ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THÈSE PRÉSENTÉE À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DU DIPLÔME DE
DOCTORAT EN GÉNIE
Ph.D.

PAR
Anis MASMOUDI

UNE MÉTHODE D'AGRÉGATION DE COMPOSANTS LOGICIELS DIRIGÉE PAR LES MÉTADONNÉES ET LES MODÈLES

MONTRÉAL, LE 2 MARS 2012

PRÉSENTATION DU JURY

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE :

M. Roger Champagne, directeur de thèse
Département génie logiciel et des TI à l'École de technologie supérieure.

M. Gilbert Paquette, codirecteur de thèse
Chaire de recherche CICE/Centre de recherche LICEF/ Télé-Université du Québec.

M. Claude Thiebault, président du jury
Département de génie électrique à l'École de technologie supérieure.

M. Hafedh Mili, membre du jury
Département d'informatique de l'Université du Québec À Montréal

Mme. Ghizlane El Boussaidi, membre du jury
Département génie logiciel et des TI à l'École de technologie supérieure.

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 8 DÉCEMBRE 2011

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

AVANT-PROPOS

Nos travaux de recherche sur l'agrégation des composants logiciels et leur réutilisation ont débuté dans le contexte des environnements et des systèmes d'apprentissage à distance dans le centre de recherche LICEF de la UQAM-Téluq. Nous avons proposé, en 2003, dans le cadre d'une maitrise ès sciences en technologie de l'information, un premier processus d'intégration de composants dirigé par les modèles UML du système d'enseignement à distance, Explor@2[1].

Ensuite, les travaux de la présente thèse sur la méthode d'agrégation de composants logiciels dirigée par les métadonnées et les modèles ont débuté en 2005 avec le projet LORNET, qui avait pour objectif principal, la mise en place d'un système générateur de systèmes d'apprentissage et, de façon plus générale, de systèmes de gestion des connaissances. Ces systèmes, à leur tour, génèreront des applications spécifiques conformément à une discipline scientifique ou à un domaine de travail. Ces applications sont accessibles par les tuteurs (chargés de cours) pour assister les apprenants lors du suivi d'un cours à distance ou lors de la participation à un processus de travail. Chaque apprenant utilise des ressources pédagogiques ou de support au travail et en fournit d'autres. Cette cascade, supportée par le système TELOS (*Tele-Learning Operating System*) du projet LORNET, est conçue par agrégation de composants logiciels et de ressources existantes. Ce processus et ses fonctionnalités sont proposés au système TELOS pour promouvoir le développement par la réutilisation de composants.

Entre 2007 et 2010, d'autres versions du processus de la méthode ont été réalisées suite à des exécutions récurrentes dans les contextes réels de construction de systèmes d'information et de portails d'entreprises publiques et de télécommunication tels que le portail de la Ville de Montréal et le portail Intranet des clients de TELUS.

---

[1] Explor@2: Un système de support à la formation Web de troisième génération, accessible à l'adresse http://explora.teluq.uquebec.ca/2.0/fr/index.htm

Dans ces cas industriels concrets, le processus de la méthode a été exécuté avec les systèmes de missions tels que la gestion des billetteries, le suivi du trafic réseau, la gestion documentaire, la gestion de sécurité, les portails d'information Internet et d'entreprise Intranet ainsi que le portail des opérations en télécommunication.

REMERCIEMENTS

Mes remerciements les plus profonds s'adressent à mon directeur et mon codirecteur qui m'ont encadré tout au long de ces années d'étude: M. Roger Champagne et M. Gilbert Paquette. Au travers de nos discussions, ils m'ont apporté une aide continue et une compréhension plus approfondie des divers aspects de la problématique de recherche. Je salue aussi, leur savoir-faire, leur souplesse et leur ouverture d'esprit qui ont su me laisser une large marge de liberté pour mener à bien ce travail de recherche en parallèle avec mon travail. Je tiens à exprimer ma reconnaissance à M. Roger Champagne pour son aide, son support financier, sa patience et son suivi pour toute la période de mes travaux de recherche.

Je veux réitérer mes remerciement à mon professeur M. Gilbert Paquette de m'avoir encouragé à me lancer dans le monde de recherche doctorale, en 2004, dans le cadre du projet LORNET. Il m'a supporté pédagogiquement et financièrement tout au long de mes études.

Je tiens à exprimer ma gratitude à M. Claude Thiebault, qui m'a fait l'honneur de présider le jury de thèse de doctorat, et aux membres du jury pour l'intérêt présenté envers mon sujet de recherche.

Je dédie cette thèse à la mémoire de M. Eric Lefebvre, qui était un des membres du jury, décédé cette année. Je tiens aussi à présenter toutes mes condoléances à sa famille.

Je veux remercier la compagnie CGI Inc. qui a supporté techniquement et financièrement mes études depuis 2005. Elle m'a permis d'expérimenter mes résultats de recherche durant mes mandats chez ses clients. Je remercie particulièrement le soutien de mes directeurs à CGI : M. Fazil Chouakri et M. Jalil ElMejjad.

Je dédie cette thèse à la mémoire de ma mère Mme Latifa Baklouti qui est morte le 20 Août 2006, un mois avant mon examen doctoral. Elle m'a accompagné durant toutes mes années

d’étude depuis le début. Elle a tant attendu ce moment. Elle a voulu le partager et le célébrer avec moi. Malheureusement, elle est absente aujourd’hui de ce monde, mais elle demeure toujours présente dans mon cœur, ma pensée et mon monde.

Je remercie mon père M. Hédi Masmoudi, qui m’a toujours supporté durant toutes mes années d’études. Il m’a appris la persévérance, la patience et de ne jamais baisser les bras dans les situations les plus difficiles.

Je remercie mon épouse Mme Hanen Jemai qui m’a encouragé et m’a supporté durant mes travaux de recherche, un vrai amour. Elle a sacrifié beaucoup de son temps pour que je puisse poursuivre mes études. Aussi je remercie la patience de mes filles Dyna et Tasnim vu mon absence de la maison durant la rédaction de ma thèse.

Je remercie ma sœur Sondes et mes frères Issam et Hamouda qui n’ont jamais cessé de me soutenir et de m’aider durant mes études depuis mon jeune âge.

UNE MÉTHODE D'AGRÉGATION DE COMPOSANTS LOGICIELS DIRIGÉE PAR LES MÉTADONNÉES ET LES MODÈLES

Anis Masmoudi

RÉSUMÉ

Le composant logiciel est devenu l'unité de base pour le développement des systèmes complexes. Plusieurs travaux ont contribué à la construction de logiciel à base de composants et leur modèle. Cependant, il existe peu d'approches et d'heuristiques basées sur les métadonnées qui assistent des architectes pour référencer, rechercher et évaluer les composants participants dans une agrégation avant la phase d'implémentation.

Dans cette thèse, nous proposons une méthode d'agrégation de composants logiciels dirigée par les métadonnées des composants et les modèles de leur agrégat. Cette méthode caractérise le composant par des métadonnées multidimensionnelles. Elle assiste les parties prenantes dans le référencement et la recherche des composants. Elle propose une phase inédite d'analyse de faisabilité qui s'appuie sur un processus et des règles d'évaluation de la faisabilité d'agrégation dirigée par les métadonnées des composants participants. Nous avons défini un cadre de travail qui a servi de support à notre méthode durant des mises en œuvre dans des contextes d'agrégation réels. Ces réalisations concrètes nous ont permis de valider l'utilisabilité et l'utilité de notre méthode en nous basant sur notre processus de validation et sur ses indicateurs de mesure.

Dans le futur, nous préconisons que les activités de référencement, de recherche et d'évaluation de la faisabilité de notre processus supportent l'automatisation afin de systématiser en partie son utilisation.

La prise en compte des paramètres financiers aux activités de notre processus de développement aidera les gestionnaires dans l'estimation des efforts des projets d'agrégation et par conséquent dans la prise de décision quant à leur faisabilité financière. Ceci enrichira

le manque des aspects financiers et économiques dans les approches de développement à base de composants (Mili, Mili, Yacoub, & Addy, 2002).

Mots clés : composant logiciel, agrégation, modèle, métadonnées, faisabilité.

UNE MÉTHODE D'AGRÉGATION DE COMPOSANTS LOGICIELS DIRIGÉE PAR LES MÉTADONNÉES ET LES MODÈLES

Anis Masmoudi

ABSTRACT

Many techniques in software development, based on software components, can be used to build different kinds of systems. Recently, software components have been used to form the technical basis of computer systems and applications. Many studies suggest modeling approaches, processes and technics to develop software component based system. However, few of them use metadata concept to reference, search within a repository and evaluate their aggregation before implementation stage.

Our research focuses on CBSD (Component-Based Software Development) in general and especially on component aggregation methods and processes. Our work aims to add to software components multidimensional metadata, which are then used to reference, categorize and search components. Also, we use our component metadata specifications and a set of assessment rules to assess the feasibility of the aggregation of components. Our method suggests an aggregation process based on component metadata and models. This process defines an original sub-process that evaluates components' aggregation feasibility before component adaptation and integration phases. All these processes, specifications and rules are supported by our aggregation framework that helps architects to execute our processes' activities. We validated our proposition in a real industrial context in order to verify and validate its usability and utility.

In the future, we plan to implement some activities of our process with business process management tools, such as component referencing, search and assessment, to promote reuse in organizational context. Our component specification can be enriched by financial and management metadata that help project managers estimate the effort required in component aggregation projects. By adding financial and management metadata, we can help project

teams to assess financial project feasibility. These kinds of metadata will improve the consideration of financial and economic aspects as suggested by (Mili, Mili, Yacoub, & Addy, 2002).

Keywords: software component, aggregation, model, metadata, feasibility.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

CBD	Component based development
CORBA	Common object request broker architecture
COM	Common object model
COTS	Commercial-off-the-shelf
CL	Composant logiciel
DCOM	Distributed COM
EJB	Enterprise Java Bean
EAI	Enterprise Application Integration
MDA	Model Driven Architecture
MDD	Model Driven Development
.NET	Microsoft Framework for Connecting Software Component
OWL	Ontology Web Language
PL-SQL	Procedural Language for SQL
SCA	Service Component Architecture
SOA	Service Oriented Architecture
SOCOM	SOftware COmponent Metadata
SOCAF	SOftware Component Aggregation Framework
SOCAP	SOftware Component Aggregation Process
SOCAM	SOftware Component Aggregation Metadata
SQL	Structured Query Language
JEE	Java Enterprise Edition

RDF	Resource Description Framework
RMI	Remote Method Invocation
UML	Unified Modeling Language
WS	Web Services
XML	eXtended Markup Language

INTRODUCTION

Les composants logiciels sont l'unité de base technique de certains systèmes informatiques. De nos jours, plusieurs organisations fournissent des composants pour répondre à un besoin d'affaires précis dans un domaine particulier (Dowling, 2000). L'approche « développement des systèmes par agrégation de composants logiciels » est une vision récente du développement logiciel (Seidewitz, 2003). Plusieurs compagnies évitent le développement classique. Elles étudient activement le développement logiciel avec et pour la ***réutilisation*** (Mili, Mili, Yacoub, & Addy, 2002). Elles utilisent souvent les blocs de code existants sous la forme de composants et s'intéressent de plus en plus à la réalisation du code logiciel dans un format bien structuré, à savoir le composant. Elles développent les solutions logicielles en se basant sur l'agrégation de composants. Parfois, elles achètent des composants du marché et les intègrent dans leurs solutions d'affaires pour enrichir leur portefeuille technologique.

Nous nous sommes intéressés, dans cette thèse, aux problèmes de la réutilisation des composants logiciels et au développement du logiciel basé sur les composants en général et, particulièrement, au problème d'agrégation des composants en utilisant les métadonnées et les modèles.

Les métadonnées et les modèles associés aux composants logiciels favorisent leur réutilisation et facilitent leur analyse et l'étude de leur agrégation (Mili, Mili, Yacoub, & Addy, 2002). Conséquemment, il est indispensable d'identifier ces métadonnées et ces modèles et de montrer leur utilité dans les phases du processus de développement à base de composants : la recherche, la sélection, l'adaptation et l'intégration. Nous prenons comme hypothèse que ces concepts sont aussi importants dans la qualification des composants que dans l'évaluation de la faisabilité de leur agrégation au début du processus de développement.

Les travaux de la thèse visent principalement quatre objectifs :

- Identifier et analyser le problème de représentation abstraite et de référencement des composants pour ainsi proposer un ***jeu de métadonnées*** qui décrivent les composants selon plusieurs points de vue et perspectives. Ces métadonnées devraient aider à la classification des composants logiciels et à leur agrégation.

- Identifier et spécifier un ***processus d'évaluation de la faisabilité*** de l'agrégation des composants logiciels en se basant sur les métadonnées et des règles de compatibilité logicielles afin d'aider des équipes d'affaires et techniques durant le développement d'un nouveau composant par agrégation.

- Élaborer un ***processus d'agrégation*** en utilisant le processus d'évaluation, les classifications, les métadonnées et les modèles des composants et des agrégations. Ce processus est validé par des réalisations d'agrégations dans des contextes réels, à l'aide d'un modèle de validation approprié.

- Définir un ***cadre conceptuel*** de base pour l'agrégation des composants. Ce cadre est composé d'entités fondamentales, constituées d'un ensemble d'outils, de spécifications, de protocoles et des technique. Le tout sera utilisé pour supporter et appuyer l'utilisation du processus d'agrégation de composants logiciels dirigée par les métadonnées et les modèles.

La méthodologie de cette thèse est basée essentiellement sur deux volets : théorique et pratique. Ces deux volets sont souvent traités en parallèle avec des synchronisations mutuelles. Autrement dit, nos travaux de recherche comportent des aspects théoriques qui sont validés par des réalisations pratiques. Les deux volets sont dépendants et se synchronisent afin d'atteindre les objectifs escomptés. En effet, nous avons effectué les travaux de recherche théoriques dans les domaines identifiés autour des concepts de la problématique de recherche. Ensuite, le volet pratique est un moyen de vérification et de validation des résultats théoriques obtenus. Nous avons recours à des réalisations techniques par des développements logiciels ou par des publications selon l'aspect étudié. Dès que nous

validons un ou plusieurs résultats théoriques, nous procédons à une synchronisation avec la problématique globale en intégrant les résultats partiels valides dans la solution globale, ce qui nous permet d'avancer dans la résolution des problèmes identifiés et nous permet d'atteindre notre but ultime : la proposition d'une méthode d'agrégation des composants logiciels dirigée par les métadonnées et les modèles.

En effet, le volet théorique concerne des recherches autour des concepts connexes au composant logiciel tels que les paradigmes de métadonnées, des modèles, des agrégations, des approches d'agrégation, les processus et la faisabilité d'agrégation. Le volet pratique concerne les développements logiciels, les publications, les revues par les pairs, les expérimentations des aspects théoriques dans des contextes réels. Par exemple, après la définition d'un jeu de métadonnées qui décrit les composants logiciels, nous avons implémenté ce jeu à l'aide d'une application Web afin de vérifier leur utilité au moment de la recherche, la classification et de la découverte des composants logiciels. Aussi, la définition d'un processus de faisabilité d'agrégation est une contribution originale de notre thèse, qui a été utilisé à l'aide d'un outil approprié dans l'un de nos projets durant des mandats chez les clients de CGI.

Ce parallélisme nous permet de valider incrémentalement nos résultats de recherche. Cette approche mixte entre la théorie et la pratique donne plus de crédibilité à nos résultats et encourage leur réutilisation dans le futur par d'autres intervenants qui sauront comment faire à travers la documentation de nos expérimentations. De plus, nous proposons une méthode avec des indicateurs qui permet la vérification et la validation du processus qui regroupe l'ensemble des résultats de recherche de la thèse, ce qui confirme les validations partielles et justifie l'apport de la solution globale proposée.

Il est important de rappeler que notre méthodologie utilise l'itération et l'incrémentation comme deux principes de base dans les volets théorique et pratique. De plus, les publications dans les conférences, les revues et les ateliers de recherche valident nos résultats conceptuels et les produits développés. Ces publications nous ont permis de participer activement avec les

communautés de recherche des domaines identifiés afin de collaborer avec l'équipe de recherche scientifique mondiale, ce qui a permis de positionner notre recherche et nos contributions.

La réutilisation étant au cœur de nos préoccupation de recherche, nous avons réutilisé principalement les principes de la méthodologie RUP[2] (itération, incrémentation, centrée sur les cas d'utilisation, centrée sur l'architecture), le parallélisme de la méthodologie en Y 2TUP[3] (deux flux parallèles entre l'analyse des besoins et la conception de la solution) et les bonnes pratiques des méthodologies agiles[4] (partage, collaboration, rencontre de groupe, discussion, etc.).

Pour atteindre les objectifs de recherche, nous proposons une démarche composée de neuf étapes (Voir Figure 1). La première étape vise l'identification et de définition de la problématique de recherche. Ces aspects sont traités au début du présent chapitre. Des thèmes et des références bibliothécaires ont été identifiés pour réaliser la revue de la littérature de l'étape 2.

Cette revue rappelle la définition des concepts connexes à la problématique et est essentiellement réalisée autour des composants logiciels, les métadonnées, les modèles et les processus d'agrégation. Durant cette étape, nous rappelons les approches existantes autour de l'agrégation et les défis dans la littérature qui sont étudiés dans cette thèse.

Le premier objectif concernant la représentation abstraite des composants est élaboré dans la troisième étape appelée « Métadonnées des composants ». Pour définir cette représentation, des métadonnées, des spécifications, des formalismes, des langages de développement et de déploiement ont été analysés de façon itérative et incrémentale, ce qui a permis la définition

---

[2] RUP – Rational Unified Process disponible via l'adresse Web http://www-01.ibm.com/software/awdtools/rup/
[3] 2TUP – Two Track Unified Process disponible via l'adresse Web : http://www.e-bancel.com/Processus_2TUP.php
[4] Les 12 principes Agile disponible via l'adresse Web : http://agilemanifesto.org/principles.html

des différentes catégories de métadonnées d'un composant logiciel. Pour valider le jeu de métadonnées, nous avons effectué des recherches de composants dans un entrepôt de composants que nous avons créé.

La quatrième étape, intitulée « Classification des agrégations des composants », décrit les travaux d'identification et de spécification des catégories de métadonnées d'agrégations des composants logiciels. Nous définissons dans cette étape des équations ensemblistes à partir d'un sous ensemble du jeu de métadonnées proposé. Ces équations définissent des classifications des composants afin de faciliter leur recherche et leur agrégation. Pour valider cette catégorisation, nous avons identifié des agrégations concrètes et nous avons appliqué les équations définies.

La cinquième étape, « Processus d'agrégation », décrit en détail le processus global d'agrégation des composants dirigé par les métadonnées et les modèles. À la sixième étape nommée « Faisabilité d'agrégation », nous avons développé un sous-processus d'évaluation de la faisabilité d'agrégation basé sur le jeu de métadonnées, les classifications des agrégations et les modèles associés déjà définis. Pour faciliter l'utilisation et la pratique de ce processus, nous proposons, dans la septième étape, un cadre de travail englobant ces processus, des règles, des outils, des techniques et des technologies. Il s'agit d'une étape de « Support au processus d'agrégation ».

La huitième étape, « Vérification & Validation par l'exécution », montre l'utilisation du processus d'agrégation, du processus de faisabilité et du cadre de travail dans des contextes réels, ce qui nous a permis – comme le nom de l'étape l'indique – de vérifier et de valider l'utilité et l'apport des solutions proposées aux domaines étudiés et surtout à celui de développement par la réutilisation.

À la fin, nous exposons une discussion de nos résultats, la conclusion, les perspectives et les recommandations sur les domaines de recherche étudiés. Pour faciliter la lecture de la thèse, nous avons consacré un chapitre pour chacune des étapes de la méthodologie. La figure

suivante montre sommairement la chaine des neuf étapes de notre démarche de recherche et de la structure de la thèse.

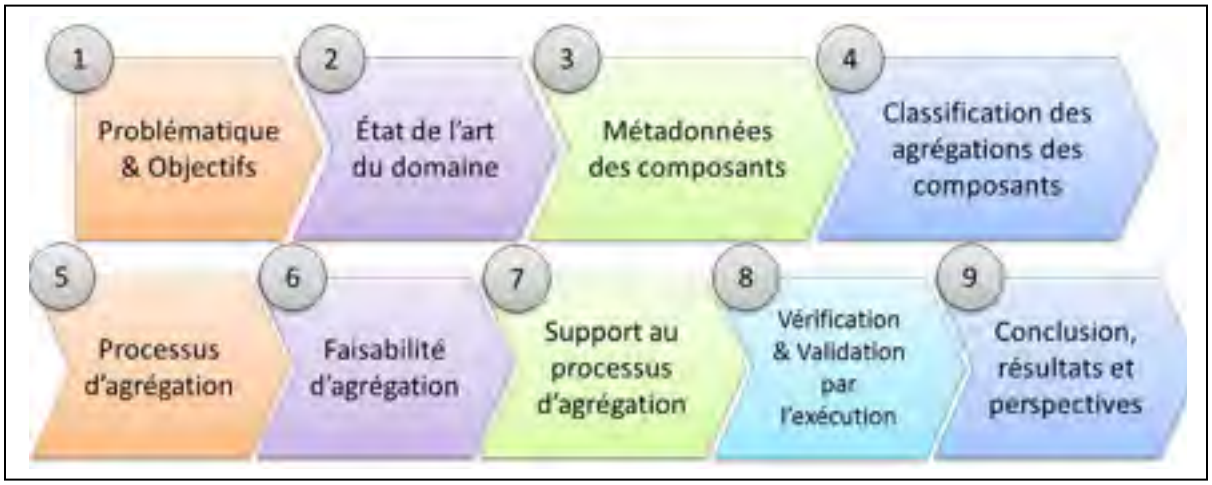


Figure 1 Les neuf étapes de la méthodologie de recherche de la thèse

Nos travaux de recherche croisent essentiellement les axes de recherche du développement à base de composants et le développement logiciel avec et pour la réutilisation. Dans le chapitre suivant, les concepts autour de ces axes sont rappelés et les défis qui s'y rattachent sont identifiés en rapport avec la problématique et nos objectifs de recherche.

CHAPITRE 1

ÉTAT DE L'ART DU DOMAINE

Ce chapitre a pour objectif de résumer la littérature relative aux concepts et aux idées présentées dans l'introduction. La section 1.1 explique brièvement l'origine du paradigme objet et ses caractéristiques. Ce paradigme a montré des limitations quant au développement par la réutilisation, ainsi le composant est proposé comme l'unité de base des systèmes complexes. Dans la section 1.2, on décrit le paradigme composant, ses différentes catégories, son utilisation et les différents types d'agrégations auxquelles il peut participer. Les métadonnées associées au composant lui offrent une visibilité et facilitent sa recherche dans les entreprises de développement logiciel. Ce concept est étudié dans la section 1.3 avec la classification et l'ontologie. Dans la section 1.4, on décrit le concept de la modélisation et son apport pour la représentation abstraite des composants logiciels. Ce dernier favorise la réutilisation au niveau conceptuel et devient pratique lorsqu'il intègre les étapes d'un processus de développement. La section 1.5 présente de manière générale l'ingénierie de réutilisation du logiciel. La section 1.6 traite particulièrement le volet de l'ingénierie du logiciel par la réutilisation des composants ainsi que les processus de développement des systèmes à base de composant (section 1.7). En théorie, les communications entre les composants sont traduites sous la forme de contrats de composants. Ce concept est défini et détaillé dans la section 1.8. En pratique, un contrat de composants s'implémente à l'aide des techniques et des technologies d'intégration qui sont décrites dans la section 1.9. Enfin, la conclusion de la section 1.10 résume les concepts élaborés et introduit les chapitres de développement de la thèse.

L'état de l'art est la deuxième étape de la démarche de notre méthodologie de recherche. La figure suivante positionne le présent chapitre dans le cheminement des étapes de la méthodologie de recherche.

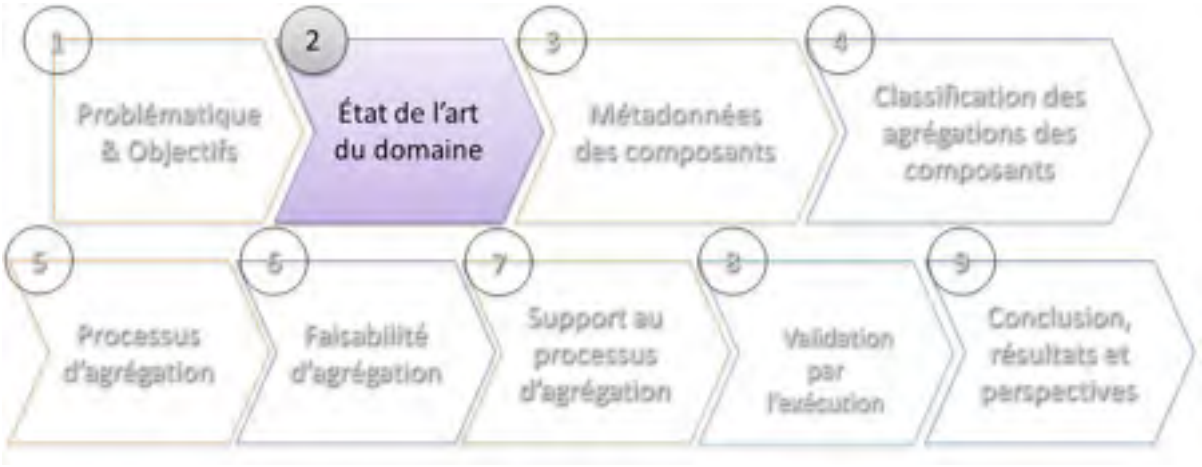


Figure 1.1 La méthodologie de recherche de la thèse – Étape 2 :
État de l'art du domaine

1.1 Le paradigme Objet

Introduit dans les années 60 dans les langages de simulation (Simula, 67), le paradigme objet était utilisé dans le domaine de l'intelligence artificielle dans les années 80. Son apport fondamental consiste à combiner au sein d'une même structure de données, appelée classe, les opérations et les données. L'idée centrale derrière cette alternative est de monter une modélisation fidèle du monde réel. Les analystes et les concepteurs de systèmes informatiques visent à analyser et à concevoir des objets en transposant ces derniers de la problématique du monde réel en objets ayant un ensemble d'attributs et de responsabilités qui définissent leur comportement. De ce fait, les langages de programmation ont suivi ce paradigme, comme en témoigne l'effervescence des langages de programmation orientés objet comme Ada, Smalltalk, Java, C++, C#, etc. Cette vague continue son émergence et son influence sur beaucoup de domaines et de disciplines (Szyperski, Gruntz, & Murer, 2002) .

Ce paradigme objet est connu par les concepts majeurs tels l'abstraction, l'encapsulation, la modularité, le polymorphisme et l'héritage (concepts introduits par le langage SmallTalk (Coad & Yourdon, 1990)). Les objets ne peuvent pas être réutilisés à grande échelle. Ils sont utiles pour le développement d'applications, mais présentent des limitations quand il s'agit

d'intégration entre des systèmes d'informations ou des applications (Szyperski, Gruntz, & Murer, 2002). Les composants, souvent de taille plus grande que les objets, sont une solution plus pragmatique dans le développement des systèmes par assemblage ou par composition. Dans une vision "objet", un composant peut être vu comme un agrégat d'objets interconnectés en vue de satisfaire un ensemble de besoins spécifiques dans un contexte particulier.

1.2 Le paradigme Composant : définition, classification et réutilisation

Il existe plusieurs définitions du paradigme « Composant logiciel ». Cette section rappelle les plus utilisées et spécifie, à la fin, celle adoptée pour le reste du document.

Selon (Booch, Rumbaugh, & Jacobson, 1999), un composant est une partie physique et remplaçable d'un système qui est conforme à la réalisation d'un ensemble d'interfaces qu'il fournit. Typiquement, il représente autrement l'empaquetage physique d'éléments logiques, tels que des classes, des interfaces et des collaborations.

Pour (Szyperski, Gruntz, & Murer, 2002), un composant logiciel est une unité de composition reliée à des interfaces particulières et se rattachant à un contexte de dépendances explicite. Un composant logiciel peut être déployé indépendamment et faire l'objet d'une composition par un tiers.

Pour sa part (Larman, 2002) précise qu'un composant est indispensable, presque indépendant. Il présente une partie remplaçable d'un système qui accomplit des fonctionnalités claires dans le contexte d'une architecture bien déterminée. Un composant est conforme à une implémentation physique d'un ensemble d'interfaces.

Dans la littérature, il y a plusieurs catégories d'associations auxquelles un composant peut participer. En effet, un composant peut contribuer à une composition, une coordination (Villalobos, 2003), une intégration (Denno, Steves, Libes, & Barkmeyer, 2003), une

orchestration ou un assemblage (Renaux, Olivier, & Jean-Marc, 2004). Ces catégories prêtent souvent à confusion. Beaucoup les utilisent pour dire la même chose alors que réellement il y a une différence qui découle du contexte de leur utilisation et du type de connexion qui s'établit entre les composants associés. Les paragraphes suivants donnent un aperçu des définitions de ces termes.

Une **association de composants** est une mise en relation de communication entre deux composants. Cette communication met en relation directe les services fournis et ceux requis des deux composants participants. De ce fait, une **agrégation de composants** est une association complexe mettant en relation de composition plus de deux composants selon une logique métier et un contexte particuliers (Ivers *et al.*, 2004).

On entend par **composition**, une manière de structurer et de construire des applications à partir de composants logiciels. Villalobos se base sur les trois types de composants – boîte noire, boîte blanche et boîte grise – pour définir ces trois types de composition (Villalobos, 2003):

- Composition boîte noire : il s'agit d'une intégration du monde d'exécution des composants. On ne connaît pas de composition fonctionnelle de l'ensemble. La composition des besoins non fonctionnels se fait à travers des langages de haut niveau d'abstraction. Techniquement, la composition se fait au niveau des méthodes et non des instructions. On parle ainsi de **connexion de composant**.
- Composition boîte grise : la composition ne se fait pas dans le monde de l'exécution. Les composants sont réutilisables du point de vue services. La composition se fait au niveau des instructions et non au niveau de la méthode. Il s'agit dans ce cas **de coordination de composant**.
- Composition boîte blanche : il n'y a pas de remontée de niveau d'abstraction. L'inclusion des structures non fonctionnelles se fait directement sur les structures du programme. On appelle cette composition "**intégration**".

L'assemblage de composant est une approche similaire à celle vue dans les industries des appareils électroniques et de l'automobile. Elle utilise les trois types de composition cités ci-dessus pour le développement d'applications par assemblage de composants.

Dans le domaine de l'ingénierie logicielle basée sur la **réutilisation**, les nouveaux composants (applications, systèmes, composants autonomes) sont développés en réutilisant des composants logiciels existants. La réutilisation nous permet de profiter des fonctionnalités qui sont déjà programmées, déboguées et maintenues par d'autres (Tuyet, 2004). Pour mettre en pratique la réutilisation des composants dans tout le processus de développement logiciel, plusieurs catégories de composants ont déjà été suggérées :

- **Composants de différents types de connaissances** : les composants peuvent avoir des connaissances de type *produit* ou *processus.* Les composants de type *produit* sont dédiés aux systèmes à l'origine de leur développement tels que les patrons de (Gamma *et al.*, 1995) et les architectures logicielles. Les composants *processus* correspondent plutôt au parcours nécessaire pour réaliser le résultat voulu. Ils sont utiles pour spécifier des fragments de démarches facilitant le développement logiciel tels que les patrons de processus (Ambler, 1998).
- **Composants de différentes couvertures**: il s'agit du degré de généralité des composants. Certains sont *généraux*, d'autres *spécifiques* à un *domaine* particulier ou a une *entreprise* (Mili, Mili, Yacoub, & Addy, 2002).
- **Composants de différentes portées**: la portée concerne la phase du cycle de développement. Il existe des patrons d'analyse, de conception ou encore d'implémentation. Les cadres de travail (frameworks) d'architecture sont par exemple des composants de conception.
- **Composants de différentes natures de solutions:** on distingue deux types de solution, de nature *conceptuelle* ou *logicielle*. Les composants logiciels se présentent sous la forme la plus répandue de composants réutilisables, soit le fragment de code. Ils sont souvent utilisés dans la phase d'implémentation (D'Souza & Wills, 1998) (Heineman *et al.*, 2001). Les composants EJB et CCM sont aussi des exemples de composants logiciels.

Les composants conceptuels sont des petites structures d'analyses ou d'architectures qui permettent de spécifier des logiciels ou des systèmes d'informations.

- **Composants de différents niveaux de complexité :** le niveau de complexité mesure le plus souvent le nombre d'unités qui composent le composant. Elle spécifie le degré de complexité de sa structure interne ou de l'architecture qu'il propose. Elle peut être faible (moins de 10 entités), moyenne (moins de 100 entités) ou forte (plus de 100 entités). L'entité atomique peut être définie par une classe. Par exemple, les patrons de conception du GoF (Gamma *et al.*, 1995), ou ceux d'analyse de Coad (Coad, 1992) offrent des composants de faible complexité (entre deux et six classes).
- **Composants de différentes ouvertures:** le niveau de transparence caractérise l'ouverture du composant. On distingue plusieurs de type de composants selon la visibilité de la structure interne et la modificabilité ou non de leur code.
    - boîte *noire* : la structure interne est invisible et non modifiable;
    - boîte blanche : la structure interne est visible et modifiable;
    - boîte en verre : la structure interne est visible, mais non modifiable;
    - boîte grise : la structure interne est visible et modifiable par paramétrisation.

Dans le cadre de cette thèse, nous traitons les composants logiciels et non conceptuels et nous adoptons la définition de Szyperski d'un composant logiciel. Le terme agrégation englobe la signification de composition pour tout le reste du document. L'utilisation du terme « agrégation », au lieu de « composition », « intégration » ou « coordination », est due à la couverture de ce terme à tous les types d'interactions possibles entre les composants. Il suffit d'ajouter une description de la nature de l'interaction au terme « agrégation » pour ainsi retrouver toutes les définitions des autres termes. En effet, une agrégation par connexion traduit des interactions directes (p. ex. client/serveur) entre deux composants, ce qui est l'équivalent d'une connexion ou une juxtaposition. Une agrégation par coordination est équivalente à série de connexions orchestrées entre les composants. Aussi, une agrégation par fusion est équivalente à une intégration des composants de type boîte blanche. Ces différents types d'agrégation sont détaillés dans le chapitre 3.

Nos travaux utilisent des composants bien formés pour construire différents types d'agrégation afin de promouvoir le développement par la réutilisation selon une méthode qui est proposée comme résultat des travaux de cette thèse.

1.3 Les métadonnées, la classification et les ontologies pour le référencement

Par définition, les métadonnées sont les données qui décrivent des données. On les appelle aussi des informations sur les données ou bien des informations sur des informations. En d'autres termes, les métadonnées sont un ensemble de données qui décrivent des informations sur des ressources. Pour raffiner cette description, on peut dire que les métadonnées sont des ***données structurées*** servant à décrire une ressource (Greenberg, 2002). Ces données structurées impliquent un ordonnancement systématique des données correspondant à une spécification d'un schéma de métadonnées. Cette spécification est une représentation officielle qui précise des règles de structuration des données en question. Pour qu'elles soient interprétables par les machines, ces métadonnées doivent être représentées par un langage standardisé tels que XML. Ceci favorise, via des programmes informatiques spécifiques, l'interopérabilité et la communication entre des applications réalisées avec des technologies hétérogènes. Il est important de préciser que ces métadonnées structurées sont facilement interprétables par les machines à l'aide des programmes spécialisés et ne le sont pas nécessairement par les humains.

Une **classification** est une manière de catégoriser un ensemble de concepts. Cette catégorisation peut être faite sous la forme d'une hiérarchie. Cette dernière ressemble à un arbre, elle possède une racine et des branches. Chaque branche pointe vers un point appelé nœud. La **classification** est aussi appelée taxonomie. En technologie de l'information, une taxonomie est une classification des entités d'information sous la forme de hiérarchie en correspondance avec les relations présumées des entités du monde réel qu'elles représentent (M.C. Daconta, L.J. Obrst, & Smith, 2003).

Par définition, une **ontologie** est un vocabulaire commun que se donnent des usagers ayant besoin de partager des informations dans un domaine. L'ontologie regroupe des définitions lisibles en machine, des concepts de base de ce domaine, ou classes, des relations entre ces concepts et des axiomes énonçant des propriétés de ces classes et de ces relations (M.C. Daconta, L.J. Obrst, & Smith, 2003). Gurber précise qu'une ontologie est une spécification d'une conceptualisation (Gurber, 1993). Ainsi, une ontologie contient plus que des données simples sur les informations ou sur les données, et plus aussi qu'une simple taxonomie.

À partir de ces définitions, on comprend que l'ontologie englobe dans sa structure plus que la structuration des métadonnées, elle englobe aussi leur sémantique. Pour rendre la relation entre les métadonnées et l'ontologie explicite, on peut dire que les ontologies complémentent la notion des métadonnées en encapsulant les données et leur sémantique dans une même structure. Ceci confirme qu'une ontologie est une nouvelle forme de représentation sémantique des métadonnées pour la description enrichie des ressources.

1.4 La modélisation logicielle

Selon Schoderbek *et al.*, un modèle se définit comme suit: "A Model: A simplified representation of something to be made or already existing. Models may be physical, schematic, or mathematical ». (Schoderbek, Schoderbek, & Kefalas, 1985)

Un modèle est une représentation simplifiée, pouvant être graphique, d'un phénomène, d'un concept ou d'un objet du monde réel. La modélisation est une démarche qui produit des modèles. Depuis des siècles, ce concept s'est introduit dans plusieurs domaines du génie: génie civil, génie mécanique, génie électrique, etc. Des concepteurs et des analystes ont réussi à traduire la majorité des fonctionnalités de leur système en un ou plusieurs modèles. Ils utilisent ces modèles pour générer des composants physiques moyennant des outils et des logiciels appropriés. Ces composants entrent en production rapidement suite à des tests bien définis. On cite quelques réalisations dans divers domaines: la conception des ponts, les cartes électroniques, la construction des bâtiments, etc.

En génie logiciel, la modélisation s'est positionnée comme un moyen de représentation abstraite des systèmes informatiques. Ces modèles sont de plus en plus utilisés dans la documentation des logiciels. Cependant, plusieurs d'entre eux se trouvent non conformes ni aux modèles des spécifications des besoins du système, ni aux modèles du code des logiciels construis, d'où la naissance des problématiques de conformité et de connectivité. La première problématique provient de la **conformité** du code à la solution conceptuelle proposée et le deuxième se manifeste dans la **synchronisation** et la **connectivité** du code des systèmes et des modèles associés (Selic, 2003).

Les faiblesses constatées à ce niveau ont poussé la communauté du génie logiciel à adopter des approches dirigées par les modèles. En effet, on parle de nos jours, entre autres, de l'architecture dirigée par les modèles MDA[5] et du développement dirigé par les modèles MDD[6].

L'approche MDA se base sur trois types de modèles pour concrétiser une solution conceptuelle d'un système informatique. MDA définit les modèles PIM (Platform Independent Model), PSM (Platform Specific Model), et modèle du Code CM (Code Model). Cette approche suggère la définition d'un langage de transformation (PIM vers PSM) permettant d'obtenir le modèle spécifique à la plateforme à partir du modèle indépendant de la plate-forme. Ensuite, un second langage de transformation (PSM vers CM) assurera le passage du modèle spécifique à la plateforme au modèle du code. Enfin, la dernière transformation permet le passage du Modèle de Code (CM) en Code (Kleppe, Warmer, & Bast, 2003).

L'approche MDD met l'emphase, non pas sur le code des logiciels à développer, mais plutôt sur leur modèle (Selic, 2003). L'avantage majeur d'une telle approche est l'utilisation de

[5] MDA: Model Driven Architecture
[6] MDD: Model Driven Development

modèles qui sont assez découplés des technologies d'implémentation (langage de programmation) et qui représentent le domaine de la problématique étudiée. Ces mêmes modèles sont réutilisables dans d'autres contextes avec quelques adaptations. Ainsi, la conformité et la connectivité demeurent des problèmes qui doivent être résolus pour qu'une telle approche puisse s'imposer dans les processus de développement logiciel en général et ceux basés sur les modèles plus particulièrement.

La force des approches dirigées par les modèles réside dans la génération automatique du code à partir des modèles associés. Cette force n'a pas été suffisamment exploitée. Elle donnait partiellement des résultats dans quelques domaines spécifiques comme l'électronique, les systèmes temps réel et d'autres. La concrétisation de cette force dépend de deux concepts clés à savoir la maturité des technologies de génération automatique du code, et l'émergence des standards industriels. Dans la même optique, les experts industriels (Selic, 2003) confirment qu'on ne peut tirer un bénéfice maximal des approches MDD que lorsqu'on exploite au maximum la génération du code à partir des modèles et la synchronisation de ces derniers à partir du code associé. On parle de la manipulation ou synchronisation bidirectionnelle Modèle-Code, Code-Modèle.

Les approches "MD" s'avèrent prometteuses et sont considérées comme des orientations futures dans le domaine de développement des logiciels et des systèmes informatiques. Ces mêmes approches sont utilisées à des fins d'intégration et d'agrégation de composants logiciels. On parle aussi d'assemblage, de coordination et de communication entre composants. Les modèles de composants sont nécessaires pour concevoir des modèles d'agrégation des composants. Dans notre étude, nous nous intéressons aux modèles des composants et aux approches dirigées par les modèles pour promouvoir la réutilisation des composants, de leur agrégat et de leurs modèles. Ces approches doivent une partie de leur efficacité, de leur utilisation et de leur apport aux processus de développement qui guident les étapes du développement du logiciel. Dans les sections suivantes (1.5 et 1.6), la réutilisation et les processus de développement à base de composants sont présentés avec quelques exemples industriels.

1.5 La réutilisation dans le développement logiciel à base de composants

La réutilisation a était évoquée par McIlroy (McIlroy, 69) dans une conférence de l'OTAN sur l'ingénierie du logiciel. Ce dernier a proposé en 1969 l'idée de catalogue de composants logiciels à partir desquelles des blocs de composants logiciels peuvent être définis et assemblés. Malgré que des études de cas aient connu du succès, la réutilisation était à une étape embryonnaire. Elle est définie comme une nouvelle approche pour le développement des systèmes informatiques à partir des composants logiciels sur étagère. Elle s'oppose ainsi aux approches traditionnelles de développement qui partent de rien et requièrent le développement de toute l'application avec tous les livrables nécessaires des étapes de développement.

La réutilisation vise essentiellement trois objectifs principaux : réduire le temps de mise sur le marché (*time-to-market*), améliorer la qualité du logiciel et diminuer les coûts de développement et de maintenance. Au fil du temps, le développement logiciel s'est enrichi de plusieurs concepts liés à la réutilisation, tels que les patrons (Gof), les processus (Mili, Mili, Yacoub, & Addy, 2002), les composants (CBSD), les modèles (MDA), les métadonnées des ressources logiciels, etc. les autres concepts ont été présentés dans les sections précédentes; dans la présente section, nous rappellerons uniquement le concept "processus de réutilisation". Deux processus complémentaires sont considérés dans cette dimension de la réutilisation : l'ingénierie de composants réutilisables (*design for reuse*) et l'ingénierie du logiciel par réutilisation de composants (*design by reuse*). Dans cette thèse, on s'intéresse davantage à l'ingénierie du logiciel par la réutilisation des logiciels en général et de l'unité "composant logiciel" en particulier.

1.6 Ingénierie du logiciel par réutilisation de composants

L'ingénierie du logiciel par la réutilisation est définie par les étapes de recherche, de sélection, d'adaptation et d'intégration de composants. Durant la recherche, on essaie de

trouver le composant qui répond aux spécifications du système à développer. Souvent, plusieurs composants sont retournés par la recherche dans un entrepôt de composants; il importe de sélectionner parmi ceux trouvés le plus approprié à l'agrégation. Une fois sélectionné, le composant pourrait être adapté au contexte du nouveau système en cours de développement. Les techniques d'adaptation sont nombreuses et commencent par la modification du composant jusqu'à l'instanciation en passant par la paramétrisation et la spécialisation (Cauvet *et al.*, 1999, 2001). Enfin, l'étape d'intégration du composant met fin au processus de réutilisation en appliquant les techniques d'intégration et en utilisant des technologies détaillées dans la section 1.9.

Les composants COTS (*Commercial off-the-shelf*) sont développés par des fournisseurs spécialisés pour être ensuite utilisés dans la construction de nouveaux systèmes, souvent hybrides. (McDermid, Talbert 1997). L'utilisation des composants COTS ne cesse de croitre. Plusieurs organisations utilisent l'approche composant pour améliorer la qualité des systèmes, réduire leur coût et le temps de développement. Cependant, les développements qui s'appuient sur les composants COTS rencontrent plusieurs problèmes spécifiques durant les phases de sélection des composants, leur intégration (agrégation), leur entretien et leur sécurité. Ces problèmes se résument essentiellement dans la visibilité (réutilisation difficile du composant COTS de type boîte noire), le contrôle (impossible de contrôler les interfaces, les fonctionnalités et leur plan d'évolution), la sécurité et l'entretien des composants COTS (Dowling, 2000). Les quatre phases précédentes intègrent de façon différente plusieurs processus de développement à base de composants tels que RUP(Larman, 2002), CUP(Renaux, Olivier, & Jean-Marc, 2004), etc. Durant la mise en pratique de ces phases, plusieurs incompatibilités entre les composants à intégrer peuvent nuire à l'aboutissement de l'intégration. En effet, la classification de ces incompatibilités dans une intégration COTS permet de détecter en amont des problèmes associés à l'hétérogénéité des systèmes, des architectures et des services fournis (Yakimovich, Travassos, & Basili, 1999). Cette classification n'est pas suffisante pour étudier la faisabilité du développement à base de composants. De ce fait, des processus pragmatiques doivent être définis et spécifiés pour réduire les échecs de réutilisation de composants causés par le manque d'analyse de la

faisabilité tôt dans le processus de réutilisation. Ce type de processus s'avère un domaine de recherche en soi et fait partie de nos travaux pour soutenir la réutilisation (Mili, Mili, Yacoub, & Addy, 2002).

Des chercheurs et des spécialistes de l'industrie visent le développement de nouveaux systèmes par la réutilisation de bout en bout de manière semi-automatique. L'intervention des développeurs se limite à configurer les besoins particuliers du système. Une telle approche doit être bien organisée et complète afin de couvrir toutes les tâches tenant compte de leur complexité. Plusieurs travaux se concentrent sur la recherche de méthodes systématiques qui sont d'avantages des adaptations au développement de systèmes par la réutilisation (Soukuo, 1995) (Albin-Amior *et al.*, 2001) (Conte *et al.*, 2001) (Guérhenceuc, 2003) (Arnauld *et al.*, 2004) (Khayati, 2005). Il est intéressent de rappeler que ces équipes privilégient la mise en œuvre de nouveaux environnements de développement, qui fournissent les fonctionnalités de gestion des collections de composants, de spécification des logiciels à base de composants, l'implémentation par génération de code à partir des spécifications et des modèles, la validation des spécifications des systèmes et l'entretien des systèmes et les processus connexes. Il est important de noter que de telles approches, appuyées par des environnements complets et complexes, n'encouragent pas les organisations à réutiliser vu l'effort requis pour leur mise en place.

1.7 Les processus de développement à base de composants

Nous connaissons d'autres adaptations du processus RUP pour assurer une intégration de composants qui s'appuie sur les modèles logiques PIM (Platform Independent Model). Nous citons le processus CUP (Component Unifed Process). L'idée fondamentale du processus est la proposition d'un métamodèle structuré en paquetages articulés se référant à la notion de composant logique. Il assure la vérification de la cohérence d'un modèle avec un formalisme dédié, la traçabilité améliorée, ainsi que la gestion de l'impact du changement. Le processus résultant présente une identification anticipée des besoins et la réutilisation de briques fonctionnelles (Renaux, Olivier, & Jean-Marc, 2004).

CUP est aussi un outil de conception à base de composant et un générateur de composant vers des plateformes technologiques récentes (Ex : Java Enterprise Edition). L'architecture de « CUP Tool » est conçue à partir du métamodèle CUP. CUP est aussi une démarche dirigée par les modèles selon une approche MDA.

Les deux avantages majeurs de CUP se manifestent en deux concepts qui sont le composant logique et le lien « ExternalUse ». Le composant logique est une réalisation d'un ensemble de cas d'utilisation. Sa granularité correspond à la notion de modèle. « External Use » est un concept qui permet d'exprimer la dépendance entre deux composants dans la vue des cas d'utilisation dans un formalisme compréhensif par un utilisateur final. Toutefois, une utilisation abondante de ce lien peut affecter le couplage du système. En effet, la projection des liens « ExternalUse » dans une plateforme technologique (PSM) puis dans la génération du code associé augmentera le couplage entre les composants techniques. Ceci nuit à l'entretien du code généré et à sa réutilisation et par conséquent à la qualité du système produit. Le projet CUP permet une véritable capitalisation aussi bien sur les développements des composants que sur leur modèle crée dans chaque étape du cycle de vie. La traçabilité et la qualité du logiciel sont améliorées ainsi que la gestion du changement.

D'Souza & Wills proposent Catalysis, qui est une démarche d'ingénierie à base de composants existants. Elle est capable de s'adapter à tout type de projet d'ingénierie. Cette démarche se distingue par l'utilisation systématique du langage de contrainte OCL, ce qui impose une rigueur dans la spécification des composants. Elle définit de nouveaux frameworks et en réutilise d'autres existants. Ses utilisateurs lui reprochent sa complexité et le fait qu'elle conduit à une décomposition très fine. Elle ne dispose pas de mécanisme d'identification de composant. De plus, cette méthode n'offre pas de véritable cadre méthodologique pour décrire tout le cycle de développement (D'Souza & Wills, 1998).

Cheesman et Daniels proposent une méthode de spécification des composants intégrée dans une démarche d'ingénierie des systèmes d'information sans aborder tous les domaines. Cette

méthode s'intitule « UML Component » et propose une spécification d'un système à base de composants (Cheesman & Daniels, 2001). Elle spécialise la démarche Catalysis et définit une méthode précise de spécification d'application à base de composants logiciels de l'étude des besoins métier à leur déploiement dans diverses technologies. La méthodologie UML Component ne suit pas une approche MDA complète. Elle sépare les préoccupations et des activités de type PIM et PSM. Cette méthode limite les problématiques d'intégration des préoccupations techniques dans le résultat d'une réflexion neutre.

Le cycle de vie des systèmes à base de composants est le cycle de vie d'un composant logiciel avec plus d'emphase sur les règles d'affaires, la modélisation des processus d'affaires, la conception, les tests, le déploiement, l'évolution et la réutilisation et la maintenance ultérieures. En général, la phase d'analyse et de conception durent plus longtemps que les processus de développement logiciel classique. En effet, durant ces phases, nous nous attardons sur les aspects de flexibilité, réutilisabilité, l'interopérabilité et la maintenabilité des composants participants. Voici trois autres modèles de processus de développement à base de composant : Select Perspective (Princeton Softech, 2000), Stojanovic (Stojanovic, 2005) et COSE (Chaitanya *& al*, 2011).

Select Perspective est une méthode de la compagnie Princeton Softech pour le développement d'applications logicielles. Elle est supportée par plusieurs outils dont les plus importants sont « Select Enterprise » et « Select Component Manager ». Le premier concerne la modélisation des processus d'affaires, l'élaboration des modèles UML et les modèles de données. Le deuxième est une bibliothèque de spécification de composants logiciels. Select Perspective est vue par ses auteurs comme une approche pratique pour le développement d'application. L'approche se concentre davantage sur les composants résultants du processus d'agrégation qui sont extensibles et flexibles pour supporter les changements, l'adaptation et l'évolution. Pour atteindre cet objectif, la méthode se base sur l'approche CBD. La compagnie Princeton Softech confirme que leur approche réutilise et étend les meilleures

pratiques des approches de développement structurées telles que RAD[7] et le développement orienté objet.

Select Perspective est une approche hiérarchique. Elle est composée par des phases et des activités principales pour chaque phase. Elle propose une méthode standard pour chaque activité. Elle est composée de trois phases majeures: la phase d'alignement, la phase d'architecture et la phase d'assemblage. La Figure 1.2 montre des rectangles gris et blancs qui représentent la portée de l'application finale : chaque rectangle blanc qui s'ajoute, durant l'avancement du processus, représente une partie de la solution développée de façon incrémentale dans le but de construire l'application finale conformément aux besoins. L'architecture globale du système est développée et livrée aussi de façon incrémentale. Les activités de l'approche sont exécutées de façon itérative et parallèle.

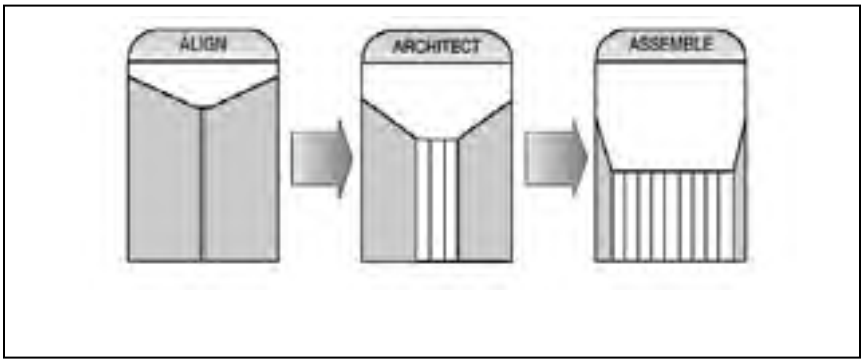


Figure 1.2 Les trois phases majeures du projet
de développement de l'approche Select Perspective
Tirée de Boertien *et al.* (2005)

Select Perspective dispose de son propre processus d'agrégation appelée LUCID. L'idée principale de toutes les étapes est le lien fort (connectivité forte) entre les modèles des processus d'affaires et les applications à travers les modèles de transformation. L'approche produit un résultat visible et exécutable après chaque phase de développement.

[7] RAD: Rapid Application Development

Stojanovic met l'emphase sur l'évolution du composant depuis la définition des exigences jusqu'à l'implémentation. Les étapes connues de développement logiciel telles que la définition des exigences, l'analyse, la conception, et l'implémentation ont été substituées par les exigences des services des composants, leur identification, leur spécification, leur assemblage et leur déploiement. Après la spécification complète du système basé sur ces composants, l'architecte décide de développer un nouveau composant, d'adapter les composants existants, d'acheter des composants COTS ou d'appeler des services à travers le réseau.

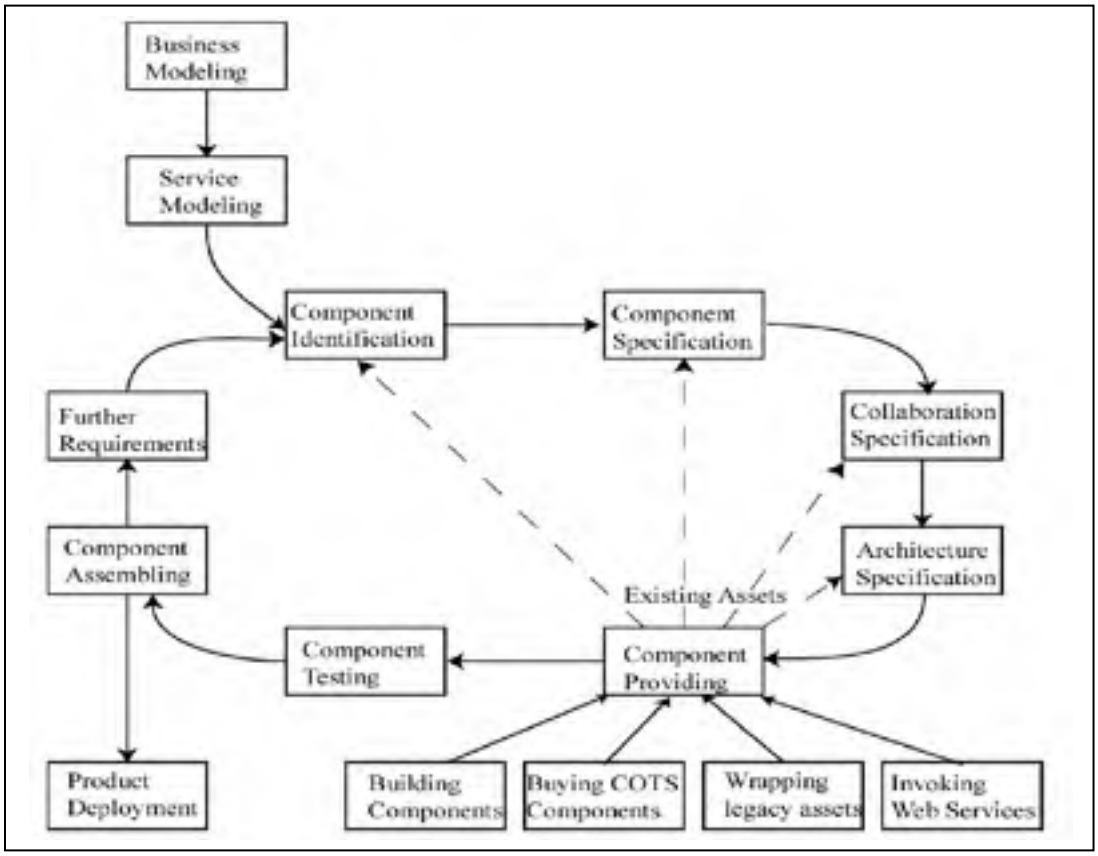


Figure 1.3 La méthodologie de développpement
à base de composants de Stojanovic
Tirée de Stojanovic (2005)

Ali H. Dogru *et al.* avancent que les méthodologies à base de composant n'ont pas atteint un niveau de maturité comparable à celui des approches OO (Dogru *et al.*, 2003). Au début, les méthodologies de développement logiciel s'inspiraient des modèles des processus en

cascade. Dès l'apparition du concept OO, ces méthodologies ont évolué pour s'aligner avec la granularité de l'objet, ainsi nous avons assisté à la création des approches orientées objets. De même, le développement à base de composant a introduit un autre niveau d'abstraction, le composant. Nous devons utiliser ce niveau d'abstraction pour définir des processus d'ingénierie logicielle basés sur les composants.

Le modèle de processus COSE utilise une approche descendante pour introduire les spécifications des blocs du système à développer. Il raffine ses spécifications pour décomposer ces briques en granularité plus fine, le composant. En même temps, il utilise une approche ascendante (de bas en haut, ou « bottom-up ») pour vérifier s'il existe des composants réutilisables ou un regroupement possible des composants existants qui satisfont ses besoins et qui réalisent les composants d'affaires déjà définies à haut niveau (Dogru *et al*., 2003).

Le processus COSE est composé principalement de cinq phases :

- spécification du système;
- décomposition du système;
- spécification des composants avec les activités de recherche, de modification et de création;
- intégration;
- tests du système.

Le processus commence avec une phase de compréhension et de spécification des requis du nouveau système. Il poursuit avec une étape de recherche pour identifier les composants existants qui satisfont les requis du système. La phase de spécification requiert l'information concernant les problèmes potentiels et ceux du domaine d'affaires. À la fin, cette phase produit les spécifications à haut niveau des requis fonctionnels, non fonctionnels et une liste des composants existants associés aux domaines d'affaires.

Durant la phase de décomposition, les requis fonctionnels et non fonctionnels sont détaillés et leur spécification est documentée. Les composants participants dans le nouveau système sont identifiés et développés dans la phase de spécification des composants. Parfois, nous réutilisons des composants existants ou nous avons recours à un nouveau développement pour satisfaire les besoins du nouveau système.

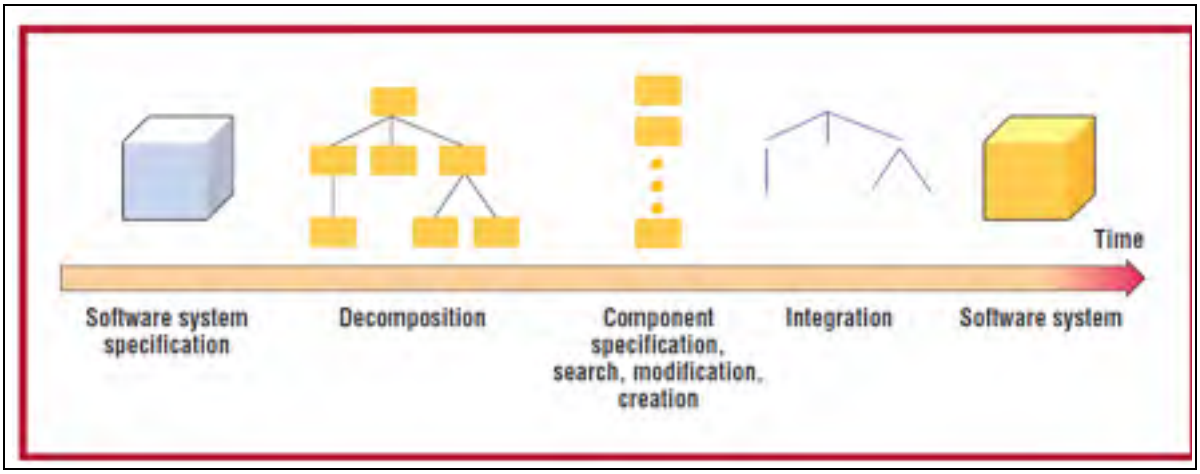


Figure 1.4 Le modèle général du process de développpement
à base de composants CORSE
(Tirée de Dogru *et al.*, 2003)

Du point de vue des environnements à base de composants, plusieurs équipes définissent des cadres de travail ayant pour objectif la génération de code à partir des modèles. La majorité d'entre elles se basent essentiellement sur les spécifications générales, techniques et fonctionnelles des composants pour le développement de l'agrégat. Rares sont celles qui traitent simultanément, en plus de ces spécifications, les volets non fonctionnels et d'intégration/agrégation. De plus, la majorité d'entre elles étudient des modèles de composants instanciés du même métamodèle, ce qui facilite l'intégration des composants au niveau des modèles. Ces composants, de technologies souvent homogènes, implémentent généralement des logiques métiers différentes. Les efforts des équipes sont souvent concentrés sur la compatibilité des interfaces de programmations des composants et la mise en correspondance semi-automatique entre leurs interfaces (API/IDL). L'objectif ultime est de réussir l'automatisation des connexions entre les composants à agréger par la génération

de code. Pourtant, les cas les plus fréquents dans l'industrie traitent de l'agrégation des composants de technologies hétérogènes, avec des modèles/métamodèles différents et en présentant différents types d'incompatibilités tels que les modèles de données, les protocoles d'interfaces, les attributs architecturaux, les méthodes d'agrégation, les types de boîte, etc. (Yakimovich, Travassos, & Basili, 1999).

RUP et les processus de développement à base de composants émergents (CUP, CORSE, Stojanovic, Catalysis et UML component) peuvent être réutilisés pour nous aider à définir un processus de développement logiciel par agrégation de nouveaux composants. Pour ce faire, on doit supporter davantage les phases de recherche, de sélection et d'identification des composants déjà développés, leur intégration, leur maintenance et leur sécurité. Ce support nécessite la définition d'un ensemble d'activités, d'outils, de règles et de techniques pour mettre en pratique les étapes et les phases d'une méthode. D'une part, l'itération du processus RUP permet la sélection itérative des composants dans un système et l'identification de ceux à développer, ceux à réutiliser et ceux à se procurer à l'externe. D'autre part, l'incrémentation du processus donnera des versions évolutives du composant produit de l'intégration (agrégation), soit l'agrégat. Nos travaux de recherche se concentrent plus sur une partie de ces processus enrichis par des étapes particulières à la réutilisation des composants existants.

Cette revue de la littérature nous porte à conclure qu'il n'existe pas de standard commun pour modéliser et construire un système à base de composants. Ce type de logiciel s'avère difficile à documenter de manière complète et cohérente. La réutilisation s'en trouve dès lors réduite. Ainsi, le domaine d'ingénierie du logiciel par réutilisation des composants demeure assoiffé au niveau des spécifications de bout en bout des composants, des processus de développement et d'évaluation des agrégations, des approches, des environnements et des techniques (Mili, Mili, Yacoub, & Addy, 2002).

1.8 Les contrats des composants

Bachman définit le contrat de composants comme une métaphore utile pour l'ingénierie logicielle à base de composants. Il caractérise le contrat de composants comme suit (Bachman, 2000) :

- Un contrat est établi entre deux ou plusieurs parties;
- Les parties négocient souvent les détails du contrat avant de le signer;
- Les contrats précisent les comportements nominatifs et mesurables de toutes les signatures;
- Les contrats peuvent être sujets à des changements qui doivent être approuvés par toutes les parties prenantes.

Un contrat crée une codépendance entre les parties participantes. Un contrat entre deux parties inclut des clauses qui obligent un respect et une dépendance mutuelle implicites. Il spécifie **des obligations réciproques** entre les deux parties.

Les contrats ne sont pas particuliers aux composants et ses concepts sont généralisables (Meyer, 1992). Les contrats peuvent être classés en quatre niveaux qui contiennent les propriétés des contrats de niveau inférieur (Beugnard, Jezequel, Plouzeau, & Watkins, 1999).

Le **contrat syntaxique** comporte les opérations que le composant peut exécuter, les paramètres d'entrée et de sortie qu'il doit avoir et les exceptions qui pourraient survenir lors de l'opération. Il permet de vérifier la conformité entre les interfaces fournies et requises du composant et facilite la vérification statique des composants.

Le **contrat comportemental** spécifie le comportement des opérations en utilisant les pré-conditions et post-conditions, pour chaque service offert et les classes d'invariants sur une déclaration d'opération (Meyer, 1992):

- La pré-condition doit être satisfaite à la demande du service par le client;

- La post-condition quant à elle doit être satisfaite par le fournisseur après la réalisation du service.

La violation d'une pré-condition indiquera une erreur à propos du client. En d'autres termes, le demandeur n'a pas observé les conditions imposées au moment des appels corrects. La violation d'une post-condition implique une erreur du côté du fournisseur, le service demandé a échoué. Plus on a de pré-conditions, plus la difficulté pour le client sera grande et plus la tâche sera facile pour le fournisseur (Meyer, 1992). Ces conditions peuvent être décrites en OCL (Object Constraint Language) (Warmer & Kleppe, 2003):

- Une pré-condition sera de la forme pré : prédicat ;
- Une post-condition sera de la forme post : prédicat.

Le **contrat de synchronisation** : le contrat comportemental considère les services comme atomiques. Le contrat de synchronisation, quant à lui, spécifie le comportement global des composants en termes de synchronisation entre les appels de méthodes. Le but de ce contrat est de décrire les dépendances entre les services d'un composant tels que la séquence ou le parallélisme. Ce contrat garantit, quel que soit le client qui demande le service, que le service soit correctement exécuté.

Le **contrat de qualité de service :** une fois toutes les propriétés comportementales spécifiées, on peut essayer de quantifier le comportement voulu, c'est-à-dire la qualité de service attendue, en énumérant les caractéristiques que doit respecter le fournisseur, par exemple:

- Le délai de réponse maximal;
- Le temps de réponse moyen;
- La qualité du résultat, que l'on peut exprimer sous forme de précision.

Dans un contexte de composition de composants logiciels, il existe plusieurs méthodes et techniques pour mettre en association deux composants. On retrouve plusieurs catégories d'agrégations dans la littérature soit la coordination, l'orchestration et l'intégration (

Villalobos, 2003). En pratique, la majorité des programmeurs utilisent la technique « Glue code » pour brancher deux ou plusieurs composants au niveau de leur code (Mili, Mili, Yacoub, & Addy, 2002). D'une part, cette technique n'est pas réutilisable parce qu'elle a été développée particulièrement pour un contexte particulier. D'autre part, ce type de pratique se traduit par des modifications dans le code des composants, ce qui peut affecter leur qualité et leur potentiel de réutilisation. En plus, l'exécution de l'agrégat résultant de la connexion est réalisée par l'un des composants; par conséquent, la connexion est dépendante de l'environnement d'exécution de l'un des composants. Toutes ces dépendances nuisent à l'entretien et à la gestion des agrégats.

En résumé, les composants entrent dans des relations d'obligations pour réaliser des agrégations en offrant des services et requièrent d'autres services pour assurer leur fonctionnement interne. Les contrats représentent ces obligations pour déclarer un nouveau composant comme une agrégation des composants existants en décrivant une ou plusieurs séquences d'appels entre leurs méthodes. Les contrats incluent aussi des contraintes qui sont exprimées à l'aide de pré et post-conditions sous la forme de prédicat.

1.9 Les technologies d'intégration des composants logiciels

Plusieurs technologies se sont succédées pour agréger des composants logiciels développés dans des technologies hétérogènes ou homogènes. L'approche EAI (Cummins, 2002) (Enterprise Application Intégration) a défini quelques concepts qui décrivent les processus d'échanges de données et de fonctionnalités (services) entre deux applications informatiques (composants logiciels) conçues indépendamment l'une de l'autre. L'interconnexion de ces applications se fait par des modèles d'intégrations, des partons architecturaux, de la génération de code et des partons de conception prédéfinis.

Les *interfaces* (Application Programming Interface, API) sont un mécanisme fourni par des composants permettant l'accès à leurs services de traitements et de données (Cummins,

2002). Chaque interface a sa propre syntaxe conformément à la technologie dans laquelle elle est développée (interfaces IDL, interfaces Java, interfaces des services Web WSDL, etc.).

L'objectif ultime de l'approche EAI est la communication et l'intégration des composants logiciels moyennant une couche logicielle intermédiaire communément appelée *intergiciel* (*middleware*). Les intergiciels sont des applications logicielles indépendantes qui fournissent des services de médiation pour les composants logiciels par l'utilisation d'interfaces ou de messages bien déterminés. Un intergiciel fournit un environnement d'exécution pour la gestion des requêtes et des réponses entre les composants logiciels agrégés.

La notion de *couplage* définit le degré d'intégration d'un composant à un autre. On parle d'un *couplage faible* quand l'intégration se fait au niveau de quelques interfaces discrètes entre les composants en question. Par contre, le *couplage fort* montre une forte dépendance des deux composants au moment de l'implémentation et de l'exécution.

Dans une perspective d'intégration, le concept de *connecteur* est très utilisé dans plusieurs technologies (Java Connector Architecture, JEE Connector, etc.). Par définition, un *connecteur* est un élément de premier plan, soit un composant logiciel qui permet l'interconnexion entre des applications ou des composants logiciels. Un connecteur fournit des ports de communication avec chaque application/composant (*adaptateur*) et les canaux entre ses adaptateurs. En général, un connecteur implémente une logique programmée permettant un accès structuré de l'extérieur à la couche de présentation, de traitement ou de données d'un composant ou une application logicielle. Le connecteur a souvent des besoins de transformation de structure de données pour réussir la communication entre les composants participants.

L'architecture orientée service (SOA) est un style d'architecture qui a pour but l'agrégation des services des composants logiciels par un faible couplage. Les services représentent des unités de communication qu'un fournisseur procure à un consommateur pour subvenir à ses besoins. Les services Web sont une implémentation de l'architecture SOA. Dans ce type

d'architecture, les applications utilisent les protocoles SOAP[8] et REST[9] pour communiquer avec les services Web.

L'architecture composant service (SCA[10]) est un ensemble de spécifications qui proposent un modèle pour construire des applications s'inscrivant dans une architecture orientée service (appelée aussi SOA). Les applications basées sur SCA sont en fait un assemblage de composants, chaque composant implémente une partie de la logique d'affaires et peut dépendre ou consommer un service.[11]

Le but de SCA est de simplifier l'écriture d'application dans un cadre SOA indépendamment des produits et langages utilisés. SCA a été écrit dans le cadre de l'architecture SOA (contrairement à d'autre système comme la plateforme JEE qui a été adapté pour le cadre SOA). Ces spécifications se focalisent sur la partie écriture de la logique métier, qui se veut indépendante d'un langage de programmation (i.e. Java, C#, BPEL etc) et du protocole d'appel de services et de circulation des données (services web, JSON, RMI, CORBA...).

SCA vise à simplifier la construction d'architectures orientées services (SOA) en adressant plus particulièrement[12] :

- La composition : comment packager un composant logiciel afin que d'autres applications puissent l'utiliser ?
- L'assemblage : comment les composants peuvent fonctionner ensemble ?
- Politique : comment associer des politiques non fonctionnels (restriction d'accès, signature numérique, etc.) aux composants ?

Dans une perspective d'agrégation, les technologies des services Web peuvent être utilisées pour faire communiquer deux applications ou deux systèmes indépendamment des

---

[8] SOAP : Simple Object Access Protocol (www.w3.org/TR/soap/)
[9] http://www.ibm.com/developerworks/webservices/library/ws-restful/
[10] Architecture Composant Service (SCA – Service Component Architecture)
[11] Le projet SCA : http://osoa.org/display/Main/Service+Component+Architecture+Home
[12] Description tirée du livre blanc « comprendre SCA »

technologies qui les implémentent à l'interne. Ils assurent une communication à la demande. Autrement dit, un service n'est exécuté que lorsqu'il est demandé par un client.

Plusieurs approches d'agrégation de composants logiciels utilisent des invocations de procédures et de méthode à distance telles que RMI (Remote Method Invocation), RPC (Remote Procedure Call). Ces approches utilisent les logiques de « stub » et « skeleton » pour simuler les appels à distances. En d'autres termes, un « stub » permet une invocation d'une méthode en local bien qu'effectivement cette dernière appelle à distance son « skeleton » qui appelle directement l'objet en question. Plusieurs plateformes supportent différents modèles d'intégration, cependant leur principe de base demeure souvent celui des intergiciels.

Pour sa part, CORBA[13] est une architecture qui permet l'invocation d'objets distribués via des médiateurs. CORBA est une plateforme indépendante qui permet la gestion de ses objets distribués à distances. Elle utilise les médiateurs d'objets ORB (Object Request Broker) qui peuvent être distribués. Ces ORBs communiquent avec d'autres ORBs pour fournir les services de ces objets distribués, sous forme de méthode, dans plusieurs plateformes.

Le Corba Component Model (CCM)[14] est un modèle qui définit un composant dans une architecture distribuée CORBA. Il est compatible avec la structure EJB (Enterprise Java Beans) de la plate-forme JEE. Le modèle CCM propose une architecture pour le composant, de même qu'une API pour l'implémentation CORBA. Il intègre aussi des interfaces pour la configuration, la définition de la composition des composants et un modèle pour le déploiement. Le CCM propose toute une structure pour définir un composant, son comportement, son intégration dans un conteneur et son déploiement dans l'environnement distribué CORBA.

---

[13] CORBA(Common Object Request Broker Architecture) :
http://www.omg.org/technology/corba/corba3releaseinfo.htm
[14] Voir la description complète d'un modèle CCM dans cette présentation disponible via l'adresse web :
http://www-igm.univ-mlv.fr/~dr/XPOSE2002/CCM/corba.htm

La technologie JEE (Java Enterprise Edition), englobe plusieurs concepts définis dans CORBA, EJB, JSP, JDBC, etc. En effet, JEE est une plate-forme de développement d'application qui s'appuie sur le langage Java, dont les spécifications sont gérées par la société Oracle Inc. JEE est aussi une plateforme de développement qui permet de développer des applications Web composées de Servlet et JSP et des applications métiers à base d'EJB. JEE comprend également des spécifications destinées aux éditeurs de logiciels qui désirent créer des serveurs d'applications compatibles JEE.

Enterprise Java Beans (EJB) est une technologie qui définit un modèle de développement et de déploiement de composants Java réutilisables du côté serveur. Ces composants sont développés en avance et sont agrégés dans des applications systèmes via les conteneurs EJB. La technologie EJB fournit plusieurs types de communications qui sont détaillés dans (Cummins, 2002). Cette technologie est un élément central de JEE, un ensemble de spécifications de technologies d'entreprise implémenté dans le langage de programmation Java.

La technologie .Net est concurrente à JEE. Microsoft .Net est une plateforme commerciale par couche. Elle définit une couche d'intergiciel similaire à celle des architectures EJB.

Dans les processus de développement par agrégation de composants, nous aurons à mettre en association ces différents types de composants hétérogènes et issus de différents modèles conceptuels. Le grand défi est de réussir à faire communiquer et interopérer ces composants avec le minimum d'effort d'adaptation et en les exécutant dans leur environnement natif.

1.10 Les résultats de recherche en matrice

Les processus de développement à base de composants suivent, dans la majorité des cas rencontrés, quatre phases principales : recherche, sélection, adaptation et intégration. La figure suivante montre une vue matricielle dont le but est de préciser à travers les chapitres

l'apport de nos travaux de recherche pour les phases du processus. Cette matrice nous sert de résumé des résultats de nos travaux et leur positionnement dans les domaines du développement à base de composants et de développement par la réutilisation. De plus, elle aide le lecteur à identifier clairement l'apport de nos travaux au fur et à mesure qu'il avance dans la lecture des chapitres de la thèse.

Tableau 1.1 Matrice des apports des résultats de recherche aux phases du processus de développement à base de composants

Résultats de la thèse	**Description des contributions des résultats par phase**
Résultat 1	- Phase de recherche : Description de la contribution. - Phase de sélection : Description de la contribution. - Phase de faisabilité : Description de la contribution. - Phase d'adaptation: Description de la contribution. - Phase d'intégration: Description de la contribution.

1.11 Conclusion

Ce chapitre a présenté sommairement et progressivement les paradigmes suivants : l'objet, le composant, les métadonnées, les ontologies, la modélisation logicielle, les processus de développement, les contrats logiciels et les technologies d'intégration logicielles. Pour chaque concept, nous avons présenté les définitions dans la littérature, leur apport et leurs défis. Ensuite, nous avons établi le lien avec différents aspects de notre sujet de recherche. Cette liste de paradigmes est nécessaire pour clairement identifier les différents concepts reliés à la problématique de recherche traitée dans cette thèse. Dans les chapitres suivants, les concepts périphériques aux composants logiciels seront élaborés, dans le but de définir une méthode d'agrégation de composants logiciels dirigée par les métadonnées et les modèles.

CHAPITRE 2

PROPOSITION DE SPÉCIFICATION DES COMPOSANTS LOGICIELS ET DE LEUR AGRÉGATION

Ce chapitre décrit les spécifications abstraites proposées pour un composant logiciel. Il présente différents types de métadonnées d'un composant logiciel : les métadonnées statiques, d'affaires, de services, non fonctionnelles et de plateformes. Le but de cette caractérisation est de rendre la description d'un composant logiciel accessible à différents types d'intervenants dans le processus de développement logiciel à base de composants tels qu'architecte, gestionnaire de projet, responsable de la qualité, responsable de la gestion des configurations, développeur et intégrateur. L'ensemble de ces métadonnées sera à partir d'ici référé comme étant la spécification SOCOM (**SO**ftware **CO**mponent **M**etadata). Un composant qui est décrit par les spécifications SOCOM est appelé "composant SOCOM". Un entrepôt de composants SOCOM est une structure qui référence ou héberge des composants SOCOM.

La définition d'un jeu de métadonnées a pour objectif particulier le référencement étendu d'un composant logiciel contrairement aux autres spécifications qui visent une catégorie particulière des attributs du composant.

Il est important de rappeler que les métadonnées présentées dans ce chapitre ne sont pas toutes proposées par l'auteur, elles proviennent des métadonnées existantes sur les objets, les services et les composants. D'autres sont proposées par l'auteur afin d'aider à la réutilisation des composants dans un contexte d'agrégation. Dans les sections suivantes, nous rappelons la source de chaque type de métadonnées.

La description de la représentation abstraite des composants logiciels est élaborée dans la troisième étape de notre méthodologie de recherche. Ce chapitre introduit le volet métadonnées d'un composant qui est un élément de base du processus cible d'agrégation des composants dirigés par les métadonnées et les modèles. Ce processus représente notre

objectif principal des travaux de cette thèse. La figure suivante positionne le présent chapitre dans le cheminement global de la thèse.

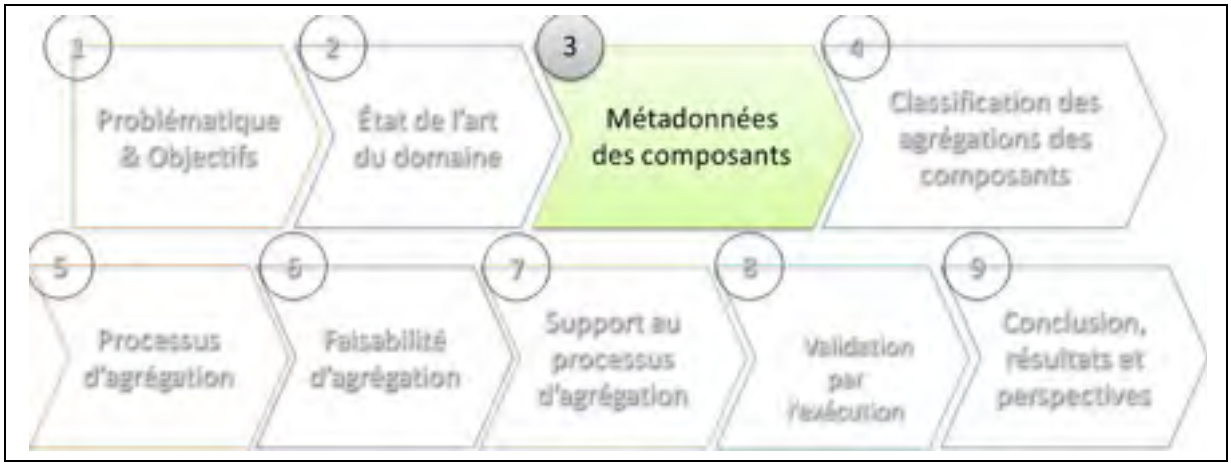


Figure 2.1 La méthodologie de recherche de la thèse – Étape 3 : Métadonnées des composants

2.1 Les différents types de métadonnées d'un composant logiciel

Dans toutes les sections suivantes, les attributs des métadonnées sont décrits par un nom logique et un nom technique. Le nom logique facilite la compréhension de l'attribut en question et il est utilisé dans le reste du document. Le nom technique est utilisé pour l'implémentation technique des spécifications SOCOM sous différents formats tels que : XML, base de données relationnelle, OWL, etc. Il est important de noter que les noms techniques sont en anglais; ceci est dû à ce que ces spécifications ont été utilisées dans des contextes anglophones des partenaires de LICEF et des clients de notre employeur (CGI).

Les métadonnées proposées sont réparties, initialement, en six catégories. En effet, cette catégorisation est utile pour servir différents types d'utilisateurs pour promouvoir la réutilisation des composants durant les phases de recherche, de sélection, d'adaptation et d'intégration des composants. Par exemple, les analystes d'affaires et les gestionnaires de projet s'intéressent davantage aux métadonnées qui introduisent le contexte d'affaires des composants à agréger alors que les architectes utilisent les métadonnées qui décrivent le

comportement des composants et des plateformes associées pour juger de la faisabilité technique de leur agrégation.

2.1.1 Spécification des métadonnées statiques

Les ***métadonnées statiques*** décrivent les caractéristiques statiques d'un composant tels que son nom logique, sa description générale, ses identificateurs etc. Elles sont utiles pour les acteurs d'affaires qui n'exigent pas la connaissance du comportement détaillé et technique du composant comme, par exemple, un architecte d'affaires ou un chef/directeur de projet. Souvent, ces acteurs ont seulement besoin de savoir que le composant existe, et de connaitre son nom et une brève description.

Ce type d'information est rarement modifiable. Chaque composant logiciel est défini par un ensemble unique de métadonnées statiques. Le nom technique de cette section des métadonnées est **<staticSpecification>[1]** (le [1] indique la cardinalité, dans ce cas-ci il doit exister exactement une instance de cette métadonnée). La majorité de ces métadonnées sont utilisées dans la description des objets JAVA ou .NET dans les environnements de développement logiciels tels que Visual Studio de Microsoft, IBM WebSphere, Eclipse IDE, etc.

- Identificateur local unique **<localIdentifier>[1..*]**: c'est l'identificateur local du composant qui le référence dans un entrepôt de composants SOCOM d'une organisation donnée.
- Identificateur global unique **<globalIdentifier>[1]**: c'est l'identificateur global du composant qui le référence dans une banque SOCOM de sa banque locale dans un contexte de référencement distribué.
- Nom technique du composant **<name>[1..*]**: c'est le nom technique du composant qui reproduit le nom du paquetage qui l'implémente (Ex : nom de l'assemblage, archive ou librairie jar, etc.).

- Description du composant **<description>[1..*]**: c'est une brève description du composant qui informe sur ses fonctionnalités générales. Cette description contient aussi le nom logique du composant exprimé dans le langage naturel. (Exemple : gestComp est un nom technique et "Gestionnaire de Composants logiciels" est un nom logique associé)
- Langage du composant **<language>[1..*]**: c'est le langage utilisé pour développer le composant. Un composant peut être développé avec un ou plusieurs langages.
- Auteur du composant **<author>[1..*]**: c'est le propriétaire du composant. La personne ou la compagnie qui a développé le composant.
- La version du composant **<version>[1]**: c'est la version actuelle du composant.
- Le nœud système du composant **<systemNode>[1..*]**: le composant est souvent hébergé dans un système distribué qui contient plusieurs ordinateurs. Cet attribut spécifie la ou les références des ordinateurs dans lesquels est hébergé le composant.
- L'emplacement du composant **<technicalLocation>[1..*]**: l'URL où se trouve l'emplacement technique du composant dans le réseau Intranet de l'entreprise ou Internet si le composant offre ses services à partir d'Internet.

2.1.2 Spécification des métadonnées d'affaires

Les métadonnées d'affaires décrivent les caractéristiques des cas d'utilisation d'affaires réalisés par un composant logiciel. Ces cas d'utilisation d'affaires traduisent les services d'affaires du composant à haut niveau. Pour chaque cas d'affaires, on associe un nom, une description et un acteur. Ce dernier peut être humain ou système et il est responsable de la réalisation du cas en question. Par ce concept, la description du composant SOCOM dispose d'une double description pour chaque service, soient des informations techniques fournies par le concept « ServiceSpecification » et celles des affaires fournies par « BusinessSpecification ».

Ce type d'information rappelle les besoins d'affaires du composant. Quant à leur provenance, ces métadonnées s'inspirent des spécifications UML qui décrivent un cas d'utilisation (Kruchten, 2004). On les retrouve dans les environnements de modélisation UML tels

qu'IBM-Rational Modeler, Eclipse Modeler, Enterprise Architect, Together, etc.(Kruchten, 2004). Les cas d'utilisation ne sont pas les seules sources pour remplir les métadonnées d'affaires. L'architecte peut utiliser un document de vision, un document des spécifications fonctionnelles, des rencontres avec les propriétaires du composant ou même explorer le composant en exécution pour alimenter le jeu de métadonnées proposées.

Chaque composant logiciel est défini par plusieurs métadonnées d'affaires. Le nom technique de cette section des métadonnées est **<businessSpecification>[1]**.

- Identificateur du cas d'affaires **<businessCaseIdentifier>[1]**: c'est l'identificateur unique du cas d'affaires associé au composant.
- Acteur du cas d'affaires **<description>[1..*]**: c'est le nom/rôle de l'acteur qui utilise le composant pour satisfaire un besoin d'affaires.
- Nom du cas d'affaires **<name>[1]**: c'est le nom du cas d'affaires réalisé par le composant. Ex : « Accès à un portail Intranet » est le nom du cas d'affaires réalisé par un employé de l'entreprise fourni par le composant logiciel « Portail Intranet ».
- Description du cas d'affaires **<description>[1..*]**: c'est la description générale du cas d'affaires. On peut décrire aussi le type d'association entre l'acteur et le cas d'affaire en question. En d'autres termes, on décrit le type de relation (acteur/cas) similaire au concept association en UML. Ex : un représentant à la caisse dans une banque traite un paiement de facture de son client. Ce cas est décrit par l'association « exécute » ou « réalise ».

2.1.3 Spécification des métadonnées des services

Les ***métadonnées des services*** concernent le volet service d'un composant logiciel. Cette section comporte la liste des services offerts par le composant à son environnement externe. Par définition, un service est une implémentation d'une fonctionnalité requise ou fournie par le composant.

Elles sont utiles pour les acteurs techniques qui exigent la connaissance du comportement détaillé et technique du composant tel qu'un architecte applicatif. Souvent, ces acteurs étudient les agrégations potentielles des services entre plusieurs composants. Quant à leur source, ces métadonnées s'inspirent des spécifications WSDL pour la description d'un service web et des spécifications des attributs du langage UML qui décrivent les méthodes des classes UML. On les retrouve aussi dans les fiches des propriétés des environnements de modélisation UML tels qu'IBM-Rational Modeler, Eclipse Modeler, Enterprise Architect, Together, etc. Ces métadonnées sont utiles dans l'évaluation de la faisabilité de l'agrégation des services à l'étape d'analyse. Elles sont aussi utilisées par les architectes pour concevoir les interactions des services du nouveau composant agrégat.

Les spécifications WSDL et des classes d'UML ne sont pas les seules sources pour remplir les métadonnées d'affaires. L'architecte peut utiliser un document d'architecture, un document de conception logicielle ou même explorer le code du composant pour alimenter le jeu de métadonnées proposées.

Chaque composant logiciel offre un ou plusieurs services. Le nom technique de cette section est **<serviceSpecification>[1..*]**. Dans ce qui suit, on détaille les attributs d'un service :

- Service du composant **<service>[1..*]**: un composant fournit des services à son environnement externe et en consomme d'autres pour assurer le fonctionnement de ses responsabilités internes. Chaque description d'un service est constituée de :
- Nom du service **<name>[1]**: nom technique identique à la signature du service décrite dans le code du composant (ex : API pour les composants JAVA et IDL pour les composants CORBA).
- Description du service **<description>[1]**: il s'agit d'une description sommaire du service identique aux commentaires que l'on retrouve typiquement dans le code des classes Java ou C++, etc.
- Version du service **<version>[1]**: cet attribut indique la version actuelle du service en question. Ceci est utile lors de l'implémentation du client qui va consommer ce service.

- Type de retour du service **<ReturnedType>[1]** : dans différents type de langages d'implémentation, un service a un type de retour. Nous avons choisi les types communs les plus utilisés des spécifications XML : « String », « Integer », « Void », « Double », « Collection » et « Boolean ». Cependant d'autres types de retour peuvent être ajoutés au besoin.
- Visibilité du service **<visibility>[1]** : identique à la visibilité des méthodes dans les langages de programmation : « Public », « Protected» ou « Private ».
- Type de service **<type>[1]**: cet attribut indique si le service décrit est un service fourni par le composant ou un service requis par celui-ci. Les deux valeurs possibles sont « ProvidedService », « RequiredService ».
- Librairies nécessaires **<requiredLibrary>[1..*]** : chaque composant utilise une ou plusieurs librairies pour implémenter ses services. Pour chaque librairie, nous spécifions son nom et son emplacement techniques.
    - Nom technique de la librairie **<name>[1].**
    - Emplacement physique de la librairie <**technicalLocation>[1]**.
- Paramètres des services **<parameters>[1..*]** : chaque service utilise des données d'entrées représentées sous formes de paramètres. Pour chaque paramètre, on indique :
    - Nom du paramètre **<name>[1]** : c'est le nom technique du paramètre.
    - Type du paramètre **<type>[1]**: identique aux types de services énumérés ci-dessus.
    - Visibilité du paramètre **<visibility>[1]** : identique à la liste indiquée ci-dessus pour l'attribut visibilité du service.
    - Rang du paramètre **<rank>[1]** : c'est l'ordre du paramètre par rapport à la liste des paramètres associés à un service donné. Par exemple, le paramètre « id » a le deuxième rang dans la description de ce service : Collection getRessource(String name, Int id).

2.1.4 Spécification des métadonnées non fonctionnelles

Les ***métadonnées non fonctionnelles*** couvrent des aspects dits non fonctionnels, la plupart du temps des attributs de qualité du composant mais qui ont un impact sur sa réutilisation potentielle dans un domaine autre que celui de son développement d'origine. Elles informent essentiellement sur la fiabilité, la sécurité, l'extensibilité, le droits d'utilisation, la version, la licence, la performance, l'intégrité, etc.

Elles sont utiles pour les responsables de qualité qui exigent la connaissance d'information non fonctionnelle du composant. Souvent ces aspects sont déterminants et utiles lors du processus d'évaluation de l'agrégation des composants (Yakimovich *et al.*, 1999). Souvent les agrégations des composants sont faisables de point de vue fonctionnel, mais des considérations non fonctionnelles font que l'agrégat fonctionnel résultant ne satisfasse pas les besoins globaux de la solution, d'où l'intérêt d'identifier et de documenter ces aspects dans un contexte de développement à base de composants. Garlan a identifié plusieurs disparités architecturales, nous en réutilisons quelques-unes dans nos travaux (Garlan, 2004).

Chaque composant est caractérisé par un ou plusieurs attributs non fonctionnels. Le nom technique de cette section des métadonnées non fonctionnelles de SOCOM est **<nonfunctionalSpecification>[1..*]**.

- Type de l'attribut **<type>[1]**: on distingue différents types de besoins non fonctionnels tels que la performance, la version, la licence, la fiabilité, etc.
- Description de l'attribut **<description>[1]** : il s'agit d'une brève description du besoin non fonctionnel du composant en question.
- Documentation de l'attribut **<documentation>[1..*]**: c'est un ou plusieurs documents qui expliquent les détails spécifiques du besoin non fonctionnel étudié. Pour chaque document, on spécifie son nom **<title>[1]** et son emplacement physique **<technicalLocation>[1]** pour le consulter au besoin.

2.1.5 Spécification des métadonnées des plateformes

Les métadonnées des plateformes couvrent les aspects environnementaux du composant au niveau des plates-formes de développement, de configuration, de modélisation, de la persistance (données), des langages de programmation, de déploiement, etc (OMG, 2003).

Elles sont utiles pour les responsables de la configuration afin de procéder à la mise en place et la configuration de ces environnements. Souvent, ces aspects sont déterminants et très utiles au début des processus de développement et à la fin durant la mise en production des composants.

Chaque composant est caractérisé par un ou plusieurs attributs des plateformes. Ces métadonnées informent des différentes plateformes que le composant utilise durant tout le cycle de son développement et de son exploitation. La description de ces métadonnées est identique à celles des spécifications non fonctionnelles. L'idée principale de proposer cette structure simple est d'étendre les spécifications SOCOM avec des données utiles et non complexes utilisées par les responsables de la configuration pour qu'ils puissent manipuler les composants lors de la réutilisation dans un contexte précis. De plus, ces métadonnées peuvent informer sommairement sur les aspects des plateformes comme elles peuvent référencer des informations détaillées, techniques, pratiques et plus complètes sur les aspects en question.

Le nom technique de cette section des métadonnées des plateformes de SOCOM est **<platformSpecification>[1..*]**

- Nom de la plateforme **<name>[1]**.
- Type de la plateforme **<type>[1]**: on distingue différents types de plateforme tels que développement, déploiement, test, modélisation, de données, etc.
- Documentation du composant associée à la plateforme <**documentation**>[1..*]: c'est un ou plusieurs documents qui expliquent les détails spécifiques des plateformes. Pour

chaque document, on spécifie son nom **<title>[1]** et son emplacement physique **<technicalLocation>[1]** pour le consulter au besoin.

2.2 Différentes représentations de SOCOM

Pour valider l'apport et les bénéfices des spécifications SOCOM, des implantations techniques avec différents formats s'imposent. Trois types de représentations sont nécessaires avec des avantages connexes et complémentaires. Principalement, la représentation relationnelle est utile pour faciliter la mise en place d'un entrepôt de composants SOCOM et expérimenter ainsi le référencement et la recherche. La représentation XML est bénéfique pour l'échange de métadonnées des composants entre les entrepôts de composants SOCOM et non SOCOM. Quant à l'implémentation ontologique, elle est utilisée pour expérimenter la recherche « intelligente » avec un langage de requête faisant appel aux capacités de déduction d'un moteur d'inférence.

2.2.1 Représentation relationnelle

2.2.1.1 Modèle relationnel de SOCOM

Le diagramme de classe de la figure suivante décrit une partie du modèle relationnel des métadonnées SOCOM. Le diagramme montre une classe « SoftwareComponent » composée de plusieurs spécifications représentées par des liens d'association UML de type « agrégation par valeur ». Chaque classe associée à la classe « SoftwareComponent » représente une catégorie des métadonnées de SOCOM. La classe « AggregateComponent » introduit le concept de composant agrégat qui est le produit résultant d'un développement à base de composants. La description détaillée de cette classe est fournie dans le chapitre 4. Elle représente l'agrégat qui est à son tour un composant logiciel, d'où le lien d'association par héritage entre les deux classes « SoftwareComponent » et « AggregateComponent ». Les classes de type « DataType » sont des stéréotypes qui définissent les types de SOCOM associés aux différents attributs proposés.

Ce modèle nous permet la création du schéma de la base de données relationnelle de SOCOM. Nous l'utilisons aussi dans le développement du code de l'application Web « Gestionnaire SOCOM » qui est décrite dans la section 2.3.

Ce modèle de classe de SOCOM n'est pas la version finale. Il est enrichit par d'autres types de métadonnées dans les chapitres suivants.

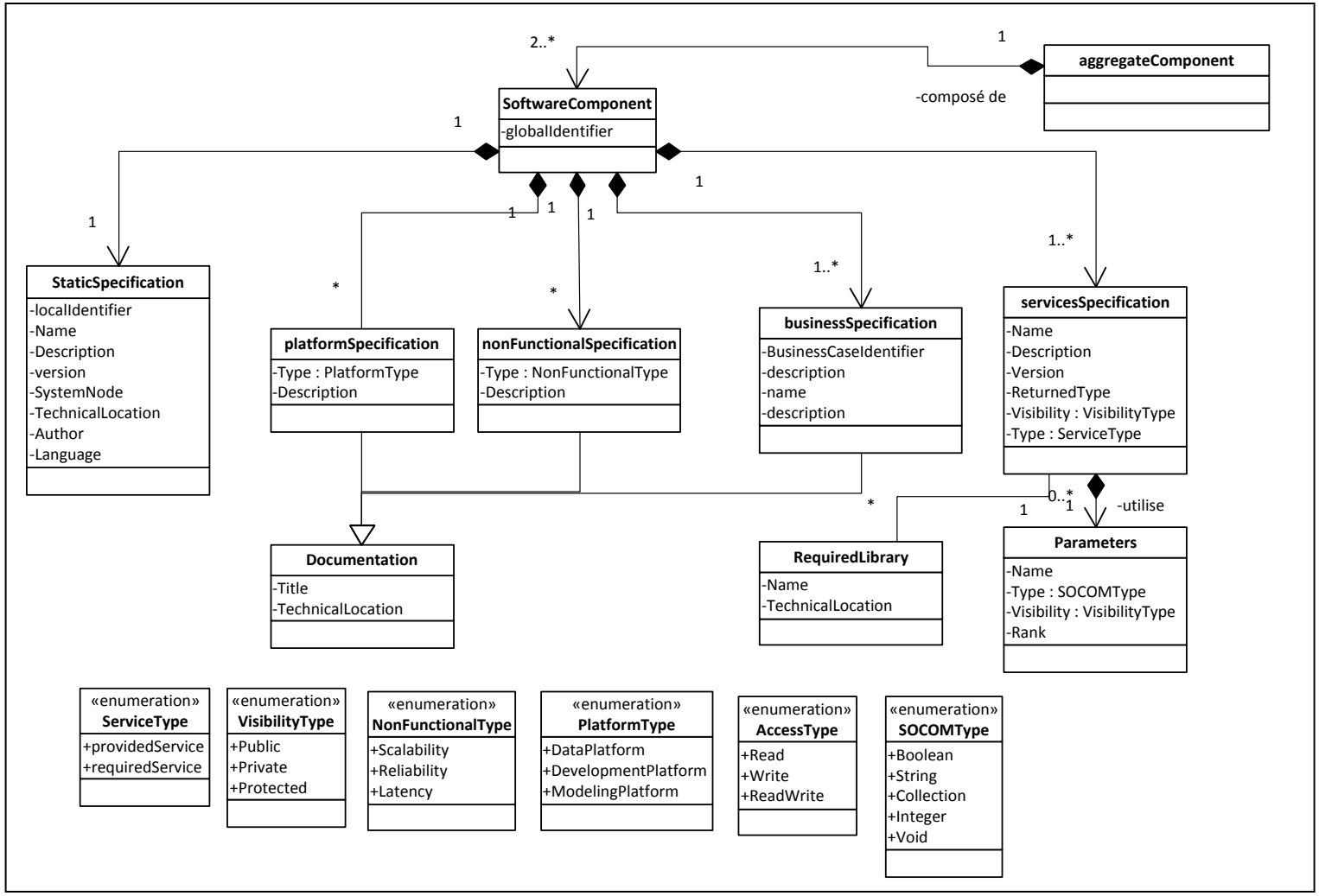


Figure 2.2 Modèle conceptuel des spécifications SOCOM (diagramme des classes UML)

2.2.1.2 Apport de la représentation relationnelle de SOCOM

Ce modèle est implanté avec un SGBDR MySQL. Des interfaces graphiques ont été développées pour répondre à différents besoins :

- implantation système des métadonnées SOCOM;
- création et gestion simple et rapide des métadonnées des composants;
- centralisation du référencement des composants d'une organisation;
- partage des informations sur les composants d'une organisation;
- recherche des composants référencés en utilisant deux modes: simple et avancé.

2.2.2 Représentation en format d'échange XML

2.2.2.1 Modèle de la représentation XML de SOCOM

La Figure 2.3 montre une partie du document XML qui représente quelques sections des spécifications SOCOM décrites ci-dessous.

```
<AggregateComponent>
        <softwareComponent>
                <globalIdentifier/>
                <staticSpecification>
                        <Name/>
                        <author/>
                        <localIdentifier/>
                        <description/>
                        <technicalLocation/>
                        <version/>
                        <systemNode/>
                <language/></staticSpecification>
                …
                </servicesSpecification>
                        <service>
                        …
                        </service>
                        …
                </servicesSpecification>
                </technicalSpecification>
                        <nonFunctionalSpecifications>
                                <nonFunctionalSpecification>
                        <type/>
                        <documentation>
                                                <title/>
                                <technicalLocation/>
                        </documentation>
                        </nonFunctionalSpecification>
                        …
                        </nonFunctionalSpecifications>
                <platformsSpecifications>
                        <platformsSpecification>
                        <type/>
                        <documentation>
                                                <title/>
                                <technicalLocation/>
                        </documentation>
                        </platformsSpecification>
                        …
                </platformsSpecifications>
        </softwareComponent>
        <softwareComponent>
                …
        </softwareComponent>
</AggregateComponent>
```

Figure 2.3 Modèle XML des spécifications SOCOM (Document XML)

2.2.2.2 Apport de la représentation XML de SOCOM

Ce type de représentation est utile pour l'importation et l'exportation des métadonnées des composants logiciels et les échanges de composants entre des environnements de référencement et de développement à base de composants. En effet, avec la représentation XML, on a échangé dans le cadre du projet LORNET des métadonnées des composants du système TELOS et de l'application Web « Gestionnaires des composants SOCOM ». Dans un premier temps, tous les composants des partenaires du projet LORNET ont été référencés dans l'application « Gestionnaire SOCOM », ce qui a permis de référencer et de rechercher au besoin l'ensemble des composants par les utilisateurs. Ensuite, quelques composants ont été exportés sous format XML pour une réutilisation par les systèmes des partenaires. En effet, lors de l'exportation, un fichier XML est généré et intégré dans un fichier archivé (.ZIP) avec le code binaire/exécutable du composant référencé. De plus, ce modèle a servi de format intermédiaire entre la représentation relationnelle et ontologique.

2.2.3 Représentation ontologique

2.2.3.1 Le modèle ontologique dans MOT[15]+ OWL[16]

Avant de parler du modèle de l'ontologie de SOCOM, nous rappelons la définition d'une ontologie. Une ontologie est un vocabulaire commun que se donnent des usagers ayant besoin de partager des informations dans un domaine. L'ontologie regroupe des définitions lisibles par une machine au moyen des concepts de base de ce domaine exprimés sous forme de classes, des relations entre ces classes et d'axiomes énonçant des propriétés de ces classes et de ces relations (Smith, Welty, & McGuinness, 2004). Dans notre cas, le domaine étudié est le développement des logiciels à base de composants et les informations que nous voulons partager sont les différents concepts de métadonnées des composants, leurs relations et leurs propriétés.

---

[15] MOT+ (Modélisation par Objets Typés): http://www.licef.teluq.uquebec.ca/realisations/
[16] OWL : Un langage d'ontologie web, http://www.w3.org/2004/OWL/

Pour comprendre le modèle OWL de SOCOM, nous expliquons rapidement quelques formes et notations graphiques des modèles ontologiques de l'outil MOT+OWL que nous avons utilisé. À l'origine, l'éditeur graphique MOT, développé au LICEF (Paquette, 1996,2002) est conçu pour concevoir des modèles pédagogiques et des modèles de connaissances de toutes sortes (Paquette *et al*., 2005). Récemment, l'éditeur a été spécialisé en un outil MOT+OWL (Paquette, 2007) permettant de construire graphiquement des ontologies selon le standard OWL (Ontology Web Language).

Cet éditeur utilise trois formes graphiques mises en relation par des liens typés. Les rectangles représentent les concepts ou classes, les hexagones représentent les propriétés entre ou relations entre les classes et les rectangles aux coins coupés représentent les individus ou instances des classes (voir l'exemple Figure 2.4). Divers types de liens sont utilisés entre ces trois types d'entités et des étiquettes sur les objets représentent les axiomes. Les types de liens incluent : le lien R entre une propriété et sa classe d'origine (domaine) ou cible (co-domaine), le lien S entre une classe et ses sous-classes, le lien DISJ indiquant que deux classes sont disjointes, le lien I entre une classe et une instance qui en fait partie, et un axiome indiquant qu'au moins deux individus du co-domaine sont associés par la propriété à chaque individu du domaine.

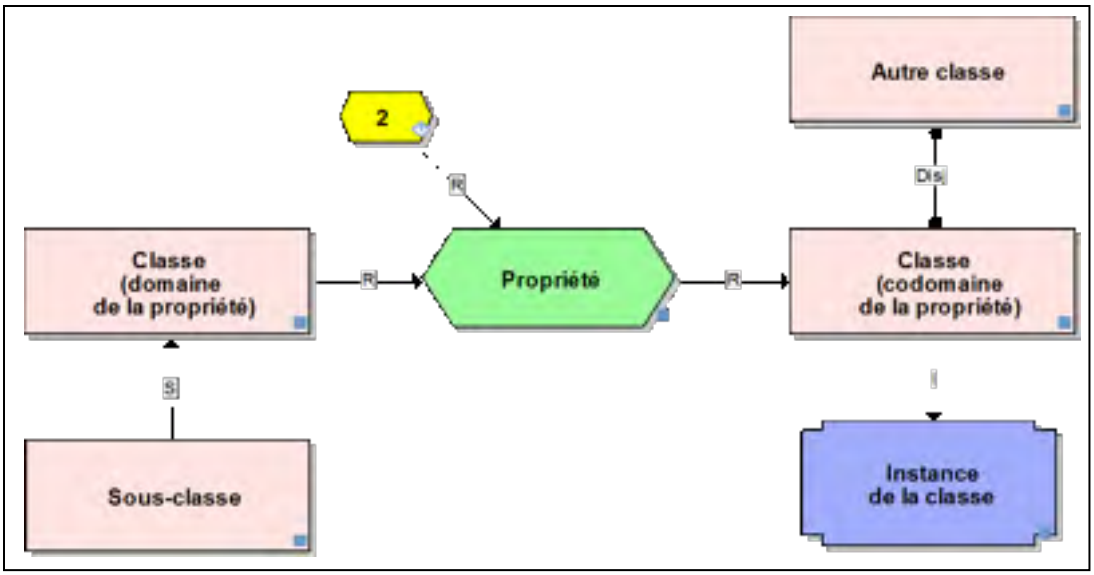


Figure 2.4 Les formes graphiques des objets et de certains liens dans MOT+OWL

2.2.3.2 Le modèle ontologique de la structure SOCOM

Pour le développement du modèle de l'ontologie de SOCOM, nous avons trouvé beaucoup d'outils tels que SWOOP, Protégé, WebOnto, etc. Quelques-uns offrent la modélisation graphique utilisant différentes formes géométriques, mais nous avons choisi MOT+OWL parce qu'il offre un formalisme graphique complet, simple et expressif pour la représentation des ontologies qui facilite l'activité de conception. Nous avons aussi utilisé l'outil Protégé à cause de ses interfaces usagers qui facilite notamment le peuplement de l'ontologie par des instances de classes. Nous ajoutons qu'il est bien documenté, soutenu et utilisé par une large communauté. Aussi Protégé offre la connexion via HTTP aux moteurs d'inférences pour fournir des déductions et tester aussi bien la cohérence que la validité de nos fichiers OWL. Avec cette connexion, Protégé transforme l'ontologie en format DIG, ce qui nous permet d'interroger l'ontologie à l'aide des requêtes qui utilisent le concept « asks ». En plus, l'utilisation de ces deux outils nous permet de gérer des ontologies de bout en bout, depuis la conception initiale jusqu'au raisonnement. Ceci justifie notre choix pour réaliser les travaux dans cet article avec des outils MOT+OWL et Protégé. Deux moteurs d'inférence compatibles avec les spécifications DIG et qui s'appuient sur la logique descriptive sont aussi

utilisés, à savoir Pellet et Fact++. Tous les détails sur notre utilisation de ces outils et des spécifications DIG sont explicités dans les sections 5 et 6.

2.2.3.3 Le modèle MOT+OWL de l'ontologie SOCOM

La Figure 2.5 montre le modèle de premier niveau de l'ontologie SOCOM. En fait, ce modèle reprend la description textuelle de la section 2.1. Le modèle de l'ontologie présente une description de SOCOM plus riche sémantiquement. Cette richesse provient du fait que le modèle comprend les noms explicites des relations logiques entre les composants de l'ontologie de SOCOM. En effet, le concept « SoftwareComponent » possède des propriétés telles que: « hasStaticSpecification », « hasServiceSpecification », « hasPlatformSpecification », « hasAggregationSpecification », « hasNonFunctionalSpecification », « hasBusinessSpecification », qui le lient par la relation R respectivement aux concepts « StaticSpecification », « ServiceSpecification », « PlatformSpecification », « NonFunctionalSpecification » et « BusinessSpecification ». S'y ajoute le concept appelé composant agrégat « AggregateComponent », qui se compose, au minimum, d'un composant logiciel. L'agrégat est lui aussi un composant logiciel et par conséquent, il est lié par un lien S au concept « SoftwareComponent ». Nous avons développé en MOT+OWL notre modèle ontologique de SOCOM, et nous explorons dans ce qui suit le sous-modèle de certains concepts et nous détaillons parfois le contenu par d'autres sous-modèles.

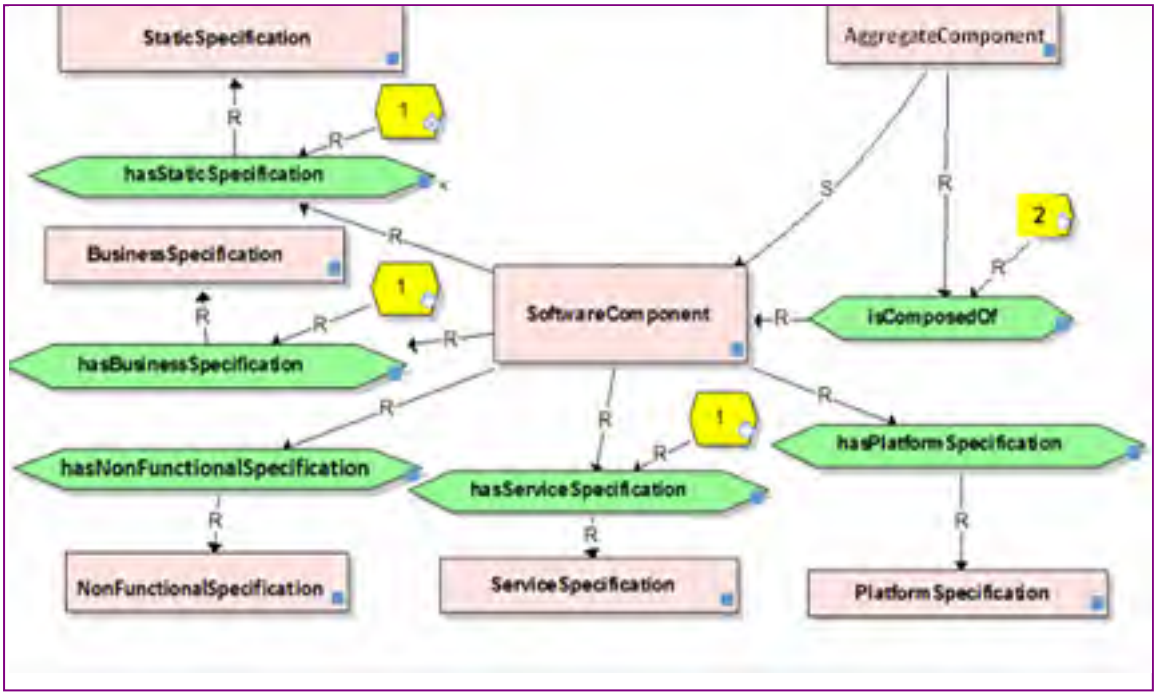


Figure 2.5 Le modèle ontologique de SOCOM (premier niveau)

La Figure 2.7 montre les différents concepts qui décrivent un service fourni ou requis par un composant. Ces concepts traduisent les attributs de service tels que le nom, le type (requis, fourni), le type de retour, la version, le mode d'invocation, la visibilité et les paramètres requis pour son invocation. Pour chacun de ces paramètres, nous décrivons son nom, son type, son ordre dans la liste des paramètres, sa visibilité et son type d'accès (Voir Figure 2.6).

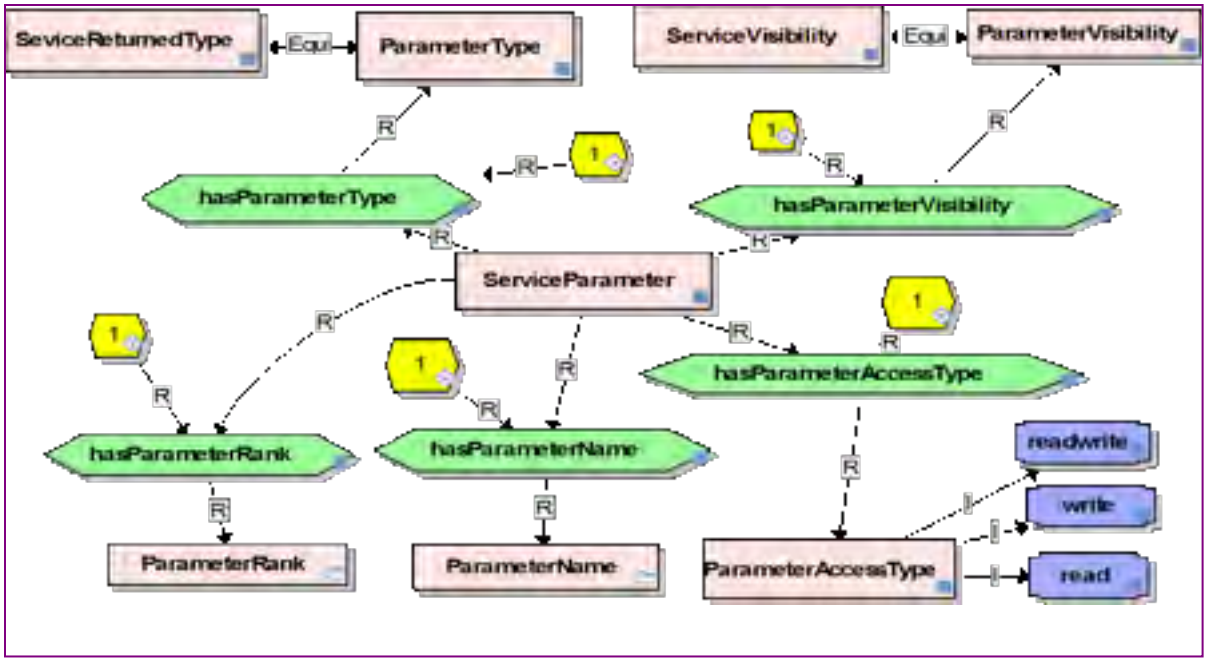


Figure 2.6 Le modèle ontologique des paramètres des services
des composants SOCOM (troisième niveau)

Nous précisons que le concept service est relié au concept des spécifications d'affaires « BusinessUseCase » par la relation « implements» (voir Figure 2.7). Le concept « BusinessUseCase » est une référence du sous modèle de l'ontologie des spécifications d'affaires « BusinessSpecification ». Cette relation est importante puisqu'elle nous permet d'ajouter une sémantique par-dessus la description syntaxique du service d'un composant, soit le cas d'utilisation d'affaire qu'il implémente. Moyennant cette relation, on peut avoir une connaissance de haut niveau sur le ou les cas d'affaire réalisés par le service du composant en question. Aussi, cette sémantique du service est utile pour un analyste qui recherche un composant particulier à des fins d'agrégations avec d'autres. Ceci nous aligne avec le paradigme de composant logique détaillé dans les travaux de Renaux *et al.* (Renaux, Olivier, & Jean-Marc, 2004).

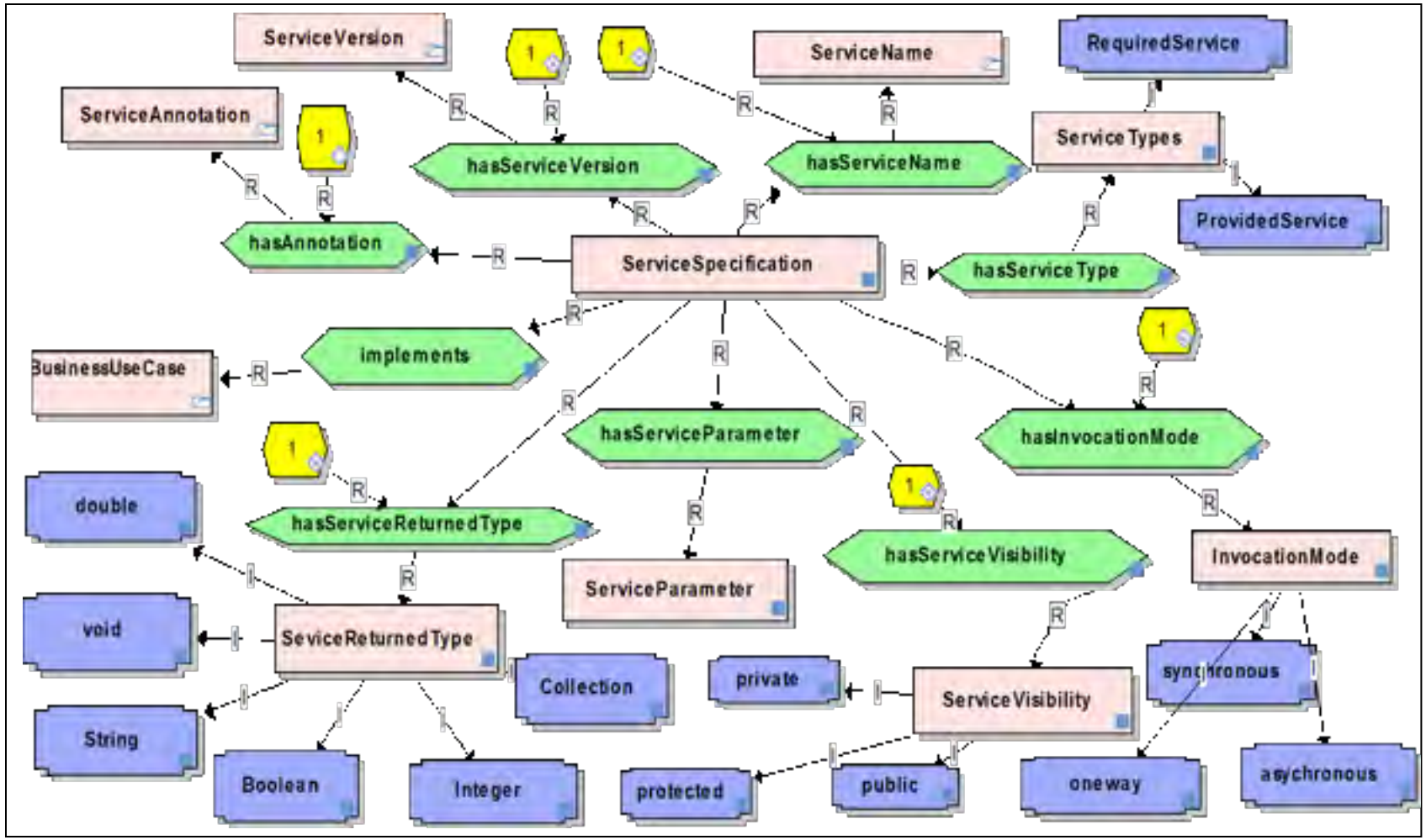


Figure 2.7 Le modèle ontologique des services du composant SOCOM (deuxième niveau)

2.2.3.4 Implémentation de l'ontologie de SOCOM

2.2.3.5 Les logiques descriptives

Notre ontologie de SOCOM utilise le langage OWL DL (Smith, Welty, & McGuinness, 2004). Nous avons choisi ce langage comme base pour un certain nombre de raisons. Il s'agit d'un des trois langages d'ontologie pour le Web recommandé par le W3C pour développer le Web sémantique. De ces trois langages, OWL DL est destiné aux utilisateurs qui demandent une expressivité maximale tout en retenant la complétude du calcul (toutes les inférences sont garanties calculables) et la décidabilité (tous les calculs s'achèveront dans un intervalle de temps fini).

Le langage OWL DL (DL signifiant « Description Logic ») a été conçu pour correspondre à une des logiques de description et fournir ainsi un langage offrant les propriétés de calcul nécessaires à des systèmes de raisonnement. Le logiciel graphique MOT+OWL que nous avons utilisé est fondé sur ce standard. Il permet d'exprimer complètement les primitives du standard OWL DL et de produire le fichier XML correspondant, lequel peut ensuite être réutilisé dans les traitements.

Les pierres d'assise des logiques de description sont les concepts, les propriétés et les individus. Les concepts décrivent les propriétés communes d'une collection d'individus. Ils peuvent être considérés comme des prédicats unaires qui sont interprétés comme un ensemble d'objets. Les propriétés sont des relations binaires entre les objets de deux classes. Des axiomes tels que « à tout individu d'une classe correspond au moins un individu d'une autre » ou « aucun individu n'appartient à la fois à la classe A et à la classe B » peuvent être ajoutés pour décrire un domaine de façon précise et guider un moteur d'inférence dans les déductions qui lui permettent d'exécuter une requête.

2.2.3.6 Les spécifications DIG

Les moteurs d'inférence qui s'appuient sur les logiques de description sont de plus en plus utilisés pour raisonner sur des ressources du Web sémantique, c'est-à-dire référencées au moyen d'ontologies. Pour permettre à un client d'interagir avec différents moteurs d'une manière standard, une interface standard et commune est souhaitable. Le groupe DIG œuvre dans la conception des systèmes compatibles avec les logiques de description. Une de ses activités consiste à développer une interface normalisée en XML pour les systèmes qui utilisent ces logiques en leur fournissant une API de base. L'interface DIG est une nouvelle norme qui permet l'accès à tout moteur d'inférence compatible via des interfaces Web utilisant le protocole HTTP. Cette interface est supportée par la majorité des moteurs d'inférence et facilite la construction des composants logiciels réutilisables. Des moteurs d'inférences comme Pellet, Racer et Fact++ sont compatibles avec les spécifications DIG.

Ces spécifications se composent essentiellement d'un schéma XML décrivant le formalisme du langage descriptif. Deux opérations sont disponibles pour construire et questionner une ontologie. L'ontologie est construite par l'opération « tells » et interrogée par l'opération « asks ». Pour plus de détails sur les spécifications DIG de version 1.1, voir (Bechhofer, 2003).

2.2.3.7 Un exemple de requête et de réponse DIG

Pour expliquer la logique des requêtes DIG, nous présentons les deux structures DIG les plus utilisées, soient « asks » et « responses ». Le script de la Figure 2.8 illustre une requête « asks » de type « relatedIndividuals » qui interroge l'ontologie présentée plus haut pour demander les individus des concepts qui sont liés à la propriété « hasServiceSpecification ». Autrement dit on cherche la liste des composants et leurs services intégrés comme des instances dans l'ontologie. Dans la requête, il faut spécifier l'identificateur de la requête « qTelosComponentServices » et la propriété « hasServiceSpecification » qui rattache les concepts « SoftwareComponent » et « ServiceSpecifcation ».

La réponse à cette requête est le fichier XML de la Figure 2.9. Ce dernier est retourné par le moteur d'inférence et montre les résultats trouvés sous la forme d'individus en langage ontologique ou bien les instances en langage objet. La liste retournée est un ensemble de paires. En effet, le premier individu de la paire est l'instance du concept source « SoftwareComponent » alors que le deuxième est l'instance du concept « ServiceSpecification». Le tout est encapsulé dans la balise « responses ». Cette réponse doit rappeler l'identificateur de la requête. On note que « TreeEditor » et « CPEModel » sont respectivement les noms des composants « Éditeur des activités » et « Content Properties Editor Model ». Le premier permet la construction de la structure arborescente des activités d'un cours ou d'un processus de travail et le deuxième permet l'édition du contenu des propriétés des activités de l'arborescence précédemment construite.

```
<asks xmlns="http://dl.kr.org/dig/2003/02/lang">
   <relatedIndividuals id="qTelosComponentServices">
        <ratom name="hasServices"/>
   </relatedIndividuals>
</asks>
```

Figure 2.8 Exemple d'une requête DIG utilisant le concept « asks »

```
<responses xmlns="http://dl.kr.org/dig/2003/02/lang">
  <individualPairSet id="qTelosComponentServices">
    <individualPair>
      <individual name="CPEModel"/>
      <individual name="getCPEModelService"/>
    </individualPair>
    <individualPair>
      <individual name="TreeEditor"/>
      <individual name="performTreeEditorService"/>
    </individualPair>
```

Figure 2.9 Exemple d'une réponse DIG utilisant le concept « responses »

2.2.3.8 Apport de la représentation ontologique de SOCOM

Le modèle ontologique de SOCOM est composé de plusieurs classes/concepts. La représentation ontologique de SOCOM a permis de relier les classes des métadonnées SOCOM entre elles avec des verbes significatifs. Ces relations permettent d'ajouter de la sémantique dans la description initiale d'un composants SOCOM. Par exemple, les cas d'affaires d'un composant logiciel et les services techniques sont reliés par le verbe « implements » Ainsi, en explorant une ontologie, il est possible d'énumérer les services qui implémentent un cas d'affaires par un simple clic sur le verbe dans « implements » associé à l'instance du composant de l'ontologie en question (voir Figure 2.7).

En général, la navigation dans l'ontologie SOCOM permet d'identifier les composants référencés ainsi que les relations d'agrégation entre eux si des agrégations ont été développées. Ces informations d'agrégation permettent à l'organisation d'avoir des statistiques sur les composants agrégats et agrégés ainsi que la fréquence de réutilisation des composants.

2.2.4 Comparaison entre les différentes représentations

Nous n'avons pas testé l'interrogation du fichier XML contenant la liste des composants. Ceci s'explique par le fait que le format XML des spécifications SOCOM est plus un format d'échange des fiches des composants entre les entrepôts des composants et non de recherche en soi. Ainsi notre comparaison portera sur les deux autres représentations, relationnelle et ontologique.

Nous avons testé la manipulation des deux formats de SOCOM et nous avons remarqué une simplicité et une complexité pour chacun des modèles relationnel et ontologique. Pour réaliser une comparaison entre les deux implantations, nous nous sommes basés sur une requête client et nous présentons les réponses à la même requête avec les deux formats. Voici l'énoncé de la requête :

« Nous recherchons un composant type boîte noire qui a une licence ‘LGPL’. Il doit être implémenté dans le langage Java avec un faible potentiel de couplage. Ce composant doit fournir un service qui retourne les ressources associées à un objet d’apprentissage ‘LOM’ ».

En langage SQL, cette recherche doit être formulée à l’aide de quatre requêtes appelées successivement comme suit :

```
R1: <SELECT componentId, componentName FROM socom_technical WHERE componentCouplingLevel
= “loosely” AND languageType = “Java”>

R2: <SELECT componentId FROM socom_nonfunctional WHERE licenseName = “LGPL” AND
nonFunctionalType = “Licensing” AND componentId=”cptID1”>

R3: <SELECT serviceId FROM component_services WHERE serviceName = “%LOM%” And
componentId=”cptId1”>

R4: <SELECT * FROM service_parameters WHERE serviceId = “Id-2121-Service” AND
componentId=”cptId1”>
```

Figure 2.10 La séquence de requêtes SQL qui répond au besoin de l’exemple

Après l’exécution de chaque sous requête, nous devons récupérer la valeur de la variable « componentId » pour l’utiliser dans la suivante comme critère de sélection. Ce traitement additionnel doit se faire à l’aide d’un langage hôte tel que Java qui utilise un connecteur spécifique à notre base de données.

Pour formuler la même requête en DIG, nous utilisons les deux commandes du concept « asks »: le concept « instances », pour chercher l’instance du concept, et le concept « stringequals », qui retourne les individus ayant les propriétés recherchées. En effet, la première requête sera appliquée au concept « SoftwareComponents », ce qui traduit notre besoin de recherche d’un composant qui satisfait les critères énoncés dans la deuxième requête. Celle-ci indiquera les relations qui doivent être validées avec les valeurs des instances des classes cibles. Dans notre cas, les relations concernées (propriétés) et les

instances (individus) dans l'ordre sont : (« hasLicenseType », LGPL), (« hasAccessibilityDegree », BlackBox), (« hasLanguageType », Java), (« hasCouplingLevel », loosely) et (« hasServiceName », « %LOM% »). La syntaxe est expliquée dans le document des spécifications DIG 1.1.

```
<asks xmlns="http://dl.kr.org/dig/2003/02/lang">
<instances id="queryGetSpecificComponent">
           <stringequals val="LGPL">
               <ratom name=" hasLicenseType"/>
           </stringequals>
           <stringequals val="BlackBox">
               <ratom name=" hasAccessibilityDegree"/>
           </stringequals>
           <stringequals val="Java">
               <ratom name=" hasLanguageType"/>
           </stringequals>
           <stringequals val="loosely">
               <ratom name="hasCouplingLevel "/>
           </stringequals>
           <stringequals val="%LOM%">
               <ratom name="hasServiceName "/>
```

Figure 2.11 La requête DIG pour faire la recherche du composant de l'énoncé

On peut constater que nous avons construit la requête (
Figure 2.11) en traduisant simplement la description de l'énoncé ci-dessus. Cette simplicité dans la formulation de la requête DIG découle de l'expressivité de la syntaxe des requêtes DIG et de la classification des concepts et des relations de l'ontologie technique de SOCOM. En effet, cette syntaxe englobe mieux la sémantique de l'énoncé de l'exemple. En plus, les résultats de recherche peuvent alimenter l'ontologie avec le concept « tells » pour enrichir l'ontologie des composants SOCOM avec de nouvelles connaissances. En d'autres termes, la force du modèle ontologique réside dans sa structure qui est conforme à la logique de description et dans son couplage avec les moteurs d'inférence pour servir du raisonnement via des requêtes intelligentes. Toutefois, il est important de rappeler que notre logique de recherche est partagée entre l'ontologie technique de SOCOM et le langage de requête DIG.

Ainsi, on note un des avantages des ontologies vis-à-vis des bases de données, soit l'expressivité et la simplicité d'implémenter les requêtes depuis une recherche exprimée en langage naturel. À ce niveau, on parle de deux dimensions de la comparaison soient, la **structure conceptuelle** et le **langage de requête** des bases de données vis-à-vis ceux des ontologies. Cette simplicité pourrait être obtenue avec une base de données, mais au prix du développement d'un programme avec un langage hôte qui implémente les structures des propriétés et des axiomes exprimés dans l'ontologie technique de SOCOM.

Les individus de l'ontologie SOCOM sont stockés dans un fichier OWL conforme à une ontologie technique épousant le standard OWL-DL. On rappelle que les individus de l'ontologie sont équivalents aux données d'une base de données. Aussi, on appelle la structure et les individus de l'ontologie de SOCOM respectivement ontologie technique et base de connaissances. Elles sont sauvegardées dans deux fichiers OWL distincts. Toutefois, l'ontologie des instances fait appel à l'ontologie technique pour valider la structure des données. Les deux fichiers OWL sont hébergés physiquement sur un serveur Web. Pour accéder à ces ontologies, il suffit de connaitre leur adresse web.

La base de données relationnelle de SOCOM est implémentée dans un SGBDR. Pour y accéder du point de vue applicatif, il faut un programme qui a une connexion à une base de données avec les droits associés. Il est évident que pour réutiliser cette base de données dans un autre contexte indépendant, il faut avoir un SGBDR équivalent et les scripts de création de la structure et des données pour reconstruire l'environnement des données et pouvoir l'exploiter. Pour ce qui est de la réutilisation de l'ontologie, il faut connaitre seulement l'adresse de l'ontologie technique. N'importe lequel des outils d'ontologie peut pointer sur cette ontologie technique, ce qui permettra à son utilisateur de peupler l'ontologie des instances. Et pour une réutilisation dans un réseau privé, il suffit de copier l'ontologie technique et celle des instances dans un serveur Web hébergé dans le réseau en question. Ainsi, on note que les ontologies sont plus avantageuses pour la **portabilité** dans différentes plateformes et la **simplicité de réutilisation**.

Quant à l'évolution, il est aujourd'hui difficile de gérer les changements suite à l'évolution d'une ontologie. En pratique, lorsqu'on a changé notre ontologie technique, nous avons trouvé beaucoup de difficultés pour mettre à jour l'ontologie des instances. On n'a pas pu réutiliser les instances de l'ancienne ontologie technique. De ce fait, on a utilisé l'outil Protégé pour peupler de nouveau, à la main, l'ontologie des instances conformément à la nouvelle ontologie technique. Avec le SGBDR MySQL, le changement des attributs des tables de la base de données de SOCOM est plus facile. Pour remédier à ce problème, on a synchronisé les deux implémentations ontologique et relationnelle par un convertisseur de donnée. Ce dernier traduit les données depuis la base de données SOCOM en fichier OWL qui représente l'ontologie des instances, tout en gardant la conformité avec l'ontologie technique SOCOM. En fait, il y a encore peu de travaux qui traitent de l'évolution des ontologies; on note la proposition d'une méthode et d'outils pour maintenir la cohérence du référencement des ressources pédagogiques, lors de l'évolution d'une ontologie (Rogozan & Paquette, 2005). En attendant la conclusion de ces travaux, il nous faut noter un avantage des bases de données relationnelles vis-à-vis des ontologies en ce qui concerne les dimensions de la gestion du changement et l'évolution de la structure des métadonnées des composants SOCOM. On conclue que les dimensions **d'évolution et de gestion des changements** sont pour le moment plus simples à gérer avec les bases de données qu'avec les ontologies.

En exécution, le problème de performance en présence d'ontologies de grande taille et le temps de réponse des moteurs d'inférence vis-à-vis des requêtes complexes demeurent les inconvénients majeurs d'une telle approche. Notre expérience nous montre une simplicité de développement de l'application Java avec une base de données relationnelle sous le SGBDR MySQL et un meilleur temps de réponse pour des requêtes simples. Par contre, l'exemple précédent nous montre que l'équivalent d'une requête simple DIG est une structure complexe lorsque réalisée en SQL. En effet, le langage SQL qui interroge la base de données SOCOM nous fournit peu de sémantique sur les composants logiciels, leur classification et la classification de leurs agrégations possibles.

En résumé, le tableau ci-dessous présente le récapitulatif de la comparaison de l'implémentation de SOCOM sous la forme d'une base de données et sous la forme d'une ontologie. On rappelle que ces implémentations ont pour objectif d'aider à la recherche des composants logiciels dans une banque de composants décris avec la structure SOCOM.

Tableau 2.1 Tableau comparatif des ontologies vs base de données

Dimensions / Implémentation de SOCOM	**Base de données**	**Ontologie**
Portabilité sur différents plateforme de stockage	+/-	+
Évolution (Gestion du changement)	+	-
Simplicité de stockage	-	+
Performance	+	-
Expressivité du Langage de requête	-	+
Simplicité du langage de requête	+/-	+
Gestion des droits d'accès et privilèges d'accès	+	-
Simplicité de réutilisation	-	+

Tous les résultats de la comparaison présentés ci-dessus montrent des avantages et des inconvénients des deux implémentations. Cependant, les avantages de l'implémentation ontologique auraient pu être obtenus par l'implémentation relationnelle sauf qu'il nous aurait fallu développer toute la logique descriptive que les moteurs d'inférence nous offrent. Par conséquent, nous adoptons une solution mixte. En effet, cette solution utilise conjointement le modèle ontologique et relationnel des métadonnées SOCOM. Un convertisseur de données a été développé pour gérer l'évolution de la structure des métadonnées des composants ainsi que pour la synchronisation des données entre la base de données et les instances de l'ontologie.

2.3 Gestionnaire des composants logiciels SOCOM

Le gestionnaire des composants logiciels SOCOM est une application Web développé en Java pour le référencement et la gestion des spécifications SOCOM. L'application offre essentiellement les fonctionnalités suivantes :

- référencement des composants SOCOM;
- gestion des fiches des composants SOCOM;
- recherche des composants SOCOM;
- extraction des métadonnées de services du code binaire des composants;
- exploration des données des fiches SOCOM à partir de la base de données en document XML;
- conversion du document XML des composants SOCOM en ontologie OWL.

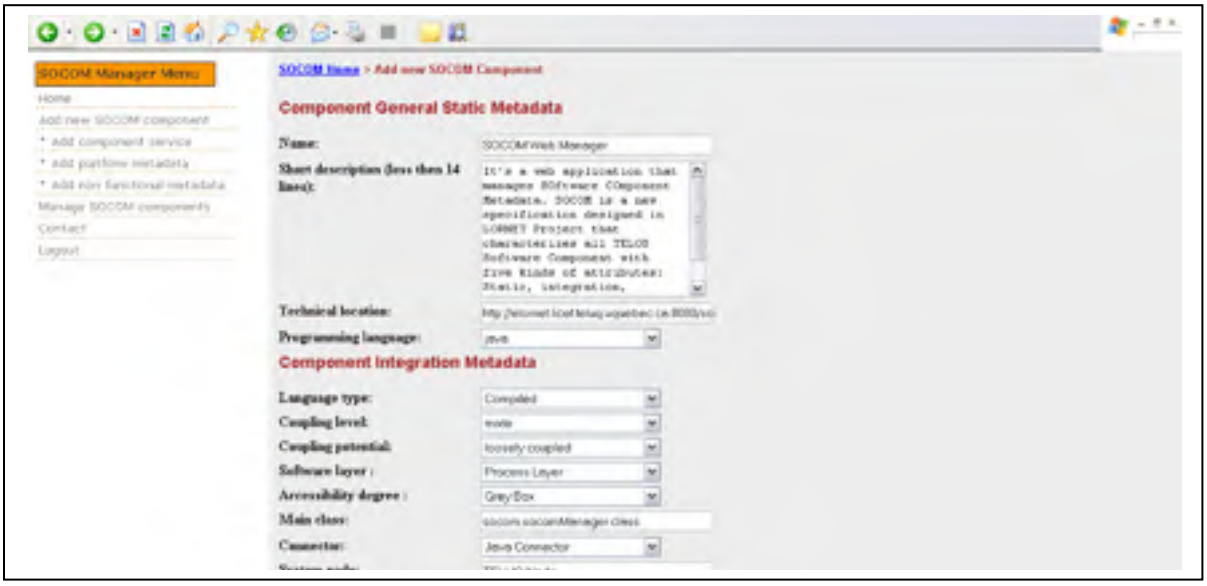


Figure 2.12 Écran de référencement d'un nouveau composant logiciel SOCOM

2.4 Recherche des composants logiciels à l'aide des attributs SOCOM

L'application « Gestionnaire des composants SOCOM » offre des interfaces graphiques aux utilisateurs pour la recherche des composants référencés à l'aide des spécifications SOCOM.

2.4.1 Recherche simple

Figure 2.13 Écran de recherche simple de composants logiciels SOCOM

La recherche simple des composants porte essentiellement sur les attributs statiques des composants. Avec ce type de recherche, l'utilisateur peut explorer rapidement l'inventaire des composants à l'aide de quelques mots clés associés au nom du composant, sa description ou son langage de programmation. Les résultats de recherche sont affichés avec le nom de chaque composant et une description sommaire (Figure 2.14). Un ensemble d'icônes est affiché à droite de chaque description sommaire pour pouvoir gérer ses métadonnées. Le nom est cliquable pour afficher la description détaillée du composant en question (Figure 2.15).

Figure 2.14 Résultat de la recherche simple de composants programmés en "Java"

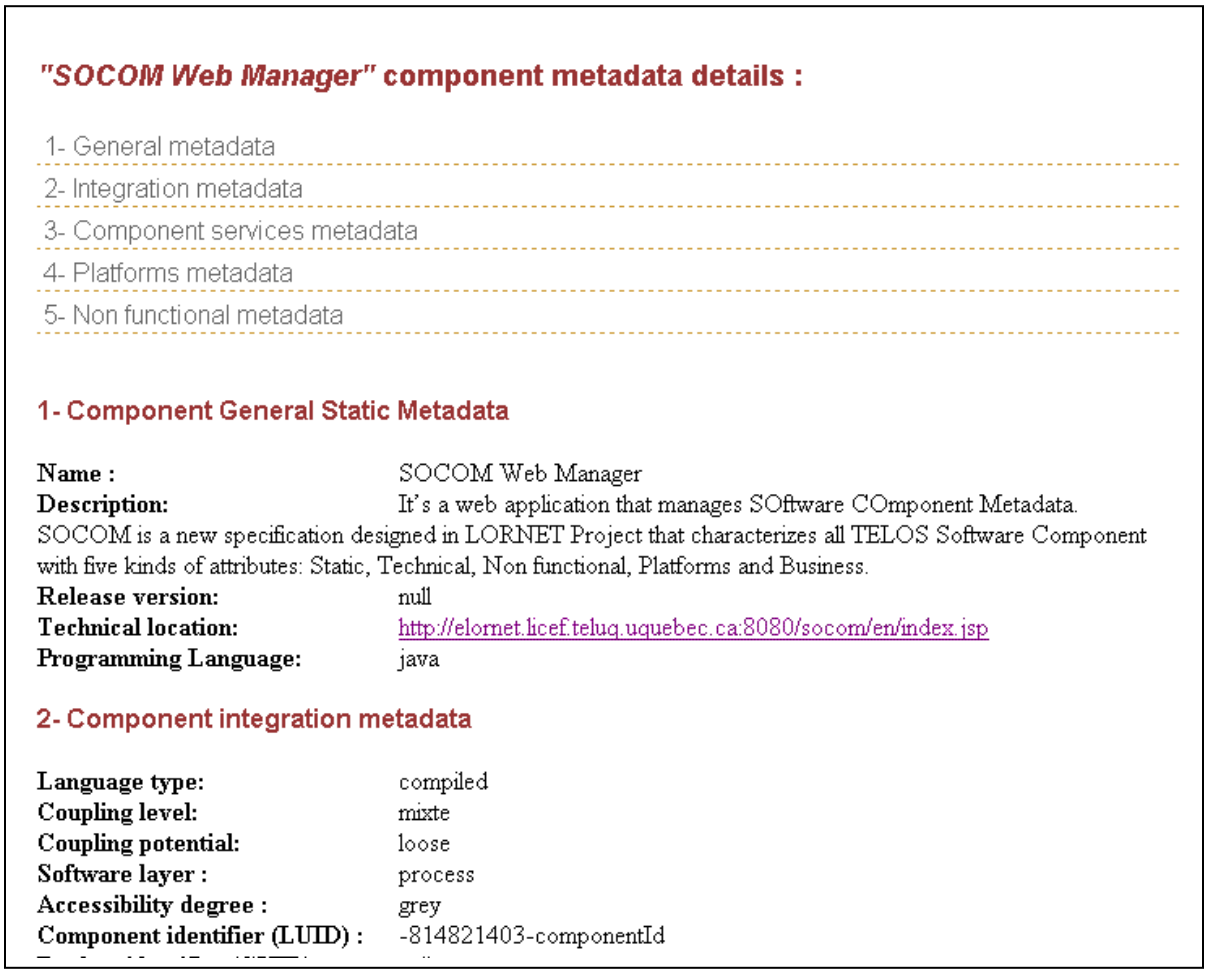


Figure 2.15 Fiche détaillée d'un composant SOCOM

2.4.2 Recherche avancée

Figure 2.16 Écran de l'interface utilisateur pour la recherche avancée des composants logiciels SOCOM

La recherche avancée des composants porte sur la majorité des attributs de SOCOM dans les différentes catégories. Avec ce type de recherche, l'utilisateur ayant une description avancée

du composant recherché peut utiliser une combinaison de mots clés sous forme de conjonctions et de disjonction de critère de recherche.

Figure 2.17 Résultat de la recherche avancée des composants logiciels

2.4.3 Recherche par ontologie

La Figure 2.18 montre un aperçu des instances des spécifications d'agrégation d'un composant de l'ontologie SOCOM décrites dans la section 2.2.3. En effet, les classes de l'ontologie SOCOM occupent le côté gauche de la figure. Les instances des classes sont affichées au milieu. À droite, nous retrouvons les valeurs des propriétés de l'instance sélectionnée dans la colonne du milieu. Par exemple, on référence une trentaine de composants de la banque en construction réunie dans TELOS.

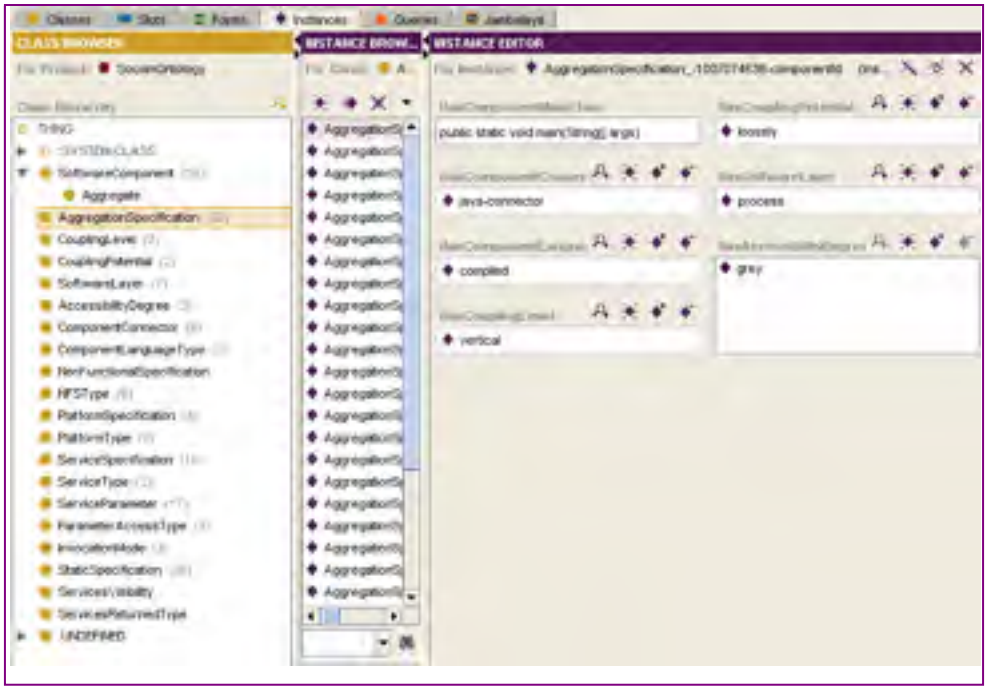


Figure 2.18 Une partie de l'ontologie SOCOM sous Protégé OWL

« SoftwareComponent» est une super-classe de « Aggregate ». De ce fait, toutes les instances de la classe « Aggregate » se trouvent avec les propriétés de la classe « SoftwareComponent » parce qu'un agrégat est aussi un composant logiciel. Les propriétés de la classe « Aggregate » sont des propriétés inférées qui sont héritées de la classe « SoftwareComponent ». Aussi, les instances de la classe « Aggregate » se retrouvent dans l'ensemble des instances de la classe « SoftwareComponent ». Elles s'identifient comme des instances inférées de la classe « Aggregate ».

2.5 Conclusion

Ce chapitre a présenté les résultats des travaux sur les spécifications des composants logiciels et les implantations possibles. Les spécifications SOCOM viennent traiter les aspects suivants du développement basé sur les composants: la caractérisation, la classification et la recherche des composants logiciels pour des fins d'agrégation. En fait, nous avons commencé par la définition de la structure SOCOM pour décrire les composants logiciels en

définissant des spécifications statiques, des services, non fonctionnelles, des plateformes, d'agrégation et métiers des composants logiciels. Nous avons développé notre modèle ontologique de SOCOM à l'aide de MOT+OWL. Nous avons utilisé Protégé pour peupler et gérer notre ontologie. Pour promouvoir la recherche des composants logiciels, nous avons adopté une solution mixte dans laquelle nous avons défini un modèle ontologique de SOCOM synchronisé avec un modèle relationnel moyennant un convertisseur de données que nous avons développé. Nous avons justifié ce choix mixte par une comparaison entre une implémentation en base de données relationnelle et une implémentation ontologique.

Il importe de rappeler à ce niveau que SOCOM est un élément de base du processus d'agrégation dirigé par les métadonnées et les modèles. SOCOM est un élément de base du volet métadonnées qui dirige la définition de notre processus d'agrégation. Ce dernier est détaillé dans les chapitres 4 et 5.

Le tableau suivant décrit l'apport de SOCOM, comme résultat de recherche, aux phases d'un processus de développement à base de composants.

Tableau 2.2 Matrice des apports de SOCOM aux phases du processus de développement à base de composants.

Résultats de la thèse	**Description des contributions des résultats par phase**
1. Métadonnées SOCOM	**Phase de recherche :** - SOCOM offre plusieurs vues d'un composant avec les différentes catégories de ses métadonnées, il aide à la démocratisation des composants dans l'entreprise. - SOCOM vise un public cible élargi et hétérogène pour découvrir les composants réutilisables et explorer les composants selon leur profil (affaires, gestion, technique, etc.). - SOCOM permet des recherches avancées avec des conjonctions ou des disjonctions de critères en utilisant la diversité de ses métadonnées. - SOCOM permet de créer une arborescence des composants basés sur les différents attributs de ses métadonnées. (p. ex : exploration de tous les composants utilisant la plateforme JEE, exploration de tous les composants déployés dans un serveur d'application TOMCAT, etc.). **Phase de sélection :** - Les métadonnées couvrent plusieurs aspects du composant, les utilisateurs disposent de plus amples informations, ce qui aide à la décision au moment de sa sélection. **Phase de faisabilité :** - SOCOM fournit les données de base pour évaluer la première itération de l'étude de la faisabilité d'agrégation des composants sélectionnés. **Phase d'intégration:** - SOCOM fournit à l'équipe technique les métadonnées des services et les métadonnées non fonctionnelles qui sont utiles pour les étapes de codage et des tests au début de la phase d'intégration.

CHAPITRE 3

CLASSIFICATION DES AGRÉGATIONS DES COMPOSANTS

Après la présentation des métadonnées SOCOM, ce chapitre propose des métadonnées autour des aspects d'agrégation des composants. Ces métadonnées d'agrégation enrichissent les spécifications SOCOM afin d'offrir davantage de couverture pour la description d'un composant logiciel. D'après la revue de la littérature de la section 1.3, peu d'auteurs proposent des classifications des agrégations durant la phase d'analyse au début du processus de développement à base de composants. Ces auteurs définissent tous un modèle d'agrégation au niveau de la conception ou du codage en se basant sur des méta-modèles (Cheesman & Daniels, 2001). D'autres proposent des approches moyennant des transformations des modèles entre les phases de développement (Renaux, Olivier, & Jean-Marc, 2004) ou en classifiant les agrégations avec le type de boîte du composant, p. ex. boîte blanche, boîte grise ou boîte noire (Villalobos, 2003).

Nous pensons que la classification des agrégations peut se faire tôt dans le processus de développement à base de composant, la phase d'analyse. Elle peut être définie à l'aide d'un ensemble d'attributs d'agrégation qui décrivent les types d'interaction d'un composant dans un contexte d'agrégation. Le but ultime de cette classification est de proposer des modèles d'agrégation durant la phase d'analyse afin d'assister un analyste/intégrateur ou un architecte dans le processus d'agrégation des composants logiciels. Autrement dit, lorsque les composants potentiels sont sélectionnés, cette classification et les modèles associés sont utilisés pour décider quelle classe d'agrégation peut être appliquée. Pour définir ces classes, nous avons sélectionné quelques attributs de métadonnées d'agrégation pour ainsi qualifier l'agrégation des composants de connexion, collection, coordination ou fusion.

Cette taxonomie d'agrégation des composants est élaborée dans la quatrième étape de la démarche de note méthodologie de recherche. Le chapitre précédent a défini des métadonnées qui ont permis une spécification étendue et une classification des composants

logiciels. Ce chapitre décrit le volet classification des agrégations et présente les modèles des agrégats qui sont utilisés dans le processus d'agrégation dirigé par les métadonnées et les modèles. Ce dernier processus est détaillé dans le chapitre 4. La figure suivante positionne le présent chapitre dans le cheminement global de la thèse.

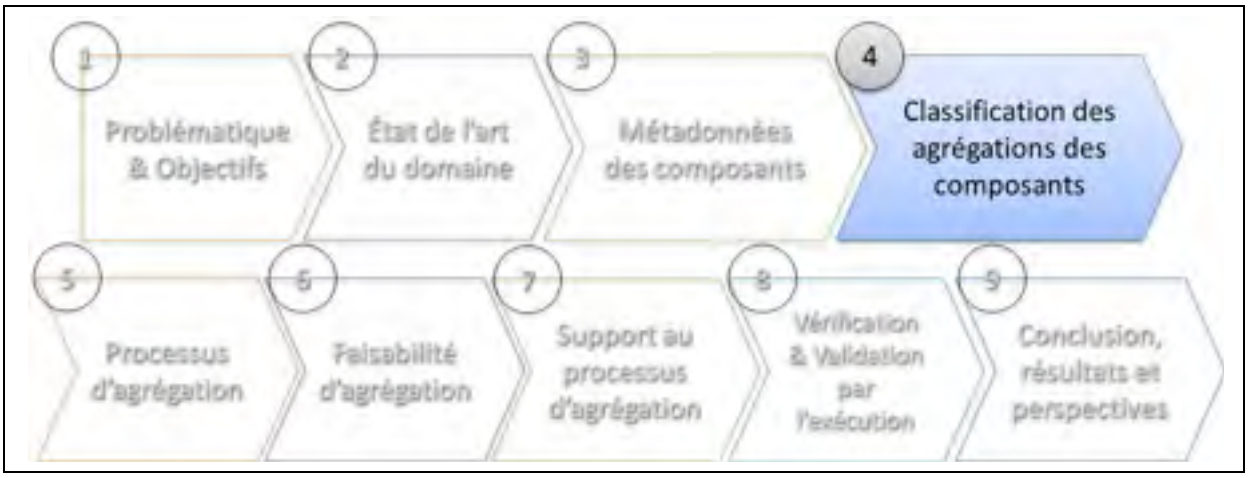


Figure 3.1 La méthodologie de recherche de la thèse – Étape 4 :
Classification des agrégations des composants

3.1 Le besoin en attributs d'agrégation

Pour aider à catégoriser les agrégations des composants logiciels, nous avons besoin d'identifier les caractéristiques d'agrégation de chacun d'eux, puis des évaluations et des comparaisons doivent être effectuées entre leurs valeurs pour s'assurer de leur compatibilité. Ces caractéristiques aident à décider du type d'agrégation auquel un composant peut participer. À son tour, l'identification du type d'agrégation et les patrons de conception associés facilitent l'adaptation des composants de même que la conception et le codage de l'agrégat. Pour définir ces attributs, nous avons réutilisé quelques caractéristiques de l'agrégation des composants à partir des sources décrites dans les sections 1.2 et 1.9. Leur regroupement sous la catégorie « agrégation » est une idée inédite de l'auteur pour servir dans la classification des agrégations qui est aussi réutilisée dans l'évaluation de la faisabilité d'agrégation (CHAPITRE 5).

Nous prenons pour hypothèse que les composants participants dans les agrégations étudiées suivent une architecture par couche. Ainsi, nous avons besoin de savoir à quelle couche appartient un composant logiciel, d'où le besoin d'un attribut « ***SoftwareLayer*** » (ex. dans une architecture à quatre couches, un composant peut appartenir à la couche de traitement (T), de présentation (P), de middleware (M) ou des données (D)).Le composant peut implanter plus d'une couche logicielle. Toutefois, les valeurs de cet attribut doivent mentionner l'ensemble des couches logicielles couvertes par le composant. Par exemple, la figure suivante montre un composant « Annotateur Émotif-AÉ » qui englobe quatre sous composants appartenant aux quatre couches telles que P, T, M et D, ainsi nous attribuons les valeurs P, T, M et D pour l'attribut « SoftwareLayer » du composant en question. L'analyse de la classification se fait entre les composants de chacune des couches.

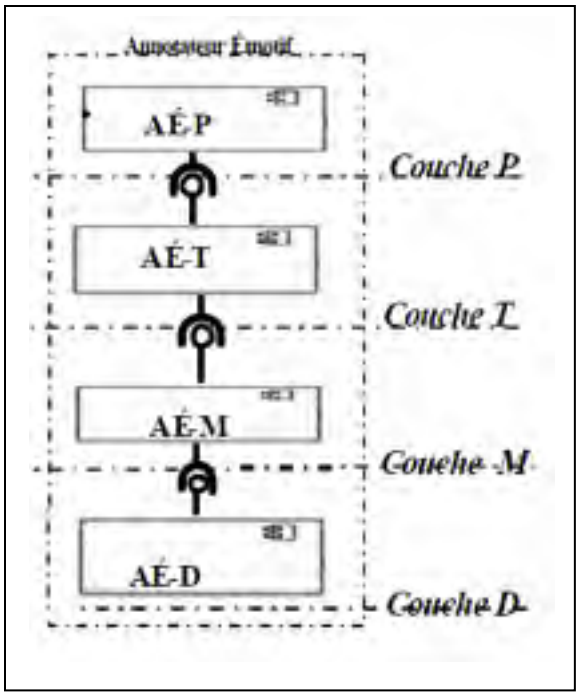


Figure 3.2 Exemples des valeurs de l'attribut
"Software Layer" : P, T, M et D

Le style d'architecture par couche et l'attribut qui rappelle les différentes couches de chaque composant logiciel nous aident à évaluer la faisabilité d'une agrégation entre plusieurs composants. Par exemple, même si un composant de présentation et un composant de

données peuvent communiquer techniquement, l'agrégation n'est pas faisable, étant donné qu'une architecture en couche stricte n'admet que les interactions entre des composants d'une même couche ou avec des composants de la couche immédiatement inférieure. C'est la règle de respect de la hiérarchie des couches logicielles. Le choix de quatre couches n'est pas obligatoire pour cet attribut. Dans d'autres contextes, d'autres couches peuvent être ajoutées ou éliminées selon les besoins. Dans le même contexte d'interaction au niveau des couches, nous définissions l'attribut potentiel de couplage « ***couplingLevel*** » qui peut avoir les valeurs suivantes : horizontal, vertical ou mixte. Le potentiel de couplage est un nouveau concept que nous avons introduit pour informer sur les possibilités de couplage du composant dans une architecture en couches. Autrement dit, nous voulons informer si le composant peut interagir avec des composants de la même couche (couplage horizontal) ou de la couche immédiatement inférieure (couplage vertical) ou les deux (couplage mixte). Ceci nous permet de décider s'il peut participer dans une agrégation horizontale, verticale ou mixte. La valeur de cet attribut est utile dans l'évaluation de la faisabilité de l'agrégation. Cette valeur est définie lorsque nous évaluons les interactions des services d'un composant.

Autrement dit, nous analysons les services offerts et requis par le composant à partir des métadonnées de services et nous précisons les interactions potentielles auxquelles peut participer le service du composant. Par exemple, un composant de traitement requiert un service pour lire les données d'une base de données. Ce service entre dans une interaction verticale avec un service d'un composant appartenant à la couche d'accès aux données ou d'intergiciel. Dans ce cas, l'attribut « ***couplingPotential*** » de ce composant de traitement prend la valeur « vertical ».

Nous constatons que le potentiel d'agrégation d'un composant avec un autre dépend aussi du niveau d'accessibilité au code « ***accessibilityDegree*** » du composant pour s'assurer du potentiel de réutilisation du composant. En d'autres termes, un composant de type boîte noire n'est pas réutilisé de la même façon qu'un composant de type boîte grise ou de type boîte blanche (à code ouvert).

Chaque composant logiciel fournit à son environnement un ou plusieurs points d'entrée « ***entryPoint*** » qui permettent aux autres composants de consommer ses services. Cet attribut est utile pour les concepteurs et le développeur/intégrateur afin d'explorer la définition des services et d'identifier ceux qui seront utilisés dans l'agrégation en cours d'analyse. Le point d'entrée peut être l'URL d'un fichier WSDL, une interface programmable ou un chemin qui mène à la classe qui expose les méthodes d'un composant OO.

Lorsque le composant participe dans un processus d'agrégation, nous voulons garder une référence de sa dépendance avec le composant agrégat, d'où l'attribut « ***aggregateComponentReference*** ». Cette information permet d'identifier les dépendances du composant envers le composant agrégat. L'architecte peut s'informer de l'utilité d'un composant en énumérant ses dépendances.

Aussi, la compatibilité du composant avec les normes et les standards de l'industrie nous aide à décider de la faisabilité de l'agrégation. Cet attribut s'intitule « ***standardsCompliancy*** ». En effet, si les composants de présentation dans deux technologies différentes de type portail implémentent la technologie des portlets en appliquant le standard WSRP[17], les deux composants peuvent s'intégrer au niveau la couche de présentation en réutilisant les portlets sous forme de service web. L'analyse de ces attributs concerne des acteurs techniques qui connaissent suffisamment le domaine d'intégration du composant tels qu'un architecte et un développeur/intégrateur.

3.2 Spécification des métadonnées d'agrégation

Avec cette catégorie de métadonnées, nous voulons enrichir les spécifications SOCOM pour informer sur les caractéristiques du composant pouvant servir dans des projets d'agrégations. Avec cette section, SOCOM compte six catégories de métadonnées. Chaque composant est caractérisé par une liste d'attributs d'agrégation. Le nom technique de cette section des

[17] WSRP : Web Service Remote Portlet, Standard Oasis

métadonnées d'agrégation de SOCOM est **<aggregationSpecification>[1]**. Dans ce qui suit, on détaille les attributs d'agrégation :

- Degré d'accessibilité **<accessibilityDegree>[1]**: il s'agit d'une vision du composant sous la forme de « boîte ». On distingue un composant boîte noire, blanche et grise (Haines et al, 1997), tel que présenté dans la section 1.5.
- Potentiel de couplage <couplingPotential>[1]: c'est le niveau de couplage d'un composant avec d'autres sur une même couche logicielle ou des couches différentes (le niveau de couplage peut être horizontal, vertical ou mixte). Les composants logiciels communiquent entre des couches du haut vers le bas en respectant la hiérarchie. Autrement dit, le composant de la présentation ne consomme que les services de la couche de traitement ou d'un autre composant de présentation. Le raisonnement est le même pour les autres couches.
- Couche logicielle **<softwareLayer>[1..N]**: c'est la couche logicielle dans laquelle le composant opère et collabore avec d'autres pour assurer ses responsabilités. Nous supposons que la couche logicielle est une combinaison logique parmi les quatre couches suivantes: présentation (P), traitement (T), middleware (M) et données (D).
- Type de langage du composant **<languageType>[1..2]**: un composant logiciel peut être programmé avec un langage compilé ou interprété. Cette information est utile pour connaitre les besoins du composant à l'exécution (interpréteur, compilateur, machine virtuelle, etc). Cet attribut nous permet de vérifier la compatibilité entre les langages des composants à agréger.
- Point d'entrée du composant **<entryPoint>[1]**: il fournit à un ingénieur ou un système, le chemin relatif du point d'entrée du composant afin de l'exécuter. Cet attribut sera utile pour l'architecte et l'intégrateur lors de la conception et le codage de l'agrégat.
- Type d'interaction du composant **<interactionMode>[1]**: un composant peut avoir besoin d'une interaction de type connecteur, pont, médiateur, etc. (Mili, Mili, Yacoub, & Addy, 2002). Ceci permet au composant dans un contexte donné de communiquer avec des systèmes spécifiques par la technologie des connecteurs. Par exemple, le noyau de TELOS communique avec les composants qu'il agrège par un ensemble de connecteurs

implémentés dans différents langages. Il sera aussi utile dans l'évaluation de la faisabilité de l'agrégation ainsi que lors de sa conception.

- Référence du composant agrégat**<aggregateComponentReference>[1..*]**: lorsque le composant participe dans des processus d'agrégation, cet attribut rappelle le ou les références des agrégats résultants.
- Compatibilité avec les standards **<standardCompliancy>[1..*]**: il s'agit des normes, cadres de travail (frameworks) et standards respectés ou utilisés par le composant.

Pour mieux comprendre ces attributs, nous proposons un exemple des métadonnées d'agrégation du composant « Portail VDM » - présenté dans l'ANNEXE I - participant dans le développement du composant agrégat « Portail Loisir En Ligne – VDM ».

Tableau 3.1 Exemple des métadonnées d'agrégation des spécifications SOCOM

Métadonnées d'agrégation	**Valeur de l'attribut**
Potentiel de couplage	Mixte
Degré d'accessibilité	Boîte grise
Couche logicielle	Présentation-Traitement – Données
Nœud du système	ville.montreal.qc.ca
Point d'entrée du composant	www.ville.montreal.qc.ca
Mode d'interaction	Portlet Oracle/ Connecteur JAVA/ADF
Type de langage	Compilé
Compatibilité avec les standards	JEE, Spring, PL-SQL, JDK 1.4, Oracle Portal, CMS
Référence du composant agrégat	CMP_SOCOM_Portail_LEL-VDM référence du composant agrégat « Portail Loisir En Ligne – VDM »

La figure suivante montre le diagramme des classes des métadonnées SOCOM bonifiées par les spécifications d'agrégation « AggregationSpecficiation ».

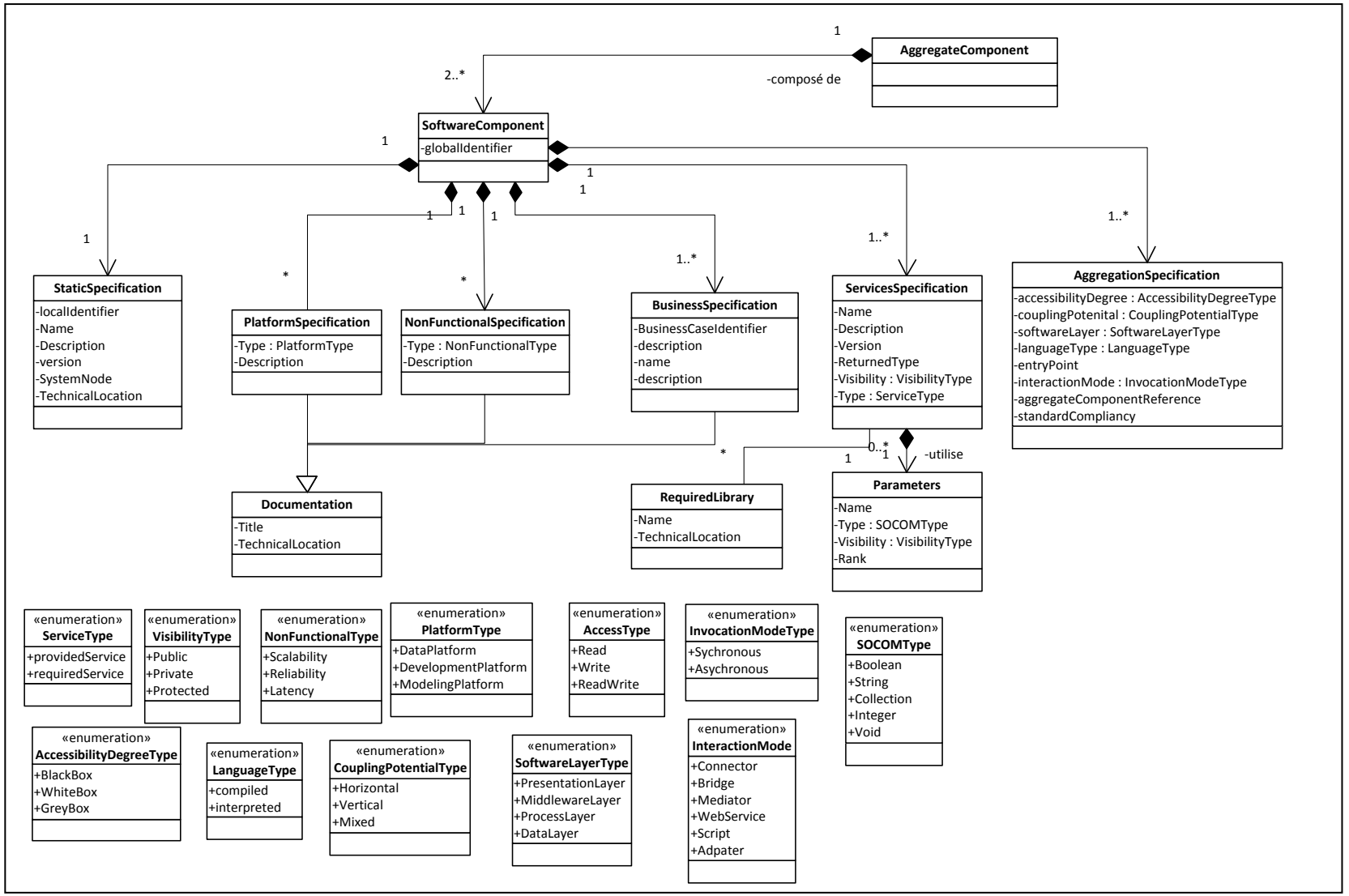


Figure 3.3 Modèle conceptuel des spécifications SOCOM avec les spécifications d'agrégations (diagramme de classes UML)

3.3 Les classes d'agrégation des composants logiciels

3.3.1 Élément de base pour la définition des classes d'agrégation

Pour décider de la classe d'agrégation, nous considérons essentiellement les dimensions suivantes :

- **CL** : couche logicielle à laquelle le composant appartient. On distingue les couches suivantes : Présentation (P), Traitement (T), Middleware (M), Données (D).
- **DA** : degré d'accessibilité au code du composant selon sa structure interne. On distingue ici trois situations : composant boîte noire (BN), composant boîte blanche (BB), composant boîte grise (BG).
- **PC** : potentiel de couplage du composant selon les possibilités d'interaction du composant avec les composants de la même couche ou de couches différentes. Nous distinguons ici le couplage horizontal (CH), le couplage vertical (CV) et le couplage mixte (CM), à la fois horizontal et vertical.

Parfois, nous utilisons occasionnellement l'attribut de compatibilité des plateformes de SOCOM et celui de compatibilité des standards pour s'assurer du bon choix de la classe d'agrégation. La figure suivante illustre un des triplets caractéristiques d'un composant dans un espace à trois dimensions, selon ses valeurs CL, DA et NC.

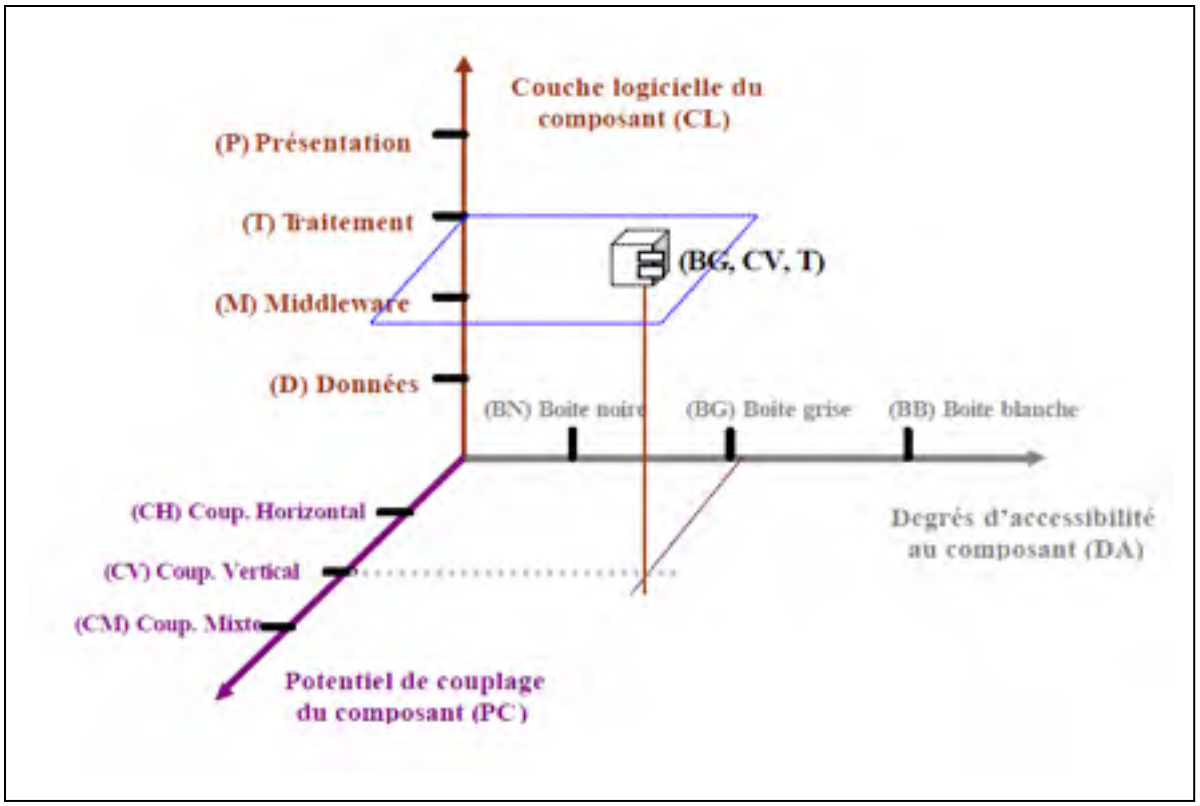


Figure 3.4 Caractérisation du composant logiciel
en fonction de trois paramètres PC, DA et CL

Après la définition des attributs d'agrégation, nous proposons quelques règles utiles pour définir des équations associées aux classes d'agrégation des composants. Ces règles sont rattachées aux métadonnées d'agrégation et sont définies comme suit :

- ***CoucheVoisineImmediate***(CpA, CpB) est la règle de couche qui vérifie que le composant CpB appartient à la couche immédiate supérieure ou immédiate inférieure du composant CpA ou l'inverse
- ***InclutUnComposantBoîteBlanche***(CpA, CpB, …, CpN) est la règle qui vérifie qu'au moins un des composant énumérés en paramètre est de type boîte blanche.
- ***CouplageFaisable***(CpA, CpB) est la règle qui vérifie que le potentiel de couplage du composant CpA tolère l'interaction avec le composant CpB. Autrement dit, si PC(CpA) = CH et PC(CpB) = CV, alors ***CouplageFaisable***(CpA, CpB) = FAUX et le couplage entre les deux composants n'est pas faisable. Par contre si (CpA) = CM et PC(CpB) =

CV, alors ***CouplageFaisable***(CpA, CpB) = VRAI et le couplage vertical entre les deux composants est faisable.

- Les valeurs possibles de la règle sont **VRAI** ou **FAUX.**

En fonction de ces trois paramètres et des règles associées, quatre types d'agrégation sont définis soient : connexion, collection, coordination et fusion. Ces types d'agrégation sont expliqués en détail plus loin. Pour chaque type, nous définissons les règles du degré d'accessibilité (RegAccess), du potentiel de couplage (RegCoup) et de la couche logicielle (RegCouche) et parfois celle de la compatibilité des plateformes (RegCompat). Nous avons proposé des abréviations pour faciliter la formulation et la lecture des règles.

À la fin de l'agrégation, nous obtenons un composant agrégat qui peut appartenir à l'un des deux types : physique ou logique. Un agrégat physique est un composant défini par une unité de déploiement unique, p. ex. un JAR en Java ou une DLL ou un EXE en .NET. Les codes des composants agrégés et la logique d'agrégation sont fusionnés dans une même structure. Le composant logique est le composant participant qui implante la logique de l'agrégation. Dans ce cas, les composants participants restent déployés dans leur plateforme d'origine.

3.3.2 Agrégation de base : agrégation par connexion

Ce type d'agrégation est une classe de base qui est utilisée par toutes les autres classes d'agrégation. Nous définissons une connexion entre deux composants logiciels par l'existence d'au moins une interaction entre un composant qui fournit un ou plusieurs services et un composant qui les consomme. Par classe de base, nous voulons définir la cardinalité minimale de l'agrégat qui est composé de deux composants au minimum.

En partant des trois attributs sélectionnés pour définir la classe d'agrégat, une agrégation par connexion peut avoir lieu entre deux composants d'une même couche ou appartenant à deux couches voisines immédiates. Autrement dit, nous ne pouvons pas établir une connexion entre la couche de présentation et la couche de données directement, il est primordial de

passer par les couches intermédiaires, soient la couche de traitement et celle d'intergiciel si elle existe. De ce fait, les composants participants dans une connexion peuvent avoir un potentiel de couplage horizontal, vertical ou mixte. Quant aux valeurs possibles de l'attribut « Degré d'accessibilité », les composants de la connexion doivent être minimalement type boîte noire pour que les interactions soient possibles; par conséquent ils peuvent être aussi de type boîte blanche ou boîte grise.

En résumé, pour définir une agrégation par connexion entre deux composants CpA et CpB, toutes les règles suivantes doivent être vérifiées:

- **RegAccess** : (**DA**(CpA), **DA**(CpB) ∈ {BN, BB, BG} **ET**
- **RegCoup** : (**PC**(CpA), **PC**(CpB) ∈ {CH, CV, CM} **ET** ***CouplageFaisable***(CpA, CpB))
- **RegCouche** : (**CL**(CpA) = **CL**(CpB) ∈{P, T, M, D}) **OU** (**CL**(CpA) ∈{P, T, M, D} **ET** ***CoucheVoisineImmediate***(CpA, CpB)).

Cette classe d'agrégation est souvent utilisée par l'architecte dans n'importe quel contexte d'agrégation. Nous proposons la connexion pour rappeler à l'architecte l'approche de décomposition en sous cas d'agrégation lors de la résolution d'un problème complexe d'agrégation. Autrement dit, n'importe laquelle des solutions d'agrégation utilise, à la base, la classe d'agrégation par connexion pour répondre à un besoin de développement à base de composant.

Soient deux composants « Gestionnaire de Formulaires Et de Contenu - GFEC» et « Gestionnaire de PROcessus » dont les attributs d'agrégation sont définis comme suit :

Tableau 3.2 Métadonnées d'agrégation SOCOM des composants participants à une agrégation par connexion

Métadonnées d'agrégation	Compsoant-GFEC	Composant GPRO
Potentiel de couplage	Mixte	Mixte
Degré d'accessibilité	Boîte blanche	Boîte blanche
Couche logicielle	Présentation	Traitement
Nœud du système	Prod55.ville.montreal.qc.ca	Prod55.ville.montreal.qc.ca
Point d'entrée du composant	gererFormulaireEtContenu	gererTableauDeBord
Méthode d'agrégation	Connecteur JAVA	Connecteur JAVA
Type de langage	Compilé	Compilé
Compatibilité avec les standards	JDK 1.4, XML, XSD, Spring	JDK 1.4, XML, XSD, Spring
Référence du composant agrégat	N/A	N/A
Plateforme de déploiement (SOCOM)	Oracle JAVA Container	Oracle JAVA Container

Le composant GFEC fournit des services pour la définition des types de contenu et génère des formulaires web pour gérer le contenu. Nous avons besoin de soumettre le contenu géré par ces formulaires à un cycle d'approbation défini et produit par le composant GPRO. Ce genre de besoin se traduit par une interaction entre les composants qui forment une agrégation par connexion.

À partir des valeurs des attributs des deux composants, nous appliquons la vérification des règles de l'agrégation par connexion.

- (**DA**(Composant-GFEC) = BB et **DA**(Composant-GPRO) = BB; **RegAccess** est vérifiée.
- (**PC**(Composant-GFEC) = CM, **PC**(Composant-GPRO) = CM, **ET** ***CouplageFaisable***(Composant-GFEC, Composant-GPRO); **RegCoup** est vérifiée.

- (**CL**(Composant-GFEC) = P et **CL**(Composant -GPRO) = T, donc **CL**(Composant-GFEC) ≠ **CL**(Composant-GPRO), mais ***CoucheVoisineImmediate***(**CL**(Composant-GFEC, Composant-GPRO)); **RegCouche** est vérifiée.

Le résultat de l'agrégation est le couple (Composant-GFEC, Composant-GPRO) que nous avons intitulé composant-GC (Composant de Gestion de Contenu). Il est important de mentionner que le composant-GC est un agrégat logique, contrairement à physique, car les deux composants ne sont pas sous le même serveur et il n'y a pas eu de fusion de code des deux composants.

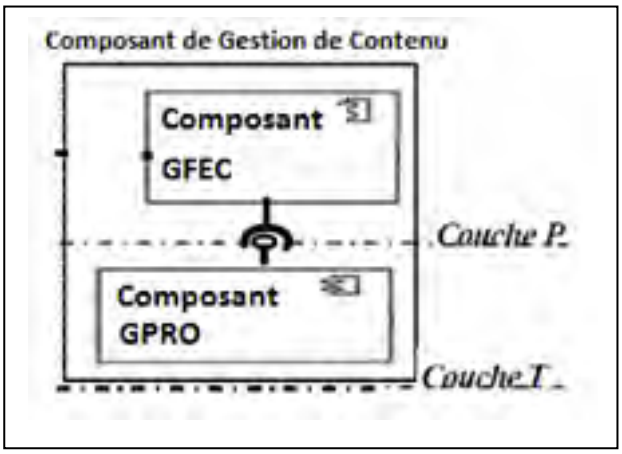


Figure 3.5 Exemple d'agrégation par connexion de deux composants.

3.3.3 Agrégation par collection

Dans cette deuxième classe d'agrégation, nous présentons une logique d'agrégation qui consiste à juxtaposer des composants de présentation de plusieurs applications. Ce genre d'agrégation n'a pas besoin d'interactions fortes entre les composants, mais d'un contenant ou d'une structure qui rassemble leur interface graphique. Nous trouvons ce besoin dans les portails des entreprises qui veulent souvent offrir une interface utilisateur intégrée des services offert aux employés; Afin que ces derniers ne soient pas obligés d'ouvrir plusieurs systèmes pour avoir accès à ces services individuellement, il suffit qu'ils s'authentifient au

portail pour y accéder. Nous assistons alors à une agrégation par collection des composants de présentation des différents systèmes tels que le système de paie, le système de déclaration des feuilles de temps, le système de billetterie et celui de la gestion des incidents, etc. Aussi les plateformes d'apprentissages à distance présentent le même besoin. Les concepteurs de ces plateformes disposent de plusieurs composants pédagogiques indépendants. Pour développer un cours, ils les agrègent dans une vue web intégrée qu'ils offrent aux étudiants. Nous pouvons citer à titre d'exemple le système TELOS du projet LORNET de la Téléuniversité.

En général, nous voulons agréger le composant de présentation de l'application A (CpA) et le composant de présentation de l'application B (CpB) dans un composant collectionneur de l'application C (CpC). Autrement dit, l'agrégation concerne uniquement les composants de présentation. Ces derniers doivent offrir au minimum un potentiel de couplage (PC) horizontal pour permettre l'interaction entre eux. Ils doivent avoir un degré d'accessibilité (DA) moyen ou fort, nous parlons dans ce cas de composants boîte grise ou blanche. En fait, comme nous voulons agréger le composant de présentation de l'application dans un composant collectionneur, nous devons avoir accès direct au code du composant participant, ou au moins aux paramètres pertinents si le composant est configurable de façon déclarative. Par conséquent, nous écartons le composant de type boîte noire pour garantir la faisabilité de l'agrégation. Pour réussir l'agrégation, les composants participants et le collectionneur doivent appartenir à une même plateforme ou au minimum à des plateformes compatibles. Dans l'industrie, les technologies portails, telles que le portail d'Oracle, Websphere d'IBM et SharePoint de Microsoft, jouent le rôle de composant collectionneur.

En résumé, pour définir une agrégation par collection entre trois composants CpA, CpB et CpC, toutes les règles suivantes doivent être vérifiées:

- **RegAccess** : (**DA**(CpA), **DA**(CpB)**, DA**(CpC) ∈ {BB, BG} **ET**
- **RegCoup** : (**PC** (CpA), **PC** (CpB), **PC** (CpC) ∈ {**CH, CM**} **ET** ***CouplageFaisable***(CpA, CpB, CpC) = VRAI) **ET**
- **RegCouche** : (**CL**(CpA) = **CL**(CpB) = **CL**(CpC) = {P}).

- **RegCompat:** les compatibilités des plateformes et des standards sont vérifiées.

Nous avons réalisé ce type d'agrégation dans le système Explor@-2 (Paquette *et al*., 2004), lors de l'adjonction d'un nouveau composant, un annotateur des émotions (Masmoudi, 2003). Le module « Gestionnaire de Ressource (GR) » permet de rechercher et de sélectionner des ressources pédagogiques (ou « learning objects ») et de gérer leur utilisation par les usagers d'Explor@-2. Un autre module important du système est le gestionnaire de la Structure Pédagogique (SP) qui fournit l'arbre de la structure d'un cours, les activités de cette structure et des ressources accessibles via le gestionnaire de ressources. Nous avons agrégé trois composants de présentation AE, GR et SP dans un composant web du système Explor@-2.

En résumé, le composant agrégat permet à un usager d'Explor@-2 - le plus souvent un apprenant – d'engager un dialogue avec d'autres usagers en commentant, au moyen de messages textes et d'émoticônes (fourni par AE), des activités de la structure pédagogique (fourni par SP) d'un cours, ou encore des ressources pédagogiques (fourni par GR) utilisées dans le cours. Il est important de rappeler que chacun des modules dispose de quatre composants appartenant aux quatre couches logicielles P, T, M et D.

Le besoin d'affaires est de fournir un client Web riche d'Explor@2 qui offre, dans une interface graphique intégrée, la structure des activités pédagogique (SA), l'annotateur émotif (AE) et le gestionnaire des ressources pédagogiques (GR). Ceci nous amène à étudier les composants de présentation des applications en question, soient SA-P, AE-P et GR-P.

La Figure 3.6 concrétise la relation entre trois composants de présentation et le composant collectionneur, le client web du système Explor@-2.

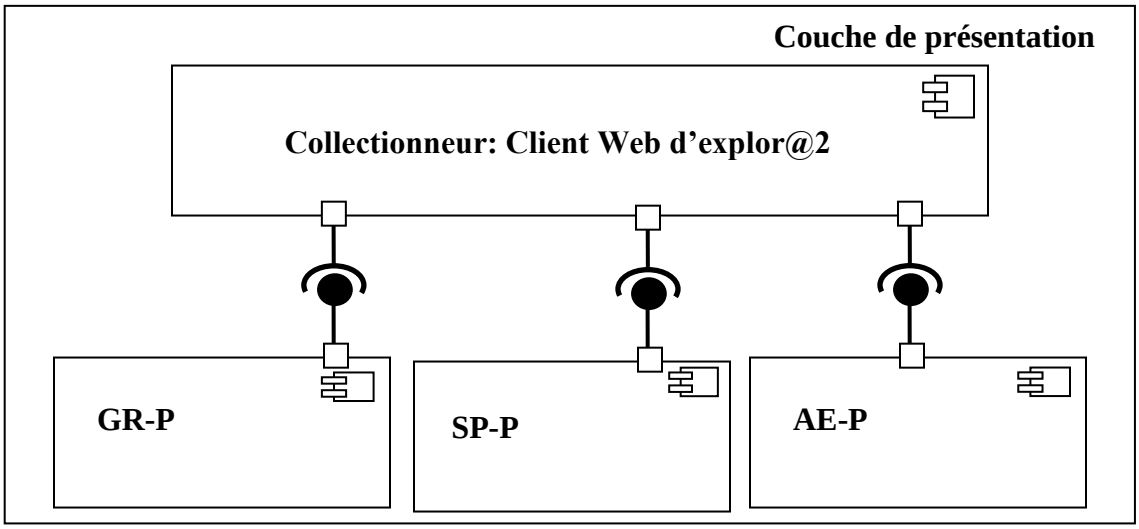


Figure 3.6 Exemple d'agrégation par collection de composants dans Explor@ 2

Soit les trois composants de présentation (P) des modules du système Explor@-2 « Gestionnaire des Ressources – GR-P», « Structure des activités Pédagogiques – SP-P» et « Annotateur Émotif – AE-P » dont les attributs d'agrégation sont définis comme suit :

Tableau 3.3 Métadonnées d'agrégation SOCOM des composants participants à une agrégation par collection

Métadonnées d'agrégation	Composant-GR-P	Composant-SP-P	Composant-AÉ-P	Client-Web - Explo@2: Collectionneur
Potentiel de couplage	Mixte	Mixte	Mixte	Horizontal
Degré d'accessibilité	Boîte grise	Boîte grise	Boîte grise	Boîte blanche
Couche logicielle	P	P	P	P
Nœud du système	ca.teluq.licef.explora2.api.gestressource	ca.teluq.licef.explora2.api.structurepedagogique	ca.teluq.licef.explora2.api.annotateurs	ca.teluq.licef.explora2.api.client
Point d'entrée du composant	getRessourceLOMService	getActivityStrucService	getAnnotationService	https://teluq.explora2.ca/contentmanagement/
Méthode d'agrégation	API JAVA	API JAVA	API JAVA	API JAVA
Type de langage	Compilé	Compilé	Compilé	Compilé
Compatibilité avec les standards	JDK 1.4, XML, XSD, Spring	JDK 1.4, XML, XSD, Spring	JDK 1.4, XML, XSD, Spring	JDK 1.4, XML, XSD, Spring
Référence du composant agrégat	CPT_SOCOM_GR	CPT_SOCOM_SP	CPT_SOCOM_AE	CPT_SOCOM_Explor@-2
Plateforme de déploiement (SOCOM)	APACHE TOMCAT	APACHE TOMCAT	APACHE TOMCAT	APACHE TOMCAT

À partir des valeurs des attributs des composants, nous appliquons la vérification des règles de l'agrégation par collection:

- (**DA**(Composant-GR-P) = {BG}, **DA**(Composant–SP-P) = {BG}, **DA**(Composant-AE-P) = {BG} et **DA**(Client-Web-Explo@2) = {BB}; **RegAccess** est vérifiée.
- (**PC**(Composant-GR-P) = {CM}, **PC**(Composant-SP-P) = {CM}, **PC**(Composant-AE-P) = {CM} et **PC**(Client-Web-Explo@2) = {CH}, **ET *CouplageFaisable***(Composant-GR-P, Composant–SP-P, Composant-AE-P, Client-Web-Explo@2)); **RegCoup** est vérifiée.
- (**CL**(Composant-GR-P) = **CL**(Composant–SP-P) = **CL**(Composant-AE-P) = **CL**(Client-Web-Explo@2) = {P} . **RegCouche** est vérifiée.
- Les composants participants sont déployés sous TOMCAT dans une plateforme JAVA identique avec celle du composant collectionneur. **RegCompat** est vérifiée.

Les règles sont vérifiées et les composants Composant-GR-P, Composant-SP-P et Composant-AÉ-P sont de type boîte blanche et communiquent avec un potentiel de couplage horizontal au niveau de la couche de présentation, ce qui correspond à notre définition d'une collection. L'agrégat résultat est le composant collectionneur « Client Web Explor@2 ».

L'agrégation par collection produit un agrégat qui regroupe les composants de présentation dans une même interface graphique web ou autre. Les interactions entre les composants ne sont pas complexes, il s'agit d'une suite d'agrégations par connexion entre le collectionneur et les composants agrégés. Autrement dit, les composants participants ne se connaissent pas entre eux, ils interagissent uniquement avec le composant collectionneur. Le collectionneur est un composant agrégat logique.

3.3.4 Agrégation par coordination

L'agrégation par coordination met en relation plus de deux composants. Dans une architecture logicielle par couche, nous avons constaté que la coordination est souvent présente entre les composants logiciels appartenant aux couches de traitement et d'intergiciel. En effet, nous pensons que les besoins en coordination entre les composants des couches de

présentation et des données peuvent se ramener à une coordination entre les couches de traitement et d'intergiciel. En effet, si nous avons besoin d'afficher une vue intégrée de l'information suite à une coordination entre différents sources de données, les architectes n'agrègent pas systématiquement les composants de présentation des applications de chaque source. Ils proposent plutôt des adaptations aux composants de traitement et d'intergiciel qui récupèrent les données séparément des sources. Ensuite, un composant de traitement assume la responsabilité d'agréger ces données. Ce dernier offre un service à la couche de présentation pour récupérer la vue intégrée des données et les afficher avec un style approprié. Le raisonnement est similaire pour les composants de la couche des données. Souvent, les sources de données sont laissées intactes et les architectes évitent la coordination entre les composants de données. Ils apportent des adaptations aux composants de la couche intergiciel ou d'accès aux données pour offrir des services responsables de récupérer ces données et les transférer à plusieurs composants de traitement qui, eux, entrent dans une coordination pour satisfaire les besoins d'agrégation des données.

Ainsi, nous pensons que la coordination est faite dans la plupart des cas entre les couches middleware (M) et de traitement (T). Ces composants communiquent généralement suivant un potentiel de couplage (PC) horizontal, vertical ou mixte. Les composants qui entrent en interaction peuvent avoir n'importe quel degré d'accessibilité (DA): boîte noire, boîte grise ou boîte blanche. Toutefois, au moins un des composants doit être de type boîte blanche pour supporter la coordination. En effet, un des composants de la coordination assume la responsabilité d'encapsuler la logique de coordination sous forme de code qui est intégré à son code existant. Si ce dernier est de type boîte noire ou grise et les composants participants ne peuvent pas encapsuler la logique de coordination, l'agrégation délègue cette responsabilité à un nouveau composant qui sera de type boîte blanche. Nous définissions le composant fédérateur comme le composant qui a le contrôle sur les autres composants, qui est responsable d'encapsuler la logique de la coordination.

Nous définissons l'agrégation par coordination comme suit: soient un composant CpA de l'application A, un composant CpB de l'application B (CpB) et un composant CpD de

l'application D. Pour définir une agrégation par coordination entre les composants CpA, CpB et CpD, toutes les règles suivantes doivent être vérifiées:

- **RegCouche**:(**CL**(CpA) ∈ {T, M}, **CL**(CpB) ∈ {T, M}, **CL**(CpD) ∈ {T, M}) **ET**
- **RegAccess :**(**DA**(CpA) ∈ {BN, BG, BB}, **DA**(CpB) ∈ {BN, BG, BB}, **DA**(CpD) ∈ {BN, BG, BB}, ***InclutUnComposantBoîteBlanche***(CpA, CpB, CpD) = VRAI). Si **DA**(CpA) et **DA**(CpB) et **DA**(CpD) ≠ {BB}, l'agrégat est un nouveau composant CpE avec **DA**(CpE) = {BB}, **CL**(CpE) ∈ {T, M}) **ET**
- **RegCoup :**(**PC** (CpA), **PC** (CpB), **PC** (CpD) ∈ {CH, CV, CM} et ***CouplageFaisable***(CpA, CpB, CpD))

Nous précisons que, parfois, toutes les règles peuvent être vérifiées, mais aucun des composants ne peut encapsuler la logique de coordination à cause des requis non fonctionnels. Par exemple, des incompatibilités de type sécurité (http au lieu de https) ou de zonage réseau (zone Internet ouverte au lieu de zone sécurisée DMZ) peuvent empêcher un des composants participants d'occuper le rôle de fédérateur. Dans cette situation, le contexte d'agrégation impose l'ajout d'un nouveau composant agrégat CpE qui assure la coordination des composants CpA, CpB et CpD.

Nous avons identifié ce type d'agrégation dans le projet EduSource[18] au sein d'une équipe qui a implanté la spécification IMS-DRI[19] pour créer un réseau de huit répertoires d'objets d'apprentissages interopérables distribués sur différents serveurs au Canada, en Australie et aux États-Unis. L'exemple montre seulement deux répertoires d'objets : Explor@2 de la Téluq et POND de SFU.

[18] Description du projet EduSource disponible sur l'adresse Web : http://edusource.netera.ca/french/home_fr.html

[19] IMS-DRI (IMS Digital Repository Initiative) Spécification disponible via l'adresse http://www.imsglobal.org/digitalrepositories/

La Figure 3.7 montre une agrégation entre les composants « Federated Search Engine » (FSE), « ECL Client », « ECL Gateway » et « Search Result Display » (SRD). ECL[20] est un protocole qui définit des verbes pour formuler les requêtes de recherche de LOM-IEEE[21] et les réponses associées (Hatala, 2004). Le besoin d'affaires se résume par un client web « Search Client interface » qui lance une requête pour la recherche d'objets d'apprentissage décrits par des spécifications LOM. Cette requête est déléguée à un composant-SRD qui est responsable d'agréger les résultats provenant de plusieurs banques LOMs (Explor@-2 et POND dans notre cas). Ce dernier extrait les préférences d'affichage des utilisateurs et transmet la requête à un engin de recherche fédérée qui connait le composant de médiation « ECL-Gateway ». À la réception de la requête, le composant de médiation transforme la requête en appliquant le protocole ECL. Le « client-ECL » traduit l'appel en une requête native puis interroge la banque locale. L'objet de réponse contenant les résultats de recherche réalise un parcours inverse jusqu'à l'interface graphique des utilisateurs à l'origine de la requête. Le client web affiche les résultats d'objets d'apprentissages trouvés formatés et agrégés par le composant SRD. Nous avons utilisé la technologie des services Web pour assurer la communication entre les composants au niveau des services.

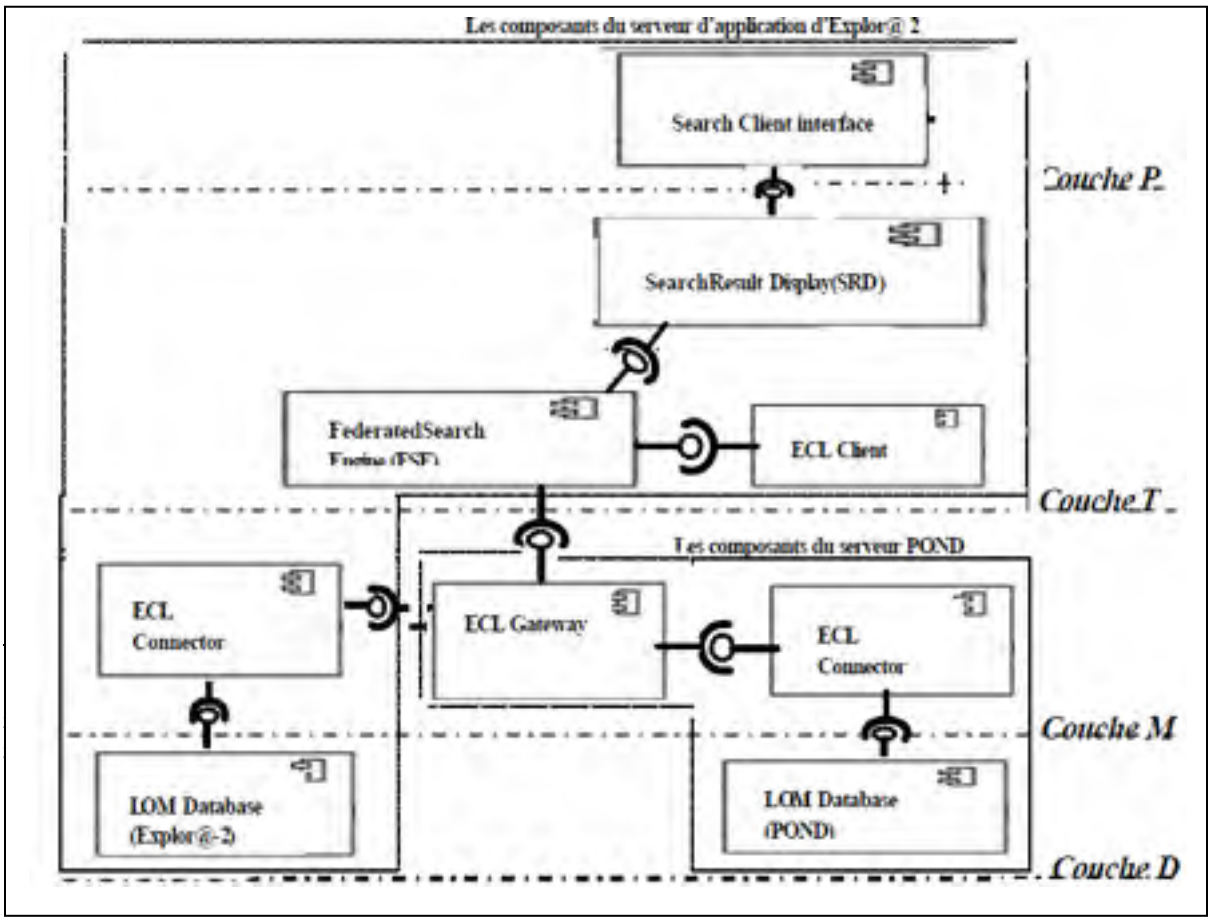


Figure 3.7 Une agrégation par coordination entre composants.

Voici les attributs d’agrégation des quatre composants choisis:

Tableau 3.4 Métadonnées d'agrégation SOCOM des composants participants à une agrégation par coordination

Métadonnées d'agrégation	Composant-FSE	Composant-SRD	Composant-ECL-Gateway	Composant-ECL-Client
Potentiel de couplage	Mixte	Horizontal	Mixte	Mixte
Degré d'accessibilité	Boîte blanche	Boîte blanche	Boîte grise	Boîte Grise
Couche logicielle	T	T	M	M
Nœud du système	ca.edusource.explora2.	ca.edusource.explora2.	ca.edusource.pond.	ca.edusource.explora2
Point d'entrée du composant	dispatchsearchLOMsService()	searchResultDisplayService()	eclProtocolService()	setECLClientRequestservices()
Méthode d'agrégation	API JAVA	API JAVA	ECL Connector (Java Connector)	ECL Connector (Java Connector)
Type de langage	Compilé	Compilé	Compilé	Compilé
Compatibilité avec les standards	JDK 1.5, XML, XSD,	JDK 1.5, XML, XSD,	JDK 1.5, XML, XSD,	JDK 1.5, XML, XSD,
Référence du composant agrégat	CPT_SOCOM_Explor@-2	CPT_SOCOM_Explor@-2	CPT_SOCOM_POND	CPT_SOCOM_Explor@-2
Plateforme de déploiement	APACHE TOMCAT	APACHE TOMCAT	APACHE TOMCAT	APACHE TOMCAT

À partir des valeurs des attributs des quatre composants, nous appliquons la vérification des règles de l'agrégation par coordination:

- **(CL**(Composant-FSE) = T, **CL**(Composant-SRD) = T, **CL**(Composant-ECL-Gateway) = M, **CL**(Composant-ECL-Client) = M); **RegCouche** est vérifiée.
- (**DA**(Composant-FSE) = {BB}, **DA**(Composant-SRD) = {BB}, **DA**(Composant-ECL-Client) = {BG}, **DA**(Composant-ECL-Gateway) = {BG} ET ***InclutUnComposantBoîteBlanche***(Composant-FSE, Composant-SRD, Composant-ECL-Client, Composant-ECL-Gateway)); **RegAccess** est vérifiée.
- (**PC**(Composant-SRD) = CH**, PC**(Composant-FSE) = CM**, PC** (Composant-ECL-Gateway) = CM, **PC** (Composant-ECL-Client) = **CM**) ET ***CouplageFaisable***(Composant-SRD, Composant-FSE, Composant-ECL-Gateway, Composant-ECL-Client)); **RegCoup** est vérifiée

Les règles sont vérifiées et les quatre composants peuvent participer à une agrégation par coordination. Dans ce cas, le composant-FSE est sélectionné comme composant fédérateur car il est l'expert. Il connait tous les autres composants et il est de type boîte blanche, par conséquent il peut encapsuler le code de la coordination. Contrairement à la collection, la coordination ne définit pas un composant agrégat qui héberge les autres composants dans une même plateforme. La coordination définit un des composants participants comme agrégat. Ce dernier contient uniquement la logique de coordination et non le code des autres composants. Il invoque les services fournis par les autres composants qui restent chacun dans leur plateforme d'origine. Dans le cas d'une coordination, le composant résultant est un agrégat logique.

3.3.5 Agrégation par fusion

Pour ce type d'agrégation, les composants des applications appartiennent à la même couche logicielle (CL) ou à des couches immédiatement voisines. Ces composants peuvent avoir des potentiels de couplage (PC) différents, mais ils doivent garantir la faisabilité de l'agrégation. En effet, si nous devons agréger deux composants appartenant à deux couches différentes

avec un potentiel de couplage vertical, nous préconisons que l'agrégat final est un seul composant physique qui est conceptuellement réparti entre les couches des composants participants.

Le type d'agrégation par fusion exige des composants avec des degrés d'accessibilité de type boîte blanche ou grise. En effet, nous devons avoir des composants à code ouvert ou semi ouvert pour que nous puissions les fusionner et les restructurer. Dans ce cas, les composants de type boîte noire ne sont pas acceptables parce que la fusion de leur structure n'est pas faisable. Ces derniers possèdent une structure interne non visible et non modifiable, contrairement aux composants de type boîte grise ou blanche dont la structure interne est visible et modifiable par paramétrisation ou par programmation.

L'agrégation par fusion se réalise selon deux modèles. Le premier fait intervenir un troisième composant agrégat qui va englober les deux autres. Le second est le modèle tout-partie dans lequel un des composants participants dispose de plus de contrôle et peut englober l'autre. Dans les deux modèles, le composant agrégat doit être de type boîte blanche pour faciliter l'implantation du code de la logique d'agrégation.

Pour définir une agrégation par fusion entre les composants CpA et CpB, toutes les règles suivantes doivent être vérifiées:

- **RegCouche** : (**CL**(CpA) = **CL**(CpB) ∈ {P, T, M, D} et Si (**CL**(CpA) ≠ **CL**(CpB) alors ***CoucheVoisineImmediate***(CpA, CpB) = VRAI) ET
- **RegAccess** : (**DA**(CpA), **DA**(CpB) = {BB, BG} **ET** ***InclutUnComposantBoîteBlanche***(CpA, CpB)). Si **DA**(CpA), **DA**(CpB) ≠ {BB}, alors l'agrégat est un nouveau composant CpD avec **DA**(CpD) = {BB}) **ET**
- **RegCoupl** : (**NC**(CpA), **NC**(CpB) ∈ {CH, CV, CM} **ET** ***CouplageFaisable***(CpA, CpB))
- **RegCompat:** Les compatibilités des plateformes et des standards sont vérifiées.

La fusion des composants suppose la fusion de leur code et leur structure. Il n'y a plus de logique d'affaires indépendante par composant, mais une logique d'affaires commune supportée par une structure commune du composant agrégat. Si l'architecte n'a pas besoin de changer la structure interne des composants pour satisfaire ses besoins d'agrégation et s'il décide de la garder intacte, nous jugeons que cette agrégation n'est pas une fusion. Il peut refaire l'analyse de classification pour vérifier si les autres types d'agrégation peuvent répondre à ses besoins.

Si tous les composants participants sont de type boîte grise, l'architecte est obligé de définir un nouveau composant agrégat CpD de type boîte blanche. Ce dernier englobe le code de la logique fusion et celui des composants CpA et CpB. Il représente le nouveau contenant physique de l'agrégat. Les frontières des composants CpA et CpB disparaissent et cèdent la place à un seul agrégat physique CpD qui est déployé dans une seule plateforme.

Soient deux composants du projet EduSource : « Search Result Merge (SRM) », un composant développé par l'équipe de la Téluq et « LOM Result Sort (LRS)», un composant développé par l'équipe SFU[22] qui implémente plusieurs algorithmes de tri des données en format XML. Notre équipe de la Téluq a développé le composant « Search Result Display (SRD) » introduit dans la section 3.3.4. SRD retourne au composant « Search Client Interface» les résultats de recherche de plusieurs banques de LOM sous forme de liste. Nous avons besoin d'ajuster les algorithmes de tri pour les appliquer aux résultats retournés selon des critères spécifiques de l'utilisateur. Nous avons ajouté du code dans le composant SRD pour recevoir les résultats de recherche de FSE sous forme d'objets LOM. Nous les avons sérialisés en format XML dans une collection. Ensuite, nous avons adapté le composant LRS pour réaliser des tris de la collection XML des LOMs. Nous avons fusionné le code des composants SRM et LRS dans SRD. L'exemple de la Figure 3.8 montre le diagramme des composants de l'agrégation par fusion des composants SRM et LRS.

[22] SFU : Simon Fraser University

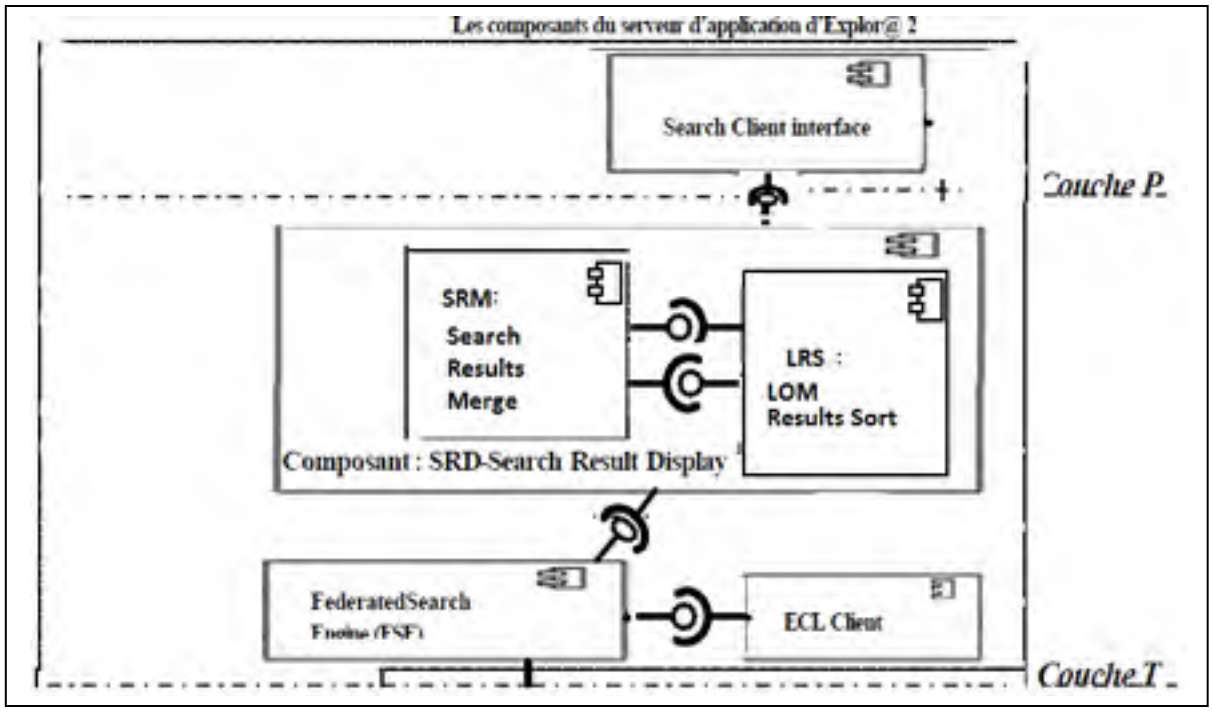


Figure 3.8 Composant de présentation « SRD» obtenu par une agrégation par fusion

Tableau 3.5 Métadonnées d'agrégation SOCOM des composants participants à une agrégation par fusion

Métadonnées d'agrégation	**Composant-SRM**	**Composant-LRS**
Potentiel de couplage	Vertical	Mixte
Degré d'accessibilité	Boîte grise	Boîte grise
Couche logicielle	T	T
Nœud du système	ca.edusource.explora2.	ca.edusource.explora2
Point d'entrée du composant	mergeLOMResultsService()	sortLOMResultsServices()
Méthode d'agrégation	API JAVA	API JAVA
Type de langage	Compilé	Compilé
Compatibilité avec les standards	JDK 1.5, XML, XSD, JEE ,	JDK 1.5, XML, XSD, JEE,
Référence du composant agrégat	CPT_SOCOM_EduSource_SRD	CPT_SOCOM_EduSource_SRD
Plateforme de déploiement (de SOCOM)	Tomcat	Tomcat

Le Tableau 3.5 montre les attributs d'agrégation des composants participants et ceux du composant agrégat. À Partir des valeurs des attributs des composants, nous appliquons la vérification des règles de l'agrégation par coordination:

- **(CL**(Composant-SRM) = T et **CL**(Composant-LRS) = T); **RegCouche** est vérifiée.
- (**DA**(Composant-SRM) = {BG}, **DA**(Composant-LRS) = {BG} **ET *InclutUnComposantBoîteBlanche***(Composant- SRM, Composant- LRS) **n'est pas vérifiée**); **RegAccess** n'est pas vérifiée, par conséquent, l'agrégat est un nouveau composant, SRD.
- (**PC**(Composant- SRM) = CV**,** **PC**(Composant- LRS) = CM et ***CouplageFaisable***(Composant-SRM, Composant-LSR)); **RegCoup** est vérifiée.
- Selon les valeurs de l'attribut « compatibilité avec standards » et celles des plateformes de SOCOM, la règle **RegCompat** est vérifiée.

Les règles sont vérifiées et les deux composants sont agrégés pas fusion. Nous rappelons que, dans ce cas, le modèle du nouveau composant englobant est choisi à cause du degré d'accessibilité de type boîte grise des composants participants, d'où la définition d'un nouveau composant-SRD qui est responsable de l'implantation du code de la fusion. Contrairement aux agrégations par connexion, collection et coordination, la fusion résulte en un seul composant physique qui est déployé dans une seule plateforme technologique. En général, les composants sources sont fusionnés dans un nouveau composant ou dans l'un des composants participants. Le code des composants participants change dans la majorité des cas. Aussi la fusion peut avoir lieu dans toutes les couches logicielles et ne se restreint pas à la présentation (comme la collection) ni aux couches de traitement et d'intergiciel (comme la coordination). La fusion est aussi différente d'une connexion car elle change le code des composants participants. L'agrégat est un composant physique.

3.4 Tableau récapitulatif des caractéristiques des différentes classes d'agrégation

Le tableau suivant présente une synthèse des valeurs possibles des attributs d'agrégation et des règles d'agrégation associées pour les classes d'agrégation proposées.

Tableau 3.6 Tableau récapitulatif des équations des classes d'agrégations proposées : connexion, collection, coordination et fusion

Classes d'agrégation / Attributs d'agrégation et les règles associées	**Agrégation par connexion**	**Agrégation par collection**	**Agrégation par coordination**	**Agrégation par fusion**
Couche logicielle	{P, T, M, D}	{P}	{T, M}	{P,T,M,D}
CoucheVoisineImmediate() requise	X			X
Degré d'accessibilité	{BN, BB, BG}	{BB, BG}	{BN, BB, BG}	{BB, BG}
InclutUnComposantBoîteBlanche() requise			X	X
Potentiel de couplage	{CH, CV, CM}	{CH, CM}	{CH, CV, CM}	{CH,CV,CM}
CouplageFaisable() Requise	X	X	X	X
Compatibilité requise des plateformes et des standards		X		X
Possibilité de création d'un nouveau composant agrégat			X	X
Type du composant agrégat	Logique	Logique	Logique	Physique
Les patrons d'agrégation possibles	Unidirectionnel, Bidirectionnel	Collection-neur	Fédérée, Répartie	En partie, Totale

3.5 Les différents patrons d'agrégation des composants logiciels

À partir des définitions des catégories d'agrégation énoncées précédemment, nous définissons dans ce qui suit des patrons d'agrégation correspondants à ces catégories. Pour chaque catégorie, nous avons élaboré des patrons d'agrégation accompagnés par des modèles sous la forme de diagrammes de composants UML2. Suite à chaque patron, nous décrivons les caractéristiques techniques du nouveau composant, soit l'agrégat.

3.5.1 Les patrons d’agrégation par connexion

3.5.1.1 Connexion unidirectionnelle

Dans ce modèle, deux composants entrent dans une interaction qui est déclenchée par le premier et qui sera servie par l’autre, soit une relation client/serveur. Le composant serveur fournit des services consommés par le composant client.

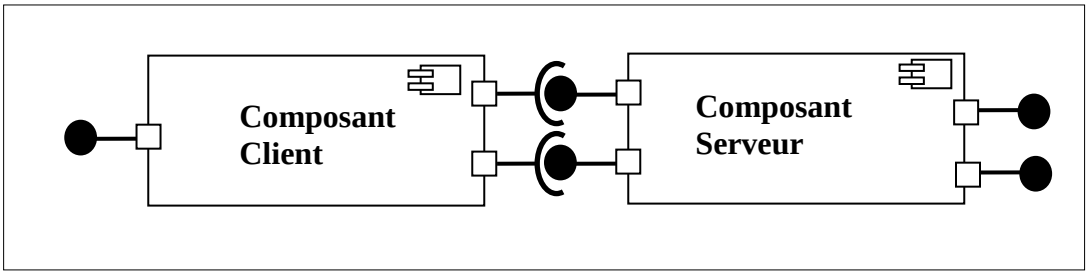


Figure 3.9 Le modèle du patron d'agrégation par connexion unidirectionnelle

3.5.1.2 Connexion bidirectionnelle

Dans cette catégorie d’agrégation, la connexion s’établit entre les deux composants par des associations bidirectionnelles. Ce style est connu sous le nom d'égal à égal (« peer-to-peer ») où tous les composants peuvent communiquer les uns avec les autres, et il n'y a pas de serveur central. Tous les composants sont des pairs qui peuvent jouer le rôle de client ou de serveur.

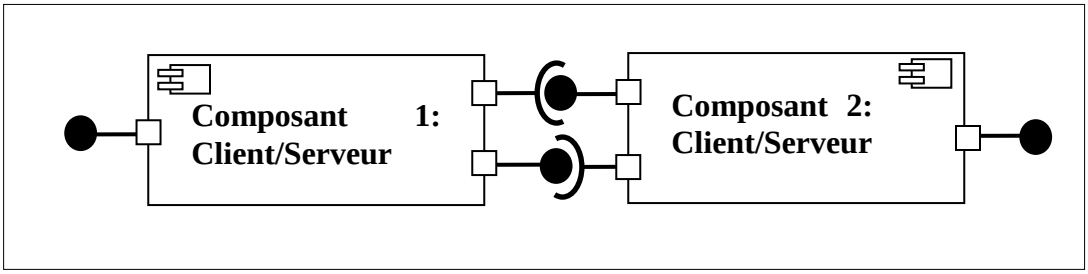


Figure 3.10 Le modèle du patron d'agrégation par connexion bidirectionnelle

Nous précisons que dans les cas des patrons d'agrégation par connexion, le résultat de l'agrégation est un composant agrégat logique formé par les composants participants.

3.5.2 Les patrons d'agrégation par collection

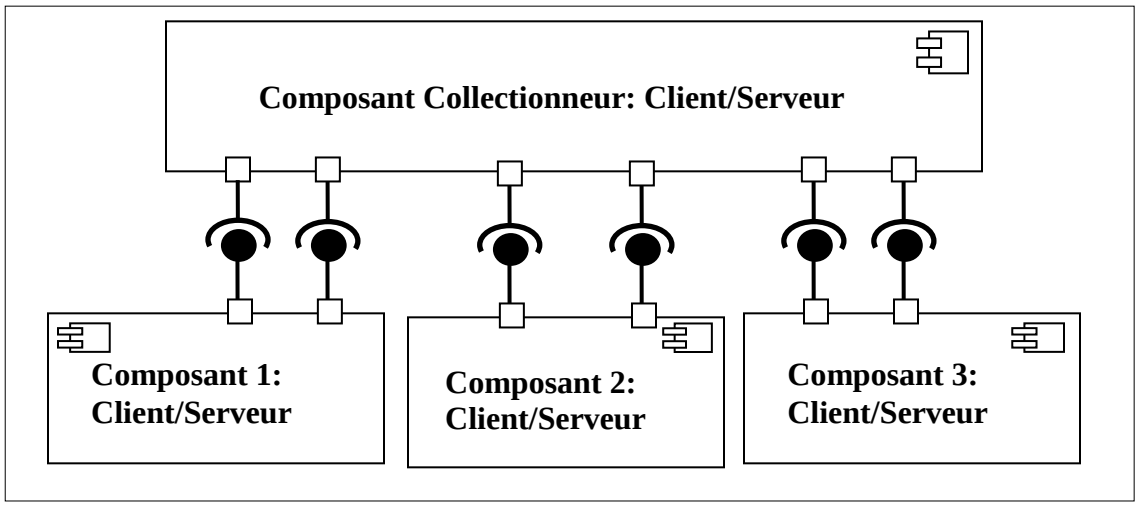


Figure 3.11 Le modèle du patron collectionneur de l'agrégation par collection

Dans un contexte d'agrégation par collection, le composant collectionneur n'implémente aucune logique de fédération ni de composition, il collectionne l'ensemble des composants dans une seule interface qui regroupe uniquement les services utilisés des composants participants par le composant collectionneur. Nous précisons que dans ce cas, il existe des agrégations par connexion unidirectionnelle entre le composant Collectionneur et les composants agrégés.

3.5.3 Les patrons d'agrégation par coordination

3.5.3.1 Un patron d'agrégation par coordination répartie

Dans ce patron d'agrégation, nous avons besoin de plus de deux composants qui entrent dans des interactions binaires. Il n'y a pas de fédération par un des composants ou un nouveau composant. Chaque composant de l'agrégat peut communiquer avec un autre composant en

établissant des interactions par des connexions unidirectionnelles ou bidirectionnelles. Ces interactions se font dans un ordre logique établi entre les composants de l'agrégat selon les besoins de la coordination. Par exemple, la figure ci-dessous montre une coordination répartie.

Voici les différents types d'interactions qui se dégagent des composants :
Composant 1 – Composant 2 : Patron d'agrégation par connexion bidirectionnelle.
Composant 3 – Composant 1 : Patron d'agrégation par connexion unidirectionnelle.
Composant 2 – Composant 3 : Patron d'agrégation par connexion unidirectionnelle.

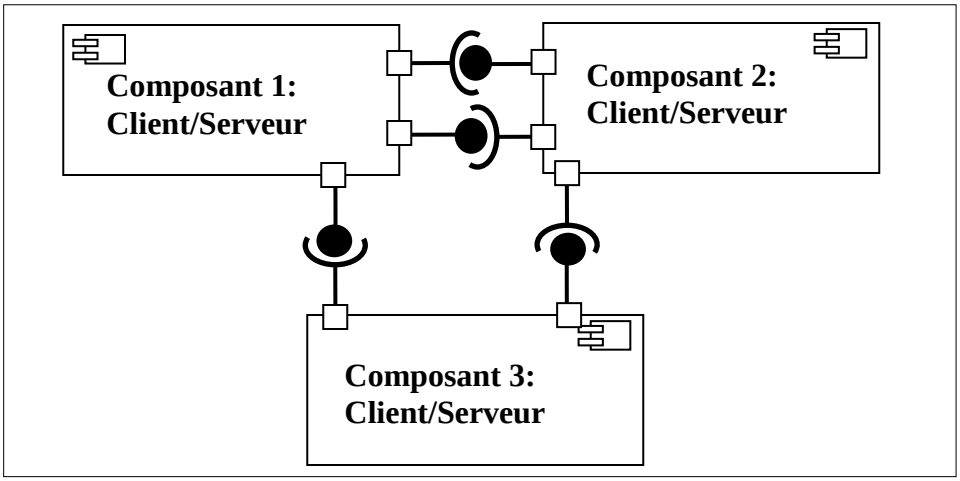


Figure 3.12 Le modèle du patron « coordination répartie »
de l'agrégation par coordination

3.5.3.2 Agrégation par coordination fédérée

Dans ce type d'agrégation, plus de deux composants logiciels entrent en communication. Toutefois, un seul composant (fédérateur) a le droit de communiquer avec les autres composants, qui sont alors des unités fédérées. Dans ce type d'agrégation, deux choix sont possibles pour former l'agrégat. Le premier suppose que l'un des composants agrégés dispose de la majorité des contrôles et il est choisi comme composant fédérateur. Le deuxième choix est la création d'un nouveau composant qui contiendra tous les échanges

possibles entre les composants fédérés. Dans ce cas aussi, les interactions pair-à-pair entre le composant fédérateur et chacun des composants fédérés seront conformes au patron d'agrégation par connexion unidirectionnelle ou bidirectionnelle.

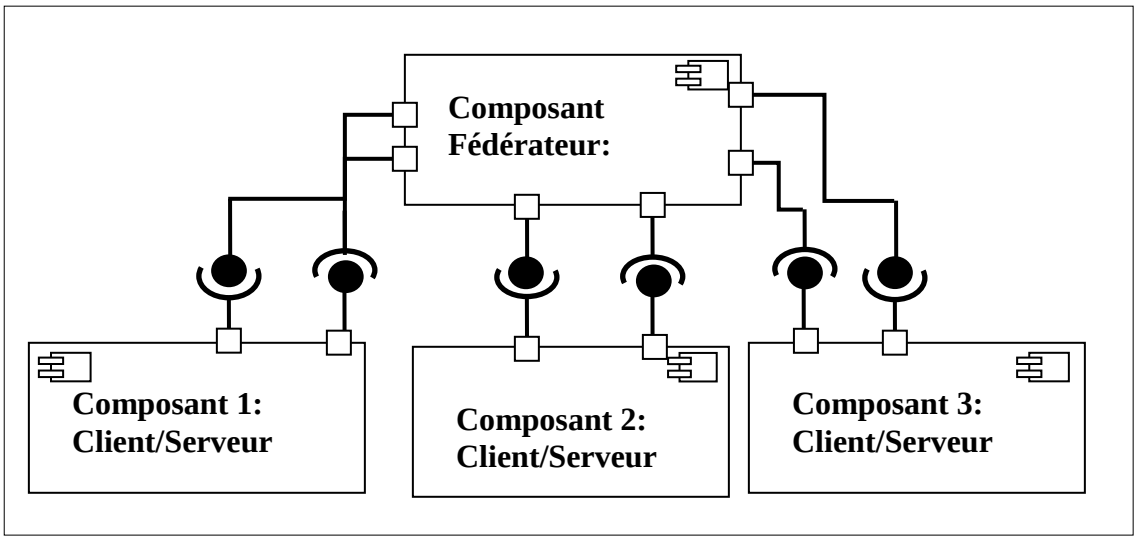


Figure 3.13 Le modèle du patron d'agrégation par coordination fédérée

Après l'application du patron d'agrégation par coordination, le composant agrégat est constitué par l'ensemble des composants et leur interaction dans le cas de la coordination répartie. Dans le cas du patron fédéré, l'agrégat est le composant fédérateur.

Dans tous les cas, le composant agrégat fournira les services qui sont déjà offerts par l'ensemble des composants auxquels on retranche ceux qui sont requis pour servir les différentes interactions entre les composants participants.

3.5.4 Les patrons d'agrégation par fusion

3.5.4.1 Les patrons d'agrégation par fusion en partie

Dans ce type d'agrégation par fusion, le résultat est un composant physique unique. Autrement dit, plusieurs composants participent à l'agrégation et un des composants

encapsulera les autres. Une seule unité de déploiement physique représente l'agrégat. Le qualificatif « en partie » provient du fait qu'une partie des composants à agréger englobe tous les autres composants. Par exemple, les trois composants ci-dessous seront agrégés dans la même unité de déploiement du composant 1. En effet, le composant 1 dispose de la majorité des fonctionnalités que l'agrégat doit offrir. Ainsi, le composant agrégat fournira les services du composant 1 et qui lui sont intégrés. Dans la majorité des cas, les composants participants sont développés dans le même langage de programmation.

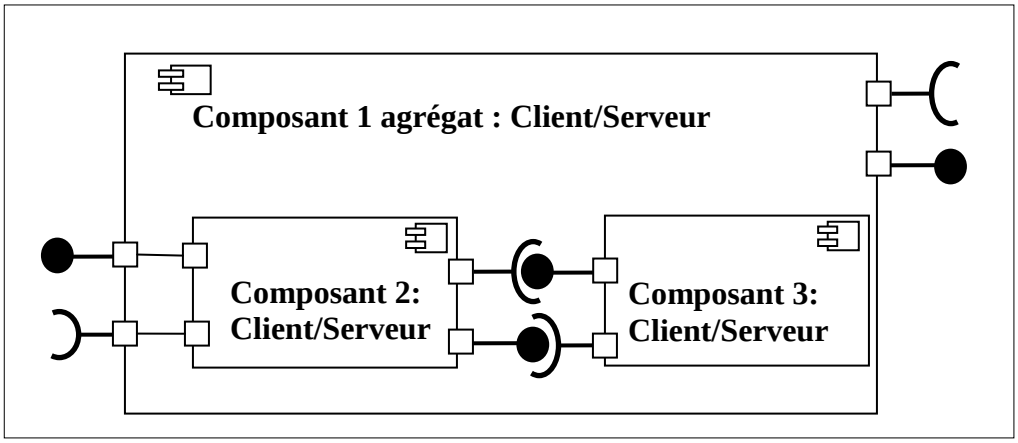


Figure 3.14 Le modèle du patron d'agrégation par fusion en partie

3.5.4.2 Les patrons d'agrégation par fusion totale

Dans ce type d'agrégation, un ensemble de composants participants partagent l'ensemble des interactions. Autrement dit, il n'y a pas de composant fédérateur ou englobant. Ainsi, un nouveau composant est créé pour englober physiquement tous les composants à agréger. Le diagramme des composants ressemble à celui présenté dans la section précédente sauf que tous les services fournis et requis sont ceux des composants participants. Le composant agrégat n'offre aucun service de plus que ceux offerts par les composants fusionnés.

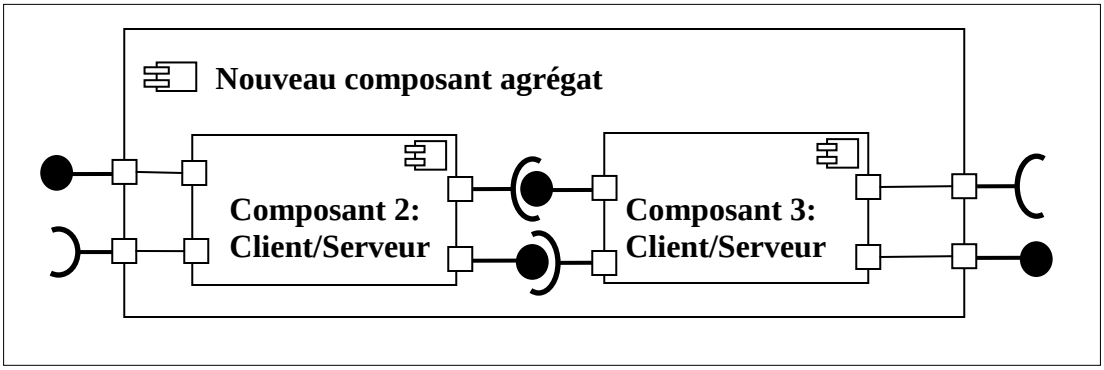


Figure 3.15 Le modèle du patron d'agrégation par fusion totale

3.6 Le modèle XML d'agrégation

Pour faciliter l'implémentation des patrons d'agrégation proposés, une représentation à l'aide d'un langage indépendant des plateformes technologiques s'impose. Nous avons choisi le langage XML pour représenter de façon structurée les différents modèles d'agrégations. Avec ce format, les développeurs intégrateurs peuvent utiliser le langage de programmation approprié à leur plateforme technologique pour transformer la composition de l'agrégat et ses interactions en appel de méthode dans le langage hôte choisi.

Il est important de rappeler que nous n'implantons pas ce modèle XML, étant donné que l'implantation des agrégats est hors portée dans cette thèse. Nous le discutons ici pour montrer que les patrons que nous proposons ne sont pas uniquement des modèles conceptuels, mais ils peuvent être utilisés pour une implémentation réelle suite à une transformation XML à partir du modèle des composants représenté en UML 2.0. Le résultat de la transformation est un document XML qui décrit la composition d'un agrégat ainsi que les interactions entre les composants agrégés. Le modèle suivant concrétise un exemple d'un document XML d'un agrégat logiciel.

```
< AggregateComponent>
<components>
<component>
      <component-name>Component 1</component-name>
      <services>
            <provided-services>
      <provided-service id=> service P1 </provided-service>
      <provided-service> service P2 </provided-service>
            </provided-services>
            <required-services>
                  < required-service> service R1 </ required-service>
            </ required-services>
      </services>
      </component>
      <component>
            <component-name>Component 2</component-name>
            <services>
                  <provided-services>
                        <provided-service>   service   P3   </provided-
                  service>
                        <provided-service>   service   P4   </provided-
                  service>
                  </provided-services>
                  <required-services>
                        <required-service>   service   R2   </required-
                  service>
                  </required-services>
            </services>
      </component>
</components>
<components-interactions>
            <interaction name=''Interaction 1''>
                  <source>
                  <service-flow>
                        <service refid=> component service 1</service>
                        <service> component service 2</service>
                  </service-flow>
      </interaction>
            <interaction name=''Interaction 2''>
                  <service-flow>
                        <service> component service 1</service>
                        <service> component service 2</service>
                  </service-flow>
      </interaction>
</components-interactions>
</AggregateComponent>
```

Figure 3.16 Exemple d'un document XML d'un agrégat de composants

3.7 Conclusion

La proposition des métadonnées SOCOM est une classification des composants logiciels qui facilite la recherche, la sélection des composants ainsi que l'analyse de faisabilité de leur

agrégation. La mise en équation de quelques attributs d'agrégation de SOCOM a permis de définir quatre classes d'agrégation des composants. Ces classifications permettent d'assister un architecte durant la phase d'analyse d'un processus de développement logiciel à base de composants. En effet, après l'identification des composants à agréger, la classification de l'agrégation de ces composants facilite la phase de conception et d'implantation. En appliquant les équations des quatre classes d'agrégation, l'architecte décide laquelle des classes d'agrégation convient le mieux à son contexte, pour pouvoir ensuite appliquer un des patrons associés. Les choix de la classe d'agrégation et du patron associé offrent à l'architecte une manière de structurer les composants à agréger et leur interaction selon un modèle prédéfini par le patron. Ceci représente un gain en temps et en productivité dans une perspective de développement par la réutilisation.

Il est important de rappeler qu'avec ces classes d'agrégation et les modèles des patrons associés, nous avons défini un ensemble de modèles pour renforcer le volet « modélisation » du processus d'agrégation cible dirigé par les métadonnées et les modèles. Ce processus est présenté en détail dans les deux chapitres suivants.

Le tableau suivant décrit l'apport des classifications des agrégats, comme résultat de recherche, aux phases d'un processus de développement à base de composants.

Tableau 3.7 Matrice des apports de la classification des agrégats
aux phases du processus de développement à base de composants

Résultats de la thèse	**Description des contributions des résultats par phase**
Classification des agrégats	**Phase de recherche :** - La classification ajoute une nouvelle option de recherche. L'architecte peut rechercher les agrégats d'une même classe ou, connaissant les composants participants du nouvel agrégat, il peut rechercher toutes les agrégations contenant un de ses composants participants. Les métadonnées des composants agrégats trouvées peuvent aider l'architecte dans le processus de développement du nouvel agrégat. **Phase d'adaptation:** - En sélectionnant une classe d'agrégation, le modèle de la classification oriente l'architecte dans la conception du composant agrégat et dans l'adaptation des composants participants. **Phase d'intégration:** - En sélectionnant une classe d'agrégation, le modèle de la classification oriente l'architecte sur la manière de coder l'agrégation.
Patrons d'agrégat	**Phase d'adaptation:** - Les patrons proposent à l'intégrateur une structure sur les interactions du composant agrégat avec les composants participants. L'intégrateur peut l'utiliser pour adapter ses composants au contexte de l'agrégation. **Phase d'intégration:** - Le choix du patron durant la phase d'adaptation aide l'intégrateur dans l'implémentation de l'agrégat. Il peut réutiliser un patron de code ou en développer un nouveau qui pourrait être réutilisable ultérieurement. (composant logique vs composant physique, fusion totale vs fusion en partie, coordination fédérée vs coordination répartie).

CHAPITRE 4

PROCESSUS D'AGRÉGATION DES COMPOSANTS LOGICIELS

Ce chapitre a pour objectif la description du processus d'agrégation des composants logiciels basé sur les modèles et les métadonnées SOCOM, que nous avons intitulé SOCAP (SOftware Component Aggregation Process). À l'aide des métadonnées SOCOM, de la classification des agrégations des composants et des modèles associés, nous avons développé un processus d'agrégation en réutilisant les phases classiques des processus de développement à base des composants (voir section 1.7) que nous avons bonifié par l'ajout d'une autre phase. Le processus proposé repose sur les métadonnées et les modèles définis dans les deux chapitres précédents.

Dans ce chapitre, le processus d'agrégation produit un composant de type agrégat. L'agrégat étant lui aussi un composant logiciel, nous le décrivons avec les métadonnées SOCOM. Cependant, pour capter ses propriétés d'agrégation, nous proposons des métadonnées de l'agrégat que nous présentons dans la section 4.8 avant la description du processus SOCAP décrit dans la section 4.9. Ces métadonnées sont intitulées SOCAM (**SO**ftware **C**omponent **A**ggregate **M**etadata).

Cette description comprend les principes, les acteurs, les hypothèses, les contraintes et une spécification générale des grandes étapes et des activités principales associées. Ce processus d'agrégation des composants est élaboré dans la cinquième étape de la démarche de notre méthodologie de recherche. La figure suivante positionne le présent chapitre dans le cheminement global de la thèse.

Figure 4.1 La méthodologie de recherche de la thèse – Étape 5 : processus d'agrégation

Pour développer un composant par agrégation, le processus définit des activités de référencement des composants sources, de recherche, de sélection, d'évaluation de la faisabilité de l'agrégation, de conception de l'agrégat, d'implémentation et de test jusqu'au référencement du nouveau composant agrégat dans l'entrepôt SOCOM. La description fournie ci-dessous se concentre sur le premier niveau des activités du processus. Les niveaux plus détaillés sont fournis dans d'autres chapitres et en annexe.

4.1 Objectifs du processus SOCAP

Le processus SOCAP vise essentiellement les objectifs suivants:

- promouvoir le développement par la réutilisation;
- faciliter la recherche et la navigation multicritère des composants logiciels dans les organisations;
- permettre l'évaluation de la faisabilité de l'agrégation avant d'entamer effectivement le développement de l'agrégat;
- détecter les agrégations non faisables tôt dans le processus de développement à base de composants;
- concevoir le composant agrégat en se basant sur une classification supportée par des modèles de composants;

- rendre visible la dépendance entre les composants agrégats et les composants agrégés à l'aide des métadonnées de l'agrégat SOCAM (des métadonnées de l'agrégat détaillées plus loin dans la section 4.8);
- supporter de bout en bout l'agrégation des composants logiciels depuis la définition du contexte de l'agrégat jusqu'à sa mise en production.

4.2 Positionnement de SOCAP avec des processus connexes

Les processus de développement à base des composants logiciels comprennent généralement les phases suivantes (Cheesman & Daniels, 2001): la recherche, la sélection, l'adaptation et l'intégration de composants.

Figure 4.2 Différentes phases d'un processus d'intégration de composants logiciels

Les auteurs de ces processus ont pour objectif principal d'assister un expert dans l'agrégation des composants et poussent davantage vers l'automatisation de la majorité des étapes du processus. Nous rappelons ici les références des auteurs et les phases du processus où ils manifestent le plus leur contribution:

- (Mikhail, 2004) – la phase d'intégration;
- (Renaux, Olivier, & Jean-Marc, 2004) – la phase d'intégration;
- (Villalobos, 2003) – la phase de sélection;
- (Yakimovich, Travassos, & Basili, 1999) – la phase de sélection;
- (Cheesman & Daniels, 2001)via CATALYSIS – les phases : recherche, sélection, adaptation;

- (Szyperski, Gruntz, & Murer, 2002) avec UML component – les phases : recherche, sélection, adaptation.

SOCAP est un processus compatible avec les quatre phases classiques d'un processus de développement à base de composants logiciels. En effet, il enrichit ces phases par des métadonnées, des règles, des classifications, des modèles de l'agrégat, des patrons et des sous processus. En plus de bonifier chaque phase par des activités particulières, SOCAP propose une phase d'évaluation de la faisabilité d'agrégation. Cette phase représente une des contributions originales de nos travaux parce qu'elle décide de l'agrégation et du comportement du processus dans les phases subséquentes suite à la décision de la faisabilité de l'agrégation. Cette phase est introduite sommairement dans ce chapitre et sera élaborée en détail dans le chapitre suivant. La figure suivante présente la composition des phases de SOCAP.

Figure 4.3 Les phases du processus SOCAP

4.3 Rôles et responsabilités des acteurs principaux

Les principaux rôles considérés comprennent un architecte de composants, un directeur de projet, un intégrateur de composant, un développeur de composants, un responsable de l'assurance qualité et un responsable de la gestion de configuration. Ces rôles peuvent être assistés, au besoin, par des rôles secondaires tels ceux de responsable de plateformes, de responsable de sécurité, de responsable réseau, de responsable des bases de données.

4.3.1 Architecte de composant

L'architecte de composant a les responsabilités suivantes dans le processus SOCAP:

- référencer des composants logiciels et les composants agrégats dans l'application SOCOM en se basant sur les spécifications SOCOM;
- identifier des composants logiciels;
- rechercher et sélectionner les composants logiciels à agréger;
- évaluer l'agrégat potentiel;
- concevoir le composant agrégat.

4.3.2 Directeur de projet

Le directeur du projet d'agrégation a les responsabilités suivantes dans le processus SOCAP:

- définir sommairement le contexte du projet d'agrégation;
- approuver le budget alloué au développement de l'agrégat;
- participer dans le volet financier de l'évaluation de la faisabilité de l'agrégation;
- suspendre ou arrêter le développement de l'agrégat en cas de dépassement des budgets ou de changement de la stratégie de l'organisation envers le projet.

Il importe de rappeler que le directeur de projet a un rôle de gestion et de planification dans les projets de développement de composant par agrégation. Toutefois, nous nous intéressons davantage à son rôle décisionnel. Nous ne mentionnons que sa décision/approbation (du point de vue financier) du budget alloué durant l'évaluation de la faisabilité de l'agrégat.

4.3.3 Intégrateur de composant

L'intégrateur de composant a les responsabilités suivantes dans le processus SOCAP :

- proposer des techniques d'agrégation à partir d'une base de connaissances prédéfinie. Cette base devra être maintenue et enrichie de façon récursive et incrémentale;

- proposer des techniques d'adaptation des composants logiciels à agréger;
- développer les interactions entre les composants participants pour construire le composant agrégat;
- autoriser le déploiement du composant agrégat en test.

4.3.4 Développeur de composant

Le développeur de composant a les responsabilités suivantes dans le processus SOCAP:

- adapter les composants sources pour satisfaire le contexte d'agrégation;
- programmer des composants intermédiaires (scripts / ponts / médiateurs / connecteurs / etc.) pour construire le composant agrégat;
- ajuster la configuration des composants sources pour satisfaire le contexte et les besoins d'agrégation.

4.3.5 Responsable de l'assurance qualité

Le responsable de qualité a les responsabilités suivantes dans le processus SOCAP:

- tester le composant agrégat;
- tester les composants à agréger individuellement à la demande de l'architecte;
- s'assurer de la conformité du composant agrégat aux exigences d'affaires;
- vérifier et valider des contraintes non fonctionnelles en faisant appel aux responsables ayant des rôles secondaires décrits dans la section suivante.

4.3.6 Responsable de la gestion des configurations

Le responsable de la gestion des configurations assure les responsabilités suivantes dans le processus SOCAP :

- identifier la liste des paramètres de configuration des environnements des composants ou de l'agrégat;

- configurer les environnements des composants participants ou de l'agrégat;
- déployer l'agrégat dans les environnements de test et de production.

4.4 Rôles et responsabilités des acteurs secondaires

Les rôles principaux seront assistés, au besoin, par des rôles secondaires. Ces rôles apportent une expertise pour remédier aux incompatibilités des plateformes des composants logiciels sources. Ces rôles ne sont pas dans la portée de la présente étude et représentent chacun un sujet indépendant qui requiert une analyse approfondie pour décrire leur dépendance forte avec l'agrégation des composants logiciels. Voici un aperçu de ces rôles et des leurs responsabilités associés.

4.4.1 Responsable des infrastructures

Le responsable des infrastructures joue le rôle de conseiller auprès de l'équipe d'affaires et technique et a les responsabilités suivantes dans le processus SOCAP :

- évaluer la compatibilité des infrastructures (serveur / système d'exploitation / environnement) des composants sources;
- évaluer la ou les infrastructures physiques du composant agrégat;
- conseiller sur les dépendances/configurations requises des composants sources/agrégat dans les infrastructures qui les hébergent.

4.4.2 Responsable des bases de données

Le responsable des bases de données joue le rôle de conseiller auprès de l'équipe d'affaires et technique. Il a les responsabilités suivantes dans le processus SOCAP :

- identifier les contraintes d'agrégation des données des composants sources;
- évaluer au besoin l'interopérabilité des plateformes de données des composants sources;
- évaluer la ou les plateformes de données du composant agrégat.

4.4.3 Responsable des réseaux

Le responsable des réseaux joue le rôle de conseiller auprès de l'équipe d'affaires et technique. Il a les responsabilités suivantes dans le processus SOCAP :

- évaluer l'interopérabilité des infrastructures des réseaux des composants sources;
- évaluer l'interconnexion des infrastructures des réseaux des composants sources;
- identifier les contraintes des réseaux qui hébergent chacun des composants sources.

4.4.4 Responsable de la sécurité

Le responsable de la sécurité joue le rôle de conseiller auprès de l'architecte et l'intégrateur de composants. Il a les responsabilités suivantes dans le processus SOCAP:

- identifier les contraintes de sécurité pour l'agrégation entre les composants sources;
- approuver la sécurité du composant agrégat (certifier le composant agrégat).

4.5 Les principes du processus SOCAP

Le processus SOCAP respecte les principes suivants :

- Le processus réutilise quatre étapes suivantes du processus d'agrégation : recherche, sélection, adaptation et intégration.
- La phase d'évaluation de la faisabilité est primordiale et elle s'exécute après les phases de recherche et de sélection, avant les phases d'adaptation et d'intégration des composants logiciels.
- La recherche et l'analyse des composants se base sur les métadonnées SOCOM et SOCAM.
- Le processus utilise uniquement les composants qui sont référencés avec les métadonnées SOCOM. Pour les composants non SOCOM, un référencement dans le gestionnaire SOCOM est requis pour l'ajout du composant dans l'entrepôt SOCOM.

- Une fois développé, l'agrégat est référencé à l'aide des spécifications SOCOM en tant que nouveau composant et des spécifications SOCAM en tant que composant agrégat.

4.6 Les hypothèses du processus SOCAP

Les hypothèses suivantes doivent être vérifiées avant d'utiliser le processus SOCAP:

- L'équipe de projet doit disposer d'une description minimale du contexte d'agrégation (voir les spécifications SOCAM, section 4.8).
- L'étude de faisabilité se base sur le processus présenté dans le chapitre 4. Toutefois, les utilisateurs peuvent étendre ou proposer une nouvelle version du processus d'évaluation de faisabilité si le domaine d'affaires l'impose.
- Les métadonnées SOCOM des composants logiciels sources sont conformes aux propriétés des composants réels.
- Les acteurs qui manipulent le processus doivent avoir les compétences nécessaires et suffisantes en architecture logicielle et d'entreprise pour réaliser les activités des cinq phases du processus.

4.7 Modèle BPMN du processus SOCAP

Le processus SOCAP est modélisé avec le formalisme BPMN. BPMN utilise un ensemble de symboles graphiques pour représenter les processus :

- les couloirs pour encadrer les activités d'un acteur particulier;
- un carré avec des coins ronds pour les activités;
- les cercles pour identifier le début et la fin du processus;
- une flèche orientée continue pour un flux de séquence entre deux activités d'un même couloir;
- une flèche orientée discontinue pour décrire un flux de message entre deux activités appartenant à deux couloirs différents;

- un losange pour présenter un contrôle de décision permettant la création ou la jonction des flux parallèles.

Ces éléments de bases du formalisme BPMN aident à la compréhension du modèle de notre processus SOCAP. Pour plus d'information sur BPMN et son utilisation et les outils disponibles, consulter le site officiel des standards de l'OMG[23].

[23]BPMN : Site official du groupe OMG: http://www.omg.org/bpmn/

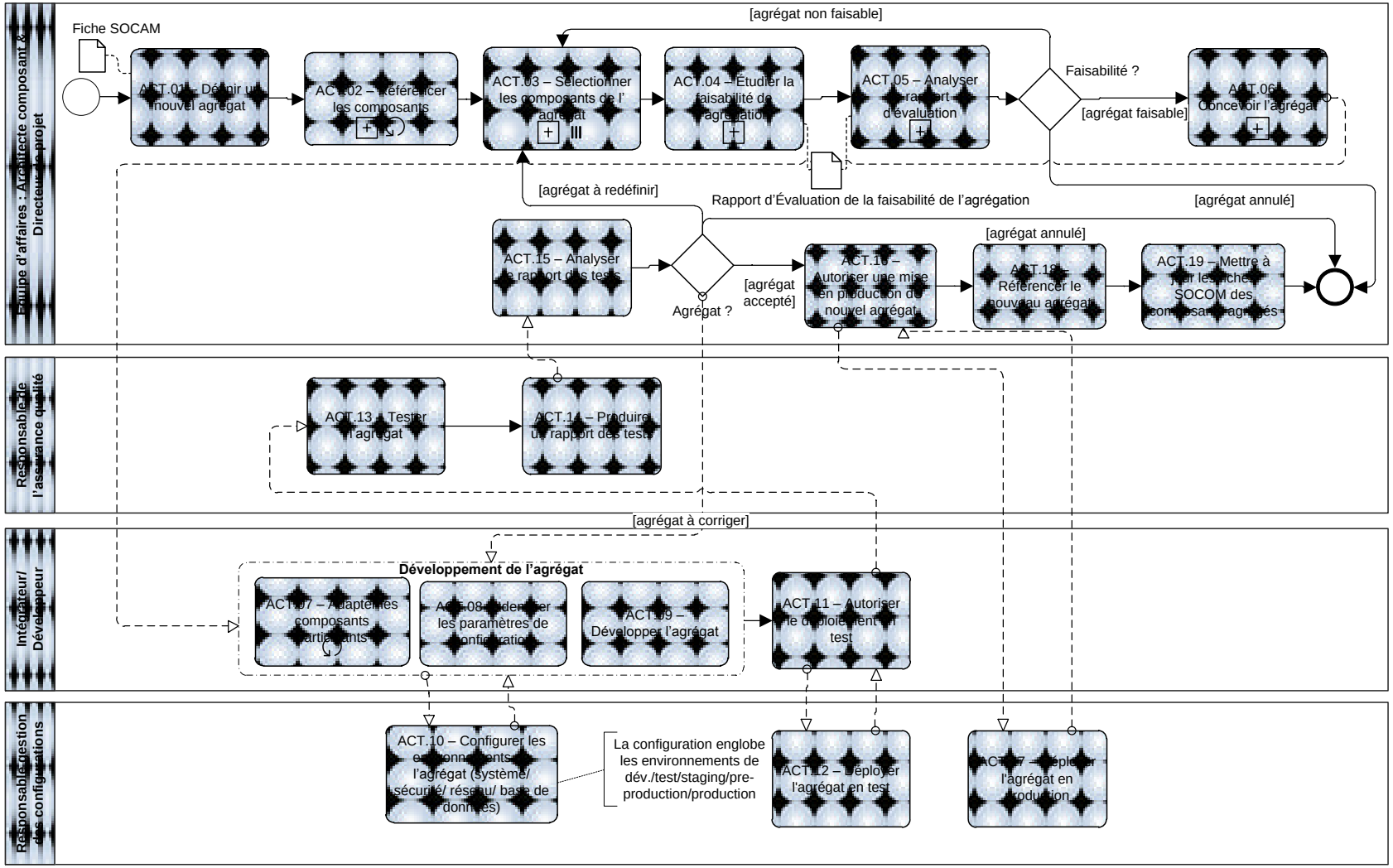


Figure 4.4 Processus d'agrégation de composants logiciels – SOCAP

4.8 Les métadonnées de l'agrégat

À la fin du processus d'agrégation des composants, l'agrégat est aussi référencé par les métadonnées SOCOM comme un nouveau composant. Toutefois, ceci n'est pas suffisant et n'informe pas sur l'aspect composé du composant. Par définition, SOCOM ne décrit pas les attributs spécifiques de l'agrégat. De ce fait, nous avons jugé nécessaire de définir une structure complémentaire à SOCOM qui représente les métadonnées de l'agrégat, soit SOCAM (SOftware Component Aggregate Metadata). SOCAM décrit les propriétés générales de l'agrégat, sa structure statique et son comportement dynamique suite à l'agrégation d'au moins deux composants logiciels. De plus, ces métadonnées décrivent les éléments du contrat d'agrégation (déjà introduit dans la section 1.8) tels que les hypothèses, les contraintes, les exigences d'affaires et non fonctionnels de l'agrégat. Chaque agrégat est décrit par les attributs suivants :

- Identifiant de l'agrégat **<aggregateIdentifier>[1]**: l'identifiant système unique de l'agrégat (nom système de l'agrégat). Cet attribut est couvert par la catégorie « staticSpecification » de SOCOM.
- Nom de l'agrégat **<aggregateName>[1]** : l'identifiant logique du composant agrégat. Cet attribut est couvert par la catégorie « staticSpecification » de SOCOM.
- Auteur **<author>[1..*]**: un ou plusieurs auteurs de l'agrégat. Cet attribut est couvert par la catégorie « staticSpecification » de SOCOM.
- Description Générale **<briefDescription>[1]**: description générale du composant agrégat, du contexte d'agrégation et des fonctionnalités de haut niveau attendues. Cet attribut est couvert par la catégorie « staticSpecification » de SOCOM
- Hypothèses **<hypothesis>[0..*]**: hypothèses reliées à l'agrégat. Elles seront prises en considération pour justifier la définition et les compositions statique et dynamique de l'agrégat.
- Contraintes **<constraints>[0..*]**: contraintes reliées à l'agrégat. Elles seront prises en considération pour la définition, l'évaluation et le développement de l'agrégat.

- Cas d'affaires **<businessCases>[0..*]**: cas d'affaires qui définissent les besoins d'affaires du nouveau composant agrégat. Cet attribut est couvert par la catégorie « businessSpecification » de SOCOM.
- Potentiel de couplage **<coupling>[1]:** traduit la densité des interactions et des dépendances des composants entre eux lors de leur déploiement ou exécution. Nous distinguons deux valeurs de couplage des composants, fort ou faible. La notion de couplage est un concept connu du génie logiciel. Il sera aussi utile dans l'évaluation de la faisabilité de l'agrégation. Les valeurs "faible" et "fort" sont décidées suite à une évaluation par un architecte/intégrateur du nombre d'interactions des composants participants à l'agrégation. Avec la valeur de cet attribut, nous pouvons décider d'annuler l'agrégation vue que l'agrégat sera produit avec un couplage fort, ce qui représente une mauvaise conception dans beaucoup de cas.
- Catégorie de l'agrégation **<aggregationCategory>[1]**: l'une de connexion, collection, coordination ou fusion.
- Patron d'agrégation **<aggregationPattern>[1..*]**: dépend de la catégorie d'agrégation adoptée. Il y a au moins un patron d'agrégation par catégorie d'agrégation (connexion, collection, coordination ou fusion – section 3.3).
- Composition statique **<staticComposition>[1..*]**: ensemble des composants participants dans l'agrégation.
- Composition dynamique **<dynamicComposition>[1..*]**: nom des interactions entre composants sources. Il s'agit d'une description sommaire des services fournis du composant agrégat avec une description des données en entrées et en sorties pour chacun d'eux.
- Exigences non fonctionnelles **<nonFunctionalRequirements>[1..*]**: on distingue différents types d'exigences non fonctionnelles (nommées aussi exigences de qualité) telles que la performance, la version, la licence, la fiabilité, etc. Cet attribut est couvert par la catégorie « nonFunctionalSpecification » de SOCOM
- Autres **<others>[1..*]**: propriétés additionnelles qui s'ajoutent pour la description du composant agrégat.

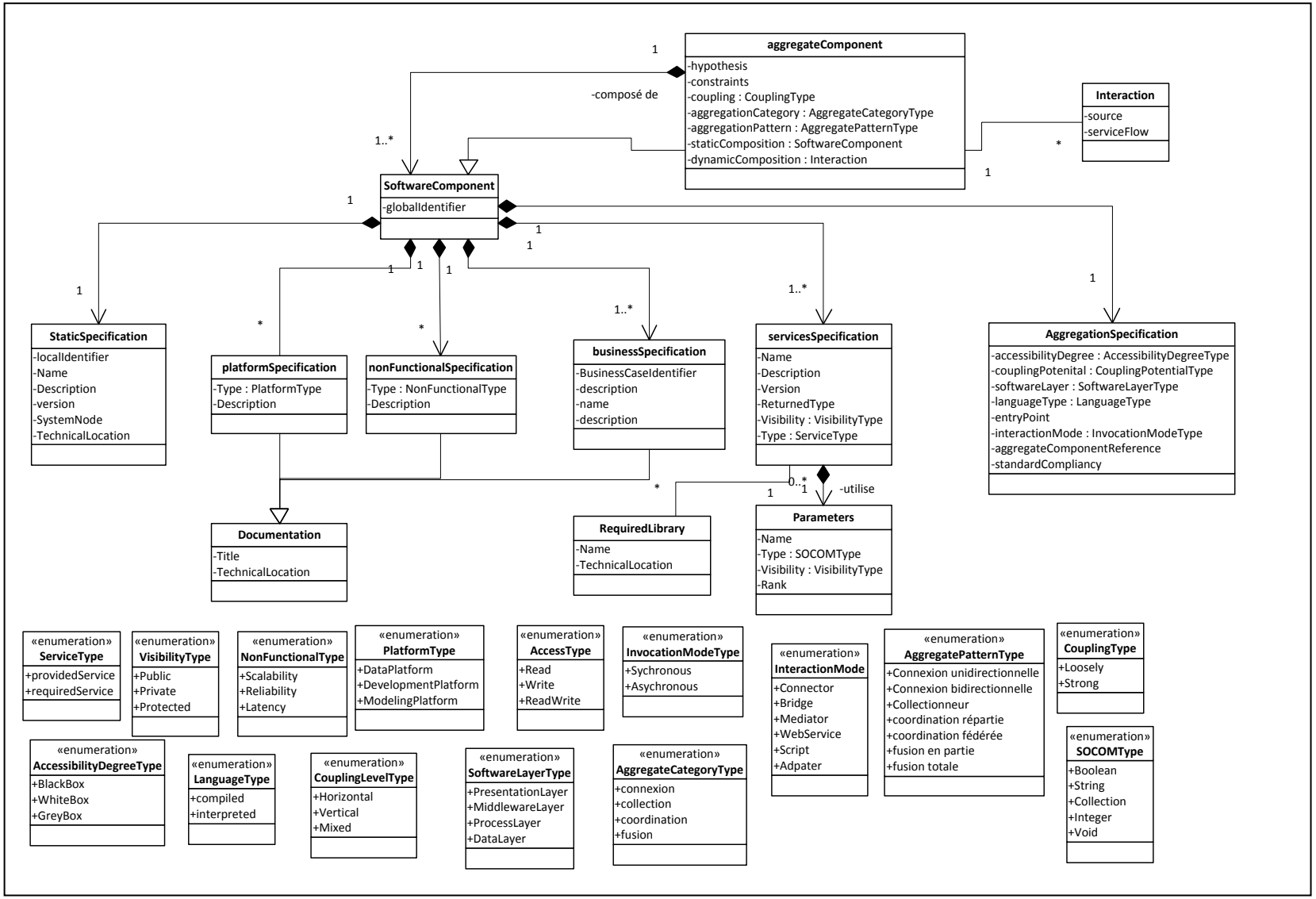


Figure 4.5 Modèle conceptuel des spécifications SOCOM avec les spécifications SOCAM (diagramme de classes UML)

4.9 Description des activités et des étapes principales du processus

À la fin de chaque activité/étape, nous rappelons l'état de l'agrégat (initial, projet, évalué, conceptuel, etc.). Ces états sont utiles essentiellement pour le suivi des projets d'agrégation lorsque ces derniers sont intégrés dans des outils d'exécution des processus d'affaires ou de suivi. En effet, si le processus SOCAP est intégré dans un système d'exécution de processus BPMN tel qu'IBM Lombardi Teamworks[24], ces états aideront à définir des notifications pour informer les différentes intervenants dans le projet de l'état courant de l'agrégat et des activités nécessaires pour poursuivre l'agrégation. De plus, lorsque l'état de l'agrégat change certains utilisateurs n'ont plus le droit de changer certains attributs de la fiche SOCAM. Par exemple, lorsque l'agrégat est évalué de « agrégat faisable », il passera à l'étape de conception, par conséquent la composition statique ne peut pas changer. Une fois conçu, l'agrégat est qualifié de « agrégat conceptuel »

4.9.1 ACT.01 – Définir le nouvel agrégat

L'architecte de composant crée un nouveau projet d'agrégation dans lequel il spécifie le nom de l'agrégat et le contexte de l'agrégation. À partir de ce contexte, il identifie et spécifie les exigences d'affaires du composant agrégat. Si la composition de l'agrégat est connue d'avance, l'acteur précisera la liste des composants qui participeront à l'agrégation. Les informations de l'agrégat sont décrites à l'aide des spécifications SOCAM. L'agrégat est qualifié de « Agrégat initial ».

4.9.2 ACT.02 – Référencer les composants participants

L'architecte de composant prépare l'agrégation en référençant les composants participants par les métadonnées SOCOM. Ces composants, une fois ajoutés dans l'entrepôt SOCOM,

---

[24] IBM Lombardi Teamworks: un outil de modélisation et d'exécution de processus BPMN voir http://www-01.ibm.com/software/info/bpm/

deviennent des composants SOCOM. Ils sont des composants potentiels qui participeront aux activités de recherche, de sélection et ensuite à l'évaluation de l'agrégation. Cette activité n'est pas requise si tous les composants participants dans l'agrégation sont des composants SOCOM. L'agrégat est qualifié de « Agrégat projet ».

4.9.3 ACT.03 – Sélectionner les composants de l'agrégat

L'architecte de composant sélectionne les composants de l'agrégation. Si la composition de l'agrégat n'est pas encore définie, l'acteur cherchera avec des critères appropriés les composants qui satisferont le contexte d'agrégation. L'acteur modifie la fiche SOCAM du composant potentiel agrégat en mettant à jour la liste des exigences non fonctionnelles de l'agrégat. Ce dernier est qualifié de « Agrégat défini en partie ».

4.9.4 ACT.04 – Étudier la faisabilité d'agrégation des composants

L'équipe de projet évalue l'agrégation en se basant sur les métadonnées (statiques, affaires, qualité, plateformes, services et agrégation) des composants participants. Elle identifie les incompatibilités potentielles en exécutant le processus d'évaluation d'agrégation des composants logiciels décrit dans le chapitre 5. L'agrégat est alors qualifié d'« Agrégat évalué ».

4.9.5 ACT.05 – Analyse du rapport de faisabilité de l'agrégation des composants

L'équipe de projet analyse la liste des incompatibilités détectées dans le rapport d'évaluation de l'agrégation. Elle procède à la classification des incompatibilités. Elle précise la possibilité de résolution (oui/non) de l'incompatibilité en spécifiant la catégorie de résolution et la solution proposée. La décision est basée sur la possibilité de résolution des incompatibilités et de l'approbation de la direction (directeur de projet) pour aller de l'avant dans le processus d'agrégation

4.9.6 ACT.06 – Conception de l'agrégat

L'architecte de composant conçoit la composition dynamique de l'agrégat. Il définit la liste des interactions entre les composants participants en se basant sur leurs services fournis, leurs services requis et les attributs de qualité des composants participants ainsi que de l'agrégat. Il pratique la catégorisation de l'agrégation pour évaluer la possibilité d'appliquer un des patrons d'agrégation: la connexion, la collection, la coordination et la fusion. Le choix d'un des patrons n'est pas obligatoire si l'agrégat ne satisfait pas les conditions de son application. L'agrégat est qualifié de « Agrégat conceptuel ».

4.9.7 Étape de développement l'agrégat

Cette étape du processus inclut les activités suivantes :

- ACT.07 – adapter des composants participants;
- ACT.08 – identifier les paramètres de configuration;
- ACT.09 – développer l'agrégat.

L'intégrateur étudie la composition statique, la composition dynamique et les exigences non fonctionnelles de l'agrégat. Ensuite, il propose des techniques d'agrégation pour implanter les interactions entre les composants participants de l'agrégat conceptuel. Il adapte les composants participants conformément aux techniques d'agrégations choisies. Ensuite, il utilise ces composants adaptés pour développer le composant agrégat.

En parallèle, il identifie les configurations requises pour la mise en place des composants adaptés et du composant agrégat. Il définit essentiellement les paramètres associés aux environnements de test et de production de l'agrégat ainsi que ceux des volets de sécurité, du réseau et des infrastructures.

Si cette étape est exécutée après l'étape de test dans une nouvelle itération, elle consiste à ajuster les composants déjà adaptés pour résoudre les anomalies identifiées. À la fin de cette étape, l'agrégat est qualifié de « Agrégat développé ».

4.9.8 Étape de configuration des environnements

En parallèle avec l'étape de développement de l'agrégat, le Responsable de la Gestion des Configurations (RGC) prépare les environnements de test et de production de l'agrégat concernant les volets de sécurité, du réseau et des infrastructures (ACT.10 du modèle SOCAP de la Figure 4.4). Il utilisera les paramètres identifiés durant l'étape de développement. À la fin de cette étape, l'agrégat demeure « Agrégat développé ».

4.9.9 Étape de déploiement de l'agrégat

Cette étape comprend les activités de déploiement dans les environnements de test et de production. En premier lieu, le responsable de la gestion des configurations procède à un déploiement de l'agrégat dans l'environnement de test. Il doit recevoir, au préalable, l'autorisation de l'intégrateur (ACT.11 du modèle SOCAP, Figure 4.4). En deuxième lieu, il déploie l'agrégat dans un environnement de production, si les tests sont acceptés et après avoir reçu au préalable l'autorisation écrite de l'équipe d'affaires (ACT.12 du modèle SOCAP, Figure 4.4). À la fin de cette étape, l'agrégat est qualifié de « Agrégat déployé en test/production».

4.9.10 Étape de test de l'agrégat

Le Responsable de l'Assurance Qualité (RAQ) prépare un plan de tests incluant les scénarios d'essais nécessaires. L'agrégat est déjà déployé dans un environnement de test. Le RAQ teste le composant agrégat en se basant sur les scénarios d'essais et en suivant le plan de tests déjà élaboré. Il doit s'assurer que l'agrégat satisfait les exigences d'affaires de l'agrégat. Il produit un rapport des résultats des tests réalisés. Il communique ensuite ce rapport à l'équipe

d'affaires composée du directeur de projet et de l'architecte qui analysera les anomalies et les commentaires associés aux cas de tests. Cette équipe formule une décision concernant la mise en production ou non de l'agrégat. Dépendamment des résultats de l'analyse, elle peut:

- demander la résolution des anomalies par l'intégrateur (redirection vers l'étape de développement de l'agrégat, section 4.9.7); autoriser la mise en production de l'agrégat (vers l'activité ACT.16);
- annuler le processus d'agrégation suite à une décision de la direction de ne pas entreprendre la résolution des anomalies (raisons financières associées aux coûts de résolution et de gestion).

À la fin de cette étape, l'agrégat est qualifié d'« Agrégat testé ». Aussi, si les tests sont acceptés, l'agrégat est qualifié d'« Agrégat accepté », sinon il sera « Agrégat annulé » ou « Agrégat en attente de résolution »

4.9.11 Étape de référencement du nouvel agrégat

L'architecte de composant ajoute une nouvelle fiche du nouveau composant SOCOM (composant agrégat, par l'activité ACT.18) et met à jour les métadonnées SOCOM des composants impliqués dans l'agrégation (composant agrégés, par l'activité ACT.19).

Le processus SOCAP peut être complété selon les cas suivants :

- processus annulé depuis le processus d'évaluation de faisabilité l'agrégat;
- processus annulé à cause d'un grand nombre d'anomalies dont la résolution est coûteuse;
- processus terminé avec succès.

À la fin du processus, l'agrégat est qualifié d'« Agrégat réalisé » ou d'« Agrégat non réalisable».

4.10 Conclusion

Ce chapitre a présenté les activités du processus d'agrégation de composants logiciels (SOCAP). Cette vue du processus identifie les acteurs du processus, les activités associées, les contraintes, les hypothèses et les livrables qui en résultent. Dans le chapitre 7, des exécutions de ces activités dans des contextes réels d'agrégation valident et consolident le processus SOCAP. Un modèle et des métriques de validation du processus sont aussi élaborés.

Le tableau suivant décrit l'apport du processus SOCAP et des métadonnées SOCAM, comme résultats de recherche, aux phases d'un processus de développement à base de composants.

Tableau 4.1 Matrice des apports de SOCAP aux phases
du processus de développement à base de composants

Résultats de la thèse	Description des contributions des résultats par phase
Processus SOCAP	**Phase de recherche :** - SOCAP propose une séquence d'activités pour aider un architecte à référencer et à rechercher un composant par les métadonnées SOCOM. **Phase de sélection :** - SOCAP propose une séquence d'activités pour aider un architecte à la sélection des composants à agréger par les métadonnées SOCOM. **Phase de faisabilité :** - SOCAP propose une phase inédite pour évaluer la faisabilité d'agrégation des composants avant d'entamer les phases d'adaptation et d'intégration.
Métadonnées SOCAM	**Phase de recherche :** - SOCAM facilite la recherche de composants ou d'agrégat similaires à l'aide des métadonnées de l'agrégat - SOCAM rend visible un agrégat et les liens avec les composants agrégés. **Phase de sélection :** - SOCAM facilite la sélection quand le nouvel agrégat présente des similarités avec des agrégats existants. **Phase de faisabilité :** - SOCAM sert de données de référence pour évaluer la faisabilité d'agrégation. Elles seront comparées avec les données SOCOM des composants sources pour identifier les incompatibilités potentielles. **Phase d'adaptation:** - Les spécifications fonctionnelles, non fonctionnelles et des services de SOCAM aident l'intégrateur lors de la conception et l'adaptation des composants à agréger. **Phase d'intégration:** - Les spécifications fonctionnelles, non fonctionnelles et des services de SOCAM aident l'intégrateur durant le codage et les tests du composant agrégat.

CHAPITRE 5

PROCESSUS D'ÉVALUATION DE LA FAISABILITÉ DE L'AGRÉGATION DES COMPOSANS LOGICIELS

Ce chapitre a pour objectif la description du sous-processus de SOCAP (SOftware Component Aggregation Process) concernant l'évaluation de la faisabilité d'agrégation des composants logiciels. Ce sous-processus est une activité de premier niveau du processus SOCAP. La section suivante décrit un ensemble de règles d'agrégation à l'aide d'exemples. Ces règles expriment la logique proposée pour évaluer la faisabilité de l'agrégation. Après l'exécution de ces règles dans un contexte d'agrégation, l'agrégat se trouve avec un des états suivants : ***faisable***, ***non faisable*** ou ***annulé***.

Ce sous-processus est une contribution originale importante de nos travaux de recherche parce qu'il introduit une phase souvent absente dans les processus de développement à base de composant (voir section 1.7). Cette phase décide du cheminement du reste des activités du processus. Elle est élaborée dans la sixième étape de la démarche de notre méthodologie de recherche. La figure suivante positionne le présent chapitre dans le cheminement global de la thèse.

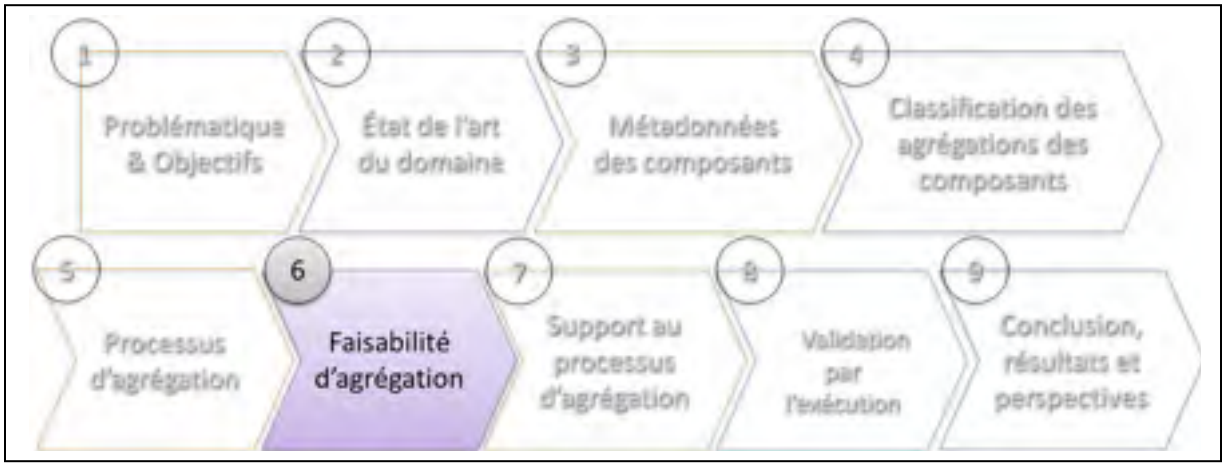


Figure 5.1 La méthodologie de recherche de la thèse –
Étape 6 : Faisabilité d'agrégation

5.1 Importance de l'évaluation de la faisabilité

Avant d'investir dans l'agrégation effective, une évaluation de la faisabilité s'avère utile. En identifiant les incompatibilités entre les composants sélectionnés tôt dans le processus d'agrégation, l'équipe de projet économisera les coûts des activités d'agrégation autour de la gestion, de l'adaptation et de l'intégration inutiles. Ce processus est souvent négligé dans les méthodologies de développement à base de composants et celles du développement pour et avec la réutilisation telles que CUP, UML Component et Catalysis (voir liste des processus dans la section 1.7). De plus Mili *et al.* rappellent que les volets techniques et technologiques de l'intégration des composants logiciels sont assez matures et recommandent davantage les travaux de recherche autour des volets méthodologique et économique (Mili, Mili, Yacoub, & Addy, 2002). Par ce processus et les spécifications SOCOM, nous voulons rappeler que l'agrégation des composants logiciels ne repose pas uniquement sur des considérations techniques ou technologiques, mais aussi des considérations d'affaires, conceptuelles, non fonctionnelles et de plateformes. Aussi, ce sous processus et SOCAP représentent notre contribution dans l'amélioration du volet processus/méthodologie dans les domaines de développement à base de composants logiciels et du développement pour et par la réutilisation.

5.2 Les règles d'agrégation des composants logiciels

Le sous-processus d'évaluation d'agrégation des composants logiciels est composé de six règles d'agrégation pour s'assurer de la faisabilité de l'agrégat. Pour simplifier la compréhension des règles, on suppose deux composants, CpA et CpB, qui participeront au développement par agrégation d'un composant agrégat CpC.

5.2.1 Règle zéro d'agrégation : règle financière (RA-00)

Chaque composant est décrit par un ensemble de métadonnées métiers qui sont l'équivalent des cas d'utilisation de l'approche de développement logiciel RUP, mais décrits à un niveau d'affaires. La règle financière est **vérifiée** si et seulement si la direction de la compagnie décide d'allouer un budget suffisant pour évaluer le développement d'un nouveau composant par agrégation. Si ce budget est approuvé la règle financière **RA-00** est **vérifiée** et l'agrégation de point de vue financier est **faisable**, sinon elle est **non vérifiée**, ainsi le processus d'agrégation s'arrête et l'agrégat est **annulé**.

5.2.2 Première règle d'agrégation : règle d'affaires (RA-01):

CpA et CpB sont compatibles pour l'agrégation du point de vue des affaires si et seulement si un des cas d'affaire de chacun des composants participants répond aux besoins d'affaires du nouveau composant agrégat CpC. Du point de vue des modèles UML d'affaires, le ou les cas d'utilisation d'affaires du composant CpC sont satisfaits, au minimum, par un des cas d'affaires des composants participants par une association (lien qui peut être de type « utilise », « inclus », « spécialise » ou un nouveau stéréotype). Dans ce cas, **RA-01** est vérifiée et l'agrégation du point de vue affaires des composants est qualifiée de **faisable.**

5.2.3 Deuxième règle d'agrégation : règles des attributs non fonctionnels (RA-02):

Si l'agrégation d'affaires des composants est faisable, alors l'évaluation de l'agrégation des exigences non fonctionnelles peut être initiée. Pour chaque attribut non fonctionnel de l'agrégat, nous vérifions les incompatibilités potentielles pour le même attribut des composants sources, **RA-02** est vérifiée et l'agrégation des attributs non fonctionnels des composants est qualifiée de **faisable** puisqu'elle satisfait les exigences non fonctionnelles de l'agrégat. Dans le cas contraire, une ou plusieurs itérations d'analyse doivent avoir lieu pour identifier les solutions potentielles afin de remédier aux incompatibilités détectées. Si les

résolutions n'aboutissent, la règle n'est pas vérifiée et l'agrégation des attributs non fonctionnels est qualifiée de **non faisable**.

5.2.4 Troisième règle d'agrégation : règles des plateformes (RA-03):

Si CpA et CpB doivent être utilisés dans une même plateforme et qu'il n'y a pas d'incompatibilités entre les plateformes sources des deux composants, **RA-03** est vérifiée et l'agrégation des plateformes est qualifié de **faisable**, sinon l'agrégation est qualifié de **non faisable**. De plus, si l'agrégation ne nécessite pas une même plateforme pour son développement, dans ce cas les composants peuvent être réutilisés à partir de leur plateforme respective et l'agrégation des plateformes est qualifié de **faisable.**

5.2.5 Quatrième règle d'agrégation : règles de l'interopérabilité des services (RA-04):

L'évaluation porte sur la compatibilité syntaxique et sémantique-pragmatique des services des composants participants. Si nous ne détectons pas d'incompatibilité, alors **RA-04** est vérifiée et l'interopérabilité des services des composants participants est qualifiée de **faisable.**

5.2.6 Cinquième règle d'agrégation : règle des attributs d'agrégation (RA-05):

Cette évaluation porte sur les attributs d'agrégations SOCOM des composants participants. Si nous ne détectons pas d'incompatibilité des attributs d'agrégation, alors la règle RA-05 est vérifiée et l'agrégation est qualifiée de **faisable.**

5.3 Cheminement des règles d'agrégation et les états de l'agrégat

En se basant sur les règles précédemment décrites, le composant agrégat peut être décrit par plusieurs états dépendamment de sa vérification des règles. Ainsi:

- Un agrégat qui ne valide pas la règle RA-00 est appelé : **agrégat annulé**.

- Un agrégat qui ne valide pas une ou un sous ensemble des règles RA-01, RA02, RA03, RA04, RA05 est appelé : **agrégat non faisable**.
- Un agrégat qui valide une des règles d'agrégation est appelé : **agrégat potentiel**.
- Un agrégat qui valide toutes les règles est appelé : **agrégat faisable**.

La figure suivante présente un des cheminements possibles pour l'application des règles d'évaluation de la faisabilité d'agrégation. Nous précisons que la règle RA-03 et les deux règles RA-04 et RA-05 peuvent être évaluées en parallèle avec deux équipes différentes. Si ces règles ne sont pas vérifiées, elles ne sont pas bloquantes pour l'agrégation vu qu'il existe souvent des techniques et des solutions pour y remédier. En effet, le cheminement proposé peut changer selon l'expertise de l'équipe de projet, la disponibilité des métadonnées SOCOM des composants, de la complexité de l'agrégation et du nombre de composants impliqués. Les règles RA-00 et RA-01 doivent être exécutées dans l'ordre proposé car si elles ne sont pas vérifiées, nous jugeons que l'évaluation des autres n'est pas utile et représente une perte de temps et donc d'argent pour la compagnie.

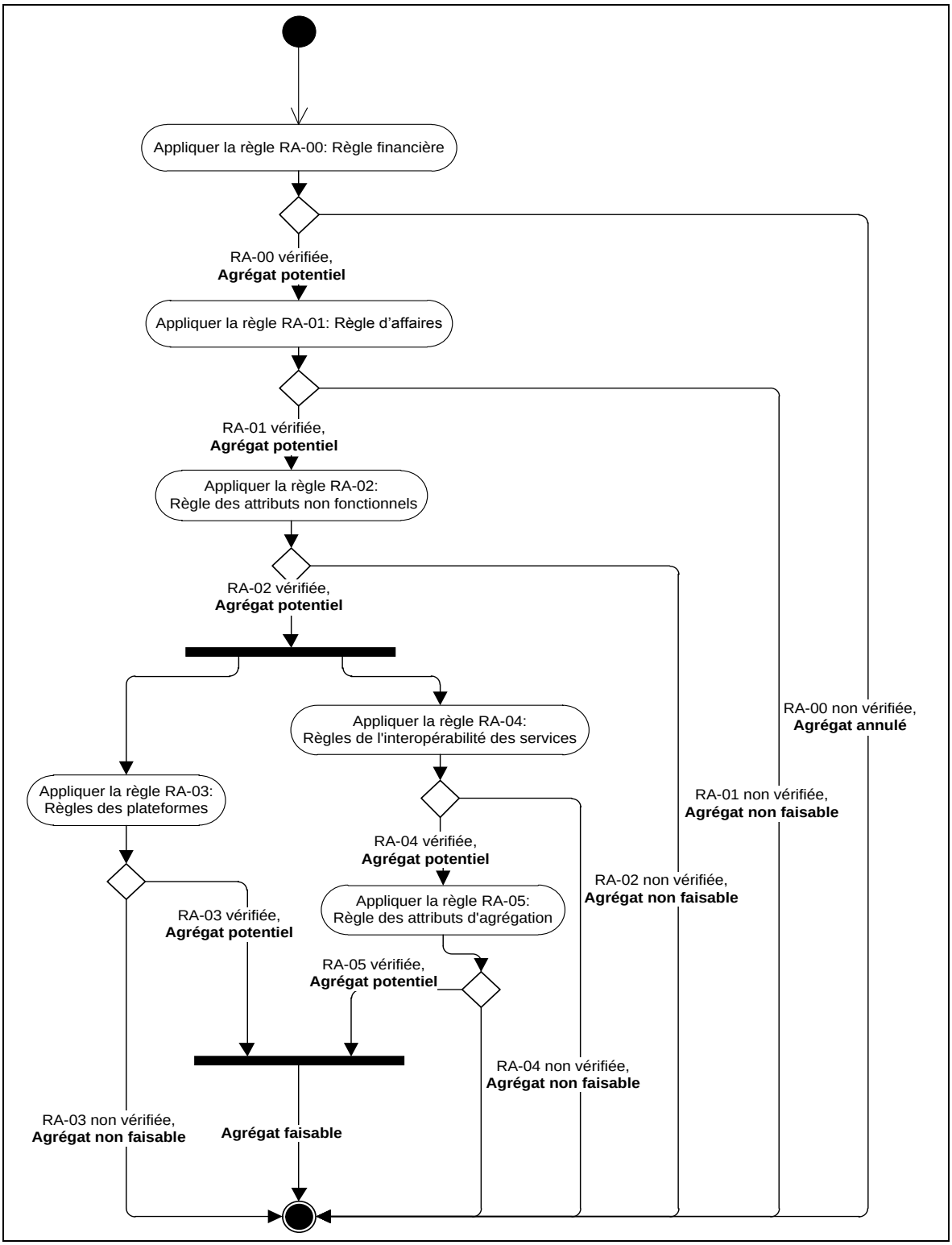


Figure 5.2 Proposition d'un ordre d'exécution de vérification des règles d'agrégation

5.4 La classification des incompatibilités des composants logiciels

Durant l'évaluation de l'agrégation, il est nécessaire de pouvoir identifier des incompatibilités et les classer à l'aide d'une taxonomie. Les travaux de Yamickovich *et al.* ont défini une liste de classes d'incompatibilités syntaxiques, sémantiques et pragmatiques entre les composants ainsi qu'entre leurs environnements. Cette liste a été réutilisée et bonifiée par les aspects architecturaux et des plateformes des spécifications SOCOM.

5.4.1 Les incompatibilités architecturales des composants logiciels

Ce type d'incompatibilité concerne essentiellement les caractéristiques architecturales des composants. Du point de vue des spécifications SOCOM, l'évaluation des attributs des métadonnées d'agrégation peut décider des agrégations potentielles possibles entre deux composants. Les définitions fournies dans les sections suivantes vont être enrichies ou modifiées selon le contexte d'agrégation et le domaine d'affaires associé. Dans ce qui suit, la liste les incompatibilités possibles est basée sur les métadonnées d'agrégations de SOCOM.

5.4.1.1 Couche logicielle

Ce type d'incompatibilité est détecté lorsque les couches logicielles auxquelles les composants appartiennent ne peuvent pas communiquer directement. L'intervention d'un composant d'un niveau intermédiaire est alors nécessaire. Ex : dans une architecture à trois couches, un composant de présentation ne peut pas communiquer avec un composant de données. Un composant de traitement est alors nécessaire pour respecter l'hypothèse d'architecture par couche stricte.

5.4.1.2 Potentiel de couplage

Ce type d'incompatibilité est détecté lorsqu'un des composants à agréger présente un couplage fort. Le résultat d'agrégation de ces derniers est un composant à couplage fort. Si le

composant agrégat présente un besoin non fonctionnel de type couplage faible, une incompatibilité de type potentiel de couplage est détectée.

5.4.1.3 Degré d'accessibilité

Ce type d'incompatibilité est détecté lorsque les composants se présentent dans un contexte d'agrégation avec des types de boîtes différentes. En effet, la résultante d'agrégation d'un composant de type boîte noire et d'un composant de type boîte blanche est un composant de type boîte grise. Par définition, un composant de type boîte grise expose une partie de ses fonctionnalités à l'extérieur à l'aide d'une API publique. Cette différence du point de vue des boîtes est qualifiée d'incompatibilité lorsqu'on impose, à l'avance, le type de boîte du composant agrégé. Toutefois, rien n'exclut que deux composants ayant des types de boîtes différents peuvent s'agréger pour créer un composant de type boîte grise.

5.4.1.4 Niveau de couplage

Ce type d'incompatibilité est détecté lorsque deux composants ont un niveau de couplage horizontal et n'appartiennent pas à la même couche logicielle. Ex : Un composant CpA et un composant CpB appartiennent respectivement aux couches logicielles présentation et traitement. Si ces composants ont un niveau de couplage horizontal, alors ils ne peuvent s'agréger qu'avec des composants de la même couche logicielle qu'eux.

5.4.1.5 Les incompatibilités syntaxiques des composants logiciels

Ce type d'incompatibilité concerne essentiellement les caractéristiques syntaxiques des composants. Ces caractéristiques rejoignent les métadonnées de services des spécifications SOCOM. En fait, l'évaluation des métadonnées des services peut décider des agrégations potentielles possibles entre deux ou plusieurs composants. Dans ce qui suit, les incompatibilités syntaxiques possibles sont associées aux métadonnées de service de SOCOM.

5.4.1.6 Compilation/Exécution

Ce type d'incompatibilité est détecté lorsque les composants présentent des erreurs dans les environnements de compilation ou d'exécution. Ces erreurs peuvent nuire au bon fonctionnement de l'agrégat utilisant ces composants.

5.4.1.7 Protocole des interfaces

Ce type d'incompatibilité est détecté lorsqu'un composant demande les services d'un autre composant avec une interface différente de celle requise. Autrement dit, le composant appelant ne peut pas satisfaire la description de l'interface du composant appelé du point de vue du nombre de paramètres ou du type de paramètres ou du type de retour. Aussi deux composants peuvent avoir des contrôles différents de type synchrone et asynchrone, et par conséquent, les échanges entre eux mènent à l'échec.

5.4.2 Les incompatibilités sémantiques et pragmatiques des composants logiciels

Ce type d'incompatibilité concerne essentiellement les caractéristiques sémantiques et pragmatiques des composants. En analysant les travaux de Yamickovich *et al.*, ces incompatibilités se basent sur des attributs similaires aux métadonnées métiers, statiques et de qualité des spécifications SOCOM. Ils peuvent décider des agrégations potentielles possibles entre deux ou plusieurs composants. Dans ce qui suit, ces incompatibilités présentées sont associées à quelques métadonnées statiques, métiers et de qualité de SOCOM.

5.4.2.1 Disfonctionnement interne

Ce type d'incompatibilité est détecté lorsqu'un composant ne fonctionne pas correctement dans un environnement de test, de pré-production ou de production. Autrement dit, les problèmes qui surviennent lors de l'utilisation de ce composant ne proviennent pas d'une interaction avec un autre composant, mais plutôt d'une interaction entre ses sous composants

internes ou d'un défaut de conception ou de codage du composant lui-même. La version du composant, qui est une métadonnée statique de SOCOM, peut indiquer si le composant possède une version stable ou non. Les attributs « fonctionnement » et « disponibilité » des métadonnées de qualité peuvent aussi nous indiquer si le composant fonctionne correctement ou pas.

5.4.2.2 Disponibilité des services

Ce type d'incompatibilité est détecté lorsqu'un composant ne peut pas rendre accessibles ses services fournis. Cette disponibilité des services peut être causée par des facteurs environnementaux ou de sécurité tels que les ports fermés des pare-feu qui bloquent l'accès au composant.

5.4.2.3 Non-respect des règles d'affaires

Ce type d'incompatibilité est détecté lorsqu'un composant ne peut pas se conformer à des règles préétablies par un autre composant afin de consommer ses services. Ces règles peuvent être des pré-conditions, des post-conditions, des invariants ou autres conditions qui portent sur le respect d'une séquence d'appel ou sur l'identification des variables de configuration du composant avant, durant ou après l'appel à ses services. Ces règles sont équivalentes aux règles de pré-conditions et post-conditions des contrats de services entre composants (voir section 1.8).

5.4.2.4 Ressources physiques

Ce type d'incompatibilité est détecté lorsque l'agrégation des composants est insatisfaite par les ressources physiques de l'environnement d'exécution. Les ressources physiques concernent essentiellement la capacité du processeur, de la mémoire vive, de l'espace de stockage sur le support de stockage permanant (disque dur), de la bande passante réseau, etc. Ce manque de ressources physiques est une incompatibilité entre le composant agrégat potentiel et l'environnement dans lequel il est déployé. Autrement dit, chaque composant

peut fonctionner avec les ressources physiques d'environnement données. Toutefois, ensemble, ils ne peuvent pas fonctionner dans ce même environnement.

5.4.3 Incompatibilités des plateformes

Ce type d'incompatibilité concerne essentiellement les caractéristiques des plateformes des composants. Du point de vue des spécifications SOCOM, l'évaluation des métadonnées des plateformes peut décider des agrégations potentielles possibles entre deux ou plusieurs composants. Dans ce qui suit, nous définissons quelques classes d'incompatibilités des plateformes en utilisant quelques métadonnées de SOCOM.

5.4.3.1 Source de données

Ce type d'incompatibilité est détecté lorsque deux composants veulent communiquer des données avec des types de stockage (persistance) différents (fichier texte, fichier XML, des enregistrements dans une base de données relationnelles, des fichiers RDF, des ontologies OWL).

5.4.3.2 Système d'exploitation

Ce type d'incompatibilité est détecté lorsqu'un composant ne peut pas fonctionner dans le même système d'exploitation qu'un autre composant. Cette incompatibilité est souvent détectée dans les catégories d'agrégation par fusion ou par collection. Pour ces deux catégories, les composants doivent être déployés dans le même système d'exploitation.

5.4.3.3 Serveur d'application

Ce type d'incompatibilité est détecté lorsqu'un composant n'est pas déployé dans le même type de serveur d'application que l'autre composant. Nous pouvons détecter des incompatibilités entre deux serveurs d'application avec des technologies hétérogènes (serveur Microsoft .NET vs serveur d'application JEE) ou avec des technologies homogènes,

par exemple où deux serveurs d'applications compatibles avec la même spécification (Web logic, WebSphere, JBoss et OC4J), tous compatibles avec une même spécification (dans le cas présent JEE), mais qui peuvent montrer des incompatibilités lors de la migration d'un composant de l'un à l'autre.

5.4.3.4 Zonage Réseau

Ce type d'incompatibilité est détecté lorsque deux composants éprouvent des difficultés d'interopérabilité et de communication rapportées aux zones réseaux dans lesquelles ils sont déployés. Par exemple, deux composants ne peuvent pas communiquer directement si l'un est dans une zone Internet publique et l'autre dans une zone Intranet sécurisée.

5.4.3.5 Sécurité

Ce type d'incompatibilité est détecté lorsque deux composants éprouvent des difficultés d'interopérabilité et de communication rapportées aux pare-feu derrière lesquels ils sont déployés ou des problèmes d'interopérabilité des répertoires utilisateurs (LDAP, Microsoft Active Directory, BD utilisateur propriétaire, etc.).

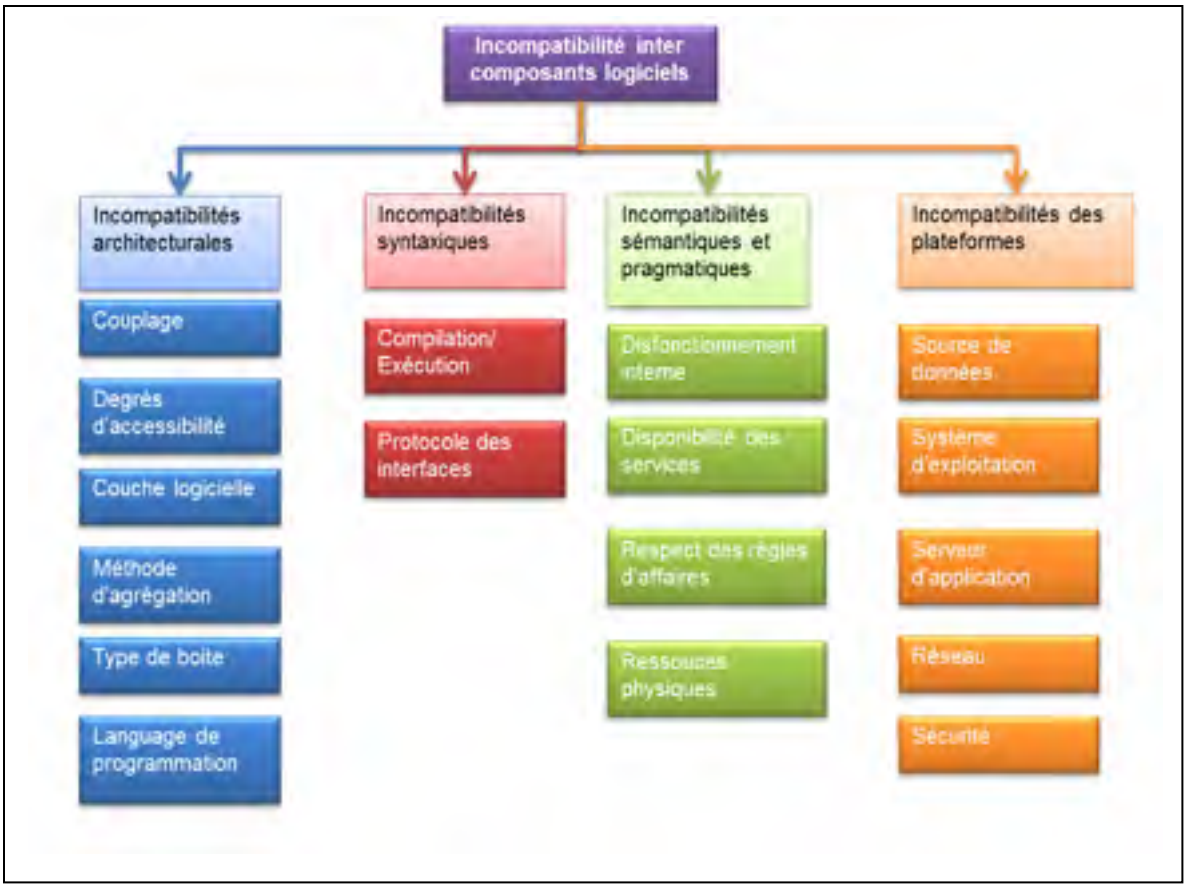


Figure 5.3 Taxonomie des incompatibilités entre les composants logiciels et leurs aspects environnementaux

La Figure 5.3 fournit une vue hiérarchique de la taxonomie proposée de ces incompatibilités.

5.5 Processus d'étude de la faisabilité d'agrégation des composants logiciels

5.5.1 Modèle BPMN du processus

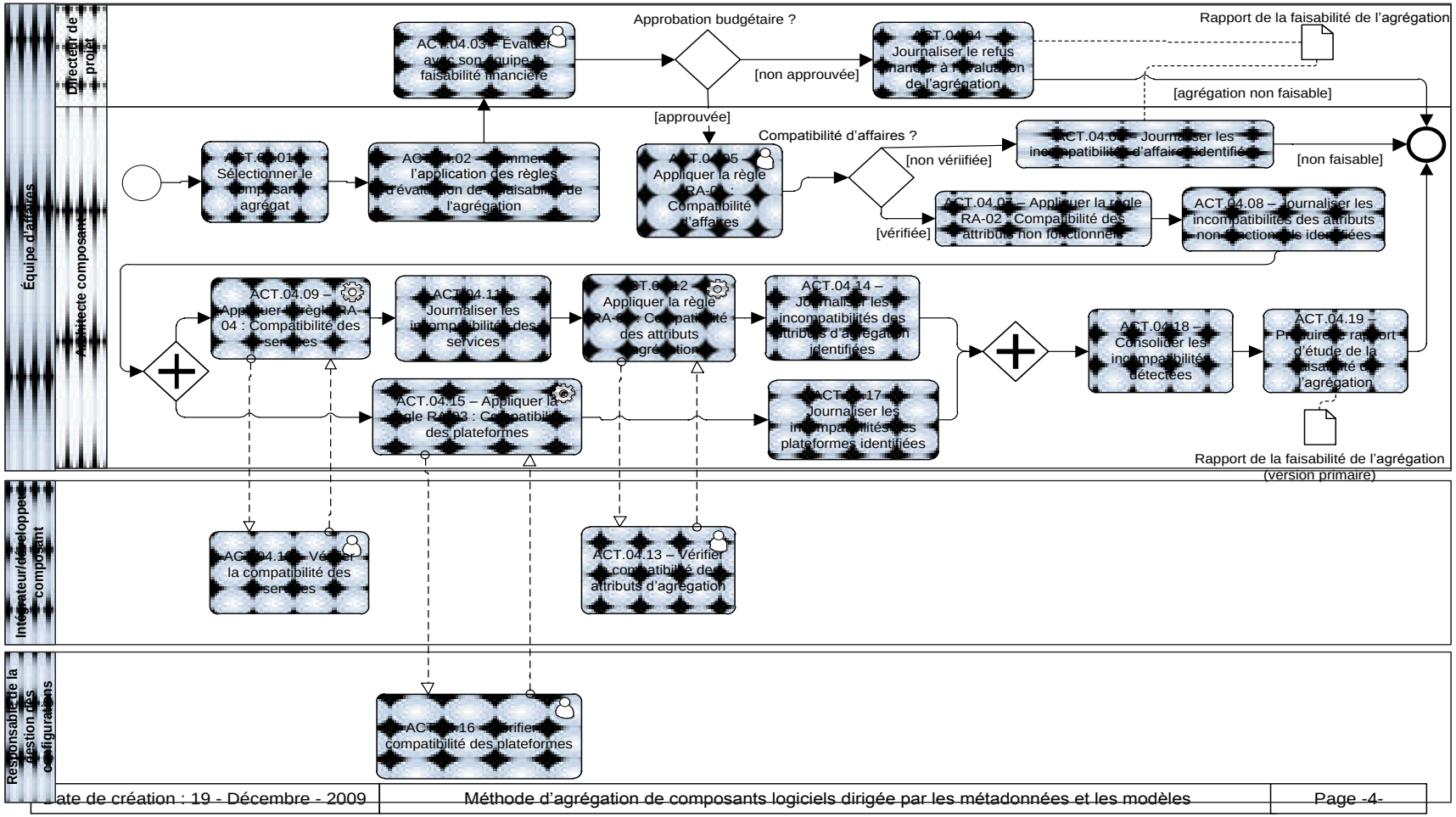


Figure 5.4 Sous processus SOCAP: ACT.04- Évaluer l'agrégation des composants

5.5.2 Acteurs du processus : rôles et responsabilités

Les acteurs de ce sous-processus sont décrits dans la section 4.3 du processus général SOCAP.

5.5.3 Contraintes générales

Afin d'exécuter ce sous-processus d'évaluation avec succès, il faut respecter les contraintes suivantes :

- Les données financières concernant l'évaluation de la faisabilité d'agrégation doivent être déjà définies au début du sous-processus.
- Une description des métadonnées SOCOM doit être disponible pour chaque composant source avec toutes les sections remplies.
- La description minimale du composant agrégat cible à l'aide des métadonnées SOCAM doit être fournie au début du sous-processus. Cette description minimale est composée du nom de l'agrégation, ses besoins d'affaires, une brève description et son contexte d'agrégation.
- Les gabarits de journalisation des résultats d'évaluation doivent être disponibles et maitrisés par l'architecte de composant.
- Les incompatibilités détectées automatiquement par la vérification des métadonnées SOCOM des composants sources doivent être révisées afin de s'assurer de leur validité et de leur pertinence.

5.5.4 Hypothèses et pré-conditions

Ce sous-processus hérite des hypothèses du processus SOCAP. De plus, les hypothèses suivantes doivent être vérifiées pour s'assurer de son utilisation efficace:

- L'information financière autour de l'agrégation doit être disponible à l'équipe d'affaires afin de pouvoir exécuter le sous-processus d'évaluation.

- Les composants sources ou leur documentation doivent être disponibles pour que l'architecte puisse réaliser l'évaluation et l'exécution des bancs d'essais.
- Les métadonnées SOCOM des composants logiciels sources sont conformes aux propriétés des composants réels.
- L'architecte doit utiliser la liste des catégories d'incompatibilités présentées dans la section 5.4 pour classer les incompatibilités détectées.
- Toutes les compatibilités et les incompatibilités sont journalisées dans un document intitulé « *Rapport de la faisabilité de l'agrégation*».
- Les acteurs qui manipulent le sous-processus doivent avoir des compétences suffisantes en architecture logicielle et d'entreprise pour réaliser leurs activités d'évaluation et de proposition de solution. Autrement dit, les compétences associées aux rôles doivent être disponibles pour pouvoir identifier les incompatibilités potentielles.
- Les résolutions potentielles proposées pour remédier aux incompatibilités détectées sont réalisables. Ces dernières sont proposées par l'architecte en collaboration avec le développeur et l'intégrateur afin de s'assurer de leur faisabilité.

5.5.5 Description détaillée des activités principales du sous-processus

Dans ce qui suit, les acteurs de la section 4.1 vont utiliser les règles d'agrégation décrites dans la section 5.2. Le modèle des activités du sous-processus d'évaluation est proposé dans la Figure 5.4.

5.5.5.1 Logique générale concernant l'évaluation des règles d'agrégation

En pratique, l'évaluation de ces règles se fait par une comparaison directe des valeurs des attributs des métadonnées associées. Si les valeurs des attributs sont différentes, ceci implique l'identification d'une incompatibilité potentielle. Toutefois, une vérification est toujours nécessaire pour valider ou écarter l'incompatibilité en question. Si elle est confirmée, l'incompatibilité devient réelle et une résolution d'impose. Dans les cas où la ou les résolutions sont concluantes, l'incompatibilité est éliminée. Dans le cas contraire, la règle

en question n'est pas vérifiée et alors le sous-processus s'arrête et l'exécution se poursuit au niveau du processus parent SOCAP. L'architecte d'affaires initie le sous processus en sélectionnant le composant agrégat à évaluer (activité ACT.04.01) et commence l'application des règles de faisabilité de l'agrégation (activité ACT.04.02). Dans la suite, nous décrivons l'exécution des activités les plus importantes. Celles de journalisation seront rappelées brièvement par leur identifiant.

5.5.5.2 ACT.04.03 – Évaluer avec son équipe la faisabilité financière

Connaissant l'information concernant le budget de l'évaluation, le directeur du projet applique la règle **RA-00** et évalue si un budget est alloué et approuvé pour l'évaluation de la faisabilité d'agrégation des composants logiciels. Deux décisions peuvent être prises : « Approuvée » ou « Non approuvée ». Si le budget est approuvé, l'agrégat est dans un état « en cours d'évaluation » et l'état est journalisé – en exécutant l'activité ACT.04.04 –sinon le sous-processus « Évaluer la faisabilité de l'agrégation » sera terminé suite au résultat négatif de l'évaluation. Dans ce cas, le refus du budget est journalisé et l'agrégat est qualifié de « annulé». L'annulation de l'agrégat entrainera l'arrêt du sous-processus en cours et de son processus parent et donc de l'agrégation.

5.5.5.3 ACT.04.05 - Appliquer la règle RA-01: compatibilité d'affaires

Connaissant les métadonnées d'affaires des composants sources et de l'agrégat, l'architecte de composant applique la règle **RA-01** pour évaluer la compatibilité des métadonnées d'affaires des composants à agréger. Il vérifie essentiellement que les composants sources disposent des requis d'affaires tout en satisfaisant les besoins de l'agrégat. Deux décisions peuvent être prises : « vérifiée » ou « non vérifiée ». Si la compatibilité d'affaires est «vérifiée », l'exécution du sous-processus se poursuit par l'application des règles subséquentes sinon l'information d'incompatibilité est journalisée – en exécutant l'activité ACT.04.06 – et le sous-processus sera terminé. Dans ce cas, l'agrégat est qualifié de « non faisable ». Ceci entrainera l'arrêt du sous-processus en cours et le processus parent doit

reprendre en exécutant de nouveau les activités de recherche de composants pour répondre aux besoins de l'agrégation en cours.

5.5.5.4 ACT.04.07 - Appliquer la règle RA-02: compatibilité des attributs non fonctionnels

RA-01 étant « vérifiée », l'architecte de composant applique la règle **RA-02** pour évaluer la compatibilité des attributs non fonctionnel des composants à agréger. Quel que soit le résultat de l'évaluation, les incompatibilités détectées ou non sont journalisées – en exécutant l'activité ACT.04.08 – et l'exécution du sous-processus se poursuit par l'application des autres règles. Dans ce cas, l'agrégat demeure dans l'état « en cours d'évaluation».

5.5.5.5 ACT.04.09 - Appliquer la règle RA-04: compatibilité des services

RA-02 étant « vérifiée », l'architecte de composant applique la règle **RA-04** pour évaluer la compatibilité des métadonnées des services des composants sources. Pour s'assurer du résultat de l'évaluation, l'architecte fait appel aux services d'un intégrateur/développeur qui vérifie conceptuellement et techniquement la compatibilité des services. Ceci correspondant à l'exécution de l'activité ACT.04.10. Cette évaluation peut être menée en parallèle avec l'évaluation de la règle de la compatibilité des plateformes (activité ACT.04.16, voir section 5.5.5.7). Quel que soit le résultat de l'évaluation, les incompatibilités des services détectées ou non sont journalisées – en exécutant l'activité ACT.04.11 – et l'exécution du sous-processus se poursuit par l'application des règles subséquentes. L'agrégat demeure avec l'état « en cours d'évaluation».

5.5.5.6 ACT.04.12 - Appliquer la règle RA-05: compatibilité des attributs d'agrégations

L'architecte de composants applique la règle **RA-05** pour évaluer la compatibilité des attributs d'agrégation des composants sources. Pour s'assurer du résultat de l'évaluation, l'architecte fait appel aux services d'un intégrateur/développeur qui vérifie techniquement la

compatibilité des attributs d'agrégation. Ceci correspond à l'exécution de l'activité ACT.04.15. Cette évaluation peut être menée en parallèle avec l'étude de la compatibilité des plateformes (activité ACT.04.16, voir section 5.5.5.7). Quel que soit le résultat de l'évaluation, les incompatibilités des attributs d'agrégation détectées ou non sont journalisées – en exécutant l'activité ACT.04.14 – et l'exécution du sous-processus se poursuit par l'application des autres règles. L'agrégat demeure dans l'état « en cours d'évaluation».

5.5.5.7 ACT.04.16 - Appliquer la règle RA-03: compatibilité des plateformes

L'architecte de composant applique la règle **RA-03** pour évaluer la compatibilité des plateformes des composants sources. Pour s'assurer du résultat de l'évaluation, l'architecte fait appel aux services d'un responsable de la gestion de la configuration qui vérifie techniquement la compatibilité des plateformes. Ceci correspondant à l'exécution de l'activité ACT.04.17. Cette évaluation est menée en parallèle avec l'étude de la compatibilité des services et des attributs d'agrégation. Quel que soit le résultat de l'évaluation, les incompatibilités des attributs des plateformes détectées ou non sont journalisées – en exécutant l'activité ACT.04.18 – et l'exécution du sous-processus se poursuit. L'agrégat demeure dans l'état « en cours d'évaluation».

5.5.5.8 ACT.04.19 - Consolider les incompatibilités détectées

L'architecte de composant revoit toutes les incompatibilités journalisées et identifie celles qui doivent être retenues et résolues de celles qui ne sont pas critiques à la faisabilité de l'agrégation et donc peuvent être écartées. Cette activité de filtrage porte aussi sur les compatibilités vérifiées automatiquement. En fait, la vérification automatique est basée sur la comparaison entre les valeurs des attributs SOCOM des composants sources. Si les valeurs sont identiques ou équivalentes la compatibilité est assurée sinon une incompatibilité est détectée. L'acteur peut invalider une compatibilité, laquelle se transforme en une incompatibilité qui nécessitera une résolution de la part de l'architecte. Pour chaque incompatibilité retenue, l'architecte propose une résolution avec l'aide des développeurs et

des intégrateurs. Ces résolutions sont potentielles et on rappelle qu'elles sont, par hypothèse, faisables pour pouvoir compléter le sous-processus d'évaluation.

Les analyses de consolidation et de résolution faites dépendent entièrement de l'expertise de l'architecte et de son jugement sur la pertinence et l'importance des incompatibilités détectées automatiquement. Un commentaire par incompatibilité écartée est souvent requis pour la traçabilité.

Toutes les activités associées à la consolidation et à la résolution doivent être traitées et commentées dans le document « rapport de la faisabilité de l'agrégation ». Ce document doit contenir, à la fin des exécutions, tous les commentaires et la liste finale des résultats d'incompatibilités et des résolutions pertinentes pour aider à la décision finale sur la faisabilité ou non de l'agrégat.

5.5.5.9 ACT.04.20 - Produire le rapport d'étude de la faisabilité de l'agrégation

À partir des résultats sur les incompatibilités et les résolutions proposées suite à l'exécution des règles d'agrégation, une décision finale est prise sur la faisabilité de l'agrégation. Le rapport présentera trois types de décision concernant l'agrégation, soient **Annulée, Non faisable** ou **Faisable**

L'architecte de composant analyse la liste des incompatibilités détectées dans le rapport d'évaluation de l'agrégation. La décision est basée sur la possibilité de résolution des incompatibilités et de l'approbation du directeur de projet pour aller de l'avant dans le processus parent d'agrégation. Suite à l'évaluation, l'agrégation sera dans un des états suivants:

- **Faisable** : quand les incompatibilités identifiées peuvent être résolues avec les composants sélectionnés. L'exécution de l'agrégation se poursuit au niveau du processus parent.

- **Non faisable** : quand les incompatibilités ne peuvent pas être résolues avec les composants sélectionnés. D'autres composants candidats équivalents, déjà identifiés et connus, peuvent alors remédier aux incompatibilités identifiées de l'agrégation courante. Suite à la sélection de nouveaux composants sources, le sous-processus d'évaluation de la faisabilité est relancé dans une nouvelle itération.
- **Annulée** : quand les incompatibilités ne peuvent pas être résolues avec les composants sélectionnés et qu'il n'existe pas d'autre composant candidat équivalent pour réaliser l'agrégat en question. L'annulation peut aussi être due à une décision purement financière du directeur du projet. Une telle décision termine le sous processus d'évaluation ainsi que le processus parent, SOCAP.

À la fin du sous-processus et dépendamment de la décision prise, l'agrégat peut être:

- potentiel (décision faisable);
- en attente (décision non faisable);
- annulé (décision annulée).

5.6 Conclusion

Ce chapitre a introduit les règles d'agrégation, les classes d'incompatibilités des composants et les activités détaillées du sous-processus d'évaluation de la faisabilité de l'agrégation des composants logiciels du processus SOCAP. Cette vue détaillée du sous-processus a permis de décrire la logique du sous-processus d'évaluation ainsi que ses acteurs et leurs responsabilités associées. Dans le chapitre 4, les activités du processus parent sont détaillées et le chapitre 7 présente la validation de SOCAP par des exécutions des activités de SOCAP dans des contextes réels d'agrégation.

Le tableau suivant décrit l'apport du sous-processus de faisabilité, comme résultat de recherche, aux phases d'un processus de développement à base de composants.

Tableau 5.1 Matrice des apports du sous-processus de faisabilité aux phases du processus de développement à base de composants

Résultats de la thèse	**Description des contributions des résultats par phase**
Processus de faisabilité	**Phase de recherche :** - La phase de recherche peut se déclencher suite à la non-faisabilité de l'agrégation des composants sélectionnés. **Phase de sélection :** - Ce processus propose des règles de détection des incompatibilités qui valident/invalident la sélection du bon composant candidat selon la réussite du test de faisabilité. - Ce processus réduit les sélections erronées de composants incompatibles avant de commencer la réalisation de l'agrégat. **Phase de faisabilité :** - Ce processus détecte les incompatibilités tôt dans le processus pour éviter des efforts d'adaptation et d'intégration inutiles. - Ce processus contrôle le cheminement du processus global en réduisant les risques d'avancer dans le processus avec des composants incompatibles. - Ce processus propose des activités de consolidation/vérification qui corrigent les incompatibilités apparentes ou qui découvrent des incompatibilités cachées. - Ce processus propose un rapport de faisabilité, journalisant les évaluations de faisabilité, qui peut être réutilisé lorsque le composant agrégat ou agrégé est candidat potentiel dans de nouveaux contextes d'agrégation. **Phase d'adaptation:** - Ce processus sauve des coûts d'adaptation si le test de faisabilité échoue. **Phase d'intégration:** - Ce processus réduit les coûts d'intégration si le test de faisabilité échoue.

CHAPITRE 6

CADRE CONCEPTUEL POUR L'AGRÉGATION DES COMPOSANTS LOGICIELS (SOCAF – SOftware Component Aggregation Framework)

Après la présentation des métadonnées SOCOM, des modèles de classification des agrégats, du processus d'agrégation SOCAP et de son sous-processus d'évaluation de la faisabilité de l'agrégation, nous définissons un cadre de travail qui décrit les outils et les logiciels nécessaires pour exécuter les processus et mettre en pratique l'ensemble de nos solutions de recherche. Ce cadre de travail prépare l'étape de vérification et de validation par l'exécution des processus proposés dans des contextes réels qui est présentée dans le chapitre suivant.

Il est important de rappeler que la composition de ce cadre est définie de façon itérative et incrémentale. Chaque résultat de recherche est supporté par un outil sélectionné ou une application développée. Ce chapitre décrit l'ensemble des applications développées, les outils utilisés, les pratiques définies et les relations entre eux pour réussir la mise en place de notre processus d'agrégations des composants dirigé par les métadonnées et les modèles. Ce cadre est élaboré dans la septième étape de la démarche de notre méthodologie de recherche. La figure suivante positionne le présent chapitre dans le cheminement global de la thèse.

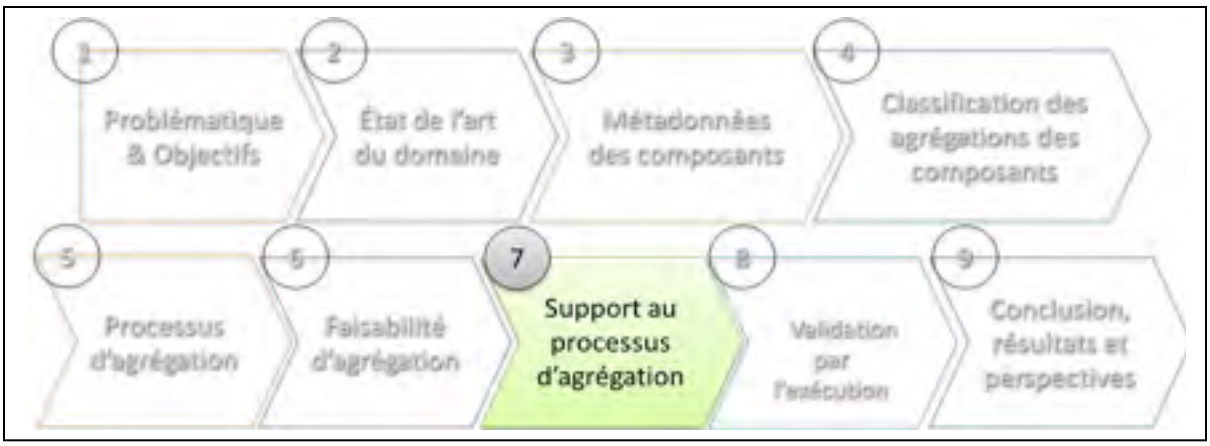


Figure 6.1 La méthodologie de recherche de la thèse – Étape 7 :
Support au processus d'agrégation

6.1 Le besoin d'un cadre de travail

L'objectif principal de la thèse est la proposition d'une méthode d'agrégation de composants logiciels dirigée par les modèles et les métadonnées. Pour appliquer la méthode, il faut définir des entités conceptuelles (aussi nommées des modules conceptuels) qui serviront de base pour la construction des composants par l'agrégation d'autres composants. Ces entités conceptuelles se traduisent, dans une phase d'implémentation, en paquetages informatiques/entités logicielles dans une architecture logicielle spécifique. Nous proposons donc un cadre de travail (*framework*) qui englobe ces entités et que nous avons baptisé "cadre de travail de l'agrégation des composants logiciels", soit **SOCAF** (**SO**ftware **C**omponents **A**ggregation **F**ramework). Il s'agit d'une description simple d'une structure des entités logicielles – nécessaires à l'agrégation – à l'aide d'un modèle abstrait (Voir Figure 6.2). Ces entités assisteront l'agrégation des composants et la construction du nouveau composant agrégat (composant *C*) par agrégation de composants existants (composants *A* jusqu'à *N*) en se basant sur les modèles et les métadonnées de ces composants.

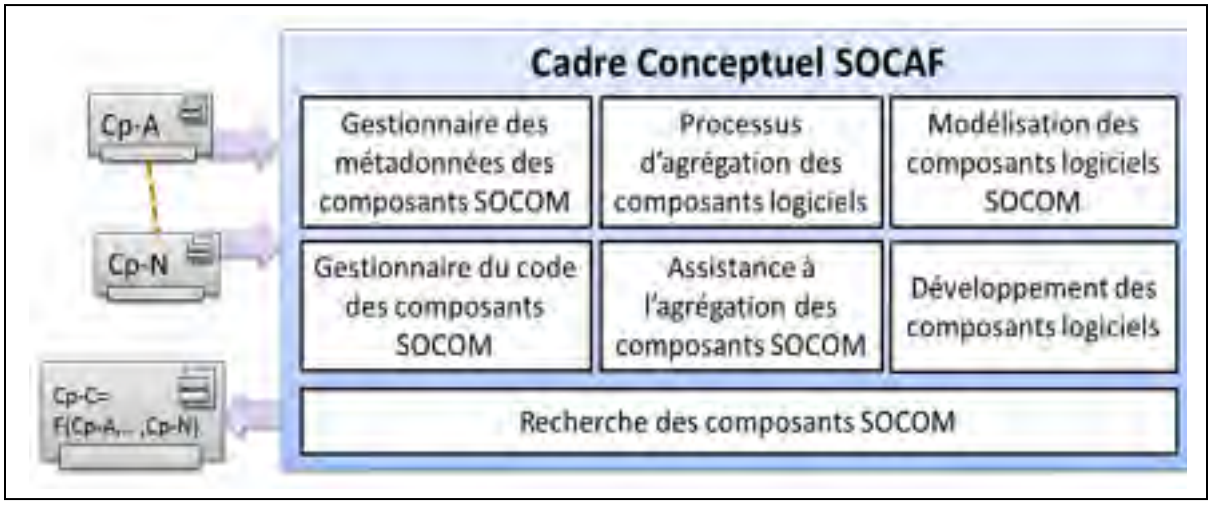


Figure 6.2 Cadre de travail pour l'agrégation de composants logiciels – SOCAF

6.2 Définition et structure générale de SOCAF

Les entités de ce cadre conceptuel forment les fondements de l'architecture logicielle qui permettra de mettre en pratique la démarche décrite par le processus d'agrégation SOCAP. Autrement dit, chaque entité de SOCAF servira de support à une ou plusieurs étapes du processus SOCAP. Ces entités peuvent communiquer entre elles pour échanger des données ou des informations durant l'exécution du processus SOCAP. Elles se traduiront dans une phase d'implantation en entités logicielles sous forme d'outils ou d'applications Web. Ces dernières peuvent être déployées et hébergées dans des environnements similaires ou différents. SOCAF traite essentiellement l'agrégation des composants de type SOCOM. Pour permettre le traitement d'autres composants, il est primordial de les référencer à l'aide des spécifications SOCOM.

La définition, dans SOCAF, de deux niveaux d'entités conceptuelles et logicielles est voulue. En effet, nous jugeons que la mise en pratique d'une méthode d'agrégation de composants logiciels (ou développement logiciel à base de composants) nécessitera les entités conceptuelles définies ci haut (Voir Figure 6.2). Les entités logicielles choisies dans le cadre de cette thèse sont une instanciation possible du cadre SOCAF. Toutefois, d'autres instanciations sont possibles avec de nouveaux outils, dans d'autres contextes d'agrégation et dans des domaines d'affaires particuliers. Il s'agit d'une ouverture du cadre de travail qui projette des extensions potentielles visant à enrichir SOCAF.

6.3 Composition de SOCAF : entités conceptuelles

Les sections suivantes décrivent les entités conceptuelles qui composent le cadre de travail **SOCAF.** Pour chaque entité, on présente une brève description, les services fournis et des exemples des outils de support potentiels pour promouvoir son utilisation durant l'exécution du processus SOCAP.

6.3.1 Gestionnaire des métadonnées des composants SOCOM

Cette entité devra permettre le référencement des composants logiciels avec la proposition des métadonnées SOCOM. Elle permet de produire une fiche de métadonnées SOCOM pour chaque composant candidat afin que ce dernier participe au processus d'agrégation. Elle permettra aussi la recherche et la classification des composants logiciels.
Cette entité offre essentiellement les services suivants :

- référencement des composants SOCOM;
- gestion des fiches des composants SOCOM;
- recherche des composants SOCOM;
- extraction des métadonnées de services du code binaire des composants;
- exportation des données des fiches SOCOM à partir de la base de données en document XML;
- conversion du document XML des composants SOCOM en ontologie OWL.

L'entité « Gestionnaire des composants SOCOM » est implantée sous la forme d'une application Web développée en Java/JEE avec une base de données MySQL. L'exportation en document XML utilise une bibliothèque Java pour la sérialisation XML. Un programme Scheme est développé pour la transformation des données des fiches SOCOM de l'XML en OWL. L'extracteur des métadonnées de services du code binaire des composants est un ensemble de composants développés en JAVA, C++ et .NET. Ces extracteurs sont exécutés sur des composants implantés dans la même technologie pour extraire automatiquement leurs métadonnées de services et, ainsi, aider dans l'automatisation du référencement des composants SOCOM.

6.3.2 Gestionnaire du code des composants

Cette entité permet de sauvegarder les codes des composants SOCOM lorsque le contexte d'agrégation le permet. Il est important de rappeler que parfois, les codes des composants participants dans les agrégations peuvent ne pas être disponibles, ce qui rend impossible la

sauvegarde des composants binaires dans le gestionnaire. Si, le code source n'est pas disponible, seul le code exécutable pourra être sauvegardé. Ce gestionnaire de code est utile durant les phases du développement de l'agrégat et, surtout, au cas où une adaptation du composant source serait requise.

Cette entité offre essentiellement les services suivants :

- sauvegarde des codes sources ou binaires des composants SOCOM;
- gestion des versions des codes des composants SOCOM;
- recherche simple des codes des composants SOCOM.

Pour le stockage et la gestion des versions des codes sources, nous utilisons l'outil « Apache Subversion » communément appelé SVN. Un logiciel libre a servi au contrôle et à la gestion du code source des projets logiciels, TortoiseSVN[25]. D'autres outils peuvent être utilisés pour contrôler et gérer les codes sources des composants comme CVS. Il important de préciser que, si les composants sources sont sauvegardés dans d'autres systèmes, des convertisseurs sont disponibles pour les exporter dans l'outil SVN.

6.3.3 Recherche des composants

Cette entité permet la recherche des composants SOCOM suite à leur référencement dans le « Gestionnaire des métadonnées des composants ». La recherche se base essentiellement sur les métadonnées des composants sauvegardées dans le modèle relationnel des spécifications SOCOM. Aussi, cette entité permet la navigation/exploration de l'ontologie SOCOM. Une recherche plus intelligente, qui utilise des langages de requête ontologique, est souhaitable.
Cette entité offre essentiellement les services suivants :

- classification des composants SOCOM;
- recherche simple des composants SOCOM;
- recherche avancée des composants SOCOM;

[25] TortoiseSVN: projet SVN disponible à l'adresse web suivante : http://tortoisesvn.net/

- recherche dans l'ontologie des composants SOCOM.

La recherche est implantée dans l'application Web du «Gestionnaire des métadonnées des composants ». En effet, les critères de la recherche utilisent quelques attributs de SOCOM pour les recherches simples et avancées. Pour parcourir l'ontologie de SOCOM, l'application Web génère l'ontologie des composants SOCOM et l'outil Protégé est utilisé pour explorer l'ontologie et voir une classification des composants SOCOM. Le langage de requête OWL fourni par l'outil Protégé peut être utilisé pour formuler des requêtes plus intelligentes. Aussi, le moteur d'inférence Pellet est utilisé pour exécuter des requêtes intelligentes et valider l'ontologie en question.

6.3.4 Modélisation des composants logiciels SOCOM

Cette entité de modélisation des composants est composée de plusieurs environnements existants qui offrent la modélisation de composants logiciels et l'agrégation de leur modèle dans un formalisme donné. Les langages de modélisation choisis dans notre cas sont UML pour les composants, l'annotation BPMN pour les processus, MOT-PLUS OWL pour les ontologies.

Cette entité offre essentiellement les services suivants :

- modéliser les composants SOCOM;
- modéliser les composants agrégats;
- modéliser les interactions des composants de l'agrégat;
- générer partiellement les signatures du code du composant agrégat;
- générer partiellement les signatures du code des composants sources.

Plusieurs outils de modélisation UML2 peuvent être utilisés pour répondre à ce besoin. Dans notre cas, nous avons utilisé Rational Rose, IBM-Rational Architect, IBM-Rational Modeler et Eclipse UML.

6.3.5 Assistance à l'agrégation des composants SOCOM

Cette entité permet d'évaluer la faisabilité de l'agrégation en se basant sur les fiches SOCOM des composants sources, les règles d'agrégation et des activités d'évaluation. Elle supporte entièrement le processus d'évaluation de la faisabilité d'agrégation des composants présenté dans la section 5.5.

Cette entité offre essentiellement le service d'étude de la faisabilité de l'agrégation des composants logiciels SOCOM. Cette entité d'assistance, développée sous forme de processus, est implantée à l'aide d'un gabarit Excel avec différents onglets dans lesquels nous avons spécifié les étapes, l'application des règles et la présentation des résultats d'exécution des activités. Cet outil est central pour le support du processus parce qu'il contient le flux des données correspondants au flux de contrôle présenté par le diagramme BPMN du processus SOCAP.

6.3.6 Développement des composants logiciels

Cette entité sert d'outil pour le développement, l'adaptation ou la transformation des composants logiciels. Durant la phase de codage, les composants peuvent subir des changements dictés par le contexte d'agrégation. Ainsi, cette entité propose un ou plusieurs environnements spécialisés afin de réaliser les activités de codage conformément au scénario proposé par le processus SOCAP. Cette entité offre essentiellement les services d'outils pour le développement, l'adaptation et l'intégration des composants logiciels SOCOM par la réutilisation.

Tous les environnements de développement logiciel peuvent servir pour implanter cette entité. Voici quelques exemples : Eclipse, IBM WebSphere, Microsoft Visual Studio, Oracle JDeveloper, etc.

6.3.7 Processus d'agrégation des composants logiciels

Cette entité définit les activités nécessaires pour réaliser une agrégation des composants logiciels dirigés par les métadonnées et les modèles. D'autres aspects tels que les principes d'agrégation, les hypothèses, les contraintes, les limites et les parties prenantes sont décrits afin de définir un processus d'agrégation cohérent et valide. Cette entité assiste un ensemble d'acteurs dans leur mission de développement par la réutilisation. Elle représente le pivot du cadre de travail SOCAF. Toutes les autres entités offrent des éléments conceptuels, logiciels et environnementaux pour contribuer au bon déroulement de ce processus.

Cette entité offre essentiellement le service de développement de nouveaux composants par agrégation de composants existants en utilisant les modèles d'agrégats et les métadonnées SOCOM et SOCAM.

Cette entité est implantée sous forme du processus SOCAP et modélisé à l'aide du formalisme BPMN. Ce processus est composé de sous processus de recherche, de sélection, d'étude de faisabilité, de conception pour le développement de composants logiciels par agrégation. Un fichier Excel sert d'outil pour l'exécution du processus SOCAP dans différents contextes d'agrégation.

6.4 Les interactions entre les entités

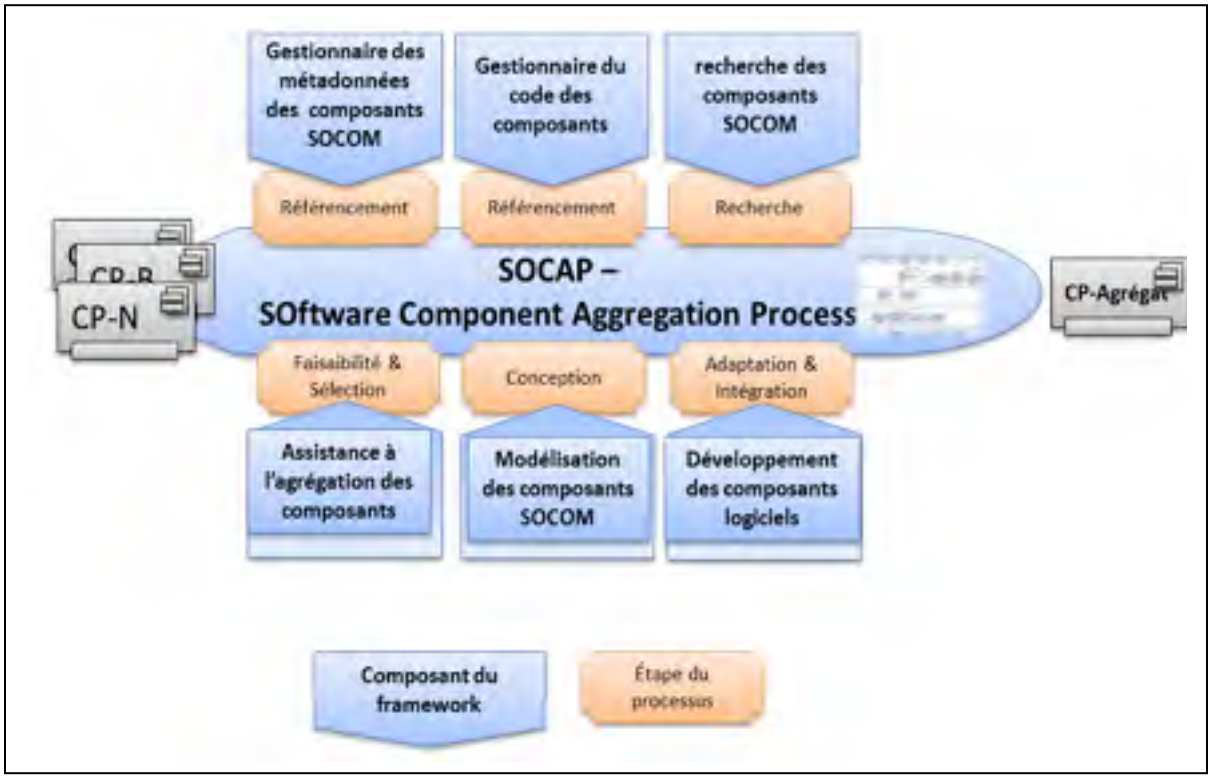


Figure 6.3. Les points d'interaction des entités du framework SOCAF

SOCAP est l'entité noyau du cadre de travail SOCAF. Les autres entités gravitent autour de SOCAP pour permettre au processus de répondre aux besoins de développement de composants par agrégation. Ces différentes entités interagissent essentiellement avec SOCAP à travers les grandes phases du processus: le référencement, la recherche, la sélection, la conception, l'adaptation et l'intégration.

Le « Gestionnaire des métadonnées des composants SOCOM » et le « Gestionnaire du code des composants » fournissent au processus SOCAP des services pour répondre aux besoins de **référencement** des composants SOCOM. La « recherche des composants SOCOM » offre à SOCAP les différents types de **recherche** avec les différentes formes du référentiel des composants SOCOM (Relationnel, XML, OWL). L'entité d'« assistance à l'agrégation des

composants» offre un processus d'étude de faisabilité capable de servir les phases de **faisabilité** et de **sélection**. Les outils de « Modélisation des composants logiciels » et de « Développement des composants logiciels » complètent le processus d'agrégation en offrant des environnements capables d'assister les acteurs du processus dans les phases de **conception**, **d'adaptation** et **d'intégration**.

6.5 Exemple d'instanciation du cadre de travail SOCAF dans un contexte réel

La Figure 6.4 montre une instanciation du cadre de travail SOCAF dans le cadre de l'agrégation par connexion «Télépaiement de Loisir-En-Ligne – VDM[26] ». L'instanciation est réalisée par le remplacement des entités conceptuelles par des entités logicielles. Par exemple, l'entité logicielle « Gestionnaire SOCOM – Application Web JEE » implante l'entité conceptuelle « Gestionnaire des métadonnées des composants SOCOM ».

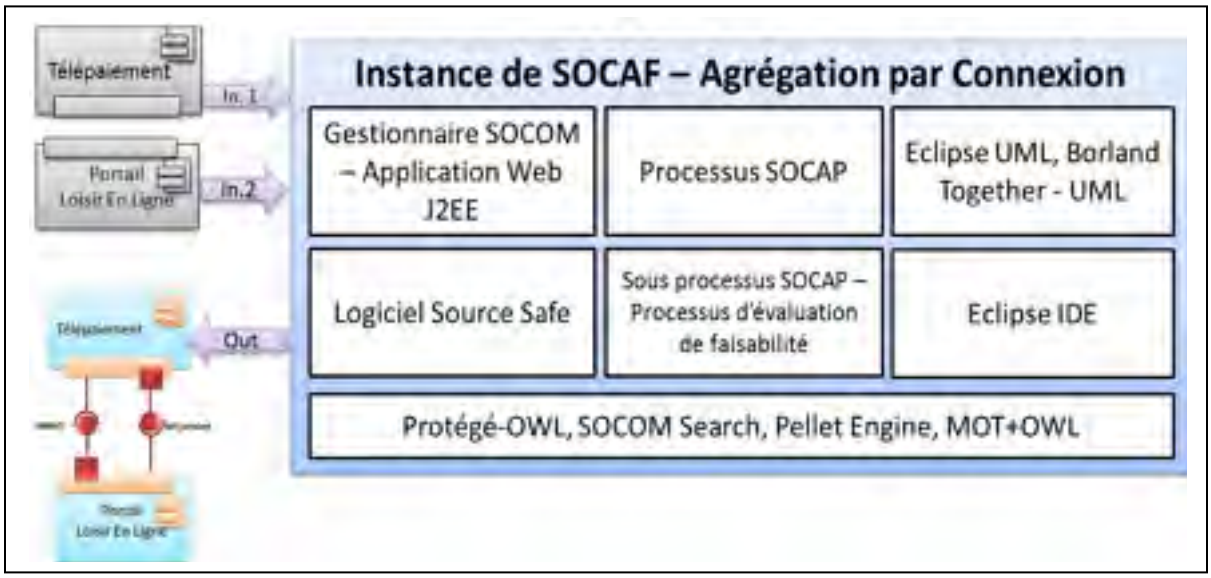


Figure 6.4 Instance de SOCAF pour une agrégation par connexion

[26] VDM : Ville De Montréal

6.6 Conclusion

Avec la définition du cadre de travail SOCAF, nous proposons aux utilisateurs du processus SOCAP un cadre applicatif pour l'exécution des activités de développement de composants par agrégation en se basant sur les métadonnées et les modèles. La définition des entités avec l'exemple d'instanciation présenté dans les sections précédentes décrit le côté conceptuel/théorique et démontre le côté pragmatique des travaux de cette thèse.

Les entités peuvent être regroupées pour s'intégrer dans un même environnement capable de répondre à tous les besoins du processus SOCAP. Autrement dit, un environnement intégré similaire à Microsoft TFS[27] ou IBM Rational Jazz peut être configuré pour permettre à différents profils de mener les activités de développement de composants par la réutilisation. Ce type d'environnement favorise davantage la traçabilité et facilite le suivi des activités du processus SOCAP sans être obligé de changer d'environnement pour chaque phase de SOCAP.

Le tableau suivant décrit l'apport du cadre de travail SOCAF, comme résultat de recherche, aux phases d'un processus de développement à base de composants.

[27] Microsoft TFS: Microsoft Team Foundation Server

Tableau 6.1 Matrice des apports du cadre conceptuel SOCAF aux phases du processus de développement à base de composants

Résultats de la thèse	**Description des contributions des résultats par phase**
SOCAF : En général, SOCAF propose une structure et une boîte d'outils qui supportent la méthode proposée et permet son utilisation dans des contextes concrets.	**Phase de recherche :** - SOCAF propose l'application de référencement « Gestionnaire SOCOM » qui implante les métadonnées, ce qui permet de faciliter l'exploration et la recherche des composants et leur agrégat. **Phase de sélection :** - SOCAF propose l'application « Gestionnaire SOCOM » qui permet l'exploration de toutes les métadonnées des composants retrouvés, ce qui facilite leur sélection. **Phase de faisabilité :** - SOCAF fournit un outil – développé avec Excel – qui implémente les règles de faisabilité et permet de décider de leur agrégation. **Phase d'adaptation:** - SOCAF rappelle des techniques, des approches et des pratiques de l'industrie pour assister l'intégrateur lors de l'adaptation des composants à agréger. **Phase d'intégration:** - SOCAF rappelle des techniques, des approches et des pratiques de l'industrie pour assister l'intégrateur lors du développement de l'agrégat (code).

CHAPITRE 7

VALIDATION PAR L'EXÉCUTION DU PROCESSUS SOCAP

Ce chapitre présente les aspects de la validation du processus d'agrégation des composants logiciels SOCAP. Dans les chapitres 4 et 5, nous avons décrit le processus SOCAP et le sous processus d'évaluation de la faisabilité d'agrégation, respectivement. Le chapitre 6 a introduit un cadre de travail qui fournit un support pour l'implantation des processus proposés. Dans ce qui suit, nous décrivons la validation du processus SOCAP en l'exécutant dans des contextes réels.

Nous voulons nous assurer que chacune des exécutions aboutit à un composant agrégat fonctionnel si l'agrégation est faisable ou à une explication valide et utile dans le cas contraire. Ceci nous confirmerait que SOCAP et nos résultats connexes sont utiles, rentables et permettent effectivement d'assister une équipe de projet lors du développement logiciel à base de composants. De plus, nous voulons nous assurer que les résultats des exécutions valident les objectifs du processus d'agrégation (section 4.1).

Le reste du chapitre décrit quatre exécutions réalisées dans l'industrie. Pour chacune d'elles, nous présentons leur particularité et les étapes exécutées les plus importantes. Nous avons réalisé treize exécutions du processus SOCAP durant nos efforts de validation, mais nous ne présentons que quatre cas type. Les cas choisis sont représentatifs des catégories d'agrégation proposées dans le chapitre 3, soit la connexion, la collection, la coordination et la fusion.

Il est important de rappeler que la validation est réalisée de façon itérative et incrémentale sous forme d'exécutions partielles associées aux différents résultats de recherches présentés dans les chapitres précédents. En effet, nous avons validé chaque résultat de recherche (SOCOM, SOCAM, SOCAF, etc.) par l'utilisation d'un outil spécifique ou par le développement d'une application adéquate (p. ex : gestionnaire SOCOM, outil Excel d'évaluation de la faisabilité, etc.). Les détails des exécutions sont fournis en annexe. Nous

avons élaboré un processus de validation avec des indicateurs synthétiques pour valider notre processus SOCAP. Après les exécutions, nous avons défini et évalué ces indicateurs de validation et nous avons commenté les valeurs obtenues.

Ces exécutions sont élaborées dans la huitième étape de la démarche de notre méthodologie de recherche. La figure suivante positionne le présent chapitre dans le cheminement global de la thèse.

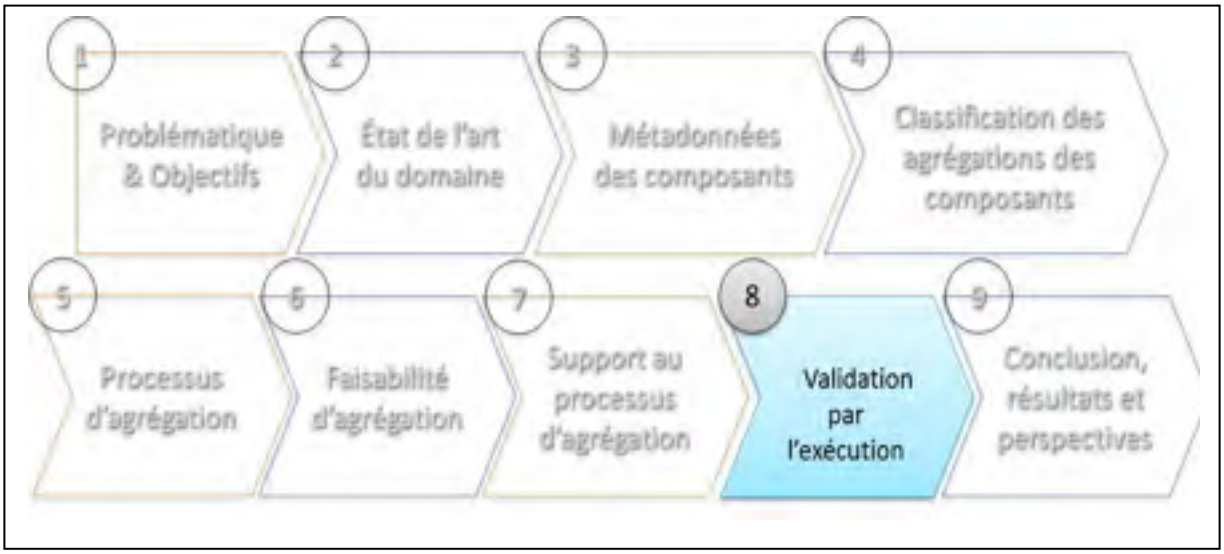


Figure 7.1 La méthodologie de recherche de la thèse – Étape 8 : Validation par l'exécution

7.1 Processus de validation du processus

Le processus de validation du processus SOCAP repose sur trois indicateurs. Le premier concerne l'achèvement du processus intitulé **I**ndicateur d'**A**chèvement du processus **S**OCAP (**IAS**). En effet, nous avons réalisé treize exécutions et nous voulons nous assurer que notre processus s'achève dans toutes les exécutions, soit avec un agrégat fonctionnel, soit avec une explication qui décrit la cause de non faisabilité de l'agrégation. Le tableau suivant montre que neuf des treize exécutions ont abouti à un agrégat et quatre se sont conclues avec des agrégats non réalisables. L'indicateur d'achèvement d'un processus exécuté valide que ce dernier est bien construit. Un processus bien construit est un processus qui se termine (atteint

l'activité finale) et présente un flux de contrôle et un flux de données bien définis et complets. La complétude se traduit par l'absence d'activités orphelines et d'activités dépourvues des donnée en entrée et en sortie. Il doit prévoir les exceptions et décrire les cheminements alternatifs.

Les deux autres indicateurs évaluent les indices de couverture des activités et des objectifs que nous avons nommés ICAS (**I**ndicateur de **C**ouverture des **A**ctivités suite à l'exécution de **S**OCAP) et ICOS (**I**ndicateur de **C**ouverture des **O**bjectifs suite à l'exécution de **S**OCAP), respectivement. Ces indicateurs seront évalués et analysés dans la section 7.8 après la présentation des exécutions du processus SOCAP dans quatre cas réels d'agrégation.

Tableau 7.1 Nombre d'exécutions complétées avec
des agrégats réalisables et non réalisables

Exécution complétée avec agrégat réalisable	**Exécution complétée avec agrégat non réalisable**	**Total des agrégations exécutées**
9	4	13

La Figure 7.2 présente le processus de validation du processus SOCAP.

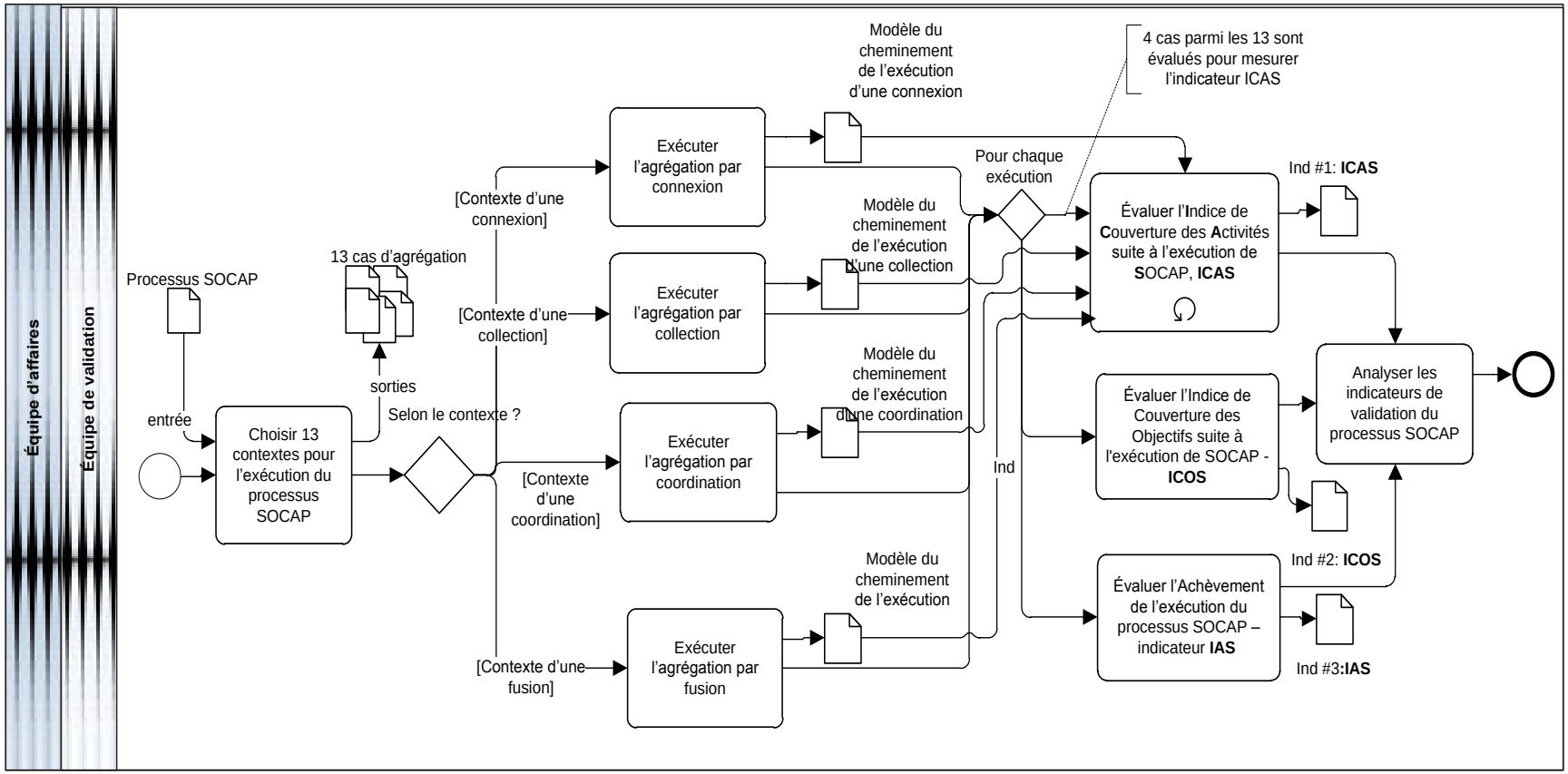


Figure 7.2 Modèle du processus de validation du processus SOCAP

Nous avons choisir treize cas d'agrégation qui couvrent nos quatre catégories d'agrégation proposées (collection, connexion, coordination et fusion). Après chaque exécution, nous vérifions si le processus se termine correctement en produisant un composant agrégat quand les agrégations sont faisables. Dans le cas contraire, le processus s'achève avec une explication qui justifie la non faisabilité de l'agrégation en question. L'analyse de ces exécutions nous permet d'évaluer nos trois indicateurs : l'indicateur d'achèvement du processus SOCAP (IAS), l'indicateur de couverture des activités suite à l'exécution de SOCAP (ICAS) et l'indicateur de couverture des objectifs suite à l'exécution de SOCAP (ICOS).

Dans les sections suivantes, nous décrivons l'exécution de quatre cas d'agrégation parmi les treize cas étudiés. Ensuite, nous définissons la formule de chaque indicateur, puis nous présentons les résultats de son évaluation dans un tableau de mesure. À la fin des mesures, nous analysons les valeurs obtenues des trois indicateurs, puis nous concluons quant à la validité du processus SOCAP.

7.2 Aspects communs à toutes les exécutions de SOCAP

Généralement, au début de toutes les agrégations, nous définissons les métadonnées SOCAM de l'agrégat (voir section 4.8): le nom de l'agrégat, son auteur, sa description générale, les hypothèses, les cas d'affaires et quelques exigences non fonctionnelles. Les compositions statique et dynamique sont définies au début selon la catégorie d'agrégation. Aussi la catégorie de l'agrégat et le patron associés sont identifiés durant l'exécution du processus SOCAP.

Chaque exécution est décrite à l'aide de la fiche commune SOCAM à trois colonnes (voir exemple du Tableau 7.2). La première colonne rappelle les attributs des métadonnées de l'agrégat SOCAM. La deuxième fournit les valeurs de ces attributs. Les notes de la troisième colonne correspondent aux identifiants des activités exécutées qui ont produit le contenu de l'attribut correspondant de SOCAM. Par exemple, dans le cas d'agrégation par

collection, la composition statique de l'agrégat est produite après l'exécution de l'activité ACT.03.11 (voir Tableau 7.2)

Les détails des exécutions sont documentés dans des fichiers Excel portant le nom de l'agrégat. Quelques saisies d'écrans Excel sont montrées en ANNEXE I pour le cas d'une agrégation par coordination. Nous avons aussi fourni une fiche SOCOM d'un composant participant dans l'ANNEXE II. Les exécutions présentées sont réalisées avec deux clients de CGI: Telus et la Ville de Montréal.

7.3 Agrégation par coordination: « Portail-outils » de TELUS

7.3.1 Fiche SOCAM de l'agrégat

Au début de l'exécution, nous avons rempli la fiche de l'agrégat « Portail-outils » avec les informations de base suivantes : le nom du composant agrégat, le nom de l'auteur, les hypothèses, la description générale, les hypothèses, les cas d'affaires et quelques exigences non fonctionnelles. Ces informations introduisent le contexte et la problématique que l'agrégation propose de résoudre.

Tableau 7.2 Fiche SOCAM de l'agrégat par coordination : Portail-outils de TELUS

Nom	"Portail-outils" de TELUS	*(Contenu complété après l'exécution de l'activité ACT.01)*
Auteur	TELUS : Client de CGI	*(Contenu complété après l'exécution de l'activité ACT.01)*
Description générale	Le portail-outils est un point d'entrée du client de Telus, centre CSPQ du Québec, pour utiliser un ensemble d'outils sous-jacents concernant la billetterie, la gestion des pare-feu, la surveillance de l'état du réseau et la gestion documentaire.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Hypothèses	- L'agrégat sera déployé dans les environnements du client - L'agrégat doit être accessible par les utilisateurs internes du Client de TELUS.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Composition statique	Le portail-outils est un composant agrégat composé des composants suivants : **1. SMIS** : Système de Gestion de l'Information (SGI) **2. SecurNet** : Outil de configuration des pare-feu **3. Tivoli WebTop** : Outil de surveillance de l'état du réseau informatique du client **4. Documentum**: Outil de gestion documentaire.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Catégorie de l'agrégation	Coordination	*(Contenu complété après l'exécution de l'activité ACT.05.02)*
Patron d'agrégation	Coordination fédérée	*(Contenu complété après l'exécution de l'activité ACT.06.02)*

Composition dynamique	Les interactions entre les composants participants se résument dans les cinq services suivants : **SER-1** « authentification-sso-portail-outils » : Le service permet aux utilisateurs d'accéder au portail-outils via une interface graphique appropriée. Ce service doit maintenir en session l'identité de l'utilisateur pour promouvoir l'authentification unique (*single sign-on*, ou SSO) aux composants sous-jacents. **SER-2** « authentification-sso-smis » : Le service récupère le nom de l'utilisateur de la session du portail-outils et assure en arrière-plan une nouvelle authentification au composant SMIS. **SER-3** « authentification-sso-documentum » : Le service récupère le nom de l'utilisateur de la session du portail-outils et assure en arrière-plan une nouvelle authentification au composant Documentum. **SER-4** « authentification-pre-sso-securnet » : Le service récupère le nom de l'utilisateur de la session du portail-outils et retourne la page d'authentification à trois paramètres (nom d'utilisateur/Mot de passe/jeton) au réseau virtuel sécurisé du composant SecurNet (VPN sécurisé). **SER-5** « authentification-pre-sso-tivoliwebtop » : Le service récupère le nom de l'utilisateur de la session du portail-outils et retourne la page d'authentification à trois paramètres (nom d'utilisateur/Mot de passe/jeton) au réseau virtuel sécurisé du composant Tivoli-Webtop (VPN sécurisé).	*(Contenu complété après l'exécution de l'activité ACT.06.12)*
Exigences non fonctionnelles	**<u>Sécurité :</u>** - ENF-01 : Authentification unique pour accéder au portail-outils et aux outils sous-jacents. - ENF-02 : Authentification faible requise	*(Contenu complété après l'exécution de l'activité ACT.01 et une mise à jour de l'ACT.10)*

	(nom d'utilisateur/mot de passe) - ENF-03 : L'accès au portail-outils se fera de façon sécurisée. **Disponibilité :** - ENF-04 : Le portail-outils doit être disponible huit heures par jour de 8h à 17h et cinq jours par semaine (UTC-05 :00, Eastern Time) (ENF-dispo-01) **Volumétrie-Charge** - ENF-05: Le portail-outils doit permettre un accès de 300 personnes - ENF-06: 30 accès concurrents sont prévisibles **Ergographie du portail-outils** - ENF-07 : Le portail-outils doit être conforme au PIV du client et aux normes W3C d'accessibilité ENF-07bis : Le portail-outils et les composants participants sont supportés par Internet Explorer Version 6. **Temps de réponse** - ENF-08 : Le portail-outils doit fournir un temps de réponse de 4 s pour l'accès aux pages. - ENF-09 : Le portail-outils doit fournir un temps de réponse de moins de 10 s pour l'accès aux outils sous-jacents.	*(Contenu complété après l'exécution de l'activité ACT.10)*
Autres	N/D	

7.3.2 Particularités de l'exécution

Dans ce qui suit, nous présentons sommairement les particularités de l'agrégation par coordination. L'ANNEXE I présente plus de détails les tableaux qui décrivent essentiellement l'exécution des activités ACT.04 et ACT.05. L'ANNEXE II décrit la fiche SOCOM détaillée de l'agrégat résultant de la présente exécution, « Portail-outils ».

Après l'exécution de l'activité ACT.01, nous constatons que l'équipe informatique de Telus souhaite offrir, à son client CSPQ[28], un portail pour centraliser l'accès aux outils des opérations de l'entreprise. Le besoin du client est d'avoir une interface web unique pour accéder aux outils existants : la billetterie, la gestion des pare-feu, la surveillance de l'état du réseau et la gestion documentaire. Ces outils sont des composants COTS déjà développés par Telus. Ils sont déployés dans les serveurs de Telus et vendus sous forme de services au CSPQ. En pratique, le besoin d'agrégation réel consiste à développer une interface d'authentification unique par-dessus les services de sécurité de ces outils pour permettre aux employés du CSPQ l'accès aux différents services des opérations selon leur profil. L'accès est fourni à partir d'un nouveau composant que le client appelle « Portail-outils ».

En analysant les informations de base de SOCAM, nous remarquons que les besoins exprimés par le CSPQ sont fournis par les composants existants suivants, déjà développés par Telus:

- SMIS : Système de Gestion de l'Information (SGI-Telus);
- SecurNet : Outil de configuration des pare-feu;
- Tivoli WebTop : Outil de surveillance de l'état du réseau informatique du client;
- Documentum : outil de gestion documentaire.

Pour étudier l'agrégation en question, nous commençons par ajouter les composants à l'entrepôt SOCOM de Telus, ce qui correspond à l'exécution de l'activité ACT.02 de SOCAP.

Durant l'exécution de l'activité ACT.03, nous n'avons pas fait la recherche, mais plutôt la sélection directe des composants déjà prédéfinis par le contexte de l'agrégation. Avec cette composition statique, nous avons évalué la faisabilité de l'agrégation (ACT.04). Les composants participants sont déployés dans des plateformes hétérogènes : SMIS est déployé dans une plateforme BEA WebLogic, Documentum est déployé dans une plateforme EMC et

[28] CSPQ: Centre des Services Partagés du Québec

SecurNet dans une plateforme Microsoft .Net. De ce fait, SOCAP détecte automatiquement plusieurs incompatibilités des plateformes suite à la première vérification.

Ces dernières incompatibilités sont potentielles et sont retirées suite à l'activité de leur résolution (ACT.05). Leur retrait est dû à ce que, dans le contexte de cette agrégation, les composants agrégés offrent chacun leurs services à partir de leur plateforme d'origine. En effet, le besoin réel est de donner accès à tous les outils avec une authentification unique. Ainsi, nous n'avons pas besoin d'unifier ou de faire interopérer les plateformes des bases de données, des serveurs d'applications ou des conteneurs des composants JEE et EMC. Toutefois, nous remarquons que les composants SMIS et Documentum sont dans une zone réseau TEN[29], alors que TivoliWebtop et SecurNet sont hébergés dans une zone OSSZ[30]. Il s'agit d'une incompatibilité détectée de type « Réseau » qui nécessite une résolution. Nous avons identifié d'autres incompatibilités de type non fonctionnelles telles que:

1. SSO[31] ne peut pas s'implémenter à tous les composants outils.
2. L'authentification faible n'est pas applicable à tous les composants outils.
3. L'authentification forte n'est pas applicable à tous les composants.
4. Le portail est le point d'entrée, il doit être disponible 24 heures par jour, sept jours par semaine.
5. Tous les composants ne peuvent pas être conformes au PIV (Protocole des Interfaces Visuelles).
6. Documentum n'offre pas un temps de réponse <= à 10 s (~35s à 45s).

Les catégories de résolution de ces incompatibilités varient: « SSO Partiel », « Extension de Support », « Nouveau développement », « Développement d'optimisation », « VPN Sécurisé ». Nous avons appliqué les résolutions proposées aux incompatibilités identifiées à l'aide de techniques de développement adéquates (p. ex : extension du code existant, optimisation du code et mise en place d'un nouveau réseau virtuel VPN). Aussi, les

[29] TEN: Telus Enterprise Network
[30] OSSZ: Operation Security Services Zone
[31] SSO : Single Sign On (authentification unique qui donne accès à plusieurs applications

exigences non fonctionnelles de l'agrégat (sécurité, disponibilité et ergographie[32]) requièrent du code additionnel et des tests appropriés pour s'assurer de leur satisfaction aux besoins et de leur conformité aux normes de développement.

L'exception identifiée dans le cas d'agrégation par coordination est associée à l'attribut d'agrégation « méthode d'agrégation ». En effet, les composants agrégés avaient la même méthode d'agrégation « Web Service ». Nous pouvons proposer une solution de type service web sécurisé pour transformer les services web simples. Cependant, les composants sont hébergés dans des zones réseaux différentes avec des niveaux de sécurité distincts et les services d'authentification ne peuvent pas être appelés d'une même page Web hébergée à l'extérieur des réseaux des composants sources. La vérification automatique n'a pas détecté cette incompatibilité. C'est au moment de décider de la rétention ou non de l'incompatibilité qu'on a détecté qu'il s'agit d'une fausse compatibilité. En effet, les liens des outils sécurisés (SecurNet et Tivoli Webtop) ajoutés au portail-outils ne sont pas fonctionnels parce que la zone « TELUS Public » ne permet pas l'accès aux outils dans la zone sécurisée. Pour remédier à cette incompatibilité, nous avons proposé une solution de type « EAI-Bridge VPN» qui consiste à rediriger les utilisateurs des outils sécurisés vers le portail Citrix. Ce dernier est accessible depuis la zone « Telus Public » et permet ainsi l'accès aux outils voulus. Les paramètres d'accès aux outils incluant les jetons (authentification forte) sont réutilisés pour accéder au portail Citrix. Une fois connecté à Citrix, les utilisateurs sont redirigés vers les interfaces d'authentification des outils sécurisés.

À ce stade de l'exécution, l'agrégation des composants participants est faisable. Nous passons à l'activité de conception pour classifier cette agrégation et déterminer sa composition dynamique. Si nous appliquons un raisonnement par élimination, cette agrégation n'est pas une connexion puisque nous avons quatre composants participants. Aussi nous écartons la fusion vu qu'il y a des composants utilisant des technologies hétérogènes (SMIS, TivoliWebtop et SecurNet). Ainsi nous n'avons donc pas de plateforme

[32] Ergographie : représente l'ergonomie des interfaces graphiques de l'utilisateur.

unique de déploiement, ni de possibilité d'avoir un seul composant agrégat physique. En analysant le besoin, nous constatons que le résultat final voulu est une coordination des services d'authentification des composants participants. Ces services relèvent de la couche de traitement, par conséquent nous écartons la collection qui est définie par les composants de présentation. Nous ajoutons un « T » pour les noms des composants évalués. Nous avons appliqué les équations d'une agrégation par coordination (définies dans la section 3.3.4) sur les composants de traitement.

Tableau 7.3 Métadonnées d'agrégation SOCOM des composants participants à une agrégation par coordination

Métadonnées d'agrégation	**Composant-Documentum-T**	**Composant-SMIS-T**	**Composant-TivoliWebtop-T**	**Composant-SecurNet-T**
Potentiel de couplage	Mixte	Mixte	Horizontal	Horizontal
Degré d'accessibilité	Boîte grise	Boîte grise	Boîte noire	Boîte noire
Couche logicielle	Traitement	Traitement	Traitement	Traitement
Nœud du système	rd.telus.com	smis.telus.com	tivoli.telus.com	securnet.telus.com
Point d'entrée du composant	https://ritm.gouv.qc.ca/ritmRD	https://smis.telus.com/	https://tivoli.gouv.qc.ca/login	https://securnet.telus.com/login
Méthode d'agrégation	Web Service	Web Service	Web Service	Web Service
Type de langage	Compilé	compilé	Compilé	compilé
Compatibilité avec les standards	j2ee, jdk 1.4	j2ee, jdk 1.4	j2ee, jdk 1.4	Microsoft .Net
Référence du composant agrégat	N/A	N/A	N/A	N/A
Plateforme de déploiement (SOCOM)	EMC Documentum	BEA WebLogic	IBM WebSphere	.NET Server

À partir des valeurs des attributs des composants, nous appliquons la vérification des règles de l'agrégation par coordination:

- **(CL**(Composant-Documentum-T) = T, **CL**(Composant-SMIS-T) = T, **CL**(Composant-TivoliWebtop-T) = T, **CL**(Composant-SecurNet-T) = T). **RegCouche** est vérifiée.
- (**DA**(Composant-Documentum-T) = {BG}, **DA**(Composant-SMIS-T) = {BG}, **DA**(Composant-TivoliWebtop-T) = {BN}, **DA**(Composant-SecurNet-T) = {BN}, ***InclutUnComposantBoîteBlanche***(Composant- Portail-outils-T, Composant-Documentum-T, Composant- SMIS-T, Composant- TivoliWebtop-T, Composant-SecurNet-T) = FAUX. **RegAccess** n'est pas vérifiée. Le composant « Portail-outils » est un nouveau composant proposé de type boîte blanche pour encapsuler la logique de la coordination.
- (**PC**(Composant- Documentum-T) = CH**, PC** (Composant-SMIS-T) = CM, **PC** (Composant-TivoliWebtop-T) = **CH**, **PC** (Composant-SecurNet-T) = **CH**) et ***CouplageFaisable***(Composant-Documentum-T, Composant- Portail-outils-T, Composant- SMIS-T, Composant-TivoliWebtop-T, Composant- SecurNet-T) = VRAI. **RegCoup** est vérifiée

À la fin de cette évaluation, nous concluons qu'il s'agit d'une agrégation par coordination. Le « Portail-outils » est un nouveau composant proposé pour encapsuler la logique de la fédération, ainsi le patron coordination fédérée est appliqué. Ce composant fédérateur est un point d'entrée des utilisateurs pour coordonner les accès aux outils d'opération. Autrement dit, si les utilisateurs veulent accéder à SMIS et Documentum, leur authentification au « Portail-outils » est suffisante. Pour les accès à Tivoliwebtop et SecurNet, le « Portail-outils » se charge d'accéder en arrière-plan au portail Cirtix et de rediriger les utilisateurs aux pages d'authentification des outils nécessitant une authentification forte, SecurNet et TivoliWebTop. À la fin de l'activité de conception, nous avons établi la solution et les services d'authentification que nous avons énumérés dans la partie « Composition dynamique » du Tableau 7.2.

Comparée aux catégories d'agrégation par collection et par connexion, l'étape de conception de cet agrégat est plus complexe et requiert un effort substantiel. Ceci s'explique par les efforts d'identification des interactions entre les services fournis des composants participants et le développement de la logique de coordination. La coordination a eu lieu entre les services d'authentification. Elle est implantée dans le nouveau composant logique « Portail-outils » qui joue le rôle de composant fédérateur.

Dans le contexte d'une agrégation par coordination, toutes les activités du modèle de premier niveau sont exécutées à part les activités du sous processus associé à l'activité « ACT.06-Concevoir l'agrégat ». Les seules sous activités exécutées de ce sous processus sont ACT.06.01, ACT.06.06, ACT.06.08 et ACT.06.12.

Après la conception, nous continuons avec les étapes de développement, de configuration, des tests qui sont souvent complexes à réaliser parce que les activités de paramétrisation sont typiquement appliquées sur les composants de type boîte grise et boîte noire, le tout pour satisfaire le besoin d'une coordination. En effet, un client est développé, pour chaque composant, pour implanter l'appel à ses services fournis. Des transformations des données en entrée et en sortie sont aussi requises pour pouvoir échanger l'information entre les services. De plus, le service agrégé doit fournir les données en sortie dans un format approprié et conformément aux protocoles de communications établis par l'architecte d'entreprise de Telus, ce qui explique un besoin de réaliser des tests de configurations locales et globales et de définir des tests de configuration locaux (par composant) et globaux (du composant agrégat).

Bien que les techniques d'adaptation, d'intégration des composants et de transformation des données soient matures et disponibles dans l'industrie, la difficulté de l'agrégation par coordination se manifeste dans la production des services agrégés tout en respectant les exigences non fonctionnelles de l'agrégat. Ce problème est aussi présent dans les agrégations par collection et par connexion, mais plus difficile à résoudre dans un contexte de coordination. Les travaux de (Villalobos, 2003) mettent en perspective les différents

problèmes associés aux exigences non fonctionnelles (appelées monde du problème par l'auteur) et proposent des solutions pour réussir des compositions de services (détail disponible dans la section 1.2). La figure suivante présente la modèle UML des composants de notre agrégat par coordination.

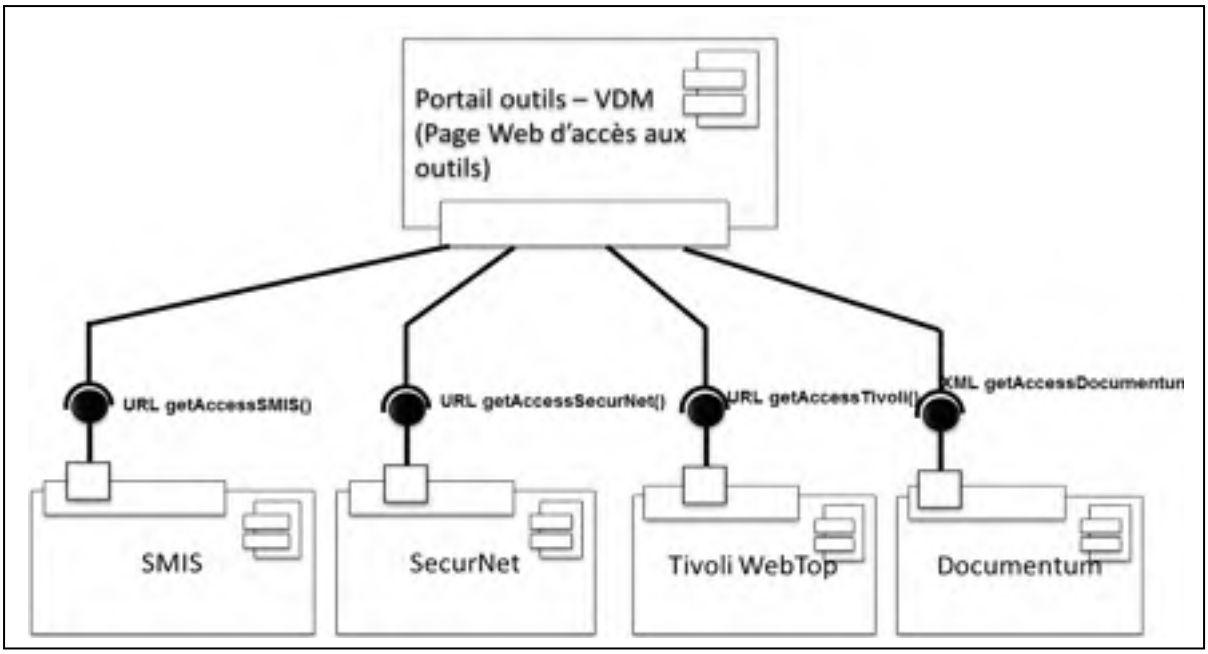


Figure 7.3 Modèle UML2 de l'agrégat par Coordination : Portail-outils – VDM

7.4 Agrégation par collection: « Portail Loisir en ligne - VDM»

7.4.1 Fiche SOCAM de l'agrégat

Au début de l'exécution, nous avons rempli la fiche de l'agrégat « Portail Loisir en ligne – VDM » avec les informations de base suivantes : le nom du composant agrégat, le nom de l'auteur, la description générale, les hypothèses, les cas d'affaires et quelques exigences non fonctionnelles. Ces informations introduisent le contexte et la problématique que l'agrégation propose de résoudre.

Tableau 7.4 Fiche SOCAM de l'agrégat par collection - Portail Loisir En Ligne – VDM

Nom	Portail Loisir en Ligne – VDM	*(Contenu complété après l'exécution de l'activité ACT.01)*
Auteur	Ville De Montréal (VDM)	*(Contenu complété après l'exécution de l'activité ACT.01)*
Description générale	Le projet consiste à offrir une page du portail VDM permettant l'accès à un ensemble de services aux citoyens de la Ville De Montréal. Ces services offrent principalement la description des activités et des loisirs. Ils permettent l'accès aux activités des centres de loisirs de la ville, aux événements culturels et des informations concernant les appels d'offres, les avis publics, les communiqués et les offres d'emplois.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Hypothèses	1. L'agrégat sera déployé dans les environnements Internet de la section culturelle du portail de la VDM. 2. La page d'accueil de l'agrégat devrait apparaitre dans la page "Activités et loisirs" du portail de la VDM.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Composition statique	Loisir-En-Ligne est un composant agrégat composé des composants suivants : 1. Portail-VDM: Composant Portail de la VDM; 2. Calendrier: Composant responsable d'afficher les événements sous forme de liste, de mini-liste ou de calendrier; 3. Annonces-VDM: Composant responsable de fournir la liste des annonces (communiqués, avis, appels, offres emplois); 4. Loisirs en ligne: Composant responsable de fournir la liste des activités sportives et culturelles de loisirs.	*(Contenu complété après l'exécution de l'activité ACT.03.11)*
Cas d'affaires	L'agrégat devrait offrir les cas d'affaires suivants : 1. Afficher un calendrier des événements culturels et sportifs; 2. Consulter et réserver les activités des loisirs; 3. Afficher la liste des annonces de la VDM; 4. Rechercher les événements avec deux modes (simple et complexe); .	*(Contenu complété après l'exécution de l'activité ACT.01)*

Catégorie de l'agrégation	**Collection**	*(Contenu complété après l'exécution de l'activité ACT.06.01)*
Patron d'agrégation	**Collectionneur**	*(Contenu complété après l'exécution de l'activité ACT.06.09 et ACT.06.10)*
Composition dynamique	"Portail des loisirs en ligne – VDM" est un composant agrégat. Il doit implémenter les services suivants : SER-1 : Afficher un calendrier des événements culturels et sportifs; SER-2 : Afficher les informations sur les annonces de la VDM; SER-3 : Fournir un accès au répertoire des activités de loisirs; SER-4 : Permettre la recherche des événements; SER-5: Intégrer le bandeau du calendrier dans le menu vertical du Portail-VDM;	*(Contenu complété après l'exécution de l'activité ACT.06.12)*
Exigences non fonctionnelles	**Sécurité :** ENF-01 : L'authentification au " Portail des loisirs en ligne – VDM " doit hériter des éléments de sécurité (authentification unique) du Portail de la VDM (utiliser le même OID que le portail - Oracle Identity Directory)	*(Contenu complété après l'exécution de l'activité ACT.01 et une mise à jour de l'ACT.10)*
	ENF-02 : Authentification faible requise (nom d'utilisateur/mot de passe)	
	ENF-03 : L'accès à l'application " Portail des loisirs en ligne – VDM "se fera de façon sécurisée.	
	Disponibilité :	
	ENF-04 : le composant " Portail des loisirs en ligne – VDM " doit être disponible 24/7 à 95% par année	
	ENF-04bis : Le composant " Portail des loisirs en ligne – VDM" doit être disponible à partir des interfaces graphiques du Portail-VDM.	
	Volumétrie-Charge	
	ENF-05 : Le composant agrégat doit permettre un accès de 100 000 personnes.	
	ENF-06 : 300 accès concurrents sont prévisibles.	
	ENF-07 : L'agrégat doit supporter un grand volume de données (plus de de 500,000 enregistrements par table maîtresse et par application)	

	Ergographie	
	ENF-08 : L'agrégat doit être confirme au PIV du Portail VDM et aux normes W3C d'accessibilité.	*(Contenu complété après l'exécution de l'activité ACT.01)*
	ENF-08bis : L'agrégat et les composants participants sont supportés par Internet Explorer Version 6.	*(Contenu complété après l'exécution de l'activité ACT.10)*
	Performance – Temps de réponse	
	ENF-09 : L'agrégat doit répondre aux requêtes simples des utilisateurs en moins de 4 s pour l'accès aux pages des événements.	
	ENF-10 : L'agrégat doit répondre aux requêtes complexes des utilisateurs en moins de 10 secondes pour l'accès aux pages des événements.	
	ENF-11 : L'agrégat doit offrir un temps de réponse acceptable pour les internautes au moment de la consultation des pages du portail (moins que 4 s)	
	Technologie, normes et Standards	
	ENF-12 : L'agrégat doit être développé en JEE/PLSQL.	
	ENF-13 : L'agrégat doit être développé dans une technologie qui permet son intégration au portail DSI.	
	Réutilisation	
	ENF-14 : L'agrégat doit réutiliser au maximum les composants déjà fournis par le Portail VDM.	
Autres	N/D	

7.4.2 Les particularités de l'exécution

Après l'exécution de l'activité ACT.01, nous constatons que le besoin du département DSI[33] de la VDM est d'offrir, à travers son portail Internet actuel, les services additionnels suivants :

[33] DSI : Direction des Services Informatiques

- affichage d'un calendrier des événements culturels et sportifs;
- consultation et réservation des activités des loisirs;
- affichage à l'externe de quelques annonces internes du portail Intranet;
- recherche des événements avec deux modes (simple et complexe).
- affichage de la liste des annonces de la VDM

En analysant les informations de base de SOCAM, nous remarquons que le client connait les composants participants, soient :

- Portail-VDM: Composant Portail de la VDM;
- Calendrier: Composant responsable d'afficher les événements sous forme de liste/mini-liste et calendrier;
- Annonces-VDM: Composant responsable de fournir la liste des annonces (communiqués, avis, appels, offre emploi);
- Loisirs En Ligne (LEL): Composant responsable de fournir la liste des activités sportives et culturelles de loisir.

En pratique, l'agrégation consiste à intégrer dans la page des activités et des loisirs du portail Internet de la VDM les services énumérés ci-dessus. Pour étudier l'agrégation en question, nous commençons par ajouter les composants à l'entrepôt SOCOM de la VDM. Ceci correspondant à l'exécution de l'activité ACT.02 de SOCAP. Les fiches SOCOM des composants de cette agrégation sont fournies dans l'ANNEXE II.

Durant l'exécution de l'activité ACT.03, nous n'avons pas réalisé de recherche sur les composants qui sont déjà prédéfinis par le contexte de l'agrégation. Avec cette composition statique déjà proposée, nous avons directement évalué la faisabilité de l'agrégation (ACT.04). Les composants participants sont de technologies hétérogènes : Portail VDM en JAVA avec la technologie portail d'Oracle. Le portail de loisir en ligne est développé en SharePoint avec une technologie .NET. De ce fait, SOCAP détecte automatiquement plusieurs incompatibilités des plateformes suite à la première vérification. Par exemple:

1. Le composant LEL est un composant .NET, alors que les autres sont des composants JEE.
2. Le composant LEL utilise est un portail SharePoint avec un serveur d'application .Net alors que les autres utilisent « JEE Oracle Container ».
3. Le composant LEL utilise un SGBDR MS-SQL Server alors que les autres composants utilisent la base de données « Oracle 9i Database ».
4. Le composant LEL utilise Windows NT Server 2003 comme OS.

Ces dernières incompatibilités sont potentielles et seront retirées suite à l'activité de leur résolution (ACT.05). Leur retrait est dû à ce que, dans le contexte de cette agrégation, les composants agrégés offrent leur service à partir de leur plateforme d'origine. En effet, le besoin réel est de juxtaposer des interfaces graphiques et des liens – amenant aux composants participants – depuis le portail VDM. Autrement dit, la VDM veut offrir aux internautes une interface graphique intégrée dans une seule page web, avec une authentification unique, pour accéder aux services du portail et des activités des loisirs offerts. Ainsi, nous n'avons pas besoin d'unifier ou de faire interopérer les plateformes des bases de données, des serveurs d'applications ou des conteneurs des composants JEE et .NET, mais plutôt d'agréger les interfaces utilisateurs de ces composants. Toutefois, quelques incompatibilités détectées nécessitent une résolution telles que:

- « Le composant LEL utilise le répertoire Active Directory (AD) alors que le portail VDM utilisent le répertoire l'OID (Oracle Internet Directory) ».
- « Le composant LEL est hébergé dans un réseau privé de la compagnie M2e[34] qui n'est pas connecté au réseau de la VDM ».

Ces deux problèmes sont respectivement de types 'Sécurité et identité' et 'Zonage réseau' (voir notre classification de la section 5.4). Deux solutions sont proposées pour les résoudre durant l'activité ACT.05. La première propose d'authentifier les utilisateurs du portail-VDM au répertoire AD du portail LEL vu que les répertoires des utilisateurs sont déjà synchronisés

[34] M2e : Compagnie privée qui offre et héberge le site de loisir en ligne de la VDM.

dans un autre projet de la VDM (OID et AD) et la deuxième consiste à rendre accessible – à l'aide d'un réseau virtuel – le portail des loisirs en ligne dans la zone Internet de M2e.

Aussi, toutes les incompatibilités détectées avec les métadonnées des services seront retirées. Leur retrait est dû à ce que l'agrégat ne requiert pas l'interopérabilité ni la composition des services de la logique d'affaires des composants sources. L'agrégat fournit un ensemble d'interfaces des utilisateurs en juxtaposant celles fournies de tous les composants agrégés. Les mêmes interprétations sont valides pour les incompatibilités des attributs d'agrégation. Elles sont retirées durant l'activité de résolution des incompatibilités puisque l'agrégat ne requiert par une intégration forte entre les composants implantés avec des langages de programmation différents. Le même raisonnement s'applique au reste des attributs d'agrégation: le degré d'accessibilité, les standards, les frameworks, les méthodes d'agrégation.

À ce stade de l'exécution, et après la résolution des incompatibilités, l'agrégation des composants participants est faisable. Nous passons à l'activité de conception pour classifier cette agrégation et déterminer sa composition dynamique. Si nous appliquons un raisonnement par élimination, cette agrégation n'est pas une connexion parce que nous avons quatre composants participants. Aussi nous écartons la fusion vu qu'il y a deux composants de technologies hétérogènes (Portail VDM et le composant Loisir En Ligne), ainsi nous n'avons pas de plateforme unique de déploiement ni l'idée d'un seul composant agrégat physique. En analysant la solution potentielle, nous constatons que le résultat final voulu est une page web qui juxtapose chacun des composants de présentation des composants participants. Nous ajoutons un « P » pour les noms des composants participants évalués pour rappeler que nous analysons uniquement leur interaction de la couche de présentation. Nous avons appliqué les équations des agrégations par collection (section 3.3.3) et par coordination (section 3.3.4) entre les composants de présentation et avons obtenu les résultats suivants.

Tableau 7.5 Métadonnées d'agrégation SOCOM des composants participants à une agrégation par collection

Métadonnées d'agrégation	Composant-Portail-VDM-P	Composant-Annonces-VDM-P	Composant-Calendrier-P	Composant-LEL-P
Potentiel de couplage	Mixte	Mixte	Mixte	Mixte
Degré d'accessibilité	Boîte blanche	Boîte grise	Boîte blanche	Boîte grise
Couche logicielle	Présentation	Présentation	Présentation	Présentation
Nœud du système	ville.montreal.qc.ca	ville.montreal.qc.ca	Prod55.ville.montreal.qc.ca	ludik.ville.montreal.qc.ca
Point d'entrée du composant	www.ville.montreal.qc.ca	www.ville.montreal.qc.ca/ AnnoncesPortlet/	rechercherElementContenuImpl.java	http://ca.qc.ludik.ville.montreal/services/activityservice.asmx
Méthode d'agrégation	API JAVA	API JAVA	API JAVA	API JAVA
Type de langage	Compilé	Compilé	Compilé	Compilé
Compatibilité avec les standards	JEE, Spring, PL-SQL, JDK 1.4, Portlet	Portlet	JDK 1.4, Portlet	Web Parts, Framework .NET
Référence du composant agrégat				
Plateforme de déploiement (SOCOM)	Oracle Portal 9i	Oracle Portal 9i	Oracle Portal 9i	SharePoint Server

À partir des valeurs des attributs des composants, nous appliquons la vérification des règles de l'agrégation par collection:

- (**DA**(Composant-LEL-P) = {BG}, **DA**(Composant–ANNONCES-VDM-P) = {BG}, **DA**(Composant-CALENDRIER-P) = {BB} et **DA**(Composant-PORTAIL-VDM-P) = {BB}; **RegAccess** est vérifiée.
- (**PC**(Composant-LEL-P) = {CM}, **PC**(Composant-Annonces-VDM-P) = {CM}, **PC**(Composant-CALENDRIER-P) = {CM} et **PC**(Composant-PORTAIL-VDM-P) = {CH}, ***CouplageFaisable***(Composant--PORTAIL-VDM-P, Composant–Annonces-VDM-P, Composant-CALENDRIER-P, Composant-LEL-P) = **VRAI**; **RegCoup** est vérifiée.
- (**CL**(Composant-LEL-P) = **CL**(Composant–Annonces-VDM-P) = **CL**(Composant-CALENDRIER-P) = **CL**(Composant-PORTAIL-VDM-P) = {P} . **RegCouche** est vérifiée.
- Les composants participants sont déployés sous TOMCAT dans une plateforme JAVA identique avec celle du composant collectionneur. **RegCompat** est vérifiée.

À partir des valeurs des attributs des quatre composants, nous appliquons la vérification des règles de l'agrégation par coordination et nous obtenons les confirmations suivantes :

- **(CL**(Composant-PORTAIL-VDM-P) = T, **CL**(Composant-LEL-P) = T, **CL**(Composant-Calendrier-P) = M, **CL**(Composant-Annonce-VDM-P) = M). **RegCouche n'est pas vérifiée**.
- (**DA**(Composant-LEL-P) = {BG}, **DA**(Composant–ANNONCES-VDM-P) = {BG}, **DA**(Composant-CALENDRIER-P) = {BB} et **DA**(Composant-PORTAIL-VDM-P) = {BB}; ***InclutUnComposantBoîteBlanche***(Composant--PORTAIL-VDM-P, Composant–Annonces-VDM-P, Composant-CALENDRIER-P, Composant-LEL-P) = VRAI. **RegAccess** est vérifiée.
- (**PC**(Composant-LEL-P) = {CM}, **PC**(Composant-Annonces-VDM-P) = {CM}, **PC**(Composant-CALENDRIER-P) = {CM} et **PC**(Composant-PORTAIL-VDM-P) = {CH}, ***CouplageFaisable***(Composant--PORTAIL-VDM-P, Composant–Annonces-

VDM-P, Composant-CALENDRIER-P, Composant-LEL-P) = **VRAI**; **RegCoup** est vérifiée.

Après l'évaluation de la classification, nous sommes dans un contexte d'agrégation par collection et le composant portail-VDM joue le rôle de collectionneur parce que l'agrégation est faite dans une de ses pages web. Le composant LEL n'est pas agrégé dans les zones de la page web du portail VDM, un lien web est ajouté pour diriger l'internaute vers le portail LEL. Le portail-VDM vérifie les paramètres d'accès de l'utilisateur dans son répertoire d'utilisateurs avant de lui donner accès au portail LEL. Un développement supplémentaire est requis pour implanter cette vérification.

Dans le contexte de cette agrégation, toutes les activités du modèle de premier niveau sont exécutées, à part les activités du sous processus associé à « ACT.06-Concevoir l'agrégat ». Les seules sous activités exécutées de ce sous processus sont ACT.06.01, ACT.06.02, ACT.06.04 et ACT.06.12.

Cette agrégation par collection a produit un agrégat qui regroupe les composants de présentation dans une même interface graphique web du portail VDM. La Figure 7.4 montre la page web du portail VDM jouant le rôle de collectionneur et des cadres noirs des composants collectionnés (1- accès au portail LEL, 2- affichage du calendrier des évènements de la VDM, 3- affichage des annones). Le composant portail-VDM collectionneur est un composant agrégat logique.

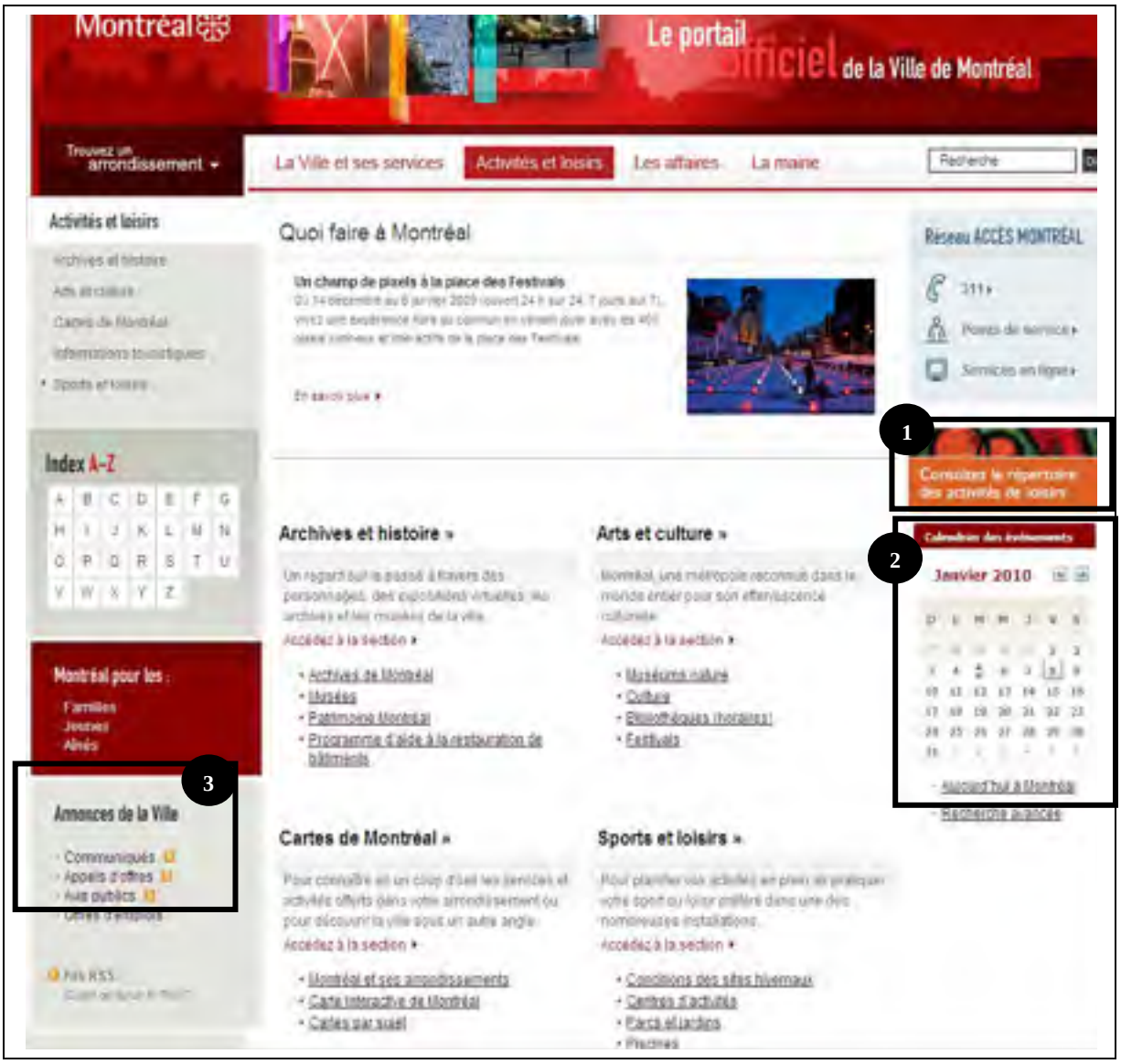


Figure 7.4 Page web du composant Agrégat : Portail Loisir En Ligne - VDM

La figure suivante présente la modèle UML des composants de notre agrégat par collection.

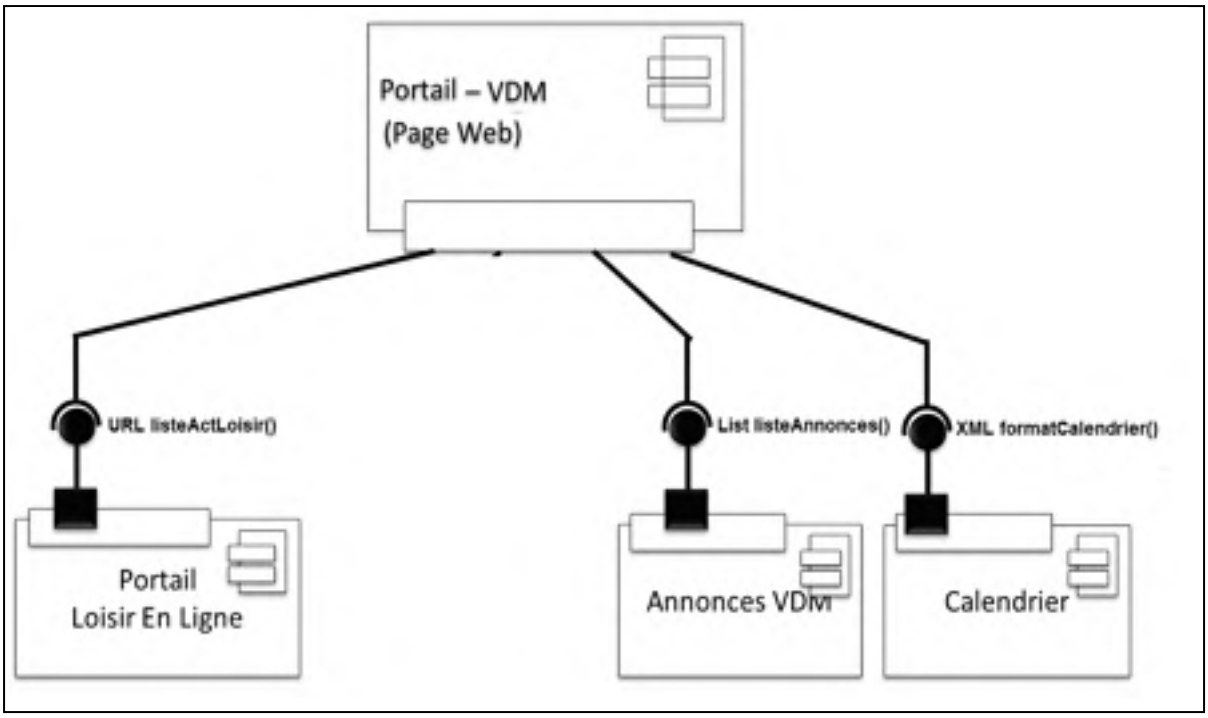


Figure 7.5 Modèle UML2 du composant Agrégat : Portail Loisir En Ligne – VDM

7.5 Agrégation par fusion: « Calendrier » de la VDM

7.5.1 Fiche SOCAM de l'agrégat

Au début de l'exécution, nous avons rempli la fiche de l'agrégat « Portail-outils » avec les informations de base suivantes : le nom du composant agrégat, le nom de l'auteur, les hypothèses, la description générale, les hypothèses, les cas d'affaires et quelques exigences non fonctionnelles. Ces informations introduisent le contexte et la problématique que l'agrégation propose de résoudre.

Tableau 7.6 Fiche SOCAM de l'agrégat par Fusion - Calendrier

Nom	Calendrier	*(Contenu complété après l'exécution de l'activité ACT.01)*
Auteur	Ville De Montréal (VDM)	*(Contenu complété après l'exécution de l'activité ACT.01)*
Description générale	La Ville De Montréal veut offrir à ses utilisateurs de son portail Internet et ses employés du portail Intranet un « Calendrier » qui informe de tous les événements ayant lieu dans l'ile de Montréal.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Hypothèses	1. L'agrégat sera déployé dans les environnements Internet et Intranet de la VDM 2. L'agrégat doit pouvoir donner accès à des éditeurs de l'extérieur de la VDM. (utilisateurs externes privilégiés)	*(Contenu complété après l'exécution de l'activité ACT.01)*
Composition statique	« Calendrier » est un composant agrégat composé des composants suivants : 1. GestionnairePhoto-PortailVDM : Composant responsable de la gestion des ressources numériques de type Photo. 2. **Portail-VDM** : Portai-VDM responsable d'intégrer les composants et d'offrir l'authentification; 3. **GFEC** : Composant responsable de la gestion des Formulaires et du Contenu; 4. **GPRO** : Composant responsable de la Gestion des Processus « Cycle d'approbation »; 5. **GREA** : Composant responsable de la recherche et de l'affichage.	*(Contenu complété après l'exécution de l'activité ACT.03.11)*
Cas d'affaires	« Calendrier » devrait offrir les cas d'affaires suivants : **1.** Publier les événements publics et privés (Intranet) ; **2.** Rechercher les événements (modes simple et complexe); **3.** Consulter les événements avec photos en mode liste, mini-liste et calendrier; **4.** Consulter les détails d'un événement; **5.** Gérer les événements publics et privés (Internet/Intranet).	*(Contenu complété après l'exécution de l'activité ACT.01)*

Catégorie de l'agrégation	**Fusion**	*(Contenu complété après l'exécution de l'activité ACT.06.01)*
Patron d'agrégation	**Fusion en partie**	*(Contenu complété après l'exécution de l'activité ACT.06.09 et ACT.06.10)*
Composition Dynamique	« Calendrier » est un composant agrégat. Il est composé des composants suivants : **1. Services administratifs** SER-1 : Créer les types d'événements. SER-2 : Créer les événements. SER-3 : Créer les entités (les groupes de personnes qui seront responsables de la gestion du contenu des événements). SER-4 : Créer les langues. SER-5 : Créer les plateformes (Internet/Intranet/extranet). SER-6 : Créer les formulaires pour un type d'événement. SER-7 : Créer les listes de valeurs (LoVs). SER-8 : Créer les Cycles d'Approbation (CA). SER-9 : Créer les groupes d'utilisateurs. SER-9bis : Créer les utilisateurs. SER-10 : Créer les privilèges d'accès des types d'événements. SER-11 : Associer un ou plusieurs privilèges d'accès aux types d'événements. SER-12 : Associer une ou plusieurs LoVs aux types d'événements. SER-13 : Associer les CAs aux types d'événements. **2. Services de gestion:** SER-14: Gérer les événements. SER-15 : Gérer des ressources numériques. SER-16 : Appliquer un cycle d'approbation. SER-17 : Utiliser un formulaire d'événement. **3. Services publics :** SER-18 : Rechercher les événements. SER-19 : Consulter les événements à partir d'une liste. SER-20 : Consulter les événements à partir d'une mini-liste.	*(Contenu complété après l'exécution de l'activité ACT.06.12)*

	SER-21: Consulter les événements à partir d'un calendrier. SER-23 : Consulter le détail d'un événement.	
Exigences non fonctionnelles	**Sécurité :** - ENF-01 : Authentification à « Calendrier » doit hériter des éléments de sécurité (authentification unique) du Portail de la VDM (utiliser le même OID que le portail, Oracle Identity Directory).	*(Contenu complété après l'exécution de l'activité ACT.01 et une mise à jour de l'ACT.10)*
	- ENF-02 : Authentification faible requise (nom d'utilisateur/mot de passe). - ENF-03 : L'accès à l'application Calendrier se fera de façon sécurisée.	
	Disponibilité :	
	- ENF-04 : Calendrier doit être disponible 24/7 à 95% par année.	
	- ENF-04bis : Calendrier doit disponible à partir des interfaces graphiques du Portail-VDM.	
	Volumétrie-Charge	
	- ENF-05 : le portail-outils doit permettre un accès à 10 000 personnes.	
	- ENF-06 : 300 accès concurrents sont prévisibles.	
	- ENF-07 : Calendrier doit supporter un volume important de données (plus de 200 000 enregistrements par table maîtresse et par application).	
	Ergographie	
	- ENF-08 : Calendrier doit être conforme au PIV du Portail VDM et aux normes W3C d'accessibilité.	*(Contenu complété après l'exécution de l'activité ACT.01)*
	- ENF-08bis : Calendrier et les composants participants sont supportés par Internet Explorer Version 6.	*(Contenu complété après l'exécution de l'activité ACT.10)*
	Performance – Temps de réponse	
	- ENF-09 : Calendrier doit répondre aux requêtes simples des utilisateurs en moins de 4 s pour l'accès aux pages des événements.	
	- ENF-10 : Calendrier doit répondre aux requêtes complexes des utilisateurs en moins de 10 s pour l'accès aux pages des événements.	

	- ENF-11 : Calendrier doit offrir un temps de réponse acceptable pour l'affichage des résultats de recherche des événements (moins de 4 s)	
	Technologie, normes et standards	
	- ENF-12 : « Calendrier » doit être développé en JEE.	
	- ENF-13 : Les portlets utilisés doivent être compatible JSR 168 et WSRP.	
	- ENF-14: Le noyau de l'application « Calendrier » doit être développé indépendamment de la plateforme Oracle, mais intégrable dans le portail Oracle de la VDM.	
	Réutilisation	
	ENF-15 : « Calendrier » doit réutiliser les mêmes ressources médiathèques que le Portail-VDM.	
Autres	N/D	

7.5.2 Particularités de l'exécution

Après l'exécution de l'activité ACT.01, nous comprenons que l'équipe du département DSI souhaite offrir à ses utilisateurs du portail-VDM une application « Calendrier » qui informe sur tous les événements de l'ile de Montréal. « Calendrier » doit satisfaire les besoins d'affaires suivants :

- publier les événements publics et privés (Intranet);
- rechercher les événements (modes simple et complexe);
- consulter les événements avec photos en mode liste, mini-liste et calendrier;
- consulter les détails d'un événement;
- gérer les événements publics et privés (Internet/Intranet).

L'activité ACT.02 n'est pas exécutée parce que les composants sont déjà référencés dans l'entrepôt SOCOM de la Ville De Montréal. En exécutant l'activité ACT.03, nous avons cherché dans cet entrepôt SOCOM des composants qui satisfont les besoins énoncés. Nous avons sélectionné le composant « Portail-VDM » qui représente le contenant qui héberge le composant agrégat. Le potentiel d'affichage et d'intégration du portail-VDM est utilisé pour

aider au développement de l'application « Calendrier ». À partir des énoncés des besoins, nous avons recherché les autres composants en utilisant les différentes catégories de métadonnées SOCOM et des mots clés « événement », « photo », « formulaire », « Recherche », « cycle d'approbation », « affichage du contenu». Nous avons obtenu la liste des composants logiciels potentiels suivants : « Médiathèque », « GénérateurFormulaire-PortailVDM », « GestionnairePhoto-PortailVDM », « GPRO », « GFEC », « GREA », « Portail-VDM ». Les composants retrouvés ne sont pas tous valides. Nous avons comparé les métadonnées SOCOM des composants qui fournissent les mêmes services et nous avons écarté les composants « GénérateurFormulaire-PortailVDM » et « Médiathèque ». Le premier composant écarté ne satisfait pas les besoins d'ergonomie d'affichage des formulaires satisfaisant les normes d'accessibilité W3C. Le second est développé en PHP et ne s'intègre pas facilement avec le portail-VDM qui est développé en Java. Par conséquent, ses composants ne satisfont pas les besoins du composant agrégat « Calendrier » et sont remplacés respectivement par les composants « GFEC » et « GestionnairePhoto-PortailVDM ». Ainsi nous avons mis à jour la fiche SOCAM avec la liste des composants retenus suivants:

- GestionnairePhoto-PortailVDM : Composant responsable de la gestion des ressources numériques de type Photo;
- Portail-VDM : Portail-VDM responsable d'intégrer les composants et d'offrir l'authentification;
- GFEC : Composant responsable de la gestion des formulaires et du contenu;
- GPRO : Composant responsable de la gestion des processus « Cycle d'approbation »;
- GREA : Composant responsable de la recherche et de l'affichage.

Durant l'exécution du sous processus d'évaluation de la faisabilité d'agrégation (ACT.04), les incompatibilités détectées ne sont pas toutes retenues et, même lorsqu'elles le sont, elles peuvent être résolues rapidement vu que les composants sont développés dans la même plateforme avec des langages interopérables (PLSQL et Java). Les composants sources utilisent des technologies homogènes. De ce fait, SOCAP détecte davantage des incompatibilités au niveau des métadonnées des services et moins d'incompatibilités des

métadonnées des plateformes, non fonctionnelles et d'agrégation de SOCOM. Ces dernières sont potentielles et sont retirées suite à l'activité de résolution des incompatibilités. Leur retrait est dû à ce que, dans le contexte de cette agrégation, les appels aux services des composants agrégés se feront par un appel aux méthodes natives dans un même langage de programmation ou via des langages interopérables.

En effet, les problèmes d'agrégation se résolvent essentiellement avec des techniques de codage, d'adaptation et de transformation des données dans une même plateforme et un même langage de programmation. La majorité des composants agrégés offrent leurs services à partir des composants sources déployés dans la même plateforme. Ainsi, le respect des exigences non fonctionnelles est plus facile à implanter que les autres types d'agrégation. Souvent, un codage supplémentaire suffit pour implanter ces exigences dans le langage de programmation du composant agrégat.

À ce stade de l'exécution, l'agrégation des composants participants est faisable. Nous passons à l'activité de conception (ACT.05) pour classifier cette agrégation et déterminer sa composition dynamique. Si nous appliquons un raisonnement par élimination, cette agrégation n'est pas une connexion puisque nous avons plus de deux composants participants. Aussi nous écartons la collection et la coordination, puisque les interactions entre les composants sélectionnés se font au niveau des couches de présentation et de traitement. Nous évaluons si cette agrégation satisfait notre définition d'agrégation par fusion dont la définition est présentée à la section 3.3.5.

Tableau 7.7 Métadonnées d'agrégation SOCOM des composants participants à une agrégation par fusion

Métadonnées d'agrégation	Composant-Portail-VDM (A)	Composant-GestionnairePhoto-VDM (B)	Composant-GFEC (C)	Composant-GREA (D)	Composant-GPRO (E)
Potentiel de couplage	Horizontal	Mixte	Mixte	Horizontal	Horizontal
Degré d'accessibilité	Boîte grise	Boîte blanche	Boîte blanche	Boîte blanche	Boîte blanche
Couche(s) logicielle(s)	Présentation-Traitement - Données	Traitement, Présentation	Traitement, Présentation	Traitement	Traitement, Présentation
Nœud du système	ville.montreal.qc.ca	Prod55.ville.montreal.qc.ca	Prod55.ville.montreal.qc.ca	Prod55.ville.montreal.qc.ca	Prod55.ville.montreal.qc.ca
Point d'entrée du composant	www.ville.montreal.qc.ca/admin/	./gererFormulaireEtContenu	./gererFormulaireEtContenu	./rechercherElementContenuImpl.java	./gererFormulaireEtContenu
Méthode d'agrégation	Portlet Oracle, API Java	Connecteur PL-SQL/ Portlet Oracle	Connecteur JAVA / Portlet Oracle	Connecteur Java / Portlet Oracle	Connecteur JAVA/ Portlet Oracle
Type de langage	Compilé	compilé	compilé	Compilé	compilé
Compatibilité avec les standards	jdk 1.4, Portlet, XML	XML, PL-SQL	JEE, jdk 1.4, XML	JEE, jdk 1.4, XML	JEE, jdk 1.4, XML
Référence du composant agrégat	N/A	Composant-Portail-VDM (A)	Composant-Portail-VDM (A)	Composant-Portail-VDM (A)	Composant-Portail-VDM (A)
Plateforme de déploiement (SOCOM)	Portail Oracle	Portail Oracle	Portail Oracle	Portail Oracle	Portail Oracle

À partir des valeurs des attributs des composants, nous appliquons la vérification des règles de l'agrégation par fusion :

- **(CL**(Composant-A) = {P, T, D}, **CL**(Composant-B) = {P, T}, **CL**(Composant-C) = {P, T}, **CL**(Composant-LRS) = {T} et **CL**(Composant-E) = {T}) Et ***CoucheVoisineImmediate***(A, B, C, D, E) = VRAI); **RegCouche** est vérifiée.
- (**DA**(Composant-A) = {BG}, **DA**(B, C, D et E) = {BB}, ***InclutUnComposantBoîteBlanche***(A, B, C, D et E) = Vrai . **RegAccess** est vérifiée.
- (**PC**(A, D et E) = CH**, PC**(B, C) = CM et ***CouplageFaisable***(A, B, C, D, E) = VRAI. **RegCoup** est vérifiée.
- Selon les valeurs de l'attribut « compatibilité avec standards » et celles des plateformes de SOCOM, la règle **RegCompat** est vérifiée. Le composant Java peut invoquer des fonctions PL-SQL, ainsi les deux langages sont interopérables.

La présente agrégation valide les équations de la fusion. Tous les composants participants sont déployés dans la même unité de déploiement que le portail-VDM. Par conséquent, ils satisfont le patron "fusion en partie" vu que le composant « Portail-VDM » est réutilisé pour encapsuler l'agrégat « Calendrier »

Durant l'exécution du processus SOCAP, dans le contexte d'une agrégation par fusion, toutes les activités du modèle de premier niveau sont exécutées à part certaines activités du sous processus associé à l'activité « ACT.06-Concevoir l'agrégat ». Les seules sous activités exécutées de ce sous processus sont ACT.06.01, ACT.06.09, ACT.06.10 et ACT.06.12.

Nous remarquons que les étapes de développement, de configuration, des tests sont typiquement simples à réaliser, étant donné des travaux d'agrégation sont souvent appliqués sur des composants de type boîte blanche ou boîte grise au niveau individuel ainsi pour l'ensemble des composants, le tout pour satisfaire le besoin d'une fusion. Cette simplicité provient du fait que le composant agrégat est finalement déployé dans une même plateforme que les composants sources. Souvent, le processus de développement du composant agrégat est similaire à un processus de développement logiciel classique. Dans ce contexte

d’agrégation, le nouvel agrégat est un composant physique, déployé dans la plateforme Oracle, contenant tous les composants participants. La figure suivante présente la modèle UML des composants de notre agrégat par fusion.

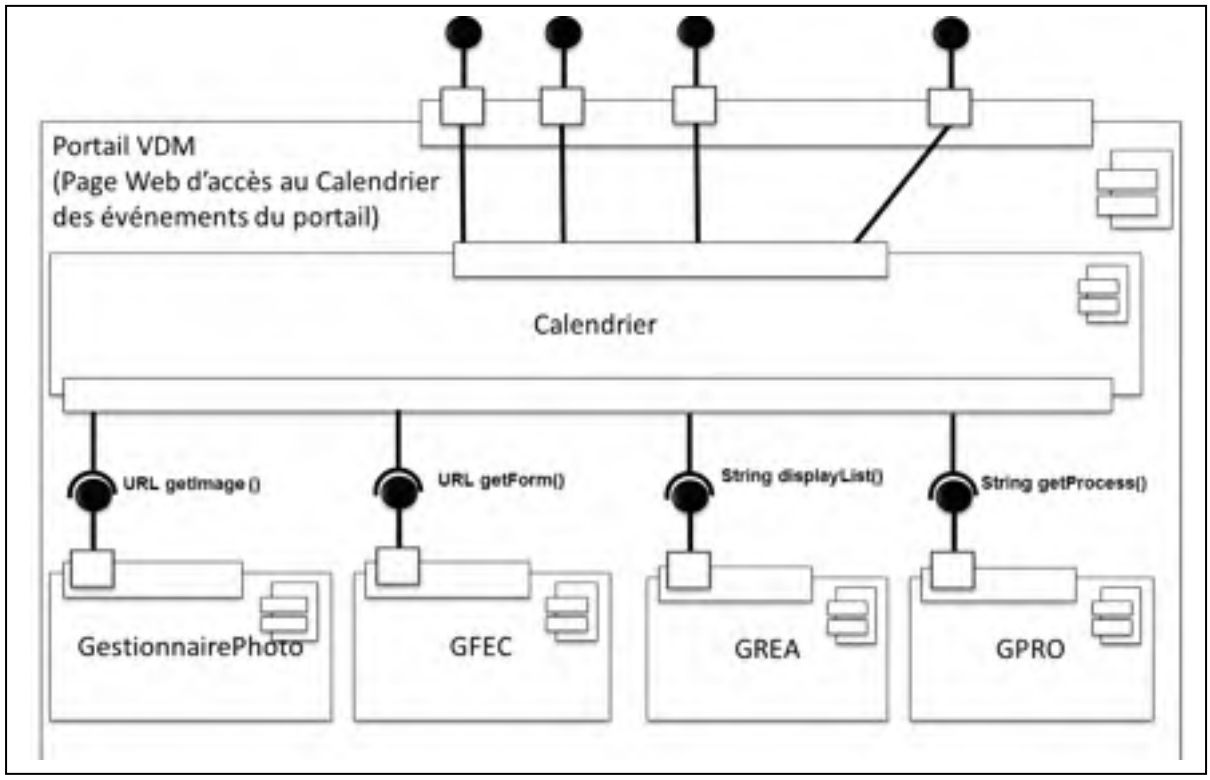


Figure 7.6 Modèle UML2 de l'agrégat par Fusion : Calendrier

7.6 Agrégation par connexion: «Télépaiement de Loisir-En-Ligne – VDM »

7.6.1 Fiche SOCAM de l’agrégat

Au début de l’exécution, nous avons rempli la fiche de l’agrégat « Portail Loisir en ligne – VDM » avec les informations de base suivantes : le nom du composant agrégat, le nom de l’auteur, la description générale, les hypothèses, les cas d’affaires et quelques exigences non fonctionnelles. Ces informations introduisent le contexte et la problématique que l’agrégation propose de résoudre.

Tableau 7.8 Fiche SOCAM de l'agrégat par Connexion :
«Télépaiement de Loisir-En-Ligne – VDM »

Nom	Télépaiement de Loisir-En-Ligne – VDM	*(Contenu complété après l'exécution de l'activité ACT.01)*
Auteur	Ville De Montréal (VDM)	*(Contenu complété après l'exécution de l'activité ACT.01)*
Description générale	Le projet consiste à fournir aux citoyens de la VDM la possibilité de payer leurs activités de loisir en ligne. Toutefois, le portail Loisir En Ligne étant indépendant du portail VDM, le module de paiement doit réutiliser le même que celui du portail VDM, soit le composant de Télépaiement.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Hypothèses	1. L'agrégat sera déployé dans les environnements Internet de la section des activités culturelles du portail Loisir En Ligne 2. Les internautes seront redirigés vers le module de Télépaiement du portail VDM lorsqu'ils auront à payer les activités des loisirs qu'ils ont choisies.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Composition statique	Loisir-En-Ligne est un composant agrégat composé des composants suivants : 1. Portail Loisir En Ligne(LEL): Composant Portail LEL de la Ville De Montréal. 2. Télépaiement : Composant responsable de gérer le paiement des services ou des produits sélectionnés dans les portails concernés.	*(Contenu complété après l'exécution de l'activité ACT.03.11)*
Cas d'affaires	Le composant agrégat doit supporter le cas d'affaires suivant : 1. Permettre à un citoyen de la VDM de payer les activités de loisirs sélectionnées dans le Portail LEL.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Catégorie de l'agrégation	**Connexion**	*(Contenu complété après l'exécution de l'activité ACT.06.01)*
Patron d'agrégation	**Connexion bidirectionnelle**	*(Contenu complété après l'exécution de l'activité ACT.06.09 et ACT.06.10)*

Composition dynamique	L'agrégat "**Télépaiement de Loisir-En-Ligne - VDM** " est un composant qui doit implémenter les services suivants : SER-1: Sélectionner les activités du portail LEL SER-2: Payer en ligne les activités sélectionnées	*(Contenu complété après l'exécution de l'activité ACT.06.12)*
Exigences non fonctionnelles	**<u>Sécurité :</u>** ENF-01: Les données des transactions des paiements sont confidentielles et doivent être protégées. ENF-02: Authentification faible requise pour accéder au portail LEL (nom d'utilisateur/mot de passe) ENF-03: L'accès à l'application "Bandeau vertical des activités et loisirs" se fera de façon sécurisée.	*(Contenu complété après l'exécution de l'activité ACT.01 et une mise à jour de l'ACT.10)*
	<u>Disponibilité :</u>	
	ENF-04 : le composant " **Télépaiement de Loisir-En-Ligne - VDM**" doit être disponible 24/7 à 99.5% par année	
	ENF-04bis: Le composant " **Télépaiement de Loisir-En-Ligne - VDM**" doit être disponible à partir des interfaces graphiques du Portail-LEL.	
	<u>Volumétrie-Charge</u>	
	ENF-05 : Le composant agrégat doit permettre un accès à 120 000 personnes	
	ENF-06 : 500 accès concurrents sont prévisibles	
	ENF-07 : L'agrégat doit supporter un grand volume de données (plus de 500 000 enregistrements par table maîtresse et par application)	
	<u>Ergographie</u>	
	ENF-08 : L'agrégat doit être confirme au PIV du Portail LEL et aux normes W3C d'accessibilité	*(Contenu complété après l'exécution de l'activité ACT.01)*
	ENF-08bis : L'agrégat et les composants participants sont supportés par Internet Explorer 7 et Firefox 3.0.	*(Contenu complété après l'exécution de l'activité ACT.10)*
	<u>Performance – Temps de réponse</u>	
	ENF-09 : L'agrégat doit répondre aux requêtes simples des utilisateurs en moins de 4 s pour l'accès aux pages des événements	
	ENF-10 : L'agrégat doit répondre aux requêtes complexes des utilisateurs en moins de 10 s pour l'accès aux pages des événements	

	ENF-11 : L'agrégat doit offrir un temps de réponse acceptable pour les internautes au moment de la consultation des pages du portail (moins que 4 s).	
	Réutilisation	
	ENF-12 : L'agrégat doit réutiliser le composant de télépaiement de la VDM.	
Autres	N/D	

7.6.2 Particularités de l'exécution

En analysant la description générale depuis la fiche SOCOM, nous comprenons que le portail LEL a besoin d'un service qui assure le paiement des activités de loisir. Pour ce faire, nous lançons une recherche dans l'entrepôt SOCOM de la VDM avec les mots clés « paiement » et « transaction ». Vu que ces mots décrivent les métadonnées des services et d'affaires du composant Télépaiement, il est retourné comme résultat de recherche par l'application. Nous soulignons la contribution des métadonnées SOCOM pour l'identification du composant dans l'entrepôt SOCOM de la VDM. De plus, lorsque l'architecte visualise les métadonnées, il s'informe davantage sur le composant de télépaiement qui offre un cas d'utilisation d'affaires et un service à travers une API Java. Ceci confirme la visibilité multi-dimensions qu'offrent les métadonnées SOCOM aux composants logiciels. Suite à l'exécution de l'activité de sélection (ACT.03), nous avons identifié la composition statique de l'agrégation entre les composants « Portail-LEL » et « Télépaiement de la VDM ».

Ce type d'agrégation se base essentiellement sur l'interopérabilité entre les services fournis et les services requis des composants participants. Nous avons cependant détecté des incompatibilités de type « Protocole des interfaces » et « Architecturales » (voir Figure 5.3). Ces incompatibilités sont souvent réelles et doivent être résolues par l'architecte. Dans le cas de la présente exécution, le portail LEL est développé en .Net alors que le composant de Télépaiement est un composant JEE. Par conséquent, l'API Java fourni par le composant JEE ne peut pas être directement consommée par le portail LEL. Nous détectons à ce niveau des

points d'entrées aux composants incompatibles. En plus, les services présentent des signatures différentes et des paramètres différents. Ainsi, nous avons un autre type d'incompatibilité au niveau des interfaces. La première évaluation de la faisabilité a échoué suite à l'exécution de l'activité ACT.04. Toutefois, à l'exécution de l'activité ACT.05, nous avons proposé une résolution à l'aide des services web sécurisés. La résolution proposée est faisable, ce qui nous permet de passer à l'activité de conception (ACT.05).

L'étape de conception de l'agrégat est souvent relativement simple et facile à réaliser, comparée aux trois autres catégories d'agrégation. Ceci s'explique du fait que souvent, l'un des composants fournit le service et l'autre le consomme. Ceci se traduit, dans le cas présent, par une transformation de l'API Java en un fichier WSDL pouvant être consommé par le code .NET du portail suite à la génération des clients appropriés. Aussi, une adaptation au niveau des types de données échangés est nécessaire pour permettre un échange de données dans les deux sens.

L'activité de classification de l'agrégation est évidente dans ce cas puisqu'il s'agit d'une agrégation par connexion entre le composant de traitement du « Portail-LEL » et le composant de « Télépaiement » appartenant lui aussi à la couche de traitement. Toutefois, nous vérifions quand même si l'agrégation en question satisfait ou non notre définition de la connexion. Le tableau suivant énumère liste les métadonnées d'agrégation des composants participants à la connexion.

Tableau 7.9 Métadonnées d'agrégation SOCOM des composants participants à une agrégation par connexion

Métadonnées d'agrégation	**Composant-Portail-LEL (LEL)**	**Composant-Télépaiement (TP)**
Potentiel de couplage	Mixte	horizontal
Degré d'accessibilité	Boîte grise	Boîte blanche
Couche logicielle	Traitement	Traitement

Métadonnées d'agrégation	Composant-Portail-LEL (LEL)	Composant-Télépaiement (TP)
Nœud du système	ludik.ville.montreal.qc.ca	ProdPaiement.ville.montreal.qc.ca
Point d'entrée du composant	http://Ludik/services/actService.asmx	APIPaiementDesJardins.class
Méthode d'agrégation	Connecteur .NET	API JAVA
Type de langage	Compilé	Compilé
Compatibilité avec les standards	Framework .NET 3.0	JEE 1.4
Référence du composant agrégat	N/A	N/A
Plateforme de déploiement (SOCOM)	.NET Framework 3.0 Server	Oracle JAVA Container

À partir des valeurs des attributs des deux composants, nous appliquons la vérification des règles de l'agrégation par connexion.

- (**DA**(Composant-LEL) = BG et **DA**(Composant-TP) = BB; **RegAccess** est vérifiée.
- (**PC**(Composant-LEL) = CM, **PC**(Composant-TP) = CH, ***CouplageFaisable***(Composant-LEL, Composant-TP) = **VRAI**; **RegCoup** est vérifiée.
- (**CL**(Composant-LEL) = T et **CL**(Composant-TP) = T. **RegCouche** est vérifiée.

Le résultat de l'agrégation est le couple (Composant-LEL, Composant-TP) qui est un agrégat logique, contrairement à physique. Nous concluons que l'agrégation entre les deux composants sélectionnés satisfait notre définition de la connexion. Les interactions entre les deux composants sont effectuées dans les deux sens, par conséquent, nous qualifions la connexion de bidirectionnelle suite à l'exécution des activités ACT.06.09 et ACT.06.10.

Dans le contexte de cette agrégation, nous avons exécuté toutes les activités du modèle de premier niveau de SOCAP à part quelques activités du processus associé à l'activité « ACT.06-Concevoir l'agrégat ». Les seules activités exécutées de ce processus sont ACT.06.01, ACT.06.02 et ACT.06.04.

Les étapes de développement, de configuration, des tests sont souvent réalisables étant donné que les composants demeurent déployés dans leur plateforme. Aussi, la majorité des travaux d'agrégation sont concentrés au niveau de la couche des services. Il suffit donc d'analyser le processus de paiement en ligne, de générer les codes clients des services web requis et développer le code nécessaire pour implanter la logique l'agrégation dans le portail LEL. La figure suivante présente la modèle UML des composants de notre agrégat par connexion.

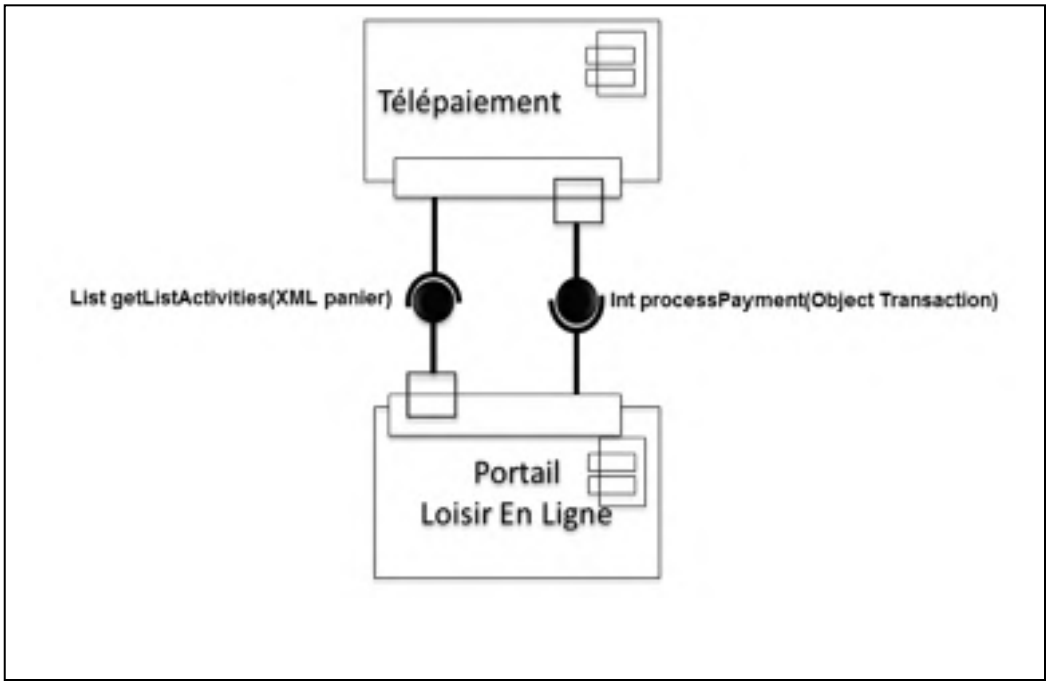


Figure 7.7 Modèle UML2 de l'agrégat par Connexion :
Télépaiement de Loisir En Ligne – VDM

7.7 Description des exécutions non réalisables

Parmi treize exécutions réalisées du processus SOCAP, quatre sont qualifiées de non réalisables. Le tableau suivant résume le contexte de chaque exécution et décrit les raisons de sa non faisabilité et les activités de SOCAP correspondantes exécutées.

Tableau 7.10 Description des agrégations non faisables

Nom de l'agrégat potentiel	Contexte de l'agrégation (Description générale)	Incompatibilités majeures identifiées	Règle(s) d'agrégation(s) non vérifiée(s) (suite à l'exécution de ACT.04)	Raisons de non faisabilité (suite à l'exécution de ACT.05)
« Portail-VDM – Médiathèque »	La Ville De Montréal veut centraliser la gestion des photos et des ressources numériques dans une seule application. Les portails Internet et Intranet devront utiliser cette source pour publier les événements et du contenu avec photo.	PLT-01 : la plateforme Médiathèque est développée en PHP alors que le portail VDM est développé avec JAVA et déployé dans un Conteneur Oracle 9i ENF-09 : Le portail VDM doit répondre aux requêtes simples des utilisateurs en moins de 4 s pour l'accès aux pages des événements	RA-03 : PLT-01 peut être résolue avec des liens http qui permettent d'accéder à la photo et aussi avec des interactions de type service web. RA-02 : L'exigence ENF-09 n'est pas satisfaite.	Les portlets Oracle qui forment les pages web du portail VDM exécutent plusieurs requêtes pour construire la page et puisque les photos ne sont pas stockées dans la base de données Oracle, parfois la requête génère un « Time Out ». L'agrégation est annulée par l'équipe d'affaires.
« GénérateurFormulaire-PortailVDM – Calendrier»	Pour produire le contenu des événements du Calendrier dans le portail-VDM, nous avons besoin d'un composant qui, à partir de la définition d'un type de contenu, génère des formulaires web dynamique.	ENF-03 : Les formulaires générés pour l'édition des événements du Calendrier de la VDM doivent répondre aux normes d'ergonomie et d'accessibilité W3C (exigence stricte du client de la VDM, les éditeurs de contenu).	RA-02 : L'exigence ENF-09 n'est pas satisfaite. Le module PLSQL du portail génère des formulaires qui ne peuvent pas juxtaposer deux champs d'un numéro de téléphone.	Les éditeurs de contenu de la VDM n'ont pas accepté le composant des formulaires. Contraintes techniques dans les portlets PLSQL résolues dans la version 10g du portail Oracle. L'agrégation est annulée par l'équipe d'affaires.

Nom de l'agrégat potentiel	Contexte de l'agrégation (Description générale)	Incompatibilités majeures identifiées	Règle(s) d'agrégation(s) non vérifiée(s) (suite à l'exécution de ACT.04)	Raisons de non faisabilité (suite à l'exécution de ACT.05)
« TivoliWebTop – SecurNet – Documentum »	Telus veut utiliser une des pages web de Documentum pour accéder à des documents de sécurité et de configuration des pare-feu déposés respectivement dans SecurNet et TivoliWebTop.	ENF-01 (Sécurité). Les utilisateurs doiven t s'authentifier uniquement à Documentum et peuvent accéder aux documents dans TivoliWebTop et SecurNet.	RA-02 : L'exigence ENF-09 n'est pas satisfaite.	SecurNet et TivoliWebTop sont dans des zones réseaux sécurisées (accès avec jeton) et Documentum dans le TEN. Les architectes réseaux n'ont pas voulu ouvrir les ports sécurisés 443 à Documentum à cause de la confidentialité des informations dans les deux outils. D'où la solution de créer le portail-outil (voir section 7.3.2). L'agrégation est annulée par l'équipe d'affaires.
« GFEC-Médiathèque»	Dans le contexte de développement de l'application Calendrier, le générateur de formulaire GFEC utilise un gestionnaire de photos pour associé du contenu à des photos. Médiathèque était proposée aux éditeurs de contenu pour fournir les photos durant l'édition du contenu des événements.	PLT-03 (plateforme de déploiement). GFEC est déployé dans le portail Oracle. Il est développé en JSP et intégré via les portlets Oracle dans les pages web. Médiathèque est hébergée dans Apache. Il est développé en PHP.	RA-03 : L'exigence PLT-09 n'est pas satisfaite.	Malgré que la compagnie qui fournit Médiathèque ait proposé de développer une interface de services web, le groupe d'architecture de la VDM a décidé de ne pas introduire une autre technologie par manque de compétence PHP à la VDM et pour des raisons stratégiques. L'agrégation est annulée par l'équipe d'affaires.

Nous constatons que les agrégations annulées au début du processus d'agrégation ont deux causes principales. La première est technique. L'architecte et son équipe technique peuvent détecter des incompatibilités qu'ils ne peuvent pas résoudre à cause des environnements et des choix technologiques existants. La deuxième est de type stratégique.

L'équipe d'affaires peut justifier sa décision d'annulation des agrégations par des raisons budgétaires, de rentabilité ou par son respect des normes et des standards adoptés par son architecture d'entreprise.

7.8 Application des mesures du modèle de validation du processus SOCAP

Nous avons utilisé treize contextes d'agrégation pour mesurer l'indicateur d'achèvement IAS et celui de couverture des objectifs, ICOS. L'indicateur de couverture des activités est mesuré à l'aide de quatre contextes associés aux quatre types d'agrégation. Le choix des quatre contextes est basé sur les critères de disponibilité de la majorité des catégories des métadonnées SOCOM des composants participants et d'aboutissement de leur exécution à un agrégat. Aussi, la mesure de l'indicateur ICAS nécessite un exemple pour chaque type d'agrégation vu que le cheminement d'exécution est presque identique pour des cas d'agrégation de même type. Ces critères nous aident à parcourir de bout en bout tout le processus et favorisent l'évaluation de la couverture des activités et des objectifs de SOCAP par chacune des exécutions. De plus, le critère de disponibilité des métadonnées SOCOM nous permet de mieux exécuter nos activités d'évaluation de la faisabilité (ACT.04), puis celle d'analyse du rapport d'évaluation (ACT.05).

7.8.1 Mesure 1 : évaluation de l'Indice de Couverture des Activités suite à l'exécution de SOCAP, ICAS

Après chaque exécution, nous avons identifié les activités exécutées dans le tableau récapitulatif suivant (Tableau 7.11) pour déterminer le premier indicateur, nommé « **I**ndice de **C**ouverture des **A**ctivités suite à l'exécution de **S**OCAP, **ICAS ».** Il traduit le rapport du

nombre des activités distinctes exécutées dans tous les cas choisis par le nombre total des activités du processus SOCAP.

ICAS = 100 * (Nombre d'activités réalisées au moins une fois dans un des contextes) / Nombre total d'activités du processus SOCAP

Le processus SOCAP est proposé pour supporter quatre types d'agrégation, soit la connexion, la collection, la coordination et la fusion. Pour nous assurer que les contextes réels d'exécution sont bien choisis et permettent effectivement de parcourir les cheminements prédéfinis pour chaque type d'agrégation, nous évaluons l'indice de couverture ICAS. Par conséquent, quand l'indice ICAS avoisine 100%, nous pouvons conclure que les cas d'agrégation choisis parcourent effectivement les cheminements alloués pour les différentes catégories d'agrégation.

Le tableau suivant énumère l'ensemble des activités et sous-activités du processus SOCAP. Les colonnes représentent les quatre exécutions et, pour chaque cas, nous vérifions si l'activité est exécutée ou non. Dans la dernière colonne, on affiche 1 si l'activité en question est exécutée au moins une fois, sinon on affiche la valeur 0.

Tableau 7.11 Évaluation de couverture des activités suite aux exécutions de SOCAP

Identificateur des activités du processus SOCAP	**Agrégat par collection « Portail Loisir En Ligne – VDM »**	**Agrégat par connexion « Télé-paiement »**	**Agrégat par coordination « Portail-outils »**	**Agrégat par fusion « Calen-drier »**	**Couverture par activité (1=Couverte, 0=Non Couverte)**
ACT.01 – Définir un nouvel agrégat.	1	1	1	1	1
ACT.02 – Référencer les composants.	1	1	1	1	1

Identificateur des activités du processus SOCAP	Agrégat par collection « Portail Loisir En Ligne – VDM »	Agrégat par connexion « Télé-paiement »	Agrégat par coordination « Portail-outils »	Agrégat par fusion « Calen-drier »	Couverture par activité (1=Couverte, 0=Non Couverte)
ACT.02.01 - Préparer un composant.	1	1	1	1	1
ACT.02.02 – Ajouter une fiche de métadonnées statiques.	1	1	1	1	1
ACT.02.03 – Ajouter une fiche de métadonnées d'affaires.	1	1	1	1	1
ACT.02.04 – Ajouter une fiche de métadonnées de la qualité.	1	1	1	1	1
ACT.02.05 – Ajouter une fiche des métadonnées d'agrégation.	1	1	1	1	1
ACT.02.06 – Ajouter les métadonnées des services.	1	1	1	1	1
ACT.02.07 – Extraire les métadonnées des services.	0	0	1	1	1
ACT.02.08 – Enrichir les métadonnées des services.	0	0	1	1	1
ACT.02.09 – Ajouter une fiche de	1	1	1	1	1

Identificateur des activités du processus SOCAP	Agrégat par collection « Portail Loisir En Ligne – VDM »	Agrégat par connexion « Télé-paiement »	Agrégat par coordination « Portail-outils »	Agrégat par fusion « Calen-drier »	Couverture par activité (1=Couverte, 0=Non Couverte)
métadonnées des plateformes.					
ACT.03 – Sélectionner les composants de l'agrégat.	1	1	1	1	1
ACT.03.01 - Sélectionner le mode de recherche.	1	1	1	1	1
ACT.03.02 – Rechercher avec les critères « Affaires, Service, qualité, agrégation ».	1	1	1	1	1
ACT.03.03 – Visualiser la liste des composants résultats.	1	1	1	1	1
ACT.03.04 – Visualiser le détail du composant.	1	1	1	1	1
ACT.03.05 - Sélectionner le composant.	1	1	1	1	1
ACT.03.06 – Parcourir l'ontologie des composants SOCOM.	0	0	1	1	1
ACT.03.07 – Questionner	0	0	1	1	1

Identificateur des activités du processus SOCAP	Agrégat par collection « Portail Loisir En Ligne – VDM »	Agrégat par connexion « Télé-paiement »	Agrégat par coordination « Portail-outils »	Agrégat par fusion « Calen-drier »	Couverture par activité (1=Couverte, 0=Non Couverte)
l'ontologie à l'aide des requêtes.					
ACT.03.08 – Visualiser la liste des composants résultats.	0	0	1	1	1
ACT.03.09 – Visualiser le détail du composant.	0	0	1	1	1
ACT.03.10 - Sélectionner le composant.	1	1	1	1	1
ACT.03.11 – Mettre à jour la fiche SOCAM de l'agrégat (composition statique).	1	1	1	1	1
ACT.04 – Étudier la faisabilité de l'agrégation.	1	1	1	1	1
ACT.04.01 – Sélectionner le composant agrégat.	1	1	1	1	1
ACT.04.02 – Commencer l'application des règles d'évaluation de la faisabilité de l'agrégation.	1	1	1	1	1
ACT.04.03 –	1	1	1	1	1

Identificateur des activités du processus SOCAP	Agrégat par collection « Portail Loisir En Ligne – VDM »	Agrégat par connexion « Télé-paiement »	Agrégat par coordination « Portail-outils »	Agrégat par fusion « Calen-drier »	Couverture par activité (1=Couverte, 0=Non Couverte)
Évaluer avec son équipe la faisabilité financière.					
ACT.04.04 – Journaliser le refus financier à l'évaluation de l'agrégation.	0	0	0	0	0
ACT.04.05 – Appliquer la règle RA-01 : Compatibilité d'affaires.	1	1	1	1	1
ACT.04.06 – Journaliser les incompatibilités d'affaires identifiées.	1	1	1	1	1
ACT.04.07 – Appliquer la règle RA-02 : Compatibilité de la qualité.	1	1	1	1	1
ACT.04.08 – Journaliser les incompatibilités de la qualité identifiées.	1	1	1	1	1
ACT.04.09 – Appliquer la règle RA-04 : Compatibilité des services.	1	1	1	1	1
ACT.04.10 – Vérifier la compatibilité	1	1	1	1	1

Identificateur des activités du processus SOCAP	Agrégat par collection « Portail Loisir En Ligne – VDM »	Agrégat par connexion « Télé-paiement »	Agrégat par coordination « Portail-outils »	Agrégat par fusion « Calen-drier »	Couverture par activité (1=Couverte, 0=Non Couverte)
des services.					
ACT.04.11 – Journaliser les incompatibilités des services.	1	1	1	1	1
ACT.04.12 – Appliquer la règle RA-05 : Compatibilité des attributs d'agrégation.	1	1	1	1	1
ACT.04.13 – Vérifier la compatibilité des attributs d'agrégation.	1	1	1	1	1
ACT.04.14 – Journaliser les incompatibilités des attributs d'agrégation identifiées.	1	1	1	1	1
ACT.04.15 – Appliquer la règle RA-03 : Compatibilité des plateformes.	1	1	1	1	1
ACT.04.16 – Vérifier la compatibilité des services.	1	1	1	1	1
ACT.04.17 – Journaliser les incompatibilités des plateformes identifiées.	1	1	1	1	1

Identificateur des activités du processus SOCAP	Agrégat par collection « Portail Loisir En Ligne – VDM »	Agrégat par connexion « Télé-paiement »	Agrégat par coordination « Portail-outils »	Agrégat par fusion « Calen-drier »	Couverture par activité (1=Couverte, 0=Non Couverte)
ACT.04.18 – Consolider les incompatibilités détectées.	1	1	1	1	1
ACT.04.19 – Produire le rapport d'étude la faisabilité de l'agrégation.	1	1	1	1	1
ACT.05 – Analyser le rapport d'évaluation.	1	1	1	1	1
ACT.05.01 – Analyser la pertinence de l'incompatibilité détectée (valide/ non valide).	1	1	1	1	1
ACT.05.02 – Décider de la rétention de l'incompatibilité/compatibilité valide.	1	1	1	1	1
ACT.05.03 – Classifier l'incompatibilité retenue.	1	1	1	1	1
ACT.05.04 – Classifier la résolution de l'incompatibilité.	1	1	1	1	1
ACT.05.05 – Décrire la résolution de	1	1	1	1	1

Identificateur des activités du processus SOCAP	Agrégat par collection « Portail Loisir En Ligne – VDM »	Agrégat par connexion « Télé-paiement »	Agrégat par coordination « Portail-outils »	Agrégat par fusion « Calen-drier »	Couverture par activité (1=Couverte, 0=Non Couverte)
l'incompatibilité classifiée.					
ACT.05.06 – Décrire l'échec de la résolution.	0	0	0	0	0
ACT05.07 – Produire la version finale du rapport de faisabilité de l'agrégation.	1	1	1	1	1
ACT.06 – Concevoir l'agrégat.	1	1	1	1	1
ACT.06.01 – Classifier l'agrégation (collection/connexion/coordination/fusion).	1	1	1	1	1
ACT.06.02 – Évaluer les patrons de connexion.	1	1	0	1	1
ACT.06.03 – Appliquer le patron « connexion unidirectionnelle ».	0	0	0	0	0
ACT.06.04 – Appliquer le patron « connexion bi-directionnelle ».	0	1	0	0	1

Identificateur des activités du processus SOCAP	Agrégat par collection « Portail Loisir En Ligne – VDM »	Agrégat par connexion « Télé-paiement »	Agrégat par coordination « Portail-outils »	Agrégat par fusion « Calen-drier »	Couverture par activité (1=Couverte, 0=Non Couverte)
ACT.06.05 – Appliquer le patron « collectionneur ».	1	0	0	0	1
ACT.06.06 – Évaluer les patrons de coordination.	1	0	1	1	1
ACT.06.07 – Appliquer le patron « coordination répartie».	0	0	0	0	0
ACT.06.08 – Appliquer le patron « coordination fédérée».	0	0	1	0	1
ACT.06.09 – Évaluer les patrons de fusion.	1	1	1	1	1
ACT.06.10 – Appliquer le patron « fusion en partie».	0	0	0	1	1
ACT.06.11 – Appliquer le patron « fusion totale».	0	0	0	0	0
ACT.06.12 – Définir la composition dynamique l’agrégat.	1	1	1	1	1
ACT.07 – Adapter les	1	1	1	1	1

Identificateur des activités du processus SOCAP	Agrégat par collection « Portail Loisir En Ligne – VDM »	Agrégat par connexion « Télé-paiement »	Agrégat par coordination « Portail-outils »	Agrégat par fusion « Calen-drier »	Couverture par activité (1=Couverte, 0=Non Couverte)
composants participants.					
ACT.08 – Identifier les paramètres de configuration.	1	1	1	1	1
ACT.09 – Développer l'agrégat.	1	1	1	1	1
ACT.10 – Configurer les environnements de l'agrégat (système/sécurité/ réseau/ base de données).	1	1	1	1	1
ACT.11 – Autoriser le déploiement en test.	1	1	1	1	1
ACT.12 – Déployer l'agrégat en test.	1	1	1	1	1
ACT.13 – Tester l'agrégat.	1	1	1	1	1
ACT.14 – Produire un rapport des tests.	1	1	1	1	1
ACT.15 – Analyser le rapport des tests.	1	1	1	1	1
ACT.16 – Autoriser une	1	1	1	1	1

Identificateur des activités du processus SOCAP	**Agrégat par collection « Portail Loisir En Ligne – VDM »**	**Agrégat par connexion « Télé-paiement »**	**Agrégat par coordination « Portail-outils »**	**Agrégat par fusion « Calen-drier »**	**Couverture par activité (1=Couverte, 0=Non Couverte)**
mise en production de l'agrégat.					
ACT.17 – Déployer l'agrégat en production.	1	1	1	1	1
ACT.18 – Référencer le nouvel agrégat.	1	1	1	1	1
ACT.19 – Mettre à jour les fiches SOCOM des composants agrégés.	1	1	1	1	1
NOMBRE D'ACTIVITÉS EXÉCUTÉES	**63/77**	**62/77**	**68/77**	**69/77**	**72**
TAUX DE COUVERTURE PAR EXÉCUTION	**82%**	**81%**	**88%**	**90%**	
Indicateur de couverture des activités suite à l'exécution de SOCAP (ICAS)				**94%**	

Nos quatre exécutions parcourent chacune plus de 80% des activités du processus SOCAP pour leur catégorie d'agrégation respective. Ceci prouve que le choix des contextes d'agrégation est adéquat et que la majorité de nos activités sont utilisées durant l'exécution des quatre types d'agrégation. De plus, les contextes d'agrégation choisis ont abouti à un agrégat, ce qui nous permet de confirmer l'utilité de ces activités pour atteindre l'objectif principal de notre thèse, le développement d'un composant par agrégation dirigée par les métadonnées et les modèles.

Les cinq autres cas d'agrégation parmi les neuf réalisables ont produit aussi un agrégat et puisqu'ils sont classés dans l'une de nos catégories, leur indice de couverture ICAS avoisine les 80%. Concernant les quatre cas d'agrégation non réalisables, leur processus exécuté ne dépasse pas l'activité d'évaluation du rapport de la faisabilité d'agrégation (ACT.05) et parfois il s'arrête à l'activité d'évaluation de la faisabilité (ACT.04). L'indice de couverture des activités ICAS avoisine 60%. Ceci nous assure que la phase de faisabilité que nous avons proposée a épargné la réalisation de entre 20% et 40% des activités d'agrégation, ce qui se traduit par une réduction des coûts de développement des phases d'adaptation et d'intégration. À titre d'exemple, la VDM a épargné $50 000 en décidant de ne pas déployer Médiathèque et s'est contentée de réutiliser son gestionnaire de photo du portail Oracle. Ce montant comprend $15 000 pour l'achat du produit et $35 000 pour son adaptation et déploiement pour cinq sections de la division des TI.

En résumé, la mesure ICAS nous a permis de valider les aspects suivants du processus SOCAP, son **utilisabilité** et son **utilité**.

7.8.2 Mesure 2 : évaluation de Indice de Couverture des Objectifs suite à l'exécution de SOCAP, ICOS

Le processus SOCAP est proposé principalement pour satisfaire un ensemble d'objectifs autour du développement logiciel à base de composants dirigé par les métadonnées et les modèles et la promotion du développement avec la réutilisation. Chaque exécution d'un type d'agrégation doit atteindre la majorité des objectifs énoncés. Lorsque l'indice avoisine 100%, ceci démontre que les exécutions ont atteint la majorité des objectifs et que le processus SOCAP est une contribution, dans le volet des processus et des méthodologies, pour les domaines CBSD et du développement par la réutilisation.

Ce deuxième indicateur associé aux objectifs de SOCAP est appelé « **I**ndice de **C**ouverture des **O**bjectifs suite à l'exécution de **S**OCAP », ou ICOS. ICOS est défini comme suit :

ICOS = Moyenne (100 * ∑ (Indicateur de couverture d'un objectif par toutes les exécutions de SOCAP d'un type d'agrégation)/ 4)

ICOS est calculé en vérifiant si chaque exécution rencontre l'objectif ou non. Si l'objectif est atteint nous indiquons le chiffre 1, sinon nous indiquons 0. Le
Tableau 7.12 énumère l'ensemble des objectifs du processus SOCAP. Pour expliquer les valeurs du tableau, nous prenons l'exemple de l'objectif Obj-2 dans le contexte des agrégations par connexion. Deux des cinq cas ont utilisé la recherche pour trouver les composants potentiels, ainsi nous avons un indice de 0.4 qui est résultant du rapport 2/5. Nous appliquons le même calcul pour les exécutions du reste des types d'agrégation pour Obj-2, ensuite nous calculons la moyenne que nous multiplions par 100 et nous obtenons la valeur 47.50%. Ensuite nous appliquons une moyenne de tous les objectifs, nous obtenons la valeur de 74.89%.

Tableau 7.12 Évaluation de couverture des objectifs suite aux exécutions de SOCAP

Liste des sept objectifs de SOCAP	**Agrégat par connexion (5 cas)**	**Agrégat par collection (3 cas)**	**Agrégat par coordination (2 cas)**	**Agrégat par fusion (3 cas)**	**% moyen de Couverture par Objectif**
Obj-1 : Promouvoir le développement par la réutilisation.	100%	100%	100%	100%	100%
Obj-2 : Faciliter la recherche multicritère des composants logiciels dans les organisations.	40%	0%	50%	100%	47.50%
Obj-3 : Rendre visible la dépendance entre les composants agrégats et les composants agrégés à l'aide des spécifications SOCAM.	100%	100%	100%	100%	100%

Liste des sept objectifs de SOCAP	**Agrégat par connexion (5 cas)**	**Agrégat par collection (3 cas)**	**Agrégat par coordination (2 cas)**	**Agrégat par fusion (3 cas)**	**% moyen de Couverture par Objectif**
Obj-4 : Permettre l'évaluation de la faisabilité de l'agrégation avant d'entamer le développement de l'agrégat.	100%	100%	100%	100%	100%
Obj-5 : Détecter les agrégations non faisables tôt dans le processus de développement à base de composants;	60% : 3 cas non réalisables	33% : 1 cas non réalisable	0%	0%	23.25%
Obj-6 : Concevoir le composant agrégat en se basant sur une classification supportée par des patrons de composants.	40%	67%	100%	100%	76.75%
Obj-7 : Supporter de bout en bout l'agrégation des composants logiciels depuis la définition du contexte de l'agrégat jusqu'à sa mise en production.	40%	67%	100%	100%	76.75%
ICOS Global					74.89%

Les exécutions faites ont toutes aidé à atteindre la majorité des objectifs prévus du processus SOCAP. En effet, toutes les exécutions sont réussies et elles ont produit des résultats valides qui varient entre des agrégats non faisables avec une justification utile et des agrégats fonctionnels qui sont actuellement en production. Ainsi, SOCAP a permis de développer des

composants logiciels par la réutilisation d'autres composants et d'identifier des agrégations non faisables, ce qui correspond aux deux objectifs majeurs à l'origine de la proposition du processus (Obj-1, Obj-5).

L'impact des exécutions non réalisables (quatre cas) se manifeste dans la valeur de 23,25% de l'objectif (Obj-5). Cette valeur, bien qu'elle montre un faible pourcentage, représente un bon résultat. En effet, elle montre que nous avons pu détecter des agrégations non réalisables, ce qui réduit les coûts de développement inutile et met en valeur notre proposition d'ajouter une phase et un processus d'évaluation de la faisabilité au processus d'agrégation à base de composant. L'indice de cet objectif doit être interprété positivement lorsque sa valeur est différente de 0%. Ceci confirme davantage l'importance du volet sémantique que nous avons introduit par notre proposition des métadonnées SOCOM et les règles d'agrégations associées pour servir le développement à base de composant.

Nous expliquons la valeur de 47,50%, correspondant à l'indice de l'objectif de recherche (Obj-2), par le fait que la composition statique de la moitié des agrégations exécutées est connue d'avance. Par conséquent, nous n'avons pas besoin d'utiliser notre service de recherche multicritère pour trouver les composants potentiels participants.

7.8.3 Mesure 3 : évaluation de l'indicateur IAS

Le troisième indicateur, nommé « **I**ndicateur d'**A**chèvement de l'exécution de **S**OCAP - **IAS** », mesure l'aboutissement de l'exécution de notre processus. En effet, nous vérifions après chaque exécution que le processus aboutit à un composant agrégat fonctionnel, si l'agrégation est faisable, et a une explication dans le cas contraire. Nous attribuons la valeur 1 à l'indicateur en cas d'achèvement ou 0 dans le cas contraire.

IAS = (Nombre de fois où l'exécution du processus SOCAP s'achève / Nombre total des exécutions)* 100

Si l'indicateur avoisine le 100%, ceci confirme que le processus SOCAP se termine pour la majorité des cas étudiés quel que soit le résultat obtenu. Nous pouvons approfondir l'interprétation de cet indicateur en proposant deux indicateurs dérivés tels que: IAS-NR et IAS-R qui représentent respectivement les cas d'achèvement de SOCAP sans agrégat (**N**on **R**éalisable, non faisabilité ou abandon de l'agrégation) et ceux d'achèvement avec agrégat (**R**éalisable : production d'un nouveau composant agrégat). Nous déduisons l'équation suivante :

$$\mathbf{IAS} = \text{IAS-R} + \text{IAS-NR}$$

En effet, si l'indicateur IAS-R s'approche de la valeur d'IAS, nous pouvons conclure que la majorité des exécutions aboutissent à un composant agrégat et nous avons atteint l'objectif principal de la proposition du processus SOCAP, soit le développement de nouveau composant par agrégation. Cependant, dans le cas où IAS-NR est différent de zéro, nous pouvons conclure que notre processus a atteint aussi un autre objectif (Obj-5) qui est la détection des cas d'agrégations non faisables.

Après treize exécutions, nous avons évalué l'indicateur IAS. Le tableau ci-dessous montre l'état d'achèvement des exécutions complétées avec succès et avec échec.

Tableau 7.13 Évaluation de l'indicateur IAS suite aux exécutions de SOCAP

Nom des agrégations / État d'achèvement des exécutions	**Exécution complétée avec agrégat réalisé (IAS-R)**	**Exécution complétée avec agrégat non réalisable (IAS-NR)**
Agrégations par connexion	2	3
« Gestion de contenu »	X	-
« Télépaiement »	X	-
« Portail-VDM – Médiathèque »	-	X
« GénérateurForumlaire-VDM – Calendrier»	-	X
« GFEC-Médiathèque»	-	X
Agrégations par collection :	2	1

Nom des agrégations / État d'achèvement des exécutions	Exécution complétée avec agrégat réalisé (IAS-R)	Exécution complétée avec agrégat non réalisable (IAS-NR)
"Client Web-Explor@2"	X	-
« TivoliWebTop – SecurNet – Documentum »	-	X
« Portail Loisir En Ligne – VDM »	X	-
Agrégations par coordination:	2	0
"FSE Federated Search Engine"	X	-
« Portail-outils »	X	-
Agrégations par fusion	3	0
« Calendrier »	X	-
« SRD-Search Result Display »	X	-
« GFEC-GPRO»	X	-
Totaux	**9 (69%)**	**4 (31%)**
Valeur de l'IAS	**100%**	

Dans les treize cas étudiés, la valeur de 100% de l'indicateur IAS montre que notre processus s'achève toujours avec un état de l'agrégation (réalisable ou non réalisable). La valeur de 100% de l'indicateur IAS valide la capacité de notre processus SOCAP à traiter les cas d'agrégation choisis et à fournir un état de l'agrégation dans tous les cas. En se basant sur les résultats des cas d'exécution choisis, nous pouvons dire que SOCAP est bien construit. Nous ne pouvons pas nous prononcer dans l'absolu pour tous les contextes d'agrégation, mais nous jugeons que les exécutions futures peuvent confirmer le volet d'achèvement de notre processus.

Nous pouvons interpréter les valeurs de l'indicateur IAS en relation avec les objectifs de l'indicateur ICOS. En effet, le Tableau 7.13 montre que neuf des treize exécutions ont abouti à des composants agrégats logiques ou physiques fonctionnels. L'exécution avec succès de 69% des cas valide l'objectif global de nos travaux qui est la proposition d'un processus fonctionnel de développement des composants par agrégation dirigée par les métadonnées et les modèles et qui encourage le développement par la réutilisation (section 4.1). De plus, le processus d'agrégation a détecté 31% de cas d'agrégations non faisables parmi tous les cas

traités au moment de l'exécution de l'activité d'évaluation de la faisabilité de l'agrégation (SOCAP.ACT.04). Ainsi, nous validons notre objectif de détection de la faisabilité ou non d'agrégation tôt dans le processus de développement à base de composants.

7.9 Les défis et les limites des exécutions

Au début de ce chapitre, nous avons présenté les exécutions qui ont abouti à des agrégats logiques ou physiques et celles qui ont été annulées pour des raisons techniques, stratégiques ou budgétaires (section 7.7). Toutefois, le processus a passé au moment de son exécution par des difficultés et des défis qui ont demandé des révisions et des efforts de résolution durant une longue période, dont voici les points les plus importants:

- Au début du processus, la préparation des métadonnées, le maintien à jour de l'entrepôt SOCOM et son évolution pour les compagnies Telus et la VDM n'étaient pas évidents vu l'absence de l'engagement de la direction.
- Les métadonnées d'affaires et les métadonnées non fonctionnelles ne sont pas faciles à trouver pour la VDM vu que la documentation des composants existants manque et les personnes responsables ont quitté la compagnie. Par contre, Telus est certifié CMMi de niveau 3, la documentation était disponible et nous y avons eu accès durant le développement de l'agrégat « Portail-outils ».
- Notre suivi des agrégations, incluant le développement des composants agrégats, a duré une année et demie pour la VDM[35] et huit mois dans le contexte de Telus. En effet, nous avons participé activement dans les agrégations par connexion, par collection, par coordination et par fusion présentées précédemment en tant qu'analyste d'affaires et architecte applicatif. Les agrégats résultants sont en production actuellement dans les environnements des compagnies en question. Nous avons pu exécuter les cinq premières activités SOCAP avant le développement des agrégats, mais chacune des compagnies

[35] Délais des développements des agrégats jusqu'à à mise en production : 4 mois pour télépaiement et Portail LEL, 4 mois pour Portail VDM – LEL, 8 mois pour Calendrier.

dispose de son propre processus de développement. Par conséquent, les activités des phases d'adaptation et d'intégration sont parfois différentes de celles de SOCAP.

- Il n'est pas évident de reconnaitre une fausse incompatibilité et de retirer une incompatibilité identifiée avec des valeurs d'attributs différents. Parfois, nous avons eu recours à des architectes principaux dans les contextes d'exécution pour nous aider dans l'identification, l'analyse et l'activité de résolution des incompatibilités détectées initialement.
- La description des activités de référencement, d'évaluation de la faisabilité et de résolution ne pouvaient pas être considérées dans la planification des projets de développements prévus de CGI chez la clientèle. Nous avons réalisé ces activités en parallèle avec le développement et nous avons fourni les résultats de nos travaux aux équipes en place pour les aider dans les prises de décision quant à la faisabilité des agrégations. Ceci étant, il était plus facile d'ajouter des efforts pour l'évaluation de la faisabilité dans l'agrégation chez Telus qu'à la VDM. Ceci s'explique par les niveaux de structuration et de standardisation des processus de développement et de documentation chez Telus par rapport à la VDM.
- Nous avons constaté que la culture de l'équipe de Telus – comme centre de développement certifié – et leur confiance dans les études empiriques et les processus d'évaluation avant le développement effectif, nous ont aidé à partager les résultats d'exécution des activités de référencement et d'évaluation de la faisabilité de SOCAP.
- Après les exécutions du processus dans le contexte de Telus, nous avons eu à ajuster les activités de référencement et d'évaluation de la faisabilité des agrégations. Par conséquent, nous avons révisé les exécutions faites à la VDM pour ajuster la structure des tableaux des résultats et l'ordre des activités d'évaluation de la faisabilité. Ceci a nécessité des efforts supplémentaires pour une révision de tous les livrables des anciennes exécutions.
- Nous avons appris que notre processus est exécuté différemment dans les compagnies choisies selon la maturité de leur processus de développement et de documentation. En effet, l'exécution des activités associées aux phases de recherche, de sélection et de faisabilité s'exécutent de façon plus directe et plus simple dans des compagnies ayant un

niveau de maturité dans les processus de développement et de documentation. Autrement dit, nous avons trouvé plus simple d'exécuter SOCAP à CGI et à Telus qu'a la VDM.

7.10 Résultat et conclusion

Suite aux exécutions, nous avons confirmé que les métadonnées et les modèles peuvent diriger avec succès un processus d'agrégation de composants logiciels. Ceci se manifeste essentiellement de l'activité de référencement (ACT.01) jusqu'à l'activité de conception (ACT.06). Durant l'exécution de ces activités de SOCAP, les métadonnées SOCOM, SOCAM et les règles d'agrégation supportées par les outils du cadre de travail SOCAF, ont facilité la recherche et la sélection adéquate des composants participants par rapport à des méthodes sans les métadonnées multidimensionnelles. Aussi, notre phase d'étude de la faisabilité détaillée des agrégations de SOCAP se distingue par la détection des agrégations non réalisables en comparaison avec les méthodes de développement à base de composants qui ne comportent pas cette phase, tels que CUP, Catalysis, UML Component (voir les détails des approches dans la section 1.7).

En plus de la validation de notre processus, les exécutions ont permis d'ajuster et de raffiner à plusieurs reprises les métadonnées SOCOM, SOCAM, le flux de contrôle, le flux de données de notre processus SOCAP et notre outil Excel pour le suivi des activités d'évaluation de la faisabilité (ACT.04) et d'analyse du rapport d'évaluation (ACT.05). Ces exécutions ont participé au développement et à la validation de la méthode d'agrégation avec tous les éléments connexes de façon itérative et incrémentale.

Quant aux mesures proposées, l'indicateur d'achèvement IAS est évalué à 100%, l'indicateur ICAS de couverture des activités est évalué à 94% alors que l'indicateur ICOS de couverture des objectifs est proche de 75%. Ces valeurs confirment que nos exécutions de SOCAP, dans différents contextes d'agrégations, s'achèvent et qu'elles ont permis de couvrir la quasi-totalité des activités du processus proposé tout en répondant à la majorité des objectifs définis. Ces indicateurs nous confirment que notre processus SOCAP est bien construit,

utilisable et utile. Par conséquent, nous jugeons que notre processus SOCAP est valide et peut être utilisé, testé et évalué, de nouveau, dans d'autres contextes d'agrégation.

DISCUSSION DES RÉSULTATS

Pour positionner notre méthode dans la littérature, nous avons choisi l'approche d'évaluation des méthodes de développement à base des composants de Boertien *et al.* Elle est basée sur des critères généraux tels que le mode de fonctionnement, la manière de modéliser et le niveau de support proposé à l'aide des processus et des outils dédiés (Boertien, 2005). Ces critères d'évaluation se résument dans les éléments suivants :

- le contexte de son développement : académique et/ou industriel;
- la fréquence de son utilisation : peu, moyenne ou fréquente;
- le support au processus de la méthode;
- le support de la méthode à la réutilisation;
- la modélisation des composants (les outils, les formalismes et les techniques de modélisation);
- les outils de support à l'utilisation de la méthode de développement à base de composant;
- l'existence d'une plateforme d'implémentation.

En évaluant notre méthode et son processus SOCAP selon les critères énumérés ci-dessus, nous constatons que nos travaux ont été réalisés initialement dans un contexte académique pour ensuite être utilisés dans un contexte industriel. Nous évaluons ce critère de notre méthode à « Académique/industriel ». Cette évaluation positionne notre méthode entre les processus industrialisés tels que RUP/Select Perspective et ceux strictement académique tel que Catalysis, UML Component, COSE, etc (voir la section 1.7).

Notre méthode a été peu utilisée au LICEF dans le cadre du projet LORNET, mais nous l'avons appliquée dans 13 cas concrets dans deux compagnies différentes. Nous attribuons la valeur « Peu utilisée » au critère de l'utilisation de la méthode, comparativement aux 500 grandes compagnies[36] utilisées pour tester Catalysis.

---

[36] Catalysis (2000). http://www.catalysis.org.

Notre méthode se base sur quatre phases de développement à base de composants auxquelles nous avons ajouté notre phase de faisabilité. Les processus industriels comme RUP et Select Perspective définissent leur processus de support à l'aide des phases, des activités et des flux de données et de contrôle. Le support au processus des méthodes académiques est souvent défini à l'aide de guides, de directives et de jalons. Le support au processus de notre méthode ressemble plus à des processus industriels.

Le processus SOCAP est modélisé par le formalisme BPMN, il décrit en détail ses activités et spécifie le flux des données et le flux de contrôle dans un ordre strict des activités requises en relation avec les intervenants impliqués. Avec cet ordre strict, SOCAP peut convenir à des adaptations ou des changements pour son application dans des contextes différents de sa création d'origine. De plus, il est utile et pratique pour les utilisateurs qui préfèrent une utilisation directe et systématique du processus d'une approche de développement de composants par agrégation.

Quant au volet de la réutilisation, notre méthode a défini un ensemble de spécifications des métadonnées des composants et de leur agrégation, des règles, des classifications, des modèles, des patrons d'agrégation, des processus et des artéfacts réutilisables. En effet, les premières activités de SOCAP utilisent les spécifications SOCOM pour représenter les composants, ce qui favorise leur réutilisation et leur visibilité selon différents points de vue. Notre bibliothèque des composants basée sur SOCOM représente un autre facteur de la réutilisation. Aussi notre méthode offre une classification des agrégats en se basant sur des équations ensemblistes. De plus, une fois la classe d'agrégation choisie, l'architecte peut réutiliser les patrons d'agrégation associés à chacune des classes d'agrégation. Durant le processus de faisabilité, la méthode encourage la réutilisation de la classification des incompatibilités entre les composants logiciels participants et propose des résolutions potentielles. Ceci permet de créer une base de connaissance de façon incrémentale quant à l'identification, la spécification et la résolution des problèmes architecturaux inter-composants. Tous ces éléments de notre méthode représentent des éléments réutilisables.

Les méthodes existantes telles que Catalysis, UML Component, RUP et CORSE ont défini des éléments connexes à leur processus basés sur les modèles UML des composants avec des spécialisations et des méta-modèles qui les distinguent les unes des autres afin de promouvoir la réutilisation des composants au niveau modèle, mais nous nous distinguons principalement par nos métadonnées SOCOM, SOCAM, nos modèles de classification des agrégations, nos règles d'agrégation et notre processus d'agrégation et celui d'évaluation de la faisabilité.

Quant à la modélisation, notre méthode utilise le formalisme UML pour la modélisation des composants et des agrégats. Aussi, nos patrons d'agrégation sont modélisés en UML, ce qui facilite leur compréhension et favorise leur partage et leur application si les conditions de leur réutilisation sont vérifiées. Ceci dit, notre méthode et son processus SOCAP n'utilisent le formalisme UML que pour des besoins de représentation et de documentation contrairement à d'autres processus qui utilisent UML dans le processus d'agrégation et les transformations des modèles selon une approche MDA. Nous avons utilisé la notation BPMN pour la modélisation des processus SOCAP et celui de l'évaluation de la faisabilité d'agrégation pour promouvoir sa systématisation et son automatisation futures à l'aide des outils BPMS appropriés.

Concernant les outils de support aux méthodes, notre méthode propose principalement un cadre de travail abstrait SOCAF qui supporte différentes spécialisations dans des contextes d'implémentation différentes. SOCAF définit des entités conceptuelles qui serviront de base pour le développement de composants par agrégation. Ces entités assistent l'architecte pour construire un environnement de développement à base de composants. SOCAF définit des artéfacts, des outils pour la spécification des composants logiciels (grilles Excel, Gestionnaire des composants SOCOM), la spécification de leur agrégation et des modèles de classification de ces composants et de leur agrégat (les outils UML). Elle propose plusieurs grilles Excel pour supporter un processus inédit concernant l'évaluation de la faisabilité d'agrégation. En plus, notre cadre de travail SOCAF supporte aussi bien l'utilisation du

processus SOCAP ainsi que la réutilisation des actifs du processus autour du développement itératif et incrémental à base de composants.

La majorité des méthodes existantes utilisent les outils UML et n'exigent pas des plateformes propriétaires sauf RUP, Select Perspective et CUP. RUP et Select Perspective sont deux approches industrielles, ce qui explique l'existence des outils de support commerciaux et avancés. CUP est un processus supporté par une plateforme JEE académique qui assiste les développeurs dans la transformation des modèles et la génération du code JAVA du composant agrégat selon une approche MDA. Notre méthode n'a pas une plateforme d'implémentation complète et propriétaire sauf le Gestionnaire des composants SOCOM, les convertisseurs des formats des métadonnées et les grilles Excel d'évaluation de la faisabilité. SOCAF est un cadre de travail générique qui supporte notre processus SOCAP. Il peut être instancié différemment pour servir d'autres contextes d'agrégation avec d'autres outils qui assument les responsabilités prédéfinies pour chacune des entités conceptuelles proposées.

Les méthodes existantes CUP, Catalysis et Select Perspective définissent leur propre modèle UML de composant, qui se transforme depuis le début du processus d'agrégation jusqu'à la génération du code. Notre méthode propose un modèle de composants et des métadonnées indépendants de la plateforme, mais nous n'avons pas réalisé des transformations de nos modèles dans des plateformes spécifiques. Notre modèle UML et les métadonnées SOCOM d'un composant sont utilisés au début pour aider au référencement, à la recherche et à la sélection des composants participants puis à l'évaluation de la faisabilité de l'agrégation. Nous avons des modèles et des patrons de classification des composants pour la phase de conception de l'agrégat. Ensuite, pour le reste du développement, nous proposons la réutilisation des pratiques, des techniques et des outils existants pour réaliser les activités des phases d'adaptation et d'intégration.

En résumé, nos contributions concernent essentiellement trois des cinq phases des processus de développement à base de composants : la recherche, la sélection et la faisabilité. Pour les autres, nous utilisons les techniques EAI et les techniques et les technologies énoncées dans

la section 1.9. Le tableau suivant (ref tableau) rappelle les éléments de comparaison entre quelques processus types et SOCAP.

Tableau-A D-1 Comparaison entre les processus de développement à base de composant et SOCAP

Processus/ Critères d'évaluation du cadre de Boertien	**Catalysis**	**CUP**	**RUP**	**Select Perspective**	**SOCAP**
1- Contexte de développement	Académique	Académique	Industriel	Industriel	Académique/ industriel
2- Fréquence de son utilisation	500 grandes compagnies	Dans le laboratoire	Adopté par la majorité	Utilisation régulière	3 compagnies
3- Support au processus de la méthode	Directives d'util.	Directives d'util.	Directives d'util., artéfact, outils	Phases, activités, flux de contrôle et de données	Phases, activités, flux de contrôle et de données
4- Unité de réutilisation	Patrons; composants; Framework	Patrons; composants; Framework	Composants logiciels	Patrons et des composants	Patrons; classification (2); composants; Framework
5- Propose des patrons	Oui	Oui	Non	Oui	Oui
6- Bibliothèque de composants	Non	Non	Non	Oui, « Select Cpt Manager »	Oui, « SOCOM Manager »

Processus/ Critères d'évaluation du cadre de Boertien	**Catalysis**	**CUP**	**RUP**	**Select Perspective**	**SOCAP**
7- techniques de modélisation utilisées	Plusieurs techniques et diagrammes UML	Diag. UML avec nouveau stéréotype	UML et autres	BPM, UML, Schéma relationnel, Diagramme ER	BPMN, UML, OWL, Schéma relationnel,
8- Outils de support à l'utilisation	Pas d'outils dédiés. Les outils compatibles UML	Outils de développement de composants JEE/CORBA	IBM Rational Suite	Select Enterprise; Select Component Manager; UML tools; BPM tools; OO tools; Data modeling tools.	SOCOM Manager; Extracteurs des métadonnées services. Outils D'éval.; Ontologie SOCOM, OWL.
9- Plateforme d'implémentation	Indépendant de la plateforme	Plateforme JEE	Indépendant de la plateforme	Indépendant de la plateforme	Indépendant de la plateforme
10 –Évaluation de la faisabilité d'agrégation	Non	Non	Guidelines	Guidelines	Oui, supporté par un processus et un outil.

Notre méthode est conçue pour des équipes de projets qui veulent maximiser la réutilisation en développant des composants par agrégation. Notre approche peut être utilisée entièrement ou partiellement. En effet, nous pensons qu'une partie de nos réalisations peut complémenter les méthodes existantes en réutilisant nos métadonnées SOCOM, notre bibliothèque de composants et notre processus de faisabilité avant d'entamer leur phase de conception et de transformation de modèles selon l'approche MDA. Toutes les approches proposent des modèles UML des composants participants. Durant la phase d'analyse, nous pouvons générer des documents XML à partir de leur modèle UML à l'aide des fonctionnalités des outils UML utilisés. Ces documents XML contiennent les métadonnées techniques et des services des composants participants. Nous pouvons développer un convertisseur XML-XML pour traduire les documents XML natifs générés à des documents XML conformes aux spécifications SOCOM en respectant notre schéma. Ensuite, nous pourrions utiliser la fonctionnalité d'importation pour ajouter les métadonnées de ces composants à notre Gestionnaire SOCOM « SOCOM Manager » ou à notre ontologie suite à une conversion XML-OWL. Ces métadonnées seraient enrichies manuellement pour pouvoir appliquer notre processus d'évaluation de la faisabilité et décider ainsi de la faisabilité de l'agrégation avant d'entamer les transformations vers les modèles spécifiques à la plateforme et au modèle du code. Si la transformation à partir des modèles UML des composants s'avère compliquée, nous pourrions utiliser le code source ou compilé des composants pour extraire nos métadonnées SOCOM à l'aide de nos extracteurs. Nous pourrions aussi utiliser du référencement entièrement manuel pour remplir les fiches des métadonnées des composants.

En se basant sur nos contributions et en comparant notre méthode aux autres, nous avons identifié ses limites et ses forces.

Les limites du processus SOCAP:

- La recherche des composants logiciels du processus SOCAP ne propose pas d'interaction avec l'utilisateur (architecte) pour raffiner ou pour ajuster les résultats de la recherche.

- La sélection des composants logiciels est manuelle et se base sur le choix de l'architecte. SOCAP supporte la phase de sélection par le processus manuel d'évaluation de la faisabilité d'agrégation des composants logiciels.
- SOCAP n'offre pas de nouvelles techniques ni d'automatisme pour la phase d'adaptation des composants logiciels.
- SOCAP n'offre pas un cadre de travail, de technique et d'outils pour l'automatisation de l'intégration des composants logiciels.
- SOCAP ne couvre pas les systèmes temps réel et les systèmes embarqués qui requièrent des évaluations de faisabilité entre les services avec plus de détail en utilisant des formalismes et des règles plus strictes et plus élaborées.

Les règles du processus d'évaluation de la faisabilité de l'agrégation sont de type qualitatif et non quantitatif. Aucune unité de mesure n'est appliquée actuellement. La faisabilité de l'agrégation des attributs de qualité est évaluée par un expert et la décision découle d'un raisonnement basé sur ses connaissances.

Notre méthode est plus recommandée dans les cas suivants :

- Quand une organisation veut référencer ses composants afin de démocratiser et vulgariser leur utilisation, les métadonnées SOCOM sont multidisciplinaires et offrent différents points de vue pour partager l'information entre différents types d'acteurs. Le concept des métadonnées SOCOM est moins formel qu'UML pour représenter les composants et nous supportons plusieurs implémentations (relationnelle, XML et OWL) pour la création d'entrepôts de composants logiciels;
- Quand on veut minimiser les risques de développement à base de composants, notre approche peut référencer ses composants à l'aide des spécifications SOCOM et exécuter le processus d'évaluation de la faisabilité d'agrégation pour ainsi décider de continuer ou non le développement du composant agrégat. Nous n'avons pas trouvé un processus similaire dans la littérature;
- Quand une organisation préfère plus de structuration pour son processus de développement à base de composants et d'évaluation de la faisabilité des agrégations, elle

peut utiliser directement le processus SOCAP, le Gestionnaire SOCOM et notre cadre conceptuel SOCAF;

- Si une organisation est spécialisée dans le conseil et qu'elle participe dans les offres de services du marché pour vendre des solutions logicielles conceptuelles basées sur les agrégations des composants, elle peut évaluer à l'aide de SOCAP la faisabilité des solutions théoriques proposées et s'assurer ainsi de proposer des solutions réalisables;
- Si le processus d'agrégation est réalisé par une grande équipe multidisciplinaire, le processus SOCAP peut être déployé dans un outil BPMS et ses activités se transforment en tâches affectées et envoyées au tableau de bord de chaque intervenant. Ceci résout les aspects pratiques des problèmes de gouvernance, favorise la collaboration, assure le suivi de l'avancement en temps réel, garantit la traçabilité et facilite le volet d'affectation des tâches dans la gestion des projets de développement à base de composants;
- Les différentes représentations des métadonnées SOCOM (XML, OWL et relationnelle) facilitent le partage et l'échange des composants entre différents fournisseurs, ce qui encourage à la réutilisation à grande échelle inter et intra entreprises.

CONCLUSION

Après le développement de nos travaux de recherche et de validation par l'exécution des résultats obtenus, nous concluons dans la dernière étape de la démarche de notre méthodologie de recherche. La figure suivante positionne le présent chapitre dans le cheminement global de la thèse.

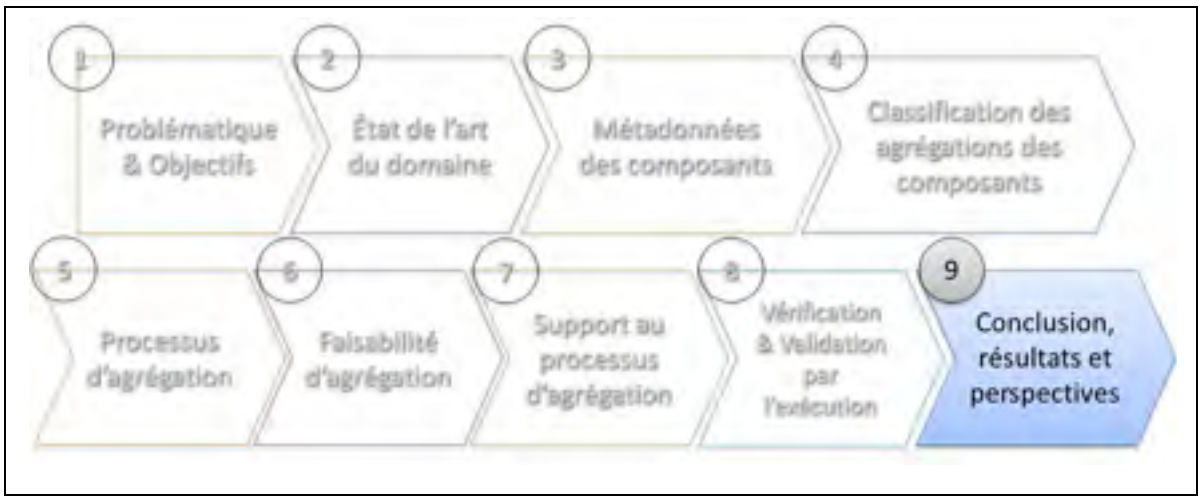


Figure-A C-1 La méthodologie de recherche de la thèse - Étape 9 :
Conclusion, résultats et perspectives

Nos travaux de recherche se situent à l'intersection du développement du logiciel à base de composants, du développement dirigé par les modèles et du développement du logiciel par la réutilisation.

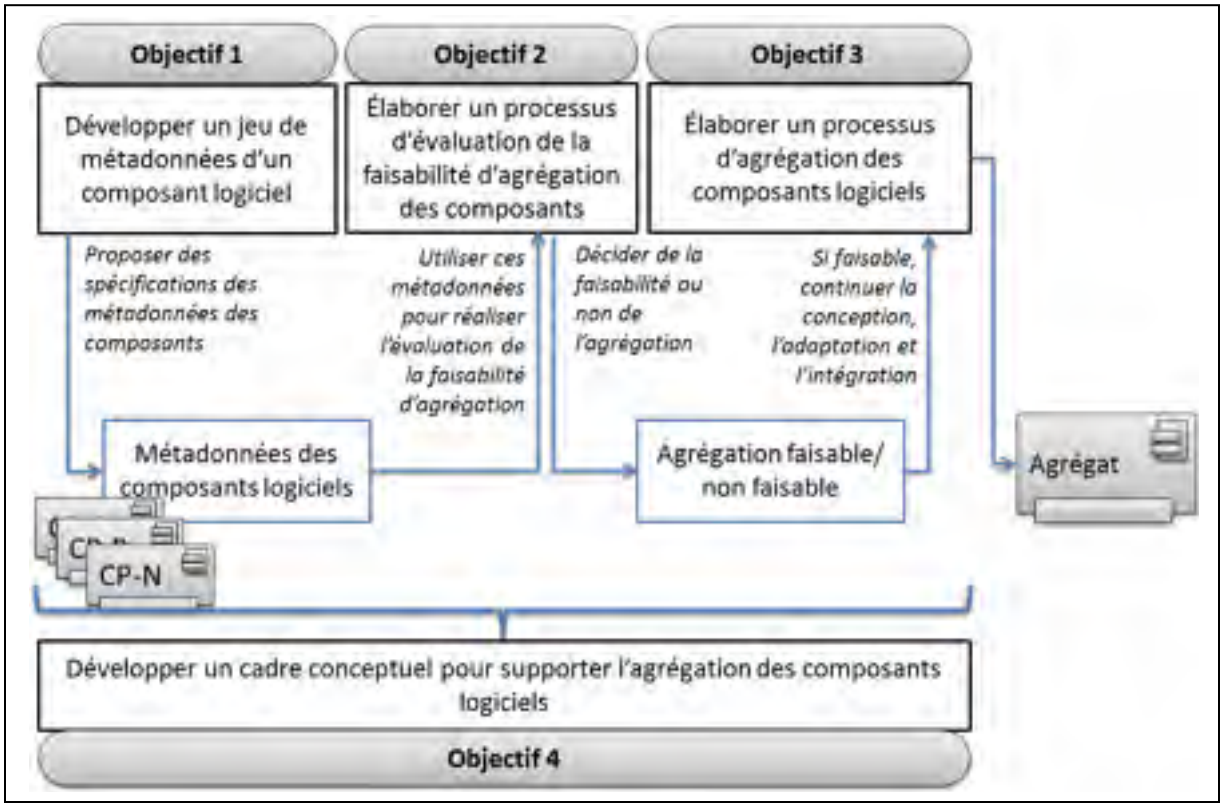


Figure-A C-2 Les relations entre les objectifs des travaux de recherche

La Figure-A C-2 rappelle les quatre objectifs inter-reliés de la thèse. Le premier concerne le développement d'un jeu de métadonnées et des modèles pour la caractérisation des composants logiciels et leur agrégation. Le deuxième consistait à élaborer un processus d'évaluation de la faisabilité de l'agrégation en utilisant les métadonnées proposées. À l'aide de ce processus, nous pouvons décider de continuer ou non le développement effectif du composant agrégat en utilisant un processus de développement logiciel à base de composants dirigé par les métadonnées et les modèles, ce qui représente notre troisième objectif. Le quatrième visait la définition d'un cadre conceptuel pour supporter l'utilisation des métadonnées et des processus connexes au moyen des outils de support aux intervenants.

Nos travaux de recherche ont produit des solutions conceptuelles (processus, métadonnées, cadre de travail, règle d'agrégation, etc.) pour servir le développement du logiciel à base de composants. Nous avons validé ces solutions par des révisions par des pairs moyennant une

dizaine de publications scientifiques dans des ateliers, des conférences et des revues. Nous avons aussi participé dans l'élaboration d'un chapitre d'un livre sur le web sémantique (Paquette, G & Masmoudi, A, 2010). En effet, nous avons publié nos propositions des représentations relationnelles et ontologiques de nos métadonnées SOCOM/SOCAM et des classifications des agrégations dans les conférences LORNET et INFORSID (Masmoudi *et al.*, 2003, 2004, 2005, 2006). La représentation ontologique de SOCOM a été utilisée dans le référencement technique des composants logiciels du système TELOS du projet LORNET (Masmoudi *et al.*, 2006). Nous avons aussi publié la première version de notre processus d'agrégation SOCAP et du cadre conceptuel SOCAF dans la revue IJAMC (Masmoudi *et al.*, 2008). De plus, dans le cadre de nos mandats professionnels comme conseiller à CGI, nous avons exécuté le processus SOCAP plusieurs fois pour valider son utilisation et son utilité dans des contextes réels (CHAPITRE 7).

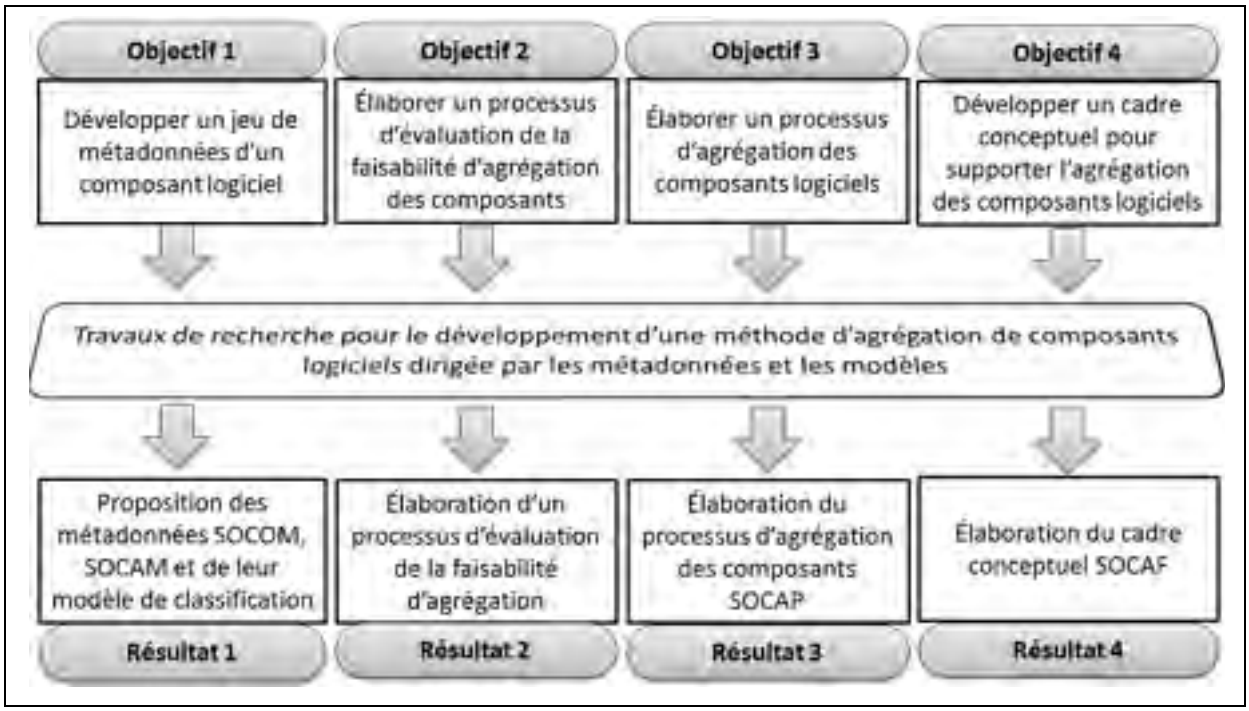


Figure-A C-3 Mise en relation entre les objectifs et les résultats des travaux de recherche

Nous avons produit plusieurs résultats de recherche durant nos travaux de recherche. La Figure-A C-3 montre les résultats qui résument nos réalisations en relation avec les objectifs visés :

- **Caractérisation d'un composant logiciel par la définition de ses différentes catégories de métadonnées.** Ces différentes dimensions de métadonnées décrites par SOCOM/SOCAM mettent en évidence plusieurs aspects décisifs concernant la réutilisation potentielle des composants. Ces métadonnées adjointes aux composants sont utilisées par différents parties prenantes pour mener des activités telles que le référencement, la classification, la recherche, la sélection, la réutilisation des composants logiciels ainsi que l'étude de faisabilité de leur agrégation. La mise en équation de quelques attributs d'agrégation des métadonnées SOCOM a permis d'identifier quatre catégories d'agrégation, soient la connexion, la collection, la coordination et la fusion. À ces catégories sont associées des patrons qui sont utilisés durant l'étape de la conception et du développement des agrégats.

- **Élaboration d'un processus d'étude de faisabilité d'agrégation des composants logiciels pour assister l'équipe de projet quant à la faisabilité de l'agrégation au début du processus de développement.** Ce processus propose des règles qui comparent entre elles les différentes catégories de métadonnées SOCOM des composants participants. Nous qualifions cette phase du processus de cruciale dans les approches de développement du logiciel à base de composants et de développement par la réutilisation. Elle aide à la prise de décision, quant à la faisabilité d'agrégation des composants sélectionnés, avant de se lancer dans un processus complexe de développement effectif de l'agrégat.

- **Proposition d'un processus d'agrégation des composants logiciels qui assiste une équipe de projet durant le développement d'un nouveau composant par l'agrégation d'autres.** Ce processus représente un flux de travail concret et pragmatique qui traduit les activités théoriques et pratiques requises de la part des différents intervenants pour développer des composants par agrégation dans un

contexte d'entreprise. Ce processus appuie notre méthode d'agrégation des composants logiciels dirigée par les métadonnées SOCOM/SOCAM et les modèles de classification des agrégations. Ce processus supporte la personnalisation, ce qui facilite sa mise en œuvre dans un cas réel d'entreprises qui disposent d'un actif important de composants logiciels. Bien que d'autres exécutions de ce processus soient utiles, nous jugeons que nos exécutions, qui couvrent les quatre types d'agrégations proposés dans différents contextes d'affaires, prouvent l'utilisabilité et l'utilité de notre proposition.

- **Définition d'un cadre conceptuel pour étudier, analyser et référencer les composants logiciels et leur agrégation**. Ce cadre se compose d'entités conceptuelles telles que les techniques, les outils, les processus et les règles d'agrégation. Une fois instanciées, ces entités permettent à une équipe de projet de capitaliser sur ses actifs logiciels de type composant et de développer de nouveaux composants par agrégation. L'instanciation du cadre se fait par le choix d'un outil de développement, d'un outil d'agrégation, le référencement d'un ensemble de composants avec les spécifications SOCOM et le tout doit être pensé dans un contexte d'agrégation bien défini. Ce cadre instancié construit un environnement de support à la réutilisation et au développement des composants logiciels par agrégation.

Ces résultats se basent principalement sur le concept des métadonnées et les modèles. Nous avons validé notre hypothèse de départ quant à l'utilité et l'apport des différentes catégories des métadonnées et des modèles adjointes aux composants logiciels au développement du logiciel à base de composants. Ces métadonnées ont permis d'assister une équipe durant le développement par agrégation de composants. Elles ont facilité leur référencement, leur classification, leur recherche et par conséquent leur réutilisation. Elles ont permis avec l'aide des règles d'agrégation, l'évaluation de la faisabilité de l'agrégation pour ainsi décider du développement effectif de l'agrégat. Un sous ensemble de ses métadonnées a permis la classification de l'agrégation et la sélection du patron approprié durant la phase de

conception. Elles ont été même réutilisées durant les phases d'adaptation et d'intégration (métadonnées des services, non fonctionnelles et des plateformes).

Notre méthode et les outils de support peuvent être utilisés en tant que processus indépendant pour le développement de bout en bout d'agrégats de composants. Pour ceux qui utilisent déjà une méthode de développement à base de composants, les éléments de notre méthode peuvent être utilisés partiellement et de façon complémentaire en réutilisant les spécifications SOCOM par la mise en place d'un entrepôt de composants et en déployant notre processus d'évaluation de la faisabilité d'agrégation.

RECOMMANDATIONS

La méthode d'agrégation dirigée par les métadonnées et les modèles utilise les métadonnées SOCOM, les métadonnées de l'agrégat SOCAM, les modèles d'agrégation et leurs patrons respectifs, le processus de faisabilité d'agrégation, le processus d'agrégation SOCAP et un cadre de travail SOCAF pour supporter le développement de composant par agrégation.

Tous les produits de la méthode présentent des dimensions qui méritent des études approfondies. Nous considérons les axes suivants de travail comme les plus importants :

- Le processus SOCAP est modélisé par le formalisme BPMN. Il existe plusieurs outils nommés BPMS[37] qui permettent de rendre les modèles BPMN exécutables. Il serait intéressent de rendre exécutable notre modèle BPMN du processus SOCAP (Figure 4.4). Ceci permettrait d'automatiser plusieurs activités du processus et faciliterait son utilisation dans les grandes entreprises. De cette façon, chaque utilisateur disposerait d'un tableau de bord qui lui dicte les activités en attente d'exécution. L'administrateur disposerait d'une vue globale sur toutes les instances du processus d'agrégation SOCAP et peut suivre leur exécution ainsi que les statistiques associées avec des indicateurs de performance prédéfinis et d'autres personnalisées.
- Du point de vue de la gestion, les outils BPMS permettent de paramétrer les activités avec des propriétés d'effort et de coût. Ceci permet de suivre l'avancement des travaux d'agrégation et les coûts associés en temps réel, ce qui renforcerait l'évaluation et l'analyse du volet économique. Mili *&* *al.* insistent que les aspects économiques des intégrations qui ne sont pas assez développés dans les domaines de développement à base de composant et de développement par la réutilisation. Dans l'implantation de SOCAP, une paramétrisation des efforts et des coûts peut contribuer à la définition d'un modèle économique des agrégations des composants.
- Du point de vue financier, nous pensons que les métadonnées SOCOM peuvent être enrichies avec des métadonnées financières en relation avec les métadonnées

[37] BPMS: Business Process Management System.

d'affaires telles que : effort de l'activité, coût maximal, coût minimal, pourcentage de contingence, etc. Ainsi, l'évaluation de faisabilité se verrait enrichie par une estimation des activités de SOCAP, ce qui fournirait aux décideurs une idée des coûts potentiels pour le développement de l'agrégat.

- Le processus de faisabilité réalise une comparaison directe entre les valeurs des attributs SOCOM des composants. Les exigences non fonctionnelles et les règles d'agrégation sont évaluées manuellement. Nous pensons que si ces exigences étaient représentées par des langages formels, nous pourrions automatiser en partie l'évaluation des règles d'agrégations et la conformité des composants agrégés aux exigences de l'agrégat.

- Du point de vue des incompatibilités, une ontologie des incompatibilités pourrait être définie et enrichie de manière itérative et incrémentale. Cette ontologie donnerait lieu à une base de connaissances concernant les incompatibilités potentielles ainsi que leur résolution possible. Un jeu de requêtes intelligentes pourrait être élaboré, à l'aide du langage OQL[38], pour servir des architectes et les assister lors de l'évaluation de la faisabilité d'agrégation.

Le Centre d'Excellence Microsoft (COE[39]) de CGI nous encourage à définir une version adaptée du cadre conceptuel SOCAF qui s'aligne avec notre contexte professionnel et particulièrement avec les services de consultation offerts aux clients. En effet, nous pensons que tous les résultats produits par la thèse peuvent être cristallisés sous forme de plusieurs offres de service tel que le référencement des composants logiciels chez le client pour l'aider à maximiser la réutilisation de ses actifs logiciels.

---

[38] OQL : Ontology Query Langugae
[39] COE : Center Of Excellence

De plus, nous participons souvent dans l'exercice de nos mandats à des validations par les pairs des solutions logicielles techniques, technologiques et applicatives dans les réponses à des appels d'offres. Le processus d'évaluation de la faisabilité de l'agrégation des composants logiciels peut être utilisé pour valider la faisabilité des solutions proposées dans les offres de services aux clients.

CGI compétitionne au niveau international dans le domaine de la consultation informatique[40], la rentabilité de ses services est un facteur important de réussite et de distinction. Nous pensons que des travaux sur les modèles économiques et financiers de nos processus proposés sont nécessaires si nous voulons vulgariser leur utilisation par d'autres conseillers techniques. Ils doivent évaluer les gains pour aider les gestionnaires à choisir entre les propositions des solutions logicielles basées sur l'agrégation des composants.

Avec l'engagement de CGI et son appui, nous pensons poursuivre nos travaux d'améliorations de nos résultats de recherche pour les adapter aux contextes du marché.

[40] http://www.cgi.com/sites/cgi.com/files/pdf/cgi_awards_rankings_f.pdf

ANNEXE I
EXÉCUTION D'UNE AGRÉGATION PAR COORDINATION:
« Portail-outils»

Cette annexe décrit en détails les phases de sélection et de faisabilité de l'exécution d'agrégation par coordination présentée dans la section 7.3. Nous nous concentrons davantage sur les activités d'évaluation de la faisabilité de l'agrégation (ACT.04) et de l'analyse du rapport de cette évaluation (ACT.05). Pour chacune des activités exécutées, nous présentons une figure qui rappelle l'activité et nous décrivons les résultats produits sous forme de données structurées avec les fiches SOCOM, SOCAM et les tableaux correspondants suite à l'application des règles d'agrégation. Pour commencer, nous rappelons brièvement, dans le tableau suivant, la fiche SOCAM de l'agrégat « Portail-outils ».

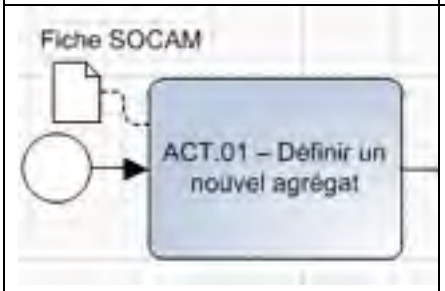
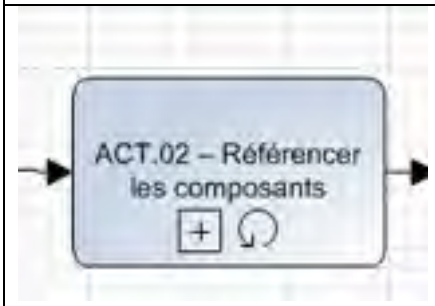
Tableau-A I-1 Fiche SOCAM de l'agrégat par coordination : « Portail-outils » de TELUS

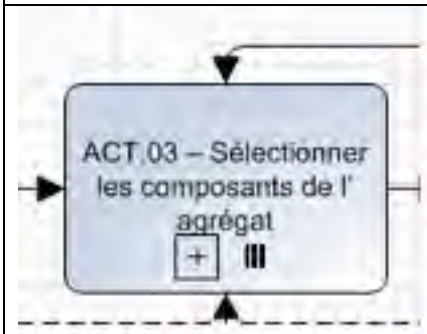
Nom	"Portail-outils" de TELUS	(Contenu complété après l'exécution de l'activité ACT.01)
Auteur	TELUS : Client de CGI	*(Contenu complété après l'exécution de l'activité ACT.01)*
Description générale	Le portail-outils est un point d'entrée du client de Telus, centre CSPQ du Québec, pour utiliser un ensemble d'outils sous-jacents concernant la billetterie, la gestion des pare-feu, la surveillance de l'état du réseau et la gestion documentaire.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Hypothèses	- L'agrégat sera déployé dans les environnements du client - L'agrégat doit être accessible par les utilisateurs internes du Client de TELUS.	*(Contenu complété après l'exécution de l'activité ACT.01)*
Composition statique	Le portail-outils est un composant agrégat composé des composants suivants : **1. SMIS** : Système de Gestion de l'Information (SGI) **2. SecurNet** : Outil de configuration des pare-feu **3. Tivoli WebTop** : Outil de surveillance de l'état du réseau informatique du client **4. Documentum**: Outil de gestion documentaire.	*(Contenu complété après l'exécution de l'activité ACT.01)*

Catégorie de l'agrégation	Coordination	*(Contenu complété après l'exécution de l'activité ACT.05.02)*
Patron d'agrégation	Coordination fédérée	*(Contenu complété après l'exécution de l'activité ACT.06.02)*
Composition dynamique	Les interactions entre les composants participants se résument dans les cinq services suivants : **SER-1** « authentification-sso-portail-outils » : Le service permet aux utilisateurs d'accéder au portail-outils via une interface graphique appropriée. Ce service doit maintenir en session l'identité de l'utilisateur pour promouvoir le SSO aux composants sous-jacents. **SER-2** « authentification-sso-smis » : Le service récupère le nom de l'utilisateur de la session du portail-outils et assure en arrière-plan une nouvelle authentification au composant SMIS. **SER-3** « authentification-sso-documentum » : Le service récupère le nom de l'utilisateur de la session du portail-outils et assure en arrière-plan une nouvelle authentification au composant Documentum. **SER-4** « authentification-pre-sso-securnet » : Le service récupère le nom de l'utilisateur de la session du portail-outils et retour de la page d'authentification à trois paramètres (nom d'utilisateur/Mot de passe/jeton) au réseau virtuel sécurisé du composant SecurNet (VPN sécurisé). **SER-5** « authentification-pre-sso-tivoliwebtop » : Le service récupère le nom de l'utilisateur de la session du portail-outils et retour de la page d'authentification à trois paramètres (nom d'utilisateur/Mot de passe/jeton) au réseau virtuel sécurisé du composant Tivoli-Webtop (VPN sécurisé).	*(Contenu complété après l'exécution de l'activité ACT.06.12)*
Exigences non fonctionnelles	**Sécurité :** - ENF-01 : Authentification unique pour accéder au portail-outils et aux outils sous-jacents. - ENF-02 : Authentification faible requise (nom d'utilisateur/mot de passe)	*(Contenu complété après l'exécution de l'activité ACT.01 et une mise à jour de l'ACT.10)*

	- ENF-03 : L'accès au portail-outils se fera de façon sécurisée. **Disponibilité :** - ENF-04 : Le portail-outils doit être disponible 8h par jour et 5 jours semaines (ENF-dispo-01) **Volumétrie-Charge** - ENF-05: Le portail-outils doit permettre un accès de 300 personnes - ENF-06: 30 accès concurrents sont prévisibles **Ergographie du portail-outils** - ENF-07 : Le portail-outils doit être conforme au PIV du client et aux normes W3C d'accessibilité ENF-07bis : Le portail-outils et les composants participants sont supportés par Internet Explorer Version 6. **Temps de réponse** - ENF-08 : Le portail-outils doit fournir un temps de réponse de 4 s pour l'accès aux pages. - ENF-09 : Le portail-outils doit fournir un temps de réponse de moins de 10 s pour l'accès aux outils sous-jacents.	*(Contenu complété après l'exécution de l'activité ACT.10)*
Autres	N/D	

Tableau-A I-2 Description des activités ACT.01, ACT.02 et ACT.03 de l'agrégat par coordination « Portail-outils»

Graphe de l'activité du processus BPMN	**Description de l'activité et Particularités de son exécution**
	Le processus débute avec l'exécution de l'activité ACT.01 par l'équipe d'affaires à l'aide du gabarit vide de la fiche SOCAM. Durant cette activité, l'équipe d'affaires utilise ce gabarit vide de la fiche SOCAM et produit quelques sections nécessaires pour la suite de l'exécution, soient le nom du composant agrégat, le nom de l'auteur, la description générale, les hypothèses, les cas d'affaires et quelques exigences non fonctionnelles. Ces informations introduisent le contexte et la problématique que l'agrégation propose de résoudre. (Voir le contenu produit durant cette activité dans la deuxième colonne du tableau SOCAM avec une annotation suivante dans la troisième colonne :« *Contenu complété après l'exécution de l'activité ACT.01 »)*
	Après le remplissage partiel de la fiche SOCAM, nous exécutons l'activité ACT.02 en référençant les composants potentiels avec les spécifications SOCOM. Le contexte de l'agrégation a défini les composants participants de Telus qui participeront à la création du nouveau composant agrégat « Portail-outils ». Nous avons ajouté leurs métadonnées à l'entrepôt SOCOM de Telus. **1. SMIS** : Système de Gestion de l'Information (SGI) **2. SecurNet** : Outil de configuration des pare-feu **3. Tivoli WebTop** : Outil de surveillance de l'état du réseau informatique du client

	4. Documentum: Outil de gestion documentaire.
	Ensuite, nous exécutons l'activité ACT.03 qui consiste à sélectionner les composants participants à l'agrégation en faisant une recherche par nom dans l'entrepôt SOCOM de Telus vu qu'ils sont définis par le contexte d'agrégation. Ceci nous permet de définir la composition statique du composant agrégat et de mettre à jour la fiche SOCAM de l'agrégat.

L'activité d'ajout des métadonnées SOCOM des composants participants dans l'entrepôt Telus n'était pas évidente. En effet, nous avons utilisé plusieurs sources pour réaliser ACT.02. Nous avons débuté le référencement avec la lecture des documents existants d'architecture qui nous ont aidés à compléter plus de 70% des métadonnées SOCOM. Nous avons aussi eu accès aux applications et à leur code source pour extraire quelques métadonnées des plateformes et des services. La mise à jour des métadonnées non fonctionnelles a été réalisée via des rencontres planifiées avec l'équipe d'architecture de Telus incluant les architectes d'affaires, d'infrastructure, réseau, sécurité, applicatif et parfois des experts du domaine de télécommunication et des multimédias.

Une fois les composants sélectionnés, nous commençons l'étude de la faisabilité de l'agrégation en exécutant l'activité ACT.04.

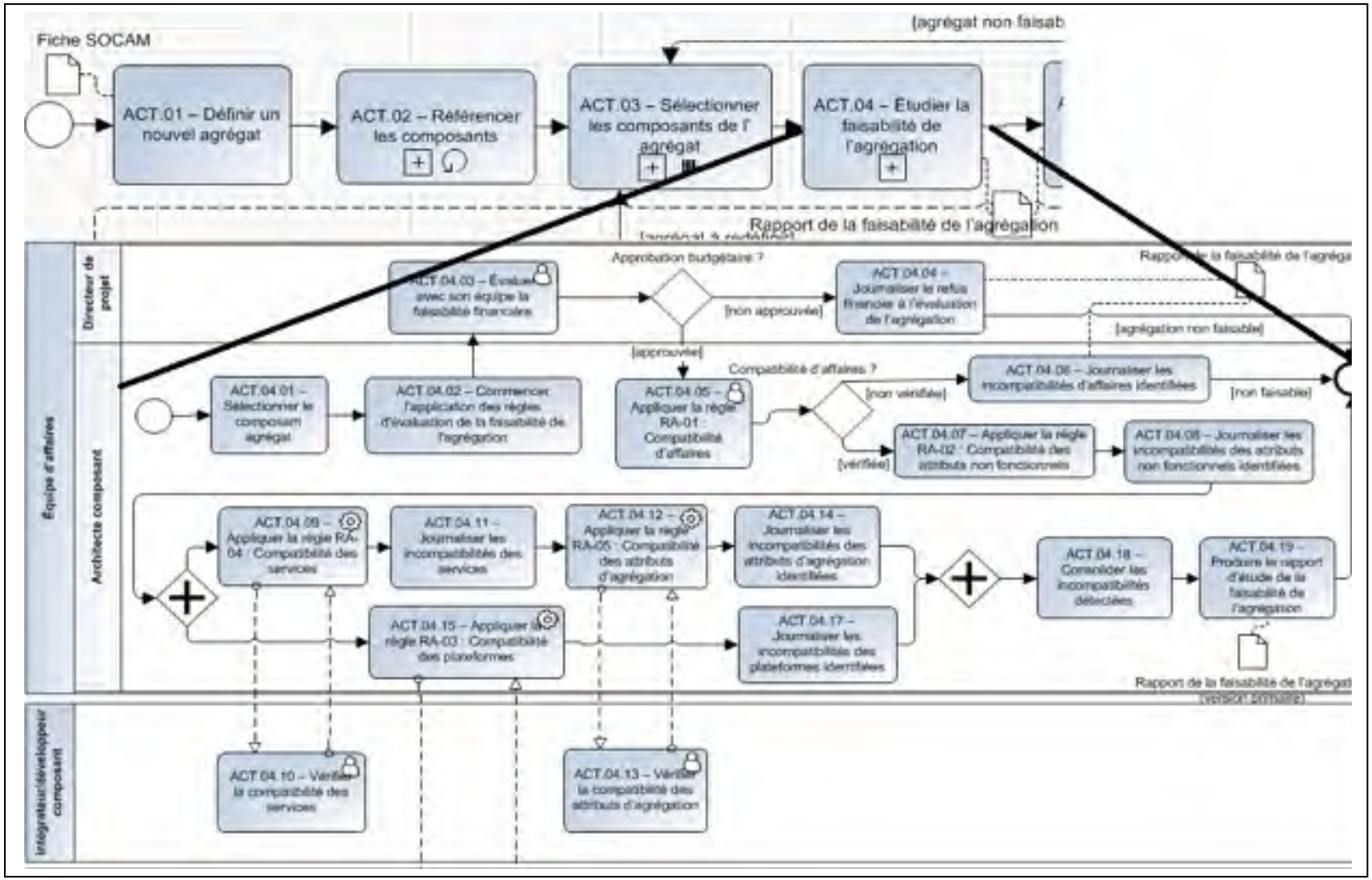
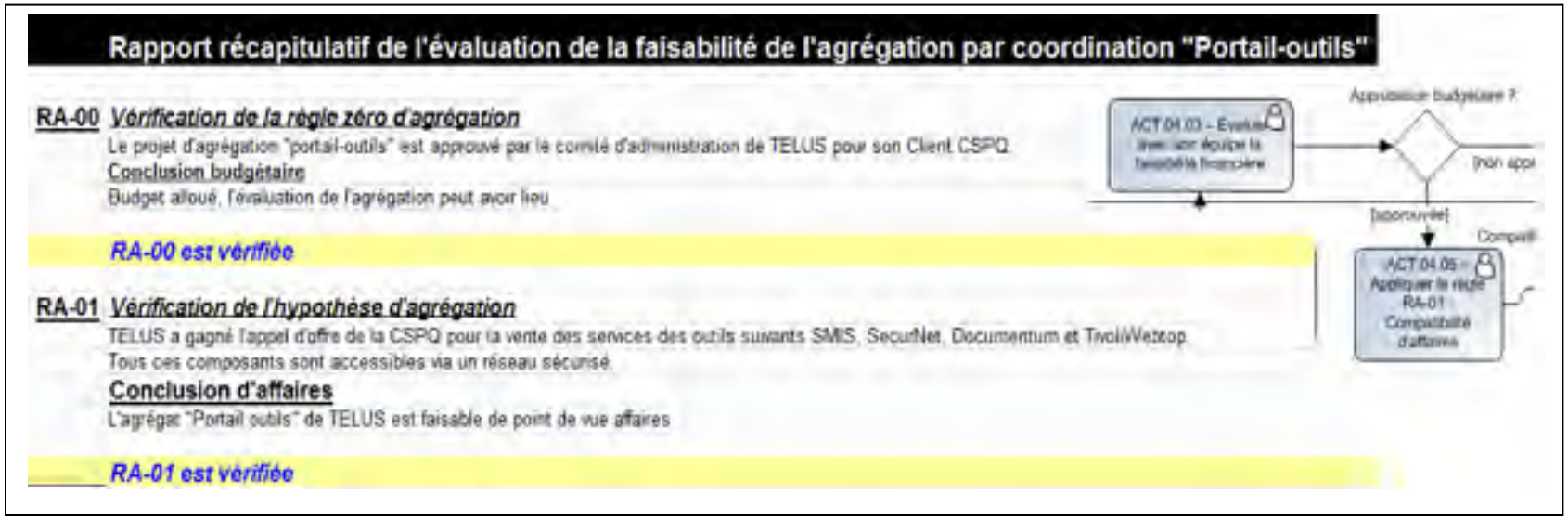


Figure-A I-1 les activités sous-jacentes de l'activité ACT.04 du processus SOCAP

Le composant agrégat « Portail-outils » est un point d'entrée aux autres composants informationnels et opérationnels de Telus. Nous avons identifié, durant l'activité ACT.04, des incompatibilités de type non fonctionnel telles que zonage réseau, disponibilité, temps de réponse, sécurité et gestion de l'identité. Aussi les plateformes réseaux des composants participants doivent être accessibles pour garantir les accès depuis le composant agrégat qui doit être hébergé dans une zone Internet publique. Ces incompatibilités appartiennent aux classes d'incompatibilités syntaxiques, sémantiques et pragmatiques. Le rapport d'évaluation de l'agrégation est composé d'un ensemble de tableaux. La fiche suivante résume l'application des règles RA-00 et RA-01 dans notre outil Excel.

Tableau-A I-3 Rapport d'évaluation de la faisabilité de l'agrégation par coordination « Portail-outils»



Rapport récapitulatif de l'évaluation de la faisabilité de l'agrégation par coordination "Portail-outils"

RA-00 ***Vérification de la règle zéro d'agrégation***

Le projet d'agrégation "portail-outils" est approuvé par le comité d'administration de TELUS pour son Client CSPQ.

Conclusion budgétaire

Budget alloué, l'évaluation de l'agrégation peut avoir lieu

RA-00 est vérifiée

RA-01 ***Vérification de l'hypothèse d'agrégation***

TELUS a gagné l'appel d'offre de la CSPQ pour la vente des services des outils suivants SMIS, SecurNet, Documentum et Tivoli/Vetstop

Tous ces composants sont accessibles via un réseau sécurisé

Conclusion d'affaires

L'agrégat "Portail outils" de TELUS est faisable de point de vue affaires

RA-01 est vérifiée

Les tableaux suivants présentent les résultats d'application de chacune de nos règles d'agrégation de la section 5.2 : RA-02, RA-03, RA-04 et RA-05. Nous énumérons toutes les valeurs des attributs de SOCOM dans la première colonne. Pour chaque catégorie de ces métadonnées, nous examinons si les valeurs de l'attribut des composants participants satisfont les exigences du composant agrégat. Quand l'exigence est satisfaite, nous ajoutons « Compatibilité vérifiée » dans la colonne des résultats et une explication au besoin, sinon nous mentionnons « Incompatibilité détectée » dans la même colonne avec une courte description. Parfois, l'attribut ne présente pas un sujet de compatibilité ou d'incompatibilité dans ce cas nous ajoutons « N/A » pour signifier « Non Applicable ».

Tableau-A I-4 Résultat de l'évaluation des exigences non fonctionnelles (ENF)
de l'agrégation par coordination « Portail-outils », application de la règle RA-02

Règle RA-02 : Compatibilité Non Fonctionnelle

Légende des résultats:

Rouge	Incompatibilité détectée
Gras	**Compatibilité vérifiée**
N/A	Compatibilité/Incompatibilité Non Applicable

Conclusion: Incompatibilités Non Fonctionnelles détectées

ID			Portail-outils	Documentum	SMIS	Ilvoil/Webtop	Securiflet	Résultat de l'évaluation des compatibilités non fonctionnelles
ID	**Sécurité :**							
ENF-01	Authentification unique pour accéder au portail-outils et aux outils sous-jacents	Faisabilité	Oui	Oui	Oui	Non	Non	*Incompatibilité détectée*: SSO n'est pas implémentable à tous les composants outils
ENF-02	Authentification faible requise pour les outils informationnels (nom d'utilisateur/mot de passe)	Faisabilité	Oui	Oui	Oui	Non	Non	**Compatibilité vérifiée:** Authentification faible n'est pas applicable à tous les composants outils
ENF-03	Authentification forte pour les outils sécurisés (Jeton)	Faisabilité	N-A	N/A	N/A	Oui	Oui	*Incompatibilité détectée*: Authentification forte n'est pas applicable à tous les composant
ENF-04	Les accès au portail-outils et aux outils sous-jacents devront être sécurisés (https)	Faisabilité	Oui	Oui	Oui	Oui	Oui	**Compatibilité vérifiée:** tous les accès au portail et au outils sont sécurisés
	Disponibilité :							
ENF-05	le portail-outils doit être disponible 5h par jour et 5 jours semaine	Faisabilité	8/5	8/5	8/5	24/7	24/7	*Incompatibilité détectée*, le portail est le point d'entrée, il doit être 24/7
	Volumétrie-Charge							
ENF-06	le portail-outils doit permettre un accès de 300 personnes	Faisabilité	Oui	Oui	Oui	N/A	N/A	**Compatibilité vérifiée**
ENF-07	10 accès concurrents sont privilégiés	Faisabilité	Oui	Oui	Oui	N/A	N/A	**Compatibilité vérifiée**
	Ergographie du portail-outils							
ENF-08	le portail-outils doit être conforme au PIV du client et aux normes W3C d'accessibilité	Faisabilité	Oui	Non	Oui	Non	Non	*Incompatibilité détectée*: Tous les composants ne peuvent pas être conforme au PIV
	Temps de réponse							
ENF-09	le portail-outils doit avoir un temps de réponse de 4s pour l'accès aux pages	Faisabilité	Oui	N/A	N/A	N/A	N/A	**Compatibilité vérifiée**
ENF-10	le portail-outils et les outils sous-jacents doivent avoir un temps de réponse de moins de 10s pour l'accès aux outils sous-jacents	Faisabilité	Oui	Non	Oui	Oui	Oui	*Incompatibilité détectée*: Documentum n'offre pas un temps de réponse <= à 10s (~35s à 45s)

Tableau-A I-5 Résultat de l'évaluation des attributs des plateformes de l'agrégation par coordination « Portail-outils», application de la règle RA-03.

Régle RA-03 : Compatibilité des plateformes

Légende des résultats:

Italique *Incompatibilité détectée*

Gras **Compatibilité vérifiée**

Conclusion: Incompatibilités des plateformes détectées N/A Compatibilité/Incompatibilité **N**on **A**pplicable

ID	Type de la plateforme		Portail-outils	Documentum	SMIS	Tivoli/Webtop	Securiet	Résultat de l'évaluation des compatibilités de qualité
PLT-01	Plateforme de développement	Compatibilité	N/A	N/A	N/A	N/A	N/A	**N/A**
PLT-02	Plateforme de déploiement	Compatibilité	N/A	N/A	N/A	N/A	N/A	**N/A**
PLT-03	Plateforme de données	Compatibilité	N/A	N/A	N/A	N/A	N/A	**N/A**
PLT-04	Plateforme d'intégration	Compatibilité	WAS/ Weblogic	N/A	WAS/ Weblogic	N/A	N/A	**Compatibilité vérifiée:** Le portail-outils et SMIS sont développés dans la même Plateforme d'intégration (WAS/ Weblogic.)
PLT-05	Plateforme portail	Compatibilité	Env Portail TELUS	N/A	Env Portail TELUS	N/A	N/A	**Compatibilité vérifiée**
PLT-06	Plateforme système	Compatibilité	Unix-AIX	Unix-AIX	Unix-AIX	RedHat	Unix-AIX	**Compatibilité vérifiée**
PLT-07	Plateforme de modélisation	Compatibilité	N/A	N/A	N/A	N/A	N/A	**N/A**
PLT-08	Plateforme réseau	Compatibilité	TEN	TEN	TEN	OSS	OSS	*Incompatibilité détectée:* Les composants sont dans des zones réseaux différents (problème d'interopérabilité potentiel)
PLT-09	Plateforme de sécurité		LDAP	LDAP	LDAP	Mst-AD	Mst-AD	*Incompatibilité détectée:* répertoires d'utilisateurs différents(Problème potentiel de communication)

Tableau-A I-6 Résultat de l'évaluation des attributs des services de l'agrégation par coordination « Portail-outils», application de la règle RA-04

Règle RA-04 : Compatibilité des services

Légende des résultats:

Italique *Incompatibilité détectée*

Gras **Compatibilité vérifiée**

N/A Compatibilité/incompatibilité Non Applicable

Conclusion: compatibilité des services vérifiée

		Portail-outils			Documentum			SMIS			Tivoli Webtop			Securilet			Résultat de l'évaluation des compatibilités des services
ID	Nom de service	Type	Param	Retour	Type	Param	Retour	Type	Param	Retour	Type	Param	Retour	Type	Param	Retour	
SRV-01	Authentification au portail-outils	fourni	n/a	object	requis	n/a	object	requis	n/a	object	requis	n/a	object	requis	n/a	object	**Compatibilité vérifiée:**
SRV-02	Authentification à Documentum	requis	n/a	object	fourni	n/a	object										**Compatibilité vérifiée:**
SRV-03	Gestion des ressources numériques				fourni	n/a	object										N/A
SRV-04	Authentification forte à SecurNet	requis	n/a	object										fourni	n/a	object	**Compatibilité vérifiée:**
SRV-05	Authentification forte à Tivoli Webtop	requis	n/a	object							fourni	n/a	object				**Compatibilité vérifiée:**
SRV-06	Authentification à SMIS	requis	n/a	object				fourni	n/a	object							**Compatibilité vérifiée:**
SRV-07	gérer les requête client							fourni	n/a	object							N/A
SRV-08	surveiller état du réseau										fourni	n/a	object				N/A
SRV-09	configurer coupe-feu													fourni	n/a	object	N/A

Note : la compatibilité est vérifiée lorsqu'un service fourni par un composant est requis par un autre composant

Tableau-A I-7 Résultat de l'évaluation des attributs d'agrégation de l'agrégation par coordination « Portail-outils», application de la règle RA-05

Règle RA-05 : Compatibilité des attributs d'agrégation

Conclusion: Incompatibilités des attributs d'agrégation

Légende des résultats:

Italique: Incompatibilité détectée

Gras: **Compatibilité vérifiée**

N/A: Compatiblité/Incompatibilité Non Applicable

ID	Attributs d'agrégation	Documentum	SMIS	TivoliWebtop	SecurNet	Résultat de l'évaluation des compatibilités des attributs d'agrégation
AGR-01	Potentiel de couplage	Mixte	Mixte	Horizontal	Horizontal	**Compatibilité vérifiée**
AGR-02	Degré d'accessibilité	Boite grise	Boite grise	Boite noire	Boite noire	N/A
AGR-03	Couche logicielle	T	T	T	T	**Compatibilité vérifiée:** Couplage faisable
AGR-04	Nœud du système	rd.telus.com	smis.telus.com	tivoli.telus.com	securnet.telus.com	N/A
AGR-05	Point d'entrée du composant	https://rnm.gov.qc.ca/rnmRD	https://smis.telus.com/	https://tivoli.gov.qc.ca/login	https://securnet.telus.com/login	**Compatibilité vérifiée**
AGR-06	Méthode d'agrégation	Web Service	Web Service	Web Service	Web Service	**Compatibilité vérifiée**
AGR-07	Type de langage	compilé	compilé	compilé	compilé	**Compatibilité vérifiée**
AGR-08	Compatibilité avec les standards	j2ee, jdk 1.4	j2ee, jdk 1.4	j2ee, jdk 1.4	Microsoft .Net	*Incompatibilité détectée*, Standard différents
AGR-09	Référence du composant agrégat	N/A	N/A	N/A	N/A	N/A

Une fois les règles appliquées, nous obtenons un premier résultat quant à la faisabilité de l'agréation. Dans le contexte de l'agrégation par coordination « Portail-outils», nous avons détecté plusieurs incompatibilités de différents types. Ainsi, dans la première itération l'agrégation est qualifiée de **non faisable**. Dans ce cas, une analyse de ces incompatibilités est nécessaire pour décider de leur validité, d'où l'importance de l'activité ACT.05.01 dont l'exécution apporte des clarifications quant à la pertinence des incompatibilités détectées et de leur rétention (ACT.05.02). Puis, nous avons classifié les incompatibilités (ACT.05.03) et leur résolution (ACT.05.04), toutefois nous avons eu recours à une deuxième itération pour les incompatibilités suivantes : conformité au PIV du client et le temps de réponse qui devrait être inférieur à 10 secondes. Pour ces dernières, nous avons opté pour un développement personnalisé pour le premier et une optimisation du code pour le deuxième.

La figure suivante montre les activités sous-jacentes de l'analyse du rapport d'évaluation. Cette figure montre le flux de contrôle de l'activité ACT.05 et les tableaux qui suivent présentent le flux des données. Ces tableaux contiennent deux lignes dans l'entête, une pour rappeler l'activité exécutée (ACT.05.xx) et l'autre pour décrire le type du résultat obtenu (p. ex : 6- classification de l'incompatibilité).

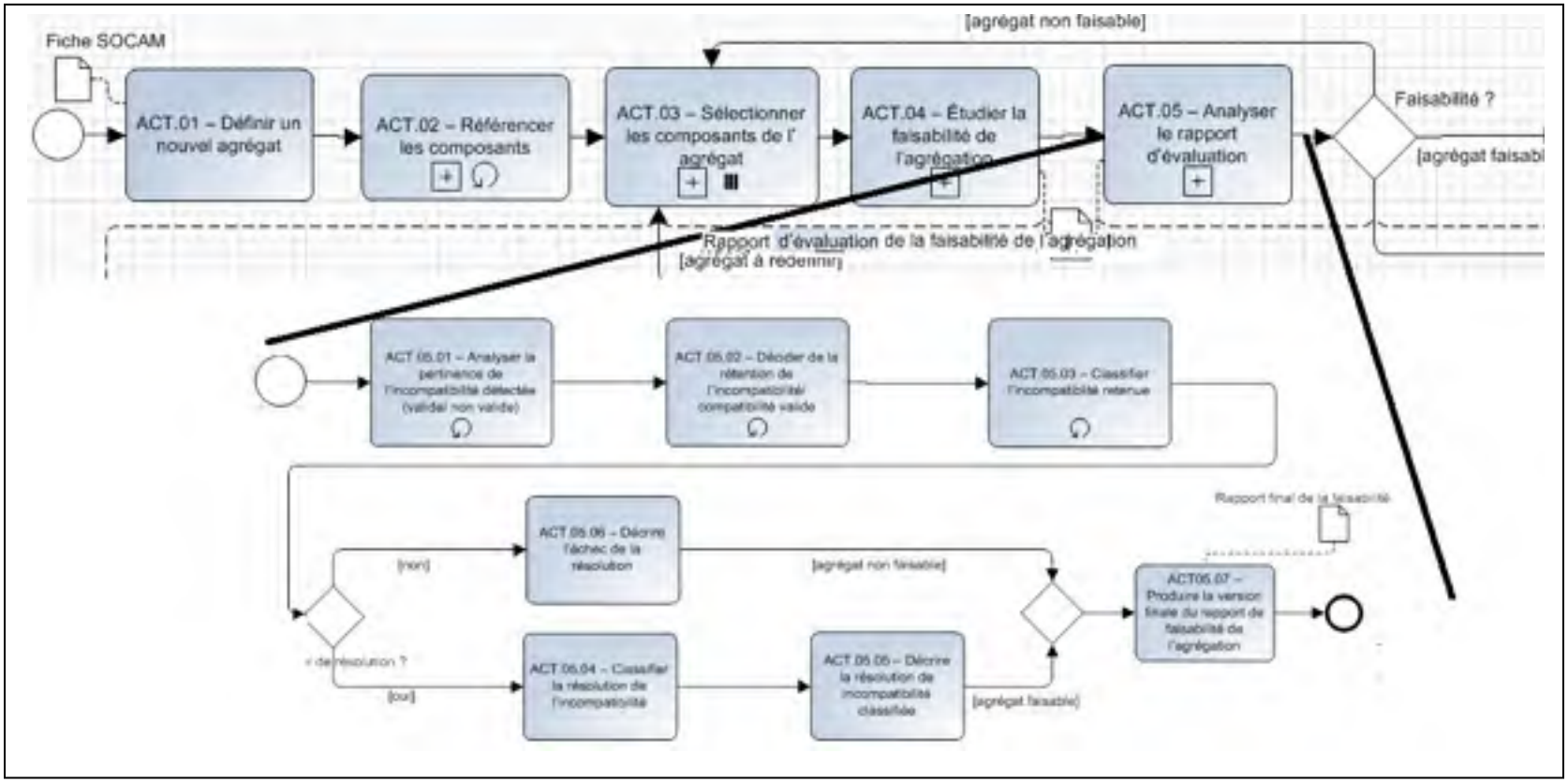


Figure-A I-2 Les activités sous-jacentes de l'activité exécutée ACT.05 du processus SOCAP

Durant l'activité ACT.05, nous avons décidé de la pertinence des incompatibilités identifiées dans l'ACT.04. Nous avons conclu qu'il y a quelques incompatibilités non valides telles que ENF-02, ENF-03, PLT-01, PLT-02, SRV-01, AGR-06, etc. et d'autres qu'il faut résoudre telles que ENF-01, ENF-05, PLT-08, PLT-09, AGR-02, etc. Toutes les résolutions possibles sont journalisées dans la neuvième colonne des tableaux suivants. À la fin de l'exercice de résolution, nous avons conclu que toutes les incompatibilités valides peuvent être résolues. Ainsi, l'exercice de résolution aboutit et la réalisation de l'agrégation devient **faisable**.

Tableau-A I-8 Résultats de la règle RA-02 suite à l'exécution des activités sous-jacentes de l'activité ACT.05

RA 02 Vérification des incompatibilités non fonctionnelles

		Exécution de l'activité ACT.04.08 et ACT.04.10	Exécution de l'activité ACT.05.01	Exécution de l'activité ACT.05.02	Exécution de l'activité ACT.05.02	Exécution de l'activité ACT.05.03	Exécution de l'activité ACT.05.04 ou ACT.05.05	Exécution de l'activité ACT.05.04	Exécution de l'activité ACT.05.05
ID	1- Sécurité :	2- Résultat de l'évaluation des compatibilités de qualité	3- Pertinence /validation de l'incompatibilité/incompatibilité détectée	4- Décision de rétention (Oui/non)	5- Commentaire/ Justification	6- Classification de l'incompatibilité	7- Potentiel de résolution (oui/non)	8- Catégorie de la résolution	9- Description de la résolution
ENF-01	Authentification unique pour accéder au portail-outils et aux outils sous-jacents	Incompatibilité détectée: SSO n'est pas implémentable à tous les composants outils	valide	oui	à résoudre	Incompatibilité syntaxique/ Protocole des interfaces	Oui	SSO Partiel	Le SSO peut s'appliquer sur le portail-outils et les composants outils informationnels (SMS et Documentum). Pour les composants sécurisés (Tivoli Webtop et Securilet), le nom de l'utilisateur peut s'ajouter aux interfaces d'authentification aux VMs des outils sécurisés. Une saisie du mots de passe et du Jeton est requise pour accéder au VMs et ensuite aux composants Securilet et Tivoli Webtop
ENF-02	Authentification faible requise pour les outils informationnels (nom d'utilisateur mot de passe)	Compatibilité vérifiée: Authentification faible n'est pas applicable à tous les composants outils	non valide, à retirer	non	pas besoin d'avoir une authentification simple pour tous les composants outils	n/a	n/a	n/a	n/a
ENF-03	Authentification forte pour les outils sécurisés (Jeton)	Incompatibilité détectée Authentification forte n'est pas applicable à tous les composant	non valide, à retirer	non	pas besoin d'avoir une authentification forte pour tous les composants outils	n/a	n/a	n/a	n/a
ENF-04	Les accès au portail-outils et aux outils sous-jacents devrait être sécurisé. (https)	Compatibilité vérifiée: tous les accès au portail et au outils sont sécurisés	valide	non	n/a	n/a	n/a	n/a	n/a
	Disponibilité :								
ENF-05	le portail-outils doit être disponible 8h par jour et 5 jours semaines	Incompatibilité détectée, le portail est le point d'entrée, il doit être 24/7	valide	oui	à résoudre	Incompatibilités sémantiques et pragmatiques/ Respect des règles d'affaires	Oui	Extension de support	Le centre de Service du Client prendra en charge le support du portail-outils en mode 24/7. Ceci permettra aux utilisateurs des outils le support pour l'accès aux outils sous-jacents en mode 24/7

Tableau-A I-9 Résultats de la règle RA-02 suite à l'exécution des activités sous-jacentes de l'activité ACT.05 (Suite)

	Disponibilité :								
ENF-05	le portail-outils doit être disponible 8h par jour et 5 jours semaines	Incompatibilité détectée, le portail est le point d'entrée, il doit être 24/7	valide	oui	à résoudre	Incompatibilités sémantiques et pragmatiques/ Respect des règles d'affaires	Oui	Extension de support	Le centre de Service du Client prendra en charge le support du portail-outils en mode 24/7. Ceci permettra aux utilisateurs des outils le support pour l'accès aux outils sous-jacents en mode 24/7
	Volumétrie-Charge								
ENF-06	le portail-outils doit permettre un accès de 500 personnes	Compatibilité vérifiée	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
ENF-07	30 accès concurrents sont prévisibles	Compatibilité vérifiée	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
	Ergographie du portail-outils								
ENF-08	le portail-outils doit être conforme au PIV du client et aux normes W3C d'accessibilité	Incompatibilité détectée. Tous les composants ne peuvent pas être conforme au PIV	valide	oui	à résoudre	Incompatibilités sémantiques et pragmatiques/ Respect des règles d'affaires	Oui	Nouveau développement	Des développements supplémentaires se feront pour adapter le portail-outils. SIHS et Documentum au PIV du Client. Les composants SecurNet et TivoliWebtop sont des outils nationaux de TELUS ils ne peuvent pas être adaptés
	Temps de réponse								
ENF-09	le portail-outils doit avoir un temps de réponse de 4s pour l'accès aux pages	Compatibilité vérifiée	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
ENF-10	le portail-outils et les outils sous jacents doivent avoir un temps de réponse de moins de 10s pour l'accès aux outils sous-jacents	Incompatibilité détectée Documentum n'offre pas un temps de réponse <= à 10s (~35s à 45s)	valide	oui	à résoudre	Incompatibilités sémantiques et pragmatiques/ Respect des règles d'affaires	Oui	développement d'optimisation	Le centre d'optimisation/qualité de TELUS fera des tests de performance et d'optimisation pour identifier les origines techniques des temps de réponse élevés. Il fournira un rapport de recommandation au centre de développement pour améliorer le composant Documentum. (Une pratique similaire a été faite pour un autre client, l'intervention a produit l'amélioration escomptée.)

RA-03 est non vérifiée

Tableau-A I-10 Résultats de la règle RA-03 suite à l'exécution des activités sous-jacentes de l'activité ACT.05

RA-03 Vérification des attributs des plateformes des composants

		Exécution de l'activité ACT.04.06 et ACT.04.10	Exécution de l'activité ACT.05.01	Exécution de l'activité ACT.05.02	Exécution de l'activité ACT.06.02	Exécution de l'activité ACT.06.03	Exécution de l'activité ACT.04.04 ou ACT.05.03	Exécution de l'activité ACT.05.04	Exécution de l'activité ACT.05.05
ID	1- Type de la plateforme	2- Résultat de l'évaluation des compatibilités de qualité	3- Pertinence/validation de	4- Décision de rétention (Oui/non)	5- Commentaire Justification	6- Classification de l'incompatibilité	7- Potentiel de résolution (oui/non)	8- Catégorie de la résolution	9- Description de la résolution
PLT-01	Plateforme de développement	N/A	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
PLT-02	Plateforme de déploiement	N/A	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
PLT-03	Plateforme de données	N/A	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
PLT-04	Plateforme d'intégration	**Compatibilité vérifiée:** La portail-outils et SMS sont développés dans le même Plateforme	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
PLT-05	Plateforme portail	**Compatibilité vérifiée**	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
PLT-06	Plateforme système	**Compatibilité vérifiée**	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
PLT-07	Plateforme de modélisation	N/A	valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
PLT-08	Plateforme réseau	Incompatibilité détectée. Les composants sont dans des zones réseaux différents (problème d'interopérabilité potentiel)	valide	oui	Les deux zonages différents des deux types d'outils peuvent bloquer l'accès, à **résoudre**	**Incompatibilités des plateformes/Réseau**	Oui	VPN sécurisé (Proxy)	Une connexion à un réseau virtuel privé sécurisé de TELUS sera proposée (solution proxy) pour permettre ensuite permettre l'accès des utilisateurs de la zone internet public de TELUS (TEN-TELUS Enterpise Network(-)) à la Zone intranet sécurisée (OSS (Oass Service Sites)
PLT-09	Plateforme de sécurité	Incompatibilité détectée répertoires d'utilisateurs différents Problème potentiel de communication)	valide	oui	les deux types de répertoires d'utilisateurs dans deux technologies différentes sont bloquants pour implanter le SSO, à **résoudre**	**Incompatibilités des plateformes/Sécurité**	Oui	développement d'optimisation	La communication entre les répertoires des utilisateurs des composants outils se fera via les services du portail-outils. Il y aura une récupération des noms d'utilisateurs du répertoire LDAP, ensuite une vérification de ces comptes avec les ADs (Active Directory) de Tivoli-WebTop et SecurNet. Une fois la vérification faite, la connexion au VPN est possible par l'inscription des noms récupérés dans l'interface d'authentification du Citrix (Application d'authentification au VPN sécurisé)

RA-03 est non vérifiée

Tableau-A I-11 Résultats de la règle RA-04 suite à l'exécution des activités sous-jacentes de l'activité ACT.05

RA-04 Vérification des attributs des services des composants

		Exécution de l'activité ACT.04.02 et ACT.04.10	Exécution de l'activité ACT.05.01	Exécution de l'activité ACT.05.02	Exécution de l'activité ACT.05.02	Exécution de l'activité ACT.05.03	Exécution de l'activité ACT.05.04 ou ACT.05.05	Exécution de l'activité ACT.05.04	Exécution de l'activité ACT.05.05
ID	Nom du service	2- Résultat de l'évaluation des compatibilités de qualité	3- Pertinence/validation de	4- Décision de rétention (Oui/non)	5- Commentaire/ Justification	6- Classification de l'incompatibilité	7- Potentiel de résolution (oui/non)	8- Catégorie de la résolution	9- Description de la résolution
SRV-01	Authentification au portail-outils	Compatibilité vérifiée:	Valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
SRV-02	Authentification à Documentum	Compatibilité vérifiée:	Valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
SRV-03	Gestion des ressources numériques	N/A	Valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
SRV-04	Authentification forte à SecurNet	Compatibilité vérifiée:	Valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
SRV-05	Authentification forte à Trivoli Webtop	Compatibilité vérifiée:	Valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
SRV-06	Authentification à SVDS	Compatibilité vérifiée:	Valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
SRV-07	gérer les requête client	N/A	Valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
SRV-08	surveiller état du réseau	N/A	Valide, à retirer	non	n/a	n/a	n/a	n/a	n/a
SRV-09	configurer coupe-feu	N/A	Valide, à retirer	non	n/a	n/a	n/a	n/a	n/a

RA-03 est vérifiée, l'agrégation est faisable

Tableau-A I-12 Résultats de la règle RA-05 suite à l'exécution des activités sous-jacentes de l'activité ACT.05

RA-05 Vérification des attributs d'agrégation des composants

		Exécution de l'activité ACT.04.08 et ACT.04.18	Exécution de l'activité ACT.05.01	Exécution de l'activité ACT.05.02	Exécution de l'activité ACT.05.03	Exécution de l'activité ACT.05.04 ou ACT.05.06	Exécution de l'activité ACT.05.04	Exécution de l'activité ACT.05.05
ID	Type de la plateforme	2- Résultat de l'évaluation des compatibilités de qualité	3- Pertinence/validation de	4- Décision de rétention (Oui/non)	5- Commentaire/ Justification	6- Classification de l'incompatibilité	7- Potentiel de résolution (oui/non)	8- Catégorie de la résolution	9- Description de la résolution
AGR-01	Potentiel de couplage	Compatibilité vérifiée	n/a	n/a	l'agrégation ne nécessite pas une intégration forte (fusion du code ou interopérabilité du code) la différence dans le langage de programmation ne	n/a	n/a	n/a	n/a
AGR-02	Méthode d'agrégation	Compatibilité vérifiée	n/a	oui	Vue que le zonage est différent l'accès au composant par un lien URL n'est pas assez pour réussir l'agrégation, **à résoudre**	**Incompatibilités architecturales/ Méthode d'agrégation**	Oui	EAI - Bridge VPN	Les liens des outils sécurisé (securiset et Tivoli Webtop) ajoutés au portail-outils ne sont pas fonctionnels vue que la zone Telus Public ne permet pas l'accès aux outils dans la zone sécurisée. La solution proposée consiste à rediriger les utilisateurs des outils sécurisés vers le portail Citrix sécurisé (ce qui permet l'accès au VPN sécurisé). Leur nom d'utilisateur seront réutilisés, ensuite il ajoute leur mot de passe et leur jeton pour accéder à Citrix. Une fois connecté à Citrix, les utilisateurs seront redirigés vers les interfaces d'authentification des outils
AGR-03	Couche logicielle	Compatibilité vérifiée: Couplagefaisable	n/a	n/a	n/a	n/a	n/a	n/a	n/a
AGR-04	Nœud du système	N/A	n/a	n/a	n/a	n/a	n/a	n/a	n/a
AGR-05	Point d'entrée du composant	Compatibilité vérifiée	n/a	n/a	n/a	n/a	n/a	n/a	n/a
AGR-06	Degré d'accessibilité	Compatibilité vérifiée	non valide	n/a	l'agrégation ne nécessite pas une intégration forte (fusion du code ou interopérabilité du code) la différence dans le langage de programmation ne	n/a	n/a	n/a	n/a
AGR-07	Type de langage	Compatibilité vérifiée	n/a	n/a	n/a	n/a	n/a	n/a	n/a
AGR-08	Compatibilité avec les standards	Incompatibilité détectée. Standard différents	valide, à retenir	non	L'agrégation ne nécessite pas une interopérabilité des standards. Cette différence ne bloquent	n/a	n/a	n/a	n/a
AGR-09	Référence du composant agrégat	N/A	n/a	n/a	n/a	n/a	n/a	n/a	n/a

RA-05 est non vérifiée

Le tableau suivant montre trois colonnes. Dans la première, nous constatons les résultats des comparaisons systématiques entre les métadonnées SOCOM des composants participants suite à l'évaluation de toutes les règles d'agrégation, ce qui correspond à l'exécution de l'activité ACT.04. Les deux autres colonnes montrent les incompatibilités pertinentes, retenues et résolues suite à l'exécution de l'activité ACT.05.

Tableau-A I-13 Sommaire des incompatibilités détectées, retenues et résolues
suite à l'exécution des activités ACT.04 et ACT.05

Conclusion suite à l'évaluation primaire	Conclusion suite à la possibilité de résolution des incompatibilités pertinentes/valides:	Conclusion suite à la résolution des incompatibilités pertinentes/valides:
Cet agrégat valide la règle RA-00 Cet agrégat valide la règle RA-01 Cet agrégat ne valide pas la règle RA-02 Cet agrégat ne valide pas la règle RA-03 Cet agrégat valide la règle RA-04 Cet agrégat ne valide pas la règle RA-05 *# L'agrégat est non faisable*	Les incompatibilités pertinentes et retenues suivar **ENF-01, ENF-05, ENF-08, ENF-10, PLT-08, PLT-09 et AGR-02** (Voir la colonne "Pistes de résolution") *# L'agrégat est faisable*	Les incompatibilités pertinentes et retenues sont **résolues.** **ENF-01, ENF-05, ENF-08, ENF-10, PLT-08, PLT-09 et AGR-02** (Voir la colonne "Description de la résolution") *# L'agrégat peut être conceptualisé*

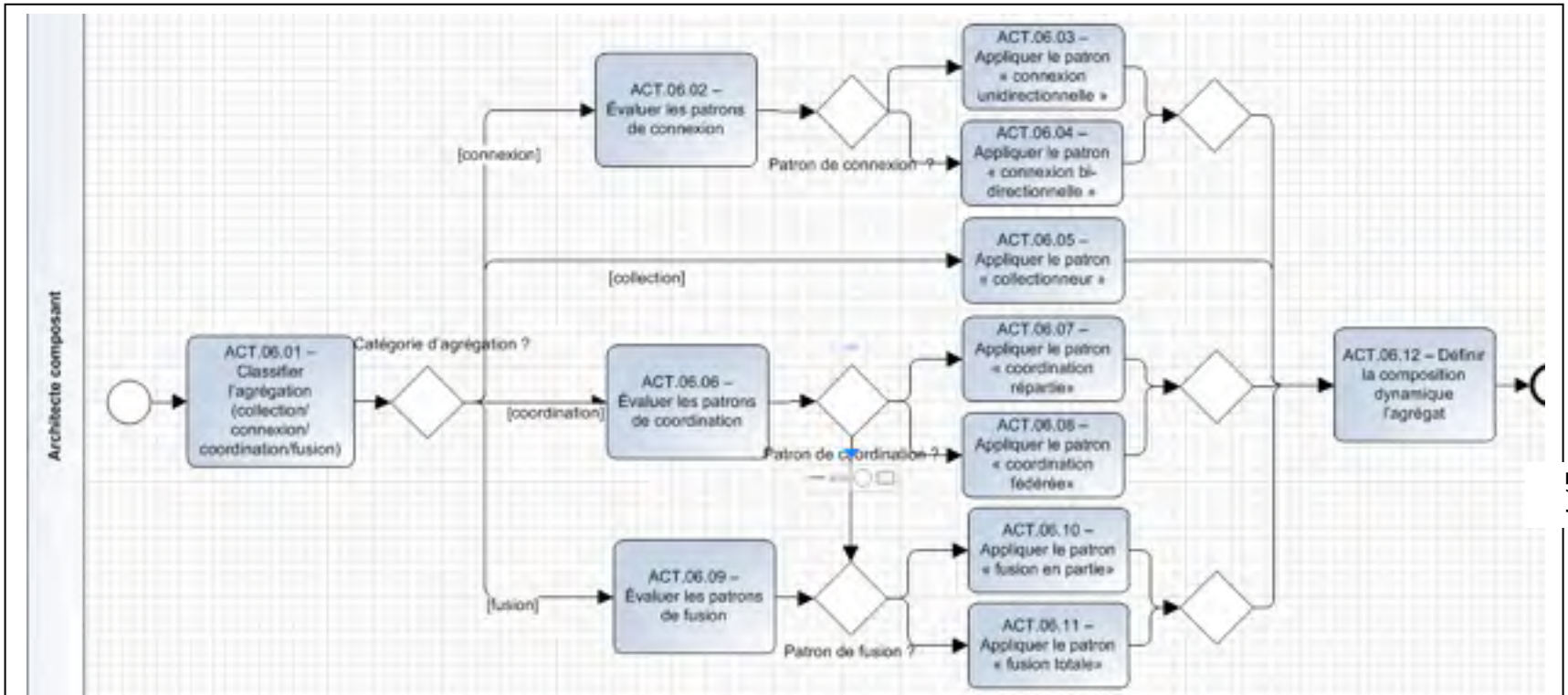


Figure-A I-3 Activités sous-jacentes de l'activité ACT.06 de SOCAP :"Concevoir un agrégat"

275

Après ces activités, l'exécution du processus continue à partir de l'activité ACT.06. Durant cette activité, nous avons qualifié cette agrégation de« **Coordination** » satisfaisant le patron « **fédérateur** ». Ceci est obtenu par l'exécution des activités ACT.06.06 et ACT.06.08. Ce résultat de classification de l'agrégation est détaillé dans la section 3.3. Aussi, durant cette activité de conception, nous avons mis à jour la fiche SOCAM de l'agrégat en définissant sa composition dynamique (voir fiche SOCOM présentée par le Tableau 7.2).

Nous n'avons pas présenté les détails de l'exécution de la suite des activités car nous jugeons que ces dernières sont connues pour la majorité des développements à base de composants et ne présentent pas une particularité de notre processus. Toutefois, nous rappelons qu'à la fin de l'exécution et en cas d'approbation par le client du composant agrégat, les fiches SOCOM des composants agrégés sont mises à jour (attribut SOCOM: référence des composants agrégats) et une nouvelle fiche SOCOM du composant agrégat est créée pour référencer ce dernier dans l'entrepôt SOCOM de Telus. La figure suivante montre la page web de l'agrégat portail-outils et le menu de gauche encadré énumère les liens d'accès aux composants outils informationnels et opérationnels.

Figure-A I-4 Page d'accueil du composant agrégat par coordination « Portail-outils »

ANNEXE II
LES FICHES SOCOM DU COMPOSANT AGRÉGAT RÉSULTANT DE L'AGRÉGATION PAR COORDINATION

Voici le tableau de fiche SOCOM du nouveau composant agrégat suite à l'agrégation par coordination: Composant « Portail-outils ».

Tableau-A II-1 Fiche SOCOM du composant agrégat « Portail-outils», résultant de l'agrégation par coordination

Liste des attributs SOCOM	Valeurs des attributs
1. Attributs statiques	
1.1. Code du composant	CMP_SOCOM_Portail-outils
1.2. Nom du composant	Portail outils TELUS
1.3 Description du composant	Point d'entrée pour l'accès aux outils informationnels et sécurisés des opérations du client de TELUS
1.4. Langage du composant	Java
1.5. Version du composant	Stable
1.6. Emplacement du composant	Privé
1.7. Auteur du composant	TELUS
2. Attributs de service	
2.1. Service 1	Accès aux outils informationnels
2.1.1. Nom du service	accessInfoTools
2.1.2. Type du service	Fourni
2.1.3. Version du service	1
2.1.4. Annotation du service	Permet l'accès aux outils informationnels du Client de TELUS : Documentum (Gestion documentaires) et SMIS (pour la gestion des commandes et le suivi de l'état des incidents).
2.1.5. Visibilité du service	Private
2.1.6. Cas d'utilisation associé	POR_CU_A_0100
2.1.7. Mode d'invocation	synchrone
2.1.8. Type de retour	Void
2.1.9. Paramètres du service	
2.1.9.1. Paramètre 1	

Liste des attributs SOCOM	Valeurs des attributs
2.1.9.1.1. Nom du paramètre	toolURL
2.1.9.1.2. Type paramètre	String
2.1.9.1.3. Visibilité du paramètre	Private
2.1.9.1.4. Rang du paramètre	1
2.1.9.1.5. Type d'accès	Read
2.1.9.2. Paramètre 2	
2.1.9.2.1. Nom du paramètre	ssoActive
2.1.9.2.2. Type paramètre	Boolean
2.1.9.2.3. Visibilité du paramètre	Private
2.1.9.2.4. Rang du paramètre	2
2.1.9.2.5. Type d'accès	Read
2.2. Service 2	Accès aux outils sécurisés
2.2.1. Nom du service	accessInfoTools
2.2.2. Type du service	Fourni
2.2.3. Version du service	1
2.2.4. Annotation du service	Permet l'accès aux outils sécurisés du Client de TELUS : TivoliWebtop (Suivi de l'état du réseau) et SecurNet (Configuration des pare-feu)
2.2.5. Visibilité du service	Private
2.2.6. Cas d'utilisation associé	POR_CU_A_0100
2.2.7. Mode d'invocation	synchrone
2.2.8. Type de retour	Void
2.2.9. Paramètres du service	
2.2.9.1. Paramètre 1	
2.2.9.1.1. Nom du paramètre	toolURL
2.2.9.1.2. Type paramètre	String
2.2.9.1.3. Visibilité du paramètre	Private
2.2.9.1.4. Rang du paramètre	1
2.2.9.1.5. Type d'accès	Read
2.2.9.2. Paramètre 2	
2.2.9.2.1. Nom du paramètre	ssoActive
2.2.9.2.2. Type paramètre	Boolean
2.2.9.2.3. Visibilité du paramètre	Private
2.2.9.2.4. Rang du paramètre	2
2.2.9.2.5. Type d'accès	Read
2.2.9.3. Paramètre 3	
2.2.9.3.1. Nom du paramètre	login
2.2.9.3.2. Type paramètre	String
2.2.9.3.3. Visibilité du paramètre	private

Liste des attributs SOCOM	Valeurs des attributs
2.2.9.3.4. Rang du paramètre	2
2.2.9.3.5. Type d'accès	read
2.3. Service 3	Autentification au portail outils
2.3.1. Nom du service	accessTSRPortal
2.3.2. Type du service	fourni
2.3.3. Version du service	1
2.3.4. Annotation du service	Retourne un écran d'authentification pour permettre au usager l'accès au portail
2.3.5. Visibilité du service	Public
2.3.6. Cas d'utilisation associé	POR_CU_A_0200
2.3.7. Mode d'invocation	synchrone
2.3.8. Type de retour	void
2.3.9. Paramètres du service	
2.3.9.1. Paramètre 1	
2.3.9.1.1. Nom du paramètre	username
2.3.9.1.2. Type paramètre	String
2.3.9.1.3. Visibilité du paramètre	private
2.3.9.1.4. Rang du paramètre	1
2.3.9.1.5. Type d'accès	Read
2.3.9.2. Paramètre 2	
2.3.9.2.1. Nom du paramètre	motdepasse
2.3.9.2.3. Type paramètre	String
2.3.9.2.3. Visibilité du paramètre	private
2.3.9.2.4. Rang du paramètre	2
2.3.9.2.5. Type d'accès	read
3. Attributs d'agrégation	
3.1. Méthode d'agrégation	Accès via un lien
3.3. Potentiel de couplage	Faible
3.4. Degrés d'accessibilité	Boîte blanche
3.5. Couche logicielle	Présentation - Traitement
3.6. Nœud du système	portail.outils.telus.com
3.7. Point d'entrée du composant	https://portail.outils.telus.com/login
3.8. Type de langage	compilé
3.9. Compatibilité avec les standards	j2ee, jdk 1.4
3.10. Référence du composant agrégat	n/a
4. Attributs de qualité	
4.1. Attribut Qualité 1	
4.1.1. Nom de l'attribut	Confidentialité des données

Liste des attributs SOCOM	Valeurs des attributs
4.1.2. Type de l'attribut	sécurité
4.1.3. Description de l'attribut	Le composant doit transiger de façon sécuritaire
4.1.4. Fichier documentation	Doc d'architecture du composant TSR
4.1.5. Valeur	HTTPS - SSL
4.2. Attribut Qualité 2	
4.2.1. Nom de l'attribut	Utilisation du composant dans d'autre contexte
4.2.2. Type de l'attribut	Réutilisation
4.2.3. Description de l'attribut	Le potentiel d'utiliser le même composant dans d'autre contexte que sa création d'origine
4.2.4. Fichier documentation	Document de support pour la réutilisation
4.2.5. Valeur	Possible
5. Attributs de plateforme	
5.1. Plateforme 1	
5.1.1. Nom de la plateforme	Unix
5.1.2. Type de la plateforme	Système d'exploitation
5.1.3. Description de la plateforme	
5.1.4. Fichier documentation	Voir site du fournisseur
5.2. Plateforme 2	
5.2.1. Nom de la plateforme	WSAD
5.2.2. Type de la plateforme	Environnement de développement IBM
5.2.3. Description de la plateforme	WebSphere Studio Application Development
5.2.4. Fichier documentation	Voir site du fournisseur
5.3. Plateforme 3	
5.3.1. Nom de la plateforme	Oracle DB 10g
5.3.2. Type de la plateforme	Base de données
5.3.3. Description de la plateforme	
5.3.4. Fichier documentation	Voir site du fournisseur
5.4. Plateforme de déploiement	
5.4.1. Nom de la plateforme	Serveur d'application IBM Application Server
5.4.2. Type de la plateforme	Déploiement
5.4.3. Description de la plateforme	Environnement de déploiement de TELUS - Division Vancouver

Liste des attributs SOCOM	Valeurs des attributs
5.4.4. Fichier documentation	Voir site du fournisseur
5.6. Plateforme d'intégration	
5.6.1. Nom de la plateforme	N/A
5.6.2. Type de la plateforme	N/A
5.6.3. Description de la plateforme	N/A
5.6.4. Fichier documentation	N/A
5.7. Plateforme du portail cible	
5.7.1. Nom de la plateforme	N/A
5.7.2. Type de la plateforme	N/A
5.7.3. Description de la plateforme	N/A
5.7.4. Fichier documentation	N/A
6. Attributs d'affaires	
6.1. Cas d'utilisation d'affaire 1	
6.1.1. Code du cas	POR_CU_A_0100
6.1.2. Nom du cas	Authentification d'accès
6.1.3. Description du cas	permet l'accès au portail outils TELUS
6.1.4. Acteur du cas	utilisateur humain
6.2. Cas d'utilisation d'affaire 2	
6.2.1. Code du cas	POR_CU_A_0200
6.2.2. Nom du cas	Accéder aux outils du portail TELUS
6.2.3. Description du cas	Permet l'accès aux outils en support aux opérations du client de TELUS
6.2.4. Acteur du cas	utilisateur, administrateur

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Allen, P., & Frosts, S. (Mars 1998). Tackling the Key Issues in CBD. In *Select Software Tools*.

Bachman, e. a. (2000). Volume II: Technical Concepts of Component-Based Software Engineering, *CMU/SEI-2000-TR-008 ESC-TR-2000-007*.

Bechhofer, S. (2003). *DIG Specifications*, version 1.1. Retrieved from http://dl-web.man.ac.uk/dig/2003/02/interface.pdf last access February, 11th 2010

Beugnard, A., Jezequel, J. M., Plouzeau, N., & Watkins, D. (1999). Making components contract aware. *IEEE Computer*, p38–45.

Boertien, N., W.A. Steen, M., & Jonkers, H. (2005). *Evaluation of Component-Based Development Methods*. 13. Retrieved from www.ahaha.demon.nl/docs/EMMSAD01.pdf last access 15 July 2011

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The unified modeling language user guide*. Reading, Mass. ; Don Mills, Ont.: Addison-Wesley.

Chaitanya, P. G., & Ramesh, K.V. (2011). Feasibility study on component based software architecture for large scale software systems. *(IJCSIT) International Journal of Computer Science and Information Technologies*, vol 2(3), 5.

Cheesman, J., & Daniels, J. (2001). *UML Components: simple process for specifying component-based software*.

Coad, P., & Yourdon, E. (1990). *Object-oriented analysis*. Englewood Cliffs, N.J.: Prentice-Hall.

Cummins, F. A. (2002). *Enterprise Integration: An Architecture for Enterprise Application and Systems Integratio*n: A book. OMG press, .

D'Souza, D. F., & Wills, A. C. (1998). *Objects, components, and frameworks with UML : the catalysis approach*. Boston, Mass. ; Montreal: Addison-Wesley.

Daconta, M. , Obrst, L. , & Smith, K. . (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. Knowledge Management* (Vol. 284, p. 281). Wiley. Retrieved from http://www.citeulike.org/user/4vgacias/article/277883

Denno, P., Steves, M. P., Libes, D., & Barkmeyer, E. J. (2003). Model-driven integration using existing models. *IEEE Software*, 20(5), 59-63.

Dowling, T. (2000, 23-25 October). Software COTS Components – Problems, And Solutions? Paper presented at t*he Strategies to Mitigate Obsolescence in Defense Systems Using Commercial Components*, Budapest, Hungary.

Dogru, A. H., & Tanik, M. M. (2003). A process model for component-oriented software engineering. *IEEE Software*, 20(2), 34-41.

Greenberg, J. (2002). *Metadata and the World Wide Web.* Encyclopedia of Library and Information Science, Vol.72, Supplement 35, 244-261.

Gurber, T. (1993). *A translation approach to portable ontologies*, Knowledge Acquisition: Academic Press.

Ivers, J., Clements, P., Garlan, D., Nord, R., Schmerl, B., & Silva, J. R. O. (2004). *Documenting Component and Connector Views with UML 2.0*. Pittsburgh: Software Engineering Institute.

IZZA, S. (2006). *INTEGRATION DES SYSTEMES D'INFORMATION INDUTRIELS, Une approche flexible basée sur les services sémantiques.* Ecole Nationale Supérieure des Mines de Saint-Etienne, Sainte-Étienne.

Kleppe, A. G., Warmer, J. B., & Bast, W. (2003). *MDA explained : the model driven architecture : practice and promise*. Boston: Addison-Wesley.

Kruchten, P. (2004). *The rational unified process : an introduction (3rd ed.)*. Boston: Addison-Wesley.

Larman, C. (2002). Applying UML and patterns : *an introduction to object-oriented analysis and design and the unified process (2nd ed.)*. Upper Saddle River, NJ: Prentice Hall PTR.

M.C. Daconta, L.J. Obrst, & Smith, K. T. (2003). *The Semantic Web, A Guide to the Future of XML, Web Services, and Knowledge Management*.

Magnan, F., & Paquette, G. (2006, June). *TELOS: An ontology driven eLearning OS.* Paper presented at the Proceeding of the SOA-AIS-2006 Worskhop, Dublin, Ireland.

Masmoudi, A., Paquette, G., & Champagne, R. (2005). Agrégation de Composant Dirigée par les Métadonnées. Paper presented at the *INFORSID'05*, Grenoble-France.

Masmoudi, A., Paquette, G., & Champagne, R. (2006a, 1 juin). Implémentation à l'aide de BPEL de trois processus d'agrégation de composants, dirigée par les modèles. Paper presented at the *INFORSID'06*, Hammamet-Tunisie.

Masmoudi, A., Paquette, G., & Champagne, R. (2006b). Metadata-Based software components repository's Reuse. Paper presented at the *I2LOR'06*, Montreal-Canada.

Masmoudi, A., Paquette, G., & Champagne, R. (2008). Metadata-driven software components aggregation process with reuse. *International Journal Advanced Medi and Communication,* p 35-58.

McIlroy, M. D. (1969). Mass-produced Software Components. . Paper presented at the In Proceedings of *NATO Conference on Software Engineering*, New York.

Meyer, B. (1992). Applying "design by contract". *IEEE Computer (Special Issue on Inheritance & Classification),* 25(10), 40–52.

Mikhail, K. (2004). *A methodology of semi-automated software integration: an approach based on logical inference.* Unpublished Computer Science, INSA de rouen.

Mili, H., Mili, A., Yacoub, S., & Addy, E. (2002). *Reuse based software engineering: techniques, organization, and measurement*. New York: Wiley.

Paquette G. (2007) *Graphical Ontology Modeling Language for Learning Environments, Technology, Instruction., Cognition and Learning* , Vol.5 , p.133-168, Old City Publishing, Inc.

Paquette, G., Léonard, M., Lundgren-Cayrol, K., Mihaila, S., & Gareau, D. (2005). Learning Design based on Graphical Knowledge-Modeling *Journal of Educational technology and Society ET&S , Special issue on Learning Design*(Proceedings of the UNFOLD-PROLEARN Joint Workshop, Valkenburh, The Netherlands, September 2005 on Current Research on IMS Learning Design, 2006).

Paquette, G., Rosca, I., Mihaila, S., & Masmoudi, A. (2006). TELOS, a service-oriented framework to support learning and knowledge Management In S. Pierre (Ed.), *E-Learning Networked Environments and Architectures: a Knowledge Processing Perspective*: Springer-Verlag.

Paquette, Gilbert (2002) *Modélisation des connaissances et des compétences, pour concevoir et apprendre*, 357 pp, Presses de l'Université du Québec.

Paquette, Gilbert (1996) La modélisation par objets typés: une méthode de représentation pour les systèmes d'apprentissage et d'aide à la tâche. *Sciences et techniques éducatives* , France, avril 1996

Paquette, G., & Masmoudi, A. (2010). *Ontology-Based Software Component Aggregation. In I. S. Reference (Ed.), Visual Knowledge Modeling for Semantic Web Technologies: Models and Ontologies* (pp. 263-277). New York: Information Science Reference.

Renaux, E., Olivier, C., & Jean-Marc, G. (2004, mars). *Chaîne de production de systèmes à base de composants logiques* Paper presented at the LMO 2004.

Rogozan, D., & Paquette, G. (2005, Septembre). *Managing Ontology Changes on the Semantic Web.* Paper presented at the WI-2005 Web Intelligence Conference, France.

Schoderbek, Schoderbek, & Kefalas. (1985). *Management Systems, Conceptuel consideration (Third Edition ed.).* Plano, Texas: Business Publication,Inc.

Seidewitz, E. (2003). What models mean. *IEEE Software*, 20(5), 26-32.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE Software*, 20(5), 19-25.

Smith, M. K., Welty, C., & McGuinness, D. L. (2004). *OWL Web Ontology Language Guide* (W3C Recommendation).

Szyperski, C., Gruntz, D., & Murer, S. (2002). *Component software : beyond object-oriented programming (2nd ed.)*. New York: ACM Press ; Addison-Wesley.

Stojanovic, Z. (2005). *An Integrated Component-Oriented Framework for Effective and Flexible Enterprise Distributed Systems Development*. 13. Retrieved from http://ftp.mi.fu-berlin.de/reports/tr-b-02-13/stojanovic.pdf, last access, 14 July 2011

Torchiano M., Jaccheri L., Srensens C.-F., & Wang A. I.. (2002). *COTS Products Characterization.* Paper presented at the Conference on *Software Engineering and Knowledge Engineering (SEKE'02)*, Ischia, Italy.

Tuyet, L. A. (2004). *Fédération : une Architecture Logicielle pour la construction d'Applications Dirigé par les Modèles*, Université Josheph Fourrier, Grenoble.

Villalobos, J. (2003). Fédération de Composants : *Une Architecture Logicielle pour la Composition par Coordination. Unpublished Informatique*, Université Joseph Fourier, Grenoble.

Warmer, J. B., & Kleppe, A. G. (2003). *The object constraint language : getting your models ready for MDA (2nd ed.*). Boston: Addison-Wesley.

Wikipedia. (2006). *Base de données relationnelle* [Electronic Version] from http://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_relationnelle.

Yakimovich, D., Travassos, G. H., & Basili, V. R. (1999). *A Classification of Software Components Incompatibilities for COTS Integration*.