long pour trois semaines et au delà.

Appliqués au lien D7.1 (méthode Scrum), les facteurs ont pour valeur d'affectation :

- Taille du projet : petite
- Complexité du projet : forte
- Type de planification : {au produit, au délai, au budget}
- Respect des délais: facilement

- Longueur d'un cycle itératif: court

Appliqués au lien D7.2 (méthode XP), les facteurs ont pour valeur d'affectation :

- Taille du projet: petite
- Complexité du projet : faible
- Type de planification : {au produit}
- Respect des délais: difficilement
- Longueur d'un cycle itératif : court

Appliqués au lien D7.3 (méthode DSDM), les facteurs ont pour valeur d'affectation :

- Taille du projet: grande
- Complexité du projet : forte
- Type de planification : {au produit}
- Respect des délais: facilement
- Longueur d'un cycle itératif: long

Les guides de configuration du point de variation D7 sont :

- G15 : Il est recommandé de choisir la branche D7.1 si le projet est de petite taille mais avec un niveau de complexité important.
- G16 : Il est recommandé de choisir la branche D7.1 si le projet a besoin d'une méthode de planification flexible mais, en contrepartie, les variations de date de livraison seront plus rigides.
- G17 : Il est recommandé de choisir la branche D7.2 si le projet est de petite taille et de faible complexité.
- G18 : Il est recommandé de choisir la branche D7.2 si le projet peut avoir plus de souplesse dans les livraisons des incréments logiciels cependant, en contrepartie, la méthode de planification utilisée est plus rigide.
- G19 : Il est recommandé de choisir la branche D7.3 si le projet est de grande taille et complexe.
- G20 : Il est recommandé de choisir la branche D7.3 si le projet comporte des itérations de développement longues avec peu de flexibilité dans la planification.

6.3.2.8. Le point de variation D8

Le point de variation D8 est de type "conditionnel", son objectif est de tester si la planification et les besoins sont suffisamment détaillés pour l'utilisateur. Si c'est le cas, le processus de la ligne de méthode s'arrête sinon l'utilisateur est orienté vers un cycle d'affinement de la planification et des besoins. Ce point de variation comporte un seul guide de configuration :

- G21 : Est-ce que la planification du projet et la définition des besoins nécessitent un affinement?

6.3.2.9. Le point de variation D9

Le point de variation D5 est le nœud "conditionnel" de l'itération du cycle d'affinement de la planification et des besoins. Un guide de configuration similaire au guide G21 est donc affecté à ce point de variation :

- G22 : Est-ce que la planification du projet et la définition des besoins nécessitent un affinement supplémentaire?

6.4. Détail des composants conceptuels et techniques de la ligne de méthode

Dans le cadre de ce cas d'application, nous avons choisi de représenter les composants de méthode conceptuels de la ligne de méthode avec l'approche de composant chunk (Rolland, 1996), (Rolland, 1998), (Ralyté, 2001b), (Ralyté, 2001c), (Ralyté, 2003), (Mirbel, 2006a). L'approche de composant chunk est expliquée à la section 2.3.2 du chapitre 2 et est positionnée à la section 4.2.3.2 du chapitre 4. Les composants de méthode de type chunk sont décrits à l'aide d'un canevas de description contenant quatre parties : le descripteur avec la situation usage, l'interface, le processus et le produit. Le descripteur de réutilisation est spécifié à l'aide des rubriques objectif, type du composant, situation source, situation cible et la méthode d'origine du composant. La partie situation d'usage est, quant à elle, décrite par le domaine d'application, l'activité de conception et l'intention associée au composant. L'interface est définie par l'intention du composant en précisant la situation sur laquelle s'applique l'intention. Afin de faciliter la lecture des composants, nous avons choisi de décrire le processus de manière informelle et de le représenter par un guideline simple dans le modèle de composant chunk. De la même manière, le produit est formalisé par un diagramme de classe UML. Pour chaque composant, la partie technique qui lui est associé est construite à l'aide de l'approche MaaS que nous présentons dans cette thèse. Nous détaillons ainsi dans cette section les onze composants de méthode intervenant dans la ligne de méthode de lancement de projet agile présentée à la section 6.3 de ce chapitre. (Les descripteurs complets des SMA présentés dans cette section sont disponibles à l'annexe I).

6.4.1 Composant C1 : Etude de marché

Le composant C1 d'étude de marché est issu de l'activité d'étude de marché de la méthode DSDM (voir figure 6.7). Nous avons cependant scindé cette activité de DSDM en deux composants car elle traite à la fois de l'étude de marché d'un projet et de la définition des besoins. Nous avons donc décrit les

aspects relatifs à l'étude de marché dans le composant C1 et ceux relatifs à la définition des besoins dans le composant C6.

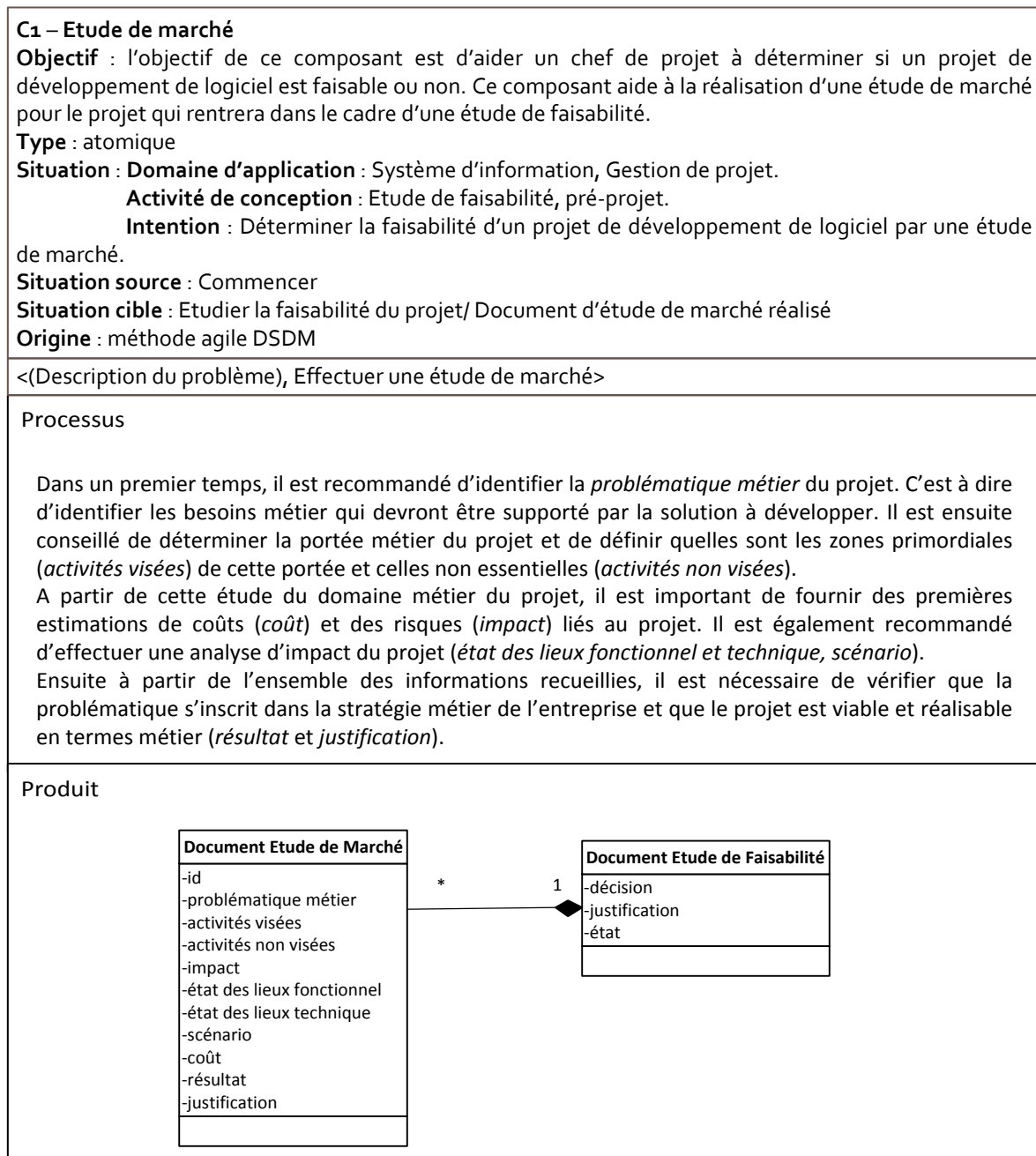


Figure 6.7 : Composant C1 - Etude de marché

A partir du composant de méthode conceptuel d'étude de marché (voir figure 6.7), nous produisons le descripteur du SMA le qui lui est associé (voir figure 6.8). Le descripteur est annoté par la méthode dont est originaire le composant (DSDM) et de l'activité dont il est issu (étude de marché) ainsi que le type générique auquel appartient l'activité dans le cycle de vie du projet (faisabilité du projet).

```

<description>
<interface name="BusinessStudyInterface"

```



```

conceptualMethodPuzzle="http://.../BusinessStudy"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../BusinessStudyPortlet.wsd1"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#DynamicSystemDevelopmentMethod
&agileMethodOntology;#ProjectFesability">
<operation name= "performBusinessStudy"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&agileMethodOntology;#BusinessStudy">
<input element="BusinessStudyRequest"/>
<output element="BusinessStudyResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="BusinessStudy" interface="BusinessStudyInterface"...>
<endpoint ... />
</service>
</description>

```

Figure 6.8 : Descripteur du composant C1 - Etude de marché

L'implémentation de ce composant vise à aider l'utilisateur dans le déroulement de l'étude de marché du projet (voir figure 6.9).

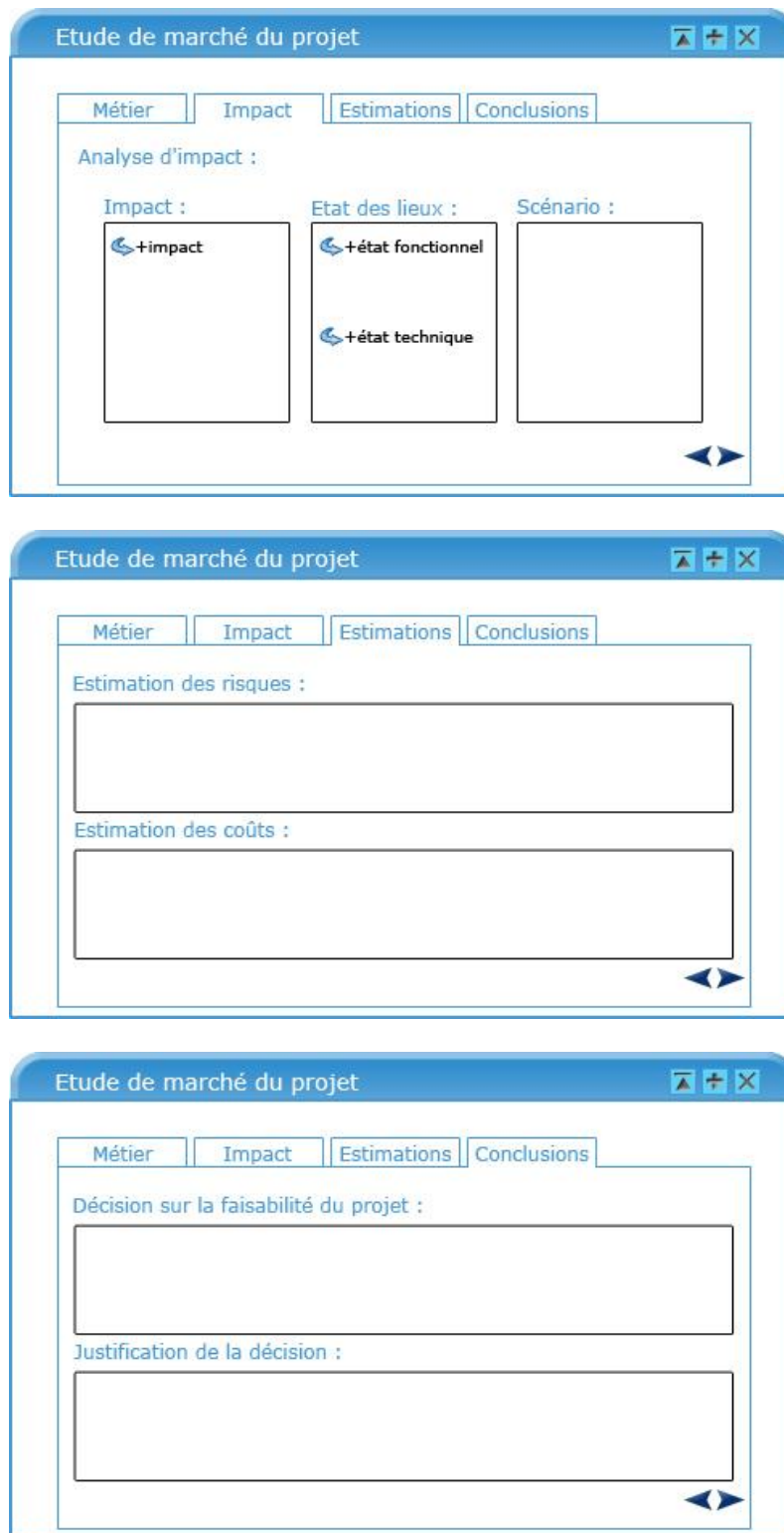


Figure 6.9 : Maquette du composant C1 - Etude de marché

Pour cela, le SMA associé au composant C1 propose une interface structurée pour faciliter la saisie des données de l'étude de marché. En effet, ce SMA assiste les utilisateurs en leur proposant une interface pour saisir la problématique métier et organiser les activités visées par la portée du projet et celles non visées pour former le champ d'application du projet. Sur un deuxième écran les utilisateurs peuvent

ensuite lister les risques liés au projet et faire une estimation du coût de celui-ci. Le SMA aide ensuite les utilisateurs à lister les impacts possibles du projet et, pour chacun d'entre eux, de dresser un état des lieux fonctionnel et technique de l'existant ainsi que le scénario d'impact envisagé. Enfin, le SMA propose aux utilisateurs de clore le dossier d'étude de marché par la saisie de la décision de l'équipe sur la continuité du projet ainsi que sa justification au regard des informations recueillies dans l'étude.

6.4.2 Composant C2 : Etude technique

Les aspects d'étude technique du projet correspondants à l'activité d'étude de faisabilité de la méthode DSDM sont représentés dans le composant C2 (voir figure 6.10). Les aspects relatifs à la planification du projet présents eux aussi dans l'activité d'étude de faisabilité sont traités par le composant C9.

C2 – Etude technique

Objectif : l'objectif de ce composant est d'aider un chef de projet à déterminer si un projet de développement de logiciel est faisable ou non. Ce composant aide à la réalisation d'une étude technique du projet qui rentrera dans le cadre d'une étude de faisabilité.

Type : atomique

Situation : **Domaine d'application** : Système d'information, Gestion de projet.

Activité de conception : Etude de faisabilité, pré-projet.

Intention : Déterminer la faisabilité d'un projet de développement de logiciel d'un point de vue technique par une étude.

Situation source : Commencer

Situation cible : Etudier la faisabilité du projet/Document étude technique réalisé

Origine : méthode agile DSDM

<(Description du problème), Effectuer une étude technique du projet>

Processus

La première étape à entreprendre dans l'étude technique est de vérifier qu'une solution technique (*solution*) au problème métier à traiter dans le projet est réalisable. Cela peut être réalisé par le développement de prototypes de faisabilité destinés à tester chacun un aspect particulier du problème métier et d'effectuer des mesures afin d'avoir une première idée des performances de la solution envisagée. Il est également indiqué de chercher des solutions alternatives possibles pour la réalisation du projet.

Il est conseillé d'établir une *documentation technique des contraintes* techniques et métiers de haut niveau impactant l'architecture matérielle et logicielle envisagée. Il est important de réaliser également une liste des *bénéfices* de la solution choisie. Une identification des processus métier de haut niveau peut être réalisée afin de cerner les parties à automatiser (*automatisation*) dans la solution et d'identifier de manière générale les données qui seront manipulées par celle-ci. Il est aussi recommandé de dresser une liste des interfaces à réaliser pour interfacier la solution avec les systèmes existants.

Ensuite une équipe projet peut être constituée pour développer la solution. Les acteurs (*Acteur*) et leurs rôles (*Rôle*) au sein du projet sont déterminés. Puis vérifier que l'application d'une méthode agile projet correspond bien au développement de la solution et à l'équipe projet.

Enfin, en utilisant conjointement les données sur la composition de l'équipe projet et les données techniques il est possible de donner une première estimation de la durée globale du projet (*estimation du délai*). Puis, une décision (*résultat et justification*) concernant la faisabilité technique du projet peut être prise avec l'ensemble des données recueillies dans cette étape.

Produit

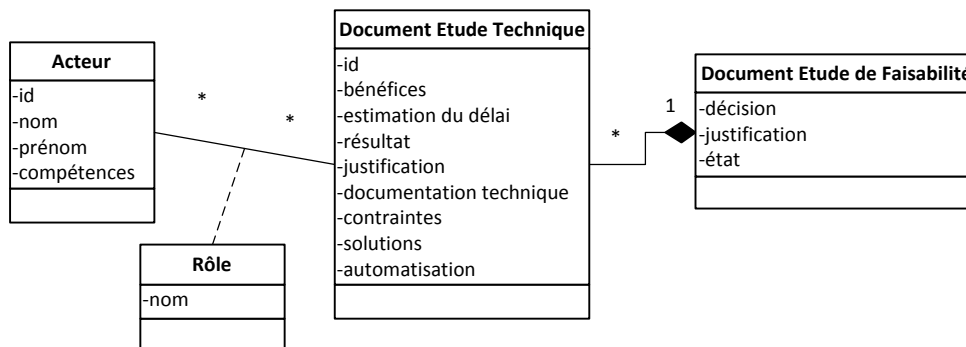


Figure 6.10 : Composant C2 -Etude technique

Le descripteur technique du composant d'étude technique est annoté par la méthode DSDM duquel il est issu ainsi que par l'activité d'étude de faisabilité dont l'étude technique fait partie dans le cycle de vie d'un projet agile (voir figure 6.11).

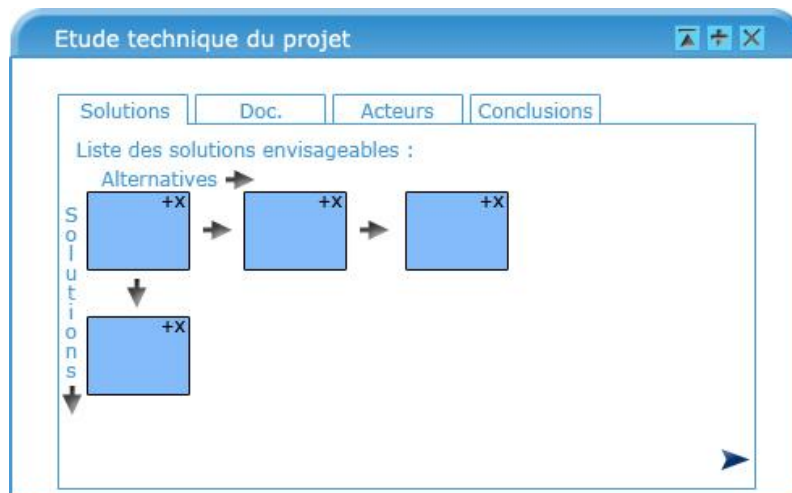
```

<description>
<interface name="TechnicalStudyInterface"
conceptualMethodPuzzle="http://.../TechnicalStudy"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../TechnicalStudyPortlet.wsdl"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#DynamicSystemDevelopmentMethod
&agileMethodOntology;#ProjectFesability">
<operation name= "performTechnicalStudy"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&agileMethodOntology;#TechnicalStudy">
<input element="TechnicalStudyRequest"/>
<output element="TechnicalStudyResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="TechnicalStudy" interface="TechnicalStudyInterface"...>
<endpoint ... />
</service>
</description>

```

Figure 6.11 : Descripteur du composant C2 - Etude technique

D'une manière similaire au composant C1, le SMA implémentant C2 propose une interface structurée facilitant la saisie des données concernant l'étude technique du projet à l'utilisateur (voir figure 6.12).



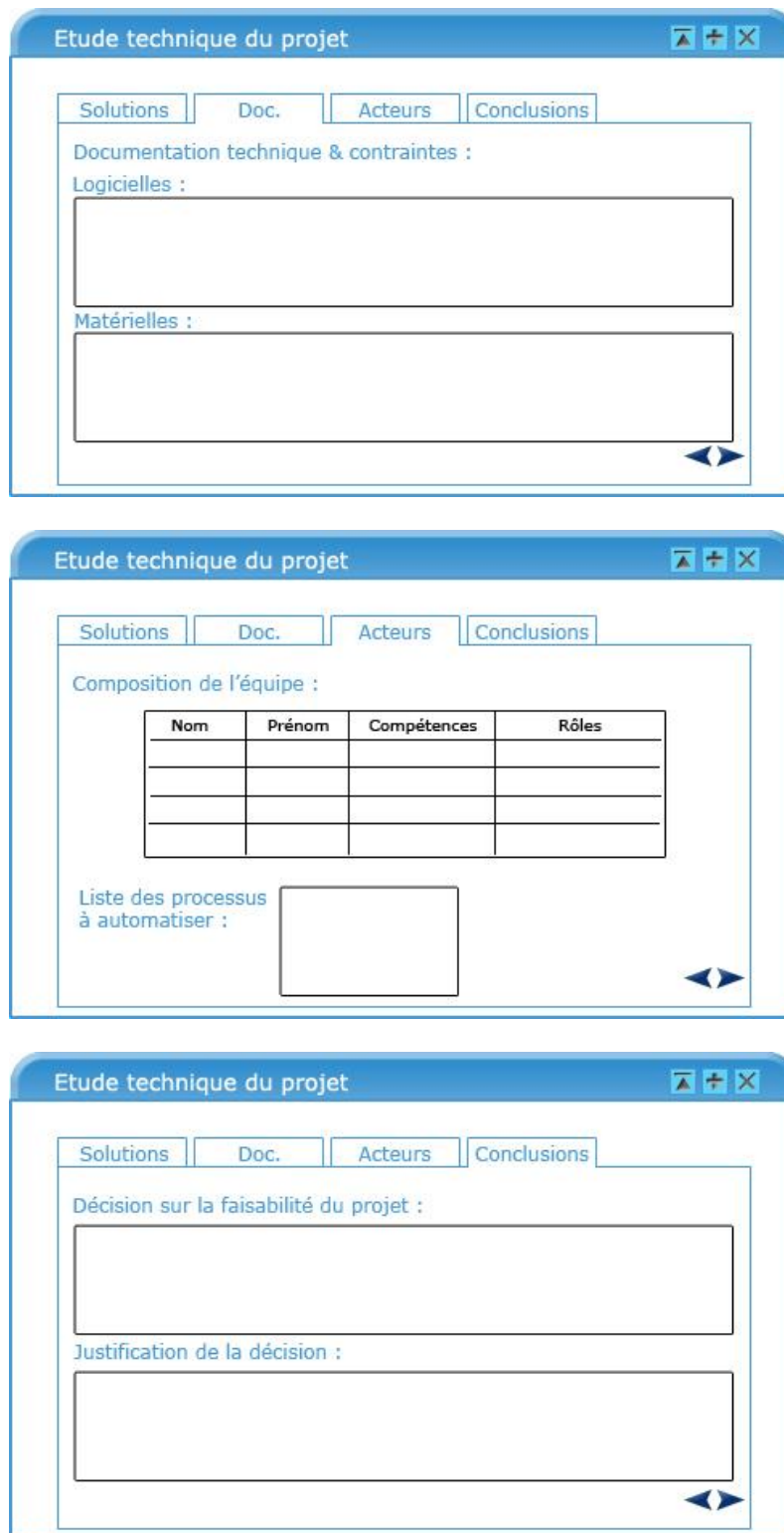


Figure 6.12 : Maquette du composant C2 - Etude technique

Ce SMA guide également l'utilisateur dans l'application de la démarche de l'étude technique. En effet, il propose aux utilisateurs une interface pour la saisie de la liste des solutions techniques envisageables avec leurs alternatives. Afin de faciliter la saisie des données relatives aux solutions, il est possible pour chacune d'elles de saisir directement ses caractéristiques attendues du prototype ainsi que

les tests à effectuer sur ce dernier, les résultats des tests et enfin les bénéfices à retirer de cette solution. Le SMA guide ensuite les utilisateurs vers l'interface de saisie des contraintes techniques de l'architecture logicielle et matérielle vis-à-vis de la solution envisagée. Dans un troisième temps, les utilisateurs peuvent saisir les processus métiers à automatiser dans la solution et être assisté par le SMA pour constituer l'équipe du projet. Enfin, un dernier écran est proposé aux utilisateurs pour saisir les résultats de l'étude technique et la décision sur la continuité du projet ainsi que les justifications relatives à cette décision.

6.4.3 Composant C3 : Abandon du projet

Le composant C3 représente l'activité de clôture d'un projet par abandon (voir figure 6.13). Cette clôture peut avoir des causes diverses, soit parce que le projet n'est pas faisable d'un point de vue métier ou technique, ou encore à l'issue de la définition des besoins car il n'est pas viable.

C3 – Abandon du projet

Objectif : l'objectif de ce composant est de prendre acte de l'arrêt du projet.

Type : atomique

Situation : **Domaine d'application** : Système d'information, Gestion de projet.

Activité de conception : Etude de faisabilité, Définition des besoins

Intention : Clôturer le dossier projet

Situation source : Document de faisabilité

Situation cible : Décision sur la faisabilité du projet

Origine : méthode agile DSDM

<(Document de faisabilité), Clôturer le projet>

Processus

Clôre le dossier projet car le projet n'est pas réalisable soit :

- D'un point de vue du marché : l'étude de marché a donné (*Document Etude de Marché*) un résultat négatif car par exemple : la problématique métier n'est pas réalisable dans ce projet ou la portée du projet est trop grande/faible. La problématique du projet ne s'inscrit pas dans la stratégie de l'entreprise. Le projet est trop cher ou trop risqué.
- D'un point de vue technique : l'étude technique (*Document Etude Technique*) a donné un résultat négatif car par exemple : il n'existe pas de solution viable au problème métier, trop de contraintes pèsent sur l'architecture matérielle et logicielle à mettre en place, l'architecture peut également être trop complexe ou non viable ou nécessiter trop interfaces avec les systèmes existants, des ressources humaines insuffisantes pour constituer une équipe de développement, une méthode agile n'est pas applicable à ce projet, ou encore une estimation de délai de réalisation du projet trop longue.
- D'un point de vue des exigences du client ou des utilisateurs : la définition des besoins (*Besoin*) a montré que le projet n'était pas faisable car par exemple : il n'est pas possible de définir clairement les besoins du client, trop de besoins sont mal définis ou erronés voire contradictoires, il n'est pas possible de cerner l'ensemble des besoins de manière complète, les besoins ne sont pas réalisables, il n'est pas possible d'établir clairement une priorisation des besoins.

Produit

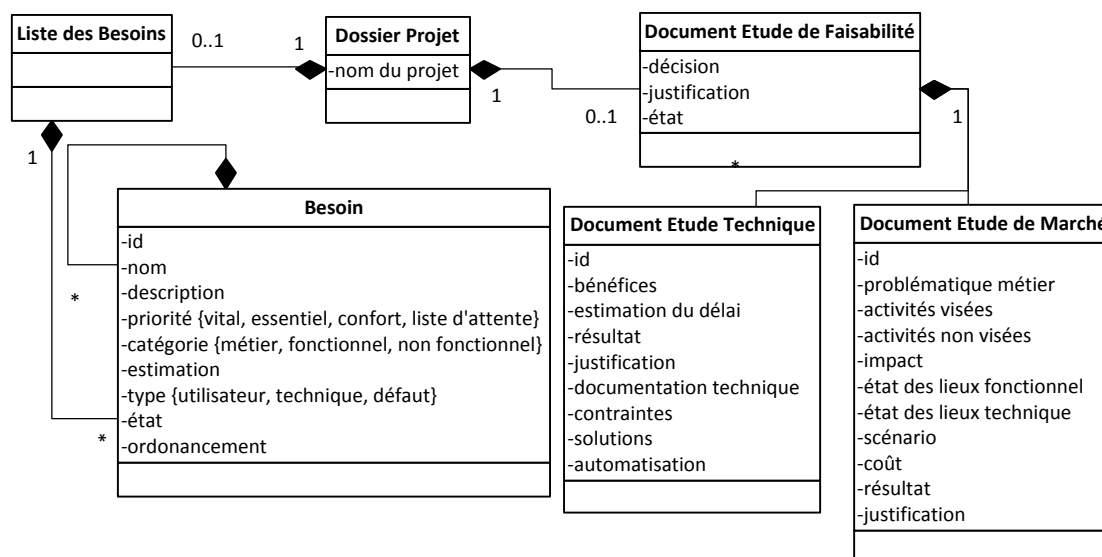


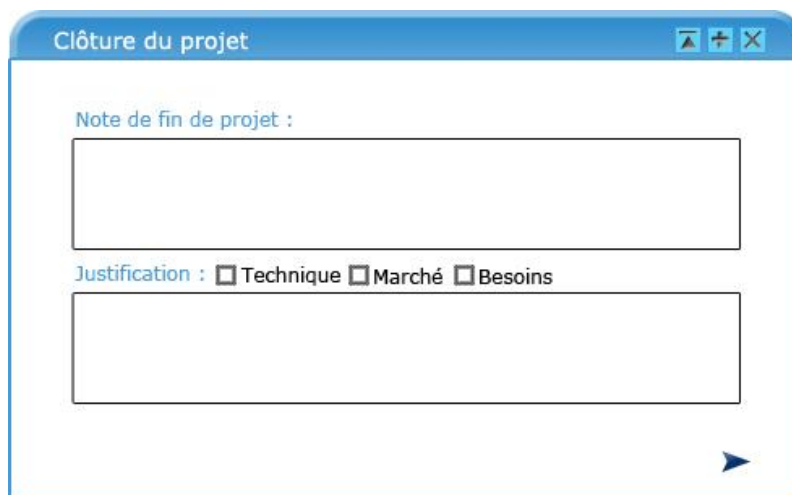
Figure 6.13 : Composant C3 - Abandon du projet

Le composant C3 étant issu de la méthode DSDM, cette information est présente dans le descripteur du SMA correspondant. Il est également indiqué que ce composant est utilisé dans l'étape de finalisation d'un projet (voir figure 6.14).

```
<description>
<interface name="ProjectCloseOutInterface"
conceptualMethodPuzzle="http://.../ProjectCloseOut"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../ProjectCloseOutPortlet.wsd1"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#DynamicSystemDevelopmentMethod
&agileMethodOntology;#ProjectCloseOut">
<operation name= "performProjectCloseOut"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&agileMethodOntology;#ProjectCloseOut">
<input element="ProjectCloseOutRequest"/>
<output element="ProjectCloseOutResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="ProjectCloseOut" interface="ProjectCloseOutInterface"...>
<endpoint ... />
</service>
</description>
```

Figure 6.14 : Descripteur du composant C3 - Abandon du projet

L'implémentation de ce composant par un SMA a pour objectif de faciliter la clôture du projet à un utilisateur (voir la figure suivante).



Clôture du projet

Note de fin de projet :

Justification : Technique Marché Besoins

Maquette d'un portlet web intitulé "Clôture du projet". Le portlet contient un champ de saisie pour la "Note de fin de projet" et une section "Justification" avec trois cases à cocher : "Technique", "Marché" et "Besoins".

Figure 6.15 : Maquette du composant C3 - Abandon du projet

Le composant C3 est outillé de manière simple par un SMA aidant l'utilisateur à rédiger la note de fin de projet comprenant la décision d'abandon du projet ainsi que la justification de cette décision (voir figure 6.15).

6.4.4 Composant C4 : Définition de la liste des besoins du produit

Le composant C4 de définition de la "liste des besoins du produit" (*Liste des Besoins*) est issu de la méthode agile Scrum. Il regroupe les bonnes pratiques de la méthode Scrum concernant la définition des besoins du client sous la forme de "besoin utilisateur" (*Besoin*) et leurs estimation de temps de développement suivant les règles du "poker de planification" (voir figure 6.16).

C4 – Définition de la liste des besoins du produit

Objectif : l'objectif de ce composant est d'aider un chef de projet et le client à définir les besoins vis-à-vis du système à développer sous la forme de besoins utilisateur. Puis de prioriser l'ensemble des besoins utilisateur dans une liste ordonnée appelée liste des besoins du produit.

Type : atomique

Situation : **Domaine d'application** : Système d'information, Gestion de projet.

Activité de conception : Définition des besoins.

Intention : Déterminer une liste de besoins utilisateur et la prioriser dans la liste des besoins du produit.

Situation source : Rien.

Situation cible : Liste de besoins utilisateur priorisées appelée liste des besoins du produit.

Origine : méthode agile Scrum

<(Description du problème), Définir la liste des besoins du produit>

Processus

La première activité de cette étape consiste pour l'équipe du projet à identifier avec l'aide du client un ensemble de besoins utilisateur (*Besoin*) capturant chacun un besoin de haut niveau du client vis-à-vis de la solution à réaliser. Pour chaque besoin utilisateur son nom, son identifiant et la description du besoin qu'il caractérise doivent être renseignés. L'état d'un besoin lorsqu'il est créé est défini par défaut à « à faire ». Il existe trois types de besoins utilisateur : les "besoins utilisateur" capturant un comportement du système attendu par les utilisateurs, les "besoins techniques" décrivant des besoins internes à caractère technique du système (elles sont connues de l'équipe de développement mais elles sont invisibles pour les utilisateurs), les "besoins défaut" servent à répertorier les anomalies du système à traiter (bug ou évolution).

Les besoins sont ensuite priorisés en fonction de cinq critères qui sont susceptibles d'évoluer au cours du projet et, par conséquent, de modifier leur priorité. La priorisation des besoins est une étape importante car leur ordonnancement a un impact sur le contenu des différents incrément livrés au client. Les critères à prendre en compte dans cette priorisation sont : le risque que l'implémentation d'un besoin permet de réduire avec un retour rapide du client, l'incertitude vis-à-vis des besoins qu'une implémentation d'un besoin utilisateur permet de diminuer, le niveau de qualité de la solution auquel un besoin contribue et enfin la dépendance de développement entre plusieurs besoins.

A l'issue de la priorisation, il est conseillé d'effectuer une première estimation du temps de développement de chaque besoin avec la méthode du poker de planification. Chaque membre de l'équipe possède un jeu de cartes sur lesquelles sont indiquées des valeurs possible d'estimation. Pour chaque besoin, après un débat pour clarifier l'objectif et le contenu, chaque membre présente simultanément la carte d'estimation qu'il aura choisie. Les membres de l'équipe débattent ensuite si des divergences d'estimation sont présentes jusqu'à converger vers une estimation commune. Les valeurs d'estimation généralement utilisées sont le début de la suite de Fibonacci complétée par d'autres valeurs. Ce qui donne cet ensemble : {0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, ?}.

L'ensemble des besoin utilisateur d'un projet forment la liste des besoins du produit (*Liste des besoins*) : une liste de besoins priorisées. Cette liste n'est pas figée, elle évolue au cours du projet. Il est possible de modifier, ajouter, supprimer des besoins et leurs priorités peuvent être ajustées.

Produit

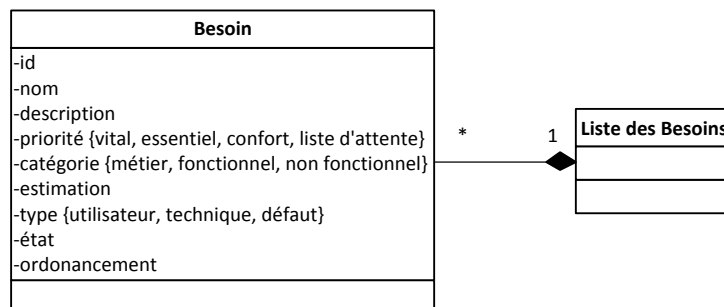


Figure 6.16 : Composant C4 - Définition de la liste des besoins du produit

Le SMA associé à ce composant conceptuel C4 est représenté par le descripteur présenté à la figure 6.17. Ce descripteur a été annoté afin d'apporter les précisions sémantiques suivantes : le composant est issu de la méthode Scrum, le composant traite de l'étape de définition des besoins du cycle de vie d'une méthode agile et son processus correspond à l'activité de définition de la "liste des besoins du produit" dans la méthode Scrum.

```
<description>
<interface name="BacklogDefinitionInterface"
conceptualMethodPuzzle="http://.../BacklogDefinition"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../BacklogDefinitionPortlet.wsdl"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#Scrum
&agileMethodOntology;#RequirementsDefinition">
<operation name= "performBacklogDefinition"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&agileMethodOntology;#BacklogDefinition">
<input element="BacklogDefinitionRequest"/>
<output element="BacklogDefinitionResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="BacklogDefinition" interface="BacklogDefinitionInterface"...>
<endpoint ... />
</service>
</description>
```

Figure 6.17 : Descripteur du composant C4 - Définition de la liste des besoins du produit

L'implémentation de ce composant aide les membres de l'équipe à constituer la "liste des besoins du produit" (*Liste des Besoins*) en leur proposant une interface pour saisir les données relatives aux "besoins utilisateur" (*Besoin*) (voir figure 6.18).

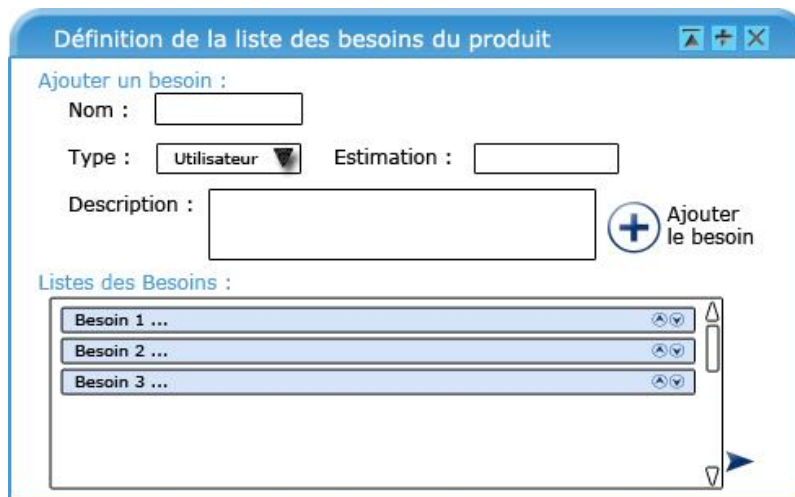


Figure 6.18 : Maquette du composant C4 - Définition de la liste des besoins du produit

Le SMA permet ensuite de faciliter la priorisation des besoins en proposant des fonctionnalités d'ordonnement de la liste des "besoins utilisateur" de la "liste des besoins du produit". Il permet également de rappeler aux utilisateurs les règles du "poker de planification" et de saisir pour chaque "besoin utilisateur" une estimation de son temps de développement.

6.4.5 Composant C5 : Définition de besoin utilisateur

L'activité de définition des "besoins utilisateur" (Besoin) dans la méthode XP est décrite dans le composant C5 (voir figure 6.19). Ce composant a pour objectif d'aider l'équipe de développement et au client pour identifier les besoins du client et les décrire sous la forme de "besoin utilisateur". De plus, ce composant traite de l'affinement ou du regroupement de "besoin utilisateur" en fonction d'une première estimation simple de leur temps de développement.

C5 – Définition de besoin utilisateur
Objectif : l'objectif de ce composant est d'aider un chef de projet et le client à définir les besoins vis-à-vis du système à développer sous la forme de besoins utilisateur. Puis de réaliser une première estimation du temps de développement de chaque besoin.
Type : atomique
Situation : **Domaine d'application** : Système d'information, Gestion de projet.
Activité de conception : Définition des besoins.
Intention : Définir un ensemble de besoin utilisateur
Situation source : rien.
Situation cible : Ensemble de besoins utilisateur
Origine : méthode agile XP

<(Description du problème), Définir un ensemble de besoin utilisateur>

Processus

La définition des besoins est réalisée par l'équipe du projet en collaboration avec le client. Les besoins du client sont capturés sous la forme de besoins utilisateur (*Besoin*). Ceux-ci sont similaires à des cas d'utilisations ayant néanmoins un objectif différent. Les besoins utilisateur ressemblent également aux scénarios d'usage, à l'exception qu'ils ne sont pas limités à la description de l'interface entre l'utilisateur et le système. Un besoin utilisateur est constitué de trois phrases écrites dans le langage du client (sans termes techniques) et explicitant un de ses besoins. L'ensemble des besoins utilisateur doit capturer les besoins de haut niveau du client. Ils seront affinés ultérieurement au début des itérations. Un besoin utilisateur doit cependant inclure suffisamment de détails pour que l'équipe puisse estimer la durée de son développement avec le minimum de risques.

Une fois écrit, le temps de développement de chaque besoin utilisateur est estimé par l'équipe de développement. Si un besoin est estimé à plus de trois semaines de développement, il doit alors être décomposé en plusieurs besoins plus simple. Au contraire, si un besoin est estimé à moins de une semaine de développement, il doit être regroupé avec d'autres besoins utilisateur pour en former un plus conséquent.

Il est conseillé que la liste des besoins utilisateur compte en moyenne entre 60 et 100 éléments pour pouvoir créer un planning des livrables. Cela varie bien entendu en fonction des projets.

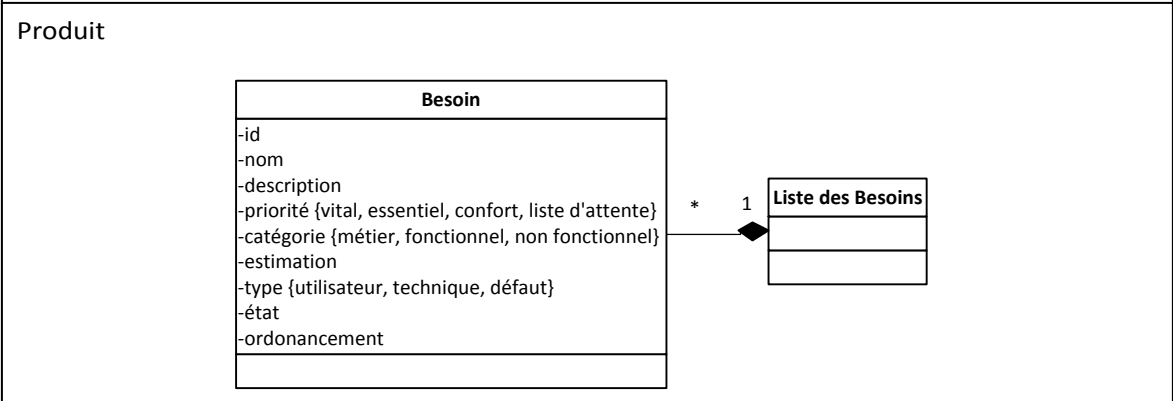


Figure 6.19 : Composant C5 - Définition de besoin utilisateur

L'implémentation du composant C5 sous la forme d'un SMA est décrite par le descripteur présenté à la figure 6.20. Dans ce descripteur, il est annoté que la méthode XP est celle d'origine du composant, qu'il concerne l'activité de définition des besoins dans le cycle de vie du projet et qu'il traite plus précisément de l'activité de définition des "besoins utilisateur" de la méthode XP.

```

<description>
<interface name="UserStoriesDefinitionInterface"
conceptualMethodPuzzle="http://.../UserStoriesDefinition"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../UserStoriesDefinitionPortlet.wsd1"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#ExtremeProgramming&agileMethodOntology;#Requi
rementsDefinition">
<operation name= "performBusinessStudy"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&agileMethodOntology;#UserStoriesDefinition">
<input element="UserStoriesDefinitionRequest"/>
<output element="UserStoriesDefinitionResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="UserStoriesDefinition"
interface="UserStoriesDefinitionInterface"...>
<endpoint ... />
</service>
</description>

```

Figure 6.20 : Descripteur du composant C5 - Définition de besoin utilisateur

Le SMA associé au composant C5 aide l'équipe du projet et le client à formuler la liste des besoins à implémenter dans la solution (voir figure 6.21).

Figure 6.21 : Maquette du composant C5 - Définition de besoin utilisateur

Ce SMA propose une interface de saisie des "besoins utilisateur" et de leur estimation du temps de développement. Un mécanisme de vérification est implémenté pour les estimations et propose aux utilisateurs de décomposer les "besoins utilisateur" trop grands en plusieurs "besoins utilisateur" ou de rassembler plusieurs "besoins utilisateur" trop petits en un "besoin utilisateur" plus conséquente. Un

second mécanisme de vérification est implémenté pour la validation de la liste complète des besoins, ce mécanisme vérifie que le nombre de "besoins utilisateur" saisis soit autour de 80 (plus ou moins 20) et affiche une recommandation aux utilisateurs si ce n'est pas le cas.

6.4.6 Composant C6 : Définition des besoins de haut niveau

Comme présenté à la figure 6.22, le composant C6 de définition des besoins de haut niveau est issu de la méthode DSDM. Il regroupe les bonnes pratiques de la méthode concernant la définition des besoins des utilisateurs finaux de la solution (*Besoin*) à partir de la problématique métier identifiée dans l'activité d'étude de marché. Cela inclut également la priorisation des besoins une fois qu'ils ont été identifiés.

C6 – Définition des besoins de haut niveau

Objectif : l'objectif de ce composant est d'aider un chef de projet et les utilisateurs à définir les besoins de haut niveau vis-à-vis du système à développer. Puis de prioriser l'ensemble des besoins en collaboration avec les utilisateurs.

Type : atomique

Situation : **Domaine d'application :** Système d'information, Gestion de projet.

Activité de conception : Définition des besoins.

Intention : Définir un ensemble de besoins de haut niveau ordonnancés.

Situation source : rien.

Situation cible : Besoins identifiés et priorisés

Origine : méthode agile DSDM

<(Description du problème), Définir un ensemble de besoins de haut niveau ordonnancés>

Processus

La définition des besoins du logiciel à développer (*Besoin*) dans le projet est réalisée par l'équipe du projet en collaboration avec les futurs utilisateurs finaux du système. Les besoins sont définis à un haut niveau d'abstraction dans un premier temps. Ils seront affinés ultérieurement dans le lancement du projet ou au début de chaque itération. L'ensemble de ces besoins doit répondre à la problématique métier que doit traiter le projet. Les besoins sont d'abords définis en termes de besoins métiers puis décomposés en besoins vis-à-vis des fonctionnalités du système à développer.

La priorisation des besoins est ensuite effectuée avec les utilisateurs en suivant les principes de la priorisation MoSCoW. Cela signifie que les besoins sont classés en cinq catégories de priorités (de la plus à la moins importante): *vital, essentiel, confort, liste d'attente*. La catégorie « vital » regroupe les besoins primordiaux sans quoi le système ne peut fonctionner ou sera inutile. La catégorie « essentiel » regroupe les besoins importants pour le système à forte valeur ajoutée pour lesquels il est possible d'avoir un retour rapide des utilisateurs. La catégorie « confort » regroupe les besoins à faible valeur ajoutée pour le système qui peuvent facilement être reportés plus tard dans le développement dans un autre incrément du système. La catégorie « liste d'attente » regroupe les besoins désirés mais qui ne seront pas implémentés dans cet incrément du système. Ils sont placés en liste d'attente pour être réalisés plus tard dans le développement.

Produit

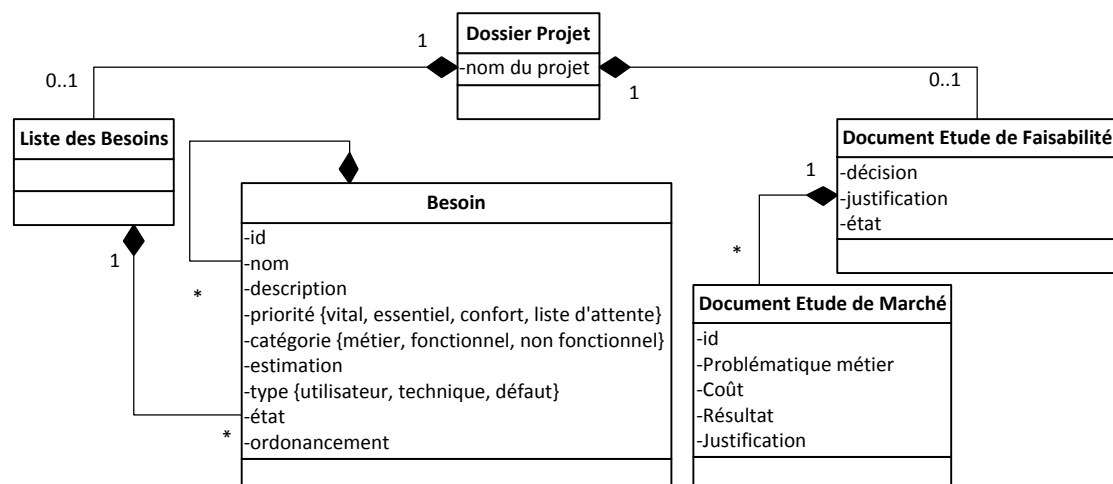


Figure 6.22 : Composant C6 - Définition des besoins de haut niveau

Le descripteur du composant technique de C6 (voir figure 6.23) comporte les précisions sémantiques suivantes : le fait que C6 soit issu de la méthode DSDM et qu'il concerne l'activité de définition des besoins appelée définition des besoins de haut niveau dans DSDM.

```

<description>
<interface name="HighLevelRequirementsDefinitionInterface"
conceptualMethodPuzzle="http://.../HighLevelRequirementsDefinition"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../HighLevelRequirementsDefinitionPortlet.wsd1"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#DynamicSystemDevelopmentMethod
&agileMethodOntology;#RequirementsDefinition">
<operation name= "performHighLevelRequirementsDefinition"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&agileMethodOntology;#HighLevelRequirementsDefinition">
<input element="HighLevelRequirementsDefinitionRequest"/>
<output element="HighLevelRequirementsDefinitionResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="HighLevelRequirementsDefinition"
interface="HighLevelRequirementsDefinitionInterface"...>
<endpoint ... />
</service>
</description>

```

Figure 6.23 : Descripteur du composant C6 - Définition des besoins de haut niveau

L'implémentation de ce composant sous la forme d'un SMA vise à faciliter l'activité de définition des besoins de haut niveau dans un projet agile (voir figure 6.24).

Figure 6.24 : Maquette du composant C6 - Définition des besoins de haut niveau

Ce SMA fournit aux utilisateurs une interface de saisie des besoins des utilisateurs finaux et du client de la solution à développer. Le SMA guide les utilisateurs en leur proposant tout d'abord de saisir les besoins métier puis de naviguer parmi ceux-ci pour les décomposer en besoins fonctionnels. Une fois la liste des besoins complétée, le SMA propose aux utilisateurs de déterminer la priorité de chaque besoin et d'ordonner la liste en fonction de la priorité des besoins.

6.4.7 Composant C7 : Planification générale des livrables

Le composant C7 de planification générale des "livrables" est issu de l'activité de planification du même nom de la méthode Scrum (voir figure 6.25). Il s'agit de la première étape de planification du projet (*Planning des Livrables*). C'est une étape de planification de haut niveau visant en priorité à organiser les "livrables" ou jalons (*Livable*) du projet avec leurs objectifs d'une manière générale. Le contenu exact des "livrables" sera déterminé au fur et à mesure de l'exécution des itérations de développement du projet.

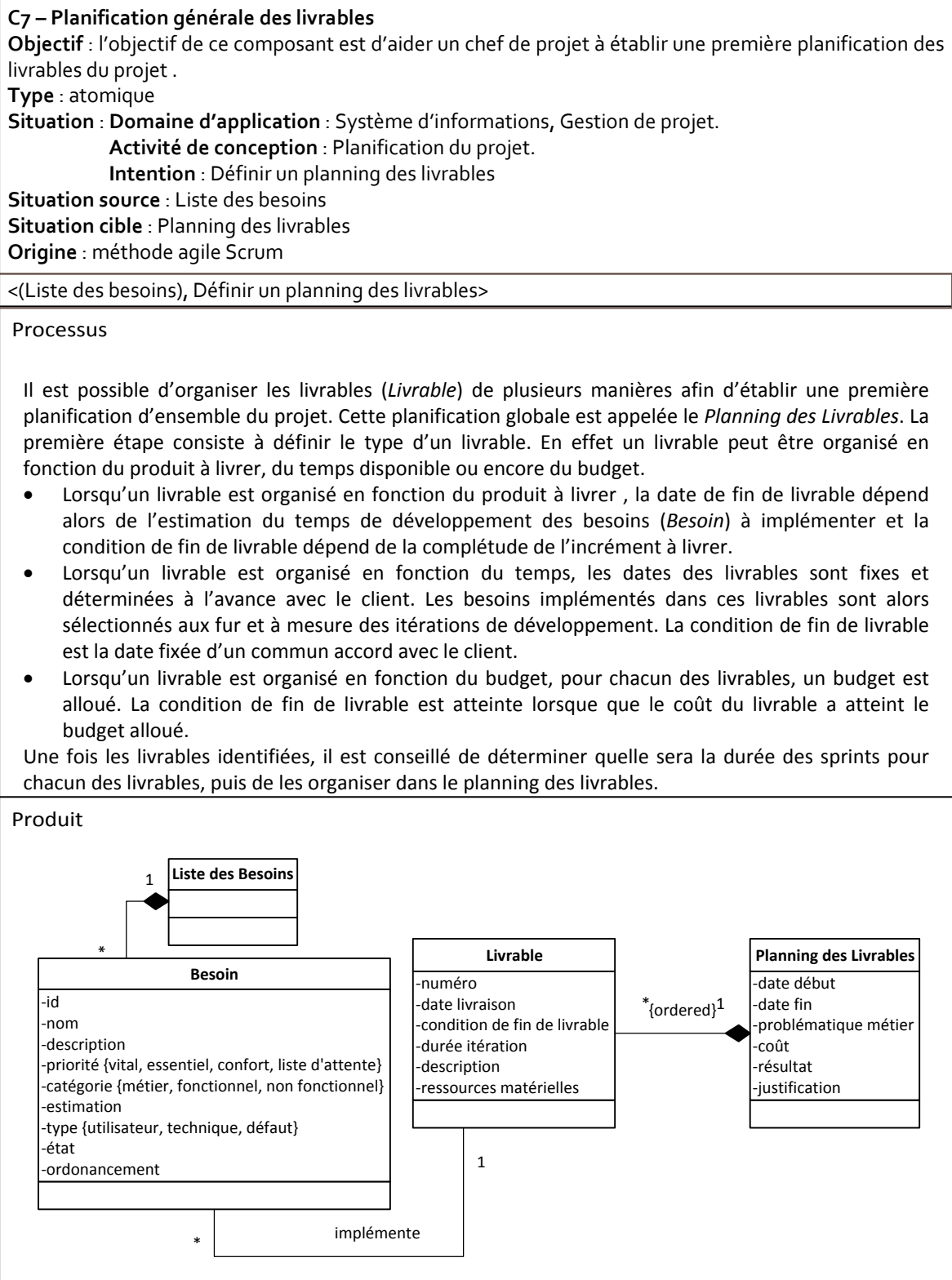


Figure 6.25 : Composant C7– Planification générale des livrables

Le descripteur du composant technique lié au composant conceptuel C7 est annoté par la méthode Scrum dont il est issu ainsi que par l'activité de planification des "livrables" du cycle de vie d'une méthode agile (voir figure 6.26).

```

<description>
<interface name="ReleasePlanning"
conceptualMethodPuzzle="http://.../ReleasePlanning"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../ReleasePlanningPortlet.wsd1"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#Scrum &agileMethodOntology;#ReleasePlanning">
<operation name= "performReleasePlanning"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&agileMethodOntology;#ReleasePlanning">
<input element="ReleasePlanningRequest"/>
<output element="ReleasePlanningResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="ReleasePlanning" interface="ReleasePlanningInterface"...>
<endpoint ... />
</service>
</description>

```

Figure 6.26 : Descripteur du composant C7 – Planification générale des livrables

Le SMA outillant le composant conceptuel C7 a pour objectif de faciliter pour les utilisateurs l'activité de planification des "livrables" du projet. Ce SMA fournit en conséquence aux utilisateurs une interface leur permettant de découper la ligne temporelle du projet en "livrables", puis de saisir directement dans le bloc graphique représentant un "livrable" les informations relatives à celle-ci, son type et sa condition de fin (voir figure 6.27).

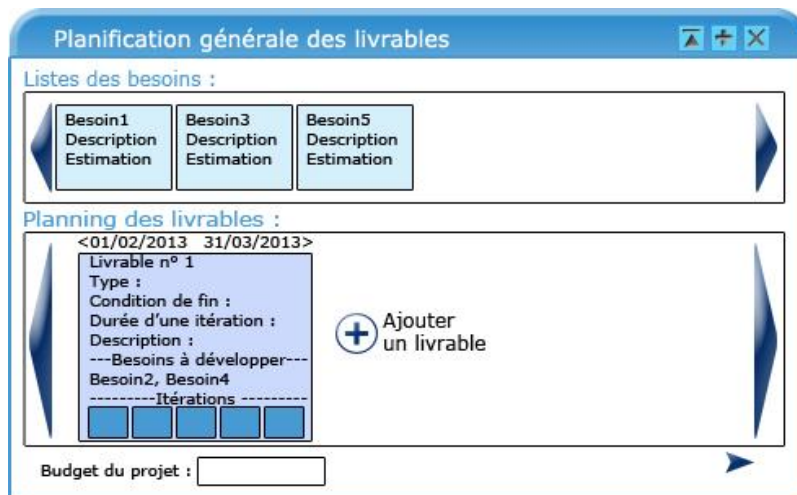


Figure 6.27 : Maquette du composant C7 – Planification générale des livrables

Pour chaque "livrable", les utilisateurs peuvent également préciser la durée des "sprints" (itérations de développement) la constituant et visualiser sur l'ensemble du projet son découpage en "livrables" et en itérations.

6.4.8 Composant C8 : Définition du plan général des livrables

La définition du plan général des "livrables" est une activité présente dans la méthode XP que nous avons intégrée dans le composant C8 de la ligne de méthode de ce cas d'application. Ce composant regroupe les bonnes pratiques de la méthode XP pour aider l'équipe du projet et le client à établir une première planification générale du projet afin de définir et d'organiser les "livrables" (*Planning des Livrables*). En d'autres termes, il s'agit d'identifier les "livrables" (*Livrables*) en rassemblant les "besoins utilisateur" (*Besoin*) pour former des incréments logiciels à implémenter puis, en fonction des estimations de ces derniers, à déterminer les dates des "livrables" (voir figure 6.28).

C8 – Définition du plan général des livrables

Objectif : l'objectif de ce composant est d'aider un chef de projet à estimer le temps de développement des besoins utilisateur puis d'établir une première planification des livrables.

Type : atomique

Situation : **Domaine d'application :** Système d'information, Gestion de projet.

Activité de conception : Planification du projet

Intention : Définir un planning des livrables.

Situation source : Liste des besoins

Situation cible : *Livrables* identifiées et ordonnées dans un planning des livrables.

Origine : méthode agile XP

<(Liste des besoins), Définir un planning des livrables>

Processus

Afin de faciliter la création du *Planning des Livrables*, il est conseillé d'écrire chaque besoin utilisateur (ou *Besoin*) présent dans la liste des besoins sur une carte. Ces cartes pourront ainsi être facilement déplacées sur une table lors des réunions de planification du projet avec l'équipe du projet et le client. Il est ensuite recommandé d'effectuer une estimation précise de l'effort en temps de développement de chaque besoin. L'unité de ces estimations est le temps de développement idéal (le temps « idéal » en semaine que l'équipe du projet mettra pour développer un besoin utilisateur). Une semaine idéale est une semaine ou le temps de travail (sans heures supplémentaires) serait exclusivement consacré au développement et au test du besoin utilisateur sans aucune autre activité annexe. Il est important de ne pas sous-estimer le temps de développement des besoins car cela causerait des problèmes ultérieurement lors des ajustements des livrables en fonction de la productivité.

L'équipe et le client regroupent ensuite les besoins utilisateur à l'aide des cartes sur une table pour former des *Livrables* (incrément logiciel fonctionnel qui seront livrés au client). Chaque groupe de besoins formera ainsi un livrable, les besoins ne sont cependant pas ordonnés à l'intérieur de ceux-ci. En effet, au début de chaque itération de développement, un ou plusieurs besoins restant à implémenter dans le livrable courant seront sélectionnés pour être implémentés. Il est également recommandé au client de sélectionner pour les premiers livrables les besoins utilisateur primordiaux au fonctionnement du système et ayant le plus de valeur ajoutée pour le domaine métier de la solution. Il est important d'avoir des retours le plus tôt possible dans le projet sur les fonctionnalités primordiales à la solution.

Il est enfin aisé de calculer les dates des livrables en effectuant la somme des estimations des besoins de chaque livrable.

Produit

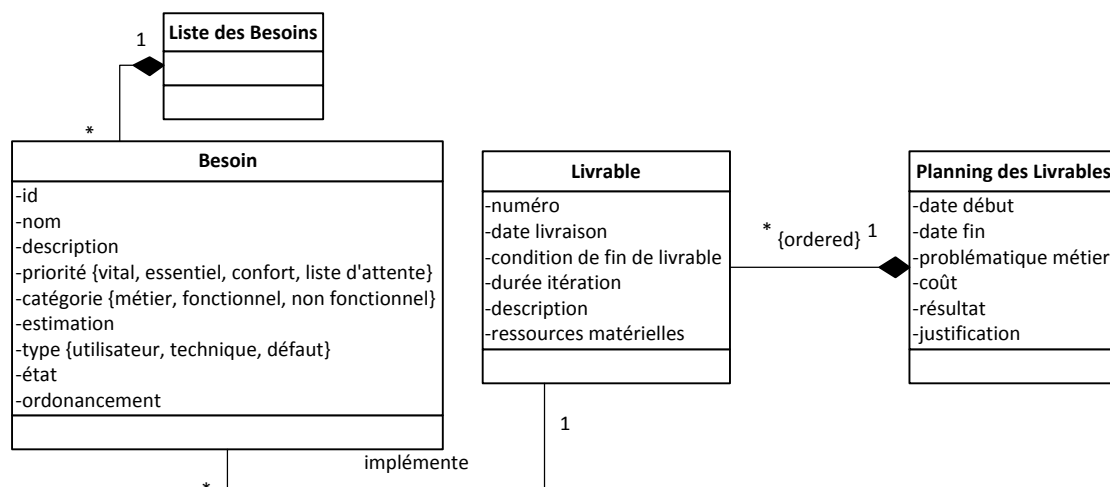


Figure 6.28 : Composant C8 - Définition du plan général des livrables

Comme présenté à la figure 6.29, le descripteur du SMA utilisant le composant C8 est annoté avec la méthode XP qui est la méthode dont est issu C8, puis par l'activité de planification des "livrables" dont traite le composant et enfin par l'activité de la méthode XP de définition du "planning des livrables" qu'il implémente.

```

<description>
<interface name="ReleasePlanDefinition"
conceptualMethodPuzzle="http://.../ReleasePlanDefinition"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../ReleasePlanDefinitionPortlet.wsd1"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#ExtremeProgramming&agileMethodOntology;#ReleasePlanning">
<operation name= "performReleasePlanDefinition"
smeConcept= "&MethodPuzzleOntology;#Process"
modelReference= "&agileMethodOntology;#ReleasePlanDefinition">
<input element="ReleasePlanDefinitionRequest"/>
<output element="ReleasePlanDefinitionResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="ReleasePlanDefinition"
interface="ReleasePlanDefinitionInterface"...>
<endpoint ... />
</service>
</description>

```

Figure 6.29 : Descripteur du composant C8 - Définition du plan général des livrables

Ce composant est implémenté par un SMA gérant la planification des "livrables" du projet (voir figure 6.30).

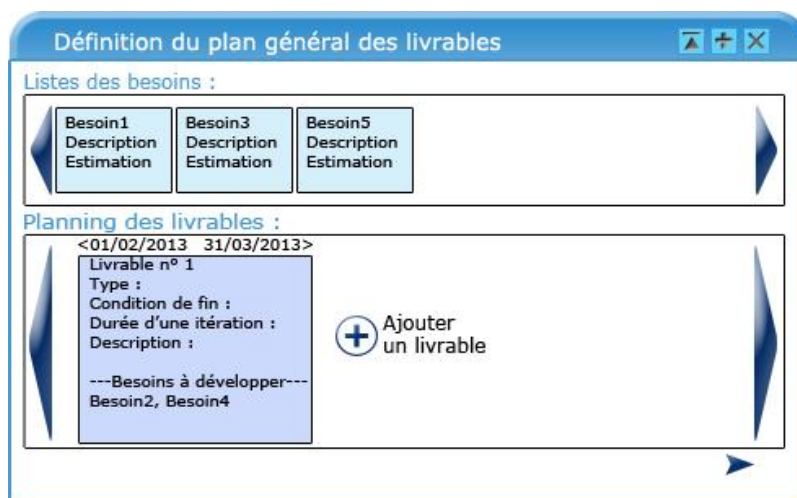


Figure 6.30 : Maquette du composant C8 - Définition du plan général des livrables

En effet, il liste l'ensemble des "besoins utilisateur" du projet et propose à un utilisateur de les organiser en "livrables". Pour ce faire, le SMA fournit une interface pour le découpage de la ligne temporelle du projet en "livrable". Ensuite, il permet à un utilisateur d'affecter les "besoins utilisateur" aux différents "livrables". Il permet ensuite aux utilisateurs de saisir les estimations de temps de développement des "besoins utilisateur" affectés aux "livrables" afin de calculer les dates de livraison de celles-ci. Il propose également aux utilisateurs de venir ajuster la date de fin de "livrable" manuellement s'il est nécessaire de prendre en compte des activités en plus de l'implémentation des besoins pour un même "livrable".

6.4.9 Composant C9 : Définition du plan directeur

Le composant de définition du plan directeur C9 est issu de la méthode DSDM. Il traite de la planification des "livrables" sous la forme d'un "plan directeur" (*Planning des Livrables*). Les besoins des utilisateurs vis-à-vis de la solution à développer (*Besoin*) sont répartis en incrément logiciel en fonction de leurs priorités (*Livrable*). Les dates des "livrables" sont ensuite fixées en fonction de leurs estimations de temps de développement (voir figure 6.31).

C9 – Définition du plan directeur

Objectif : l'objectif de ce composant est d'aider un chef de projet à établir une première planification des livrables du projet en fonction des *Besoins* à développer dans un plan directeur.

Type : atomique

Situation : **Domaine d'application** : Système d'information, Gestion de projet.

Activité de conception : Planification du projet

Intention : Définir un planning des livrables.

Situation source : Liste des besoins de haut niveau

Situation cible : Livrables identifiées et ordonnées dans un plan directeur.

Origine : méthode agile DSDM

<(Liste des besoins), Définir un planning des livrables>

Processus

Le plan directeur, ou encore *Planning des Livrables*, est la première planification effectuée dans le projet. Elle permet de définir les dates des différents jalons ou livrables (*Livable*) auquel un incrément de la solution doit être livré au client.

L'ensemble des besoins (*Besoin*) sont répartis sur les différents livrables avec le client en fonction de leurs priorités.

Une estimation de l'effort et du temps de développement des besoins à implémenter dans chaque livrable est ensuite réalisée. A l'aide de la somme de ces estimations les dates des livrables vont pouvoir être fixées et les livrables organisés dans un planning.

Produit

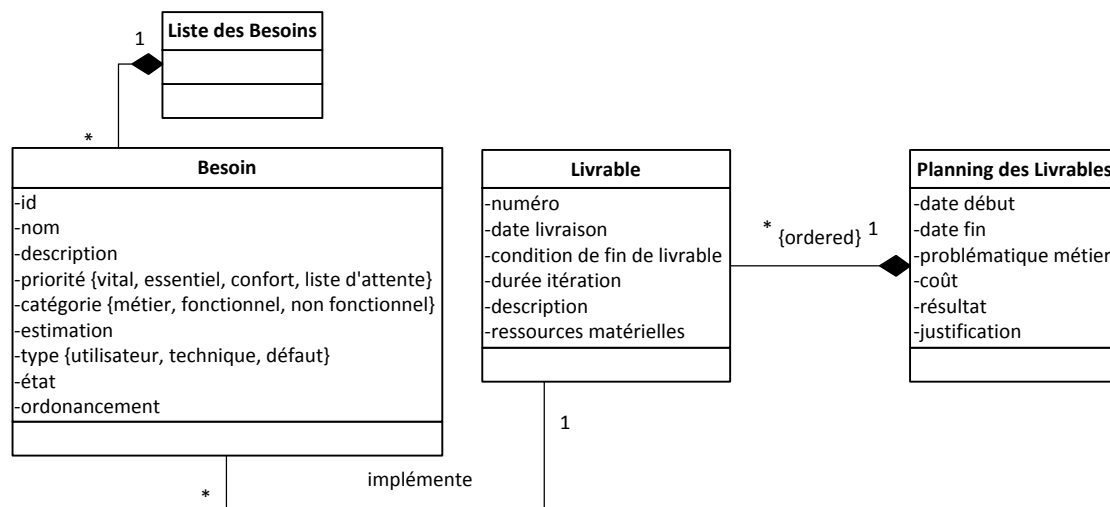


Figure 6.31 : Composant C9 - Définition du plan directeur

Le composant C9 est implémenté par un SMA dont le descripteur reprend les informations sémantiques suivantes : la méthode d'origine DSDM du composant, le fait que le composant traite de l'activité de planification des "livrables" dans le cycle de vie d'une méthode agile et qu'il correspond à l'activité de définition du "plan directeur" dans la méthode DSDM (voir figure 6.32).

```
<description>
<interface name="OutlinePlanDefinition"
conceptualMethodPuzzle="http://.../OutlinePlanDefinition"
puzzleType="atomic-method-service"
```

```

technicalMethodPuzzle="http://.../OutlinePlanDefinitionPortlet.wSDL"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#DynamicSystemDevelopmentMethod
&agileMethodOntology;#ReleasePlanning">
<operation name="performOutlinePlanDefinition"
smeConcept="&MethodPuzzleOntology;#Process"
modelReference="&agileMethodOntology;#OutlinePlanDefinition">
<input element="OutlinePlanDefinitionRequest"/>
<output element="OutlinePlanDefinitionResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="OutlinePlanDefinition"
interface="OutlinePlanDefinitionInterface"...>
<endpoint ... />
</service>
</description>

```

Figure 6.32 : Descripteur du composant C9 - Définition du plan directeur

L'implémentation du composant C9 prend la forme d'un SMA aidant les utilisateurs à planifier leur projet en le découpant en "livrables" puis en y répartissant l'ensemble des besoins en fonction de leur priorité (voir figure 6.33).

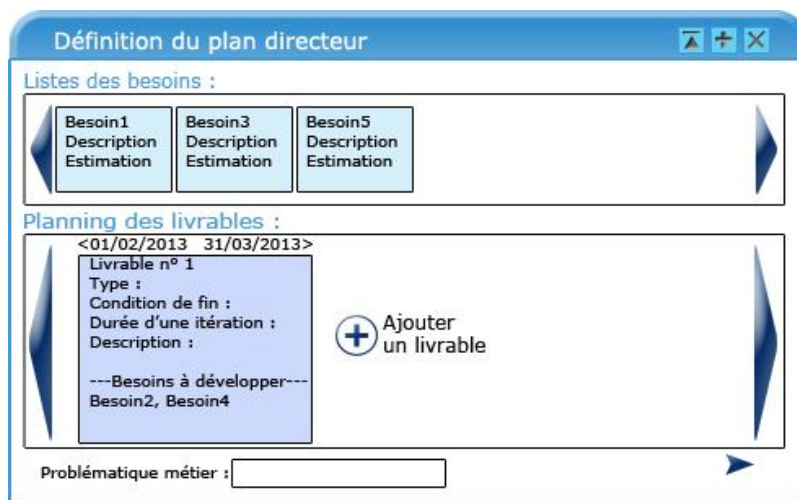


Figure 6.33 : Maquette du composant C9 - Définition du plan directeur

Ce SMA se différencie de celui outillant le composant C8 sur ce point. En effet, dans le composant C8 les besoins sont répartis par l'équipe et le client sur les "livrables" alors qu'ici les besoins sont répartis en fonction de leur priorité. Le SMA implémentant C9 doit donc prendre en charge la gestion de l'ordonnement de la liste des besoins et établir une correspondance entre celle-ci et la répartition des besoins dans la planification du projet. Il propose aux utilisateurs de découper le projet en

"livrables" puis d'affecter, en fonction de leur priorité, des besoins aux "livrables" et d'estimer le temps de développement de chaque besoin pour pouvoir calculer les dates de livraison des "livrables".

6.4.10 Composant C10 : Affinement des besoins

La méthode DSDM propose, après avoir établi une première planification du projet, de revenir sur la définition des besoins (*Besoin*) afin de les affiner. Cet aspect est réalisé dans le composant C10 de la ligne de méthode (voir figure 6.34). Il regroupe la décomposition des besoins de haut niveau en sous-besoins et une première identification générale des besoins non fonctionnels de la solution à développer.

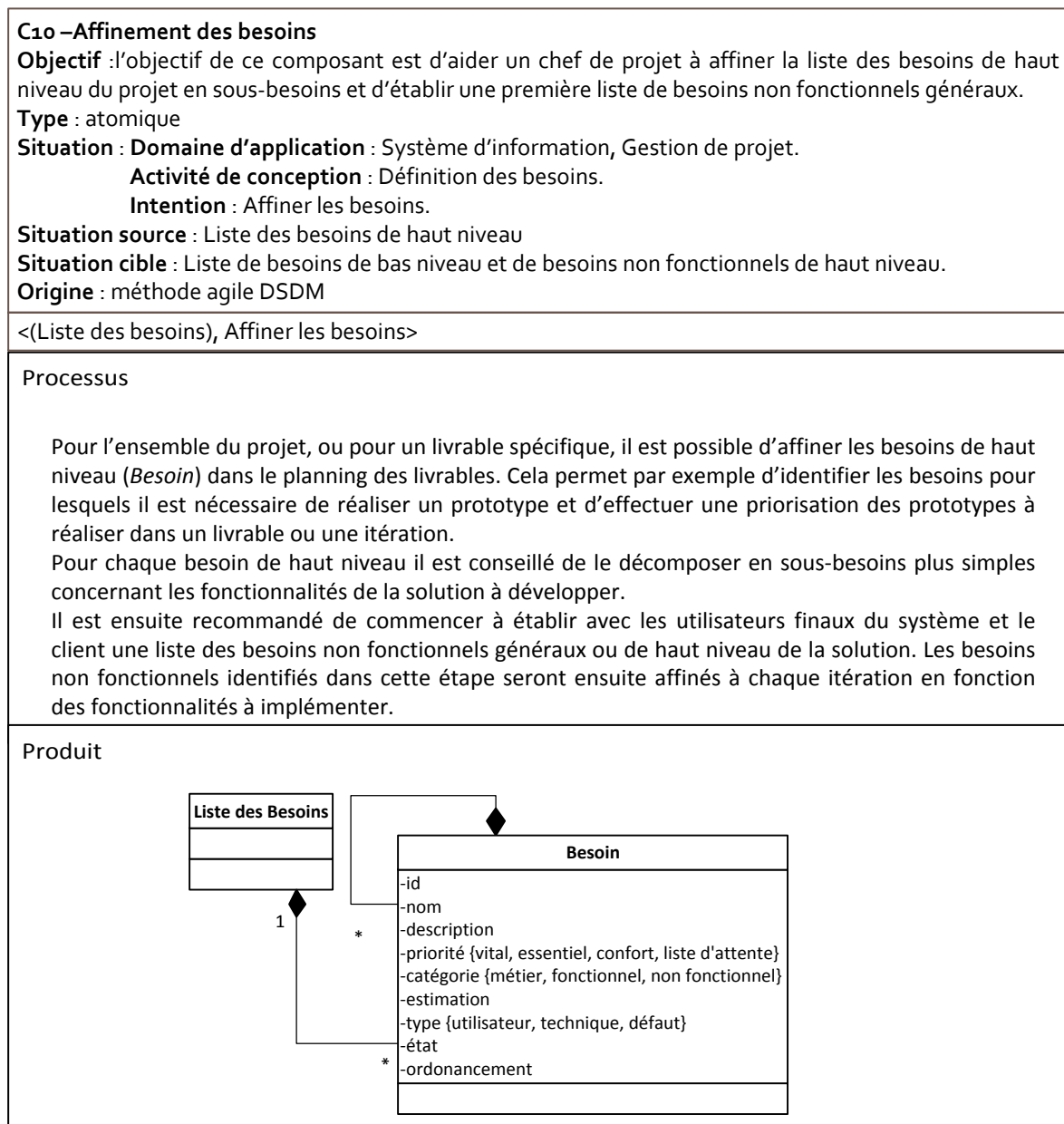


Figure 6.34 : Composant C10 - Affinement des besoins

Le descripteur du SMA outillant le composant C10 est annoté par les informations sémantiques suivantes : DSDM (la méthode d'origine du composant), ainsi que l'activité de définition des besoins auquel est rattaché se composant et enfin l'activité d'affinement des besoins qu'il traite dans la méthode DSDM (voir figure 6.35).

```

<description>
<interface name="RequirementsRefinement"
conceptualMethodPuzzle="http://.../RequirementsRefinement"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../RequirementsRefinementPortlet.wsd1"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#DynamicSystemDevelopmentMethod
&agileMethodOntology;#RequirementsDefinition">
<operation name="performRequirementsRefinement"
smeConcept=" &MethodPuzzleOntology;#Process"
modelReference=" &agileMethodOntology;#RequirementsRefinement">
<input element="RequirementsRefinementRequest"/>
<output element="RequirementsRefinementResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="RequirementsRefinement"
interface="RequirementsRefinementInterface"...>
<endpoint ... />
</service>
</description>

```

Figure 6.35 : Descripteur du composant C10 - Affinement des besoins

Le SMA implémentant ce composant propose d'aider les utilisateurs à affiner les besoins du projet en leur proposant une interface leur permettant de décomposer chaque besoin fonctionnel en sous-besoins plus simples avec une granularité plus faible (voir figure 6.36).



Figure 6.36 : Maquette du composant C10 - Affinement des besoins

Ce composant propose également aux utilisateurs de saisir la liste des besoins non fonctionnels de haut niveau de la solution à développer.

6.4.11 Composant C11 : Définition du planning de développement

Le composant C11 de définition du "planning de développement" consiste en un affinement de la première planification des "livrables" établie précédemment dans le processus de la ligne de méthode (*Planning des Livrables*). En effet, ce composant issu de la méthode DSDM propose de décomposer en itérations (*Itération*) les "livrables" (*Livrables*) présentes sur le planning afin de leur affecter des besoins (*Besoin*) à implémenter et de déterminer leur durée (voir figure 6.37).

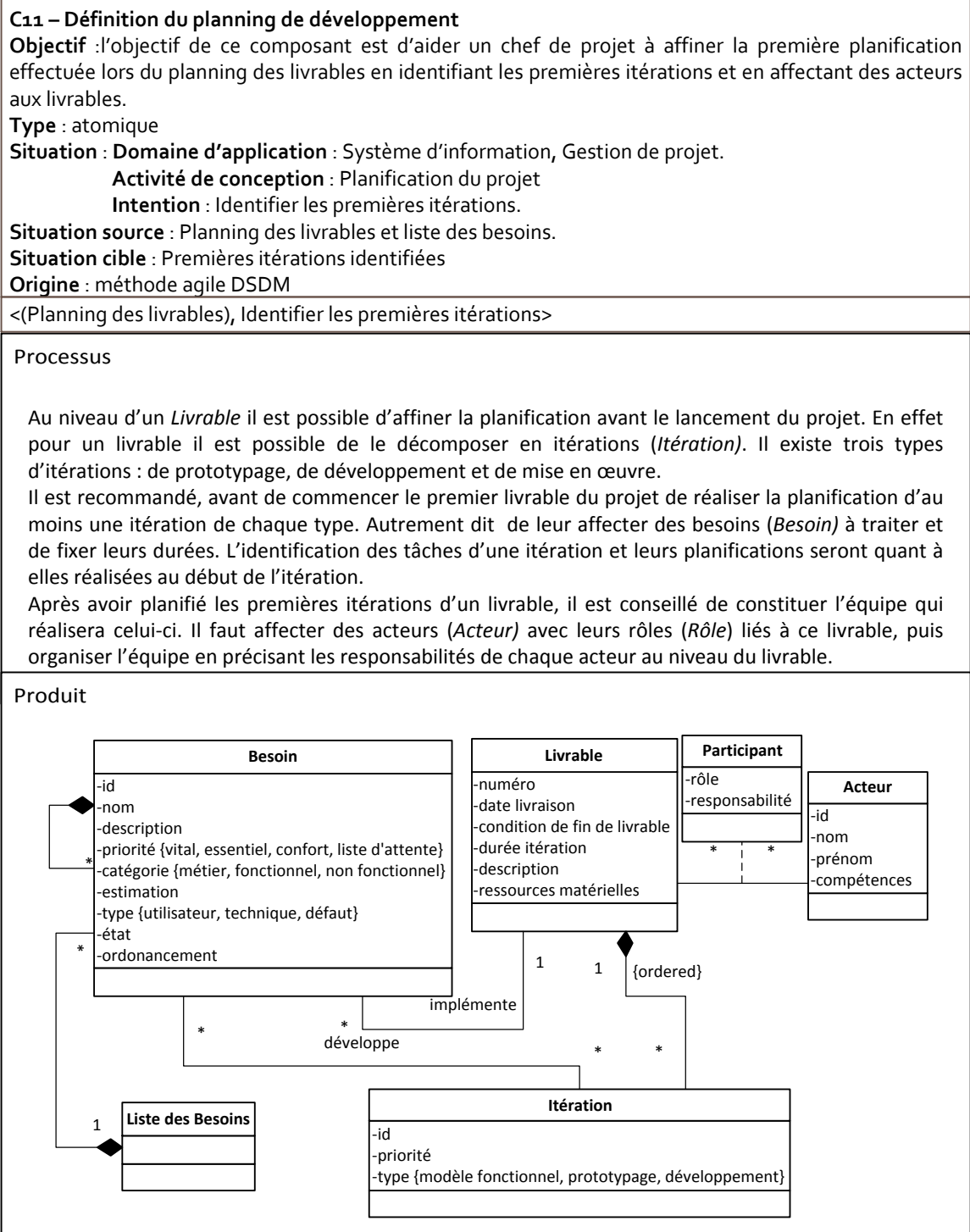


Figure 6.37 : Composant C11- Définition du planning de développement

Le composant C11 est outillé par un SMA dont le descripteur à été annoté par le fait que le composant est issu de la méthode DSDM, qu'il concerne l'activité de planification des "livrables" dans le cycle de vie d'une méthode agile et qu'il correspond à l'activité de définition du "planning de développement" dans la méthode DSDM (voir figure 6.38).

<description>

```

<interface name="DevelopmentPlanDefinition"
conceptualMethodPuzzle="http://.../DevelopmentPlanDefinition"
puzzleType="atomic-method-service"
technicalMethodPuzzle="http://.../DevelopmentPlanDefinitionPortlet.wSDL"
smeConcept="&MethodPuzzleOntology;#Method
&MethodPuzzleOntology;#MethodPuzzle"
modelReference="&agileMethodOntology;#DynamicSystemDevelopmentMethod
&agileMethodOntology;#ReleasePlanning">
<operation name="performDevelopmentPlanDefinition"
smeConcept="&MethodPuzzleOntology;#Process"
modelReference="&agileMethodOntology;#DevelopmentPlanDefinition">
<input element="DevelopmentPlanDefinitionRequest"/>
<output element="DevelopmentPlanDefinitionResponse"/>
</operation>
</interface>
<binding ...> ...</binding>
<service name="DevelopmentPlanDefinition"
interface="DevelopmentPlanDefinitionInterface"...>
<endpoint ... />
</service>
</description>

```

Figure 6.38 : Descripteur du composant C11 - Définition du planning de développement

Le composant conceptuel C11 est outillé en pratique par un SMA aidant les utilisateurs à affiner la planification des "livrables" de leur projet en leur proposant de découper les "livrables" en itération et de fixer leur durée (voir figure 6.39).



Figure 6.39 : Maquette du composant C11 - Définition du planning de développement

Dans un deuxième temps, il permet aux utilisateurs d'affecter des besoins à implémenter dans un "livrable" aux itérations. Enfin, ce SMA aide, dans un troisième temps, les utilisateurs à constituer les équipes affectées à la réalisation des "livrables" en leur permettant de saisir les acteurs intervenant dans chaque "livrable" ainsi que leur rôle et leur responsabilité.

6.4.12 Modèle de produit global de la ligne de méthode

Cette section présente le modèle de produit de la ligne de méthode de ce cas d'application. Conformément aux recommandations de l'approche MaaS, le modèle de produit global de la ligne est construit par une agrégation des modèles de produit des SMA composant la ligne. Par conséquent, il regroupe l'ensemble des modèles des produits utilisés dans le flux du processus de la ligne tel que présenté à la figure 6.4. La figure ci-dessous présente ce modèle global pour la ligne de méthode de lancement d'un projet agile.

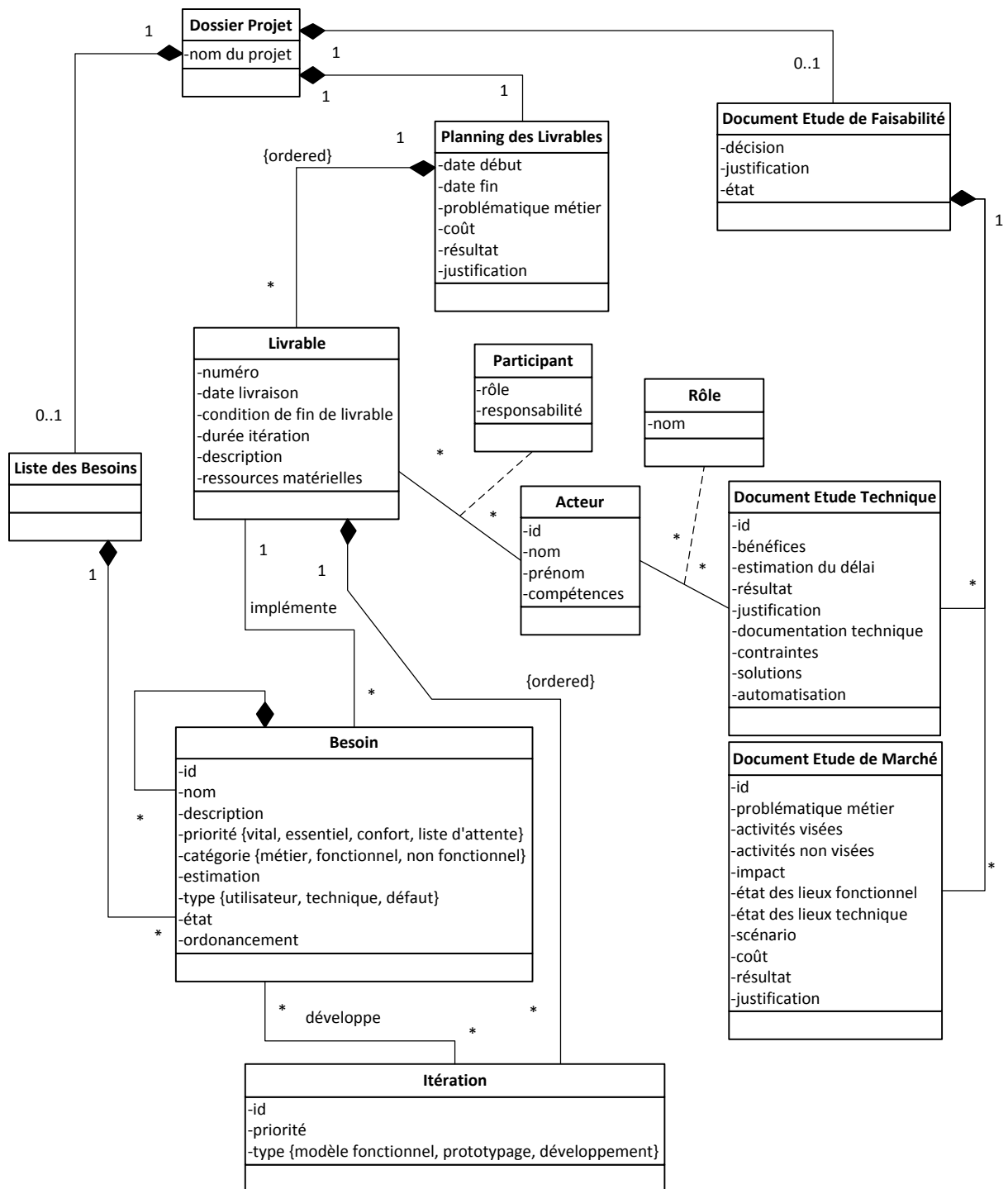


Figure 6.40 : Modèle de produit global de la ligne de méthode de lancement de projet agile

La ligne de ce cas d'application étant la première à être exécutée dans un projet, elle ne prend pas de paramètre en entrée. Néanmoins, tous les éléments de produit de cette ligne sont regroupés dans un dossier projet représenté dans le modèle produit de la ligne par la classe *Dossier Projet* ayant un lien de composition vers toutes les classes de plus haut niveau du modèle : *Document Etude de Faisabilité*, *Liste des Besoins* et *Planning des Livrables*.

Les premiers SMA pouvant être exécutés dans cette ligne permettent d'effectuer une étude de marché du projet et une étude de faisabilité technique du projet. Ces deux SMA produisent respectivement un ou plusieurs documents d'étude de marché et documents d'étude technique qui sont représentés dans le modèle de produit par les classes *Document Etude de Marché* et *Document Etude Technique*. Ces deux types de document sont ensuite regroupés dans un document d'étude de faisabilité représenté par la classe *Document Etude de Faisabilité* reliée aux classes *Document Etude de Marché* et *Document Etude Technique* par un lien de composition dans le modèle.

Les SMA de définitions des besoins du projet viennent ensuite dans le flux d'exécution du processus de la ligne de méthode. Ces SMA aident les utilisateurs à définir leurs besoins et à les ordonner dans une liste de besoins. Les besoins vis-à-vis du logiciel à développer peuvent être définis à des niveaux de granularité différents, c'est-à-dire qu'ils peuvent être définis à un haut niveau d'abstraction puis être affinés ensuite. Cela est introduit dans le modèle de produit de la ligne par les classes *Besoin* et *Liste des Besoins* reliées par un lien de composition. L'affinement des besoins est représenté par le lien de composition réflexif de la classe *Besoin* sur elle-même.

Si le projet est faisable, le processus de la ligne invite les utilisateurs à établir une première planification de celui-ci. Dans le cas contraire, la clôture du projet est effectuée par le SMA de clôture du projet et l'état de l'objet *Document Etude de Faisabilité* est modifié. Les SMA de planification du projet proposent de définir un "planning des livrables" en identifiant les différentes "livrables" du projet. Cela est représenté dans le modèle de produit de la ligne par les classes *Planning des Livrables* et *Livrable*. Le lien de composition entre ces deux classes représente le fait que les "livrables" sont ordonnées dans le planning. Certains SMA préconisent d'affecter à cette étape des besoins à implémenter dans les "livrables". Un "livrable" pouvant implémenter plusieurs besoins, cela est représenté dans le modèle par l'association *implémente* avec une cardinalité de 1 à n entre les classes *Livrable* et *Besoin*.

Un utilisateur peut ensuite, s'il le désire, affiner la définition des besoins et la planification du projet. Comme mentionné précédemment, l'affinement des besoins est modélisé par une composition réflexive de la classe *Besoin* sur elle-même. L'affinement de la planification prend place, quant à elle, par un affinement des "livrables" en itération de développement. Cela est représenté sur le modèle par la classe *Itération* et un lien de composition entre la classe *Livrable* et la classe *Itération*. Durant une itération, plusieurs besoins peuvent être développés, mais les besoins peuvent également être implémentés sur plusieurs itérations dans le temps. L'affectation des besoins à une itération correspond dans le modèle à l'association *développe* avec une cardinalité de n à n entre les classes *Besoin* et *Itération*.

Ce modèle de produit conceptuel de la ligne de méthode est ensuite transformé en schéma XSD dans le cadre de son opérationnalisation. Ce schéma XSD étant trop volumineux, nous présentons ci-dessous quelques uns de ces éléments clefs (le reste du schéma est disponible à l'annexe J).

Dans ce schéma XSD, toutes les classes sont transformées en *element* comme le montre la figure 6.41 ci-dessous pour la classe *Livable*.

```
<xsd:element name="Livvable">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="tns:Iteration" minOccurs="0" maxOccurs="unbounded" />
<xsd:element ref="tns:Implemente" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="numero" use="required" type="xsd:string" />
<xsd:attribute name="dateLivraison" use="required" type="xsd:dateTime" />
<xsd:attribute name="conditionDeFinDeLivvable"
use="required" type="xsd:string" />
<xsd:attribute name="dureeIteration" use="required" type="xsd:string" />
<xsd:attribute name="description" use="optional" type="xsd:string" />
<xsd:attributenam="ressourcesMaterielles" use="optional"
type="xsd:string" />
</xsd:complexType>
</xsd:element>
```

Figure 6.41 : Exemple de définition XSD de la classe Livvable

La définition de la classe *Livable* est regroupée dans l'élément *element* (en bleu sur la figure 6.41) et les attributs de la classe sont définis avec l'élément *attribute* à l'intérieur d'un *complexType*.

Pour une composition, l'élément représentant la référence de définition XSD de la classe entrant dans la composition est défini dans un élément *sequence* de l'élément de la classe mère (en vert sur la figure).

Concernant les associations avec une cardinalité de 1 à n, l'élément correspondant à l'association est référencé dans un élément *sequence* de la définition XSD de la classe mère (en rouge sur la figure).

La figure suivante présente la définition en XSD des concepts d'association et de classes associations d'un digramme de classe UML.

```
<xsd:element name="Developpe">
<xsd:complexType>
<xsd:attribute name="idBesoin" use="required" type="xsd:string" />
<xsd:attribute name="idIteration" use="required" type="xsd:string" />
</xsd:complexType>
</xsd:element>

<xsd:element name="Implemente">
<xsd:complexType>
<xsd:attribute name="idBesoin" use="required" type="xsd:string" />
</xsd:complexType>
</xsd:element>

<xsd:element name="Participant">
<xsd:complexType>
<xsd:attribute name="idActeur" use="required" type="xsd:string" />
<xsd:attribute name="numeroLivvable" use="required" type="xsd:string" />
<xsd:attribute name="role" use="required" type="xsd:string" />
<xsd:attribute name="responsabilite" use="optional" type="xsd:string" />
```

```
</xsd:complexType>
</xsd:element>
```

Figure 6.42 : Exemple de définition XSD d'association

La figure 6.42 présente la définition des associations *Developpe* et *Implemente*. L'association *Developpe* a une cardinalité n-n. Elle possède donc deux attributs (en bleu sur la figure) faisant référence à l'identifiant des classes ayant un rôle dans l'association. L'association *Implemente* à une cardinalité 1-n, elle ne possède donc qu'un seul attribut (en bleu sur la figure) servant à référencer les besoins à implémenter dans un "livrable".

Les classes associations se définissent dans le XSD de la même manière que les associations de type n-n comme le présente la figure 6.42 par l'exemple de définition de la classe association *Participant*.

La figure ci-dessous présente la définition de l'élément racine du modèle de produit de la ligne.

```
<xsd:element name="ModeleDeProduit">
<xsd:complexType>
<xsd:sequence>
<xsd:elementref="tns:DossierProjet" minOccurs="1" maxOccurs="1" />
<xsd:elementref="tns:Acteur" minOccurs="0" maxOccurs="unbounded" />
<xsd:elementref="tns:Role" minOccurs="0" maxOccurs="unbounded" />
<xsd:elementref="tns:Participant" minOccurs="0" maxOccurs="unbounded" />
<xsd:elementref="tns:Iteration" minOccurs="0" maxOccurs="unbounded" />
<xsd:elementref="tns:Developpe" minOccurs="0" maxOccurs="unbounded" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

Figure 6.43 : Exemple de définition XSD de l'élément racine *ModeleDeProduit*

L'élément racine *ModeleDeProduit* regroupe les références pour toutes les classes de plus haut niveau du modèle de produit, c'est-à-dire toutes les classes n'entrant pas comme composante d'une composition. Il regroupe également les références des associations de type n-n et des classes associations (voir figure 6.43). C'est pour cette raison que l'élément *ModeleDeProduit* regroupe les éléments de type classe : *DossierProjet*, *Acteur* et *Iteration*, les éléments de type association : *Developpe* et les éléments de type classes associations : *Role* et *Participant*.

Etant donné que les SMA de cette ligne de méthode ont comme modèle de produit des fragments de ce modèle de produit global de la ligne, ils ont par conséquent comme représentation XSD de leur modèle de produit des fragments du schéma XSD du modèle de produit de la ligne.

6.5. Modèle de la ligne de méthode d'un lancement de projet agile

Afin de produire le modèle de la ligne de méthode du cas d'application de ce chapitre, nous appliquons la démarche et les règles de transformation d'un modèle d'acquisition d'une ligne de méthode telles que présentées à la section 5.3.3.3 du chapitre 5. L'arbre d'exécution de la ligne de méthode du lancement d'un projet agile est produit à partir du modèle d'acquisition de cette ligne présenté à la figure 6.4. Cet arbre étant volumineux, sa représentation au format XML est disponible dans l'annexe K et la figure 6.44 présente, quant à elle, sa représentation graphique (les feuilles de l'arbre de couleur bleue

représentent les différents points d'arrêt possibles du processus). Pour des raisons de lisibilité, les attributs des feuilles de type *AtomicMethodService* ne sont pas renseignés sur la représentation graphique de l'arbre. Ils sont néanmoins accessibles dans la version XML disponible à l'annexe K.

La première transformation de ce modèle concerne le point de variation D1. Ce point de variation est particulier, il propose aux utilisateurs de choisir ou non de réaliser une étude de faisabilité du projet. Etant donné que l'étude de faisabilité peut amener à terminer le projet, le point de variation D1 va donc séparer en deux l'arbre d'exécution de la ligne : une partie avec le processus contenant l'étude de faisabilité et une autre partie identique, mais sans l'étude de faisabilité. Le nœud D1 de type *DecisionPoint* représente donc ce point de variation. Etant de type XOR, il a pour cardinalité de choix min et max la valeur 1.

Nous développons, dans un premier temps, la partie de l'arbre présentant l'étape de faisabilité du projet. Ceci nous amène alors au deuxième point de variation du modèle d'acquisition, le point D2. La deuxième transformation est l'ajout du nœud D2 de type *DecisionPoint* en tant que fils du nœud D1. D2 étant un point de variation de type OR, ses cardinalités sont de 1 pour le min et 2 pour le max car il propose un choix optionnel entre les deux SMA C1 et C2. Ces SMA sont également ajoutés à l'arbre en tant que feuilles *AtomicMethodService* et fils de D2.

Viens ensuite dans le processus une séquence reliant le bloc formé par le point de variation D2 et celui formé par D4. La troisième étape consiste donc en la création d'un nœud *Sequence* Seq1 comme parent du nœud D2 et fils de D1. Le nœud D4 est ensuite ajouté en tant que deuxième fils de la *Sequence* Seq1. Cela représente la troisième étape de la transformation du modèle.

Le nœud D4 représente un point de variation de type XOR, par conséquent il possède une cardinalité min et max de 1. Il propose soit d'arrêter le processus par l'exécution du SMA C3, soit de reprendre le processus à la définition des besoins. Une feuille *AtomicMethodService* C3_1 représentant le SMA est donc ajoutée comme premier fils de D4. L'ensemble de ces opérations constitue la quatrième transformation du modèle.

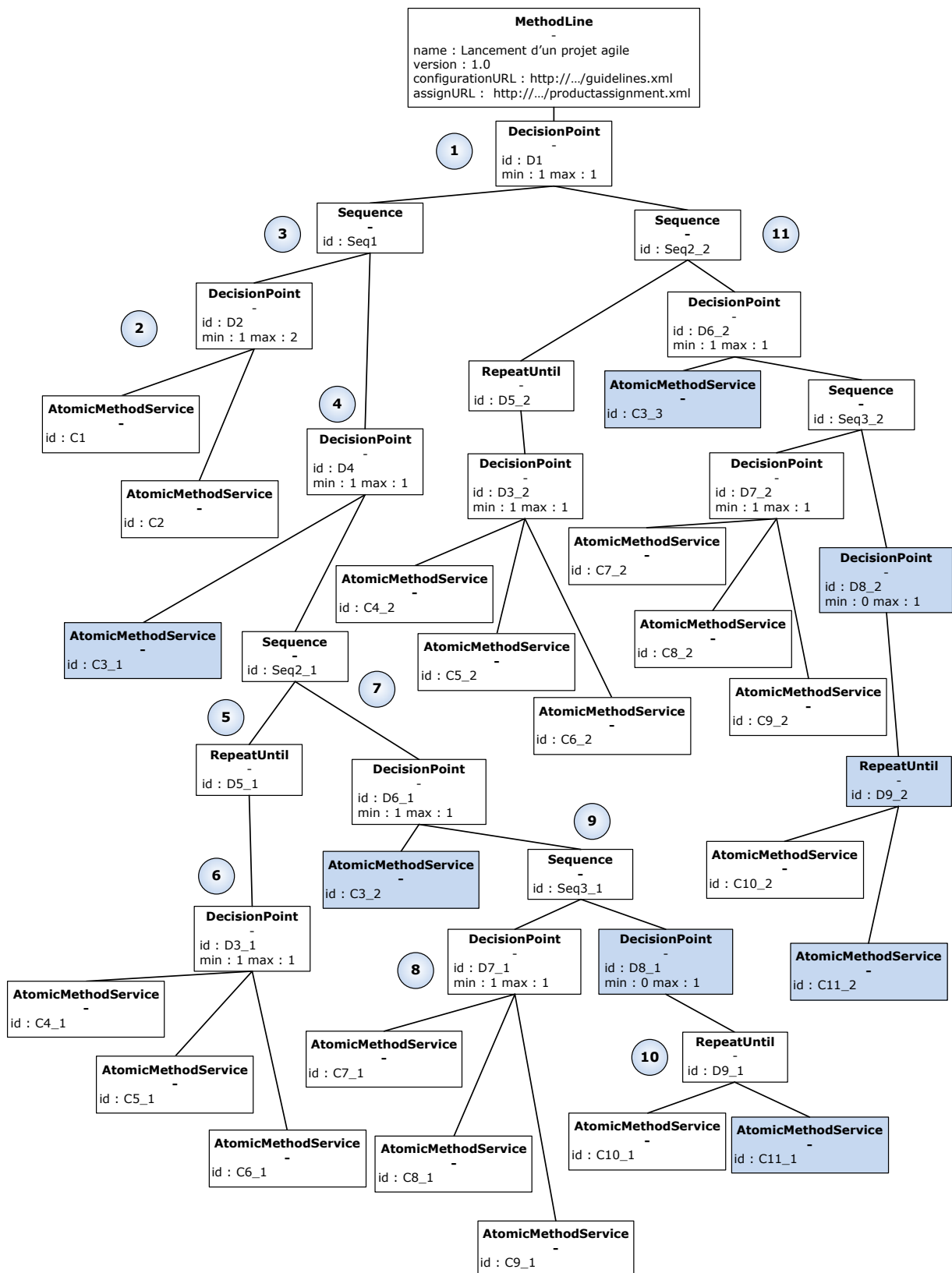


Figure 6.44 : Représentation graphique de l'arbre d'exécution de la ligne du cas d'application

Le deuxième choix proposé par le point de variation D4 est de continuer le processus au point itératif D5. Cela est décrit dans l'arbre comme la cinquième transformation, par l'ajout d'un deuxième fils de type *RepeatUntil* (nommé D5_1) au nœud D4.

La sixième étape prend alors la forme du point de variation D3 qui est le sujet de la boucle de l'itération D5. Ce point est de type XOR et ses cardinalités sont donc de 1 pour le min et le max. Ce point de variation permet aux utilisateurs de choisir entre plusieurs méthodes pour identifier leurs besoins, c'est-à-dire les SMA C4, C5 et C6. Le nœud D3_1 de type *DecisionPoint* est alors ajouté comme unique fils de D5_1. Ce nœud D3_1 est affecté de trois fils C4_1, C5_1 et C6_1 de type *AtomicMethodService* représentant respectivement les SMA C4, C5 et C6.

La septième étape de transformation concerne la séquence reliant le bloc de processus constitué par l'itération D5 qu'elle relie avec le point de variation D6. Un nœud *Sequence* nommé Seq2_1 est créé comme parent du nœud *RepeatUntil* D5_1 car ce dernier ne possède pas de nœud agrégatif itératif ou séquentiel comme parent. Un nœud *DecisionPoint* représentant le point de variation D6 est ajouté en tant que second fils du nœud *Sequence* Seq2_1. Ce point de variation propose soit d'arrêter le projet en exécutant un dernier SMA C3, soit de passer à l'étape de planification du projet. Le SMA C3 est, quant à lui, représenté par une feuille *AtomicMethodService* nommée C3_2 ajoutée comme fils de D6_1.

Le point de variation D7 propose aux utilisateurs le choix entre trois manières différentes (les SMA C7, C8 et C9) pour établir la planification initiale de leur projet. Cette huitième étape consiste en la création du nœud D7_1 de type *DecisionPoint* avec des cardinalités min et à max à 1 car il est de type XOR. Les SMA C7, C8, C9 sont ajoutés respectivement comme fils du nœud D7_1 par le biais des feuilles *AtomicMethodService* C7_1, C8_1 et C9_1.

La neuvième étape concerne la transformation de la séquence reliant le bloc de processus formé par le point de variation D7 avec celui formé par le point de variation D8. Le nœud parent du nœud D7_1 n'étant pas actuellement de type *Sequence* ou *RepeatUntil*. Un nœud *Sequence* nommé Seq3_1 est créé comme parent du nœud D7_1 et du nœud D8_1. Ce dernier représente le point de variation D8, il est de type *DecisionPoint*. Les cardinalités de D8_1 sont de 0 pour le min et 1 pour le max car le point D8 est de type XOR et une de ses branches de sortie est reliée à l'élément "arrêter" du processus.

Le point de variation D8 porte sur le choix d'effectuer une boucle d'affinage des besoins ou d'arrêter le processus. Etant donné que le choix de l'arrêt est mentionné par la cardinalité minimum de 0 du nœud D8_1, ce nœud n'aura donc qu'un unique fils de type *RepeatUntil* nommé D9_1 et représentant l'itération D9. Ce point d'agrégation itératif porte sur la répétition séquentielle des SMA C10 et C11. Ces derniers sont par conséquent ajoutés comme fils du nœud itératif D9_1. Ils sont respectivement représentés par les feuilles *AtomicMethodService* C10_1 et C11_1. Ces opérations constituent la dixième étape de la transformation du modèle.

Cette partie du processus intégrant les activités d'étude de faisabilité de projet prend fin au composant C11 représenté par le nœud C11_1. La deuxième partie du processus est le même processus avec l'omission des activités concernant les études de faisabilité. Les transformations à effectuer sont identiques à celles de l'étape 5 jusqu'à l'étape 10, c'est-à-dire la production du sous-arbre qui a pour racine le nœud *Sequence* Seq_1. Il est alors nécessaire de répéter ces étapes de transformation en renommant les nœuds et feuilles pour que leur identifiant soit unique et de rattacher ce sous-arbre produit (sous arbre de racine Seq2_2 à la figure 6.44) en tant que deuxième fils du nœud D1 de type *DecisionPoint*. En effet ce nœud propose le choix entre les deux branches (avec étude de faisabilité ou non) du processus de la ligne de méthode.

6.6. Modèle de transformation des entrées/sorties

Le document XML de transformation des entrées/ sorties de la ligne de méthode de lancement d'un projet agile est construit à partir du schéma XML représentant son arbre d'exécution (voir respectivement l'annexe K et la figure 6.44 de ce chapitre).

Pour les entrées/sorties de la ligne de méthode et pour celles de chaque SMA identifié dans l'arbre d'exécution de la ligne, un ensemble de règles de transformation est ajouté au modèle de transformation. Le document XML intégrant toutes ces règles de transformation étant trop volumineux, il est disponible à l'annexe L. Nous présentons dans cette section les éléments principaux de ce modèle de transformation.

Le premier élément du modèle de transformation est celui décrivant les règles de transformation de la ligne en elle-même (voir la figure suivante).

```
<am:MethodLine name="Lancement d'un projet agile">
<am:Input/>
<am:Output>
<am:Transform source="mlpm:ModeleDeProduit" target="mlpm:ModeleDeProduit"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
</am:Output>
</am:MethodLine>
```

Figure 6.45 : Élément MethodLine du modèle de transformation des entrées/sorties

Comme présenté à la figure 6.45, cette ligne de méthode ne prend pas de données en entrée et elle donne en sortie l'instance complète de son modèle de produit.

Les deux premiers SMA susceptibles d'être exécutés dans la ligne sont les SMA C1 et C2. La figure ci-dessous présente les règles de transformation des entrées/sorties du SMA C1.

```
<am:AssignparentId="D2" leafId="C1">
<am:Input/>
<am:Output>
<am:Transform source="c1:DocumentEtudeDeMarche"
target="mlpm:DocumentEtudeDeFaisabilite"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/insert.xslt"/>
</am:Output>
</am:Assign>
```

Figure 6.46 : Règles de transformation des entrées/sorties du SMA C1

Tout comme la ligne de méthode, ce SMA ne prend pas de données en entrée. Il délivre en sortie un document d'étude de marché qui est intégré au document d'étude de faisabilité du projet de la ligne de méthode (voir figure 6.46).

Le SMA C2 possède des règles de transformation similaires à C1 excepté qu'il délivre en sortie un document d'étude technique, un ensemble d'acteurs et leur rôle dans la solution envisagée, comme présenté à la figure 6.47.

```
<am:AssignparentId="D2" leafId="C2">
<am:Input/>
<am:Output>
<am:Transform source="c2:DocumentEtudeTechnique"
target="mlpm:DocumentEtudeDeFaisabilite"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/insert.xslt"/>
<am:Transform source="c2:Acteur" target="mlpm:Acteur"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/insert.xslt"/>
<am:Transform source="c2:Role" target="mlpm:Role"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/insert.xslt"/>
</am:Output>
</am:Assign>
```

Figure 6.47 : Règles de transformation des entrées/sorties du SMA C2

Le SMA C3 apparait trois fois dans l'arbre d'exécution de la ligne de méthode sous les identifiants : C3_1, C3_2 et C3_2. Il est donc nécessaire d'ajouter au modèle de transformation des entrées/sorties de ligne trois ensemble de règles de transformation : une par utilisation du SMA dans l'arbre. La figure ci-dessous présente les règles de transformation pour C3_1 et C3_2.

```
<am:AssignparentId="D4" leafId="C3_1">
<am:Input>
<am:Transform source="mlpm:DocumentEtudeDeFaisabilite"
target="c3:DocumentEtudeDeFaisabilite"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
</am:Input>
<am:Output>
<am:Transform source="c3:DocumentEtudeDeFaisabilite"
target="mlpm:DocumentEtudeDeFaisabilite"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
</am:Output>
</am:Assign>

<am:AssignparentId="D6_1" leafId="C3_2">
<am:Input>
<am:Transform source="mlpm:DocumentEtudeDeFaisabilite"
target="c3:DocumentEtudeDeFaisabilite"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
<am:Transform source="mlpm:ListeDesBesoins" target="c3:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
</am:Input>
<am:Output>
<am:Transform source="c3:DocumentEtudeDeFaisabilite"
target="mlpm:DocumentEtudeDeFaisabilite"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
</am:Output>
</am:Assign>
```

Figure 6.48 : Règles de transformation des entrées/sorties du SMA C3

La figure 6.48 présente les deux utilisations différentes du SMA C3 (les règles de transformation de C3_2 et C3_3 sont identiques). C3_1 est exécuté lorsque le résultat de l'étude de faisabilité du projet s'est soldé par un échec et prend donc en entrée le document complet d'étude de faisabilité pour le rendre modifié en sortie. C3_2 et C3_3 sont exécutés, quant à eux, en sortie de l'étape de définition des besoins des projets lorsque le projet est décrété comme n'étant pas réalisable. Ils prennent donc en entrée la liste des besoins et le document d'étude de faisabilité du projet s'il existe. Ils donnent en sortie le document d'étude de faisabilité dont l'état a été mis à jour.

L'étape de définition des besoins du projet est réalisable dans cette ligne à l'aide des composants C4, C5 ou C6. Leurs entrées et sorties sont similaires et, par conséquent, les règles de transformation de ces dernières le sont également. Ces SMA sont présents, pour chacun d'entre eux, deux fois dans l'arbre d'exécution de la ligne. La figure suivante présente les règles de transformation du SMA C6.

```
<am:AssignparentId="D3_1" leafId="C6_1">
<am:Input>
<am:Transform source="mlpm:DocumentEtudeDeMarche"
target="c6:DocumentEtudeDeMarche"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
<am:Transform source="mlpm:ListeDesBesoins" target="c6:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
</am:Input>
<am:Output>
<am:Transform source="c6:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
</am:Output>
</am:Assign>
```

Figure 6.49 : Règles de transformation des entrées/sorties du SMA C6

Le SMA C6 est utilisé deux fois dans la ligne sous l'identifiant C6_1 et C6_2. Dans ces deux cas d'utilisation, il possède les mêmes règles de transformation. Ce SMA prend en entrée le document d'étude de marché s'il existe, car il peut se servir de la problématique métier du projet comme base pour la définition des besoins (voir figure 6.49). Les besoins étant définis de manière itérative, il peut également recevoir en entrée une liste des besoins existants pour la compléter. C6 délivre en sortie une liste des besoins mise à jour. Les SMA C4 et C5 ont un ensemble de règles similaires à C6 à l'exception qu'ils ne prennent en entrée qu'un seul paramètre : une liste des besoins si elle est déjà existante et ceux pour leurs utilisations sous les identifiants C4_1, C4_2, C5_1 et C5_2 dans l'arbre d'exécution de la ligne.

Viens ensuite dans le processus de la ligne l'étape de planification du projet réalisable au travers des SMA C7, C8 ou C9. Ces trois SMA sont présents deux fois chacun dans l'arbre d'exécution de la ligne et ils possèdent les même règles de transformation, comme présenté ci-dessous pour le SMA C7 :

```
<am:AssignparentId="D7_1" leafId="C7_1">
<am:Input>
<am:Transform source="mlpm:ListeDesBesoins" target="c7:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
</am:Input>
<am:Output>
```

```

<am:Transform source="c7:PlanningDesLivrables" target="mlpm:PlanningDesLivrables"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
</am:Output>
</am:Assign>

```

Figure 6.50 : Règles de transformations des entrées/sorties du SMA C7

Les SMA C7, C8 et C9 sont utilisées dans l'arbre de la ligne sous les identifiants C7_1, C7_2, C8_1, C8_2, C9_1 et C9_2. Sur l'ensemble de ces utilisations, ils possèdent les mêmes types de règles de transformation. L'exemple présenté à la figure 6.50 pour le SMA C7_1 décrit qu'il a pour entrée la liste des besoins à partir de laquelle il va pouvoir établir une planification du projet qu'il donnera en sortie.

Après avoir obtenu une première planification du projet, il est possible d'affiner la définition des besoins pour ensuite affiner la planification. Le composant C10 propose un guide pour l'affinement des besoins et les règles de transformation de ces entrées et sorties sont décrites à la figure suivante.

```

<am:AssignparentId="D9_1" leafId="C10_1">
<am:Input>
<am:Transform source="mlpm:ListeDesBesoins" target="c10:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
</am:Input>
<am:Output>
<am:Transform source="c10:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
</am:Output>
</am:Assign>

```

Figure 6.51 : Règles de transformation des entrées/sorties du SMA C10

Comme présenté à la figure 6.51, le SMA C10 prend en entrée une liste des besoins, il va aider un utilisateur à les affiner pour délivrer une liste de besoins mise à jour en sortie. Ce SMA est présent deux fois dans l'arbre d'exécution sous les identifiants C10_1 et C10_2. Pour ces deux identifiants, il possède le même type de règles de transformation.

Enfin, le SMA C11 propose un guide pour affiner la planification du projet et possède les transformations de produit suivantes :

```

<am:AssignparentId="D9_1" leafId="C11_1">
<am:Input>
<am:Transform source="mlpm:Livrable" target="c11:Livrable"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
<am:Transform source="mlpm:Acteur" target="c11:Acteur"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
<am:Transform source="mlpm:ListeDesBesoins" target="c11:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
</am:Input>
<am:Output>
<am:Transform source="c11:Livrable" target="mlpm:Livrable"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
<am:Transform source="c11:Developpe" target="mlpm:Developpe"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
<am:Transform source="c11:Acteur" target="mlpm:Acteur"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
<am:Transform source="c11:Participant" target="mlpm:Participant"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
</am:Output>
</am:Assign>

```

Figure 6.52 : Règles de transformation des entrées/sorties du SMA C11

Il est nécessaire d'avoir comme donnée d'entrée pour le SMA C11 une instance du premier "livrable" à réaliser, ainsi que l'ensemble des acteurs existants et la liste des besoins du projet. A partir de ces données, il va guider un utilisateur pour l'affectation des acteurs au premier "livrable", puis pour la définition des premières itérations de développement d'une partie des besoins de ce "livrable". Ce SMA délivre donc en sortie les données du premier "livrable" mise à jour avec des itérations, la nouvelle liste des acteurs avec leur participation dans ce "livrable" et les besoins développés dans les itérations (voir figure 6.52).

6.7. Conclusion

Nous avons présenté dans ce chapitre un cas d'utilisation d'une ligne de méthode d'un lancement de projet agile de développement de SI. Ce cas d'application présente la construction de la ligne depuis son acquisition à partir des processus de trois méthodes agiles : DSDM, Scrum et XP. Sont détaillés ensuite la structure de chaque SMA utilisés dans cette ligne et les composants conceptuels relatifs à ces SMA.

La ligne de méthode d'un lancement de projet agile est ensuite opérationnalisée par les trois modèles : l'arbre d'exécution de la ligne correspondant au modèle d'acquisition de son processus, le guide de configuration pour cet arbre d'exécution fournissant une aide aux utilisateurs dans leurs décisions vis-à-vis de l'exécution de la ligne et enfin le modèle de transformation des entrées/sorties de la ligne reposant sur l'implémentation au format XSD du modèle de produit de la ligne et de ses SMA.

Chapitre 7. Plateforme MaaS

7.1. Introduction

Afin de faciliter l'usage des services méthodologiques par les entreprises, nous proposons dans la solution MaaS, une plateforme permettant la recherche et l'invocation des SMA et des LM. Cette plateforme applique une architecture AOS et elle fournit à un utilisateur des facilités (1) de recherche des SMA sur un annuaire de service méthodologique, (2) d'invocation des services méthodologiques et (3) de configuration et d'exécution de ligne de méthode. C'est pour cela que nous proposons de fragmenter cette plateforme en plusieurs composants autonomes et de leur affecter un rôle précis en fonction des fonctionnalités de la plateforme. L'architecture globale de la plateforme de l'approche MaaS est ainsi constituée d'un module de recherche de SM interagissant avec l'annuaire de SM. Elle est également constituée d'un module d'exécution de service méthodologique comprenant un module d'exécution des lignes de méthode, un module dédié à la gestion des interactions avec les utilisateurs.

Les deux solutions d'implémentation montrent la faisabilité de la plateforme dans deux environnements d'exécution différents.

En effet, l'environnement de développement de la plateforme de recherche et d'exécution de services méthodologiques (MaaS) dépend de la technologie orientée service choisie pour développer les SMA. Nous avons vu à la section 5.2.1.5 du chapitre 5 que les SMA peuvent être développés comme des portlets exécutables à distance (technologie WSRP). Néanmoins, une autre technique orientée service appelée OSGI (Open Service Gateway Initiative) (OSGI, 2011) satisfait les contraintes de développement des SMA. La particularité de cette technique est d'étendre dynamiquement les capacités d'une machine virtuelle en téléchargeant à la volée les services nécessaires et en les exécutants en local.

Le premier environnement d'exécution se compose de la technologie des portails Web et des portlets distants (WSRP) (OASIS/WSRP, 2008) afin de montrer un environnement web avec une exécution distante des SMA. L'usage de la technologie OSGI dans le développement de plugin Eclipse permet d'élaborer le deuxième environnement d'exécution possible. Cette deuxième solution permet de développer une plateforme MaaS intégrable à l'environnement de développement intégré (EDI) Eclipse (Eclipse, 2012) et les SMA sont alors téléchargés avant d'être exécutés en local.

La première implémentation de la solution repose une exécution distante de la plateforme et des services et utilise la technologie des portails web et des portlets WSRP (OASIS/WSRP, 2008). Elle prend alors la forme d'un portail où l'implémentation de chaque service méthodologique atomique est réalisée par un portlet. Un portlet est un service web et son interface graphique encapsulés dans un seul et même composant, le portlet. La plateforme est constituée d'une partie générique indépendante

de l'implémentation du portail sur laquelle elle est exécutée. Les fonctionnalités prises en compte par cette partie générique concernent la recherche des SMA et la gestion des LM. La partie spécifique de la plateforme regroupe les fonctionnalités nécessaires à l'invocation des SMA dans le portail. Ces SMA invoqués peuvent être issus de l'exécution d'une LM ou alors directement recherchés par un utilisateur dans l'annuaire de services méthodologiques. Les fonctionnalités d'invocation de portlets étant propres à chaque implémentation de portail, cela implique que cette partie de notre plateforme soit spécifique au portail sur laquelle elle est exécutée.

La deuxième implémentation de la plateforme MaaS est destinée à une utilisation dans des environnements de développement intégré d'application reposant sur la technologie OSGI. La plateforme MaaS est développée dans cette solution comme un composant appelé plugin intégrable à l'environnement de développement Eclipse. Cet environnement utilise la technologie OSGI pour fournir un environnement de développement modulaire et orienté service. Les services correspondent ici à des composants logiciels appelés plugins téléchargeables à partir du fournisseur et exécutés sur la machine de l'utilisateur. Dans ce type d'implémentation de la plateforme, les SMA sont également réalisés comme des plugins Eclipse exposés sur un réseau. Le modèle d'application Eclipse est un ensemble de plugins interdépendants et la technologie OSGI permet de gérer dynamiquement durant l'exécution de l'application ces dépendances entre les plugins. Cela nous permet de construire la plateforme MaaS de manière modulaire comme un ensemble de plugins Eclipse.

Ce chapitre présente dans un premier temps à la section 7.2 les spécifications générales de la plateforme puis dans un deuxième temps les spécifications détaillées de chacun de ses modules. La section 7.3 illustre notre proposition par un prototype de plateforme MaaS utilisant les technologies des portails web et WSRP (OASIS/WSRP, 2008). Un second prototype est présenté à la section 7.4 de ce chapitre. Il prend quant à lui la forme d'une implémentation de la plateforme MaaS à l'aide de la technologie OSGI dans l'environnement Eclipse.

7.2. Une plateforme pour l'approche MaaS

7.2.1 Spécifications de la plateforme de recherche et d'exécution de services méthodologiques

La plateforme d'exécution de l'approche MaaS fournit un environnement d'exécution pour les services méthodologiques qu'ils soient sous la forme de SMA ou de ligne de méthode. D'une manière générique, cette plateforme repose sur les principes d'une architecture AOS et permet l'orchestration de services interactifs. L'architecture de la plateforme en elle-même est présentée à la figure 7.1 et décrit les principaux modules de la plateforme MaaS ainsi que leurs interactions.

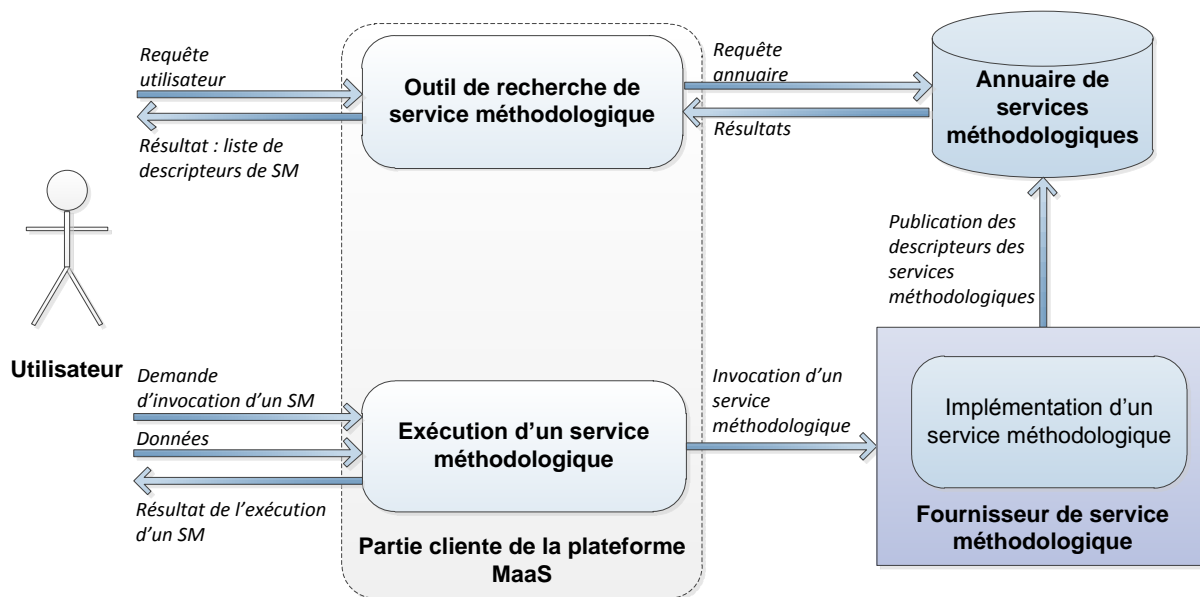


Figure 7.1 : Architecture générale de la plateforme d'exécution d'une ligne de méthode

L'architecture de la plateforme se compose de trois parties principales : un fournisseur de SM, un annuaire de SM, une partie client permettant de rechercher et d'exécuter des SM (voir figure 7.1).

Les fournisseurs de SM développent des services méthodologiques, c'est-à-dire des implémentations de SMA ou de ligne de méthode. Ils produisent ensuite leurs descripteurs à l'aide de l'ontologie OMCM et d'ontologies de domaine. Puis ils publient ces descripteurs sur l'annuaire de service méthodologique afin de les rendre accessible au plus grand nombre d'utilisateurs.

L'annuaire de service méthodologique fournit des fonctionnalités de publication, de stockage et d'indexation des descripteurs de SM. Cette centralisation des descripteurs en un seul point de stockage permet de faciliter la recherche de SM pour les utilisateurs. D'un point de vue des utilisateurs l'annuaire doit posséder des facilités de recherche des SM, pour cela il fournit une interface de requêtage de SM accessible à distance. Concernant la résolution des requêtes, une implémentation d'un algorithme de recherche permet de calculer la similarité sémantique entre une requête et les descripteurs de SM indexés dans l'annuaire. Cette correspondance est calculée à partir des concepts de l'ontologie OMCM et des ontologies de domaine utilisés comme annotation dans les requêtes et les descripteurs de SM. L'annuaire trie ensuite les descripteurs de SM en fonction des résultats de ces calculs et retourne une liste de descripteurs ordonnés par pertinence en réponse à une requête.

La partie cliente de la plateforme MaaS est constituée d'un outil de recherche de SM et d'une partie dédiée à l'exécution des SM.

L'outil de recherche a pour objectif de faciliter les interactions entre un utilisateur et l'annuaire de SM. En effet, il propose des facilités pour assister l'utilisateur à formuler une requête de recherche de SM. Il construit ensuite la requête destinée à l'annuaire à partir des informations saisies par l'utilisateur.

Après l'envoi de la requête à l'annuaire et la récupération des résultats, l'outil de recherche a la charge de présenter ces résultats afin de faciliter leur lecture et aider ainsi l'utilisateur à choisir un SM.

La partie cliente de la plateforme MaaS dédiée à l'exécution des SM fournit les fonctionnalités d'invocation des lignes de méthodes et des SMA. Elle gère également les interactions avec les utilisateurs nécessaires à l'exécution des SM. La figure suivante présente les différents modules logiciels constituant la partie cliente de la plateforme MaaS dédiée à l'exécution des SM.

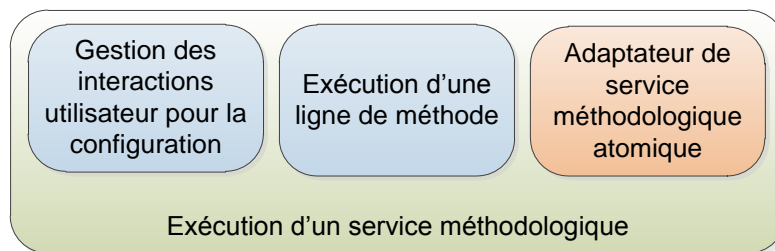


Figure 7.2 : Architecture générale de la plateforme d'exécution d'une ligne de méthode

Trois modules logiciels sont utiles pour de l'exécution d'un SM (voir figure 7.2) : le module de gestion des interactions utilisateurs, le module d'exécution d'une ligne de méthode et l'adaptateur de service méthodologique atomique.

La gestion de l'exécution d'une ligne de méthode effectuée par le module d'exécution comprend dans un premier temps la récupération des modèles opérationnels de la ligne à partir de son descripteur et leur interprétation. Dans un deuxième temps, l'exécution d'une ligne de méthode nécessite la gestion de l'exécution de son processus pas à pas en offrant à l'utilisateur le choix entre les différentes alternatives du processus à chaque point de variation. Après chaque exécution d'un SMA de la ligne, le produit résultant de son exécution est intégré au produit global de la ligne stocké en mémoire dans le module d'exécution d'un SM par l'application des règles de transformations concernant ce SMA dans le modèle de transformation de la ligne. Au cours de ce processus le module de gestion des interactions utilisateur gère l'ensemble des interactions avec l'utilisateur telles que l'affichage de l'état courant du processus, l'affichage des guides de configuration contextuels à l'état du processus pour aider l'utilisateur à faire ses choix et la saisie des choix de l'utilisateur.

Une fois qu'un SMA a été sélectionné pour être exécuté dans une ligne de méthode ou directement depuis l'outil de recherche, l'url du descripteur est transmise à l'adaptateur de service méthodologique atomique ce qui lui permet d'invoquer l'implémentation du SMA. Une fois invoqué sur la plateforme MaaS un SMA peut avoir besoin de données d'entrées pour pouvoir être exécuté. Ces données peuvent être fournies par le module d'exécution dans le cas d'un SMA issu d'une ligne de méthode ou saisies par l'utilisateur dans le cas d'un SMA invoqué directement depuis l'outil de recherche. A la fin de l'exécution d'un SMA le produit résultat est transmis à l'utilisateur et si besoin il est intégré au produit global d'une ligne de méthode.

7.2.2 Recherche des services méthodologiques

L'outil de recherche est constitué de deux modules logiciels : un module de construction de requêtes et un module de gestion des interactions avec l'annuaire de SM. Comme présenté à la figure suivante ces deux modules interagissent pour aider l'utilisateur à formuler des requêtes afin de rechercher des SM sur l'annuaire.

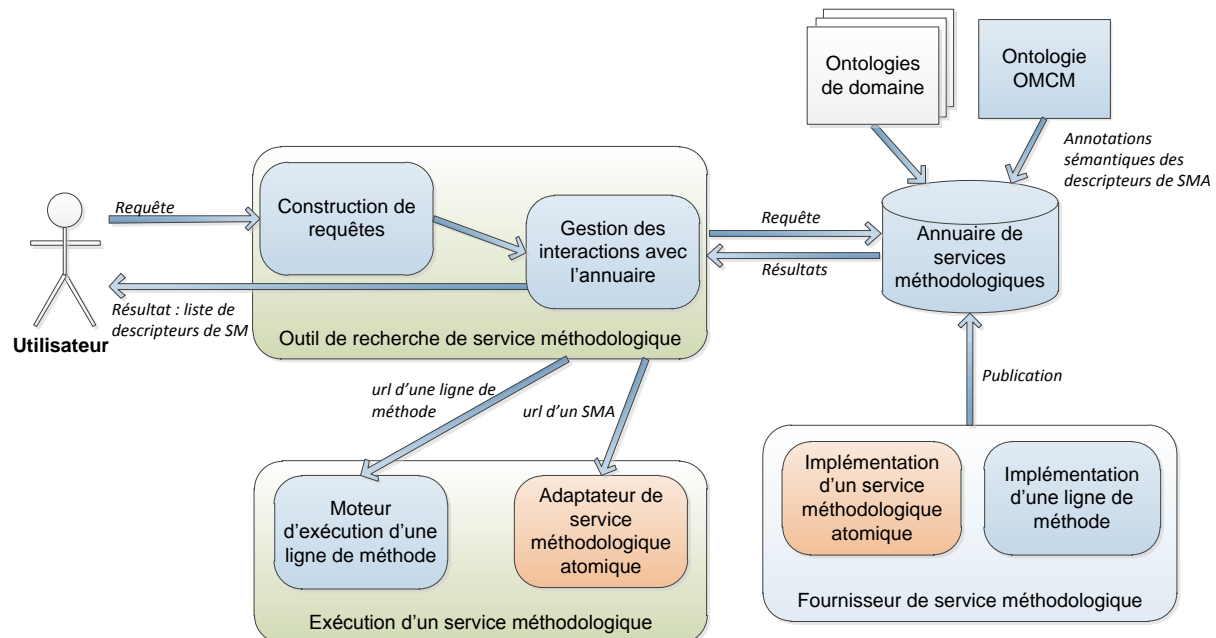


Figure 7.3 : Scénario d'utilisation de l'outil de recherche de la plateforme MaaS

Le module de construction de requêtes a pour objectif de faciliter un utilisateur à formuler une requête. Pour cela, il peut aider graphiquement l'utilisateur à choisir des concepts dans l'ontologie OMCM et de proposer un ensemble de concepts d'ontologie de domaine présélectionnés. Il peut également implémenter un processus de guidage de construction d'une requête orienté vers l'utilisateur en faisant abstraction du format technique de la requête. Une fois que l'utilisateur ai saisi l'ensemble des paramètres de la requête ce module construit la requête dans le format technique attendu par l'annuaire tel que nous l'avons défini dans le chapitre 4 de cette thèse. Il transmet ensuite la requête destinée à l'annuaire au module de gestion des interactions avec l'annuaire.

Le module de gestion des interactions avec l'annuaire a la connaissance de l'interface d'accès à distance de l'annuaire de SM. Il encapsule donc la requête qui lui a été transmise par le module de construction de requêtes dans le format de message attendu par l'interface d'accès à distance de l'annuaire et lui envoie. Il récupère en réponse un message contenant la liste des descripteurs de SM sélectionnés par l'algorithme de recherche sémantique de l'annuaire. Le module de gestion des interactions avec l'annuaire a la charge d'interpréter ces résultats et de les présenter à l'utilisateur de manière à faciliter ses choix entre les différents services proposés. L'utilisateur peut alors effectuer son choix de SM à exécuter, puis le module de gestion des interactions avec l'annuaire transfère l'adresse

du descripteur (URL) du SM à exécuter en fonction de son type (LM ou SMA) soit au module d'exécution d'une ligne de méthode soit à l'adaptateur de SMA.

Concernant le fournisseur de service méthodologique, son rôle est de développer les implémentations des SMA et des LM puis de constituer leurs descripteurs par des annotations à l'aide de l'ontologie OMCM et d'ontologies de domaine. L'implémentation des SMA est néanmoins dépendante de la technologie utilisée pour l'implémentation de la plateforme MaaS. Après avoir réalisé les implémentations et les descripteurs des SM, un fournisseur doit les publier sur l'annuaire de SM. Pour cela, il doit avoir connaissance de l'interface de publication de l'annuaire et construire un message de publication de SM conforme au format attendu par l'annuaire.

Du point de vue de l'annuaire, il doit fournir des facilités de publication et de recherche de SM accessible à distance. Il doit donc définir une interface de publication et une interface de recherche pour son accès distant. Ces interfaces comprennent la définition des méthodes de recherche et de publication disponible ainsi que les formats de message de requête et de réponse de ces méthodes. D'un point de vue interne, l'annuaire implémente une base de stockage pour les descripteurs des SM et des informations sur les fournisseurs qui les ont implémentés. En complément à cette base l'annuaire implémente également un algorithme de recherche sémantique tel que par exemple celui proposé par l'approche YASA (Chabeb, 2008) et (Chabeb, 2010). Cet algorithme permet de calculer la similarité sémantique entre une requête de recherche de SM et des descripteurs de SM. Pour cela il compare les annotations sémantiques effectuées à l'aide de l'ontologie OMCM et d'ontologies de domaine dans la requête avec celles des descripteurs. Enfin, il construit une liste résultat de descripteurs de SM ordonnés en fonction de leur similarité sémantique avec la requête. Cette liste est ensuite retournée à l'initiateur de la requête, c'est-à-dire le module de gestion des interactions avec l'annuaire dans la plateforme MaaS.

7.2.3 Exécution d'un SMA

L'exécution d'un SMA dans la plateforme MaaS est prise en charge par le module logiciel nommé adaptateur de service méthodologique. Comme présenté à la figure 7.4, ce module interagit également avec le module d'exécution d'une ligne de méthode et l'outil de recherche de SM.

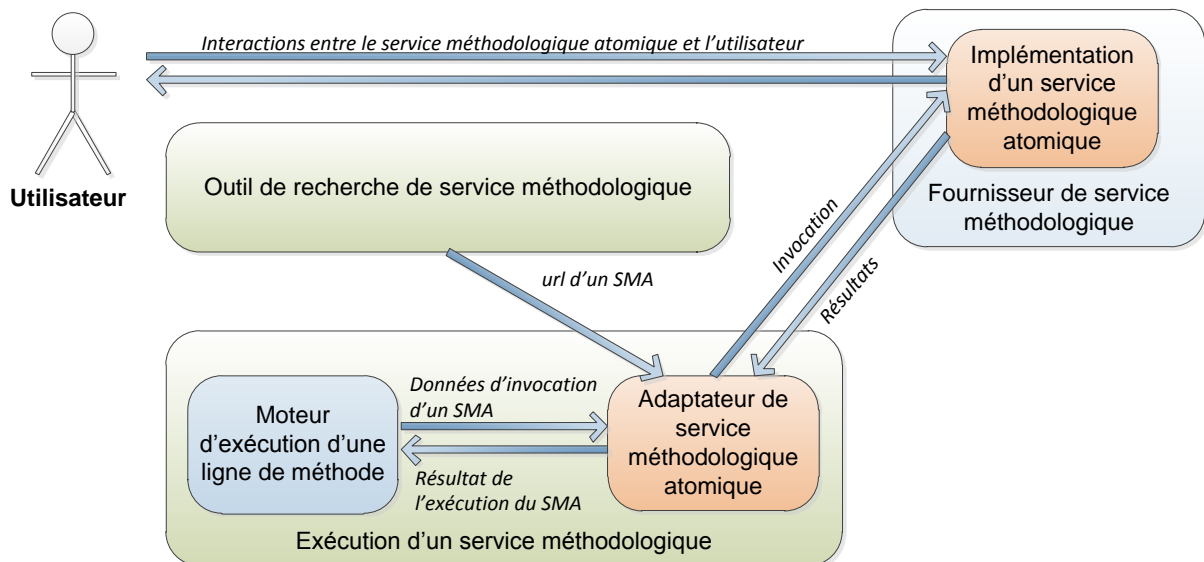


Figure 7.4 : Scénario d'exécution d'un SMA sur la plateforme MaaS

L'adaptateur de SMA a pour fonctionnalité l'invocation d'un AMS dans l'environnement d'exécution de la plateforme MaaS. Les méthodes d'invocation de service interactif sont propres à chaque environnement et font appel aux fonctionnalités de bas niveau de la technologie utilisée. Ce module est par conséquent fortement dépendant de l'environnement dans lequel la plateforme est exécutée et, si cet environnement change, il sera alors nécessaire de redévelopper ce module. Pour pouvoir invoquer un SMA ce module prend en tant que paramètre d'entrée les données techniques d'invocation du SMA ainsi que les produits nécessaires à son fonctionnement une fois l'invocation réalisée. Ces données d'invocation peuvent lui être transmises soit par l'outil de recherche de SM, soit par le module d'exécution d'une ligne de méthode. Un SMA étant un service interactif, l'utilisateur interagit directement avec ce dernier. Par conséquent, durant l'exécution d'un SMA, ce module est mis en attente de la réception du produit résultat du SMA. Lorsqu'il le reçoit, il le transmet à l'utilisateur et si besoin au module d'exécution d'une ligne de méthode, puis révoque le SMA.

7.2.4 Exécution d'une ligne de méthode

L'exécution d'une ligne de méthode est gérée dans la plateforme MaaS par le module de gestion des interactions utilisateur pour la configuration et le module d'exécution d'une ligne de méthode. La figure 7.5 illustre les différentes interactions entre ces deux modules et les autres modules logiciels de la plateforme.

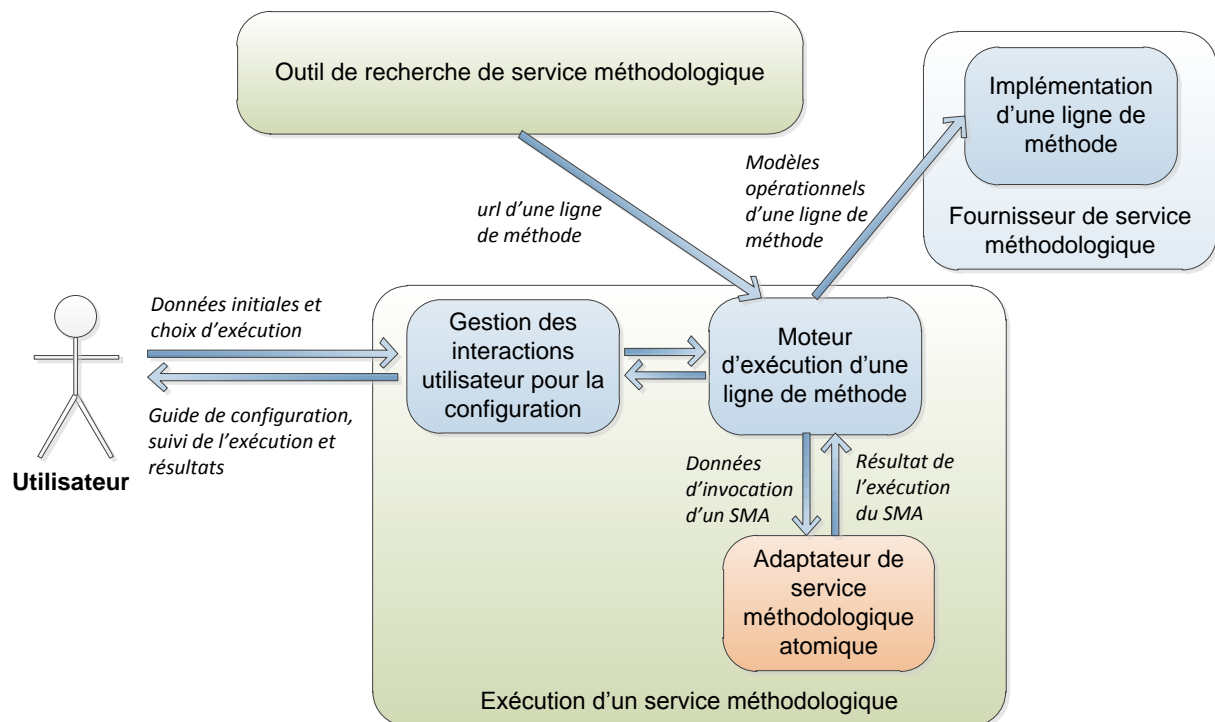


Figure 7.5 : Scénario d'exécution d'une ligne de méthode sur la plateforme MaaS

Le moteur d'exécution d'une ligne de méthode récupère la description du service méthodologique à partir de son adresse qui lui a été transmise (voir figure 7.5). Après avoir vérifié la validité du descripteur du service méthodologique, il regarde de quel type est le service : atomique ou ligne de méthode. Si le service méthodologique est atomique il transmet ses données d'invocation à l'adaptateur de SMA. Dans le cas où le service méthodologique est une ligne de méthode, le module récupère alors les trois modèles nécessaires à l'exécution de la ligne : l'arbre d'exécution, le modèle de configuration et le modèle des transformations de produit. Après la validation de ces modèles, l'arbre d'exécution de la ligne de méthode est chargé en mémoire et transmis au module d'interactions utilisateur pour être affiché. Ensuite, les données d'entrée de la ligne sont lues à partir du modèle de transformations de produit et transmises au module d'interaction pour être saisies par un utilisateur. Après cette étape d'initialisation, l'exécution de la ligne est lancée. En d'autres termes, le parcours de l'arbre d'exécution est initié.

A chaque nœud décisionnel de l'arbre d'exécution, les guides de configuration sont transmis au module d'interactions pour être affichés, le choix de l'utilisateur est alors récupéré puis reporté par le module d'exécution de la ligne par la mise à jour de l'état du parcours dans l'arbre.

Lorsqu'un SMA doit être exécuté dans l'arbre, ses données d'invocation sont transmises au module spécifique et le module générique est alors mis en attente jusqu'à la réception du produit résultat. Le produit résultat d'exécution du SMA est alors intégré au produit de la ligne selon les transformations décrites dans le modèle de transformation de produit et l'état du parcours dans l'arbre est mis à jour. Ces mises à jour sont également transférées au module de gestion des interactions pour leur affichage.

Les SMA sont donc invoqués les uns après les autres en fonction des choix de l'utilisateur et de leur position dans l'arbre d'exécution.

Lorsque le parcours de l'arbre arrive à son terme, le moteur d'exécution envoie le produit final de la ligne de méthode au module interagissant avec l'utilisateur pour qu'il le lui transmette.

Le module de gestion des interactions avec l'utilisateur fournit une interface avec la plateforme MaaS pour un utilisateur. Ce module doit posséder des fonctionnalités de saisie des produits en entrée d'une ligne de méthode à exécuter ainsi qu'une fonctionnalité de consultation de ses données (descripteur MaaS et composant conceptuel). Il doit également fournir une fonctionnalité d'affichage de l'arbre d'exécution d'une ligne de méthode ainsi que son état courant. Toujours dans l'exécution d'une ligne, ce module fournit au niveau de chaque nœud de l'arbre les mécanismes d'affichage des guides de configuration associés au nœud et une fonctionnalité de capture des décisions des utilisateurs. A la fin de l'exécution d'un service méthodologique, le module de gestion des interactions avec l'utilisateur récupère son résultat et le propose à l'utilisateur.

7.2.5 Caractéristiques de l'architecture

L'architecture de la plateforme MaaS peut être implémentée de plusieurs manières en utilisant des technologies différentes. En effet, la solution présentée à la section 7.2 de ce chapitre est la vision conceptuelle et architecturale de la plateforme. Concernant l'implémentation de celle-ci, les modules logiciels la constituant présentent des niveaux de réutilisabilité différents. En d'autres termes, certains modules sont réutilisables dans la plupart des implémentations (même langage de développement), d'autres sont réutilisables sur les plateformes implémentées avec la même technologie (par exemple portail web ou encore OSGI) et les modules non réutilisables sont spécifiques à leur environnement d'exécution (par exemple un type de portail ou un outil de développement intégré précis).

Les modules logiciels réutilisables de la partie cliente de la plateforme prennent la forme de bibliothèques logicielles contenant les algorithmes du cœur de celle-ci. Ce niveau de réutilisabilité comprend les modules suivants (représentés en bleu aux figures 7.2, 7.3, 7.4 et 7.5) : construction des requêtes, gestion des interactions avec l'annuaire, gestion des interactions utilisateur pour la configuration et exécution d'une ligne de méthode. L'implémentation d'une ligne de méthode étant un ensemble de modèles opérationnels implémentés en XML, elle est totalement indépendante des technologies utilisées pour le développement de la plateforme.

Du point de vue de l'annuaire de SM, c'est un composant autonome de l'architecture de la plateforme MaaS. Les accès à l'annuaire étant distants, c'est-à-dire que les interactions entre l'annuaire et la plateforme sont réalisés par des messages transitant sur un réseau. L'implémentation de l'annuaire est par conséquent entièrement dissociée de celle de la plateforme et elle est indépendante des technologies employées par cette dernière.

Les modules de la partie cliente de plateforme MaaS étant dépendants des technologies utilisées par la plateforme sont l'outil de recherche de SM et le module d'exécution d'un SM (représentés en vert aux figures 7.2, 7.3, 7.4 et 7.5). Ces deux modules sont la couche de présentation permettant à un utilisateur d'interagir avec la plateforme, ils sont le ciment entre les différentes briques logicielles réutilisables de la plateforme.

Le module non réutilisable de la plateforme est l'adaptateur de SMA (représentés en orange aux figures 7.2, 7.3, 7.4 et 7.5). En effet ce module doit faire appel aux fonctionnalités de bas niveau de l'environnement d'exécution de la plateforme pour l'invocation des SMA. Par conséquent l'adaptateur est implémenté pour un environnement d'exécution précis. Il en va de même pour l'implémentation d'un SMA, elle est spécifique à son environnement d'exécution.

7.3. Réalisation d'un prototype avec les technologies des portails web et WRSP

Dans le cadre de cette thèse, nous avons choisi de prendre pour le premier prototype une solution orientée portail web, avec le portlet (OASIS/WSRP, 2008) comme élément de base capturant un service interactif. D'une manière générale, un portail est un site sur lequel un utilisateur peut s'authentifier et organiser son propre espace de travail par l'utilisation de services interactifs appelés des portlets. Ce type d'implémentation permet à la plateforme MaaS d'être intégrée à une interface web et avoir l'avantage d'être exécutable et accessible à distance. De plus, le portlet offre directement la possibilité d'implémenter un service interactif, c'est-à-dire l'encapsulation dans un même composant logiciel de la couche de présentation, de la gestion des interactions avec l'utilisateur et de l'exécution ou l'invocation d'un service métier. Dans l'application de ces principes pour la construction d'une plateforme d'exécution de services méthodologique, les SMA sont implémentés sous la forme de portlets. La partie cliente de la plateforme MaaS se présente alors sous la forme d'un portail web avec les modules de recherche SM, d'exécution d'un SM et l'adaptateur de SM implémentés par trois portlets distinct réalisés en langage Java (ORACLE/JAVA, 2011) respectant la norme JSR 286 (ORACLE, 2008). La structure un portlet de la plateforme implémentant cette norme est présentée à la figure 7.6.

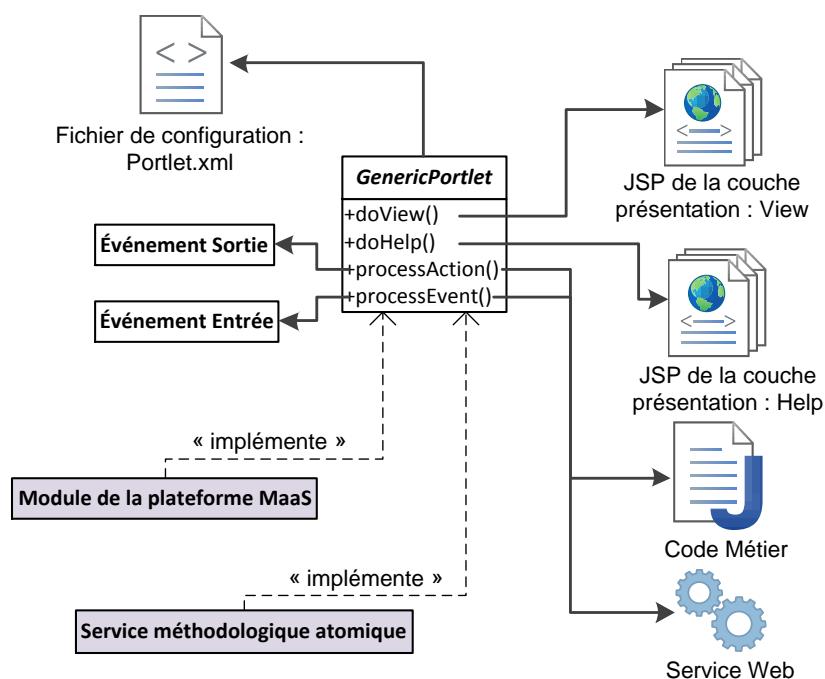


Figure 7.6 : Spécifications d'un portlet Java de la plateforme MaaS

Les trois modules de cette plateforme MaaS réalisée avec la technologie WSRP (OASIS/WSRP, 2008) ainsi que les SMA sont des implémentations de l'interface "GenericPortlet". En implémentant cette interface, ces portlets fournissent une implémentation pour les méthodes génériques définies dans l'interface. Cela permet au portail de manipuler les portlets au travers de l'interface "GenericPortlet" et de la signature des méthodes définies dans cette dernière. Dans cette implémentation de la plateforme nous utilisons les méthodes "doView", "doHelp", "processAction" et "processEvent" de cette interface (voir figure 7.6).

La couche de présentation du portlet est séparée en plusieurs parties et est gérée par les méthodes "doView" et "doHelp". La méthode "doView" gère l'enchaînement des différentes vues constituant la couche de présentation des fonctionnalités métiers du portlet. Chacune de ces vues est implémentée par une page serveur au format JSP (ORACLE/JSP, 2006). La méthode "doHelp" gère de la même manière la couche de présentation des aides et des guides destinés à l'utilisateur.

Les interactions avec l'utilisateur sur les différentes vues du portlet sont prises en charge par la méthode "processAction" qui va gérer les événements utilisateurs et déclencher les traitements correspondants.

La communication entre les portlets invoqués dans un même portail est réalisée d'une manière événementielle de type publication/souscription. En d'autres termes, un portlet va publier des événements, ce sont des événements en sortie du portlet et il va souscrire à des événements, ce sont les événements en entrée du portlet. Lorsqu'un événement est publié par un portlet au niveau du portail,

tous les portlets ayant souscrit à cet événement reçoivent une notification et peuvent déclencher un traitement. Les événements en eux même sont implémentés par des classes Java standard ("Événement entrée" et "Événement sortie" à la figure 7.6), ils peuvent contenir des données et par conséquent doivent être sérialisables. La publication des événements est implémentée dans la méthode "processAction" et le traitement d'un événement pour lequel le portlet a souscrit est implémenté par la méthode "processEvent". La déclaration des événements publiés et souscrits par un portlet est effectuée dans un fichier de configuration écrit en langage XML et nommé "portlet.xml". Concernant la plateforme MaaS la communication entre les différents modules de la plateforme est implémentée par ce type d'événement. Il en va de même pour les SMA avec les données en entrée nécessaire à leur fonctionnement et avec les données qu'ils fournissent en sortie. Ces données de type XML sont transmises par deux événements utilisés par tous les SMA dans cette plateforme: un événement d'entrée de SMA et un événement de sortie de SMA.

Les traitements en eux même ne sont pas directement effectués par le portlet, ils sont délégués à des modules métiers pouvant prendre la forme de services web SOAP pour les SMA ou de classes Java pour les modules de la plateforme MaaS. Le rôle des méthodes "processAction" et "processEvent" est de choisir le traitement à effectuer en fonction de l'événement (utilisateur ou externe) reçu.

Le module de recherche de SM de la plateforme MaaS est implémenté par un portlet (voir figure 7.7) utilisant les modules de construction de requêtes et de gestion des interactions avec l'annuaire sous la forme de librairie JAVA. L'implémentation de ce portlet est indépendante du portail respectant la norme JSR 286 (ORACLE, 2008) sur lequel il sera exécuté.

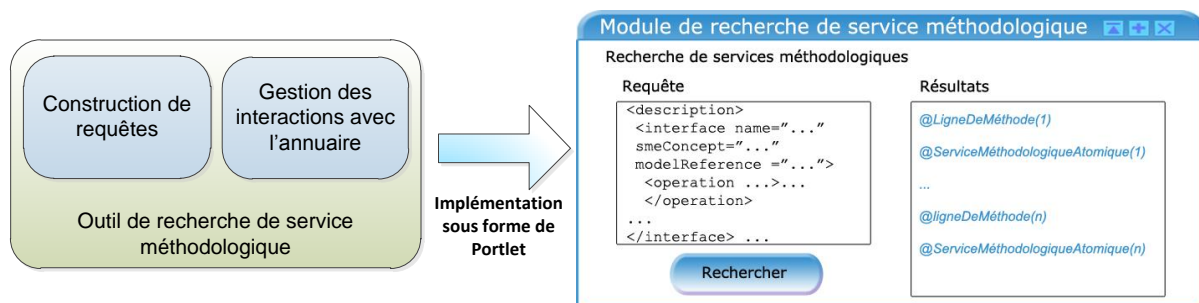


Figure 7.7 : Implémentation sous forme de portlet de l'outil de recherche de SM

Comme présenté à la figure 7.7, le portlet implémentant l'outil de recherche de SM fournit à l'utilisateur une interface lui permettant de saisir une requête de recherche de SM, d'afficher des résultats et de choisir un SM à exécuter parmi les résultats.

Le module d'exécution des SM est quant à lui découpé en deux portlets : un dédié à l'exécution des lignes de méthode et un dédié à l'invocation des SMA dans le portail comme présenté aux figures 7.8 et 7.9.

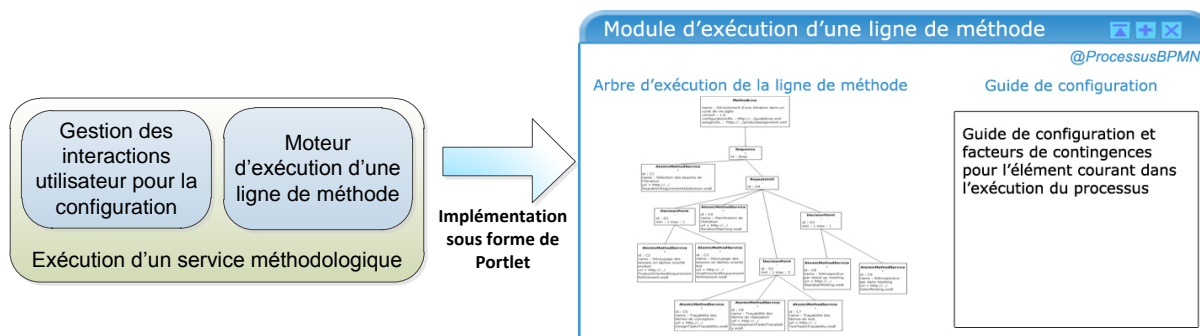


Figure 7.8 : Implémentation sous forme de portlet du module d'exécution d'une ligne de méthode

Le portlet implémentant le module d'exécution d'une ligne de méthode est un portlet générique pouvant être exécuté sur n'importe quel type de portail. Comme présenté à la figure 7.8, ce portlet intègre le module d'interaction avec les utilisateurs et le module d'exécution d'une ligne de méthode sous la forme de bibliothèques JAVA manipulées dans les méthodes "processAction" et "processEvent" du portlet. L'invocation des SMA de la ligne est déléguée à l'adaptateur de SMA. La communication entre le moteur d'exécution d'une LM et l'adaptateur de SMA est réalisée au travers des événements entre portlets. En effet, le moteur d'exécution de LM publie un événement d'invocation de SMA et est consommé par l'adaptateur de SMA. A l'issue de l'exécution du SMA, l'adaptateur produit un événement de fin d'exécution consommé par le moteur d'exécution de LM. Ces deux événements contiennent les informations d'invocation et les données produit en entrée et en sortie du SMA.

La couche de présentation de l'exécution d'une LM est constituée de l'affichage de l'arbre d'exécution de la ligne et de la mise à jour dynamique de son état courant avec les guides de configuration liés à cet état. La partie aide de cette couche de présentation affiche le processus BPMN d'acquisition de la ligne afin que l'utilisateur puisse bénéficier d'une vue graphique conceptuelle du processus.

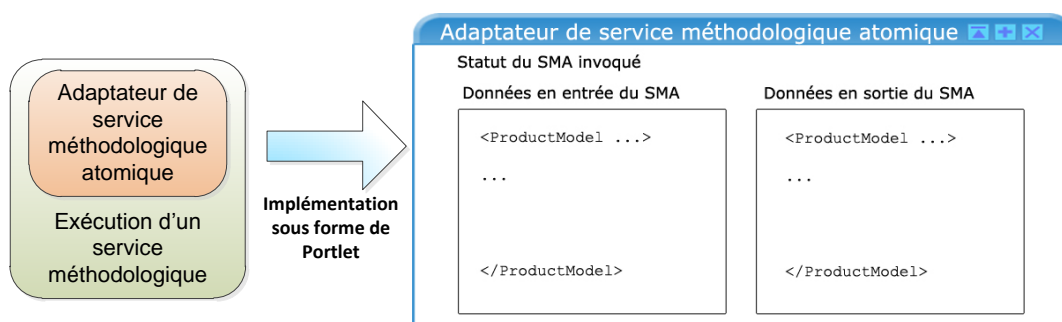


Figure 7.9 : Implémentation sous forme de portlet de l'adaptateur de SMA

L'adaptateur de SMA est implémenté par un portlet dépendant du portail sur lequel est exécutée la plateforme MaaS puisque que la fonctionnalité d'invocation d'un portlet est prise en charge par la couche de bas niveau du portail utilisé. Il interagit avec le portail pour invoquer ou révoquer les portlets implémentant les SMA. En pratique dans cette implémentation, il s'agit d'une extension du

consommateur de portlets WSRP fourni par le portail et adapté aux spécificités de l'approche MaaS. Les fonctionnalités proposées par ce portlet étant automatisées, sa partie de présentation est réduite à la trace de l'état courant de l'invocation et des entrées/sorties d'un AMS.

La plateforme MaaS est par conséquent un portail web constitué de plusieurs portlets (voir figure 7.10).

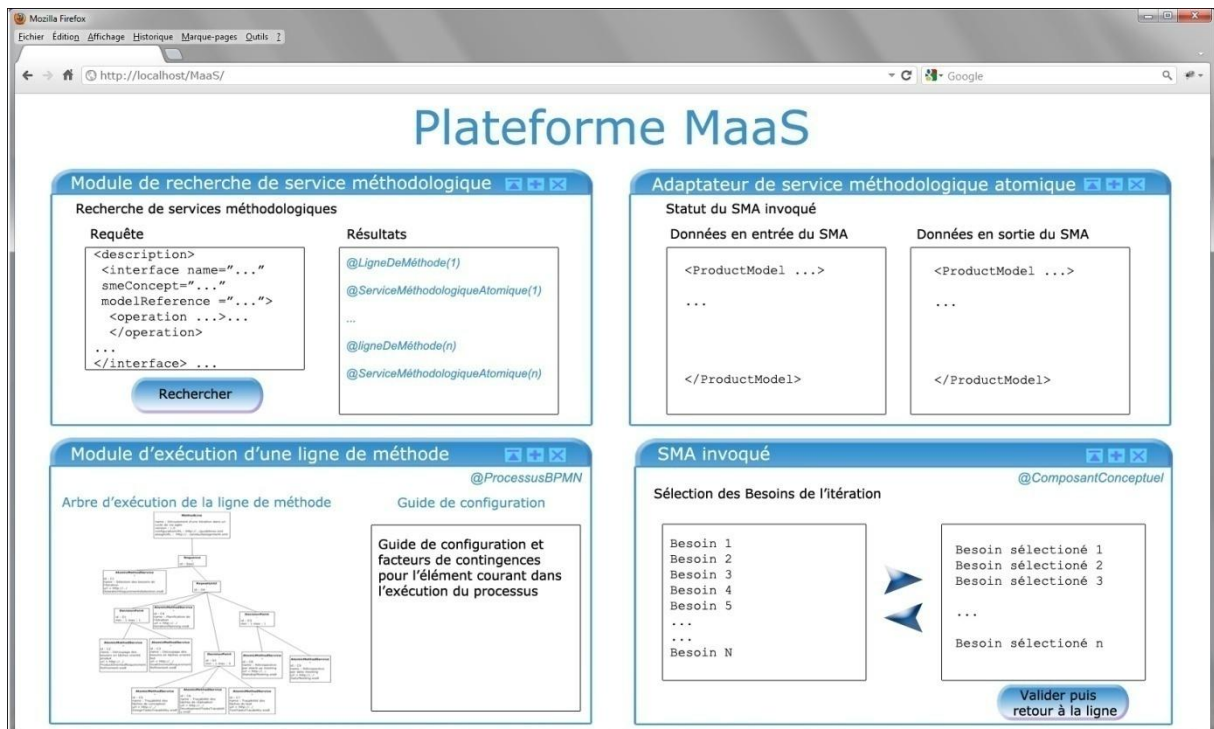


Figure 7.10 : Vue d'ensemble de la plateforme MaaS sous forme de portail web

Comme présenté à la figure 7.10, la plateforme MaaS est accessible à distance par un navigateur web. Cette vue d'ensemble de la plateforme comprend les trois portlets de la partie cliente : l'outil de recherche de SM, du module d'exécution d'une LM, de l'adaptateur de SMA et le portlet du SMA en cours d'exécution dans la ligne de méthode.

Cette première solution permet d'illustrer l'usage d'une technologie orientée service web SOAP au travers de la technologie des portails web et des portlets. Elle permet d'implémenter la plateforme MaaS par une architecture d'exécution *distribuée*. La deuxième solution correspond à une architecture d'exécution des SM *centralisée* sur le client (la machine de l'utilisateur).

7.4. Réalisation d'un prototype avec la technologie OSGI pour son utilisation dans un environnement de développement intégré

Le deuxième prototype de la plateforme MaaS est une solution orientée sur la technologie des composants logiciels Open Services Gateway Initiative (OSGI) (OSGI, 2011) pour une

implémentation de la plateforme réalisée dans un environnement de développement intégré (EDI). Les EDI (Eclipse (Eclipse, 2012), Netbeans (Netbeans, 2012) ...) sont largement utilisés dans les entreprises car ils constituent un regroupement modulable d'outils de développement de logiciels. La plateforme Eclipse a été retenue pour cette solution car c'est l'un des EDI les plus utilisés par les entreprises. De plus, les capacités de cet environnement sont extensibles et adaptables grâce au concept de plugin Eclipse. En effet, ce concept permet d'installer de nouveaux supports de développement dans un même environnement intégré. Cette solution permet de fédérer la plateforme MaaS qui outil est ciblé sur la configuration de méthode à l'environnement Eclipse centré sur le développement de logiciel. Elle fournit également la possibilité d'implémenter un service interactif à l'instar de la solution du premier prototype grâce aux concepts de plugin Eclipse et de composant logiciel OSGI. Les différents modules de la plateforme MaaS ainsi que les SMA sont implémentés dans cette solution sous la forme de plugins Eclipse reposant sur la technologie OSGI. En effet, l'environnement Eclipse est une couche logicielle construite au dessus de la technologie OSGI.

Cette technologie propose une plateforme d'exécution dynamique de composants logiciels indépendants. Un composant logiciel OSGI se présente sous la forme d'une archive exécutable possédant la structure suivante :

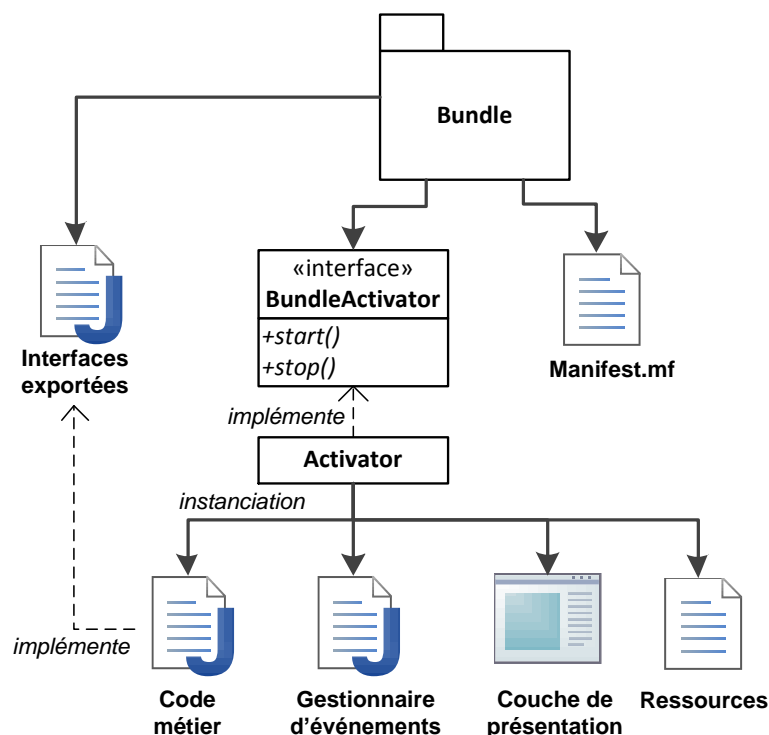


Figure 7.11 : Structure d'un composant logiciel OSGI

Comme présenté à la figure 7.11 un composant logiciel OSGI appelé "bundle" est un regroupement de classes Java auxquelles des "ressources" supplémentaires peuvent être ajoutées. Un composant OSGI

est associé à un descripteur nommé "manifest.mf". Ce descripteur regroupe les informations du composant telles que :

- son nom,
- un identifiant unique,
- une description,
- le numéro de version du format du descripteur,
- le numéro de version du composant,
- la référence de la classe java permettant d'initialiser le composant ("Activator"),
- les paquetages de classes contenant les services que le composant propose et
- les paquetages de classes à importer pour le fonctionnement du composant.

La classe "Activator" implémente l'interface "BundleActivator" et permet l'initialisation du composant OSGI. Elle contient les méthodes de lancement et d'arrêt d'un composant, elle permet de contrôler son cycle de vie. La classe "Activator" a pour rôle l'instanciation de l'ensemble des "classes métier", de la "couche de présentation" ainsi que des classes dédiées à la "gestion des événements". Elle va en plus gérer les dépendances entre les objets ainsi instanciés. Les services rendus par le composant OSGI sont décrits par une ou plusieurs "interfaces" et sont implémentés dans des "classes métiers". Les classes de la "couche de présentation" permettent de construire l'interface graphique du composant avec laquelle l'utilisateur interagit. Les interactions utilisateur sont gérées de manière événementielle et entraînent l'exécution de traitements en fonction de l'événement d'interface graphique produit.

L'intérêt de la technologie OSGI est de décrire les dépendances entre bundles dans un fichier de configuration et de résoudre ces dépendances durant l'exécution en les installant dynamiquement dans la machine virtuelle en fonction des besoins.

Intégrer la technologie OSGI à l'environnement Eclipse permet d'étendre les fonctionnalités d'eclipse à chaud en installant durant son exécution des plugins Eclipse comme des bundles OSGI.

L'environnement de développement intégré Eclipse repose sur cette technologie OSGI. En effet, Eclipse est constitué d'un assemblage de composants logiciels appelés "plugin Eclipse" et un plugin est une extension d'un "bundle" OSGI (voir figure 7.12).

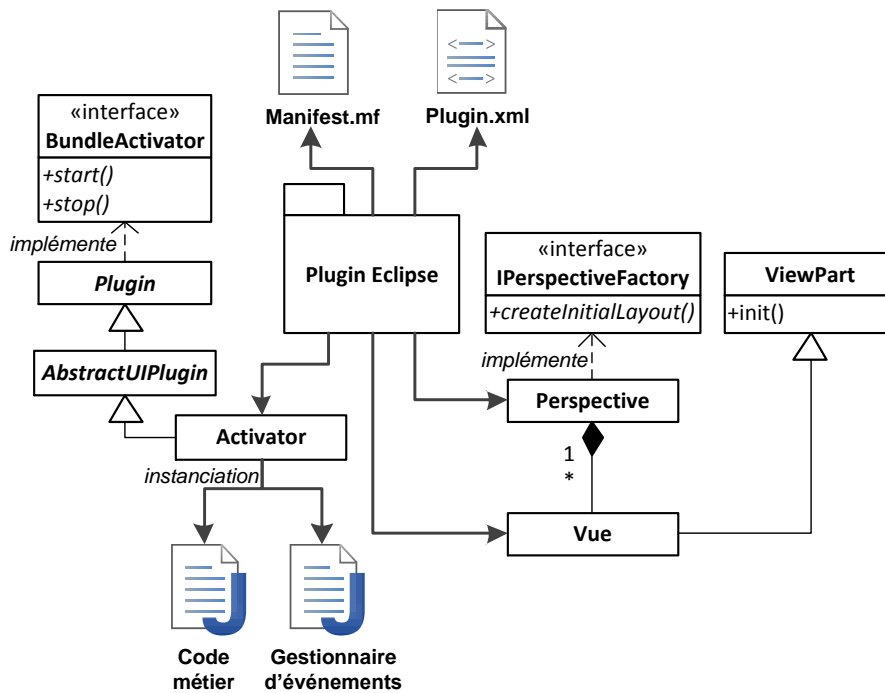


Figure 7.12 : Structure d'un plugin Eclipse

A l'instar d'un composant logiciel OSGI, un "plugin Eclipse" est un regroupement de classes Java et de ressources décrit par un fichier "manifest.mf" (voir figure 7.12). Le fichier de configuration "plugin.xml" est une extension du composant OSGI contenant des informations supplémentaires pour l'intégration du plugin dans l'architecture d'Eclipse, c'est-à-dire les données concernant les différentes "vues" et leur regroupement en "perspectives" constituant la couche de présentation du plugin. Ce fichier de configuration permet de décrire à l'aide du modèle d'application Eclipse la couche présentation du plugin. La classe "Activator" d'un plugin a un rôle similaire à une classe "Activator" d'un "bundle" OSGI excepté pour la couche présentation qui est automatiquement prise en charge par l'environnement Eclipse. Cette classe gère l'instanciation et les dépendances des différents objets composant le plugin. Elle doit également fournir des méthodes d'accès aux objets qu'elle a instanciés. Une classe "Activator" d'un plugin Eclipse est une implémentation d'une des classes abstraites proposées par la plateforme Eclipse, telle que par exemple la classe "AbstractUIPlugin" qui permet d'implémenter un "Activator" d'un plugin Eclipse contenant une partie graphique à afficher dans la plateforme. Ces classes abstraites sont dérivées de la classe "Plugin", elle aussi abstraite et qui implémente l'interface "BundleActivator" de OSGI. La couche de présentation d'un plugin est organisée en perspectives chacune regroupant un ensemble de vues. Une perspective est une classe Java implémentant l'interface "IPerspectiveFactory" proposée par Eclipse et elle prend en charge l'organisation de vues (positionnement et taille) dans un écran de l'application Eclipse. Une vue est une classe étendant la classe "ViewPart", également proposée par Eclipse. Elle organise les composants graphiques formant cette vue et fournit des méthodes d'abonnement aux classes dédiées à la gestion

événementielle. Concernant les classes métiers du plugin, c'est-à-dire celles contenant l'implémentation des services rendus par le plugin, elles sont prises en charge par la classe "Activator" de la même manière que pour un composant OSGI.

La communication entre les "plugins" est réalisée par le biais du service de sélection fourni par la plateforme Eclipse. Le service de sélection fonctionne comme un gestionnaire d'événement Java classique reposant sur le principe d'élément observateur et d'élément observé. En d'autres termes, un plugin recevant un événement contenant des données est un observateur du service de sélection et il implémente le traitement répondant à l'événement. Un plugin émetteur d'un événement est quant à lui l'élément observé, il doit signaler ses changements d'état par la production d'un événement envoyé au service de sélection.

D'un point de vue de la plateforme MaaS, cela signifie que les communications entre les composants logiciels de la partie cliente décrite à la figure 7.13 utilisent le service sélection à travers le principe observé/observateur.

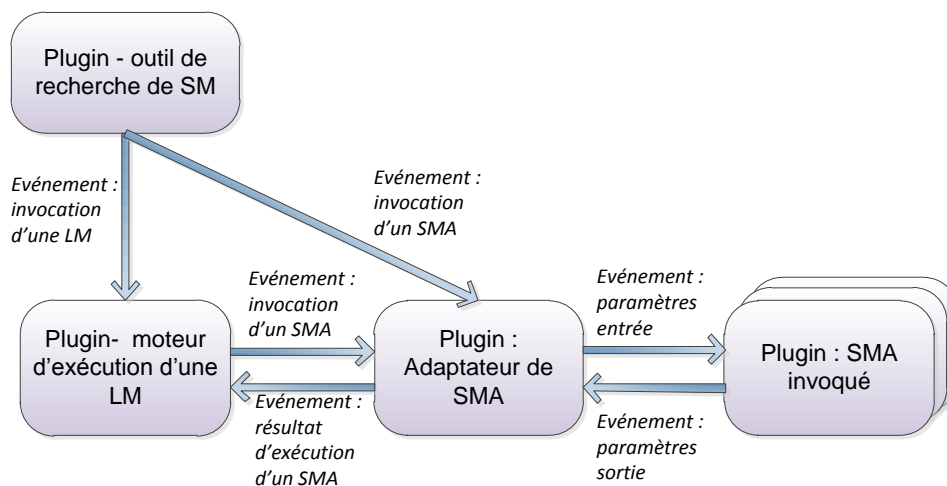


Figure 7.13 : Structure événementielle de la plateforme MaaS dans Eclipse

Comme illustré à la figure 7.13, le plugin recherche de SM joue le rôle d'observé car il peut envoyer un événement d'invocation de LM ou de SMA et les plugins d'exécution d'une LM et l'adaptateur de SMA sont ses observateurs car ils traitent l'événement d'invocation. D'un point de vue du plugin d'exécution d'une LM, il est observé par le plugin adaptateur de SMA pour l'événement d'invocation d'un SMA et inversement pour la restitution des résultats par le plugin adaptateur de SMA au plugin d'exécution d'une LM. Il en va de même pour les événements d'entrée et de sortie des SMA invoqués sur la plateforme par l'adaptateur de SMA.

La partie cliente plateforme MaaS est découpée dans cette solution en trois plugins Eclipse : recherche de SM, moteur d'exécution d'une LM et adaptateur de SMA. Ce découpage est similaire à celui effectué dans la solution reposant sur la technologie des portails et ces plugins disposent des mêmes

fonctionnalités que leurs portlets analogues. La figure 7.14 présente la vue d'ensemble de la plateforme MaaS dans l'environnement Eclipse.

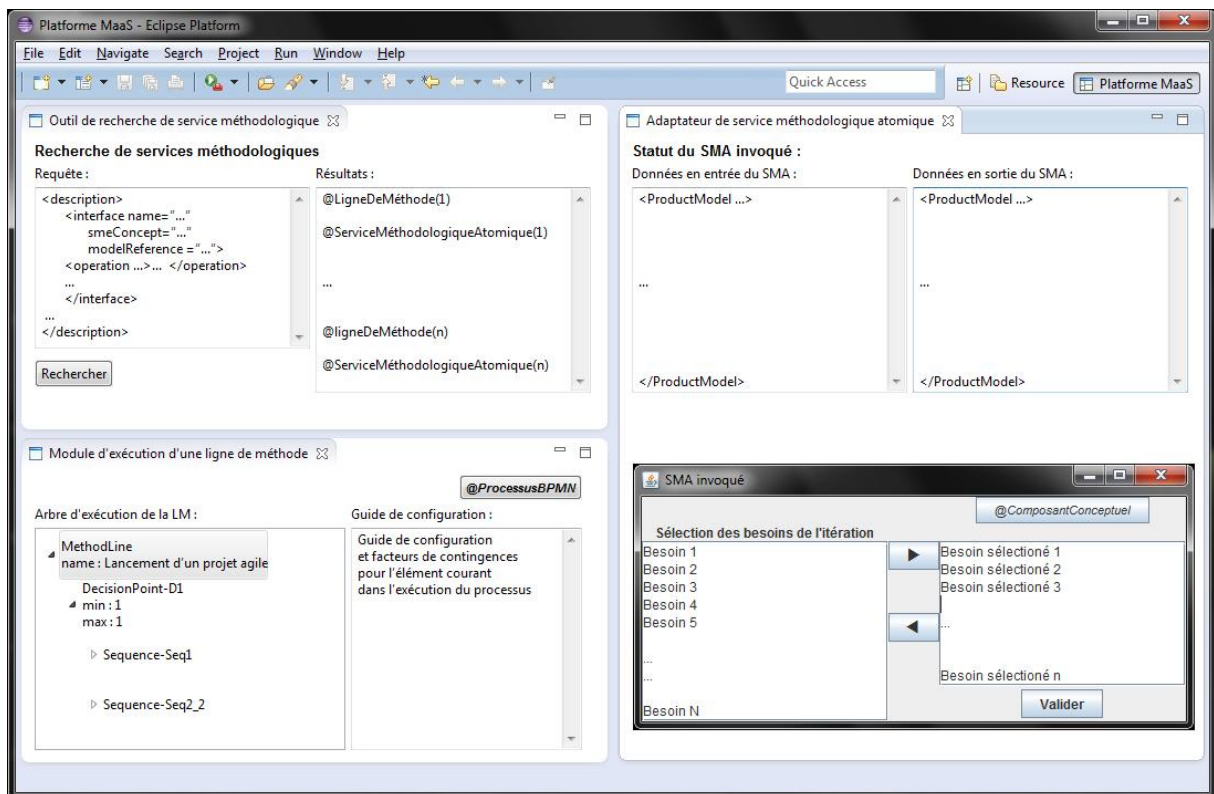


Figure 7.14 : Vue d'ensemble de la plateforme MaaS dans l'environnement Eclipse

Comme présenté à la figure 7.14 on retrouve les trois modules de la partie cliente ainsi qu'un SMA invoqué sous la forme de plugins Eclipse. Il est à noter qu'un plugin supplémentaire représentant la plateforme MaaS est nécessaire. Son rôle est d'assembler les plugins des modules de la plateforme dans une même perspective Eclipse ce qui permet d'organiser les différentes vues des modules dans une même couche de présentation.

Dans cette solution, l'adaptateur de SMA a pour rôle d'installer dynamiquement avant son exécution le composant logiciel implémentant le SMA au niveau de la couche OSGI d'environnement Eclipse. L'implémentation de l'adaptateur de SMA est par conséquent dépendante de l'environnement Eclipse et de la technologie OSGI.

Cette seconde solution permet d'illustrer l'usage d'une technologie OSGI pour l'implémentation de la plateforme MaaS dans un environnement de développement intégré tel que Eclipse et ainsi fournir une exécution *centralisée* des SM sur la machine de l'utilisateur.

7.5. Conclusion

Dans ce chapitre, nous avons présenté l'architecture générique de la plateforme MaaS organisée autour du fournisseur publiant des SM sur un annuaire pouvant être recherchés puis exécutés depuis la partie cliente de la plateforme. La partie cliente de cette plateforme est elle-même décomposée en plusieurs modules logiciels afin de maximiser leur indépendance et réutilisation lors de l'implémentation de la plateforme sur différents environnements techniques.

Nous avons ensuite présenté deux implémentations possibles de la plateforme MaaS, l'une reposant sur la technologie des portails et des portlets et l'autre sur l'environnement de développement intégré Eclipse et la technologie des composants logiciels OSGI. Ces deux solutions correspondent à des objectifs d'usage différents :

- La première offre une solution de plateforme MaaS sous la forme d'un portail et d'une partie cliente légère accessible à distance depuis un navigateur internet. Cette solution repose sur une exécution distribuée des SMA et des modules de la plateforme sur des serveurs distants.
- La seconde solution est intégrée à l'environnement Eclipse sous la forme d'un ensemble de plugins regroupés dans une même perspective MaaS. Cette solution offre une exécution centralisée sur une même machine cliente des SMA et des différents modules de la plateforme.

Ces deux implémentations permettent de démontrer la faisabilité technique et la généricité de la plateforme MaaS sur différentes architectures orientées services.

Chapitre 8. Conclusion

8.1. Contributions

Une quinzaine d'années après l'émergence du domaine de recherche de l'ingénierie des méthodes situationnelles, ces dernières peinent toujours à s'implanter dans les entreprises. Ces approches répondent cependant aux besoins des entreprises en termes d'adaptations de leurs méthodes de développement de SI aux différentes situations de projet.

La plupart des approches d'IMS proposent de construire une nouvelle méthode adaptée au contexte de projet en réutilisant des composants de méthode. Cette stratégie est coûteuse car elle nécessite une expertise méthodologique pour définir la méthode adaptée à chaque projet et est contraire à la politique de normalisation des entreprises qui est de définir une méthode qui s'appliquera pour tous les projets. En effet, les entreprises préféreraient configurer les méthodes qu'elles utilisent à leurs contextes.

De plus, l'application d'une approche d'IMS en entreprise est très lourde à mettre en œuvre car elle est transverse à l'entreprise, remet en cause les processus et impacte beaucoup d'acteurs. Elle ne peut pas s'appliquer d'emblée à une entreprise qui n'a aucune réflexion sur ses processus et leur évolution. La maturité de l'entreprise au regard de ses processus est un facteur important à prendre en compte.

La diversité des approches d'IMS et le manque de lisibilité, quant à leurs différences, est une source de confusion pour les utilisateurs qui devient un frein à leur adoption en entreprise. Il est renforcé par le manque d'interopérabilité entre les composants issus des différentes approches et le manque d'un socle commun aux composants de méthode pour favoriser leur diffusion et leur réutilisation.

Malgré les efforts de standardisation effectués par la communauté avec la production du standard ISO/IEC 24744 (ISO/IEC 24744, 2010), la standardisation des composants de méthode est insuffisante. Il existe également un manque de support outillé pour les méthodes situationnelles et les composants de méthode produits par les approches d'IMS. D'une manière plus générale, si toutes les approches d'IMS s'accordent sur la définition d'une méthode selon les cinq aspects : paradigme, produit, processus, organisationnel et outil, la couverture de ces cinq aspects par ces approches dans la définition de leurs composants de méthode est inégale.

Ces constats au sujet du domaine de l'IMS nous ont amené à poser la problématique suivante :

- *Comment améliorer l'usage de l'ingénierie des méthodes situationnelles dans les entreprises ?*

Cette problématique peut s'envisager selon différents axes d'améliorations contribuant positivement à l'usage des approches d'IMS dans les entreprises. Ces différents axes d'amélioration sont :

- Une base commune (ontologie) pour le domaine des approches d'IMS,
- l'interopérabilité des composants de méthode issus d'approches différentes,
- la prise en compte de l'aspect outil dans les approches d'IMS afin de fournir un outil aux méthodes situationnelles,
- l'intégration du concept de variabilité dans la définition des méthodes permettant de configurer des méthodes selon le contexte de projet,
- l'introduction d'une progressivité dans l'application d'une approche d'IMS en entreprise.

L'application du paradigme AOS au domaine de l'IMS est l'hypothèse sur laquelle nous avons fondé notre solution pour améliorer l'usage des approches d'IMS. Cette proposition s'articule autour des trois acteurs de l'architecture AOS : client, annuaire de services et fournisseur.

Notre solution fondée sur l'orientation service propose :

- une *ontologie de métamodèles de composant de méthode* permettant de réaliser une base commune aux approches d'IMS,
- un *modèle de SMA* fournissant l'aspect outil des composants méthode tout en garantissant leur interopérabilité,
- un *modèle de ligne de méthode* procurant une intégration du concept de variabilité dans les méthodes et leur fournissant un aspect outil,
- *l'architecture de la plateforme d'exécution* venant compléter l'aspect outil des SMA et des lignes de méthode en leur offrant un environnement d'exécution, et
- *l'adaptation des niveaux de maturité CMMI* permettant l'introduction d'une progressivité dans l'application d'une approche d'IMS en entreprise.

L'*ontologie de métamodèles de composant de méthode* (OMCM) offre une base commune aux approches d'IMS pour la définition de leurs composants de méthode. Cette ontologie est basée sur les cinq aspects de la définition d'une méthode. Elle décrit l'ensemble des concepts utilisés par les approches d'IMS pour la définition de leurs métamodèles de composants. En effet, chaque métamodèle de composant peut être défini sémantiquement à partir de l'ontologie de métamodèles OMCM. Cette base commune pour les composants trouve une application dans le paradigme AOS pour la construction d'un annuaire multi-composants. En d'autres termes, un annuaire sur lequel il est possible de publier, stocker et rechercher des SM et leurs composants de méthode associés indépendamment des approches d'IMS utilisées pour modéliser ces composants. Ces fonctionnalités sont réalisées dans la plateforme MaaS par l'usage de l'ontologie au niveau du descripteur de SM. Ceci assure une manière unique de décrire les composants de méthode issus d'approches différentes. Ce descripteur sémantique est lié au descripteur technique du SM supportant le composant. L'utilisation d'une technique de recherche sémantique conjointement à l'utilisation de cette ontologie pour la construction des descripteurs permet de rechercher les SM d'une manière uniforme.

Le *modèle de SMA* fournit un support multi-approches pour l'implémentation de l'aspect outil des composants de méthode. Il permet le développement d'outils sous la forme de services interactifs pour

les composants dont la réalisation est basée sur une extension des standards AOS. Du point de vue du fournisseur de méthode, cette solution apporte une réponse pour la réalisation d'outils interopérables pour les composants de méthode.

Le *modèle de ligne de méthode* offre une solution pour la définition de paquetages méthodologiques ou de méthodes dans une entreprise. Il permet d'introduire de la variabilité dans le processus méthodologique pour la configuration des méthodes en fonction des situations de projet d'une entreprise. L'opérationnalisation de ce modèle de ligne par: le *modèle d'arbre d'exécution*, le *modèle de configuration* et le *modèle de transformation des entrées/sorties* fournissent, avec l'utilisation conjointe de SMA, une implémentation d'un support outillé complet pour les méthodes. De plus, ces lignes sont vues comme des services méthodologiques et sont exposables par un fournisseur dans un annuaire.

L'*architecture de la plateforme d'exécution* de notre solution fournit un support outillé à l'approche MaaS vis-à-vis des utilisateurs, c'est-à-dire aux clients dans l'architecture AMOS. La partie cliente de cette plateforme s'organise en deux modules logiciels, un outil de recherche de SM, un outil d'exécution des SM.

L'*adaptation des niveaux de maturité* de l'approche CMMI (CMMI, 2006) au cadre de la démarche de notre solution offre d'une manière plus générique un processus de gestion d'une approche d'IMS dans une entreprise et permet une intégration progressive d'une démarche d'IMS dans celle-ci.

8.2. Perspectives

Le travail présenté dans cette thèse soulève de nouvelles questions de recherche. Il ouvre la voie à de nouvelles perspectives et il peut être poursuivi dans plusieurs directions.

- Evolution de l'ontologie de métamodèles de composant de méthode.

La granularité de la spécialisation des concepts de l'ontologie peut encore être affinée pour certains de ces concepts, notamment au niveau des concepts de processus et de produit qui sont les concepts principaux des approches d'IMS. De nombreux travaux portent sur la définition de ces concepts.

- Evolution de la définition d'une méthode situationnelle.

L'ontologie de métamodèles de composant de méthode est basée sur la définition d'une méthode autour des cinq aspects : paradigme, produit, processus, organisationnel et outil. Cette définition est commune à toutes les approches d'IMS. Il existe cependant des concepts importants utilisés dans les approches que nous avons retranscrits dans le métamodèle qui ne font pas partie de la définition de base d'une méthode. En effet, les méthodes produites par les approches d'IMS sont des méthodes situationnelles et, par conséquent, des concepts capturant les situations de réutilisation des méthodes situationnelles et de leurs composants de méthode sont nécessaires. Il est donc intéressant d'étudier la

perspective d'une définition d'une méthode situationnelle par l'ajout d'un aspect réutilisation à la définition d'une méthode. Cet aspect devrait décrire les caractéristiques des situations d'utilisation et de réutilisation des composants par les concepts de but et d'annotations descriptives.

- De nouvelles applications pour l'ontologie de métamodèles de composant de méthode.

La mise en correspondance des approches d'IMS avec l'ontologie de métamodèles de composant met en relief que les approches ont une couverture partielle de la définition d'une méthode selon les cinq aspects alors qu'elles adhèrent à celle-ci. Il convient donc de s'interroger sur la cause de cette couverture partielle et sur la nécessité d'une couverture complète. Les approches d'IMS existantes ont été créées avec des objectifs d'application pouvant différer d'une approche à l'autre. Il existe donc une perspective d'application de contextualisation de la démarche de l'IMS en elle-même en fonction de ses objectifs et de son domaine d'application. L'ontologie peut alors être utilisée comme base pour la contextualisation de nouvelles approches d'IMS.

- Production d'ontologies de domaine.

Du point de vue de l'annuaire de services méthodologiques, la publication et la recherche de services passent par l'utilisation conjointe d'ontologies de domaine avec notre ontologie de descripteurs de composant. Il est donc nécessaire de collecter les ontologies existantes dans le domaine du développement de SI et d'assurer la complétude de ce panel par la production de nouvelles ontologies de domaine, comme par exemple une ontologie des paradigmes et des langages de modélisation.

- De nouvelles implémentations.

Les implémentations de notre plateforme sont actuellement basées sur la technologie des portails/portlets et Eclipse/OSGI (OSGI, 2011). Il convient d'effectuer un portage de la plateforme sur d'autres environnements ou d'autres technologies afin de la rendre accessible à un maximum d'utilisateurs. Par exemple, un portage de la plateforme dans un autre environnement de développement intégré (EDI) tel que Netbeans (Netbeans, 2012) est envisageable. Les perspectives d'implémentation passent également par le développement de SMA et de lignes de méthode afin d'alimenter l'annuaire de services méthodologiques.

- Développement de métriques de qualité pour l'adaptation de CMMI aux démarches d'IMS.

L'adaptation des niveaux de capacité et de maturité aux démarches d'IMS nous a montré que les approches existantes sont définies à un niveau ajusté. Des métriques de qualité de la démarche d'IMS doivent être identifiées pour atteindre les deux derniers niveaux de l'échelle CMMI. Cela permettrait de piloter quantitativement la démarche d'IMS et de définir et d'appliquer un processus d'optimisation de cette démarche. Il convient également de poursuivre cette adoption de CMMI au domaine de l'IMS par l'identification de bonnes pratiques pour l'évaluation et l'amélioration du niveau de maturité de la démarche d'IMS.

Bibliographie

- (Aaen, 1992) I. Aaen, A. Siltanen, C. Sorensen, V. Tahvanainen, *A tale of two countries: CASE experience and expectations*, The Impact of Computer Supported Technology on Information Systems Development, North Holland Pub, Amsterdam, pp 61-93, 1992.
- (Agerfalk, 2003) P.J. Agerfalk, *Information systems actability: Understanding Information Techology as a Tool for Business Action and Communication*, Thèse de Doctorat, Dept. of Computer and Information Science, Linköping University, 2003.
- (Agerfalk, 2007) P. Agerfalk, S. Brinkkemper, C. Gonzales-Perez, B. Henderson-Sellers, F. Karlsson, S. Kelly, J. Ralyté, J., *Modularization Constructs in Method Engineering: Towards Common Ground?*, in: Panel of ME'07, Springer, Geneva, Switzerland, 2007.
- (Aristote, 2008) Aristote (384 av J.C. - 322 av J.C.), *La métaphysique*, GF-Flammarion, réédition, 2008.
- (Bachmann, 2001) F. Bachmann, L. Bass, *Managing variability in software architecture*, ACM Press, NY, USA, 2001.
- (Bennasri, 2005) S. Bennasri, *Une approche intentionnelle de représentation et de réalisation de la variabilité dans un système logiciel*, Thèse de Doctorat, Université de Paris I, Février 2005.
- (Brinkkemper, 1996) S. Brinkkemper, *Method Engineering: engineering of information systems development method and tools*, Information and Software Technology, vol. 38, 7, 1996.
- (Brinkkemper, 1999) S. Brinkkemper, M. Saeki, F. Harmsen, *Meta-Modelling Based Assembly Techniques for Situational Method Engineering*, Information Systems, vol 24, 3, pp 209-228, 1999.
- (Brinkkemper, 1998) S. Brinkkemper, M. Saeki, F. Harmsen, *Assembly Techniques for Method Engineering*, in proceedings of the international conference CAiSE'98, pp. 381-400, Pisa, Italy, 1998.
- (Brinkkemper, 2001) S. Brinkkemper, M. Saeki, A.F. Harmsen, *A method engineering Language for the description of systems development methods*, in proceedings of the international conference CAISE'01. Springer Verlag. Interlaken, Switzerland, 2001.
- (Bühne, 2004) S. Bühne, K. Lauenroth, K. Pohl, *Why is it not Sufficient to Model Requirements Variability with Feature Models?*, in: proceedings of the AURE'04 workshop in conjunction with the RE'04 conference, Japan, 2004.

- (Chabeb, 2008) Y. Chabeb, S. Tata, *Yet Another Semantic Annotation For WSDL*, in: Proceeding of International Conference WWW/Internet IADIS'08, 2008.
- (Chabeb, 2010) Y. Chabeb, S. Tata, A. Ozanne, *YASA-M: A Semantic Web Service Matchmaker*, in: proceedings of the Advanced Information Networking and Applications AINA'10 conference, 2010.
- (CMMI, 2006) CMMI Product Team, *Capability Maturity Model Integration for Development, Version 1.2*, disponible à l'adresse : <http://www.sei.cmu.edu/library/abstracts/reports/06tr008.cfm>, 2006.
- (Czarnecki, 2000) K. Czarnecki, U.W. Eisenecker, *Generative Programming – Methods, Tools and Applications*, Addison-Wesley, 2000.
- (Deneckere, 2008) R. Deneckère, A. Iacovelli, E. Kornyshova, C Souveyet, *From Method Fragments to Method Services*, in: EMMSAD Workshop of CAISE'08, Montpellier, France, 2008.
- (Deneckere, 2011) R. Deneckere, E. Kornyshova, I. Rychkova, *Des lignes de processus aux familles de processus*, actes de la conférence INFORSID, Lille, France, 2011.
- (Dik, 1989) S.C. Dik, *The theory of functional grammar*, Foris Publications, The Netherlands, 1989.
- (DSDM, 2009) DSDM Consortium, *DSDM Atern the Handbook*, disponible à l'adresse : <http://www.dsdm.org/dsdm-atern/atern>, 2009.
- (Eclipse, 2012) Eclipse, disponible à l'adresse : <http://www.eclipse.org/>, consulté le 15 mars 2012.
- (ECMA, 2011) ECMA International, *Standard ECMA-262 ECMAScript Language Specification Version 5.1*, disponible à l'adresse : <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>, 2011.
- (Fillmore, 1968) C.J. Fillmore, *The case of case*, in: Universals in linguistic theory, Holt, Rinehart and Winston Inc., 1968.
- (Firesmith, 2002) D.G. Firesmith, B. Henderson-Sellers, *The OPEN Process Framework. An introduction*, Addison-Wesley, 2002.
- (Gonzalez-Perez, 2007) C. Gonzalez-Perez, *Supporting Situational Method Engineering with ISO/IEC 24744 and the Work Product Pool Approach*; in: IFIP, Situational Method Engineering: Fundamentals and Experiences, 2007.

- (Gruber, 1993) T. R. Gruber, *A Translation Approach to Portable Ontology Specification*, Knowledge Acquisition, vol 5, 2, pp 199–220, 1993.
- (Gruber, 2009) T. Gruber, *Ontology*, in *the Encyclopedia of Database Systems*, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2009.
- (Guzélian, 2007a) G. Guzélian, C. Cauvet, *SO2M : Towards a Service-Oriented Approach for Method Engineering*, in: proceedings of the international conference IKE'07, USA, 2007.
- (Guzélian, 2007b) G. Guzélian, *Conception de systèmes d'information : une approche orientée service*, Thèse de Doctorat, Université de Paul Cézanne - Aix-Marseille 3, Juillet 2007.
- (Halmans, 2003) G. Halmans, K. Pohl, *Communicating the variability of a software product family to customers*, Software and System Modeling, Springer-Verlag, 2003.
- (Harmsen, 1994) F. Harmsen, S. Brinkkemper, H. Oei, *Situational Method Engineering for information Systems Project Approaches*, in: Int. IFIP WG8. 1 Conference in CRIS series: "Methods and associated Tools for the Information Systems Life Cycle" (A-55), North Holland (Pub.), 1994.
- (Harmsen, 1995) F. Harmsen, S. Brinkkemper, *Design and Implementation of a Method Base Management System for a Situational CASE Environment*, in proceedings of the Software Engineering Conference, Asia Pacific, 1995.
- (Harmsen, 1997) A.F. Harmsen, *Situational Method Engineering*, Thèse de Doctorat, Moret Ernst & Young, 1997.
- (Heather, 2001) K. Heather, *Web services conceptual architecture (wsca 1.0)*, IBM, 2001.
- (Henderson-Sellers, 1996) B. Henderson-Sellers, I.M. Graham, *OPEN: toward method convergence?*, IEEE Computer, vol 29, 4, pp 86-89, 1996.
- (Henderson-Sellers, 2002) B. Henderson-Sellers, *Process meta-modelling and process construction: examples using the OPF*, Ann. Software Engineering, vol. 14, 1-4, 2002.
- (Henderson-Sellers, 2004) B. Henderson-Sellers, M. Serour, T. McBride, C. Gonzalez-Perez, L. Dagher, *Process Construction and Customization*, Journal of Universal Computer Science, vol 10, 4, pp 326-358, 2004.
- (Iacovelli, 2008) A. Iacovelli, C. Souveyet, *Framework for Agile Methods Classification*, in: proceedings of the Workshop MoDISE'08 in conjunction with the CAISE'08 conference, 2008.

- (IBM, 2012) IBM, *UML-to-XSD transformation*, disponible à l'adresse : <http://publib.boulder.ibm.com/infocenter/rsdvhel/v6r0m1/index.jsp?topic=%2Fcom.ibm.xtools.transformations.doc%2Ftopics%2Fcxsdtransf.html>, consulté le 15 mars 2012.
- (ISO/IEC 24744, 2010) International Organization for Standardization, *ISO/IEC 24744, Software Engineering – Metamodel for Development Methodologies*, 2010.
- (Jackson, 1995) M. Jackson, *Software Requirements & Specifications – a Lexicon of Practice, Principles and Prejudices*, ACM Press, Addison-Wesley, 1995.
- (Kaabi, 2007) R.S. Kaabi, *Une Approche Méthodologique pour la Modélisation Intentionnelle des Services et leur Opérationnalisation*, Thèse de Doctorat, Université de Paris I, Février 2007.
- (Karlsson, 2004) F. Karlsson, P.J. Agerfalk *Method Configuration: Adapting to Situational Characteristics While Creating Reusable Assets*, Information and Software Technology, vol 46, 9, pp 619-633, 2004.
- (Karlsson, 2005) F. Karlsson, *Method Configuration: Method and Computerized Tool Support*, Thèse de Doctorat, Dept of Computer and Information Science, Linköping University, 2005.
- (Karlsson, 2006) F. Karlsson, K. Wistrand, *Combining method engineering with activity theory: theoretical grounding of the method component concept*, European Journal of Information Systems, vol 15, pp 82-90, 2006.
- (Kornysheva, 2011) E. Kornysheva, R. Deneckère, C. Rolland, *Method Families Concept: Application to Decision-Making Methods*, in proceedings of the EMMSAD'11 conference in conjunction with CAISE'11 conference, London, United Kingdom, LNBP, 2011.
- (Kruchten, 1998) P. Kruchten, *The Rational Unified Process-An Introduction*, Addison-Wesley, 1998.
- (Liferay, 2012) Liferay, *Liferay Portal*, disponible à l'adresse : <http://www.liferay.com/downloads/liferay-portal/available-releases>, consulté le 15 mars 2012.
- (Liu, 2006) C. Liu, J. Li, *Designing Quality XML Schemas from E-R Diagrams*, Advances in Web-Age Information Management, LNCS, pp 508-519, 2006.
- (Lyytinen, 1987) K. Lyytinen, *Different perspectives on information systems : problems and solutions*, ACM, Computing Surveys, vol 19, 1, 1987.
- (Mellal, 2007) N. Mellal, *Réalisation de l'interopérabilité sémantique des systèmes, basée sur les ontologies et les flux d'information*, Thèse de Doctorat, Université de Savoie, Décembre 2007.

- (Mirbel, 2006a) I. Mirbel, J. Ralyté, *Situational method engineering: combining assembly-based and roadmap-driven approaches*, in Requirement Engineering, vol 11, 1, pp 58-78, 2006.
- (Mirbel, 2006b) I. Mirbel, *A Reuse Frame for Method Engineering*, dans le rapport de recherche ISRN I3S/RR-2006-05-FR, 2006.
- (Mirbel, 2007) I. Mirbel, *Connecting Method Engineering Knowledge: a Community Based Approach*, in: proceedings of ME'07 conference, Geneva, Switzerland, 2007.
- (Netbeans, 2012) Netbeans, disponible à l'adresse : <http://netbeans.org/>, consulté le 15 mars 2012.
- (Niknafs, 2007) A. Niknafs, M. Asadi, H. Abolhassani, *Ontology-Based Method Engineering*, in: International Journal of Computer Science and Network Security. IJCSNS. vol. 7, 8, 2007.
- (OASIS, 2012) OASIS, disponible à l'adresse : <http://www.oasis-open.org/>, consulté le 15 mars 2012.
- (OASIS/BPEL, 2007) OASIS, *Web Services Business Process Execution Language Version 2.0*, disponible à l'adresse : <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>, 2007.
- (OASIS/UDDI, 2002) OASIS, *UDDI Version 2.04 API Specification*, disponible à l'adresse : <http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm>, 2002.
- (OASIS/WSRP, 2008) OASIS, *Web Services for Remote Portlets Specification v2.0*, disponible à l'adresse : <http://docs.oasis-open.org/wsrp/v2/wsrp-2.0-spec.html>, 2008.
- (OMG/BPMN, 2011) OMG, *Business Process Model and Notation (BPMN) Version 2.0*, disponible à l'adresse : <http://www.omg.org/spec/BPMN/2.0/PDF/>, 2011.
- (OMG/MOF, 2006) Object Management Group (OMG), *Meta Object Facility (MOF) v2.0*, disponible à l'adresse : <http://www.omg.org/spec/MOF/2.0/>, 2006.
- (OMG/UML, 2011) OMG, *OMG Unified Modeling Language (OMG UML), Infrastructure, version 2.4.1*, disponible à l'adresse : <http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/>, 2011.
- (OMG/XMI, 2011) Object Management Group (OMG), *XML Meta-data Interchange (XMI) v2.4.1*, disponible à l'adresse : <http://www.omg.org/spec/XMI/2.4.1/PDF/>, 2011.
- (Open Portal, 2012) Open Portal, *Open Portal Portlet Container Project*, disponible à l'adresse : <http://portlet-container.java.net/>, consulté le 15 mars 2012.
- (OPF, 2012) Open Process Framework, disponible à l'adresse : <http://www.opfro.org/>, consulté le 15 mars 2012.

- (ORACLE, 2008) ORACLE, *JSR-000286 Portlet Specification 2.0 2.0*, disponible à l'adresse : <http://download.oracle.com/otndocs/jcp/portlet-2.0-fr-oth-JSpec/>, 2008.
- (ORACLE/JAVA, 2011) ORACLE, *Java™ Platform, Standard Edition 7 API Specification*, disponible à l'adresse : <http://docs.oracle.com/javase/7/docs/api/>, 2011.
- (ORACLE/JSP, 2006) ORACLE, *Java Server Page Specification 2.1*, disponible à l'adresse : <http://download.oracle.com/otndocs/jcp/jsp-2.1-fr-eval-spec-oth-JSpec/>, 2006.
- (OSGI, 2011) OSGI Alliance, *OSGI Service Platform Core Specification Version 4.3*, disponible à l'adresse : <http://www.osgi.org/download/r4v43/r4.core.pdf>, 2011.
- (Papazoglou, 2003) M.P. Papazoglou, *Service-Oriented Computing: Concepts, Characteristics and Directions*, in: proceedings of the Fourth International Conference on Web Information Systems Engineering WISE'2003, 2003.
- (Papazoglou, 2007) M.P. Papazoglou, W.J. van den Heuvel, *Service oriented architectures: approaches, technologies and research issues*, The VLDB Journal, vol 16, pp 389–415, 2007.
- (Pin-shan Chen, 1976) P. Pin-shan Chen, *The Entity-Relationship Model: Toward a Unified View of Data*, ACM Transactions on Database Systems, 1976.
- (Plihon, 1995) V. Plihon, C. Rolland, *Modelling ways-of-working*, in: proceedings of the international CAISE'95 conference, LNCS, vol 932, pp 126-139, 1995.
- (Prieto-Diaz, 1991) R. Prieto-Diaz, *Implementing Faceted Classification for Software Reuse*, Communications of the ACM. Special issue on software engineering, vol 34, 5, pp 88-97, 1991.
- (Psyche, 2003) V. Psyche, O. Mendes, J. Bourdeau, *Apport de l'ingénierie ontologique aux environnements de formation à distance*, Revue STE, pp 89–126, 2003.
- (Punter, 1996) H.T. Punter, K. Lemmen, *The MEMA model: Towards a new approach for Method Engineering*, Information and Software Technology, vol. 38, No.4, pp.295-305, 1996.
- (Ralyté, 2001a) J. Ralyté, *Ingénierie des méthodes à base de composants*, Thèse de Doctorat, Université de Paris I, Janvier 2001.
- (Ralyté, 2001b) J. Ralyté, C. Rolland, *An Approach for Method Reengineering*, in proceedings of the 20th International Conference on Conceptual Modelling ER'01, LNCS 2224, Springer-Verlag, pp 471-484, Yokohama, Japan , 2001.
- (Ralyté, 2001c) J. Ralyté, C. Rolland, *An Assembly Process Model for Method Engineering*, in proceedings of the international conference CAISE'01, pp. 267-283, Interlaken, Switzerland, 2001.

- (Ralyté, 2003) J. Ralyté, R. Deneckère, C. Rolland, *Towards a Generic Model for Situational Method Engineering*, in: proceedings of the international conference CAISE'03, Austria, 2003.
- (Rolland, 1996) C. Rolland, N. Prakash, *Aproposal for context-specific method engineering*, in: Principles of method construction and tool support, Chapman & Hall, pp 191-208, 1996.
- (Rolland, 1998) C. Rolland, V. Plihon, J. Ralyté, *Specifying the reuse context of scenario method chunks*, in: proceedings of the international conference. CAiSE'98, Pise, 1998.
- (Rolland, 2005) C. Rolland, *L'ingénierie des méthodes : une visite guidée*, e-TI - la revue électronique des technologies d'information, vol. 1, 2005.
- (Routledge, 2002) N. Routledge, L. Bird, A. Goodchild, *UML and XML Schema*, in: proceedings of the Thirteenth Australasian Database Conference ADC2002, Melbourne, Australia, 2002.
- (Russo, 1995) Russo et al, *The use and adaptation of system development methodologies.*, in: proc. 1995 Intl. Resources Management. Association Conference, Atlanta, 1995.
- (SAP, 2005) SAP, IBM, *WS-BPEL Extension for People – BPEL4People*, White Paper, disponible à l'adresse : <http://scn.sap.com/docs/DOC-1294>, 2005.
- (Schwaber, 2011) K. Schwaber, J. Sutherland, *The Scrum Guide*, disponible à l'adresse : http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf, 2011.
- (Seligmann, 1989) P.S Seligmann, G .M Wijers, H.G Sol, *Analysing the structure of IS methodologies, an alternative approach*, in: proceedings of the 1st Dutch conference on Information Systems, Amersfoort, The Netherlands, 1989.
- (Si-Said Cherfi, 2002) S. Si-Said Cherfi, J. Akoka, I. Comyn-Wattiau, *Conceptual Modeling Quality - From EER to UML Schemas Evaluation*, in: proceedings of the International Conference on Conceptual Modeling ER'02, Tampere, Finland, LNCS, 2002.
- (Takeuchi, 1986) H. Takeuchi, I. Nonaka, *The New Product Development Game*, Dans Harvard Business Review, Janvier/Février, 1986.
- (VanGurp, 2000) J. Van Gurp, *Variability in Software Systems, the key to Software Reuse*, Licentiate Thesis, University of Groningen, Sweden, 2000.
- (W3C, 2012) W3C World Wide Web Consortium, disponible à l'adresse : <http://www.w3.org/>, consulté le 15 mars 2012.
- (W3C/CSS, 2011) W3C, *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*, disponible à l'adresse : <http://www.w3.org/TR/CSS2/>, 2011.
- (W3C/DOM, 2004) W3C, *Document Object Model (DOM) Technical Reports*, disponible à l'adresse : <http://www.w3.org/DOM/DOMTR>, 2004.

- (W3C/OWL, 2004) W3C, *OWL Web Ontology Language Reference*, disponible à l'adresse : <http://www.w3.org/TR/owl-ref/>, 2004.
- (W3C/RDF, 2004) W3C, *RDF Standards*, disponible à l'adresse : <http://www.w3.org/RDF/>, 2004.
- (W3C/SAWSDL, 2007) W3C, *Semantic Annotations for WSDL and XML Schema*, disponible à l'adresse : <http://www.w3.org/TR/sawSDL/>, 2007.
- (W3C/SOA, 2004) W3C, *Web Services Architecture*, disponible à l'adresse : <http://www.w3.org/TR/ws-arch/>.
- (W3C/SOAP, 2007) W3C, *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, disponible à l'adresse : <http://www.w3.org/TR/soap12-part1/>, 2007.
- (W3C/SPARQL, 2008) W3C, *SPARQL Query Language for RDF*, disponible à l'adresse : <http://www.w3.org/TR/rdf-sparql-query/>, 2008.
- (W3C/WSDL, 2001) W3C, *Web Services Description Language (WSDL) 1.1*, disponible à l'adresse : <http://www.w3.org/TR/wsdl/>, 2001.
- (W3C/XHTML, 2002) W3C, *XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)*, disponible à l'adresse : <http://www.w3.org/TR/xhtml1/>, 2002.
- (W3C/XML, 2008) W3C, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, disponible à l'adresse : <http://www.w3.org/TR/2008/REC-xml-20081126/>, 2008.
- (W3C/XSD, 2012) W3C, *XML Schema*, disponible à l'adresse : www.w3.org/XML/Schema, 2012.
- (Weerd, 2007) I. Weerd, S. Brinkkemper, J. Versendaal, *Concepts for incremental method evolution: Empirical exploration and validation in requirements management*, the international conference CAISE'07, pp 469-484, Trondheim, Norwa, 2007.
- (Welke, 1992) R.J. Welke, K. Kumar, *Method Engineering: A Proposal for Situation-specific Methodology Construction*, in *Systems Analysis and Design: A Research Agenda*, Cotterman and Senn(eds), Wiley, pp 257-268, 1992.
- (Wells, 2009) D. Wells, *Extreme programming*, disponible à l'adresse : <http://www.extremeprogramming.org>, consulté le 11 mars 2012, 2009.
- (Wijers, 1990) G. M. Wijers, H.E. Van Dort, *Experiences with the use of CASE tools in the Netherlands*, *Advanced Information Systems Engineering*, pp 5-20, 1990.
- (Wistrand, 2004) K. Wistrand, F. Karlsson, *Method components: Rationale revealed*, in *proceedings of CAISE 04*, Springer-Verlag. Riga, Latvia, 2004.

(Yourdon, 1992) E. Yourdon, *The decline and fall of the american programmer*, Prentice Hall, Englewood Cliffs, NJ, 1992.

Annexes

Table des annexes

| | |
|---|--------------|
| Annexe A | iii |
| Méta-ontologie de descripteurs de méthode..... | iii |
| (chapitre 4 section 4.2.2) | iii |
| Annexe B | xi |
| Schéma XML de la structure d’annotation d’un descripteur de service méthodologique..... | xi |
| (chapitre 4 section 4.3.3.1) | xi |
| Annexe C | xii |
| Descripteur du service méthodologique ObjectifyAMS | xii |
| (chapitre 5 section 5.2.2) | xii |
| Annexe D | xiv |
| Schéma XML de l’arbre d’exécution d’une ligne de méthode..... | xiv |
| (chapitre 5 section 5.3.3.3) | xiv |
| Annexe E | xvi |
| Schéma XML du modèle de configuration d’une ligne de méthode..... | xvi |
| (chapitre 5 section 5.3.4.3) | xvi |
| Annexe F | xvii |
| Schéma XML du modèle de transformation des entrées/sorties d’une ligne de méthode..... | xvii |
| (chapitre 5 section 5.3.5.3) | xvii |
| Annexe G | xix |
| Modèle de transformation des entrées/sorties de la ligne de méthode GIDPA | xix |
| (chapitre 5 section 5.3.5.4) | xix |
| Annexe H | xxii |
| Modèle de configuration de la ligne de méthode de lancement d’un projet agile..... | xxii |
| (chapitre 6 section 6.3.2) | xxii |
| Annexe I | xxvii |
| Descripteurs des SMA de la ligne de méthode de lancement d’un projet agile | xxvii |
| (chapitre 6 section 6.4) | xxvii |

| | |
|--|--------------|
| 1. Descripteur du SMA C1 - Etude de marché | xxvii |
| 2. Descripteur du SMA C2 - Etude technique | xxix |
| 3. Descripteur du SMA C3 - Abandon du projet | xxx |
| 4. Descripteur du SMA C4 - Définition de la liste des besoins du produit | xxxii |
| 5. Descripteur du SMA C5 - Définition de besoin utilisateur | xxxiii |
| 6. Descripteur du SMA C6 - Définition des besoins de haut niveau | xxxiv |
| 7. Descripteur du SMA C7 - Planification générale des livrables | xxxv |
| 8. Descripteur du SMA C8 - Définition du plan général des livrables | xxxvi |
| 9. Descripteur du SMA C9 - Définition du plan directeur | xxxvii |
| 10. Descripteur du SMA C10 - Affinement des besoins | xxxviii |
| 11. Descripteur du SMA C11 - Définition du planning de développement | xxxix |
| Annexe J | xl |
| Schéma XML du modèle de produit global de la ligne de méthode de lancement d'un projet agile . xl | |
| (chapitre 6 section 6.4.12) | xl |
| Annexe K | xliii |
| Arbre d'exécution de la ligne de méthode de lancement d'un projet agile | xliiii |
| (chapitre 6 section 6.5) | xliiii |
| Annexe L | xliv |
| Modèle de transformation des entrées/sorties de la ligne de méthode de lancement d'un projet agile | |
| | xliv |
| (chapitre 6 section 6.6) | xliv |

Annexe A

Méta-ontologie de descripteurs de méthode

(chapitre 4 section 4.2.2)

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<Ontology xmlns="http://www.w3.org/2002/07/owl#"
  xml:base="http://crinfo.univ-paris1.fr/ontologies/2011/12/SME-Otology.owl"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
  ontologyIRI="http://crinfo.univ-paris1.fr/ontologies/2011/12/SME-Otology.owl">
  <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#" />
  <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#" />
  <Prefix name="" IRI="http://www.w3.org/2002/07/owl#" />
  <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
  <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#" />
  <Declaration>
    <Class IRI="#Annotation" />
  </Declaration>
  <Declaration>
    <Class IRI="#Business" />
  </Declaration>
  <Declaration>
    <Class IRI="#Concept" />
  </Declaration>
  <Declaration>
    <Class IRI="#Conceptual_Process_Puzzle" />
  </Declaration>
  <Declaration>
    <Class IRI="#Conceptual_Product_Puzzle" />
  </Declaration>
  <Declaration>
    <Class IRI="#Goal" />
  </Declaration>
  <Declaration>
    <Class IRI="#Method" />
  </Declaration>
  <Declaration>
    <Class IRI="#Method_Package" />
  </Declaration>
  <Declaration>
    <Class IRI="#Method_Puzzle" />
  </Declaration>
  <Declaration>
    <Class IRI="#Modelling_Element" />
  </Declaration>
  <Declaration>
    <Class IRI="#Modelling_Language" />
  </Declaration>
  <Declaration>
    <Class IRI="#Modelling_Rule" />
  </Declaration>
  <Declaration>
    <Class IRI="#Organisational_Unit" />
  </Declaration>
  <Declaration>
    <Class IRI="#Paradigm" />
  </Declaration>
</Ontology>
```

```

<Declaration>
  <Class IRI="#Parameters"/>
</Declaration>
<Declaration>
  <Class IRI="#Post_Condition"/>
</Declaration>
<Declaration>
  <Class IRI="#Pre_Condition"/>
</Declaration>
<Declaration>
  <Class IRI="#Process_Puzzle"/>
</Declaration>
<Declaration>
  <Class IRI="#Product_Puzzle"/>
</Declaration>
<Declaration>
  <Class IRI="#Puzzle_Annotation"/>
</Declaration>
<Declaration>
  <Class IRI="#Situation_Annotation"/>
</Declaration>
<Declaration>
  <Class IRI="#Target"/>
</Declaration>
<Declaration>
  <Class IRI="#Technical_Process_Puzzle"/>
</Declaration>
<Declaration>
  <Class IRI="#Technical_Product_Puzzle"/>
</Declaration>
<Declaration>
  <Class IRI="#Verb"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#definesCollaborationOf"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#describes"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasGoalComposedOfParameters"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasGoalComposedOfTarget"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasGoalComposedOfVerb"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasMethodComposedOf"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasMethodPackageComposedOf"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasMethodPuzzleComposedOf"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasMethodPuzzleIsRelatedTo"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingElement"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingRule"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasParadigmComposedOf"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasPostCondition"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasPostConditionConstraints"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasPreCondition"/>

```

```

</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasPreConditionConstraints"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#isDefinedIn"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#isModellisedBy"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#isOrganisedIn"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#isRefined"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#isRelatedTo"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#manages"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#models"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#performs"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#produces"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#supports"/>
</Declaration>
<SubClassOf>
  <Class IRI="#Business"/>
  <Class IRI="#Method_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Conceptual_Process_Puzzle"/>
  <Class IRI="#Process_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Conceptual_Product_Puzzle"/>
  <Class IRI="#Product_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Method"/>
  <Class IRI="#Method_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Method_Package"/>
  <Class IRI="#Method_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Modelling_Language"/>
  <Class IRI="#Method_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Organisational_Unit"/>
  <Class IRI="#Method_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Process_Puzzle"/>
  <Class IRI="#Method_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Product_Puzzle"/>
  <Class IRI="#Method_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Puzzle_Annotation"/>
  <Class IRI="#Annotation"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Situation_Annotation"/>
  <Class IRI="#Annotation"/>

```

```

</SubClassOf>
<SubClassOf>
  <Class IRI="#Technical_Process_Puzzle"/>
  <Class IRI="#Process_Puzzle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Technical_Product_Puzzle"/>
  <Class IRI="#Product_Puzzle"/>
</SubClassOf>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#definesCollaborationOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#definesCollaborationOf"/>
    <Class IRI="#Business"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#describes"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#describes"/>
    <Class IRI="#Annotation"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasGoalComposedOfParameters"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasGoalComposedOfParameters"/>
    <Class IRI="#Goal"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasGoalComposedOfTarget"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasGoalComposedOfTarget"/>
    <Class IRI="#Goal"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasGoalComposedOfVerb"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasGoalComposedOfVerb"/>
    <Class IRI="#Goal"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasMethodComposedOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasMethodComposedOf"/>
    <Class IRI="#Method"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasMethodPackageComposedOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasMethodPackageComposedOf"/>
    <Class IRI="#Method_Package"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasMethodPuzzleComposedOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasMethodPuzzleComposedOf"/>
    <Class IRI="#Method_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasMethodPuzzleIsRelatedTo"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasMethodPuzzleIsRelatedTo"/>
    <Class IRI="#Method_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingElement"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingElement"/>
    <Class IRI="#Modelling_Language"/>
  </ObjectSomeValuesFrom>

```

```

    </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingRule"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingRule"/>
    <Class IRI="#Modelling_Language"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasParadigmComposedOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasParadigmComposedOf"/>
    <Class IRI="#Paradigm"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasPostCondition"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasPostCondition"/>
    <Class IRI="#Process_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasPostConditionConstraints"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasPostConditionConstraints"/>
    <Class IRI="#Post_Condition"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasPreCondition"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasPreCondition"/>
    <Class IRI="#Process_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasPreConditionConstraints"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasPreConditionConstraints"/>
    <Class IRI="#Pre_Condition"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#isDefinedIn"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isDefinedIn"/>
    <Class IRI="#Method_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#isModellisedBy"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isModellisedBy"/>
    <Class IRI="#Method_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#isOrganisedIn"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isOrganisedIn"/>
    <Class IRI="#Business"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#isRefined"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isRefined"/>
    <Class IRI="#Goal"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#isRelatedTo"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isRelatedTo"/>
    <Class IRI="#Process_Puzzle"/>
  </ObjectSomeValuesFrom>

```

```

    </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#manages"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#manages"/>
    <Class IRI="#Method_Package"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#models"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#models"/>
    <Class IRI="#Modelling_Element"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#performs"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#performs"/>
    <Class IRI="#Organisational_Unit"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#produces"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#produces"/>
    <Class IRI="#Organisational_Unit"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#supports"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#supports"/>
    <Class IRI="#Modelling_Language"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyRange>
  <ObjectProperty IRI="#definesCollaborationOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#definesCollaborationOf"/>
    <Class IRI="#Organisational_Unit"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#describes"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#describes"/>
    <Class IRI="#Method_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasGoalComposedOfParameters"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasGoalComposedOfParameters"/>
    <Class IRI="#Parameters"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasGoalComposedOfTarget"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasGoalComposedOfTarget"/>
    <Class IRI="#Target"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasGoalComposedOfVerb"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasGoalComposedOfVerb"/>
    <Class IRI="#Verb"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasMethodComposedOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasMethodComposedOf"/>
    <Class IRI="#Method_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>

```

```

    </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasMethodPackageComposedOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasMethodPackageComposedOf"/>
    <Class IRI="#Method_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasMethodPuzzleComposedOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasMethodPuzzleComposedOf"/>
    <Class IRI="#Method_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasMethodPuzzleIsRelatedTo"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasMethodPuzzleIsRelatedTo"/>
    <Class IRI="#Goal"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingElement"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingElement"/>
    <Class IRI="#Modelling_Element"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingRule"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasModellingLanguageComposedOfModellingRule"/>
    <Class IRI="#Modelling_Rule"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasParadigmComposedOf"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasParadigmComposedOf"/>
    <Class IRI="#Concept"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasPostCondition"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasPostCondition"/>
    <Class IRI="#Post_Condition"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasPostConditionContraints"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasPostConditionContraints"/>
    <Class IRI="#Product_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasPreCondition"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasPreCondition"/>
    <Class IRI="#Pre_Condition"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasPreConditionContraints"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasPreConditionContraints"/>
    <Class IRI="#Product_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#isDefinedIn"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isDefinedIn"/>
    <Class IRI="#Paradigm"/>
  </ObjectSomeValuesFrom>

```



```

    </ObjectSomeValuesFrom>
  </ObjectPropertyRange>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#isModellisedBy"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isModellisedBy"/>
    <Class IRI="#Modelling_Language"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#isOrganisedIn"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isOrganisedIn"/>
    <Class IRI="#Method_Package"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#isRefined"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isRefined"/>
    <Class IRI="#Goal"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#isRelatedTo"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#isRelatedTo"/>
    <Class IRI="#Product_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#manages"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#manages"/>
    <Class IRI="#Method_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#models"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#models"/>
    <Class IRI="#Concept"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#performs"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#performs"/>
    <Class IRI="#Process_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#produces"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#produces"/>
    <Class IRI="#Product_Puzzle"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#supports"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#supports"/>
    <Class IRI="#Paradigm"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
</ObjectPropertyRange>
</Ontology>
<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->

```

Annexe B

Schéma XML de la structure d'annotation d'un descripteur de service méthodologique

(chapitre 4 section 4.3.3.1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace="http://crinfo.univ-paris1.fr/schema/WSDLExt"
  xmlns="http://crinfo.univ-paris1.fr/schema/WSDLExt"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wSDL="http://www.w3.org/ns/wSDL"
>

  <xsd:simpleType name="listOfAnyURI">
    <xsd:list itemType="xsd:anyURI"/>
  </xsd:simpleType>

  <xsd:attribute name="smeConcept" type="listOfAnyURI" />
  <xsd:attribute name="modelReference" type="listOfAnyURI" />
  <xsd:attribute name="conceptualMethodPuzzle" type="xsd:anyURI" />
  <xsd:attribute name="puzzleType" type="xsd:string" />
  <xsd:attribute name="technicalMethodPuzzle" type="xsd:anyURI" />

</xsd:schema>
```

Annexe C

Descripteur du service méthodologique ObjectifyAMS

(chapitre 5 section 5.2.2)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-paris1.fr/schema/ObjectifyAMS.wsdl"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://ObjectifyAMS/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewsdsl="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/ObjectifyAMS"
  xmlns:wsoap="http://www.w3.org/ns/wsdl/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
  xmlns:xmidocument="http://schema.omg.org/spec/XMI/2.1"
>

  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/ObjectifyAMS.xsd">
      <xsd:import namespace="http://schema.omg.org/spec/XMI/2.1"
        schemaLocation="http://www.omg.org/spec/XMI/20071213/XMI.xsd"/>
      <xsd:element name="ObjectifyAMSRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xmidocument:XMI" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="ObjectifyAMSResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xmidocument:XMI" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>

  <interface name="ObjectifyAMSInterface">
    smewsdsl:conceptualMethodPuzzle="http://crinfo.univ-
    paris1.fr/schema/Objectify.html"
    smewsdsl:puzzleType="atomic-method-service"
    smewsdsl:technicalMethodPuzzle="http://crinfo.univ-
    paris1.fr/wsrp/wsdl/ObjectifyAMSPortlet.wsdl"
    smewsdsl:smeConcept="http://crinfo.univ-
    paris1.fr/schema/MethodPuzzleOntology#MethodPuzzle
      http://crinfo.univ-
    paris1.fr/schema/MethodPuzzleOntology#Paradigm
      http://crinfo.univ-
    paris1.fr/schema/MethodPuzzleOntology#ModellingLanguage"
    smewsdsl:modelReference="http://crinfo.univ-
    paris1.fr/schema/designPatternsOntology#Objectify
      http://crinfo.univ-
    paris1.fr/schema/modellingOntology#ObjectOriented
      http://crinfo.univ-
    paris1.fr/schema/modellingOntology#UML"
  >
    <operation name="Objectify">
      smewsdsl:smeConcept="http://crinfo.univ-
    paris1.fr/schema/MethodPuzzleOntology#Process"
      smewsdsl:modelReference="http://crinfo.univ-
    paris1.fr/schema/designPatternsOntology#Objectify"
    >
      <input messageLabel="ObjectifyAMSInputMessage" element="xsdio:ObjectifyAMSRequest"
        smewsdsl:smeConcept="http://crinfo.univ-
    paris1.fr/schema/MethodPuzzleOntology#Product"
        smewsdsl:modelReference="http://crinfo.univ-
    paris1.fr/schema/modellingOntology#UMLClassDiagram"
      />
    </operation>
  </interface>

```

```

        <output messageLabel="ObjectifyAMSOutputMessage" element="xsdio:ObjectifyAMSResponse"
            smewsdsl:smeConcept="http://crinfo.univ-
parisl.fr/schema/MethodPuzzleOntology#Product"
            smewsdsl:modelReference="http://crinfo.univ-
parisl.fr/schema/modellingOntology#UMLClassDiagram"
            />
    </operation>
</interface>

<binding name="ObjectifyAMSBinding"
    interface="tns:ObjectifyAMSInterface"
    type="http://www.w3.org/ns/wsdl/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <operation ref="tns:Objectify" />
</binding>

<service name="ObjectifyAMS"
    interface="tns:ObjectifyAMSInterface">

    <endpoint name="ObjectifyAMSEndpoint"
        binding="tns:ObjectifyAMSBinding"
        address =""/>

</service>
</description>

```

Annexe D

Schéma XML de l'arbre d'exécution d'une ligne de méthode

(chapitre 5 section 5.3.3.3)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://crinfo.univ-paris1.fr/schema/MethodLine"
  xmlns:tns="http://crinfo.univ-paris1.fr/schema/MethodLine"
  elementFormDefault="qualified"
  >

  <xsd:simpleType name="nonNegativeInteger" id="nonNegativeInteger">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
    </xsd:restriction>
  </xsd:simpleType>
  <!-- root element -->
  <xsd:element name="MethodLine">
    <xsd:complexType>
      <xsd:group ref="tns:MethodElement" minOccurs="1" maxOccurs="1"/>
      <xsd:attribute name="name" use="required" type="xsd:string" />
      <xsd:attribute name="version" use="required" type="xsd:string" />
      <xsd:attribute name="configurationUrl" use="required" type="xsd:anyURI" />
      <xsd:attribute name="assignUrl" use="required" type="xsd:anyURI" />
    </xsd:complexType>
  </xsd:element>
  <!-- root element -->

  <xsd:element name="Sequence">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:group ref="tns:MethodElement" minOccurs="2" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Parallel">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:group ref="tns:MethodElement" minOccurs="2" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RepeatUntil">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:group ref="tns:MethodElement" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="DecisionPoint">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:group ref="tns:MethodElement" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" use="required" type="xsd:string" />
      <xsd:attribute name="min" use="required" type="tns:nonNegativeInteger" />
      <xsd:attribute name="max" use="required" type="xsd:positiveInteger" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="AtomicMethodService">
    <xsd:complexType>
      <xsd:attribute name="id" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

```

```
        <xsd:attribute name="name" use="required" type="xsd:string" />
        <xsd:attribute name="url" use="required" type="xsd:anyURI" />
    </xsd:complexType>
</xsd:element>

<xsd:group name="MethodElement">
    <xsd:choice>
        <xsd:element ref="tns:Sequence" />
        <xsd:element ref="tns:Parallel" />
        <xsd:element ref="tns:RepeatUntil" />
        <xsd:element ref="tns:DecisionPoint" />
        <xsd:element ref="tns:AtomicMethodService" />
    </xsd:choice>
</xsd:group>
</xsd:schema>
```

Annexe E

Schéma XML du modèle de configuration d'une ligne de méthode

(chapitre 5 section 5.3.4.3)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://crinfo.univ-paris1.fr/schema/ConfigurationModel"
  xmlns:tns="http://crinfo.univ-paris1.fr/schema/ConfigurationModel"
  elementFormDefault="qualified"
  >

<!-- root element -->
  <xsd:element name="ConfigurationModel">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:Decision" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="name" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>
<!-- root element -->

  <xsd:element name="Decision" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:ConfigurationGuidelineList" />
        <xsd:element ref="tns:ContingencyFactorList" />
      </xsd:sequence>
      <xsd:attribute name="targetId" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="ConfigurationGuidelineList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:ConfigurationGuideline" minOccurs="0"
          maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="ContingencyFactorList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:ContingencyFactor" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="ConfigurationGuideline">
    <xsd:complexType mixed="true">
      <xsd:attribute name="name" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="ContingencyFactor">
    <xsd:complexType mixed="true">
      <xsd:attribute name="name" use="required" type="xsd:string" />
      <xsd:attribute name="targetId" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

Annexe F

Schéma XML du modèle de transformation des entrées/sorties d'une ligne de méthode

(chapitre 5 section 5.3.5.3)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://crinfo.univ-paris1.fr/schema/AssignmentModel"
  xmlns:tns="http://crinfo.univ-paris1.fr/schema/AssignmentModel"
  elementFormDefault="qualified">
<!-- root element -->
  <xsd:element name="AssignmentModel">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:ProductModels" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="tns:MethodLine" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="tns:Assign" minOccurs="1" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
<!-- root element -->
  <xsd:element name="ProductModels" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
          processContents="strict"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="MethodLine" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:Input" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="tns:Output" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="name" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Assign">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:Input" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="tns:Output" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="parentId" use="required" type="xsd:string" />
      <xsd:attribute name="leafId" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Input">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:Transform" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:Transform" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```



```
<xsd:element name="Transform">
  <xsd:complexType>
    <xsd:attribute name="source" use="required" type="xsd:QName" />
    <xsd:attribute name="target" use="required" type="xsd:QName" />
    <xsd:attribute name="xsltRef" use="required" type="xsd:anyURI" />
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Annexe G

Modèle de transformation des entrées/sorties de la ligne de méthode GIDPA

(chapitre 5 section 5.3.5.4)

```
<?xml version="1.0" encoding="UTF-8"?>
<am:AssignmentModel xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:am='http://crinfo.univ-paris1.fr/schema/AssignmentModel'
xsi:schemaLocation='http://crinfo.univ-paris1.fr/schema/AssignmentModel AssignmentModel.xsd
http://www.w3.org/1999/XMLSchema http://www.w3.org/1999/XMLSchema'
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:mlpm="http://crinfo.univ-paris1.fr/schema/ML-PM-Iteration"
xmlns:c1="http://crinfo.univ-paris1.fr/schema/IterationRequirementsSelection"
xmlns:c2="http://crinfo.univ-paris1.fr/schema/ProductOrientedRequirementRefinement"
xmlns:c3="http://crinfo.univ-paris1.fr/schema/GoalOrientedRequirementRefinement"
xmlns:c4="http://crinfo.univ-paris1.fr/schema/IterationPlanning"
xmlns:c5="http://crinfo.univ-paris1.fr/schema/DesignTasksTracabiliby"
xmlns:c6="http://crinfo.univ-paris1.fr/schema/DevelopmentTasksTracability"
xmlns:c7="http://crinfo.univ-paris1.fr/schema/TestTasksTracability"
xmlns:c8="http://crinfo.univ-paris1.fr/schema/StandUpMeeting"
xmlns:c9="http://crinfo.univ-paris1.fr/schema/DailyMeeting"
>
<am:ProductModels>
  <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/AM-Iteration.xsd">
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/ML-PM-Iteration"
schemaLocation="http://crinfo.univ-paris1.fr/schema/ML-PM-Iteration.xsd"/>
    <xsd:import namespace="http://crinfo.univ-
paris1.fr/schema/IterationRequirementsSelection" schemaLocation="http://crinfo.univ-
paris1.fr/schema/IterationRequirementsSelection.xsd"/>
    <xsd:import namespace="http://crinfo.univ-
paris1.fr/schema/ProductOrientedRequirementRefinement" schemaLocation="http://crinfo.univ-
paris1.fr/schema/ProductOrientedRequirementRefinement.xsd"/>
    <xsd:import namespace="http://crinfo.univ-
paris1.fr/schema/GoalOrientedRequirementRefinement" schemaLocation="http://crinfo.univ-
paris1.fr/schema/GoalOrientedRequirementRefinement.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/IterationPlanning"
schemaLocation="http://crinfo.univ-paris1.fr/schema/IterationPlanning.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/DesignTasksTracabiliby"
schemaLocation="http://crinfo.univ-paris1.fr/schema/DesignTasksTracabiliby.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/DevelopmentTasksTracability"
schemaLocation="http://crinfo.univ-paris1.fr/schema/DevelopmentTasksTracability.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/TestTasksTracability"
schemaLocation="http://crinfo.univ-paris1.fr/schema/TestTasksTracability.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/StandUpMeeting"
schemaLocation="http://crinfo.univ-paris1.fr/schema/StandUpMeeting.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/DailyMeeting"
schemaLocation="http://crinfo.univ-paris1.fr/schema/DailyMeeting.xsd"/>
  </xsd:schema>
</am:ProductModels>

<am:MethodLine name="Déroulement d'une itération dans un cycle de vie agile">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="mlpm:ModeleDeProduit"
xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="mlpm:ModeleDeProduit" target="mlpm:ModeleDeProduit"
xsltRef="http://.../copyAll.xslt"/>
  </am:Output>
</am:MethodLine>
<am:Assign parentId="Seq1" leafId="C1">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c1:ListeDesBesoins"
xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
```

```

    <am:Transform source="c1:Iteration" target="mlpm:Iteration"
xsltRef="http://.../copyAll.xslt"/>
  </am:Output>
</am:Assign>
<am:Assign parentId="D1" leafId="C2">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c2:ListeDesBesoins"
xsltRef="http://.../copyAll.xslt"/>
    <am:Transform source="mlpm:Iteration" target="c2:Iteration"
xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c2:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://.../replaceAll.xslt"/>
  </am:Output>
</am:Assign>
<am:Assign parentId="D1" leafId="C3">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c3:ListeDesBesoins"
xsltRef="http://.../copyAll.xslt"/>
    <am:Transform source="mlpm:Iteration" target="c3:Iteration"
xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c3:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://.../replaceAll.xslt"/>
  </am:Output>
</am:Assign>
<am:Assign parentId="D4" leafId="C4">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c4:ListeDesBesoins"
xsltRef="http://.../copyAll.xslt"/>
    <am:Transform source="mlpm:Iteration" target="c4:Iteration"
xsltRef="http://.../copyAll.xslt"/>
    <am:Transform source="mlpm:Acteur" target="c4:Acteur" xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c4:Iteration" target="mlpm:Iteration"
xsltRef="http://.../replaceAll.xslt"/>
    <am:Transform source="c4:Acteur" target="mlpm:Acteur" xsltRef="http://.../copyAll.xslt"/>
    <am:Transform source="c4:Role" target="mlpm:Role" xsltRef="http://.../copyAll.xslt"/>
  </am:Output>
</am:Assign>
<am:Assign parentId="D2" leafId="C5">
  <am:Input>
    <am:Transform source="mlpm:Tache" target="c5:Tache" xsltRef="http://.../copyAll.xslt"/>
    <am:Transform source="mlpm:Iteration" target="c5:Iteration"
xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c5:Tache" target="mlpm:Tache" xsltRef="http://.../replaceAll.xslt"/>
  </am:Output>
</am:Assign>
<am:Assign parentId="D2" leafId="C6">
  <am:Input>
    <am:Transform source="mlpm:Tache" target="c6:Tache" xsltRef="http://.../copyAll.xslt"/>
    <am:Transform source="mlpm:Iteration" target="c6:Iteration"
xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c6:Tache" target="mlpm:Tache" xsltRef="http://.../replaceAll.xslt"/>
  </am:Output>
</am:Assign>
<am:Assign parentId="D2" leafId="C7">
  <am:Input>
    <am:Transform source="mlpm:Tache" target="c7:Tache" xsltRef="http://.../copyAll.xslt"/>
    <am:Transform source="mlpm:Iteration" target="c7:Iteration"
xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c7:Tache" target="mlpm:Tache" xsltRef="http://.../replaceAll.xslt"/>
  </am:Output>
</am:Assign>
<am:Assign parentId="D3" leafId="C8">
  <am:Input>
    <am:Transform source="mlpm:Tache" target="c8:Tache" xsltRef="http://.../copyAll.xslt"/>

```

```
<am:Transform source="mlpm:Iteration" target="c8:Iteration"
xsltRef="http://.../copyAll.xslt"/>
</am:Input>
<am:Output>
  <am:Transform source="c8:Iteration" target="mlpm:Iteration"
xsltRef="http://.../replaceAll.xslt"/>
</am:Output>
</am:Assign>
<am:Assign parentId="D3" leafId="C9">
  <am:Input>
    <am:Transform source="mlpm:Tache" target="c9:Tache" xsltRef="http://.../copyAll.xslt"/>
    <am:Transform source="mlpm:Iteration" target="c9:Iteration"
xsltRef="http://.../copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c9:Iteration" target="mlpm:Iteration"
xsltRef="http://.../replaceAll.xslt"/>
  </am:Output>
</am:Assign>
</am:AssignmentModel>
```

Annexe H

Modèle de configuration de la ligne de méthode de lancement d'un projet agile

(chapitre 6 section 6.3.2)

```
<?xml version="1.0" encoding="UTF-8"?>
<cm:ConfigurationModel xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:cm='http://crinfo.univ-paris1.fr/schema/ConfigurationModel'
xsi:schemaLocation='http://crinfo.univ-paris1.fr/schema/ConfigurationModel
ConfigurationModel.xsd' name="Modèle de configuration de la ligne de méthode de lancement d'un
projet agile">
  <cm:Decision targetId="D1">
    <cm:ConfigurationGuidelineList>
      <cm:ConfigurationGuideline name="G1">Il est recommandé s'intéresser au type du projet et de
choisir la Seq1 si c'est un grand projet ou un projet d'une complexité
élevée.</cm:ConfigurationGuideline>
      <cm:ConfigurationGuideline name="G2">Il est recommandé de choisir la Seq1 si le projet est
innovant.</cm:ConfigurationGuideline>
      <cm:ConfigurationGuideline name="G3">Il est recommandé de choisir la Seq1 si le projet est
risqué en termes de besoins.</cm:ConfigurationGuideline>
    </cm:ConfigurationGuidelineList>
    <cm:ContingencyFactorList>
      <cm:ContingencyFactor name="Taille du projet" targetId="Seq1">grande</cm:ContingencyFactor>
      <cm:ContingencyFactor name="Complexité du projet"
targetId="Seq1">forte</cm:ContingencyFactor>
      <cm:ContingencyFactor name="Le degré d'innovation du projet"
targetId="Seq1">fort</cm:ContingencyFactor>
      <cm:ContingencyFactor name="Le degré de risques liés au projet"
targetId="Seq1">faible</cm:ContingencyFactor>
      <cm:ContingencyFactor name="Taille du projet"
targetId="Seq2_2">petite</cm:ContingencyFactor>
      <cm:ContingencyFactor name="Complexité du projet"
targetId="Seq2_2">faible</cm:ContingencyFactor>
      <cm:ContingencyFactor name="Le degré d'innovation du projet"
targetId="Seq2_2">faible</cm:ContingencyFactor>
      <cm:ContingencyFactor name="Le degré de risques liés au projet"
targetId="Seq2_2">faible</cm:ContingencyFactor>
    </cm:ContingencyFactorList>
  </cm:Decision>
  <cm:Decision targetId="D2">
    <cm:ConfigurationGuidelineList>
      <cm:ConfigurationGuideline name="G4">Il est recommandé de choisir la branche D2.1 si le
projet est innovant.</cm:ConfigurationGuideline>
      <cm:ConfigurationGuideline name="G5">Il est recommandé de choisir la branche D2.2 si le
projet est risqué en termes de besoins.</cm:ConfigurationGuideline>
    </cm:ConfigurationGuidelineList>
    <cm:ContingencyFactorList>
      <cm:ContingencyFactor name="Le degré d'innovation du projet"
targetId="C2"><fort/cm:ContingencyFactor>
      <cm:ContingencyFactor name="Le degré de risques liés au projet"
targetId="C2">faible</cm:ContingencyFactor>
      <cm:ContingencyFactor name="Le degré d'innovation du projet"
targetId="C1">faible</cm:ContingencyFactor>
      <cm:ContingencyFactor name="Le degré de risques liés au projet"
targetId="C1">fort</cm:ContingencyFactor>
    </cm:ContingencyFactorList>
  </cm:Decision>
  <cm:Decision targetId="D4">
    <cm:ConfigurationGuidelineList>
      <cm:ConfigurationGuideline name="G12">Est-ce que le projet à été décrété valide dans les
études techniques et de marchés?</cm:ConfigurationGuideline>
    </cm:ConfigurationGuidelineList>
    <cm:ContingencyFactorList/>
  </cm:Decision>
  <cm:Decision targetId="D3_1">
    <cm:ConfigurationGuidelineList>
```

```

    <cm:ConfigurationGuideline name="G6_1">Il est recommandé de choisir C4_1 si le projet est de
petite taille mais avec un niveau de complexité important.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G7_1">Choisir C4_1 impliquera d'avoir une forte implication
du client pour la définition des besoins. Cependant la mise en œuvre des interactions entre le
client et l'équipe seront faiblement contraignantes. </cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G8_1">Il est recommandé de choisir C5_1 si le projet est de
petite taille et de faible complexité.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G9_1">Choisir C5_1 impliquera d'avoir une forte implication
du client pour la définition des besoins et la mise en œuvre des interactions entre le client
et l'équipe seront contraignantes pour le client.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G10_1">Il est recommandé de choisir C6_1 si le projet est
de grande taille et complexe.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G11_1">Choisir C6_1 impliquera d'avoir une forte
implication des utilisateurs finaux pour la définition des
besoins.</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList>
    <cm:ContingencyFactor name="Taille du projet" targetId="C4_1">petite</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C4_1">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Le degré d'interaction avec les utilisateurs finaux du système"
targetId="C4_1">faible</cm:ContingencyFactor>
    <cm:ContingencyFactor name="L'implication du client dans le processus de développement"
targetId="C4_1">moyenne</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Coût de mise en œuvre de l'alternative pour l'équipe"
targetId="C4_1">2</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Taille du projet" targetId="C5_1">petite</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C5_1">faible</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Le degré d'interaction avec les utilisateurs finaux du système"
targetId="C5_1">faible</cm:ContingencyFactor>
    <cm:ContingencyFactor name="L'implication du client dans le processus de développement"
targetId="C5_1">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Coût de mise en œuvre de l'alternative pour l'équipe"
targetId="C5_1">6</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Taille du projet" targetId="C6_1">grande</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C6_1">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Le degré d'interaction avec les utilisateurs finaux du système"
targetId="C6_1">fort</cm:ContingencyFactor>
    <cm:ContingencyFactor name="L'implication du client dans le processus de développement"
targetId="C6_1">moyenne</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Coût de mise en œuvre de l'alternative pour l'équipe"
targetId="C6_1">4</cm:ContingencyFactor>
  </cm:ContingencyFactorList>
</cm:Decision>
<cm:Decision targetId="D3_2">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G6_2">Il est recommandé de choisir C4_2 si le projet est de
petite taille mais avec un niveau de complexité important.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G7_2">Choisir C4_2 impliquera d'avoir une forte implication
du client pour la définition des besoins. Cependant la mise en œuvre des interactions entre le
client et l'équipe seront faiblement contraignantes. </cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G8_2">Il est recommandé de choisir C5_2 si le projet est de
petite taille et de faible complexité.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G9_2">Choisir C5_2 impliquera d'avoir une forte implication
du client pour la définition des besoins et la mise en œuvre des interactions entre le client
et l'équipe seront contraignantes pour le client.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G10_2">Il est recommandé de choisir C6_2 si le projet est
de grande taille et complexe.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G11_2">Choisir C6_2 impliquera d'avoir une forte
implication des utilisateurs finaux pour la définition des
besoins.</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList>
    <cm:ContingencyFactor name="Taille du projet" targetId="C4_2">petite</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C4_2">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Le degré d'interaction avec les utilisateurs finaux du système"
targetId="C4_2">faible</cm:ContingencyFactor>
    <cm:ContingencyFactor name="L'implication du client dans le processus de développement"
targetId="C4_2">moyenne</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Coût de mise en œuvre de l'alternative pour l'équipe"
targetId="C4_2">2</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Taille du projet" targetId="C5_2">petite</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C5_2">faible</cm:ContingencyFactor>

```

```

    <cm:ContingencyFactor name="Le degré d'interaction avec les utilisateurs finaux du système"
targetId="C5_2">faible</cm:ContingencyFactor>
    <cm:ContingencyFactor name="L'implication du client dans le processus de développement"
targetId="C5_2">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Coût de mise en œuvre de l'alternative pour l'équipe"
targetId="C5_2">6</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Taille du projet" targetId="C6_2">grande</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C6_2">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Le degré d'interaction avec les utilisateurs finaux du système"
targetId="C6_2">fort</cm:ContingencyFactor>
    <cm:ContingencyFactor name="L'implication du client dans le processus de développement"
targetId="C6_2">moyenne</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Coût de mise en œuvre de l'alternative pour l'équipe"
targetId="C6_2">4</cm:ContingencyFactor>
  </cm:ContingencyFactorList>
</cm:Decision>
<cm:Decision targetId="D5_1">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G13_1">Est-ce que la liste des besoins est
complète?</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList/>
</cm:Decision>
<cm:Decision targetId="D5_2">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G13_2">Est-ce que la liste des besoins est
complète?</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList/>
</cm:Decision>
<cm:Decision targetId="D6_1">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G14_1">Au vu de l'ensemble des besoins définis dans la
liste des besoins est que le projet est viable et faisable?</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList/>
</cm:Decision>
<cm:Decision targetId="D6_2">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G14_2">Au vu de l'ensemble des besoins définis dans la
liste des besoins est que le projet est viable et faisable?</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList/>
</cm:Decision>
<cm:Decision targetId="D7_1">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G15_1">Il est recommandé de choisir C7_1 si le projet est
de petite taille mais avec un niveau de complexité important.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G16_1">Il est recommandé de choisir C7_1 si le projet a
besoin d'une méthode de planification flexible mais en contrepartie les variations de date de
livraison seront plus rigides.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G17_1">Il est recommandé de choisir C8_1 si le projet est
de petite taille et de faible complexité.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G18_1">Il est recommandé de choisir C8_1 si le projet peut
avoir plus de souplesse dans les livraisons des incréments logiciel en contrepartie la méthode
de planification utilisée est plus rigide.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G19_1">Il est recommandé de choisir C9_1 si le projet est
de grande taille et complexe.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G20_1">Il est recommandé de choisir C9_1 si le projet
comporte des itération de développement longues avec peu de flexibilité dans la
planification.</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList>
    <cm:ContingencyFactor name="Taille du projet" targetId="C7_1">petite</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C7_1">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Type de planification" targetId="C7_1">{au produit, au délai, au
budget}</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Permet de respecter les délais"
targetId="C7_1">facilement</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Longueur d'un cycle itératif"
targetId="C7_1">court</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Taille du projet" targetId="C8_1">petite</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C8_1">faible</cm:ContingencyFactor>

```

```

    <cm:ContingencyFactor name="Type de planification" targetId="C8_1">{au
produit}</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Permet de respecter les délais"
targetId="C8_1">difficilement</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Longueur d'un cycle itératif"
targetId="C8_1">court</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Taille du projet" targetId="C9_1">grande</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C9_1">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Type de planification" targetId="C9_1">{au
produit}</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Permet de respecter les délais"
targetId="C9_1">facilement</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Longueur d'un cycle itératif"
targetId="C9_1">long</cm:ContingencyFactor>
  </cm:ContingencyFactorList>
</cm:Decision>
<cm:Decision targetId="D7_2">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G15_2">Il est recommandé de choisir C7_2 si le projet est
de petite taille mais avec un niveau de complexité important.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G16_2">Il est recommandé de choisir C7_2 si le projet a
besoin d'une méthode de planification flexible mais en contrepartie les variations de date de
livraison seront plus rigides.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G17_2">Il est recommandé de choisir C8_2 si le projet est
de petite taille et de faible complexité.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G18_2">Il est recommandé de choisir C8_2 si le projet peut
avoir plus de souplesse dans les livraisons des incréments logiciel en contrepartie la méthode
de planification utilisée est plus rigide.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G19_2">Il est recommandé de choisir C9_2 si le projet est
de grande taille et complexe.</cm:ConfigurationGuideline>
    <cm:ConfigurationGuideline name="G20_2">Il est recommandé de choisir C9_2 si le projet
comporte des itération de développement longues avec peu de flexibilité dans la
planification.</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList>
    <cm:ContingencyFactor name="Taille du projet" targetId="C7_2">petite</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C7_2">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Type de planification" targetId="C7_2">{au produit, au délai, au
budget}</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Permet de respecter les délais"
targetId="C7_2">facilement</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Longueur d'un cycle itératif"
targetId="C7_2">court</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Taille du projet" targetId="C8_2">petite</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C8_2">faible</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Type de planification" targetId="C8_2">{au
produit}</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Permet de respecter les délais"
targetId="C8_2">difficilement</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Longueur d'un cycle itératif"
targetId="C8_2">court</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Taille du projet" targetId="C9_2">grande</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Complexité du projet"
targetId="C9_2">forte</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Type de planification" targetId="C9_2">{au
produit}</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Permet de respecter les délais"
targetId="C9_2">facilement</cm:ContingencyFactor>
    <cm:ContingencyFactor name="Longueur d'un cycle itératif"
targetId="C9_2">long</cm:ContingencyFactor>
  </cm:ContingencyFactorList>
</cm:Decision>
<cm:Decision targetId="D8_1">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G21_1">Est-ce que la planification du projet et la
définition des besoins nécessitent un affinement?</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList/>
</cm:Decision>
<cm:Decision targetId="D8_2">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G21_2">Est-ce que la planification du projet et la
définition des besoins nécessitent un affinement?</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>

```



```
<cm:ContingencyFactorList/>
</cm:Decision>
<cm:Decision targetId="D9_1">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G22_1">Est-ce que la planification du projet et la
définition des besoins nécessitent un affinement supplémentaire?</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList/>
</cm:Decision>
<cm:Decision targetId="D9_2">
  <cm:ConfigurationGuidelineList>
    <cm:ConfigurationGuideline name="G22_2">Est-ce que la planification du projet et la
définition des besoins nécessitent un affinement supplémentaire?</cm:ConfigurationGuideline>
  </cm:ConfigurationGuidelineList>
  <cm:ContingencyFactorList/>
</cm:Decision>
</cm:ConfigurationModel>
```

Annexe I

Descripteurs des SMA de la ligne de méthode de lancement d'un projet agile

(chapitre 6 section 6.4)

1. Descripteur du SMA C1 - Etude de marché

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-paris1.fr/schema/wSDL/BusinessStudy.wSDL"
  xmlns="http://www.w3.org/ns/wSDL"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://BusinessStudy/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewSDL="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/BusinessStudy"
  xmlns:wssoap="http://www.w3.org/ns/wSDL/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wSDLx="http://www.w3.org/ns/wSDL-extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/BusinessStudy">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/BusinessStudy"
        schemaLocation="http://crinfo.univ-paris1.fr/schema/BusinessStudy.xsd"/>
      <xsd:element name="BusinessStudyRequest"/>
      <xsd:element name="BusinessStudyResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:DocumentEtudeDeMarche" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <interface name="BusinessStudyInterface"
    smewSDL:conceptualMethodPuzzle="http://crinfo.univ-paris1.fr/chunk/BusinessStudy.html"
    smewSDL:puzzleType="atomic-method-service"
    smewSDL:technicalMethodPuzzle="http://crinfo.univ-paris1.fr/wsrp/wSDL/BusinessStudyPortlet.wSDL"
    smewSDL:smeConcept="http://crinfo.univ-paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewSDL:modelReference="http://crinfo.univ-paris1.fr/schema/agileMethodOntology.owl#DynamicSystemDevelopmentMethod http://crinfo.univ-paris1.fr/schema/agileMethodOntology.owl#ProjectFesability"
  >
    <operation name="performBusinessStudy"
      smewSDL:smeConcept="http://crinfo.univ-paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewSDL:modelReference="http://crinfo.univ-paris1.fr/schema/agileMethodOntology.owl#BusinessStudy"
    >
      <input messageLabel="BusinessStudyInputMessage" element="xsdio:BusinessStudyRequest"
/>
      <output messageLabel="BusinessStudyOutputMessage"
        element="xsdio:BusinessStudyResponse" />
    </operation>
  </interface>

  <binding name="BusinessStudyBinding"
    interface="tns:BusinessStudyInterface"
    type="http://www.w3.org/ns/wSDL/soap"
    wssoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <operation ref="tns:performBusinessStudy" />
  </binding>

  <service name="BusinessStudy"
    interface="tns:BusinessStudyInterface">
```

```
<endpoint name="BusinessStudyEndpoint"
          binding="tns:BusinessStudyBinding"
          address =""/>

</service>

</description>
```

2. Descripteur du SMA C2 - Etude technique

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-paris1.fr/schema/wsdl/TechnicalStudy.wsdl"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://TechnicalStudy/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewSDL="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/TechnicalStudy"
  xmlns:soap="http://www.w3.org/ns/wsdl/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wSDLx="http://www.w3.org/ns/wsdl-extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/TechnicalStudy">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/TechnicalStudy"
        schemaLocation="http://crinfo.univ-paris1.fr/schema/TechnicalStudy.xsd"/>
      <xsd:element name="TechnicalStudyRequest"/>
      <xsd:element name="TechnicalStudyResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:DocumentEtudeTechnique" minOccurs="1"
              maxOccurs="1"/>
            <xsd:element ref="xsdio:Acteur" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="xsdio:Role" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>

  <interface name="TechnicalStudyInterface"
    smewSDL:conceptualMethodPuzzle="http://crinfo.univ-
    paris1.fr/chunk/TechnicalStudy"
    smewSDL:puzzleType="atomic-method-service"
    smewSDL:technicalMethodPuzzle="http://crinfo.univ-
    paris1.fr/wsrp/wsdl/TechnicalStudyPortlet.wsdl"
    smewSDL:smeConcept="http://crinfo.univ-
    paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-
    paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewSDL:modelReference="http://crinfo.univ-
    paris1.fr/schema/agileMethodOntology.owl#DynamicSystemDevelopmentMethod http://crinfo.univ-
    paris1.fr/schema/agileMethodOntology.owl#ProjectFesability"
  >
    <operation name="performTechnicalStudy"
      smewSDL:smeConcept="http://crinfo.univ-
      paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewSDL:modelReference="http://crinfo.univ-
      paris1.fr/schema/agileMethodOntology.owl#TechnicalStudy"
    >
      <input messageLabel="TechnicalStudyInputMessage" element="xsdio:TechnicalStudyRequest"
/>
      <output messageLabel="TechnicalStudyOutputMessage"
        element="xsdio:TechnicalStudyResponse" />
    </operation>
  </interface>

  <binding name="TechnicalStudyBinding"
    interface="tns:TechnicalStudyInterface"
    type="http://www.w3.org/ns/wsdl/soap"
    soap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
  >
    <operation ref="tns:performTechnicalStudy" />
  </binding>
  <service name="TechnicalStudy"
    interface="tns:TechnicalStudyInterface">
    <endpoint name="TechnicalStudyEndpoint"
      binding="tns:TechnicalStudyBinding"
      address =""/>
  </service>
</description>
```

3. Descripteur du SMA C3 - Abandon du projet

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-paris1.fr/schema/wsdl/ProjectCloseOut.wsdl"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://ProjectCloseOut/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewSDL="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/ProjectCloseOut"
  xmlns:soap="http://www.w3.org/ns/wsdl/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wSDLx="http://www.w3.org/ns/wsdl-extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/ProjectCloseOut">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/ProjectCloseOut"
        schemaLocation="http://crinfo.univ-paris1.fr/schema/ProjectCloseOut.xsd"/>
      <xsd:element name="ProjectCloseOutRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:DocumentEtudeDeFaisabilite" minOccurs="1"
maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="ProjectCloseOutResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:DocumentEtudeDeFaisabilite" minOccurs="1"
maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <interface name="ProjectCloseOutInterface">
    smewSDL:conceptualMethodPuzzle="http://crinfo.univ-
paris1.fr/chunk/ProjectCloseOut.html"
    smewSDL:puzzleType="atomic-method-service"
    smewSDL:technicalMethodPuzzle="http://crinfo.univ-
paris1.fr/wsrp/wsdl/ProjectCloseOutPortlet.wsdl"
    smewSDL:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewSDL:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#DynamicSystemDevelopmentMethod http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#ProjectCloseOut"
  >
    <operation name="performProjectCloseOut">
      smewSDL:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewSDL:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#ProjectCloseOut"
    >
      <input messageLabel="ProjectCloseOutInputMessage"
element="xsdio:ProjectCloseOutRequest" />
      <output messageLabel="ProjectCloseOutOutputMessage"
element="xsdio:ProjectCloseOutResponse" />
    </operation>
  </interface>
  <binding name="ProjectCloseOutBinding">
    interface="tns:ProjectCloseOutInterface"
    type="http://www.w3.org/ns/wsdl/soap"
    soap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
    <operation ref="tns:performProjectCloseOut" />
  </binding>
  <service name="ProjectCloseOut">
    interface="tns:ProjectCloseOutInterface">
      <endpoint name="ProjectCloseOutEndpoint"
        binding="tns:ProjectCloseOutBinding"
        address =""/>
    </service>
</description>
```

4. Descripteur du SMA C4 - Définition de la liste des besoins du produit

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-paris1.fr/schema/wsdl/BacklogDefinition.wsdl"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://AgileProjetBeginning/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewSDL="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/BacklogDefinition"
  xmlns:soap="http://www.w3.org/ns/wsdl/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wSDLx="http://www.w3.org/ns/wsdl-extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/BacklogDefinition">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/BacklogDefinition"
        schemaLocation="http://crinfo.univ-paris1.fr/schema/BacklogDefinition.xsd"/>
      <xsd:element name="BacklogDefinitionRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="BacklogDefinitionResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <interface name="BacklogDefinitionInterface"
    smewSDL:conceptualMethodPuzzle="http://crinfo.univ-paris1.fr/chunk/BacklogDefinition"
    smewSDL:puzzleType="atomic-method-service"
    smewSDL:technicalMethodPuzzle="http://crinfo.univ-paris1.fr/wsrp/wsdl/BacklogDefinitionPortlet.wsdl"
    smewSDL:smeConcept="http://crinfo.univ-paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewSDL:modelReference="http://crinfo.univ-paris1.fr/schema/agileMethodOntology.owl#Scrum http://crinfo.univ-paris1.fr/schema/agileMethodOntology.owl#RequirementsDefinition "
  >
    <operation name="performBacklogDefinition"
      smewSDL:smeConcept="http://crinfo.univ-paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewSDL:modelReference="http://crinfo.univ-paris1.fr/schema/agileMethodOntology.owl#BacklogDefinition"
    >
      <input messageLabel="BacklogDefinitionInputMessage"
        element="xsdio:BacklogDefinitionRequest" />
      <output messageLabel="BacklogDefinitionOutputMessage"
        element="xsdio:BacklogDefinitionResponse" />
    </operation>
  </interface>
  <binding name="BacklogDefinitionBinding"
    interface="tns:BacklogDefinitionInterface"
    type="http://www.w3.org/ns/wsdl/soap"
    soap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
  >
    <operation ref="tns:performBacklogDefinition" />
  </binding>
  <service name="BacklogDefinition"
    interface="tns:BacklogDefinitionInterface">
    <endpoint name="BacklogDefinitionEndpoint"
      binding="tns:BacklogDefinitionBinding"
      address = ""/>
  </service>
</description>
```

5. Descripteur du SMA C5 - Définition de besoin utilisateur

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<description targetNamespace="http://crinfo.univ-
paris1.fr/schema/wsd/USerStoriesDefinition.wsd"
  xmlns="http://www.w3.org/ns/wsd"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://UserStoriesDefinition/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewsd="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/UserStoriesDefinition"
  xmlns:wsoap="http://www.w3.org/ns/wsd/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsdlx="http://www.w3.org/ns/wsd-extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/UserStoriesDefinition">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/UserStoriesDefinition"
        schemaLocation="http://crinfo.univ-paris1.fr/schema/UserStoriesDefinition.xsd"/>
      <xsd:element name="UserStoriesDefinitionRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="UserStoriesDefinitionResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <interface name="UserStoriesDefinitionInterface"
    smewsd:conceptualMethodPuzzle="http://crinfo.univ-
paris1.fr/chunk/UserStoriesDefinition"
    smewsd:puzzleType="atomic-method-service"
    smewsd:technicalMethodPuzzle="http://crinfo.univ-
paris1.fr/wsrp/wsd/USerStoriesDefinitionPortlet.wsd"
    smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#ExtremeProgramming http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#RequirementsDefinition"
  >
    <operation name="performUserStoriesDefinition"
      smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#UserStoriesDefinition"
    >
      <input messageLabel="UserStoriesDefinitionInputMessage"
        element="xsdio:UserStoriesDefinitionRequest" />
      <output messageLabel="UserStoriesDefinitionOutputMessage"
        element="xsdio:UserStoriesDefinitionResponse" />
    </operation>
  </interface>
  <binding name="UserStoriesDefinitionBinding"
    interface="tns:UserStoriesDefinitionInterface"
    type="http://www.w3.org/ns/wsd/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
  >
    <operation ref="tns:performUserStoriesDefinition" />
  </binding>
  <service name="UserStoriesDefinition"
    interface="tns:UserStoriesDefinitionInterface">
    <endpoint name="UserStoriesDefinitionEndpoint"
      binding="tns:UserStoriesDefinitionBinding"
      address =""/>
  </service>
</description>
```

6. Descripteur du SMA C6 - Définition des besoins de haut niveau

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-
paris1.fr/schema/wsd/HighLevelRequirementsDefinition.wsd"
  xmlns="http://www.w3.org/ns/wsd"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://HighLevelRequirementsDefinition/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewsd="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/HighLevelRequirementsDefinition"
  xmlns:wsoap="http://www.w3.org/ns/wsd/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsdlx="http://www.w3.org/ns/wsd/extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-
paris1.fr/schema/HighLevelRequirementsDefinition">
      <xsd:import namespace="http://crinfo.univ-
paris1.fr/schema/HighLevelRequirementsDefinition" schemaLocation="http://crinfo.univ-
paris1.fr/schema/HighLevelRequirementsDefinition.xsd"/>
      <xsd:element name="HighLevelRequirementsDefinitionRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:DocumentEtudeDeMarche" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="HighLevelRequirementsDefinitionResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <interface name="HighLevelRequirementsDefinitionInterface"
    smewsd:conceptualMethodPuzzle="http://crinfo.univ-
paris1.fr/chunk/HighLevelRequirementsDefinition"
    smewsd:puzzleType="atomic-method-service"
    smewsd:technicalMethodPuzzle="http://crinfo.univ-
paris1.fr/wsrp/wsd/HighLevelRequirementsDefinitionPortlet.wsd"
    smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#DynamicSystemDevelopmentMethod http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#RequirementsDefinition"
  >
    <operation name="performHighLevelRequirementsDefinition"
      smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#HighLevelRequirementsDefinition"
    >
      <input messageLabel="HighLevelRequirementsDefinitionInputMessage"
        element="xsdio:HighLevelRequirementsDefinitionRequest" />
      <output messageLabel="HighLevelRequirementsDefinitionOutputMessage"
        element="xsdio:HighLevelRequirementsDefinitionResponse" />
    </operation>
  </interface>
  <binding name="HighLevelRequirementsDefinitionBinding"
    interface="tns:HighLevelRequirementsDefinitionInterface"
    type="http://www.w3.org/ns/wsd/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <operation ref="tns:performHighLevelRequirementsDefinition" />
  </binding>
  <service name="HighLevelRequirementsDefinition"
    interface="tns:HighLevelRequirementsDefinitionInterface">
    <endpoint name="HighLevelRequirementsDefinitionEndpoint"
      binding="tns:HighLevelRequirementsDefinitionBinding"
      address="" />
  </service>
</description>
```


7. Descripteur du SMA C7 - Planification générale des livrables

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-paris1.fr/schema/wsdl/ReleasePlanning.wsdl"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://ReleasePlanning/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewSDL="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/ReleasePlanning"
  xmlns:soap="http://www.w3.org/ns/wsdl/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wSDLx="http://www.w3.org/ns/wsdl-extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/ReleasePlanning">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/ReleasePlanning"
        schemaLocation="http://crinfo.univ-paris1.fr/schema/ReleasePlanning.xsd"/>
      <xsd:element name="ReleasePlanningRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="ReleasePlanningResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:PlanningDesReleases" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <interface name="ReleasePlanningInterface"
    smewSDL:conceptualMethodPuzzle="http://crinfo.univ-paris1.fr/chunk/ReleasePlanning"
    smewSDL:puzzleType="atomic-method-service"
    smewSDL:technicalMethodPuzzle="http://crinfo.univ-paris1.fr/wsrp/wsdl/ReleasePlanningPortlet.wsdl"
    smewSDL:smeConcept="http://crinfo.univ-paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewSDL:modelReference="http://crinfo.univ-paris1.fr/schema/agileMethodOntology.owl#Scrum http://crinfo.univ-paris1.fr/schema/agileMethodOntology.owl#ReleasePlanning"
  >
    <operation name="performReleasePlanning"
      smewSDL:smeConcept="http://crinfo.univ-paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewSDL:modelReference="http://crinfo.univ-paris1.fr/schema/agileMethodOntology.owl#ReleasePlanning"
    >
      <input messageLabel="ReleasePlanningInputMessage" element="xsdio:ReleasePlanningRequest" />
      <output messageLabel="ReleasePlanningOutputMessage"
        element="xsdio:ReleasePlanningResponse" />
    </operation>
  </interface>
  <binding name="ReleasePlanningBinding"
    interface="tns:ReleasePlanningInterface"
    type="http://www.w3.org/ns/wsdl/soap"
    soap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
  >
    <operation ref="tns:performReleasePlanning" />
  </binding>
  <service name="ReleasePlanning"
    interface="tns:ReleasePlanningInterface">
    <endpoint name="ReleasePlanningEndpoint"
      binding="tns:ReleasePlanningBinding"
      address =""/>
  </service>
</description>
```

8. Descripteur du SMA C8 - Définition du plan général des livrables

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-
paris1.fr/schema/wsd/ReleasePlanDefinition.wsd"
  xmlns="http://www.w3.org/ns/wsd"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://ReleasePlanDefinition/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewsd="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/ReleasePlanDefinition"
  xmlns:wsoap="http://www.w3.org/ns/wsd/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/ReleasePlanDefinition">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/ReleasePlanDefinition"
schemaLocation="http://crinfo.univ-paris1.fr/schema/ReleasePlanDefinition.xsd"/>
      <xsd:element name="ReleasePlanDefinitionRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="ReleasePlanDefinitionResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:PlanningDesReleases" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <interface name="ReleasePlanDefinitionInterface"
    smewsd:conceptualMethodPuzzle="http://crinfo.univ-
paris1.fr/chunk/ReleasePlanDefinition"
    smewsd:puzzleType="atomic-method-service"
    smewsd:technicalMethodPuzzle="http://crinfo.univ-
paris1.fr/wsrp/wsd/ReleasePlanDefinitionPortlet.wsd"
    smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#ExtremeProgramming http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#ReleasePlanning"
  >
    <operation name="performReleasePlanDefinition"
      smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#ReleasePlanDefinition"
    >
      <input messageLabel="ReleasePlanDefinitionInputMessage"
element="xsdio:ReleasePlanDefinitionRequest" />
      <output messageLabel="ReleasePlanDefinitionOutputMessage"
element="xsdio:ReleasePlanDefinitionResponse" />
    </operation>
  </interface>
  <binding name="ReleasePlanDefinitionBinding"
    interface="tns:ReleasePlanDefinitionInterface"
    type="http://www.w3.org/ns/wsd/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
  >
    <operation ref="tns:performReleasePlanDefinition" />
  </binding>

  <service name="ReleasePlanDefinition"
    interface="tns:ReleasePlanDefinitionInterface">
    <endpoint name="ReleasePlanDefinitionEndpoint"
      binding="tns:ReleasePlanDefinitionBinding"
      address =""/>
  </service>
</description>
```

9. Descripteur du SMA C9 - Définition du plan directeur

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-
paris1.fr/schema/wsd/OutlinePlanDefinition.wsd"
  xmlns="http://www.w3.org/ns/wsd"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://OutlinePlanDefinition/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewsd="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/OutlinePlanDefinition"
  xmlns:wsoap="http://www.w3.org/ns/wsd/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsdlx="http://www.w3.org/ns/wsd-extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/OutlinePlanDefinition">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/OutlinePlanDefinition"
schemaLocation="http://crinfo.univ-paris1.fr/schema/OutlinePlanDefinition.xsd"/>
      <xsd:element name="OutlinePlanDefinitionRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="OutlinePlanDefinitionResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:PlanningDesReleases" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <interface name="OutlinePlanDefinitionInterface"
    smewsd:conceptualMethodPuzzle="http://crinfo.univ-
paris1.fr/chunk/OutlinePlanDefinition"
    smewsd:puzzleType="atomic-method-service"
    smewsd:technicalMethodPuzzle="http://crinfo.univ-
paris1.fr/wsrp/wsd/OutlinePlanDefinitionPortlet.wsd"
    smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#DynamicSystemDevelopmentMethod http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#ReleasePlanning"
  >
    <operation name="performOutlinePlanDefinition"
      smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl# OutlinePlanDefinition"
    >
      <input messageLabel="OutlinePlanDefinitionInputMessage"
element="xsdio:OutlinePlanDefinitionRequest" />
      <output messageLabel="OutlinePlanDefinitionOutputMessage"
element="xsdio:OutlinePlanDefinitionResponse" />
    </operation>
  </interface>
  <binding name="OutlinePlanDefinitionBinding"
    interface="tns:OutlinePlanDefinitionInterface"
    type="http://www.w3.org/ns/wsd/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
  >
    <operation ref="tns:performOutlinePlanDefinition" />
  </binding>
  <service name="OutlinePlanDefinition"
    interface="tns:OutlinePlanDefinitionInterface">
    <endpoint name="OutlinePlanDefinitionEndpoint"
      binding="tns:OutlinePlanDefinitionBinding"
      address = ""/>
  </service>
</description>
```

10. Descripteur du SMA C10 - Affinement des besoins

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<description targetNamespace="http://crinfo.univ-
paris1.fr/schema/wsdl/RequirementsRefinement.wsdl"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://RequirementsRefinement/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewSDL="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/RequirementsRefinement"
  xmlns:wsoap="http://www.w3.org/ns/wsdl/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wSDLx="http://www.w3.org/ns/wsdl-extensions"
>
  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/RequirementsRefinement">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/RequirementsRefinement"
schemaLocation="http://crinfo.univ-paris1.fr/schema/RequirementsRefinement.xsd"/>
      <xsd:element name="RequirementsRefinementRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="RequirementsRefinementResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <interface name="RequirementsRefinementInterface"
    smewSDL:conceptualMethodPuzzle="http://crinfo.univ-
paris1.fr/chunk/RequirementsRefinement"
    smewSDL:puzzleType="atomic-method-service"
    smewSDL:technicalMethodPuzzle="http://crinfo.univ-
paris1.fr/wsrp/wsdl/RequirementsRefinementPortlet.wsdl"
    smewSDL:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewSDL:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#DynamicSystemDevelopmentMethod http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#RequirementsDefinition"
  >
    <operation name="performRequirementsRefinement"
      smewSDL:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewSDL:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#RequirementsRefinement"
    >
      <input messageLabel="RequirementsRefinementInputMessage"
element="xsdio:RequirementsRefinementRequest" />
      <output messageLabel="RequirementsRefinementOutputMessage"
element="xsdio:RequirementsRefinementResponse" />
    </operation>
  </interface>

  <binding name="RequirementsRefinementBinding"
    interface="tns:RequirementsRefinementInterface"
    type="http://www.w3.org/ns/wsdl/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
  >
    <operation ref="tns:performRequirementsRefinement" />
  </binding>

  <service name="RequirementsRefinement"
    interface="tns:RequirementsRefinementInterface">

    <endpoint name="RequirementsRefinementEndpoint"
      binding="tns:RequirementsRefinementBinding"
      address =""/>

  </service>
</description>
```

11. Descripteur du SMA C11 - Définition du planning de développement

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<description targetNamespace="http://crinfo.univ-
paris1.fr/schema/wsd/DevelopmentPlanDefinition.wsd"
  xmlns="http://www.w3.org/ns/wsd"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:tns="http://DevelopmentPlanDefinition/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:smewsd="http://crinfo.univ-paris1.fr/schema/SMEWSDL.xsd"
  xmlns:xsdio="http://crinfo.univ-paris1.fr/schema/DevelopmentPlanDefinition"
  xmlns:wsoap="http://www.w3.org/ns/wsd/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsdlx="http://www.w3.org/ns/wsd-extensions"
>

  <types>
    <xsd:schema targetNamespace="http://crinfo.univ-
paris1.fr/schema/DevelopmentPlanDefinition">
      <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/DevelopmentPlanDefinition"
schemaLocation="http://crinfo.univ-paris1.fr/schema/DevelopmentPlanDefinition.xsd"/>
      <xsd:element name="DevelopmentPlanDefinitionRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:Release" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="xsdio:Acteur" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="xsdio:ListeDesBesoins" minOccurs="1" maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="DevelopmentPlanDefinitionResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsdio:Release" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="xsdio:Develope" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="xsdio:Acteur" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="xsdio:Participant" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>

  <interface name="DevelopmentPlanDefinitionInterface"
    smewsd:conceptualMethodPuzzle="http://crinfo.univ-
paris1.fr/chunk/DevelopmentPlanDefinition"
    smewsd:puzzleType="atomic-method-service"
    smewsd:technicalMethodPuzzle="http://crinfo.univ-
paris1.fr/wsrp/wsd/DevelopmentPlanDefinitionPortlet.wsd"
    smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Method http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#MethodPuzzle"
    smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#DynamicSystemDevelopmentMethod http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#ReleasePlanning"
  >
    <operation name="performDevelopmentPlanDefinition"
      smewsd:smeConcept="http://crinfo.univ-
paris1.fr/schema/MethodPuzzleOntology.owl#Process"
      smewsd:modelReference="http://crinfo.univ-
paris1.fr/schema/agileMethodOntology.owl#DevelopmentPlanDefinition"
    >
      <input messageLabel="DevelopmentPlanDefinitionInputMessage"
element="xsdio:DevelopmentPlanDefinitionRequest" />
      <output messageLabel="DevelopmentPlanDefinitionOutputMessage"
element="xsdio:DevelopmentPlanDefinitionResponse" />
    </operation>
  </interface>
  <binding name="DevelopmentPlanDefinitionBinding"
    interface="tns:DevelopmentPlanDefinitionInterface"
    type="http://www.w3.org/ns/wsd/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <operation ref="tns:performDevelopmentPlanDefinition" />
  </binding>
</description>
```

```
</binding>
<service name="DevelopmentPlanDefinition"
  interface="tns:DevelopmentPlanDefinitionInterface">
  <endpoint name="DevelopmentPlanDefinitionEndpoint"
    binding="tns:DevelopmentPlanDefinitionBinding"
    address =""/>
</service>
</description>
```

Annexe J

Schéma XML du modèle de produit global de la ligne de méthode de lancement d'un projet agile

(chapitre 6 section 6.4.12)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://crinfo.univ-paris1.fr/schema/ML-PM-LancementProjetAgile"
  xmlns:tns="http://crinfo.univ-paris1.fr/schema/ML-PM-LancementProjetAgile"
  elementFormDefault="qualified"
  >

  <xsd:element name="ModeleDeProduit">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:DossierProjet" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="tns:Acteur" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="tns:Role" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="tns:Participant" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="tns:Iteration" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="tns:Developpe" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="DossierProjet">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:DocumentEtudeDeFaisabilite" minOccurs="0" maxOccurs="1" />
        <xsd:element ref="tns:ListeDesBesoins" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="tns:PlanningDesLivrables" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="DocumentEtudeDeFaisabilite">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:DocumentEtudeDeMarche" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="tns:DocumentEtudeTechnique" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="decision" use="optional" type="xsd:string" />
      <xsd:attribute name="justification" use="optional" type="xsd:string" />
      <xsd:attribute name="etat" use="required" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="DocumentEtudeDeMarche">
    <xsd:complexType>
      <xsd:attribute name="id" use="required" type="xsd:string" />
      <xsd:attribute name="problematiqueMetier" use="optional" type="xsd:string" />
      <xsd:attribute name="activitesVisees" use="optional" type="xsd:string" />
      <xsd:attribute name="activitesNonVisees" use="optional" type="xsd:string" />
      <xsd:attribute name="impact" use="optional" type="xsd:string" />
      <xsd:attribute name="etatDesLieuxFonctionnel" use="optional" type="xsd:string" />
      <xsd:attribute name="etatDesLieuxTechnique" use="optional" type="xsd:string" />
      <xsd:attribute name="scenario" use="optional" type="xsd:string" />
      <xsd:attribute name="cout" use="optional" type="xsd:string" />
      <xsd:attribute name="resultat" use="optional" type="xsd:string" />
      <xsd:attribute name="justification" use="optional" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="DocumentEtudeTechnique">
    <xsd:complexType>
      <xsd:attribute name="id" use="required" type="xsd:string" />
      <xsd:attribute name="benefices" use="optional" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

```

```

        <xsd:attribute name="estimationDuDélai" use="optional" type="xsd:string" />
        <xsd:attribute name="resultat" use="optional" type="xsd:string" />
        <xsd:attribute name="justification" use="required" type="xsd:string" />
        <xsd:attribute name="documentationTechnique" use="required" type="xsd:string" />
        <xsd:attribute name="contraintes" use="required" type="xsd:string" />
        <xsd:attribute name="solutions" use="required" type="xsd:string" />
        <xsd:attribute name="automatisation" use="required" type="xsd:string" />
    </xsd:complexType>
</xsd:element>

<xsd:element name="ListeDesBesoins">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="tns:Besoin" minOccurs="1" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="Besoin" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="tns:Besoin" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="id" use="required" type="xsd:string" />
        <xsd:attribute name="nom" use="required" type="xsd:string" />
        <xsd:attribute name="description" use="optional" type="xsd:string" />
        <xsd:attribute name="priorite" use="optional" type="xsd:string" />
        <xsd:attribute name="categorie" use="optional" type="xsd:string" />
        <xsd:attribute name="estimation" use="optional" type="xsd:string" />
        <xsd:attribute name="type" use="optional" type="xsd:string" />
        <xsd:attribute name="etat" use="required" type="xsd:string" />
        <xsd:attribute name="ordonancement" use="optional" type="xsd:string" />
    </xsd:complexType>
</xsd:element>

<xsd:element name="PlanningDesLivrables">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="tns:Livable" minOccurs="1" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="dateDebut" use="required" type="xsd:dateTime" />
        <xsd:attribute name="dateFin" use="required" type="xsd:dateTime" />
        <xsd:attribute name="problematiqueMetier" use="optional" type="xsd:string" />
        <xsd:attribute name="cout" use="optional" type="xsd:string" />
        <xsd:attribute name="resultat" use="optional" type="xsd:string" />
        <xsd:attribute name="justification" use="optional" type="xsd:string" />
    </xsd:complexType>
</xsd:element>

<xsd:element name="Livable">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="tns:Iteration" minOccurs="0" maxOccurs="unbounded" />
            <xsd:element ref="tns:Implemente" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="numero" use="required" type="xsd:string" />
        <xsd:attribute name="dateLivraison" use="required" type="xsd:dateTime" />
        <xsd:attribute name="conditionDeFinDeLivable" use="required" type="xsd:string" />
        <xsd:attribute name="dureeIteration" use="required" type="xsd:string" />
        <xsd:attribute name="description" use="optional" type="xsd:string" />
        <xsd:attribute name="ressourcesMaterielles" use="optional" type="xsd:string" />
    </xsd:complexType>
</xsd:element>

<xsd:element name="Iteration">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:attribute name="id" use="required" type="xsd:string" />
            <xsd:attribute name="type" use="optional" type="xsd:string" />
            <xsd:attribute name="priorite" use="optional" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="Developpe">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:attribute name="idBesoin" use="required" type="xsd:string" />
            <xsd:attribute name="idIteration" use="required" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>

```



```

</xsd:element>

<xsd:element name="Implemente">
  <xsd:complexType>
    <xsd:attribute name="idBesoin" use="required" type="xsd:string" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="Acteur">
  <xsd:complexType>
    <xsd:attribute name="id" use="required" type="xsd:string" />
    <xsd:attribute name="nom" use="required" type="xsd:string" />
    <xsd:attribute name="prenom" use="required" type="xsd:string" />
    <xsd:attribute name="competences" use="optional" type="xsd:string" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="Role">
  <xsd:complexType>
    <xsd:attribute name="idActeur" use="required" type="xsd:string" />
    <xsd:attribute name="idDocumentEtudeTechnique" use="required" type="xsd:string" />
    <xsd:attribute name="nom" use="required" type="xsd:string" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="Participant">
  <xsd:complexType>
    <xsd:attribute name="idActeur" use="required" type="xsd:string" />
    <xsd:attribute name="numeroLivrable" use="required" type="xsd:string" />
    <xsd:attribute name="role" use="required" type="xsd:string" />
    <xsd:attribute name="responsabilite" use="optional" type="xsd:string" />
  </xsd:complexType>
</xsd:element>

</xsd:schema>

```

Annexe K

Arbre d'exécution de la ligne de méthode de lancement d'un projet agile

(chapitre 6 section 6.5)

```
<?xml version="1.0" encoding="UTF-8"?>
<ml:MethodLine xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:ml='http://crinfo.univ-paris1.fr/schema/MethodLine'
xsi:schemaLocation='http://crinfo.univ-paris1.fr/schema/MethodLine MethodLine.xsd'
name="Lancement d'un projet agile" version="1.0" configurationUrl="http://.../guidelines.xml"
assignUrl="http://.../productassignment.xml">
<ml:DecisionPoint id="D1" min="1" max="1">
  <ml:Sequence id="Seq1">
    <ml:DecisionPoint id="D2" min="1" max="2">
      <ml:AtomicMethodService id="C1" name="Etude de marché"
url="http://.../BusinessStudyPortlet.wsdl"/>
      <ml:AtomicMethodService id="C2" name="Etude technique"
url="http://.../TechnicalStudyPortlet.wsdl"/>
    </ml:DecisionPoint>
    <ml:DecisionPoint id="D4" min="1" max="1">
      <ml:AtomicMethodService id="C3_1" name="Abandon du projet"
url="http://.../ProjectCloseOutPortlet.wsdl"/>
      <ml:Sequence id="Seq2_1">
        <ml:RepeatUntil id="D5_1">
          <ml:DecisionPoint id="D3_1" min="1" max="1">
            <ml:AtomicMethodService id="C4_1" name="Définition de la liste des besoins du
produit"
url="http://.../BacklogDefinitionPortlet.wsdl"/>
            <ml:AtomicMethodService id="C5_1" name="Définition de besoin utilisateur"
url="http://.../UserStoriesDefinitionPortlet.wsdl"/>
            <ml:AtomicMethodService id="C6_1" name="Définition des besoins de haut niveau"
url="http://.../HighLevelRequirementsDefinitionPortlet.wsdl"/>
          </ml:DecisionPoint>
        </ml:RepeatUntil>
        <ml:DecisionPoint id="D6_1" min="1" max="1">
          <ml:AtomicMethodService id="C3_2" name="Abandon du projet"
url="http://.../ProjectCloseOutPortlet.wsdl"/>
        </ml:DecisionPoint>
        <ml:Sequence id="Seq3_1">
          <ml:DecisionPoint id="D7_1" min="1" max="1">
            <ml:AtomicMethodService id="C7_1" name="Planification générale des livrables"
url="http://.../ReleasePlanningPortlet.wsdl"/>
            <ml:AtomicMethodService id="C8_1" name="Définition du plan général des livrables"
url="http://.../ReleasePlanDefinitionPortlet.wsdl"/>
            <ml:AtomicMethodService id="C9_1" name="Définition du plan directeur"
url="http://.../OutlinePlanDefinitionPortlet.wsdl"/>
          </ml:DecisionPoint>
          <ml:DecisionPoint id="D8_1" min="0" max="1">
            <ml:RepeatUntil id="D9_1">
              <ml:AtomicMethodService id="C10_1" name="Affinement des besoins"
url="http://.../RequirementsRefinementPortlet.wsdl"/>
              <ml:AtomicMethodService id="C11_1" name="Définition du planning de développement"
url="http://.../DevelopmentPlanDefinition"/>
            </ml:RepeatUntil>
          </ml:DecisionPoint>
        </ml:Sequence>
      </ml:DecisionPoint>
    </ml:Sequence>
  </ml:DecisionPoint>
  <ml:Sequence id="Seq2_2">
    <ml:RepeatUntil id="D5_2">
      <ml:DecisionPoint id="D3_2" min="1" max="1">
        <ml:AtomicMethodService id="C4_2" name="Définition de la liste des besoins du
produit"
url="http://.../BacklogDefinitionPortlet.wsdl"/>
        <ml:AtomicMethodService id="C5_2" name="Définition de besoin utilisateur"
url="http://.../UserStoriesDefinitionPortlet.wsdl"/>
      </ml:DecisionPoint>
    </ml:RepeatUntil>
  </ml:Sequence>
</ml:MethodLine>
```

```

    <ml:AtomicMethodService id="C6_2" name="Définition des besoins de haut niveau"
        url="http://.../HighLevelRequirementsDefinitionPortlet.wsdl"/>
</ml:DecisionPoint>
</ml:RepeatUntil>
<ml:DecisionPoint id="D6_2" min="1" max="1">
<ml:AtomicMethodService id="C3_3" name="Abandon du projet"
    url="http://.../ProjectCloseOutPortlet.wsdl"/>
<ml:Sequence id="Seq3_2">
    <ml:DecisionPoint id="D7_2" min="1" max="1">
        <ml:AtomicMethodService id="C7_2" name="Planification générale des livrables"
            url="http://.../ReleasePlanningPortlet.wsdl"/>
        <ml:AtomicMethodService id="C8_2" name="Définition du plan général des livrables"
            url="http://.../ReleasePlanDefinitionPortlet.wsdl"/>
        <ml:AtomicMethodService id="C9_2" name="Définition du plan directeur"
            url="http://.../OutlinePlanDefinitionPortlet.wsdl"/>
    </ml:DecisionPoint>
    <ml:DecisionPoint id="D8_2" min="0" max="1">
        <ml:RepeatUntil id="D9_2">
            <ml:AtomicMethodService id="C10_2" name="Affinement des besoins"
                url="http://.../RequirementsRefinementPortlet.wsdl"/>
            <ml:AtomicMethodService id="C11_2" name="Définition du planning de développement"
                url="http://.../DevelopmentPlanDefinition"/>
        </ml:RepeatUntil>
    </ml:DecisionPoint>
</ml:Sequence>
</ml:DecisionPoint>
</ml:Sequence>
</ml:DecisionPoint>
</ml:MethodLine>

```

Annexe L

Modèle de transformation des entrées/sorties de la ligne de méthode de lancement d'un projet agile

(chapitre 6 section 6.6)

```
<?xml version="1.0" encoding="UTF-8"?>
<am:AssignmentModel xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:am='http://crinfo.univ-paris1.fr/schema/AssignmentModel'
xsi:schemaLocation='http://crinfo.univ-paris1.fr/schema/AssignmentModel AssignmentModel.xsd
http://www.w3.org/1999/XMLSchema http://www.w3.org/1999/XMLSchema'
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:mlpm="http://crinfo.univ-paris1.fr/schema/ML-PM-LancementProjetAgile"
xmlns:c1="http://crinfo.univ-paris1.fr/schema/BusinessStudy"
xmlns:c2="http://crinfo.univ-paris1.fr/schema/TechnicalStudy"
xmlns:c3="http://crinfo.univ-paris1.fr/schema/ProjectCloseOut"
xmlns:c4="http://crinfo.univ-paris1.fr/schema/BacklogDefinition"
xmlns:c5="http://crinfo.univ-paris1.fr/schema/UserStoriesDefinition"
xmlns:c6="http://crinfo.univ-paris1.fr/schema/HighLevelRequirementsDefinition"
xmlns:c7="http://crinfo.univ-paris1.fr/schema/ReleasePlanning"
xmlns:c8="http://crinfo.univ-paris1.fr/schema/ReleasePlanDefinition"
xmlns:c9="http://crinfo.univ-paris1.fr/schema/OutlinePlanDefinition"
xmlns:c10="http://crinfo.univ-paris1.fr/schema/RequirementsRefinement"
xmlns:c11="http://crinfo.univ-paris1.fr/schema/DevelopmentPlanDefinition"
>
<am:ProductModels>
  <xsd:schema targetNamespace="http://crinfo.univ-paris1.fr/schema/AM-
LancementProjetAgile.xsd">
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/ML-PM-LancementProjetAgile"
schemaLocation="http://crinfo.univ-paris1.fr/schema/ML-PM-LancementProjetAgile.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/BusinessStudy"
schemaLocation="http://crinfo.univ-paris1.fr/schema/BusinessStudy.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/TechnicalStudy"
schemaLocation="http://crinfo.univ-paris1.fr/schema/TechnicalStudy.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/ProjectCloseOut"
schemaLocation="http://crinfo.univ-paris1.fr/schema/ProjectCloseOut.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/BacklogDefinition"
schemaLocation="http://crinfo.univ-paris1.fr/schema/BacklogDefinition.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/UserStoriesDefinition"
schemaLocation="http://crinfo.univ-paris1.fr/schema/UserStoriesDefinition.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/HighLevelRequirementsDefinition"
schemaLocation="http://crinfo.univ-paris1.fr/schema/HighLevelRequirementsDefinition.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/ReleasePlanning"
schemaLocation="http://crinfo.univ-paris1.fr/schema/ReleasePlanning.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/ReleasePlanDefinition"
schemaLocation="http://crinfo.univ-paris1.fr/schema/ReleasePlanDefinition.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/OutlinePlanDefinition"
schemaLocation="http://crinfo.univ-paris1.fr/schema/OutlinePlanDefinition.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/RequirementsRefinement"
schemaLocation="http://crinfo.univ-paris1.fr/schema/RequirementsRefinement.xsd"/>
    <xsd:import namespace="http://crinfo.univ-paris1.fr/schema/DevelopmentPlanDefinition"
schemaLocation="http://crinfo.univ-paris1.fr/schema/DevelopmentPlanDefinition.xsd"/>
  </xsd:schema>
</am:ProductModels>

<am:MethodLine name="Lancement d'un projet agile">
  <am:Input/>
  <am:Output>
    <am:Transform source="mlpm:ModeleDeProduit" target="mlpm:ModeleDeProduit"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Output>
</am:MethodLine>

<am:Assign parentId="D2" leafId="C1">
  <am:Input/>
  <am:Output>
```

```

    <am:Transform source="c1:DocumentEtudeDeMarche" target="mlpm:DocumentEtudeDeFaisabilite"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/insert.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D2" leafId="C2">
  <am:Input/>
  <am:Output>
    <am:Transform source="c2:DocumentEtudeTechnique" target="mlpm:DocumentEtudeDeFaisabilite"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/insert.xslt"/>
    <am:Transform source="c2:Acteur" target="mlpm:Acteur" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/insert.xslt"/>
    <am:Transform source="c2:Role" target="mlpm:Role" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/insert.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D4" leafId="C3_1">
  <am:Input>
    <am:Transform source="mlpm:DocumentEtudeDeFaisabilite"
target="c3:DocumentEtudeDeFaisabilite" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c3:DocumentEtudeDeFaisabilite"
target="mlpm:DocumentEtudeDeFaisabilite" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D6_1" leafId="C3_2">
  <am:Input>
    <am:Transform source="mlpm:DocumentEtudeDeFaisabilite"
target="c3:DocumentEtudeDeFaisabilite" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/copyAll.xslt"/>
    <am:Transform source="mlpm:ListeDesBesoins" target="c3:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c3:DocumentEtudeDeFaisabilite"
target="mlpm:DocumentEtudeDeFaisabilite" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D6_2" leafId="C3_3">
  <am:Input>
    <am:Transform source="mlpm:DocumentEtudeDeFaisabilite"
target="c3:DocumentEtudeDeFaisabilite" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/copyAll.xslt"/>
    <am:Transform source="mlpm:ListeDesBesoins" target="c3:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c3:DocumentEtudeDeFaisabilite"
target="mlpm:DocumentEtudeDeFaisabilite" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D3_1" leafId="C4_1">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c4:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c4:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D3_2" leafId="C4_2">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c4:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>

```

```

    <am:Transform source="c4:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D3_1" leafId="C5_1">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c5:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c5:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D3_2" leafId="C5_2">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c5:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c5:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D3_1" leafId="C6_1">
  <am:Input>
    <am:Transform source="mlpm:DocumentEtudeDeMarche" target="c6:DocumentEtudeDeMarche"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
    <am:Transform source="mlpm:ListeDesBesoins" target="c6:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c6:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D3_2" leafId="C6_2">
  <am:Input>
    <am:Transform source="mlpm:DocumentEtudeDeMarche" target="c6:DocumentEtudeDeMarche"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
    <am:Transform source="mlpm:ListeDesBesoins" target="c6:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c6:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D7_1" leafId="C7_1">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c7:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c7:PlanningDesLivrables" target="mlpm:PlanningDesLivrables"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D7_2" leafId="C7_2">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c7:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c7:PlanningDesLivrables" target="mlpm:PlanningDesLivrables"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D7_1" leafId="C8_1">
  <am:Input>

```

```

    <am:Transform source="mlpm:ListeDesBesoins" target="c8:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c8:PlanningDesLivrables" target="mlpm:PlanningDesLivrables"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D7_2" leafId="C8_2">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c8:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c8:PlanningDesLivrables" target="mlpm:PlanningDesLivrables"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D7_1" leafId="C9_1">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c9:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c9:PlanningDesLivrables" target="mlpm:PlanningDesLivrables"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D7_2" leafId="C9_2">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c9:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c9:PlanningDesLivrables" target="mlpm:PlanningDesLivrables"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D9_1" leafId="C10_1">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c10:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c10:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D9_2" leafId="C10_2">
  <am:Input>
    <am:Transform source="mlpm:ListeDesBesoins" target="c10:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c10:ListeDesBesoins" target="mlpm:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
  </am:Output>
</am:Assign>

<am:Assign parentId="D9_1" leafId="C11_1">
  <am:Input>
    <am:Transform source="mlpm:Livable" target="c11:Livable" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/copyAll.xslt"/>
    <am:Transform source="mlpm:Acteur" target="c11:Acteur" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/copyAll.xslt"/>
    <am:Transform source="mlpm:ListeDesBesoins" target="c11:ListeDesBesoins"
xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
  </am:Input>
  <am:Output>
    <am:Transform source="c11:Livable" target="mlpm:Livable" xsltRef="http://crinfo.univ-
paris1.fr/schema/xslt/replaceAll.xslt"/>

```

```

    <am:Transform source="c11:Developpe" target="mlpm:Developpe" xsltRef="http://crinfo.univ-
    paris1.fr/schema/xslt/replaceAll.xslt"/>
    <am:Transform source="c11:Acteur" target="mlpm:Acteur" xsltRef="http://crinfo.univ-
    paris1.fr/schema/xslt/replaceAll.xslt"/>
    <am:Transform source="c11:Participant" target="mlpm:Participant"
    xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
    </am:Output>
</am:Assign>

<am:Assign parentId="D9_2" leafId="C11_2">
  <am:Input>
    <am:Transform source="mlpm:Livable" target="c11:Livable" xsltRef="http://crinfo.univ-
    paris1.fr/schema/xslt/copyAll.xslt"/>
    <am:Transform source="mlpm:Acteur" target="c11:Acteur" xsltRef="http://crinfo.univ-
    paris1.fr/schema/xslt/copyAll.xslt"/>
    <am:Transform source="mlpm:ListeDesBesoins" target="c11:ListeDesBesoins"
    xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/copyAll.xslt"/>
    </am:Input>
    <am:Output>
      <am:Transform source="c11:Livable" target="mlpm:Livable" xsltRef="http://crinfo.univ-
      paris1.fr/schema/xslt/replaceAll.xslt"/>
      <am:Transform source="c11:Developpe" target="mlpm:Developpe" xsltRef="http://crinfo.univ-
      paris1.fr/schema/xslt/replaceAll.xslt"/>
      <am:Transform source="c11:Acteur" target="mlpm:Acteur" xsltRef="http://crinfo.univ-
      paris1.fr/schema/xslt/replaceAll.xslt"/>
      <am:Transform source="c11:Participant" target="mlpm:Participant"
      xsltRef="http://crinfo.univ-paris1.fr/schema/xslt/replaceAll.xslt"/>
      </am:Output>
    </am:Assign>

</am:AssignmentModel>

```