UNIVERSITÉ DU QUÉBEC À MONTRÉAL

SOA : LES IMPACTS SUR LE CYCLE DE VIE DU PROJET DE DÉVELOPPEMENT LOGICIEL

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE DE GESTION

PAR

RIM BEJAOUI

OCTOBRE 2008

REMERCIEMENTS

A mes très chers Elies et Elissa, j'offre le fruit de ce travail de recherche. Merci à Elies, pour avoir su m'épauler et m'encourager, en supportant et soulageant, au passage, les humeurs et angoisses liées à toute recherche. Merci à Elissa, qui par sa simple présence m'a donné le souffle recherché pour achever ce travail.

Merci à mes directeurs de recherche, Ivan Maffezzini et Elie Elia, qui m'ont généreusement apporté aide et soutien. Merci à Ivan Maffezzini pour ses remarques toujours pertinentes et enrichissantes. Merci à Elie Elia qui, par sa touche personnelle, a su parfaire ce travail.

TABLE DES MATIÈRES

LISTE DES FIGURES ix

LISTE DES TABLEAUX xi

LISTE DES ACRONYMES xiii

RÉSUMÉ xv

ABSTRACT xvii

CHAPITRE I INTRODUCTION 1

I.1 PROBLÉMATIQUE 2
I.2 OBJECTIFS 5
I.3 QUESTIONS DE RECHERCHE 7
I.4 MÉTHODE DE RECHERCHE 8
I.4.1 Étapes 9
I.4.2 Étude de cas 9
I.4.3 Chronologie 11
I.5 STRUCTURE 11

CHAPITRE II REVUE DE LA LITTÉRATURE 13

II.1 L'INGÉNIERIE LOGICIELLE AXÉE SUR LES SERVICES (SOSE) 14
II.1.1 Structure englobant SOSE 16
II.1.2 La vision de SOSE 18
II.1.3 Développement SOSE vs. Développement AOSE 20
II.2 L'ARCHITECTURE AXÉE SUR LES SERVICES (SOA) 21
II.2.1 Architecture d'application 21
II.2.2 Architecture d'entreprise 23

II.2.3 SOA : Concepts de base 24
II.2.4 Les éléments d'une SOA 31
II.2.5 Exemple d'une SOA d'entreprise 35
II.3 LES MÉTHODES DE DÉVELOPPEMENT TRADITIONNELLES 39
II.3.1 Aperçu sur les méthodes de développement traditionnelles 40
II.3.2 Unified Process (UP) 43
II.3.3 Rational Unified Process (RUP) 48
II.4 DÉVELOPPEMENT LOGICIEL AXÉ SUR LES SERVICES : DÉFIS ET PRATIQUES 56
II.4.1 Approches de développement axées sur les services 59
II.4.2 Les stratégies du développement logiciel axé sur les services 70
II.4.3 La gestion de projet axée sur les services 75
II.4.4 Les principes de la gouvernance SOA 78
II.4.5 Les rôles du développement axé sur les services 79
II.5 CONCLUSION 85

CHAPITRE III ACTIVITÉS, RÔLES ET TÂCHES SOA 87

III.1 TERMINOLOGIE 88
III.2 ACTIVITÉS SOA 91
III.2.1 Activités SOA de gestion de projet (Groupe A) 91
III.2.2 Activités SOA de pré-développement (Groupe B) 94
III.2.3 Activités SOA de développement (Groupe C) 95
III.2.4 Activités SOA de post-développement (Groupe D) 98
III.2.5 Activités SOA de support (Groupe E) 99
III.3 RÔLES ET TÂCHES SOA 103
III.4 CONCLUSION 109

CHAPITRE IV ÉTUDE DE CAS 111

IV.1 SÉLECTION DES CAS 112
IV.2 LE PROTOCOLE DE L'ÉTUDE DE CAS 112
IV.2.1 Aperçu du projet de l'étude de cas 113
IV.2.2 Méthodes de collecte des données 114
IV.2.3 Guide des entrevues 118
IV.2.4 Rapport de l'étude de cas 122

CHAPITRE V DISCUSSION 129

V.1 LITTÉRATURE 129
V.2 ÉTUDE DE CAS 130
V.2.1 Cas 1 130
V.2.2 Cas 2 131
V.3 RÉPONSE À LA QUESTION PRINCIPALE 132

CHAPITRE VI CONCLUSION 133

VI.1 SURVOL 133
VI.2 APPRENTISSAGE 134
VI.3 DIFFICULTÉS RENCONTRÉES 135
VI.4 LIMITES ET CONTRAINTES 136
VI.5 CONTRIBUTION ET TRAVAUX FUTURS 136

RÉFÉRENCES 137

APPENDICE A LE DOSSIER DE SANTÉ DU QUÉBEC (DSQ) 143

APPENDICE B GUIDE DES ENTRETIENS 145

APPENDICE C ÉVOLUTION DU PROJET DE RECHERCHE 189

LISTE DES FIGURES

Figure		page
Figure 1.1	Une initiative SOA implique l'élaboration de plusieurs projets pilotes (Source : Krafzig et al., 2004)	3
Figure 1.2	Relation entre *activité*, *tâche* et *rôle*	7
Figure 2.1	Structure englobant SOSE (Source : Stojanovic et al., 2005)	17
Figure 2.2	Vision de SOSE (Source : Stojanovic et al., 2005)	19
Figure 2.3	Les services d'entreprise : services d'affaires et services TI (Source : Huang et al., 2004)	26
Figure 2.4	Architecture de base des Services Web (Source : Newcomer et Lomow, 2004)	29
Figure 2.5	Éléments d'un système basé sur SOA (Source : Kajko-Mattsson et al., 2007)	31
Figure 2.6	Éléments d'une SOA (Source : Krogdahl et al., 2005)	34
Figure 2.7	Une EA traditionnelle avec des systèmes couplés par leurs interfaces propriétaires (Source : Pulier et Taylor, 2006)	36
Figure 2.8	Nouvelle EA non flexible, avec de nouveaux systèmes aussi fortement couplés qu'auparavant (Source : Pulier et Taylor, 2006)	37
Figure 2.9	Nouvelle configuration avec une SOA : chaque composant du système est exposé en tant que service Web (Source : Pulier et Taylor, 2006)	38
Figure 2.10	L'implémentation de changements dans une SOA est beaucoup plus simple que dans une EA traditionnelle (Source : Pulier et Taylor, 2006)	39
Figure 2.11	Les différentes méthodes de gestion de projet : approches lourdes vs. légères et linéaires vs. itératives (Source : Krafzig et al., 2005)	43

Figure 2.12	L'évolution d'UP : versions du produit dans les rectangles (Source : Jacobson et al., 2000)	44
Figure 2.13	Un processus de développement logiciel (Source : Jacobson et al., 2000)	44
Figure 2.14	Les quatre phases de RUP avec les niveaux d'effort dans chaque discipline à travers le CVP (Source : Kruchten, 2000)	51
Figure 2.15	SOAD et ses ingrédients: OOAD, BPM et EA (Zimmermann et al., 2004)	59
Figure 2.16	La méthode SOMA (Source : Arsangani, 2004)	60
Figure 2.17	Le cycle de vie des phases de M4SOD (Source : Yukyong et Hongran, 2006)	64
Figure 2.18	Processus de modélisation et de conception d'une SOA basé sur MDA (Rahmani et al., 2006)	66
Figure 2.19	Le modèle du processus SOUP durant le déploiement SOA initial (Mittal, 2006)	67
Figure 2.20	Les processus SOUP et XP superposés (Source : Mittal, 2005)	69
Figure 2.21	Différents niveaux de granularité d'artefacts logiciels durant le développement et durant l'exécution (Source : Krafzig et al., 2004)	77
Figure 2.22	Le conseil SOA utilise les contrats de service pour coordonner plusieurs projets ou sous-projets (Source : Krafzig et al., 2004)	78
Figure 2.23	Les rôles nécessaires au développement, à l'évolution et à la maintenance des systèmes fondés sur une SOA (Source : Kajko-Mattsson et al., 2007)	82
Figure 3.1	Terminologie utilisée	90
Figure 3.2	Activités SOA réparties selon les groupes d'activités de la norme IEEE/EIA std 1074	102
Figure 3.3	*Rôles* et *tâches* SOA	108

LISTE DES TABLEAUX

Tableau		page
Tableau 4.1	Spécificités des répondants pour les deux cas d'étude	116
Tableau 4.2	Documentation relative aux deux cas d'étude	118
Tableau 4.3	Objectif(s) et origine(s) des questions de Q1	119
Tableau 4.4	Objectif(s) et origine(s) des questions de Q2	120
Tableau 4.5	Objectif(s) et origine(s) des questions de Q3	121
Tableau 4.6	Objectif(s) et origine(s) des questions de Q5	122
Tableau 4.7	Synthèse des données collectées à partir du cas 1 et du cas 2	123
Tableau 4.8	Correspondance entre rôles de la littérature vs. rôles du cas 1	125
Tableau A.1	Évolution du travail de recherche	191

LISTE DES ACRONYMES

AOSE	Application Oriented Software Engineering
AE	Architecture d'entreprise
BPEL	Business Process Execution Language
CC SOA	Centre de compétences SOA
CIMOSA	CIM Open Systems Architecture
CRM	Customer Relationship Management
CVP	Cycle de vie du projet de développement logiciel
DSDM	Dynamic Software Development Method
DSÉ	Dossier de Santé Électronique
DSQ	Dossier de Santé du Québec
EA	Enterprise Architecture
ERP	Enterprise Resource Planning
ESB	Enterprise Services Bus
GC SOA	Government of Canada Service Oriented Architecture
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
iDSÉ	Infostructure du DSÉ
IEEE	Institute of Electrical and Electronics Engineers
ISAS	International Service Availability Symposium
IW-SOSE	International Workshop on Service-Oriented Software Engineering
J2EE	Java 2 Enterprise Edition
MSF	Microsoft Solution Framework
MSPI	Middard Service Programming Interface
OMT	Object Modeling Technique
RAD	Rapid Application Development

RCI	Retour sur le Capital Investi
RUP	Rational Unified Process
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAIC	International Symposium on Service-Oriented Applications, Integration and Collaboration
SOAP	Simple Object Access Protocol
SOSE	Service Oriented Software Engineering
SRC-Labo	Services Régionaux de Conservation et Laboratoires
TI	Technologies de l'information
TOGAF	The Open Group Architecture Framework
TPM	Transaction Processing Monitor
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
UP	Unified Process
W3C	World Wide Web Consortium
WCCIA	Workshop on Collaborative Computing, Integration, and Assurance
WOA	Web Oriented Architecture
WS	Web Service
WSDL	Web Services Description Language
XML	Extensible Markup Language
XP	Extreme Programming

RÉSUMÉ

Conceptualisée par le Gartner Group, leader mondial dans la fourniture de recherches et d'analyses dans l'industrie des technologies de l'information (TI), la notion d'architecture axée sur les services (notée SOA pour *Service Oriented Architecture*) est présentée comme le bon modèle pour le développement des applications d'entreprise, aujourd'hui et dans le futur (Marks et Bell, 2006). Selon une étude effectuée par le Gartner Group, plus de 75 % des projets d'entreprise, à partir de 2008, reposeront sur les principes de SOA. Ce chiffre donne une idée sur l'ampleur du « phénomène » SOA, qui suscite tant d'intérêt de la part des chercheurs. Cependant, d'après la littérature, le développement d'applications d'entreprise fondées sur SOA présente de nouveaux défis : apparition de nouvelles activités, de nouveaux rôles et de nouvelles tâches au niveau de l'entreprise ; nécessité d'avoir de nouveaux outils de support au développement ; incapacité des méthodes de développement traditionnelles à soutenir le développement axé sur les services ; etc. Des ajustements doivent donc être apportés au cycle de vie du projet de développement logiciel pour pouvoir s'adapter à cette nouvelle façon de construire les applications d'entreprise, et permettre, par conséquent, une gestion de projet plus efficace.

L'objectif de notre recherche est de savoir si la mise en œuvre d'une SOA par une organisation apporte des changements aux *rôles* et aux *tâches* du cycle de vie du projet de développement logiciel. Pour atteindre son objectif, notre recherche prend en considération les aspects spécifiques soulignés dans la littérature SOA, ainsi que les résultats d'une étude de cas exploratoire menée auprès de deux organisations ayant entamé une initiative SOA. La conclusion à laquelle nous sommes parvenus est que SOA introduit de nouvelles pratiques au niveau de l'organisation qui refaçonnent, lors des projets de développement logiciel, les *rôles* existants et les *tâches* qui leurs sont associées.

Mots-clés : architecture axée sur les services, développement axé sur les services, cycle de vie du projet de développement logiciel, rôles.

ABSTRACT

First coined by the Gartner Group, world leader in the supply of research and analyses in the information technology (IT) industry, the concept of service-oriented architecture (SOA) is considered like the right model for enterprise applications' development, today and in the future (Marks and Bell, 2006). According to a study carried out by Gartner Group, more than 75% of company's projects will have adopted SOA by the end of 2008. This gives an idea on the extent of the SOA "phenomenon" which causes such an amount of interest by researchers. However, according to the literature, the development of enterprise's IT and software based on SOA present new challenges: emergence of new organisational activities, roles, and tasks; need for new tools supporting the development work; incapacity of traditional development methods to support SOA development; etc. Thus, there is a need to adjust the software development project lifecycle to this new way of building enterprise's applications for a more effective project management.

The objective of our research is to determine if the adoption of SOA by an organization brings changes to the *roles* and *tasks* involved in the software development project lifecycle. To achieve its goal, our research takes into account specific aspects in SOA literature, as well as the results of an exploratory case study undertaken in two organizations having started an SOA initiative. Our conclusion is that SOA adoption introduces new organisational practices which reshape *roles* and *tasks* characterising software development projects.

Keywords: Service Oriented Architecture, Service Oriented development, software project lifecycle, roles.

CHAPITRE I

INTRODUCTION

Par le passé, les applications d'entreprise étaient souvent réalisées en construisant des applications qui satisfaisaient certaines exigences spécifiques et, ensuite, il fallait faire dialoguer ces applications – dialogue qui souvent obligeait à montrer certaines structures internes. Tout cela créait des problèmes d'interfaces qui poussaient les équipes de projets à gérer diverses manières d'invoquer les interfaces et de relier les applications entre elles. Le processus de liaison entre les modules logiciels ou les sous-systèmes, représentait alors le plus grand problème de développement et de maintenance pour les entreprises.

Toute l'histoire de la programmation est fondée sur l'importance des modules en tant que boîtes fermées et l'approche objet, par exemple, n'est, d'un certain point de vue, que la tentative de montrer les services qu'une classe offre. Si on passe du micro-niveau (ou conception détaillée) au niveau de la conception globale, par analogie on veut des boîtes fermées. Le développement basé sur les composants a d'ailleurs constitué une tentative de plus haut niveau pour assembler des composants logiciels afin de former des logiciels conçus pour les systèmes d'entreprise.

Définis par le *World Wide Web Consortium* (W3C), les services Web sont des composants applicatifs accessibles sur le Web, qui reposent sur un ensemble de normes dont le but est de transformer les problématiques d'intégration entre systèmes hétérogènes en objectif d'interopérabilité. Tel que défini par W3C, un service Web « *est un système logiciel conçu pour soutenir une interaction intéropérable machine-à-machine à travers un réseau. Il possède une interface décrite en un format traitable par la machine ... les autres systèmes*

interagissent avec le service Web, tel que défini par sa description ». L'arrivée des services Web et leur adoption par les distributeurs logiciels a été suivie par l'émergence du concept d'« axé sur les services » pour les entreprises à travers SOA. Dans le contexte de l'entreprise, SOA n'est pas juste une technologie. Il ne s'agit pas non plus d'un produit ou d'une solution logicielle, comme veulent le faire croire les fournisseurs de solutions logicielles (Marks et Bell, 2006).

I.1 Problématique

SOA est un acronyme qui s'est répondu partout depuis l'année 2002. On le retrouve comme mot clé dans un grand nombre d'articles de journaux scientifiques, comme slogan utilisé par les fournisseurs de solutions logicielles (ex. « *Stop Talking. Start Doing. The Smart SOA™ Approach* » d'IBM), comme titre pour des dizaines de livres, comme avantage concurrentiel mis en avant par les entreprises actuelles et comme approche choisie par de grands projets gouvernementaux, dont le modèle de référence GC SOA (*Government of Canada SOA*) qui guide l'utilisation et l'adoption de SOA par le gouvernement du Canada.

En fait, l'acronyme SOA couvre deux domaines : l'architecture d'entreprise et l'architecture d'application. SOA est fondée sur la construction de services réutilisables et interopérables, neutres par rapport à la plateforme de communication et qui correspondent aux processus d'affaires de l'entreprise. Elle peut être vue comme « *une approche utilisée pour normaliser l'architecture logicielle à travers l'entreprise* » (Erl, 2005). L'idée sous-jacente est de mettre au centre l'architecture, car elle risque d'avoir une durée de vie plus longue que les exigences et les besoins qui sont au fondement des applications. Les bénéfices promis par SOA pour l'organisation incluent (Newcomer et Lomow, 2004) :

- une augmentation de l'agilité de l'entreprise,
- une réduction des coûts d'intégration des technologies utilisées,
- une amélioration du rendement du capital investi,
- un meilleur alignement des affaires, etc.

L'adoption de SOA par une entreprise constitue une grande initiative qui peut s'étendre sur plusieurs années. Cette initiative, selon certains auteurs, implique l'élaboration de plusieurs projets pilotes qui ont pour objectif de développer l'infrastructure SOA (Bloomberg et Schmelzer, 2006 ; Krafzig et al., 2004). Tel qu'illustré par la figure suivante, les projets individuels SOA (projets pilotes) et le développement de l'infrastructure SOA sont très dépendants.

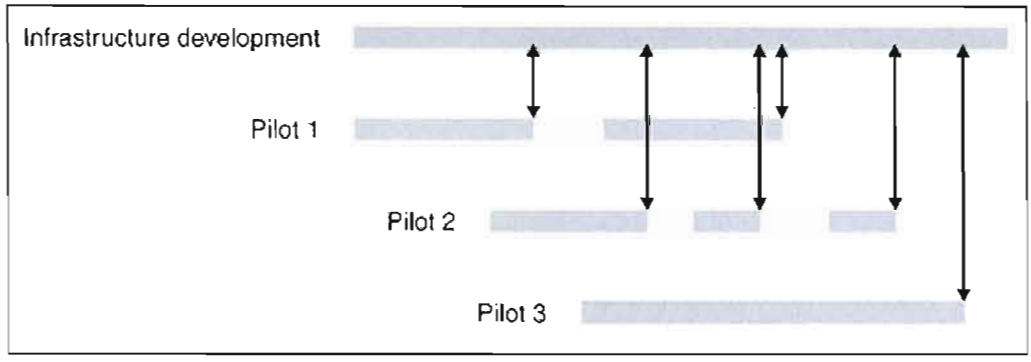


Figure 1.1 Une initiative SOA implique l'élaboration de plusieurs projets pilotes (Source : Krafzig et al., 2004)[1]

Au cours d'un projet SOA, les membres de l'équipe de projet (architectes, analystes d'affaires, développeurs, etc.) sont guidés par les principes de l'axé sur les services pour la définition des services et la construction des applications SOA (Newcomer et Lomow, 2004). Les principes de l'axé sur les services les plus importants sont les suivants (Erl, 2005) :

1. Les services sont réutilisables : ils sont conçus de façon à ce qu'ils puissent être réutilisés ultérieurement.

2. Les services interagissent à travers un contrat formel : pour pouvoir interagir, les services utilisent un contrat formel qui décrit chaque service et définit les termes de l'échange d'information.

3. Les services sont faiblement couplés : ils sont conçus pour interagir avec le minimum d'interdépendances.

4. Les services font abstraction de la logique d'affaires : la seule partie visible du service est la partie exposée par le contrat de service (appelée interface publique). La logique sous-jacente du service est invisible et hors de portée du demandeur du service.

5. Les services sont composables : les services peuvent être associés entre eux pour réaliser une fonction particulière (ex. processus d'affaire). Ceci permet la représentation de la logique d'affaires selon plusieurs niveaux de granularité et facilite la réutilisation et la création de plusieurs niveaux d'abstraction.

6. Les services sont autonomes : les tâches accomplies par un service possèdent des limites. Le service possède un contrôle suivant cette limite et ne dépend pas d'autres services pour accomplir sa tâche.

Lorsque dans un domaine de l'entreprise on applique les principes de l'axé sur les services, on parle d'environnement axé sur les services. L'adoption des principes de l'axé sur les services est un processus complexe. Tel que déjà énoncé, selon certains auteurs, ce processus requiert plusieurs projets pilotes et peut s'étendre sur plusieurs années. Les projets SOA ressemblent, en surface, aux projets de développement d'applications distribuées (Erl, 2005). Cependant, afin de construire et de positionner les services d'une façon convenable dans une solution SOA, les cycles des projets traditionnels nécessitent quelques ajustements. Ces ajustements se reflètent dans les différentes approches de développement axées sur les services proposées par quelques récentes recherches (Zimmermann et al., 2004 ; Arsanjani, 2004 ; Ivanyukovich et al., 2005 ; Yukyong et Hongran, 2006 ; Rahmani et al., 2006 ; Mittal, 2006). Certains de ces travaux ont tenté d'adapter des méthodes de développement existantes (ex. RUP et XP) pour qu'elles puissent répondre aux caractéristiques spécifiques des

[1] Toutes les figures extraites des documents en anglais ont été laissées telles qu'elles. C'est-à-dire qu'elles n'ont pas été traduites.

environnements axés sur les services. D'autres ont essayé de formaliser le processus de développement SOA en mettant l'accent sur certaines phases spécifiques du processus (ex. identification des services, modélisation des services ou réalisation des services).

Certaines autres recherches — mais pas des plus nombreuses — se sont intéressées aux changements encourus à la structure organisationnelle, suite à l'adoption de SOA. Bierberstein et al. (2005) parlent de la nécessité de reconcevoir l'organisation en incluant de nouveaux comportements culturels et individuels. Mis à part les rôles existants. Kajko-Mattsson et al. (2007) ont proposé un ensemble initial de rôles pour le développement, l'évolution, la maintenance et le support, qu'ils ont catégorisés en quatre groupes :

1. support SOA ;
2. stratégie et gouvernance SOA ;
3. conception et gestion de la qualité SOA ;
4. développement et évolution SOA.

Actuellement, tous les travaux dans la littérature SOA que nous avons consultés considèrent que l'adoption de SOA n'est pas triviale, qu'elle présente beaucoup de défis et qu'elle nécessite des changements au sein de l'organisation. La question est de savoir, jusqu'à quel point le choix de SOA, peut influencer la façon avec laquelle l'organisation structure et gère ses projets de développement logiciel.

I.2 Objectifs

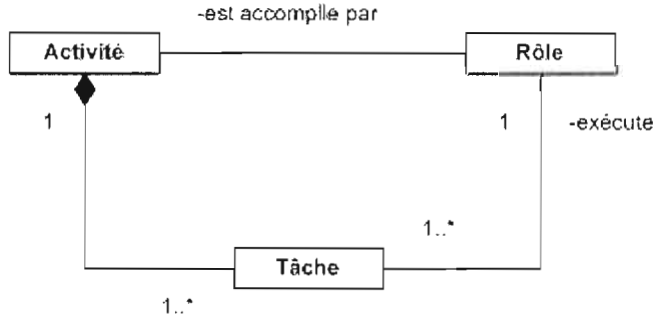
D'une façon générale, un projet logiciel est défini par l'ensemble des activités techniques et managériales, nécessaires à la satisfaction des termes et des conditions d'un accord de projet qui spécifie, entre autres, l'étendue, les objectifs, le budget et l'échéancier du

projet (IEEE 1058-1998). Habituellement, lors de la réalisation d'un projet, les organisations divisent ce dernier en plusieurs phases afin d'améliorer le contrôle de gestion[2] et de fournir un lien entre les différentes opérations (IEEE 1490, 2003). L'ensemble des phases constitue le cycle de vie du projet de développement logiciel (CVP).

Le CVP définit généralement (IEEE 1490-2003) :

- Le travail technique à effectuer durant chaque phase (ex. est-ce que le travail architectural fait partie de la phase de définition ou de la phase d'exécution ?)
- Les individus impliqués durant chaque phase (ex. quels sont les individus qui doivent être impliqués durant la phase de conception ?)

En d'autres termes, le CVP définit les *activités* et les *rôles* nécessaires à l'accomplissement de chaque phase. Nous considérons qu'*activités* et *rôles* sont très corrélés par la notion de *tâche* (voir figure ci-dessous). En effet, une *activité* est définie par un ensemble de *tâches* coordonnées ; d'autre part, un *rôle* est défini par l'ensemble des *tâches* qui lui sont associées. Ainsi, une façon d'identifier les *activités* à accomplir durant le CVP serait de d'identifier les *rôles* impliqués et l'ensemble des *tâches* qui leurs sont associées.



[2] D'une façon générale, le contrôle de gestion est un processus permanent qui permet à une organisation de s'assurer que les ressources qui lui sont confiées sont obtenues et utilisées d'une façon efficace (OQLF, 2007). Dans le cas d'un projet logiciel, le contrôle de gestion devrait être imposé via les tests, les révisions, les audits, la vérification, la validation, etc. (IEEE 12207.0, 1996).

Figure 1.2 Relation entre *activité*, *tâche* et *rôle*

La description du CVP peut être générale ou détaillée ; les approches détaillées, quand elles sont généralisées (ou normalisées), sont qualifiées de « méthodes de développement ». Les approches utilisées par les organisations sont très variées. Même si plusieurs cycles de vie de projet possèdent des phases avec des noms et des activités similaires, peu sont identiques (IEEE std. 1490-2003). Certains cycles de vie possèdent quatre à cinq phases (ex. modèle en spirale, processus unifié), alors que d'autres en possèdent neuf (ex. modèle en V).

Le CVP a longtemps été analysé. Au fur et à mesure de l'apparition de nouvelles pratiques de développement, il a été adapté. D'après la littérature SOA, la tendance actuelle vers le développement de systèmes fondés sur SOA nécessite quelques ajustements au CVP. Cependant, on ne présente pas de cas où l'on définit, en détail, les changements qui y sont apportés.

L'objectif de notre recherche est de savoir si le passage vers un environnement axé sur les services pour une organisation demande des changements aux *rôles* et aux *tâches* du CVP. Pour atteindre notre objectif, nous avons pris en considération les aspects spécifiques soulignés dans la littérature SOA, ainsi que les résultats d'une étude de cas par entrevues menée auprès de deux organisations ayant entamé une initiative SOA.

I.3 Questions de recherche

Notre question de recherche principale est la suivante :

« Est-ce que le passage vers un environnement axé sur les services pour une organisation demande des changements aux rôles et aux tâches du CVP ? ».

Pour faciliter la réponse à cette question nous allons répondre aux questions secondaires suivantes :

- Dans la littérature SOA, existe-t-il une conceptualisation cohérente des *rôles* impliqués dans le CVP et des *tâches* qui leurs sont associées ?

- Quels sont les changements aux *rôles* ?

- Quels sont les changements aux *tâches* associées aux *rôles* ?

I.4 Méthode de recherche

Les méthodes de recherche qualitatives sont les plus adéquates pour les études qui cherchent à décrire un phénomène nouveau ou innovateur afin d'y atteindre une compréhension plus approfondie (Denzin et Lincoln, 2003). La recherche qualitative privilégie les données non numériques pour en faire surgir des significations (Morse, 1994). Ce type de recherche est pertinent pour explorer en profondeur les problématiques moins connues ou plus contemporaines, comme c'est le cas des changements, en termes de *rôles* et de *tâches*, encourus au CVP dans un environnement axé sur les services.

Il existe plusieurs méthodes pour conduire une recherche qualitative : la recherche-action, l'étude de cas, l'enquête par questionnaires, l'analyse de contenu, etc. En raison de :

1. La nouveauté relative du phénomène SOA,

2. L'inexistence d'une théorie déjà éprouvée sur la question des *rôles* et des *tâches* caractérisant le CVP dans un environnement axé sur les services,

3. La parution, durant ces quatre dernières, d'un grand nombre de publications provenant de différentes sources concernant SOA,

Il nous a paru judicieux de regarder cette recherche dans une logique visant à la compréhension de ce phénomène, en effectuant une étude exploratoire combinant une revue de la littérature existante avec une étude de cas par entrevues. Bien évidemment, cette démarche s'inscrit dans une logique de recherche qualitative.

I.4.1 Étapes

Pour atteindre notre objectif de recherche, qui est de savoir si le passage vers un environnement axé sur les services pour une organisation demande des changements aux *rôles* et aux *tâches* du CVP, nous avons :

1. Effectué une analyse approfondie des principaux articles et livres publiés durant les cinq dernières années sur SOA ;
2. Étudié quelques références (surtout des normes) sur la gestion de projet et les méthodes en génie logiciel ;
3. Synthétisé les connaissances tirées de la littérature qui sont en relation avec notre objectif de recherche ;
4. Conduit une étude de cas exploratoire par entrevues semi-structurées (incluant deux cas d'étude) portant sur les *rôles* et les *tâches* caractérisant le CVP dans un environnement axé sur les services. Pour chaque cas, nous avons ;
    a. Bâtit le protocole d'entrevues et définit les questions d'investigation ;
    b. Conduit l'étude ;
    c. Synthétisé les connaissances tirées du cas et confronté ces dernières avec la littérature ;
5. Présenté une conclusion ;

I.4.2 Étude de cas

I.4.2.1 Le choix des cas

À présent, beaucoup d'entreprises sont en train d'adopter SOA (entreprises manufacturières, institutions financières ou entreprises spécialisées dans un tout autre domaine). Tel qu'expliqué plus en détail dans la section IV.1, la sélection de nos deux cas d'étude a dépendu de trois paramètres :

- Les organisations en question ont déjà entamé une initiative SOA.
- Elles peuvent nous accorder des ressources pour mener notre investigation (plusieurs répondants, possibilité d'entrevues, sources de preuves qualitatives, etc.).
- Il existe une grande différence entre les environnements axés sur les services des deux cas choisis : l'un est centralisé et l'autre est décentralisé.

I.4.2.2 Les répondants

Les intervenants sollicités pour l'étude de cas sont au nombre de cinq (5), dont deux (2) gestionnaires de projet (R1 et R5), un (1) architecte d'intégration (R2) et deux (2) architectes de solutions (R3 et R4). R1 a constitué le répondant unique pour le premier cas d'étude. R2, R3, R4 et R5 sont les répondants sollicités pour notre deuxième cas d'étude. Malheureusement, pour les raisons qui seront expliquées dans la section IV.2.2, la deuxième étude de cas a été interrompue avant que les entretiens avec R4 et R5 n'aient eu lieu.

Les raisons du choix des répondants, les détails les concernant ces derniers (rôles, expérience, nombre de rencontres, etc.), ainsi que le déroulement des entrevues sont présentés dans la section IV.2.2.

I.4.2.3 Construction de l'outil et du protocole des entrevues

Vu qu'il s'agit d'une étude exploratoire, les entrevues accomplies étaient semi-structurées. En effet, les questions d'investigation –ayant pour objectif d'identifier les *rôles* et *tâches* du CVP dans un environnement axé sur les services – ont été élaborées de façon à laisser de la place aux commentaires du répondant et à la description. La construction de l'outil et du protocole des entrevues est expliquée plus en détail dans la section IV.2

I.4.3 Chronologie

Notre projet de recherche a été entamé en février 2007 et a été finalisé en juin 2008, avec une période d'inactivité de 3 mois. Ce qui compte une durée totale de 14 mois de travail pour l'accomplissement des étapes présentées dans la section I.4.1. Au cours du temps, et pour les raisons qui seront expliquées dans la section « Limites et contraintes » (section VI.4), notre recherche a fait l'objet de plusieurs changements. Le passage d'une version à une autre (objectif, questions de recherche, raisons du changement et travail effectué sur chaque version) est expliqué dans le tableau de l'appendice C.

I.5 Structure

Le présent chapitre (chapitre 1) ayant défini la problématique, les objectifs et les questions de recherche de notre projet de recherche et décrit la méthode que nous avons adoptée pour atteindre les objectifs fixés, le reste du document s'articule autour de cinq chapitres. Le deuxième chapitre (chapitre 2) présente une revue de la littérature des domaines de recherche concernés par ce travail. Le troisième chapitre (chapitre 3) effectue une synthèse des éléments de la littérature qui sont en relation avec notre objectif de recherche. Le quatrième chapitre (chapitre 4) décrit notre étude de cas. Le cinquième chapitre (chapitre5) présente une discussion concernant les résultats auxquels nous sommes arrivés. Et pour finir, le dernier chapitre de notre mémoire (chapitre 6) constitue une conclusion dans laquelle nous effectuons un survol de notre recherche et nous présentons notre apprentissage, les difficultés rencontrées, les limites et les contraintes, la contribution de notre recherche, ainsi que les possibilités de travaux futurs.

CHAPITRE II

REVUE DE LA LITTÉRATURE

La revue de la littérature commence (section II.1) par définir le nouveau paradigme de développement logiciel SOSE (*Service Oriented Software Engineering)*, qui est apparu en réponse aux problèmes reliés à l'ingénierie logicielle axée sur les applications, et qui se base sur SOA pour le développement de solutions logicielles axées sur les services. Pour définir le paradigme SOSE, nous allons présenter la structure (ou l'environnement) qui englobe SOSE, définir la vision de SOSE selon le point de vue des entités les plus importantes de la structure présentée (demandeurs de services, fournisseurs de services, intermédiaires de services et organisations de services) et établir une comparaison entre le développement SOSE et le développement axé sur les applications.

La deuxième section de ce chapitre (section II.2) est consacrée à SOA. Tel que déjà énoncé dans la problématique (section I.1), SOA couvre deux domaines : l'architecture d'entreprise et l'architecture d'applications. La section II.2 commence par définir ces deux notions avant de présenter les concepts de base de SOA (les services, les principes de l'axé sur les services et la plateforme des services Web), les éléments d'une SOA et un exemple illustratif d'une SOA d'entreprise.

La troisième section de la revue de la littérature (section II.3) s'intéresse aux méthodes de développement traditionnelles. Elle donne d'abord un aperçu sur les méthodes de développement traditionnelles existantes (modèle en cascade, *Crystal*, XP, dX, ...) et s'intéresse ensuite, plus particulièrement, à *Unified Process* (UP) et à son dernier produit

RUP (*Rational Unified Process*), vu que, dans la littérature, plusieurs recherches (Krafzig et al., 2005 ; Mittal, 2006 ; Stojanovic et Dahanayake, 2005) considèrent que les approches les plus appropriées pour le développement axé sur les services sont celles qui sont fondées sur le développement itératif, faisant de UP un bon candidat.

La quatrième section de la revue de la littérature (section II.4) présente les défis et les pratiques du développement logiciel axé sur les services. Cette section est importante pour l'atteinte de notre objectif de recherche, car le développement des questions d'investigation de notre étude de cas est principalement fondé sur les pratiques du développement logiciel axé sur les services citées dans la littérature. La section II.4 commence par présenter les approches et les stratégies du développement axé sur les services (sections II.4.1 et II.4.2). Elle s'intéresse ensuite aux pratiques de la gestion de projet axée sur les services (section II.4.3) et aux principes de la gouvernance SOA (section II.4.4). Elle finit par présenter les rôles dans le développement logiciel axé sur les services (section II.4.5).

Enfin, la revue de la littérature se termine par une conclusion (section II.5).

II.I L'ingénierie logicielle axée sur les services (SOSE)

Dans le marché du logiciel, les besoins de l'utilisateur, les technologies, l'expertise, les programmeurs et les partenaires sont hautement dynamiques et changeants. En effet, les besoins des utilisateurs changent, les employés quittent leurs emplois, de nouvelles technologies émergent et des relations entre partenaires se mettent en place. Il est désormais indispensable pour les fournisseurs de solutions logicielles d'utiliser des méthodes de développement agiles et flexibles qui favorisent l'adaptation aux changements (Z. Stojanovic et al., 2005).

Actuellement, la plupart des applications logicielles commerciales sont vendues sur une base de propriété (K. Bennett et al., 2000). Ainsi, le client achète le code exécutable accompagné d'une licence d'utilisation. Le fournisseur de l'application a pour responsabilité

de fournir les mises à jour de son produit, et n'importe quelle tentative de modification de l'application risque de compromettre la garantie, ainsi que le support fourni pour l'application. Cette forme de commercialisation, qui suit une politique économique de l'offre[3], est celle caractérisant le développement logiciel axé sur les applications (noté AOSE pour *Application Oriented Software Engineering*) (Z. Stojanovic et al., 2005). Mis à part ses frontières d'affaires rigides et sa politique de fourniture de logiciel fondée sur la propriété, le développement AOSE place la technologie au premier plan, nécessite de très longs délais et est réservé aux organisations de grande taille (Stojanovic et al., 2005).

En réponse à ces problèmes, des études ont été accomplies partout dans le monde, dans le but de développer de nouvelles approches de développement logiciel. Le concept clé qui a émergé est que le logiciel est un service plutôt qu'une application (Papazoglou et al., 2003 ; Stojanovic et al., 2005). Cette vision a permis d'écarter le modèle AOSE pour laisser la place au modèle SOSE, fondé sur la fourniture de solutions logicielles selon une politique économique de la demande. La politique économique de la demande a été développée et appliquée plus tôt dans l'industrie manufacturière. Il s'agit d'un concept développé par Toyota, qui a commencé dans les années 1980 à révolutionner l'industrie de l'automobile avec son approche du « *Lean Manufacturing* » ou « production à valeur ajoutée », avec l'intention d'éliminer le gaspillage (matière, temps, etc.), de rationaliser la chaîne de valeurs (même entre entreprises), de produire sur demande (juste à temps) et de focaliser sur les personnes qui créent la valeur ajoutée (Krogdahl et al., 2005). Les nœuds dans la chaîne, qui sont des compagnies de petite taille, travaillent ensemble dans une approche « requête de service/réponse rapide ». Ceci a eu pour effet d'accélérer le temps de mise en marché de nouveaux produits, de réduire le risque d'investissement et de répondre rapidement aux changements du marché (Stojanovic et al., 2005). D'une façon similaire, avec la maturation du développement axé sur les composants et l'apparition des standards de service, SOSE émerge comme un nouveau paradigme qui reflète une transition importante dans l'industrie du logiciel pour le monde des affaires. L'objectif de l'industrie n'est plus de fabriquer des produits matériels et logiciels, mais de fournir les services nécessaires aux utilisateurs (Tsai

[3] Il s'agit d'une politique économique qui favorise les producteurs plutôt que les consommateurs.

et al., 2007), à travers des systèmes de développement agiles, qui facilitent la conception, la mise en œuvre, les tests, l'évaluation et l'accès à des services à travers le Web (Stojanovic et al., 2005).

Une grande attention a été portée à SOSE, que ce soit dans les compagnies de logiciels, que dans les groupes de recherche académiques. Le succès connu dans le milieu de la recherche académique est dû en grande partie aux séries de conférences et d'ateliers tenus durant les cinq dernières années, incluant *IEEE International Symposium on Service-Oriented System Engineering*, *International Service Availability Symposium* (ISAS), *International Workshop on Collaborative Computing, Integration, and Assurance* (WCCIA), *International Symposium on Service-Oriented Applications, Integration and Collaboration* (SOAIC) et *International Workshop on Service-Oriented Software Engineering* (IW-SOSE). Les grandes compagnies telles que BEA, Cisco, Hewlett Packard, IBM, Intel, Juniper, Microsoft, Oracle, SAP et Sun Microsystems sont passées au paradigme SOSE. De même, les grandes agences gouvernementales tel que le ministère de la défense des États-Unis et le Gouvernement du Canada, avec son initiative appelée « Stratégie d'architecture axée sur les services » ou, en anglais, GC SOA *Strategy* (*Government of Canada Service Oriented Architecture Strategy*), ont adopté le paradigme SOSE.

II.1.1 Structure englobant SOSE

Stojanovic et al. (2005) proposent une description détaillée de l'environnement englobant SOSE. Le modèle est illustré dans la figure ci-dessous. Dans ce qui suit, nous allons expliquer chacune des entités représentées dans ce modèle.

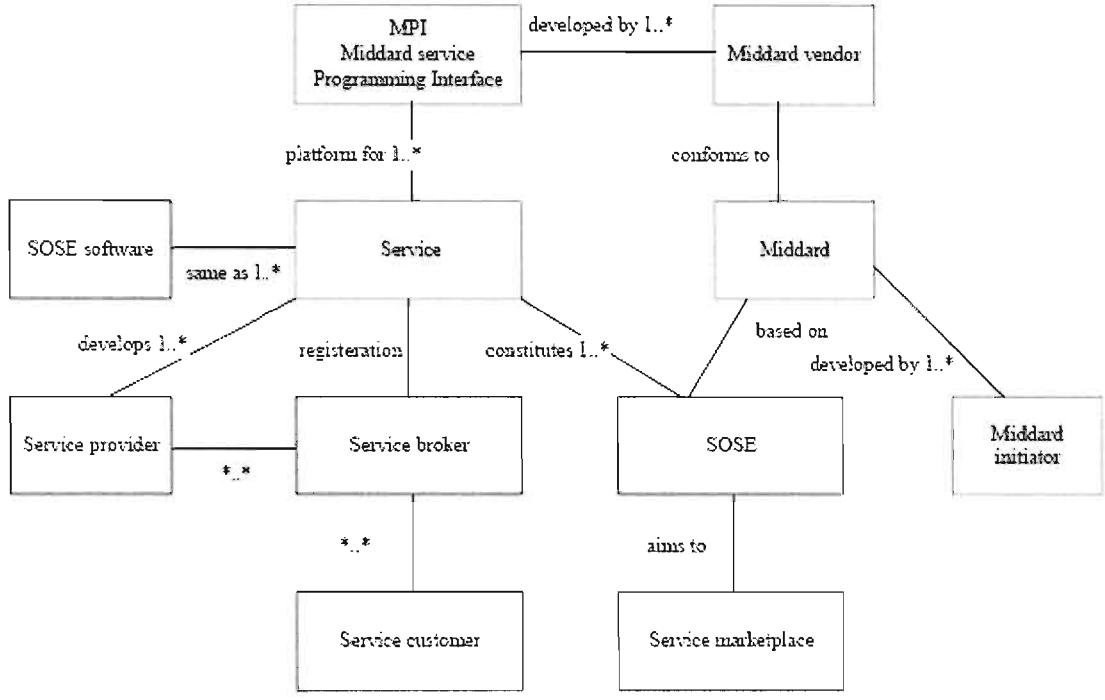


Figure 2.1 Structure englobant SOSE (Source : Stojanovic et al., 2005)

SOSE décompose une application en plusieurs services qui sont développés et configurés indépendamment et séparément conformément à un *middard*. Un *middard* est un accord (agrément) documenté, contenant les spécifications (normes, méthodes, règles et directives) à utiliser de manière régulière pour le développement, la description et la gestion de services partageables par les parties prenantes du logiciel. Un *middard* est généralement initié, développé et géré par un initiateur de *middard*. TPM, HTML et XML sont considérés comme des exemples de notations pour *middard*.

Un fournisseur de *middard* est n'importe quel intervenant qui met en œuvre un ou une partie d'un *middard* sous la forme d'une interface de programmation MSPI (*Middard Service Programming Interface*) destinée aux fournisseurs de service, afin de leur faciliter le développement de service. Un *middard* peut avoir plusieurs fournisseurs de *middard*. Un

fournisseur de service développe et effectue la maintenance de services fondés sur un *middard*.

Les intermédiaires de service sont des entités de jonction entre les demandeurs de service et les fournisseurs de service, qui facilitent la circulation des services sur le marché. Les demandeurs de service, appelés aussi utilisateurs de service, sont ceux qui acquièrent, sélectionnent et utilisent les services.

Un contrat de service contient les termes convenus entre le demandeur et le fournisseur de service pour la fourniture du service. Un profil de service contient les valeurs acceptables des termes et des politiques du contrat pour gouverner la manière avec laquelle ces valeurs seront négociées. Un profil de client contient les systèmes légaux acceptés pour les contrats, la performance de service minimale requise, le coût maximum accepté et le pourcentage du coût moyen du marché pour lequel une négociation est possible. Un profil de fournisseur contient les systèmes légaux acceptables pour les contrats, les niveaux de performance garantis et le coût de fourniture du service.

II.1.2 La vision de SOSE

Stojanovic et al. (2005) illustrent la vision de SOSE du point de vue des demandeurs (ou consommateurs) de service, des fournisseurs de service, des intermédiaires de service et des organisations de service. La vision de SOSE est décrite dans la figure suivante :

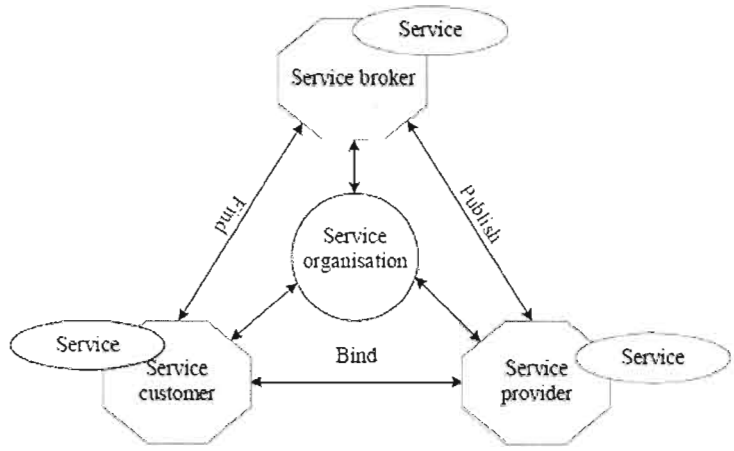


Figure 2.2 Vision de SOSE (Source : Stojanovic et al., 2005)

Point de vue des clients : Le logiciel est un service, non une application. Les services sont facilement personnalisables par les clients eux-mêmes. Les clients obtiennent les droits d'exploitation (*copyright*) des services à travers une négociation de contrat. Dans ce sens, il est nécessaire pour les clients de changer leur attitude de propriété du logiciel par une attitude d'utilisation du logiciel.

Point de vue des fournisseurs : Le logiciel est une unité de service qui peut être utilisée de façon indépendante mais qui sera fort probablement plus utilisée en tant que module d'un système intégré. Le fournisseur est responsable de la gestion du module durant son cycle de vie (ex. conception, mise en œuvre et maintenance du module). Les fournisseurs de service obtiennent les besoins du service à partir du marché des services, et non directement à partir des clients. Suite à la mise en œuvre des services, les fournisseurs soumettent l'information détaillée des services (ex. fonctionnalités, caractéristiques non fonctionnelles et informations d'affaires) aux intermédiaires de services, plutôt qu'aux clients.

Point de vue des intermédiaires : Le logiciel est un service à enregistrer et à vendre. Les intermédiaires fournissent certains moyens pour répondre aux requêtes des fournisseurs

de service et des demandeurs de service incluant le classement et la sélection des services candidats, la visualisation des services et de leur qualité (ou degré) d'ajustement par rapport aux besoins et le support automatisé pour la certification. Les intermédiaires partagent les bénéfices de la vente des services avec les fournisseurs de service.

Point de vue des organisations de service : Contrairement au développement logiciel axé sur les applications, fondé sur des frontières d'affaires rigides, sur une politique de fourniture de logiciel basée sur la propriété et consommatrice de temps, les organisations qui appliquent SOSE prennent en considération les activités d'achat de service, d'alliance entre organisations et de sélection de *middard* durant la phase initiale du développement logiciel. Les « organisations SOSE » focalisent principalement sur les affaires des clients, plutôt que sur les technologies de support durant la phase de mise en œuvre du logiciel. Elles mettent l'accent sur l'amélioration de leurs compétences clé et fournissent plus d'effort dans la qualité des services durant la phase de maintenance du logiciel. Dans ce sens, les organisations logicielles seront plus centrées sur le client, de plus petite échelle, plus agiles et plus compétitives.

II.1.3 Développement SOSE vs. Développement AOSE

Contrairement au développement logiciel axé sur les applications, SOSE possède les avantages suivants (Stojanovic et al., 2005) :

- *Une solution rapide :* SOSE a pour objectif de fournir à l'utilisateur une solution, plutôt qu'un produit. C'est-à-dire que SOSE accentue l'analyse des besoins spécifiques des utilisateurs et du marché des services. Une fois identifiés, ces besoins seront transmis aux fournisseurs de service, qui pourront y répondre rapidement.

- *Limiter l'insuffisance des ressources :* Le manque d'argent et de personnel constituent l'une des causes d'échec lors du développement de systèmes logiciels. SOSE permet au développement logiciel d'explorer et d'utiliser les services

disponibles sur le marché. Par conséquent, SOSE rend la coopération d'une organisation avec un partenaire compétent[4], pour la fourniture d'une solution rapide aux utilisateurs, plus avantageuse en termes de coûts et de temps.

- *Faciliter le développement logiciel complexe et à grande échelle :* Un système complexe implique, généralement, des participants provenant de différents domaines (ex. un logiciel d'ordinateur de bord d'un avion). De nombreux problèmes peuvent survenir lors du développement et de la maintenance d'un système logiciel complexe en l'absence d'une norme de communication unifié. SOSE fournit cette norme unifiée, une sorte de *middard*, qui rend l'intégration de différents domaines plus facile.

- *Productivité :* Les bénéfices de productivité réels de SOSE sont obtenus en permettant le développement parallèle des services. La construction de systèmes par des composants normalisés a déjà été adoptée par l'industrie automobile, électronique, de construction et tant d'autres industries. Son adoption par l'industrie du logiciel à travers le nouveau paradigme[5] SOSE mènera sans doute aussi vers une industrie plus productive et plus profitable.

II.2 L'architecture axée sur les services (SOA)

SOA couvre deux domaines : l'architecture d'application et l'architecture d'entreprise. Nous allons commencer par définir ces deux concepts avant d'introduire SOA.

II.2.1 Architecture d'application

Les applications d'entreprise constituent un genre particulier d'applications logicielles, car elles sont à la base des dépendances entre les départements et des relations d'affaires externes de l'entreprise. Comme son nom l'indique, une application d'entreprise est

---

[4] Ce qui a une certaine affinité (sinon une affinité certaine) avec les logiciels *Open Source*.

fortement couplée avec l'organisation interne, les processus et le modèle d'affaires de l'entreprise (Krafzig et al., 2004). Par analogie, « *une architecture d'application pour une équipe de développement d'application est ce qu'est un plan pour une équipe de construction* » (Erl, 2005). Elle permet de construire une application conformément à un nombre important de besoins différents. Ces besoins sont en relation avec les changements permanents du marché, l'organisation de l'entreprise et ses objectifs d'affaires. C'est cette implication dans tous les aspects de l'entreprise qui fait que les applications d'entreprise sont hautement complexes (Krafzig et al., 2004).

Les organisations documentent différents niveaux de l'architecture d'application. Certaines s'en tiennent à des représentations de haut niveau fournissant des représentations abstraites physiques et logiques des modèles techniques de leurs applications ; d'autres incluent plus de détails :

- des modèles de données,
- des diagrammes de flux de communication[6],
- les besoins en sécurité de l'application et
- des aspects d'infrastructure (T. Erl, 2005).

Une organisation peut avoir plusieurs architectures d'applications, si elle est en présence de plusieurs environnements techniques (ex. .NET et J2EE). Chaque architecture d'application reflète les besoins immédiats de la solution, ainsi que les objectifs de la stratégie TI à long terme. Pour cette raison lorsqu'une organisation possède plusieurs architectures d'applications, ces dernières sont alignées avec une architecture d'entreprise qui les gouverne (Erl, 2005).

---

[5] Relativement nouveau car apparenté au paradigme *Component Based* SE.
[6] Concept à notre avis assez flou dans ce contexte.

II.2.2 Architecture d'entreprise

Les environnements TI des grandes entreprises sont caractérisés par des architectures d'applications disparates qui coexistent et qui, parfois, s'intègrent. Par conséquent, des spécifications de haut niveau sont créées pour fournir une vue d'ensemble de toutes les formes d'hétérogénéité qui existent dans l'entreprise, ainsi qu'une définition de l'infrastructure qui les soutient. Tel est le rôle de l'architecture d'entreprise (notée EA pour *Enterprise Architecture*). Par analogie avec la définition d'architecture d'application citée plus haut : « *l'EA pour une entreprise est ce qu'est un plan d'urbanisme pour une ville* » (Erl, 2005). Pour Pulier et Taylor (2006), l'EA implique une approche globale à la conception de systèmes TI dans le but de « construire » une structure qui satisfasse les besoins en affaires de l'entreprise. Les bénéfices apportés par une EA sont les suivants (Anaya et Ortiz, 2005) :

- Une documentation de l'entreprise facilement disponible.
- La capacité à unifier et à intégrer les processus d'affaires à travers l'entreprise et à former des liens avec les partenaires externes.
- Une plus grande agilité dans les changements d'affaires.
- Une maximisation de la réutilisation des modèles d'entreprise.
- L'acquisition d'une vision commune des communautés d'affaires et TI.

Anaya et Ortiz (2005) classifient grossièrement les EA proposées en :

1. des initiatives de normes *de facto*, comme le *Zachman's framework*, CIMOSA (*CIM Open Systems Architecture*) ou TOGAF (*The Open Group Architecture Framework*).
2. des normes d'organisations internationales, comme ISO 19439.

À long terme, une EA fournit une vision de la manière avec laquelle l'organisation planifie l'évolution de sa technologie et de ses environnements (Erl, 2005). Par conséquent, une caractéristique importante des EA est qu'elles ne sont pas statiques. L'architecte d'entreprise peut mettre en place un ensemble de systèmes TI magnifiquement conçus, et, peu de temps après, les processus d'affaires de l'entreprise nécessitent des changements majeurs (ex. l'entreprise vend une de ses filiales située en Europe et rachète une nouvelle compagnie d'assurance). Une nouvelle configuration des systèmes TI est alors nécessaire. Comment sera-t-elle réalisée sans mettre en danger la structure existante ?

Ceci a longtemps été le problème des systèmes TI fortement couplés, incompatibles et non flexibles (E. Pulier et H. Taylor, 2006). Les raisons de ce problème sont : des normes propriétaires, un mélange de logiciels provenant de distributeurs hétérogènes, des systèmes d'exploitation hétérogènes, des protocoles et des langages de programmation changeants et différents, etc. L'avènement des architectures axées sur le service (SOA) promet une EA plus flexible et plus facilement adaptable (E. Pulier et H. Taylor, 2006).

II.2.3 SOA : Concepts de base

Une SOA est une approche à la conception architecturale qui guide tous les aspects de la création et de l'utilisation des services de l'entreprise tout au long de leur cycle de vie (de la conception à l'abandon du service), et qui définit et fournit l'infrastructure TI qui permet à différentes applications d'échanger des données et de participer aux processus de l'entreprise, indépendamment des langages de programmation et des environnements sous-jacents à ces applications (E. Newcomer et G. Lomow, 2004). Le concept clé d'une SOA est le service. Pour cette raison, nous allons commencer par définir la notion de service.

II.2.3.1 Les services

Selon une perspective d'affaires, les services correspondent aux fonctions d'affaires réelles de l'entreprise. Ces fonctions peuvent être accédées selon des politiques précédemment établies, telles que (Newcomer et Lomow, 2004) :

- qui ou qu'est ce qui est autorisé à accéder au service,
- le coût d'utilisation du service,
- les niveaux de fiabilité du service (ex. délai moyen entre les pannes[7]),
- les niveaux de sécurité du service (les exigences en termes de domaines privés et d'intégrité),
- les niveaux de performance du service (ex.. temps de réponse), etc.

D'un point de vue technique, les services sont des fonctions à granularité fine encapsulées dans des composants réutilisables (Newcomer et Lomow, 2004). Ces fonctions sont interrogeables à travers des interfaces bien définies appelées « contrats de service » (section c) et qui servent à séparer les interfaces de leur mise en œuvre. Selon le modèle d'affaires axé sur les services, les services d'entreprise incluent deux groupes de services : les services d'affaires et les services TI. La figure ci-dessous montre une catégorisation de ces deux groupes de services :

[7] Ce qui est plus connu, dans la littérature traitant de la qualité, comme MTBF (*Mean Time Between Failures*) qui, avec MTTR (*Mean Time To Repair*), permet de calculer la disponibilité.

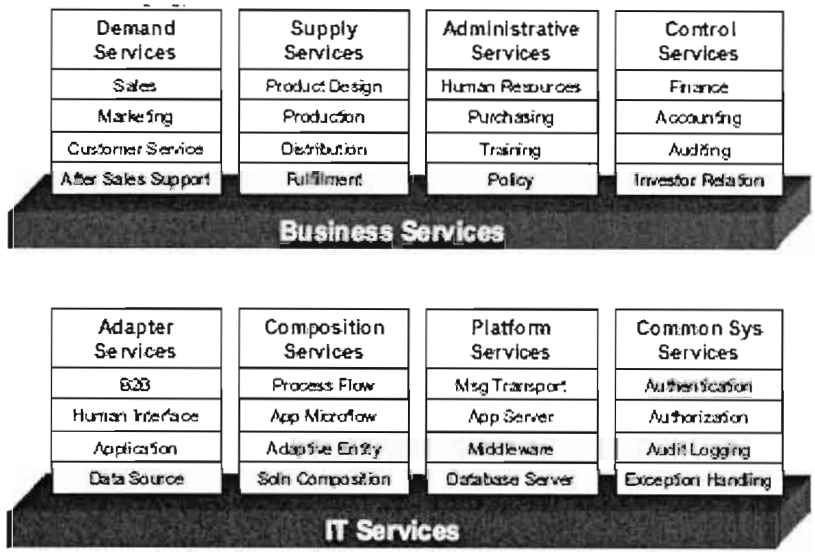


Figure 2.3 Les services d'entreprise : services d'affaires et services TI (Source : Huang et al., 2004)

a- Les services d'affaires

Cette couche fournit des services qui automatisent les services qu'offre l'organisation à ses clients et partenaires, et qui sont mis en œuvre par les systèmes TI de l'organisation ou par de tierces parties (Newcomer et Lomow, 2004). Les services d'affaires ont tendance à être spécifiques à un domaine particulier de service, tel que la finance, les ventes, le marketing, la fabrication, etc. Cependant, quelques services d'affaires sont réutilisables dans différents domaines de service. Dans un domaine de service particulier, tous les services communiquent à travers un vocabulaire de données commun qui les rend faciles à composer et à utiliser ensemble (Newcomer et Lomow, 2004).

b- Les services TI

Les services TI fournissent le support aux besoins des services d'affaires et à leur intégration (Huang et al., 2004). Ils sont réutilisables dans plusieurs domaines et incluent la

transformation et l'accès aux données, l'audit, la journalisation et la gestion des identités. Cette catégorie de services permet de remplir l'une des fonctions essentielles des SOA, qui est l'atténuation des risques pour l'organisation par la séparation de la logique d'affaires de l'ensemble des fonctionnalités de l'infrastructure (Newcomer et Lomow, 2004). Ceci permet d'augmenter la fiabilité et la flexibilité de l'infrastructure qui sont essentielles au bon fonctionnement des services d'affaires.

c- Les contrats de service

Chaque service (service d'affaires et service TI) possède un contrat de service qui sert à découpler les demandeurs et les fournisseurs de service qui pourront évoluer indépendamment les uns des autres tant que le contrat reste inchangé. Ainsi, dans une SOA, le contrat de service est un mécanisme qui permet de formaliser les limites du système, de minimiser les dépendances de ses composants et de maximiser son adaptabilité (E. Newcomer et G. Lomow, 2004).

II.2.3.2 Les principes de l'axé sur les services

Les principes de l'axé sur les services sont les principes qui guident les architectes et les développeurs lors de la définition des services d'affaires et des services TI (Newcomer et Lomow, 2004). Les principes les plus importants sont les suivants (Erl, 2005) :

1. Les services sont réutilisables : ils sont conçus de façon à ce qu'ils puissent être réutilisés ultérieurement.

2. Les services interagissent à travers un contrat formel : pour pouvoir interagir, les services utilisent un contrat formel qui décrit chaque service et définit les termes de l'échange d'information.

3. Les services sont faiblement couplés : ils sont conçus pour interagir avec le minimum d'interdépendances.

4. Les services font abstraction de la logique d'affaires : la seule partie visible du service est la partie exposée par le contrat de service (appelée interface publique). La logique sous-jacente du service est invisible et hors de portée du demandeur du service.

5. Les services sont composables : les services peuvent être associés entre eux pour réaliser une fonction particulière (ex. processus d'affaire). Ceci permet la représentation de la logique d'affaires selon plusieurs niveaux de granularité et facilite la réutilisation et la création de plusieurs niveaux d'abstraction.

6. Les services sont autonomes : les tâches accomplies par un service possèdent des limites. Le service possède un contrôle suivant cette limite et ne dépend pas d'autres services pour accomplir sa tâche.

II.2.3.3 La plateforme des services Web

La plateforme des services Web constitue une des formes possibles de mise en œuvre d'une SOA (Erl, 2005). D'autres approches, tel que celles fournies par les plateformes distribuées traditionnelles existent. Les SOA contemporaines se fondent sur les services Web combinés avec les principes de l'axé sur les services (Erl, 2005). La plateforme des services Web définit les normes et les moyens d'exécution disponibles pour tous les services, leur permettant ainsi de pouvoir interagir et interopérer d'une manière cohérente, indépendamment de la technologie utilisée (Newcomer et Lomow, 2004). Il s'agit d'une « plateforme de services » parce que son premier objectif est de faciliter la livraison des services (par opposition à une « plateforme d'application », qui facilite la création d'applications). Les éléments clé de la plateforme des services Web sont les services Web et les normes qui les régissent que nous allons définir avant de présenter les caractéristiques de cette plateforme.

a- Services Web et normes

Les avantages majeurs de la mise en œuvre d'une SOA utilisant les services Web, c'est que les services Web sont simples et neutres par rapport à la plateforme utilisée. L'architecture de base des services Web utilise les spécifications SOAP, WSDL et UDDI (voir figure ci-dessous).

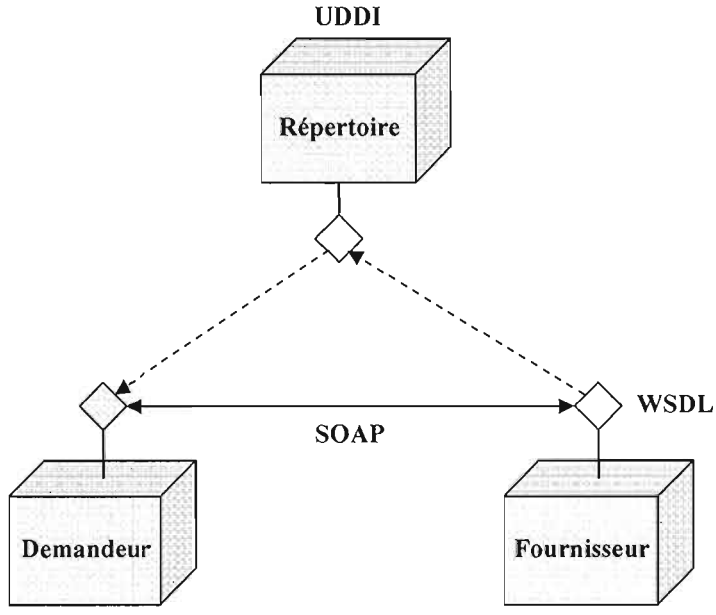


Figure 2.4 Architecture de base des Services Web (Source : Newcomer et Lomow, 2004)

La description du service, consistant à décrire les paramètres d'entrée du service, le format et le type des données retournées se fait selon un format normalisé. Le principal format de description de services est WSDL (*Web Services Description Language*), normalisé par le W3C. La publication, qui consiste à publier dans un répertoire les services disponibles aux utilisateurs, et la découverte (*discovery*), qui recouvre la possibilité de rechercher un service parmi ceux qui ont été publiés utilisent le standard UDDI (*Universal Description Discovery and Integration*) normalisé par OASIS. L'invocation, représentant la

connexion et l'interaction du demandeur avec le service utilise l'un des principaux protocoles pour l'invocation de service qui est SOAP (*Simple Object Access Protocol*). Tel que décrit par la Figure 3.4, le fournisseur publie une description WSDL de son service Web, et le demandeur accède à la description en utilisant UDDI. Il demande ensuite au fournisseur l'exécution du service en envoyant un message SOAP.

Mis à part les spécifications de base qui viennent d'être présentées, il existe une multitude de spécifications d'extension pour les services Web. Ces spécifications d'extension concernent la sécurité et la fiabilité des transactions (tel que *WS-Security* et *WS-Transactions*), la gestion des métadonnées (avec *WS-MetadataExchange*) et l'orchestration des services (avec *WS-BPEL*), qui fournissent à une SOA les solutions nécessaires pour garantir un bon niveau de qualité de service pour les entreprises.

b- Demandeurs et fournisseurs de service

Tel que déjà énoncé plus haut dans la section II.2.3.3, la principale technologie d'exécution associée à la création de services pour les SOA est la plateforme de services Web (Newcomer et Lomow, 2004). Celle-ci inclut les spécifications de base et d'extension des services Web qui viennent d'être énoncées et est en interaction avec des fournisseurs et des demandeurs de services.

Le fournisseur de service est un module logiciel qui met en œuvre le service selon le contrat de service. Il peut y avoir plusieurs fournisseurs différents pour un même service. Une instance de service est une instanciation exécutable d'un fournisseur de service. Il peut y avoir plusieurs instances de service pour un même fournisseur de service. Le demandeur de service, quant à lui, est un module logiciel qui invoque un service mis en œuvre par un fournisseur de service pour l'accomplissement de certaines tâches. Il utilise les moyens fournis par la plateforme de services Web pour localiser le service (UDDI) et communiquer avec lui d'une façon sûre (SOAP, *WS-Security*,...).

II.2.4 Les éléments d'une SOA

Cette section décrit et illustre les éléments qui constituent une SOA selon deux perspectives différentes : la première est une perspective de haut niveau et la deuxième, une perspective d'applications.

Selon une perspective de haut niveau, un système basé sur SOA est constitué de trois éléments (Kajko-Mattsson et al., 2007) (voir figure ci-dessous) :

(1) Des services.

(2) Des applications, qui découvrent et utilisent des services.

(3) Une infrastructure SOA[8], qui connecte les applications aux services.

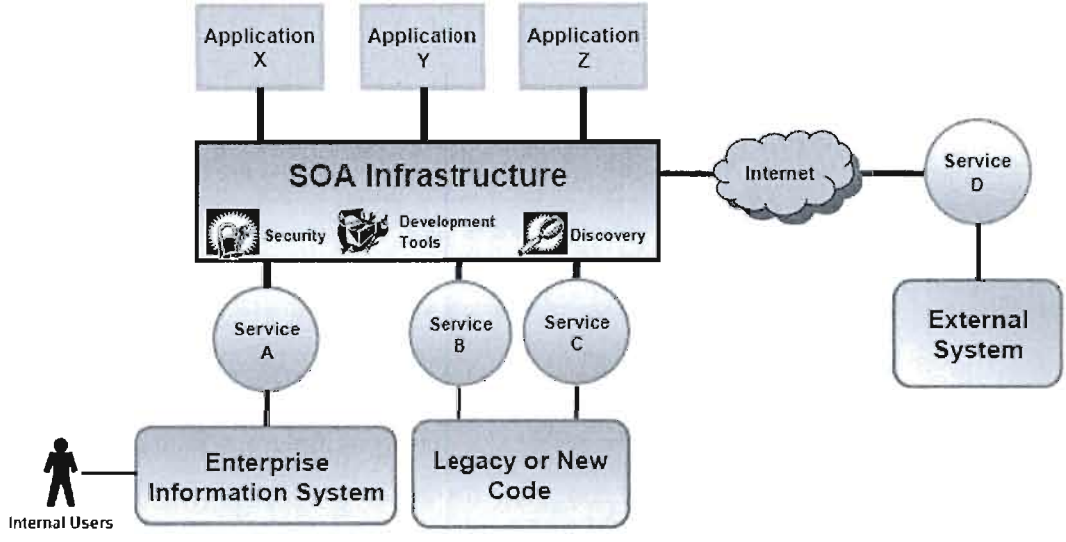


Figure 2.5 Éléments d'un système basé sur SOA (Source : Kajko-Mattsson et al., 2007)

Tel qu'illustré par la figure ci-dessus, l'infrastructure SOA fournit un mécanisme de communication standardisé entre les applications et les services. Toutes les applications invoquent des services de la même façon. Chaque service représente idéalement une tâche d'un processus d'affaires et fournit une interface qui est invoquée selon un protocole et un format de données compréhensible par tous les clients potentiels de ce service. Ainsi, selon cette configuration, les services peuvent être distribués à travers plusieurs organisations et peuvent être reconfigurés pour correspondre à de nouveaux processus d'affaires au fur et à mesure que les organisations évoluent (Kajko-Mattsson et al., 2007).

Kajko-Mattsson et al. (2007) définissent trois types de développeurs qui ont pour tâche de développer les éléments composant un système basé sur une SOA : les développeurs de l'infrastructure SOA, les développeurs d'application et les fournisseurs de service. Dans ce qui suit, nous citons les différentes tâches de ces trois types de développeurs.

Les tâches pour les développeurs de l'infrastructure SOA sont (Kajko-Mattsson et al., 2007) :

- Sélectionner les standards et les produits qui seront mis en œuvre dans l'infrastructure.
- Développer un ensemble de services d'infrastructure tel que la découverte, la communication et la sécurité.
- Identifier et développer des mécanismes de liens qui pourront satisfaire un grand ensemble d'utilisateurs de services potentiels.
- Fournir des outils pour les développeurs de services et d'applications.
- Documenter et maintenir l'infrastructure.

---

[8] Dans la littérature SOA, l'expression « bus de services » sert aussi à désigner l'infrastructure SOA.

Les principales tâches des développeurs d'application sont (Kajko-Mattsson et al., 2007) :

- Comprendre l'infrastructure SOA.
- Découvrir quels sont les services appropriés qui doivent être incorporés aux applications.
- Invoquer les services identifiés incluant la conversion des données, le traitement des erreurs et la gestion de la disponibilité.
- Tester l'exactitude des services dans le contexte de l'application en cours de développement.

Les principales tâches des fournisseurs de services sont (M. Kajko-Mattsson et al., 2007) :

- Comprendre les besoins des utilisateurs potentiels du service.
- Comprendre l'infrastructure SOA.
- Développer le code qui reçoit la requête de service, traduit la requête en appels destinés à de nouveaux systèmes ou à des systèmes existants et produit une réponse.
- Décrire et publier le service.
- Développer le code d'initialisation et les fonctionnalités du service.

Krogdahl et al. (2005) définissent les éléments constitutifs d'une SOA selon une perspective d'applications. Selon ces derniers, et tel qu'illustré par la figure ci-dessus, une SOA d'entreprise est constituée de quatre couches : la couche des processus d'affaires, la

---

Nous préférons l'emploi de « bus de services » car « infrastructure SOA » porte à confusion avec

couche des applications d'interface utilisateur, la couche de chorégraphie des services et la couche des applications de service. Dans ce qui suit, nous allons décrire les interactions entre ces quatre couches telles que présentées par Krogdahl et al. (2005).

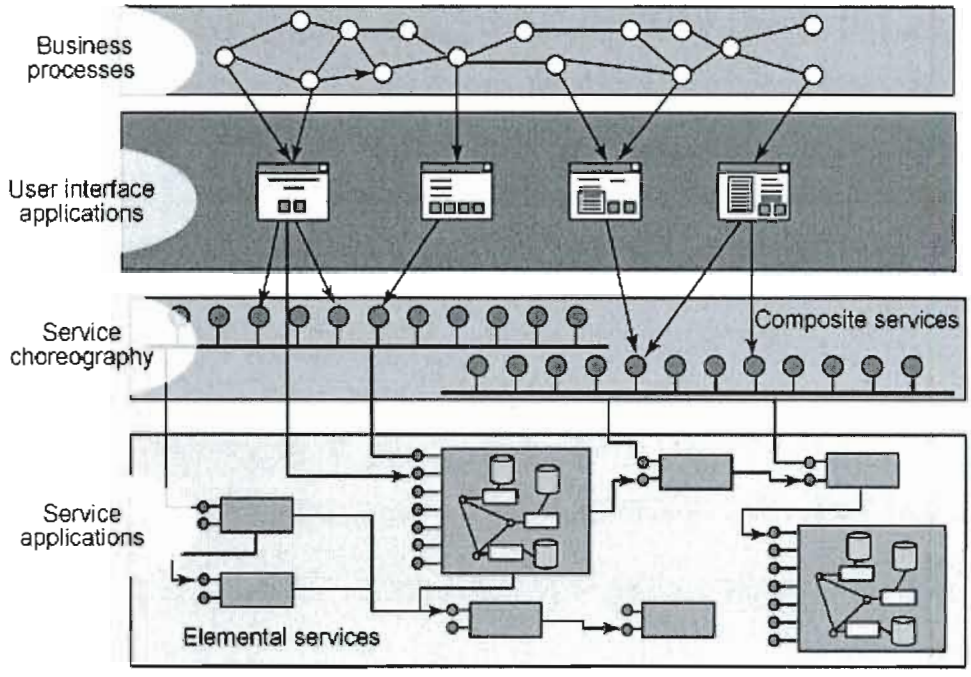


Figure 2.6 Éléments d'une SOA (Source : Krogdahl et al., 2005)

Les processus d'affaires sont totalement ou partiellement soutenus par les applications des interfaces utilisateur et les applications de service. Une étape d'un processus d'affaires est soit exécutée manuellement, soit accomplie à travers une application d'interface utilisateur. Les applications d'interfaces utilisateur mettent en œuvre un ensemble de flux de travaux et de services d'appels qui réalisent la fonctionnalité d'affaires.

Dans la couche de chorégraphie de service, les services composés sont définis au moyen d'un langage de chorégraphie tel que BPEL. La chorégraphie des services composés

---

l'architecture SOA au complet.

défini leur flux ou leur composition en services élémentaires. Cette couche devrait être réalisée à l'aide d'un outil de chorégraphie qui permettrait les spécifications graphiques (ex. *IBM Websphere Business Integration Modeler).*

Les services élémentaires, utilisés par la couche de chorégraphie des services ainsi que par les applications d'interface utilisateur, sont mis en œuvre par les applications de service. Les mises en œuvre des services appellent d'autres services souvent à partir d'applications de services.

II.2.5 Exemple d'une SOA d'entreprise

Afin d'illustrer d'une façon plus concrète ce qu'est qu'une SOA d'entreprise et ce qu'elle pourrait apporter comme avantages, nous avons choisi de citer un exemple tiré du livre de Pulier et Taylor (2006) « *Understanding Enterprise SOA* ».

Dans cet exemple, une organisation possède un ordinateur central placé au siège de l'entreprise, qui héberge un entrepôt de données et un logiciel de génération de rapports financiers. L'entreprise reçoit de l'information provenant d'une division (vente et comptabilité), qui fait tourner des applications Oracle sur Sun, et d'une usine, qui fait tourner une suite d'applications SAP sur des mini-ordinateurs IBM. L'usine qui est gérée par le siège de l'entreprise se connecte, à travers une interface sécurisée par un pare-feu (*firewall*), à un fournisseur qui fait tourner un système de gestion de la relation client (système CRM – *Customer Relationship Management*) du fournisseur de solutions Siebel sur Windows. Cette architecture (voir figure ci-dessous) ne pose aucun problème. Mis à part qu'elle dépend d'interfaces propriétaires, elle tourne bien et satisfait parfaitement aux besoins de l'entreprise.

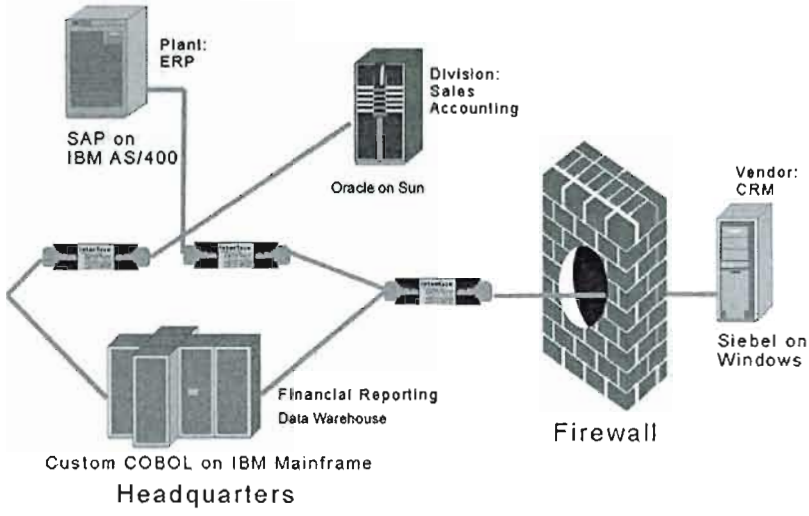


Figure 2.7 Une EA traditionnelle avec des systèmes couplés par leurs interfaces propriétaires (Source : Pulier et Taylor, 2006)

Mais que se passerait-il si jamais le fournisseur changeait son système CRM vers une application Java sur Sun, l'usine changeait vers SAP sur Sun et le siège remplaçait son ordinateur central par une grappe d'ordinateurs Windows ? Ce genre de changements est un fait réel des grandes entreprises.

Dans le cas qui vient d'être présenté, selon une EA traditionnelle, les ajustements de l'architecture seraient effectués en se basant sur les nouveaux processus d'affaires. Le résultat (voir figure ci-dessous) serait une autre EA non flexible et avec de nouveaux systèmes aussi fortement couplés par leurs interfaces propriétaires que les systèmes précédents. La mise en place de la nouvelle architecture peut prendre des mois et des millions de dollars et, un an plus tard, de nouveaux ajustements pourraient être nécessaires.

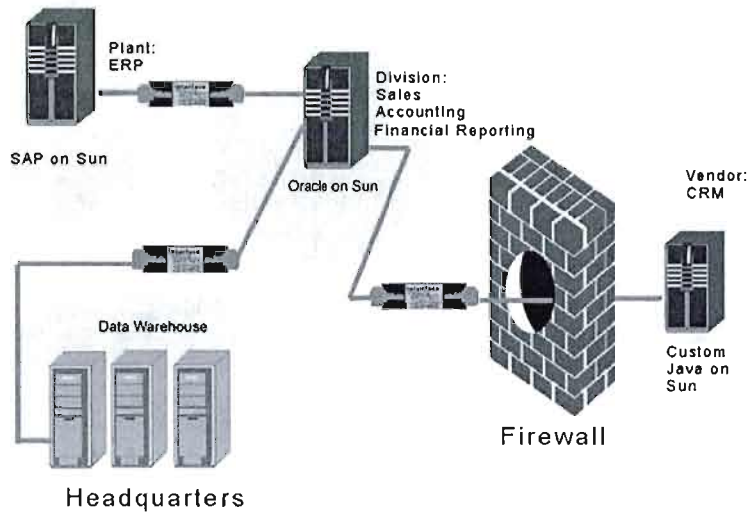


Figure 2.8 Nouvelle EA non flexible, avec de nouveaux systèmes aussi fortement couplés qu'auparavant (Source : Pulier et Taylor, 2006)

Voyons comment la compagnie serait si elle avait une SOA. La figure ci-dessous montre la nouvelle configuration. Maintenant, chaque application logicielle dans l'architecture est exposée en tant que service Web. Ainsi, elle peut être accédée de n'importe où à travers le réseau, peu importe les caractéristiques de l'ordinateur qui effectue la requête.

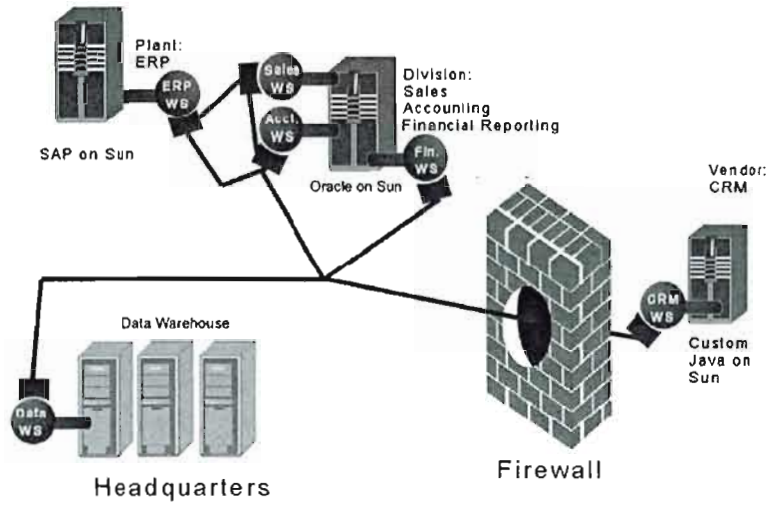


Figure 2.9 Nouvelle configuration avec une SOA : chaque composant du système est exposé en tant que service Web (Source : Pulier et Taylor, 2006)

Le modèle SOA a plusieurs avantages par rapport au modèle traditionnel. La compagnie pourra sauver de l'argent en réduisant le nombre d'interfaces propriétaires à maintenir. La gestion du changement sera aussi beaucoup plus simple. Si par exemple, le fournisseur change la mise en œuvre Java sur son système SUN actuel vers un système IBM, les demandeurs du service Web CRM n'en seront pas affectés.

Supposons qu'un des gestionnaires de la compagnie fictive décident de changer la structure de sa division en éliminant la division qui était responsable de la comptabilité, des ventes et de la publication de l'information financière et en assignant la responsabilité de toutes ces fonctions, ainsi que l'ERP (*Enterprise Resource Planning*), à l'usine. Et admettons qu'il décide d'éliminer Oracle et de mettre le tout sur SAP. Comme le montre la figure ci-dessous, ces changements ne nécessitent pas l'achat d'interfaces propriétaires. Tous les systèmes communiquent toujours avec SOAP, donc la modification de l'EA est simple comparé à toute l'écriture de code nécessaire et en plus coûteuse à une EA traditionnelle

fortement couplée. Le temps et l'argent nécessaire pour effectuer les changements seraient une fraction de ce qui est nécessaire au modèle traditionnel.

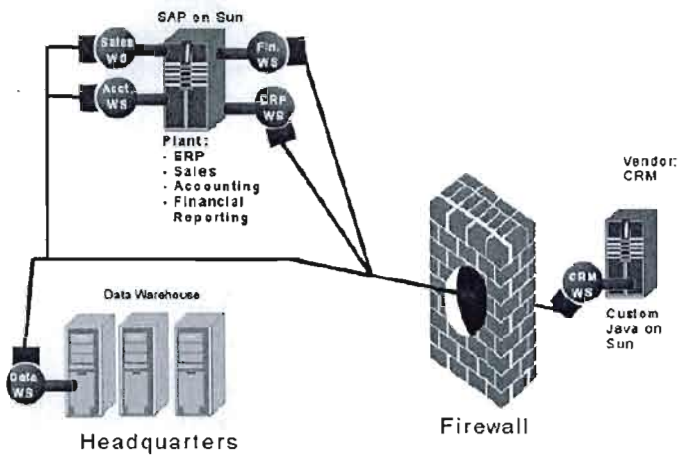


Figure 2.10 L'implémentation de changements dans une SOA est beaucoup plus simple que dans une EA traditionnelle (Source : Pulier et Taylor, 2006)

Ainsi, le principal bénéfice apporté par une SOA est de fournir l'agilité à l'entreprise. Le gestionnaire d'affaires peut agir selon les stratégies d'affaires sans les contraintes typiques des TI, et le gestionnaire TI fournit les résultats d'une façon efficace et peu coûteuse (Pulier et Taylor, 2006).

II.3 Les méthodes de développement traditionnelles

Un environnement axé sur les services se réfère à un domaine logique de l'entreprise où sont appliqués les principes de l'axé sur les services (Erl, 2005). Avant de nous intéresser aux caractéristiques du développement logiciel dans un environnement axé sur les services à travers un survol de la littérature, nous allons d'abord nous intéresser aux méthodes de développement traditionnelles. Cette section commence par la présentation des diverses méthodes de développement traditionnelles (section II.3.1), en s'intéressant en particulier à *Unified Process* (section II.3.2) et à *Rational Unified Process*, qui est une instanciation par

Rational Software (IBM) des principes d'UP (section II.3.3). Les raisons de l'intérêt particulier accordé à *Unified Process* (UP) seront explicitées dans la section qui lui est consacrée.

II.3.1 Aperçu sur les méthodes de développement traditionnelles

Comme tout projet d'ingénierie, de fabrication ou de développement de produit, un projet de développement logiciel doit venir à bout des besoins conflictuels en temps, en ressources et en fonctionnalités (D. Krafzig et al., 2005). Cependant, les projets logiciels possèdent des spécificités qui posent des problèmes qu'on ne rencontre pas dans les autres industries. En particulier, les interfaces entre les logiciels d'entreprise et les utilisateurs humains ont une plus grande complexité que les interfaces entre les utilisateurs humains et d'autres types de technologies. Par exemple, un système logiciel tel qu'un système CRM ou un système intégré de planification d'entreprise, nécessite des interfaces utilisateur complexes et une interaction directe et fréquente avec l'utilisateur final (Krafzig et al., 2005). C'est pour cette raison que les projets logiciels d'entreprise nécessitent une plus grande interaction avec le client durant la phase de développement.

Malheureusement, les premières méthodes de gestion de projets logiciels qui ont vu le jour n'étaient pas capables de venir à bout de cette difficulté. Le modèle de développement en cascade assume implicitement, au début du projet, que les besoins du client sont fixes, et que les changements à ces besoins sont l'exception, non la norme (Niclot, 2005). Les phases du modèle en cascade incluent la spécification des exigences, la conception générale et la conception détaillée, le codage, les tests de modules et d'intégration, et les tests d'acceptation (Krafzig et al., 2005). Le modèle en cascade est basé sur l'élaboration d'une documentation complète à la fin de chaque phase, qui doit être approuvée formellement avant de passer à la phase suivante. La livraison finale est un produit monolithe (en un seul bloc)[9].

[9] Il s'agit d'une interprétation un peu simpliste du modèle qu'on retrouve souvent dans la littérature. Au fait, des itérations étaient presque toujours prévues.

En raison des limites de cette approche, en particulier l'incapacité du modèle en cascade à venir à bout des exigences instables du client, des approches évolutionnaires pour la gestion de projets logiciels ont émergé durant les vingt dernières années. Ces modèles plus incrémentaux et plus itératifs ont été construits selon une philosophie d'interaction et de changement, qui convient aux exigences fonctionnelles instables (Krafzig et al., 2005). Ces modèles impliquent typiquement des livraisons fréquentes, qui sont utilisées comme base pour obtenir un *feedback* continu du client (Niclot, 2005). Au lieu de livrer le résultat du premier travail après des mois ou même des années (comme dans le modèle en cascade), les itérations de ces approches évolutionnaires durent quelques semaines ou même jours.

Un des représentants de ces processus de développement est le modèle *Rapid Application Development* (RAD) de James Martin, qui est fondé sur des étapes incrémentales. Chaque incrément représente un modèle en cascade à courte durée. Barry Boehm a développé l'un des premiers modèles complètement évolutionnaire, appelé le modèle en spirale, et qui est fondé sur un ensemble de cycles de développement complets qui raffinent continuellement les connaissances sur le produit final et aident à contrôler les risques du projet (Krafzig et al., 2005).

Un nombre de modèles de processus de développement complexes et sophistiqués a été commercialisé durant les dix dernières années. La plupart de ces modèles sont fondés sur une approche itérative combinée avec le concept de l'orienté objet et utilisent souvent UML (*Unified Modeling Language*), comme notation de modélisation. La méthode la plus représentative de cette classe de processus de développement est *Rational Unified Process* (RUP), qui est une instanciation par *Rational Software* (IBM) des principes d'UP (Niclot, 2005). RUP est itératif, dépend fortement de la visualisation à travers UML et utilise les architectures axées sur les composants (Krafzig et al. 2005). Dans la même arène des processus de développement itératifs relativement lourds, on trouve aussi *Dynamic Software Development Method* (DSDM), *Microsoft Solution Framework* (MSF), et *Catalysis* (Krafzig et al., 2005).

Durant les années 1990, des approches de développement logiciel itératives moins lourdes ont émergé. Elles sont souvent qualifiées de méthodes de développement agiles. La notion de méthode agile est née en 2001, à travers un manifeste signé par dix-sept personnalités (créateurs de méthodes de développement et dirigeants de sociétés) (Krafzig et al., 2005). Un représentant de ces nouvelles méthodes est *Extreme Programming* (XP), qui dépend essentiellement de la programmation en paire : deux développeurs travaillant conjointement sur une partie de code, l'un travaille sur le codage, et l'autre sur la conception et le test de ce qui a été développé (Niclot, 2005). Avec XP, le nouveau code peut être intégré immédiatement et des tests sont exécutés rapidement.

La méthode dX de Robert Martin a comme objectif de faire la balance entre les méthodes lourdes comme RUP, et les méthodes légères comme XP (Krafzig et al., 2005). dX est une implémentation de RUP pour intégrer la méthode XP. Elle permet de combiner la puissance d'UP et la souplesse des méthodes agiles (Niclot, 2005). La figure ci-dessous illustre les différentes méthodes de gestion de projet. Les différentes approches sont classées en approches lourdes ou légères et approches linéaires ou itératives.

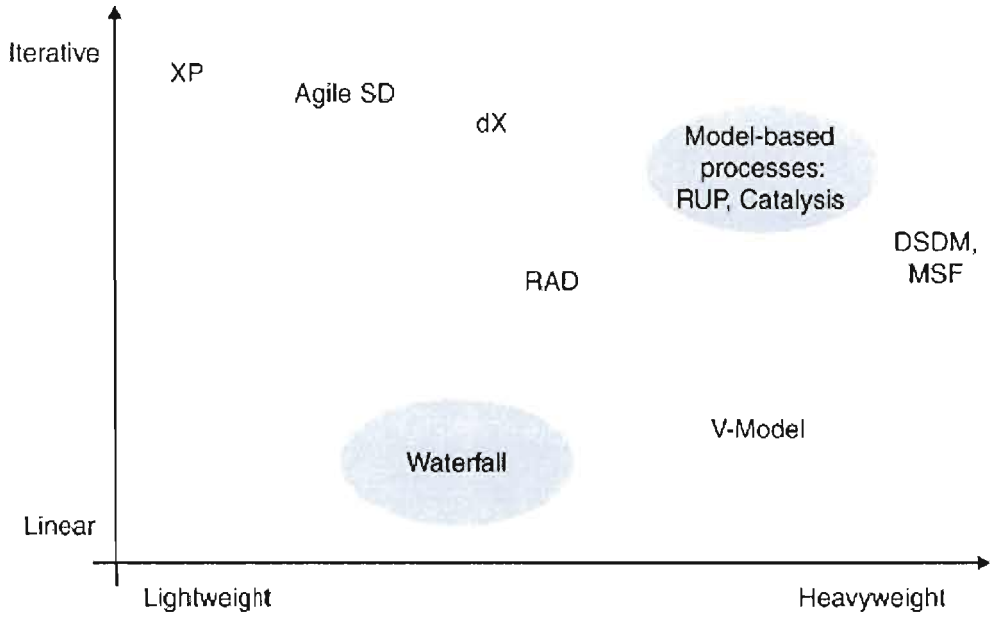


Figure 2.11 Les différentes méthodes de gestion de projet : approches lourdes vs. légères et linéaires vs. itératives (Source : Krafzig et al., 2005)

II.3.2 Unified Process (UP)

Dans la littérature SOA, certaines recherches (Ivanyukovich et al., 2005 ; Mittal, 2005) ont choisi UP comme méthode destinée au développement axé sur les services. C'est ce qui explique notre intérêt particulier pour ce processus de développement. Voici quelques caractéristiques d'UP qui en ont peut être fait un bon candidat pour le développement logiciel dans les environnements axés sur les services :

UP est une méthode de développement mature : UP est l'aboutissement de trois décennies de développement et d'usage pratique (Jacobson et al., 2000). La figure ci-dessous retrace la succession des produits qui en sont issus, depuis le processus Objectory (dont la première version est sortie en 1987), jusqu'au Rational Unified Process (sorti en 1998), et à partir desquels il a pris de la maturité.

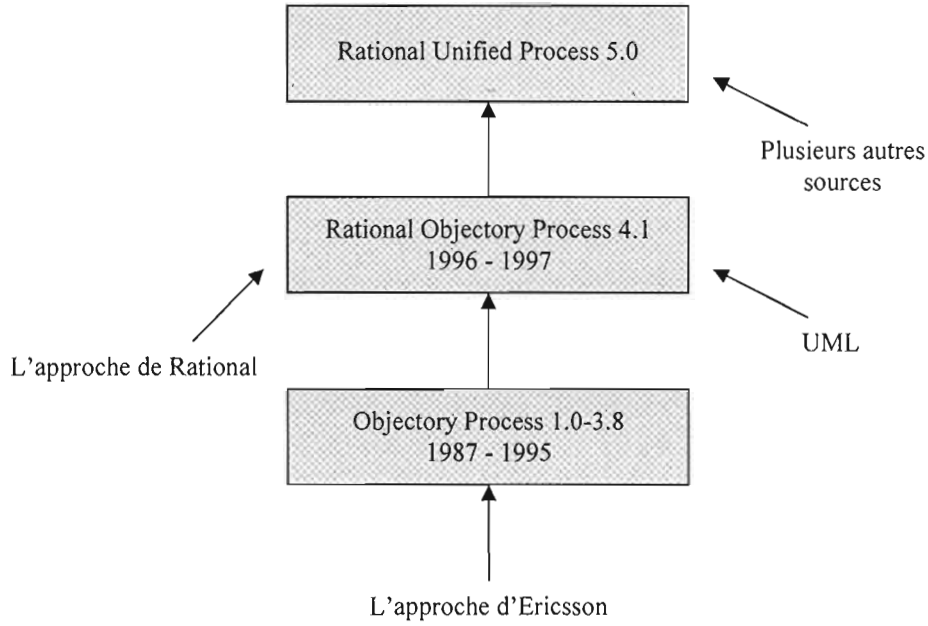


Figure 2.12 L'évolution d'UP : versions du produit dans les rectangles (Source : Jacobson et al., 2000)

UP est un processus de développement logiciel générique : UP est un processus de développement, c.-à-d. qu'il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel (voir figure ci-dessous). Mais c'est plus qu'un simple processus. Il s'agit d'un *framework* de processus générique pouvant être adapté à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprises, à différents niveaux de compétence et à différentes tailles de projets (Jacobson et al., 2000).

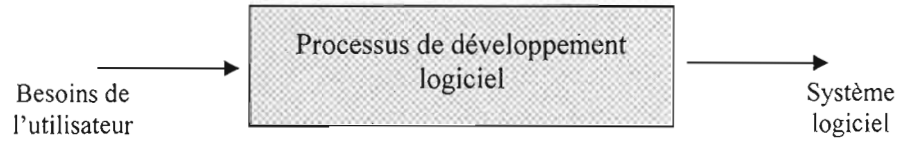


Figure 2.13 Un processus de développement logiciel (Source : Jacobson et al., 2000)

UP utilise la notation UML : UP utilise UML pour la création des plans d'élaboration et de construction du système logiciel. En fait UML fait partie intégrante d'UP : l'un et l'autre ont été développés de concert. Plusieurs recherches récentes (S. Johnston, 2005 ; Z. Stojanovic et al., 2004 ; T. Zhang et al., 2006) présentent des approches pour la modélisation SOA utilisant la notation UML.

Les traits véritablement distinctifs d'UP tiennent en trois expressions clé : piloté par les cas d'utilisation, centré sur l'architecture, itératif et incrémental. Dans ce qui suit, nous allons expliquer ces trois expressions clé, pour ensuite nous intéresser au produit UP le plus repandu, qui est *Rational Unified Process* (RUP).

UP est piloté par les cas d'utilisation

Un cas d'utilisation est la description d'une fonctionnalité du système produisant un résultat satisfaisant pour l'utilisateur (utilisateur humain ou autre, comme par exemple un système dialoguant avec le système en cours de développement). Les cas d'utilisation saisissent les besoins fonctionnels et leur ensemble constitue le modèle des cas d'utilisation. Mais les cas d'utilisation ne sont pas un simple outil de spécification des besoins du système, ils en guident également la conception, l'implémentation et les tests ; c.-à-d.qu'ils guident le processus de développement (Jacobson et al., 2000).

Piloté par les cas d'utilisation signifie que le processus de développement suit une voie spécifique ; c.-à-d. qu'il procède par une série d'enchaînements d'activités dérivés des cas d'utilisation. Les cas d'utilisation sont spécifiés, ils sont conçus et ils constituent la source qui permettra aux testeurs d'élaborer les cas de tests (Jacobson et al., 2000). Les cas d'utilisation guident le processus, mais doivent absolument être développés en tandem avec l'architecture du système.

UP est centré sur l'architecture

L'architecture logicielle émerge des besoins de l'entreprise, tels qu'ils sont exprimés par les utilisateurs et autres intervenants et tels qu'ils sont reflétés par les cas d'utilisation. Elle subit néanmoins l'influence d'autres facteurs, tels que (Jacobson et al., 2000) :

- La plate-forme sur laquelle devra s'exécuter le système, par exemple : l'architecture matérielle, le système d'exploitation, le système de gestion de bases de données et les protocoles de communication réseau.

- Les briques de base réutilisables disponibles pour le développement, par exemple une infrastructure préfabriquée réutilisable (*framework*) pour les interfaces utilisateur graphiques.

- Les systèmes existants.

- Les besoins non fonctionnels (ex. la performance, la fiabilité).

Pour déterminer l'architecture du système à développer, les architectes doivent travailler à partir des cas d'utilisation les plus significatifs ; ceux qui constituent le cœur même du système. Il est indispensable qu'il y ait une interaction entre les cas d'utilisation et l'architecture pour créer un produit réussi. En clair, l'architecte (Jacobson et al., 2000) :

- Crée une ébauche grossière de l'architecture, en partant de l'aspect qui n'est pas propre au cas d'utilisation (ex. la plate-forme), tout en ayant une compréhension globale de ces derniers avant d'esquisser l'architecture.

- Il travaille, ensuite, sur un sous-ensemble des cas d'utilisation identifiés – ceux qui représentent les fonctions essentielles du système et qui seront décrits en détail et réalisés sous forme de sous-systèmes, de classes et de composants.

- L'architecture se dévoile peu à peu, au rythme de la spécification et de la maturation des cas d'utilisation, qui favorisent, à leur tour, le développement d'un nombre important de cas d'utilisation.

Ce processus se poursuit jusqu'à ce que l'architecture soit jugée stable. Ainsi, les cas d'utilisation guident l'architecture du système, qui influence, à son tour, leur sélection. L'architecture et les cas d'utilisation évoluent donc de façon parallèle au cours du CVP.

UP est itératif et incrémental

Le développement d'un produit logiciel destiné à la commercialisation est un travail vaste qui peut s'étendre sur plusieurs mois, voir même plusieurs années. Pour cette raison, il est utile de découper le travail en plusieurs parties, qui constituent des sous-projets. Chaque sous-projet représente une itération qui donne lieu à un incrément. Les itérations désignent des étapes de l'enchaînement d'activités, tandis que les incréments correspondent à des stades du développement du produit. Une itération (Jacobson et al., 2000) :

1. Part des cas d'utilisation pertinents, identifiés et sélectionnés par les développeurs,
2. Se prolonge par le travail de développement normal (analyse, conception, implémentation et tests) qui est guidé par l'architecture choisie,
3. Réalise sous forme de code exécutable les cas d'utilisation définis au cours de l'itération.

Les itérations, qui doivent être sélectionnées et menées de façon planifiée, présentent les avantages suivants (Jacobson et al., 2000) :

- Limiter les coûts, en termes de risques, aux strictes dépenses liées à une seule itération.

- Limiter les risques de retard de mise sur le marché du produit développé, par l'identification des risques dès les premiers stades de développement.

- Permettre aux développeurs de travailler plus efficacement vers des objectifs clairs à court terme, plutôt qu'en fonction d'un planning à long terme.

- Faciliter l'adaptation à l'évolution des besoins des utilisateurs, qui se dégagent peu à peu des itérations successives.

L'architecture fournit la structure qui servira de cadre au travail effectué au cours des itérations, tandis que les cas d'utilisation définissent les objectifs et orientent le travail de chaque itération (Jacobson et al., 2000).

II.3.3 *Rational Unified Process* (RUP)

Nous allons commencer par un aperçu historique avant de nous lancer dans les pratiques spécifiques du processus de RUP.

II.3.3.1 Histoire de RUP

Le développement de RUP a commencé durant les années 1980 à *Rational Software Corporation*, une compagnie qui était dédiée au développement de systèmes logiciels gros et complexes. Durant les années 1990, des dizaines de langages de modélisation étaient utilisés, incluant Booch, Buhr, Object Modeling Technique (OMT), et Shlaer-Mellor. Le marché était tellement fragmenté qu'il était difficile de développer un outil unique pour supporter la majorité des efforts de développement logiciel (Gibbs, 2006).

En réponse à ce besoin, *Rational* a décidé de se lancer dans la création d'une méthode standardisée pour la modélisation des systèmes logiciels. L'effort de *Rational* s'est traduit par l'utilisation de la méthode développée par Grady Booch (l'inventeur de la méthodologie Booch et qui était déjà employé de *Rational*), l'embauche de James Rumbaugh (OMT) et de

Ivar Jacobson (*Objectory Process*) et l'acquisition de la compagnie de Jacobson, *Objectory* (Gibbs, 2006). Ensemble, ces trois acteurs (connus sous le nom des « trois amis » ou « three amigos ») ont travaillé sur un projet qui a donné naissance au langage de modélisation UML, qui allait remplacer les nombreux langages existants. Le deuxième effort de *Rational*, a été d'embarquer dans le développement d'un ensemble documenté de meilleures pratiques pour le développement logiciel et qui serait supporté par l'arsenal d'outils de *Rational*. Ceci a permis la création de RUP.

RUP a beaucoup hérité de *Objectory Process*, plus particulièrement de la notion de cas d'utilisation pour décrire l'interaction entre les utilisateurs et le système (Gibbs, 2006). En 2003, IBM fait l'acquisition de *Rational Software*. RUP continue à évoluer et à être mis à jour au rythme du changement des pratiques de l'industrie.

II.3.3.2 Les six pratiques de RUP

Quand RUP a été développé, il était centré sur l'application de six pratiques. De la version initiale de RUP à celle de 2005, ces pratiques étaient les suivantes (Gibbs, 2006) :

1. Développer itérativement,
2. Gérer les exigences[10],
3. Utiliser une architecture axée sur les composants,
4. Modéliser visuellement,
5. Vérifier constamment la qualité,
6. Gérer les changements.

---

[10] Le lecteur a sans doute l'impression que nous employons « besoins » et « exigences » comme des synonymes. Il ne s'agit pas d'une fausse impression, mais dans la suite de notre étude nous allons différencier ces deux concepts (qui sont souvent confondus surtout dans la littérature française).

Les concepteurs de RUP ont continué à faire évoluer le processus au fur et à mesure que les méthodes et les pratiques ont mûri. En Octobre 2005, un article est apparu dans le journal « *e-zine The Rational Edge* » de IBM *Rational*. Dans cet article, Koll et Royce mettent à jour les six meilleures pratiques, comme suit :

1. Adapter le processus,
2. Gérer les priorités des parties prenantes,
3. Collaborer au travers des équipes,
4. Démontrer la valeur itérativement,
5. Élever le niveau d'abstraction,
6. Focaliser continuellement sur la qualité.

II.3.3.3 Phases du cycle de vie de RUP

Dans RUP, le CVP est divisé en quatre phases : Création, Élaboration, Construction et Transition. Chaque phase possède un but et un objectif spécifique. La figure ci-dessous illustre les quatre phases, ensemble avec les niveaux d'effort dans chaque discipline à travers le CVP. Examinons chaque phase :

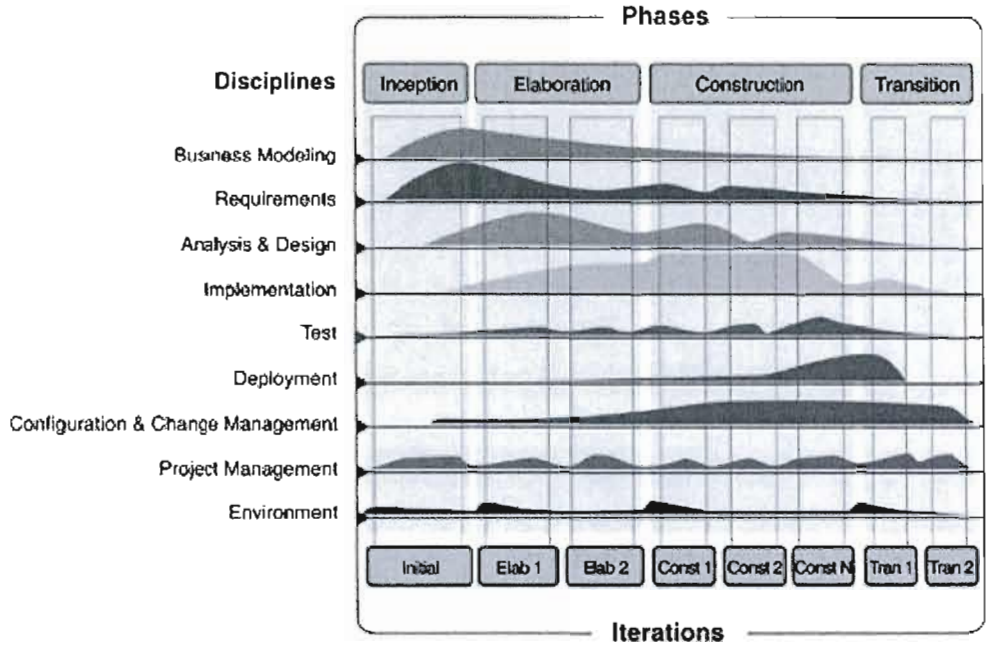


Figure 2.14 Les quatre phases de RUP avec les niveaux d'effort dans chaque discipline à travers le CVP (Source : Kruchten, 2000)

La phase de création établit la faisabilité

Le premier objectif de la phase de création est de procéder à l'étude de rentabilité du produit. Cette phase ne propose pas une étude complète du système envisagé, mais vise à dégager un faible pourcentage des cas d'utilisation qui prendront part à l'étude initiale. L'étude initiale s'articule autour de quatre étapes (Jacobson et al., 2000) :

1. Délimiter la portée du système proposé : définir les frontières du système et commencer à identifier les interfaces avec d'autres systèmes.

2. Décrire et esquisser l'architecture candidate du système, en particulier les parties inédites, risquées ou complexes du système.

3. Identifier les risques les plus sérieux, ceux qui affectent la capacité à construire le système et voir si l'on peut envisager un moyen de les réduire.

4. Démontrer aux clients et aux utilisateurs potentiels que le système proposé est en mesure de prendre en charge leurs objectifs en construisant un prototype de mise à l'épreuve du concept.

La phase d'élaboration s'intéresse à la « capacité de réalisation »

La phase d'élaboration livre comme principal résultat une architecture stable du système et une estimation fidèle des coûts (Jacobson et al., 2000). Pour atteindre le premier objectif, les membres de l'équipe créent une architecture de référence composée des artefacts des modèles, de la description de l'architecture et de sa mise en œuvre exécutable. L'architecture de référence couvre les exigences fonctionnelles et les exigences supplémentaires du système qui sont significatives sur le plan architectural ou importantes pour les intervenants, comme :

- le nombre d'utilisateurs concurrents que le système devra gérer ;
- les exigences en temps de réponse ;
- le niveau fiabilité (temps moyen entre les pannes) ;
- le niveau de criticité du système (Gibbs, 2006).

Pour atteindre le deuxième objectif, une offre abordant les questions de calendrier, de personnel et de budget est élaborée et les risques les plus significatifs qui sont de nature à bouleverser ces estimations sont identifiés (Jacobson et al., 2000). À la conclusion de la phase d'élaboration, les cas d'utilisation qui couvrent les exigences les plus importantes pour les utilisateurs ont été formulés.

Les exigences et l'architecture représentent l'essentiel du travail des phases de création et d'élaboration (Jacobson et al., 2000).

La phase de construction « construit » le système

L'objectif de la phase de construction est de réaliser les capacités opérationnelles initiales du système (Gibbs, 2006). Ceci peut ne pas constituer le produit complètement fini, mais un produit, qui peut être appelé « version bêta », et qui met en œuvre toutes les exigences clé du système. Il peut y avoir des produits qui manquent ou qui sont incomplets, comme les fichiers d'aide et les scripts d'installation, mais, à ce niveau, tous les risques majeurs liés au développement du système ont été identifiés et atténués, l'architecture a été déterminée et la plupart des exigences du système ont été identifiées.

La majorité du temps et des ressources du projet sont alloués durant la phase de construction (Gibbs, 2006), qui comporte généralement un nombre d'itérations supérieur à ceux des phases précédentes (Jacobson, 2000). Dans les projets basés sur RUP, les tests ont lieu durant chaque itération, quelle que soit la phase. Ceci veut dire que durant chaque itération, le test des nouvelles fonctionnalités commence aussitôt qu'elles deviennent disponibles et les tests de régression des fonctionnalités qui ont été construites durant les itérations précédentes sont aussi effectués. Les défauts découverts durant les tests sont documentés et évalués pour déterminer leurs priorités. Les défauts les plus sérieux sont corrigés durant la même itération et les défauts de plus faible priorité peuvent être différés à l'itération suivante. Le but de chaque itération est de produire une version exécutable, présentable et stable (Gibbs, 2006).

La phase de Transition aborde l'environnement utilisateur

Durant la phase de transition, les itérations finales incorporent (Jacobson et al., 2000) :

- La correction des anomalies détectées sur la version bêta,

- La modification du logiciel suite à des demandes d'amélioration ou en réponse à des problèmes qui n'avaient pas été prévus,

- L'adaptation du logiciel à l'environnement utilisateur (ex. paramétrage, transfert des données si le produit remplace un système existant),

- La recommandation au client sur la mise à jour de l'environnement (matériel, système d'exploitation, protocoles de communication, etc.) destiné à accueillir le logiciel.

- Élaboration des manuels et de la documentation à partir de la première documentation de la version bêta.

La phase de transition peut être triviale ou complexe, dépendamment de la nature du produit : dans une extrême, on peut avoir un produit qui réside dans un système unique et qui implique quelques utilisateurs experts localisés dans un même endroit ; et dans l'autre extrême, on peut avoir un très grand système critique avec des centaines d'utilisateurs, avec une phase de transition très étendue (Gibbs, 2006). Dans ce cas, il est important d'avoir un groupe significatif de développeurs et un personnel en alerte, prêt à résoudre les problèmes qui seront découverts.

II.3.3.4 RUP est-il un processus agile ?

Tel que déjà énoncé dans la section II.3.1, il existe plusieurs variations de processus agiles : XP (*Extreme Programming*), *Scrum*, *Crystal*, etc. Ils varient selon des facteurs comme la durée de l'itération et d'autres facteurs ayant différentes emphases.

RUP est-il agile? La réponse est qu'il peut l'être, dépendamment de la façon avec laquelle il est adapté au projet (Gibbs, 2006). Si on veut se conformer avec les valeurs exprimées dans l'*Agile Manifesto*, il faut considérer les points suivants lors de la mise en place de RUP (Gibbs, 2006) :

- Choisir seulement les artefacts les plus vitaux qui sont nécessaires au client et à l'équipe de développement. Tâcher d'éliminer tout processus ou document inutile.

- Tâcher de maintenir les itérations aussi courtes que possible. Se rappeler que le but de chaque itération est de produire une version partielle mais fonctionnellement acceptable. La version doit être testée durant le développement de façon à ce que le livrable fourni à la fin de l'itération fonctionne correctement.

- Les méthodes agiles accentuent, par-dessus tout, la collaboration entre tous les membres de l'équipe de développement ainsi que la collaboration avec les clients et les utilisateurs. Lorsqu'il existe un doute concernant la fonctionnalité ou les priorités, le client est consulté et c'est lui qui prend une décision.

- Si possible, impliquer le client dans la définition des tests d'acceptation des différentes caractéristiques du système logiciel. Les tests seront exécutés à la fin de l'itération, et le client donne son accord final sur la fonctionnalité à la fin de l'itération. L'avantage de cette approche est qu'elle élimine le besoin d'avoir un test d'acceptation massif à la fin du projet. Même si le client exige un test d'acceptation complet, la probabilité d'avoir des surprises est plus mince.

- Les méthodes agiles accentuent l'automatisation des tâches redondantes quand cela est possible. En particulier, tester les exigences sera automatisé, car, au fur et à mesure que les itérations continuent, la quantité de fonctionnalités à tester croît (que ce soit les tests de régression des fonctionnalités précédemment livrées et les tests des fonctionnalités nouvellement développées).

Les pratiques de RUP sont parfaitement compatibles avec les valeurs agiles. La clé est d'adapter RUP pour être aussi simple que possible et de focaliser sur la production fréquente de versions que le client réexamine et accepte (Gibbs, 2006).

II.4 Développement logiciel axé sur les services : défis et pratiques

Actuellement, dans la littérature, il est admis que le concept d'« axé sur les services » définit des principes architecturaux qui sont analogues à ceux de la conception logicielle « axée sur les composants » (Ivanyukovich et al., 2005 ; Stojanovic et al., 2005 ; Yukyong et Hongran, 2006). C'est pour cette raison que les méthodes actuellement utilisées pour le développement d'applications SOA sont fondées sur celles utilisées pour le développement de systèmes basés sur les composants. Cependant, certains aspects qui caractérisent les projets de développement dans un environnement axé sur les services (ex. l'aspect des services faiblement couplés, le problème de l'identification des services avec une granularité optimale, etc.) engendrent de nouveaux défis et des différences considérables dans la façon avec laquelle les projets basés sur une SOA sont conduits.

D'après Stojanovic et Dahanayake (2005), le développement axé sur les services présente beaucoup de défis pour les organisations, parmi lesquels :

- Les approches traditionnelles de développement logiciel, comme le modèle en cascade ne conviennent pas au développement de systèmes axés sur le service : les composants développés peuvent ne pas supporter les besoins spécifiés au départ, puisque les besoins sont identifiés aux premières étapes du modèle et que le choix des composants est effectué durant les dernières étapes. Dans les méthodes évolutives, où des besoins additionnels peuvent être ajoutés si nécessaire, l'inaccessibilité du code du composant, qui constitue la base de la construction des services, empêche les développeurs d'ajuster ces derniers à leurs besoins.

- Il y a un manque d'outils d'analyse qui supportent le développement axé sur la réutilisation, en particulier pour le développement en boîte fermée (*black box*), qui consiste à développer des composants logiciels dont le fonctionnement est connu et documenté, mais dont la structure interne est inconnue.

- Il y a un manque de méthodes pour effectuer la concordance entre les fonctionnalités et les services, ainsi que le regroupement logique des services en domaines.

L'identification des services et la détermination des fournisseurs de services correspondants est une première étape critique de l'élaboration d'une solution axée sur le service. Le regroupement de services ou le *clustering* a une influence directe sur plusieurs caractéristiques importantes du système tel que le contrôle d'accès, la performance, la maintenabilité, la gestion de la sécurité, etc.

- Les modèles de services actuels ne définissent pas de schéma pour contrôler, définir et autoriser les changements à un ensemble de services existants faisant partie d'un même domaine.

D'après Mittal (2006), les facteurs inhérents au succès de la mise en œuvre d'une SOA sont :

- Un processus de développement clairement défini.
- Une communication améliorée à travers les équipes de projet.
- Des politiques de support et de gouvernance claires.

Nous avons remarqué que le point qui revient le plus souvent dans la littérature en ce qui concerne les défis du développement axé sur les services est la définition d'une méthode ou d'un processus de développement adéquat. Quelles seraient donc les caractéristiques d'une méthode de développement axée sur les services, telles que vues par les recherches actuelles ?

Toutes les méthodes de développement (RUP, XP, dX, etc.) ont leurs forces et faiblesses, et le choix et l'application d'une méthode va dépendre de l'expérience et des meilleurs pratiques établies par chaque entreprise. Cependant, en présence d'un environnement axé sur les services, les approches les plus appropriées sont celles qui supportent le développement itératif (Krafzig et al., 2005 ; Stojanovic et Dahanayake, 2005). Les projets de développement de solutions axées sur le service ressemblent, en surface, aux projets de développement d'applications distribuées (Erl, 2005). Cependant, afin de

construire et de positionner les services d'une façon convenable dans une solution axée sur le service, les cycles des projets traditionnels nécessitent quelques ajustements. Ces ajustements se reflètent dans les différentes approches de développement axées sur le service proposées par quelques récentes recherches, que nous allons présenter brièvement selon un ordre chronologique.

- *Service Oriented Analysis and Design* (SOAD) est une approche améliorée et interdisciplinaire de modélisation de service, proposée par O. Zimmermann et al. (2004), partant de processus de développement et de notations existants.

- *Service-Oriented Modeling and Architecture* (SOMA) illustre les activités d'une méthode de modélisation axée sur le service, proposée par Arsanjani (2004). Pour l'identification et la spécification de service, elle combine les trois approches d'analyse ascendante, descendante et *middle-out*.

- Ivanyukovich et al. (2005) proposent l'adoption du processus de développement RUP pour le développement axé sur les services. Mais proposent, par ailleurs, de l'adapter pour qu'il puisse répondre à certaines caractéristiques spécifiques de ce type d'environnement.

- Un modèle conceptuel, présenté par Yukyong et Hongran (2006) et appelé M4SOD (*Method For Service Oriented Development*), a pour but de formaliser le processus de développement SOA. Cette méthode met l'accent sur les phases d'identification et de réalisation des services.

- Rahmani et al. (2006) proposent une approche de modélisation et de conception de systèmes basés sur une SOA qui utilise l'architecture dirigée par les modèles (*Model Driven Architecture*, MDA).

- *Service-Oriented Unified Process* (SOUP), un processus de développement destiné aux systèmes basés sur SOA et proposé par Mittal (2006), reprend les meilleurs éléments de RUP et de XP.

Après avoir défini plus en détail les approches de développement que nous venons de citer (section II.4.1), nous allons nous intéresser à d'autres aspects retrouvés dans la littérature, qui sont particulièrement importants pour l'atteinte de notre objectif de recherche. Ces aspects incluent les stratégies de développement logiciel axé sur les services (section II.4.2), les pratiques de la gestion de projet axée sur le service (section II.4.3), les principes de la gouvernance SOA (section II.4.4), ainsi que les rôles du développement logiciel axé sur les services (section II.4.5).

II.4.1 Approches de développement axées sur les services

II.4.1.1 L'approche SOAD (Service-Oriented Analysis and Design)

Zimmermann et al. (2004) proposent une approche interdisciplinaire, nommée SOAD pour *Service Oriented Analysis and Design*, en tant que discipline de modélisation qui est construite sur les fondations bien établies de l'analyse et de la conception orientée objet (*Object Oriented Analysis and Design*, OOAD), l'architecture d'entreprise (*Enterprise Architecture*, EA) et la gestion des processus d'affaires (*Business Process Management*, BPM). La figure ci-dessous illustre l'approche SOAD et les éléments sur lesquels elle se base.

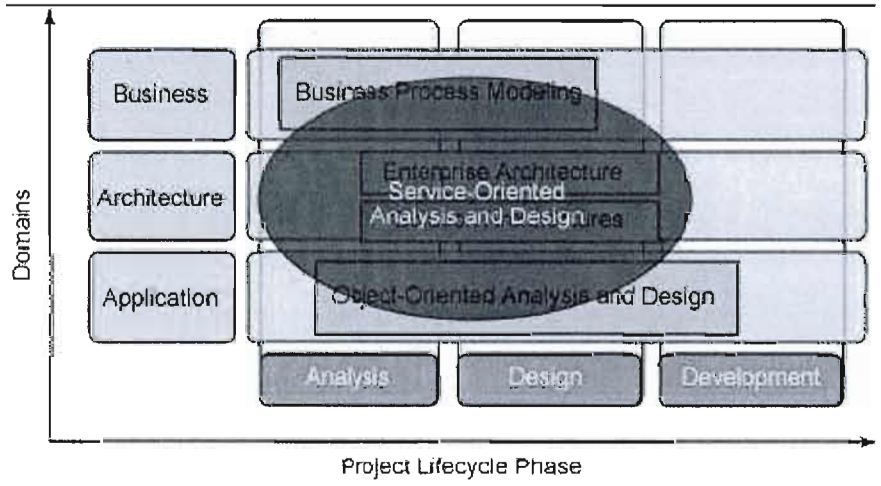


Figure 2.15 SOAD et ses ingrédients: OOAD, BPM et EA (Zimmermann et al., 2004)

Zimmermann et al. (2004) considèrent que les trois disciplines de modélisation OOAD, EA et BPM fournissent des pratiques de haut niveau qui peuvent être combinées pour fournir le support requis au paradigme SOA. Mis à part la reprise de notions déjà existantes, SOAD a apporté des innovations en ce qui concerne les répertoires de service, l'orchestration de service et le bus de services d'entreprise. Cependant, les notations et les processus de SOAD restent à être définis en détail.

II.4.1.2 L'approche SOMA (Service-Oriented Modeling and Architecture)

SOMA (Arsangani, 2004) est une approche d'IBM, qui définit trois activités de modélisation de service qui sont l'identification, la spécification et la réalisation de service. Ces étapes sont constituées de plusieurs sous-étapes décrivant les artefacts à délivrer et recommandant les techniques appropriées (Zimermann et al, 2005). Tel qu'illustrée par la figure ci-dessous, la méthode SOMA applique des approches combinées pour l'identification de service incluant la modélisation *goal-service*, la décomposition de processus et l'analyse des systèmes existants.

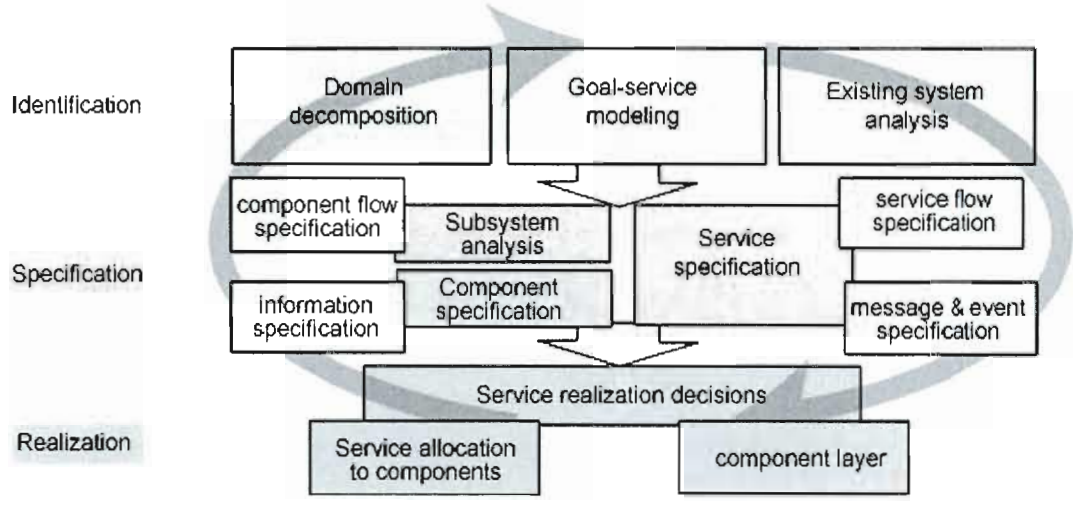


Figure 2.16 La méthode SOMA (Source : Arsangani, 2004)

L'identification des services est effectuée à partir du modèle d'affaires à travers une décomposition de domaine, ainsi qu'à partir des systèmes existants. La technique additionnelle appelée « *goal-service modeling* » est reliée aux objectifs d'affaires. Cette dernière technique est, par exemple, exprimée en termes d'indicateurs clé de performance des abstractions de service identifiées, facilitant le contrôle des objectifs d'affaires (Arsanjani, 2004).

Toutes les étapes SOMA sont accomplies d'une manière itérative et incrémentale. Durant la *spécification de service*, les artefacts sont clairement définis, comme par exemple les services atomiques et les services composés, ainsi que les composants qui les mettent en œuvre avec leurs interfaces. Dans leur ensemble, ces spécifications forment le modèle de service, un livrable clé SOMA, qui couvre la syntaxe et les sémantiques de l'invocation de service, ainsi que d'autres caractéristiques opérationnelles tel que la possession de service, les dépendances et le versionnement (Zimmermann et al., 2004). La réalisation des services et des composants est assistée par des outils et des techniques telles que les *patterns* (par exemple les *patterns* d'IBM pour le *e-business* peuvent être utilisés).

II.4.1.3 Adopter et adapter RUP pour le développement axé sur les services

Ivanyukovich et al. (2005) considèrent que RUP est le meilleur candidat pour la spécification du processus de développement logiciel dans un environnement axé sur les services. Mais, la fourniture d'applications SOA développées sous RUP nécessite la personnalisation du processus pour qu'il puisse adresser un certain nombre de problèmes, parmi lesquels (Ivanyukovich et al., 2005) :

- Le faible couplage des services permet la planification et le développement de chaque service avec la façon qui lui convient le mieux. Par conséquent, le CVP, en sa totalité, doit être supporté par la méthode RUP, via des sous-projets parallèles pour chaque service qui doivent converger vers une intégration et des tests finaux, selon une séquence particulière.

- Une attention particulière doit être portée aux méthodes, aux outils et aux technologies durant les phases de création et d'élaboration pour la discipline d'intégration. La facilité d'intégration des services dépend fortement de la qualité de planification des opérations entre les services.

- La planification de la communication *ad hoc* et potentiellement asynchrone affecte la conception du système en entier. C'est ce qui constitue la différence avec les systèmes fortement couplés basés sur les composants. Même si une partie du projet utilise des mécanismes de communication synchrones, le traitement asynchrone devrait être considéré comme un moyen pour minimiser le temps de traitement des données.

- La nature faiblement couplée des services et leur forte granularité requièrent le détachement de l'étape d'intégration de l'enchaînement d'activités de l'étape de mise en œuvre.

- Un paradigme axé sur les services atténue les changements dans les besoins d'affaires avec une plus grande flexibilité de l'architecture suggérée ; d'autre part, RUP recommande les changements au niveau de l'architecture tôt durant le CVP.

En partant d'une perspective axée sur le service, les auteurs établissent la définition et l'ajout de deux caractéristiques majeures de la personnalisation de RUP, qui sont :

L'ajout d'une nouvelle discipline au processus, appelée « Integration », qui résout le problème de la convergence des services individuels en une application parfaitement fonctionnelle. Cette discipline est spécialement active durant la phase d'élaboration, lorsque l'architecture de l'application doit être définie. Une telle discipline est importante, car elle évite le travail non nécessaire, tôt dans la phase de construction quand les modifications majeures nécessaires sont liées à l'intégration des différents services.

Un changement dans la façon avec laquelle la discipline de « mise en œuvre » est perçue et planifiée. La ***mise en œuvre*** consiste en un certain nombre de sous-projets individuels, chacun livrant un composant particulier à l'application SOA envisagée. Il est important que tous les sous-projets finissent selon une séquence correcte durant la phase de construction. Contrairement au RUP traditionnel, la phase de construction est caractérisée par une activité croissante de la discipline ***intégration*** et une activité décroissante de la discipline ***mise en œuvre***.

II.4.1.4 La méthode M4SOD (*Method for Service-Oriented Development*)

M4SOD est une méthode présentée par Yukyong et Hongran (2006) dans le but de formaliser le processus de développement SOA. Elle est constituée de 6 phases qui sont composées d'activités, de travaux et d'artefacts. M4SOD définis les procédures et les directives pour l'accomplissement de chaque phase en mettant surtout l'accent sur les phases d'identification et de réalisation des services. Elle supporte à la fois le développement du nouveau système et l'intégration des systèmes hérités basés sur SOA. La figure ci-dessous illustre ses différentes phases.

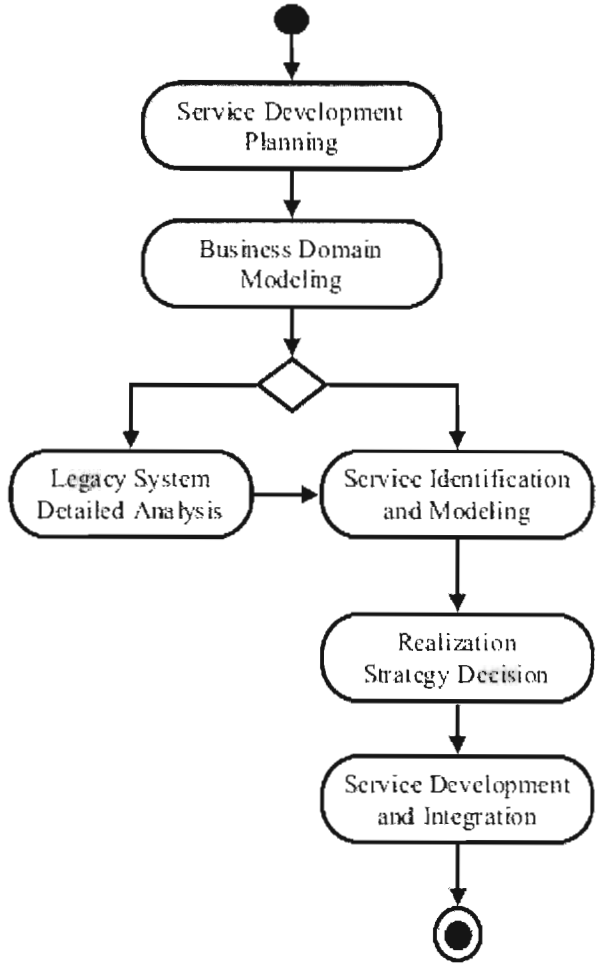


Figure 2.17 Le cycle de vie des phases de M4SOD (Source : Yukyong et Hongran, 2006)

Dans M4SOD, les services sont dérivés à partir du modèle des cas d'utilisation qui représente le *workflow*. La phase d'identification et de modélisation de service implique trois activités :

1. Analyse du modèle des cas d'utilisation préparé durant la phase précédente.
2. Identification des services unitaires.
3. Modélisation de l'architecture du service.

Après avoir analysé le *workflow* de chaque cas d'utilisation, un ensemble de tâches importantes peut être dérivé de ce dernier. Les services unitaires sont identifiés par la comparaison de ces différentes tâches en se basant sur des directives, établies par Yukyong et Hongran (2006), et qui ont pour but de définir des services unitaires selon une granularité optimale.

Durant la phase qui correspond à la stratégie de la décision de réalisation, on décide de la stratégie à utiliser pour mettre en œuvre chaque service. Les stratégies pour la réalisation d'un service sont au nombre de trois :

1. Développer le service,
2. Envelopper le service dans un composant existant,
3. Adopter l'un des services publiés.

Pour une meilleure prise de décision, Yukyong et Hongran (2006) suggèrent une méthode détaillée pour répondre aux contraintes techniques et économiques. Cette méthode se base sur des tables de critères de prise de décision établies à partir des descriptions des services unitaires identifiés lors de la phase précédente.

II.4.1.5 Une SOA dirigée par les modèles

Dans le but de construire des systèmes basés sur une SOA d'une façon efficace, Rahmani et al. (2006) proposent une approche de modélisation et de conception impliquant l'architecture dirigée par les modèles (MDA pour *Model Driven Architecture*). MDA n'est pas un concept nouveau. Il est largement utilisé dans plusieurs disciplines d'ingénierie telles que le génie mécanique, architectural, électrique et chimique (Tsai et al., 2007). Cependant, il est relativement nouveau par rapport à l'informatique où il constitue un nouveau paradigme qui a été proposé pour le développement de systèmes complexes distribués (Rahmani et al., 2006).

Le principe de base de MDA est l'élaboration de modèles indépendants de plateformes (*Platform Independent Model*, PIM) et la transformation de ceux-ci en modèles dépendants de plateformes (*Platform Specific Model*, PSM) pour la mise en œuvre du système. La transformation entre ces modèles est accomplie via des outils automatisés qui emploient des techniques de modélisation et de transformation de modèles.

Dans l'approche de modélisation et de conception de SOA proposée par Rahmani et al. (2006) (voir figure ci-dessous), le PIM du système est crée et, ensuite, le PSM fondé sur SOA est généré (ce PSM est un PIM pour le niveau suivant). Et finalement, le PSM final fondé sur une plateforme cible (tel que la plateforme des services Web) est généré. L'approche de modélisation proposée emploie la notation UML.

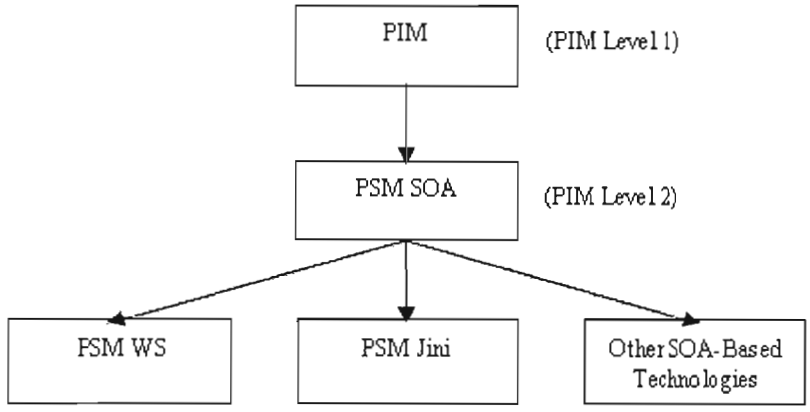


Figure 2.18 Processus de modélisation et de conception d'une SOA basé sur MDA (Rahmani et al., 2006)

II.4.1.6 Le processus SOUP (*Service-Oriented Unified Process*)

SOUP est une méthode de développement logiciel qui utilise les meilleurs éléments de RUP et de XP afin de construire et de gérer un projet SOA (Mittal, 2006). Elle se base, d'abord, sur RUP pour créer une fondation SOA et utilise, par la suite, XP pour construire,

assembler et réutiliser les services. Elle vise les projets de développement SOA dans n'importe quelle organisation.

SOUP est une approche de développement logiciel à six phases. Chaque phase présente un ensemble distinct d'activités critiques au succès du déploiement d'une SOA. Bien évidemment, comme dans tout projet, le processus doit être adapté à l'organisation.

Mittal (2006) divise le processus SOUP en deux catégories :

- SOUP pour le déploiement SOA initial
- SOUP pour la gestion SOA continue

Chaque catégorie du processus admet différentes phases caractérisées par des livrables et des activités clé, que nous allons définir conformément à la description proposée par Mittal (2006).

La figure ci-dessous montre les phases du processus SOUP durant le déploiement SOA initial.

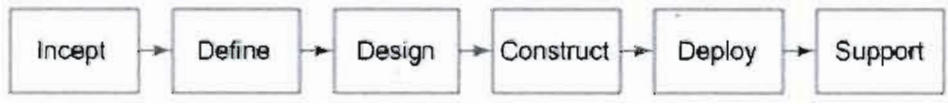


Figure 2.19 Le modèle du processus SOUP durant le déploiement SOA initial (Mittal, 2006)

Démarrage : Durant cette phase, les besoins du projet SOA dans l'organisation en question sont définis. Les analystes et les gestionnaires de projet analysent l'organisation du client afin de déterminer les avantages d'une solution fondée sur SOA. Ils examinent les opérations internes de l'organisation ; ses interactions avec ses partenaires, fournisseurs et

clients ; et son modèle d'affaires dans sa totalité. Ces facteurs aident les analystes à développer et à recommander une stratégie SOA. À ce niveau, une analyse RCI (Rendement du Capital Investi) complète de la stratégie SOA recommandée doit être établie. Cette analyse doit montrer les bénéfices en termes de coûts à court, moyen et long terme. Le plan de communication est le livrable le plus important durant cette phase, ce dernier assure que les parties prenantes comprennent et son impliquées dans le processus.

Définition : Il s'agit de la phase la plus critique du projet SOA. Les membres de l'équipe doivent participer activement à la définition des exigences et des cas d'utilisation développés durant la phase initiale du déploiement de SOA. D'après Mittal (2006), le livrable le plus important à ce niveau est le modèle de support et de gouvernance. Il définit comment l'organisation devra effectuer la maintenance de SOA et quelles sont les directives de sa gouvernance (ex. gestion des SLA). Si une SOA est déployée et que personne ne l'utilise, c'est un échec de projet. Le document du modèle de support et de gouvernance permet de prévenir ce genre de situations. Un autre livrable important est celui de l'infrastructure technique qui va soutenir SOA dans son ensemble, incluant des prévisions budgétaires détaillées sur les serveurs, l'infrastructure du réseau et les besoins en formation.

Conception : Durant cette phase, les artefacts de conception de la phase de définition seront détaillés. Les réalisations des cas d'utilisation et le document de l'architecture logicielle seront traduits en des documents de conception détaillée. À ce niveau, les architectes SOA – typiquement un sous-ensemble des architectes de l'entreprise – doivent impliquer les architectes du projet pour s'assurer que la conception présentée par l'équipe SOA pourra marcher pour des projets spécifiques. Les architectes SOA peuvent même accomplir des petits projets de démonstration de faisabilité pour prouver la vision SOA.

Construction : Durant cette phase, une méthode de construction itérative doit être utilisée. Cette phase inclus de nouvelles activités de développement et d'intégration. Seulement, les activités ne se limitent pas seulement au logiciel, mais peuvent aussi impliquer des sous projets reliés à l'infrastructure (ex. efforts de centralisation de l'hébergement des

serveurs). La totalité des efforts accomplis est gérée par un plan de projet SOA unique, mais différentes sous équipes de petite taille travailleront sur différentes activités de construction.

Déploiement : Durant la phase de déploiement, le livrable le plus important est l'application en production. Cependant, un livrable encore plus important est le plan des projets pilotes SOA. Ceci amène à l'étape suivante de la méthode SOUP, qui détermine comment les projets en cours vont utiliser la nouvelle architecture.

Support : La phase finale du cycle de développement logiciel est très importante. Les activités impliquées dans cette phase sont la maintenance, la réparation des bogues et la formation.

La méthode SOUP est de même utilisée pour les projets utilisant une SOA qui a déjà été déployée. Dans de telles situations, SOUP s'inspire plus de la méthode XP. La figure ci-dessous montre une superposition de ces deux processus.

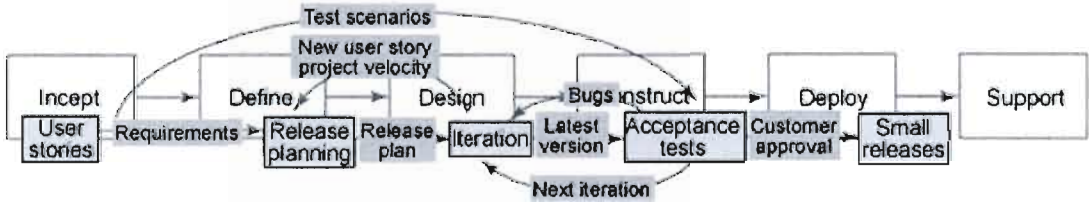


Figure 2.20 Les processus SOUP et XP superposés (Source : Mittal, 2005)

Une fois déployée, l'infrastructure SOA offre un cadre sur lequel les applications peuvent être construites en tant que services. Les principales tâches des équipes de projet sont alors de construire et d'utiliser des services en suivant les mêmes directives que celles énoncées dans la section précédente ; cependant, chaque phase est significativement réduite. Par exemple, la phase de conception devient courte, puisque la plupart de la conception a été réalisée durant le déploiement initial de SOA. Durant cette phase, il faut juste focaliser sur ce qui doit être construit au dessus des services existants et sur la manière de construire les

nouveaux services. La phase de construction, quant à elle implique un travail d'assemblage plus que de développement. Au fur et à mesure que plus de services deviennent disponibles, chaque projet impliquera plus de réutilisation et moins de construction.

Les approches de développement axées sur les services que nous venons de présenter sont fondées sur des stratégies de développement qui peuvent être ascendantes, descendantes ou une combinaison des deux. Par exemple, la méthode M4SOD (Yukyong et Hongran, 2006) suit une stratégie descendante puisqu'elle part des modèles de cas d'utilisation pour identifier les services, alors que la méthode SOMA (Arsanjani, 2004) combine la stratégie ascendante et descendante puisqu'elle identifie les services à travers une décomposition du domaine, ainsi qu'à partir des systèmes existants.

Dans ce qui suit, nous allons nous intéresser aux différentes stratégies de développement axées sur les services citées dans la littérature.

II.4.2 Les stratégies du développement logiciel axé sur les services

Trois stratégies de développement d'applications SOA ont été citées par la littérature : la stratégie descendante (*top-down*), la stratégie ascendante (*bottom-up*) et la stratégie de développement de compromis (*meet-in-the-middle* ou *middle-out*[11]). À notre avis les différentes étapes de ces trois stratégies de développement n'ont pas été assez détaillées dans la littérature et les différents intervenants dans le processus de développement n'ont pas été identifiés. Néanmoins, Perepletchikov et al. (2005) proposent, pour chaque stratégie, diverses tâches qui améliorent les attributs de qualité logicielle[12]. Les attributs de qualité logicielle

---

[11] Cette stratégie est aussi qualifiée d'« agile », en traduction à « *agile development strategy* ». Nous avons préféré l'emploi de « stratégie de compromis » puisque « agile » introduit un homonyme avec les méthodes de développement agiles.

[12]La qualité logicielle n'a volontairement pas été abordée dans la revue de la littérature car il s'agit d'un domaine fort étendu et mouvant. De plus nous considérons la qualité comme un élément pratiquement non négociable dans un projet d'envergure et donc toutes nos considérations se fondent sur l'hypothèse (non vérifiable dans le cadre de notre étude) que la qualité ne se détériore pas lors du passage à SOA.

considérés dans leur étude sont de deux sortes : les attributs liés surtout à la gestion du projet (coûts en infrastructure et effort de développement) et les attributs de qualité interne du logiciel (couplage, cohésion et complexité). Dans ce qui suit, nous allons présenter les trois stratégies de développement d'applications SOA, accompagnées des tâches les plus importantes identifiées par Perepletchikov et al. (2005) et qui ont pour effet d'améliorer les attributs de qualité logicielle.

III.4.2.1 Stratégie de développement descendante

Une stratégie descendante débute avec la description des besoins et la modélisation des processus d'affaires. Les descriptions et les modèles sont raffinés souvent à l'aide de notations semi-formelles jusqu'à obtenir le code des programmes. Cette stratégie de développement consiste en une décomposition du domaine d'affaires de l'entreprise en domaines ou champs fonctionnels et sous-systèmes (Karhunen et al., 2005). Les modèles des processus d'affaires servent à identifier les différents services qui seront fournis par les fournisseurs de services et utilisés par les demandeurs de services.

Principales tâches pour améliorer la qualité logicielle

Il y a plusieurs activités qui sont conduites lors de la réalisation de services selon l'approche descendante Ces activités incluent la construction de nouveaux services, la réutilisation de services existants et l'achat de nouveaux services.

Les tâches les plus importantes, lors de la construction de nouveaux services sont les suivantes :

- Identifier les petites unités logicielles (composants de services) qui peuvent être réutilisées dans différents contextes. Les composants de services doivent par la suite

être recomposés en des services à forte granularité[13] qui correspondent aux processus d'affaires.

- Structurer le système en un ensemble de services hautement réutilisables et faiblement couplés. De cette façon, l'entreprise peut augmenter le RCI en raison de la diminution des coûts de maintenance et de la capacité à réutiliser des services dans les projets futurs.

- Mettre en place des ESB (*Enterprise Services Bus*) pour faciliter la connectivité et le routage des messages entre les services et des ISE (*Integrated Services Environments*) pour concevoir, configurer, tester et débuguer les processus d'affaires. Même si ces produits augmentent les coûts en infrastructure, ils vont réduire l'effort de développement.

Les tâches les plus importantes, lors de la réutilisation des services sont les suivantes :

- Engager un expert en modélisation de processus d'affaires afin d'identifier les services existants qui peuvent être réutilisés.

- Incorporer, dans l'entreprise, un répertoire de services privé pour centraliser les descriptions des services publiés, afin de faciliter la future réutilisation des services.

- Intégrer les services préexistants au système en cours de développement par l'intermédiaire du code de composition ; les services eux-mêmes ne sont pas modifiés. Ceci diminuera le temps de test, puisqu'il n'est plus nécessaire de conduire des tests unitaires sur les services préexistants.

Finalement, avant de prendre la décision d'acheter des services, l'entreprise doit conduire une analyse de coût-bénéfice afin d'évaluer le pour et le contre de la décision d'achat ou de construction de nouveaux services.

---

[13] Ce qui constitue une stratégie ascendante ! (ou une tactique ?)

III.4.2.2 Stratégie de développement ascendante

Une stratégie de développement ascendante part des technologies de base pour aller vers la modélisation des besoins et des processus d'affaires. Elle construit des services au dessus des systèmes existants (systèmes hérités). La première étape de cette approche consiste à analyser les transactions et les modules des systèmes hérités tel que les applications ERP.

Par la suite, cette stratégie inclut deux différentes activités (Perepletchikov et al., 2005) :

(1) Les développeurs peuvent ajouter une couche de services au dessus des systèmes hérités en créant des classes enveloppantes (*wrappers*[14]) et des adaptateurs pour les logiciels des systèmes hérités.

(2) Les systèmes hérités peuvent être réusinés (*refactoring*) de façon à ce que le comportement externe du code reste le même, alors que sa structure interne devient axée sur le service.

Principales tâches pour améliorer la qualité logicielle

Une tâche importante dans cette approche de développement est l'utilisation de métriques de qualité pour mesurer les propriétés structurales des systèmes hérités[15], afin de décider s'il serait préférable de réusiner le système, ou d'ajouter une couche de services. Il est aussi important de prendre en considération les modèles des processus d'affaires lors de la détermination des services nécessaires.

---

[14] Le *wrapping* est le style de modernisation le plus utilisé, surtout lorsqu'il s'agit d'applications sur lesquelles l'entreprise a beaucoup investi (Kavianpour, 2007). D'ailleurs, Les fournisseurs de solutions axées sur les services proposent des outils (IDE – *Integrated Developement Environment*) pour le *wrapping* des composants hérités CORBA, COBOL, .NET, J2EE, Java, et C++ en services Web.

[15] Ce qui est faisable seulement dans une entreprise assez mature qui a depuis un certain temps adopté des métriques et qui a une historique de la qualité qui facilite les prises de décision.

Les tâches les plus importantes lors du *refactoring* sont les suivantes :

- Adopter une approche itérative, en focalisant sur les modules fortement couplés et hautement complexes. Ceci permettra de mesurer le RCI avant de d'initier de grands changements et d'acquérir de l'expérience avant de s'attaquer aux grands problèmes.

- Engager du personnel expérimenté dans la conception, la mise en œuvre et l'architecture des systèmes hérités, afin de réduire le coût de développement.

Les tâches les plus importantes lors de l'ajout d'une couche de services sont les suivantes :

- Considérer l'achat d'adaptateurs/*wrappers* pour les logiciels. Ceci a pour effet d'augmenter les coûts en infrastructure, mais diminue considérablement l'effort de développement.

- Viser le développement de services à granularité fine car c'est le facteur qui a le plus d'influence sur les propriétés de qualité interne des services : il augmente la cohésion et diminue la complexité et le couplage[16].

III.4.2.3 Stratégie de développement de compromis

La stratégie de compromis est une combinaison des stratégies de développement descendante et ascendante. À présent les techniques de cette approche n'ont pas été assez traitées par les recherches. La seule stratégie de compromis bien décrite est celle d'Arsanjani (2004) qui est présentée comme une technique de modélisation axée sur les services.

Tel que déjà énoncé dans la section II.4.1.2, dans cette technique, la fonctionnalité des processus d'affaires de haut niveau est utilisée pour former des services à forte granularité. En examinant la fonctionnalité héritée existante et en décidant de la façon de créer les

adaptateurs et les *wrappers*, il est possible de spécifier des services à granularité fine. Finalement, une approche transversale peut être appliquée afin de réduire le nombre de services candidats qui ont été précédemment identifiés. Cette technique comprend aussi des indicateurs de performance et des métriques pour les services.

Principales tâches pour améliorer la qualité logicielle

Les tâches à accomplir pour l'amélioration des attributs de qualité logicielle dans cette approche de développement sont une combinaison des tâches décrites pour les stratégies de développement ascendante et descendante.

II.4.3 La gestion de projet axée sur les services

Un projet de développement logiciel typique est composé du processus de développement de l'application, de la gestion du projet et des technologies utilisées (Mittal, 2006). Après avoir abordé la question du processus de développement à travers les différentes approches et stratégies de développement logiciel axé sur les services (sections II.4.1 et II.4.2), nous allons nous intéresser aux aspects de la gestion de projet dans un environnement de développement axé sur les services, qui ont été soulevés par les recherches actuelles.

Erl (2005) présente les meilleures pratiques pour la planification de projets axés sur les services Web, incluant :

1. savoir quand utiliser les services Web,
2. savoir comment utiliser les services Web,
3. savoir quand éviter les services Web,
4. construire sur ce que l'on possède déjà,

---

[16] Ce qui est, bien sûr, fort loin d'être automatique !

5. comprendre les limites des systèmes hérités,

6. préparer le budget des dépenses qui suivent l'introduction des services Web dans une entreprise,

7. aligner le rendement du capital investi avec les stratégies de migration.

Cependant, les meilleures pratiques ne sont pas liées à des rôles particuliers, comme nous essayons de faire dans notre étude.

Krafzig et al. (2004) soulignent que l'incorporation des principes de l'axé sur les services dans une entreprise n'implique pas nécessairement l'utilisation d'une nouvelle méthode de gestion de projet, mais nécessiterait plutôt l'amélioration de la méthode existante. Voici en bref les aspects les plus importants de cette amélioration qui ont été cités par Krafzig et al. (2004) :

Utiliser les services comme éléments de contrôle de projet

Une des grandes difficultés de la gestion de projet logiciel est la mise en correspondance entre les éléments de contrôle de projet (ex. tâches, organigramme technique de projet, etc.) et les artefacts logiciels (ex. code de programme, modèle de données, spécifications, etc.). Les services sont les meilleurs artefacts pouvant être utilisés comme des éléments de contrôle de projet.

- d'abord, ils fournissent un niveau de granularité idéal, par rapport aux modules et aux classes, pour être utilisés comme des éléments de contrôle de projet (voir figure ci-dessous).

- ensuite, mis à part les services TI, les services tendent à être relativement axés sur les activités de l'organisation. Ceci en fait un outil de communication idéal les individus (techniques et non techniques) impliqués dans le projet.

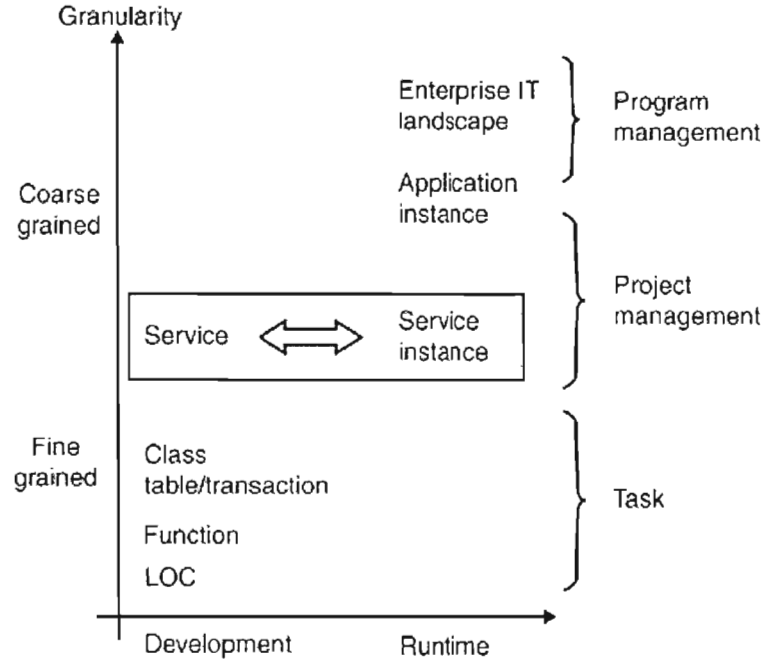


Figure 2.21 Différents niveaux de granularité d'artefacts logiciels durant le développement et durant l'exécution (Source : Krafzig et al., 2004)

Appliquer une approche de développement fil léger *(Thin Thread)*

Une approche fil léger consiste à développer individuellement plusieurs parties fonctionnelles complètes du système (appelées *fils*). Un fil inclut toutes les couches du système. L'utilisation des contrats de service impliqués dans une approche fil léger permet de mieux diriger les processus de développement et de test.

Utiliser les contrats de service en tant qu'outil de gestion de projet

Généralement, dans un environnement axé sur les services, le conseil SOA est chargé de la synchronisation des services qui sont partagés par différents projets ou sous-projets. Le conseil SOA doit aussi inciter à l'utilisation des contrats de service en tant qu'outils de

gestion de projet. Les contrats de service peuvent servir de base pour mener les aspects les plus importants d'un projet, incluant (voir figure ci-dessous) :

- L'estimation des coûts et du temps du projet,
- La planification des itérations et du développement du projet,
- La planification des tests et du déploiement.

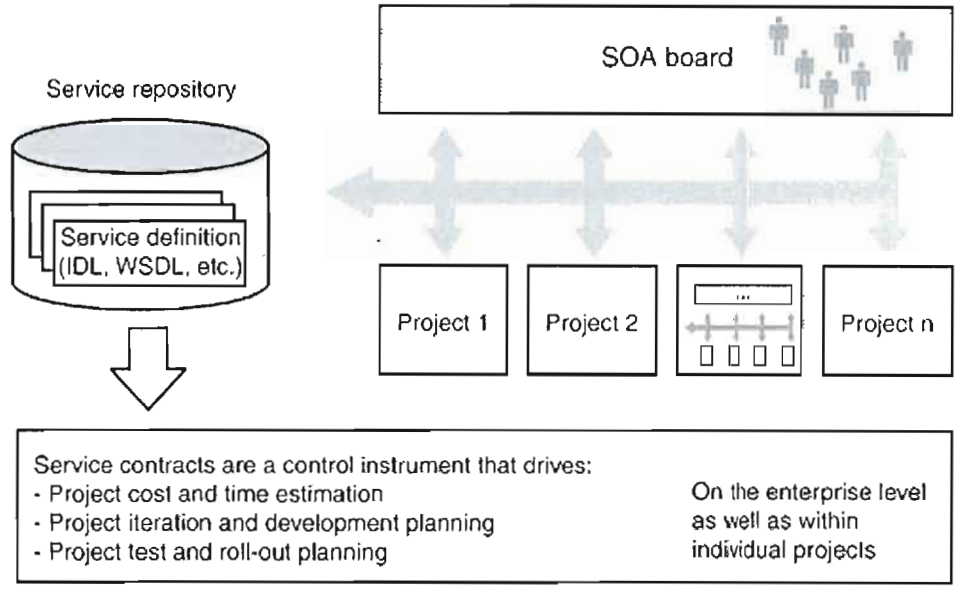


Figure 2.22 Le conseil SOA utilise les contrats de service pour coordonner plusieurs projets ou sous-projets (Source : Krafzig et al., 2004)

II.4.4 Les principes de la gouvernance SOA

En raison de l'importance qui lui a été accordée dans la littérature SOA, nous ne pouvions présenter les pratiques du développement axé sur les services sans aborder la gouvernance SOA.

La gouvernance SOA est définie comme étant l'ensemble des processus et des règles qui permettent de s'assurer de l'atteinte des objectifs de la SOA mise en œuvre. Dans la

littérature SOA, on conseille la mise en place d'un groupe spécifique chargé de définir les pratiques de la gouvernance SOA, spécialement dans les initiatives SOA à grande envergure (Balzer, 2004 ; Marks et Bell, 2006). Par exemple, il aurait pour responsabilité de (Marks et Bell, 2006) :

- Déterminer la supervision de l'architecture SOA (ex. Qui possède les normes et le contrôle de la conformité aux directives SOA ? Qui détermine les niveaux appropriés de granularité des services ?).
- Établir des directives SOA (ex. directives de conformité aux objectifs et aux standards SOA, directives de sécurité).
- Mettre en œuvre le processus de gouvernance SOA (ex. gestion des interdépendances entre les services partagés, gestion des conflits entre les organisations).
- Gouverner la définition, la création et la publication des services (ex. comment les services seront définis, développés et modifiés ? Qui aura la responsabilité de la conception ? En se basant sur quelles exigences ? Qui possède les services ?)
- Établir des politiques et des processus pour la gestion de la qualité des services (ex. Qui se charge de renforcer les contrats de service ? Quelles sont les technologies qui pourront permettre de renforcer les directives et de mettre en œuvre la gestion des services ?).

II.4.5 Les rôles du développement axé sur les services

Même si la littérature concernant SOA est très abondante, rares sont les recherches qui définissent clairement les nouveaux rôles qui sont apparus avec l'émergence du développement axé sur les services. Néanmoins, les travaux de Bieberstein et al. (2005) et Kajko-Mattsson et al. (2007) ont introduit les aspects relatifs aux rôles et aux tâches lors du développement de systèmes basés sur une SOA.

Bieberstein et al. (2005) ont identifié de nouveaux rôles ajoutés en présence d'une SOA, mis à part les rôles existants. Les rôles existants sont :

- le gestionnaire de projet TI,
- l'analyste d'affaires,
- l'architecte,
- le développeur,
- le spécialiste de la sécurité,
- l'administrateur de système et de base de données,
- le spécialiste du déploiement de service,
- le spécialiste des tests et de l'intégration de services,
- le développeur d'outils,
- le facilitateur de transfert des connaissances.

Les nouveaux rôles sont :

- l'architecte SOA,
- le concepteur de service,
- le concepteur de flux de processus,
- le développeur de services,
- le spécialiste de l'intégration,
- le testeur d'interopérabilité,

- l'administrateur UDDI,
- le spécialiste de la gouvernance des services,
- le gestionnaire de projet SOA,
- l'administrateur de système SOA.

Kajko-Mattsson et al. (2007) proposent un ensemble initial de rôles pour le développement, l'évolution, la maintenance et le support des systèmes basés sur une SOA. Ils regroupent ces rôles en quatre catégories (voir figure ci-dessous) qui sont :

1. Les rôles de support SOA
2. Les rôles de stratégie et de gouvernance SOA
3. Les rôles de conception et de gestion de la qualité SOA
4. Les rôles de développement et d'évolution SOA

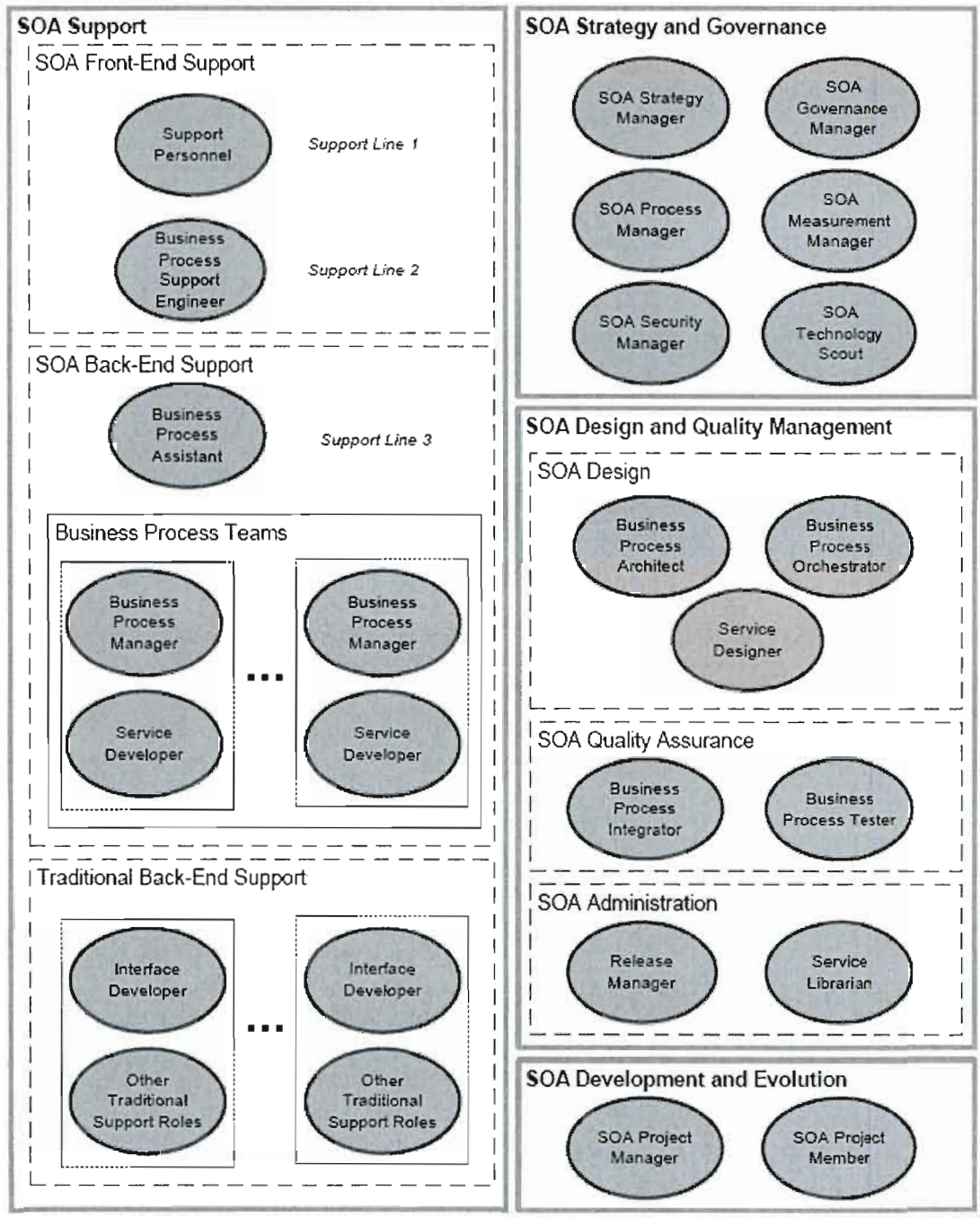


Figure 2.23 Les rôles nécessaires au développement, à l'évolution et à la maintenance des systèmes fondés sur une SOA (Source : Kajko-Mattsson et al., 2007)

Dans ce qui suit, nous allons donner un aperçu de chacun de ces quatre catégories de rôles en nous basant sur la description proposée par M. Kajko-Mattsson et al. (2007).

a- Les rôles de support SOA

Ce groupe de rôles est responsable du support déployé pour les systèmes fondés sur SOA. Il se divise en support SOA frontal, support SOA dorsal et support dorsal traditionnel.

Support SOA frontal : ce groupe de rôles est responsable du support apporté aux applications basées sur SOA qui sont en contact direct avec les clients. Les individus responsables de ces rôles ont besoin de connaître les processus d'affaires et leur structure sous-jacente, de façon à ce qu'ils puissent fournir un support aux clients dans leurs opérations journalières. Les rôles spécifiques incluent le personnel de support et le personnel de support des processus d'affaires.

Support SOA dorsal : ce groupe est responsable de la création, de l'évolution, de la maintenance et de la réutilisation des services et des processus d'affaires. Les individus responsables de ces rôles possèdent une grande connaissance des processus d'affaires et des services qui les mettent en œuvre. Les rôles spécifiques incluent l'assistant de processus d'affaires, le gestionnaire des processus d'affaires et le développeur de services.

Support Dorsal Traditionnel : ce groupe est responsable de la création, de l'évolution et de la maintenance des interfaces des systèmes traditionnels qui fournissent les fonctionnalités nécessaires aux services. Le rôle principal dans ce groupe est celui du développeur d'interface qui est responsable de la conception, de la mise en œuvre et du test de l'interface destinée aux systèmes traditionnels de l'organisation.

b- Les rôles de stratégie et de gouvernance SOA

Ce groupe de rôles est responsable de la gestion et de la gouvernance des systèmes fondés sur SOA selon une stratégie SOA. Il s'assure que les besoins en affaires sont satisfaits selon un certain niveau stratégique, tactique et opérationnel. Ce groupe possède aussi une responsabilité et une autorité au travers de l'entreprise à prioriser les processus d'affaires. Les individus responsables de ces rôles doivent posséder une expertise et une compréhension du rôle de SOA dans le contexte organisationnel. Les rôles spécifiques incluent le gestionnaire de stratégie SOA, le gestionnaire de gouvernance SOA, le gestionnaire de processus SOA, le gestionnaire de mesures SOA, le gestionnaire de sécurité SOA et la vigie de la technologie SOA.

c- Les rôles de conception et de gestion de la qualité SOA

Ce groupe de rôles est responsable de la modélisation et de l'architecture des processus d'affaires, de leur intégration, de l'assurance de leur interopérabilité et de leur qualité, ainsi que de la gestion du registre des services et de l'administration des systèmes basés sur une SOA qui ont été mis en œuvre. Il se divise en conception SOA, assurance qualité SOA et administration SOA.

Conception SOA : ce groupe est responsable de la modélisation et de l'architecture des processus d'affaires et de leur liaison aux nouveaux services et aux services existants. Les rôles spécifiques incluent l'architecte des processus d'affaires, l'orchestrateur de processus d'affaires et le concepteur de service.

Assurance Qualité SOA : ce groupe est responsable de l'intégration des processus d'affaires et de l'assurance de leur interopérabilité et de leur qualité. Les rôles spécifiques incluent l'intégrateur des processus d'affaires et le testeur des processus d'affaires.

Administration SOA : ce groupe est responsable de l'administration des systèmes fondés sur SOA et de leur réalisation. Les rôles spécifiques incluent le bibliothécaire des services et le gestionnaire de la configuration.

d- Rôles de développement et d'évolution SOA

Dans le contexte d'une SOA d'entreprise, le développement et les changements majeurs apportés aux processus d'affaires et à leurs composants sont conduits dans des projets. Pour cette raison, les rôles suivants sont nécessaires : le gestionnaire de Projet SOA, responsable de la gestion et du contrôle des projets, ainsi que de la définition de plans de projet et de leur mise en œuvre et les membre des projet SOA, qui peuvent être composé d'un groupe de différents rôles venant de différentes équipes de processus d'affaires, de concepteurs SOA, d'assurance qualité SOA et d'autres groupes de rôles.

D'après Kajko-Mattsson et al. (2007), tous les rôles qui viennent d'être présentés sont nécessaires au développement, à l'évolution, à la maintenance et au support des systèmes basés sur une SOA. Cependant, si on est en présence d'un environnement de développement distribué (ex. des services qui sont externes à une organisation développant un systèmes fondé sur SOA, ou une organisation qui est strictement un fournisseur de service avec un ensemble connu de demandeurs de service), ils doivent être adaptés. En raison de la pauvreté des tâches accomplies par certains des rôles proposés par Kajko-Mattsson et al. (2007), nous pensons que certains d'entre eux sont plutôt des activités camouflées en rôles. Néanmoins, leur classification peut être utile pour éliciter les différentes activités accomplies lors du développement de systèmes basés sur une SOA.

II.5 Conclusion

Le nombre d'articles et de livres publiés durant les cinq dernières concernant SOA est si important et en croissance constante, que la revue de la littérature qui vient d'être présentée ne peut être exhaustive. Nous avons seulement souhaité présenter les concepts fondamentaux

liés aux paradigmes de l'axé sur les services (SOSE et SOA) et ceux qui sont le plus en rapport avec les défis et les pratiques du développement axé sur les services.

CHAPITRE III

ACTIVITÉS, RÔLES ET TÂCHES SOA

Dans ce chapitre, nous présentons une synthèse des différents éléments de notre revue de la littérature qui sont en relation directe avec notre objectif de recherche. Rappelons que l'objectif de notre recherche est de savoir si le passage vers un environnement axé sur les services pour une organisation demande des changements aux *rôles* et aux *tâches* du CVP. La littérature qui a été considérée dans ce chapitre est la suivante :

(1) La norme IEEE/EIA std 1074 – 2006 qui donne une méthode pour définir des processus pour le CVP.

(2) La littérature qui définit les activités, les rôles et les tâches impliqués dans une initiative SOA et dans le cycle des projets qui sont élaborés dans le cadre de cette initiative (projets SOA).

La norme IEEE/EIA std 1074 classe les activités impliquées dans le CVP de tout projet de développement logiciel sous cinq groupes d'activités :

1. **Groupe d'activités de gestion de projet :** ce groupe inclut les activités qui initient et contrôlent un projet de développement logiciel tout au long de son cycle de vie.

2. **Groupe d'activités de pré-développement :** ce groupe inclut les activités qui explorent et allouent les exigences du système avant que le développement logiciel ne débute.

3. **Groupe d'activités de développement :** ce groupe inclut les activités qui sont accomplies durant le développement et l'amélioration du projet logiciel.

4. **Groupe d'activités de post-développement :** ce groupe inclut les activités d'installation, d'opération, de support, de maintenance et de retrait d'un produit logiciel.

5. **Groupe d'activités de support :** ce groupe inclut les activités qui sont nécessaires au succès de l'accomplissement des activités du projet. Ces activités sont utilisées pour assurer la complétude et la qualité des fonctions du projet.

Ce chapitre se présente de la façon suivante. Dans la première section (section III.1), nous définissons la terminologie utilisée. Dans la deuxième section (section III.2), nous présentons les différentes activités SOA qui ont été identifiées à partir de la littérature, réparties selon la classification proposée par la norme IEEE/EIA std 1074 pour les groupes d'activités du CVP. Dans la troisième section (section III.3), nous décrivons les différents rôles SOA et leurs tâches associées, identifiés à partir de la littérature. Finalement, ce chapitre termine par une conclusion (section III.4).

III.1 Terminologie

Cette section définit la terminologie que nous allons utiliser dans le reste du document. Par la suite, elle présente un diagramme UML (voir figure ci-dessous) qui relie les différents éléments de la terminologie.

Processus d'affaires **:** Un processus d'affaires peut être vu comme un enchaînement de tâches au cours duquel des procédures et des informations sont traitées ou exécutées successivement en vue de réaliser un produit ou de fournir un service (OQLF, 2007).

Tâche : Une tâche est une unité de travail conceptuelle faisant partie d'un processus d'affaires (Chang et Kim, 2007). Une tâche peut être décomposée en des tâches de plus faible granularité (sous-tâches).

Service : Il s'agit d'un élément logiciel dont l'exécution accomplit une tâche, de n'importe quelle granularité, faisant partie d'un processus d'affaires (Chang et Kim, 2007). La différence entre une tâche et un service est que la tâche est une unité conceptuelle définie selon une perspective d'affaires, et qu'un service est l'activité qui lui correspond définie selon une perspective d'ingénierie logicielle. Quelques services peuvent être communs à différents processus d'affaires. Ils sont donc réutilisables à travers plusieurs processus d'affaires.

Interface de service/Contrat de service : L'interface (ou contrat) de service décrit le service et les fonctionnalités offertes par ce dernier. Elle contient toute l'information utile à l'utilisation du service (pré-conditions, post-conditions, données en entrée, ...) et sert à garder la mise en œuvre du service hors de portée des clients du service (Newcomer et Lomow, 2004). Elle peut, par exemple, être décrite avec CORBA IDL, WSDL, ou un format XML *ad hoc* (Krafzig et al., 2004)

Composant de service : Élément logiciel qui met en œuvre un service (Chang et Kim, 2007). Dans le paradigme de l'axé sur les services, on distingue clairement les deux niveaux logique et physique : le service est le concept du niveau logique, le composant est le concept physique de mise en œuvre (Bonnet, 2005). Ainsi, à la différence du service, le composant de service représente une entité logicielle physique et exécutable. Certains composants de services sont similaires à ceux du développement basé sur les composants (par exemple, dans l'infrastructure J2EE, un composant est un EJB, un JavaBeans, une Servlet ou une RMI) ; d'autres composants de service peuvent simplement être des classes enveloppantes (*wrappers*) d'applications héritées (Bonnet, 2005). Ces deux types de composants sont mis en œuvre d'une façon différente, mais tous les deux doivent fournir des interfaces physiques, appelées interfaces de composants.

Interface de composant : Les interfaces de composants sont conformes aux interfaces de services (Chang et Kim, 2007). Par exemple, les composants de service peuvent être mis en œuvre en tant qu'EJB et fournissent des interfaces physiques sous la forme d'interfaces EJB *Home* ou *Remote* conformes aux interfaces de services WSDL (Chang et Kim, 2007).

Tel qu'illustré par le diagramme ci-dessous, un *processus d'affaires* est composé d'une ou plusieurs *tâches*, chacune étant accomplie par un *service*. Chaque *service* est mis en œuvre par un *composant de service* conforme à une *interface de composant*, et est décrit à l'aide d'une *interface/contrat de service*, qui elle-même correspond à une *interface de composant*.

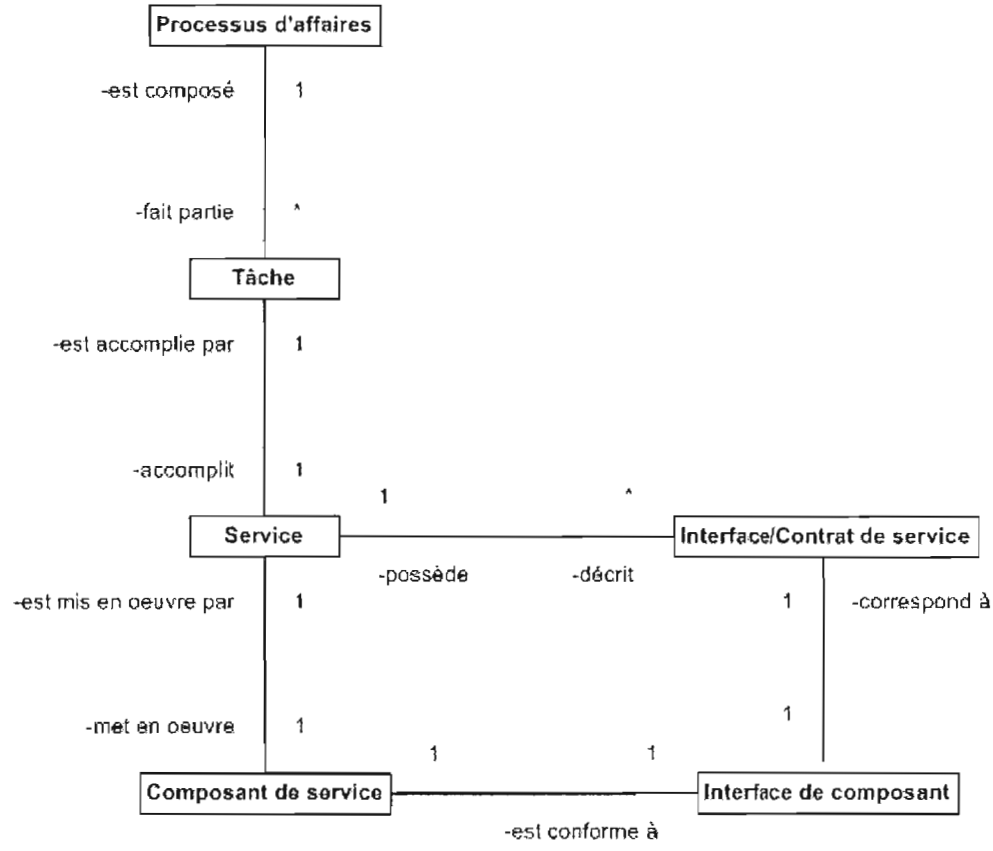


Figure 3.1 Terminologie utilisée

III.2 Activités SOA

Cette section présente les activités qui sont impliquées dans le CVP des projets SOA élaborés dans le cadre d'une initiative SOA. Tel que déjà énoncé, ces activités ont été réparties selon les cinq groupes d'activités définis par la norme IEEE/EIA std 1074.

III.2.1 Activités SOA de gestion de projet (Groupe A)

Les activités qui sont impliquées dans une initiative SOA et qui font partie du groupe d'activités de gestion de projet sont les suivantes :

A.1. Former un comité SOA : le comité SOA est une entité organisationnelle responsable de l'initiative SOA et du contrôle de son application dans les projets en-cours (Krafzig et al., 2004). Il doit aussi s'assurer que tous les acteurs de l'organisation ne s'opposent pas à l'initiative SOA mais plutôt la soutiennent. Étant donné que l'un des objectifs fondamentaux de SOA est de construire des services réutilisables à travers toute l'entreprise, il est insensé qu'un département ou un secteur d'activité unique puisse diriger à lui seul les projets SOA, sans impliquer d'autres groupes dans l'organisation (Bloomberg et Schmelzer, 2006). Le comité SOA devrait donc représenter tous les employés de l'entreprise. De préférence, les individus choisis doivent avoir une bonne connaissance de l'organisation en question (Krafzig et al., 2004).

A.2. Développer une stratégie SOA : la stratégie SOA est élaborée grâce à la définition de la vision, de l'étendue, de la mission et des objectifs de haut niveau de l'initiative SOA. Cette activité n'est pas triviale lorsque l'organisation se situe au début de son initiative SOA (Marks et Bell, 2006). Dans ce cas, l'organisation peut commencer par identifier des processus d'affaires existants ou envisagés, qui contiennent des tâches redondantes et qui peuvent donc donner lieu à des services réutilisables. Il vaut mieux commencer avec des processus d'affaires simples, non primordiaux afin de permettre aux projets pilotes SOA de démontrer un RCI rapide et vérifiable (Bloomberg et Schmelzer, 2006).

A.3. Établir une analyse RCI de la stratégie SOA développée : cette analyse doit montrer les coûts et les économies qui seront réalisés par l'infrastructure SOA à court, moyen et long terme (Mittal, 2006).

A.4. Définir des critères d'acceptation spécifiques au projet SOA : des critères d'acceptation spécifiques, qui agissent comme des objectifs que l'équipe doit poursuivre tout en planifiant et exécutant le projet, doivent donc être définis pour le projet SOA. Les critères varient d'une compagnie à une autre et d'un projet à un autre, selon l'objectif qu'essaie d'atteindre la compagnie avec son initiative SOA. Mais les critères d'acceptation communs à tous les projets SOA sont (Bloomberg et Schmelzer, 2006) :

- Une conception architecturale complète : la conception doit être adéquate aux besoins spécifiés pour le projet et ne doit pas seulement inclure le plan de départ, mais aussi des détails à propos des contrats de services et des autres éléments de mise en œuvre de SOA.

- Des services réutilisables : les critères d'acceptation doivent spécifier le niveau de réutilisabilité des services à atteindre, incluant le nombre de services et le nombre de clients par service.

- Des critères de gouvernance : l'initiative SOA doit inclure une infrastructure de gouvernance (voir activité A.8) qui supporte les politiques et les règles pour l'évolution et la gestion ultérieures de la mise en œuvre de SOA. Définir des critères d'acceptation pour l'infrastructure de gouvernance est important pour le succès des projets futurs, qui vont se greffer sur l'infrastructure SOA.

A.5. Établir un plan de communication : il s'agit d'un document qui explique comment l'équipe SOA (ex. les analystes et les architectes) va communiquer avec les autres parties prenantes (Mittal, 2006). Ce plan de communication assure que toutes les équipes de projet, ainsi que les utilisateurs comprennent et sont impliqués dans le processus.

*A.6. **Établir des prévisions budgétaires sur l'infrastructure technique :*** il s'agit de définir l'infrastructure technique nécessaire au soutien de l'initiative SOA dans son ensemble. Cette activité inclut des prévisions budgétaires détaillées sur les serveurs, l'infrastructure du réseau et les besoins en formation (Mittal, 2006).

*A.7. **Définir un plan de projet :*** créer un plan de l'ensemble du projet qui inclut les jalons et les dépendances, ainsi qu'un échéancier et des prévisions budgétaires (Mittal, 2006). Les cycles de développement planifiés dans un projet SOA sont plus courts[17] que ceux qui sont planifiés pour les autres types de projets (Mittal, 2006b).

*A.8. **Définir une infrastructure de gouvernance SOA :*** l'initiative SOA doit inclure une infrastructure de gouvernance qui supporte les politiques et les règles pour l'évolution et la gestion ultérieure des mises en œuvre SOA (Bloomberg et Schmelzer, 2006). Cette infrastructure définit les processus, les normes et les outils qui seront utilisés tout au long du cycle de vie des services, c.-à-d. de la création des services jusqu'à leur utilisation et leur modification au cours du temps. Un comité spécifique peut être chargé de définir les meilleures pratiques de la gouvernance SOA. Par exemple, il aurait pour responsabilité de spécifier (Hurwitz et al., 2007) :

- qui est autorisé à changer un service ;
- qui alerter et quels sont les impacts sur les autres services en cas de changement apporté à un service ;
- les normes pour les noms des services ;
- comment l'organisation vérifie la qualité et la performance du service ;
- le processus de suivi des problèmes.

---

[17] Si on adopte des méthodes agiles.

III.2.2 Activités SOA de pré-développement (Groupe B)

B.1. Modéliser les processus d'affaires : les processus d'affaires sélectionnés pour le projet SOA sont modélisés sous forme de *business use cases*. Un *business use case* (BUC) est une description du processus d'affaires (Kim et Doh, 2007). Les BUC peuvent être spécifiés en détail conformément à un patron de cas d'utilisation qui inclut le nom, la description, les acteurs impliqués, l'objectif dans le contexte, l'étendue, le niveau, les pré-conditions, les post-conditions, les déclencheurs, le scénario principal de succès, les extensions, les sous-scénarios, la priorité, la fréquence, la performance, etc. (Stojanovic et al, 2004). Les modèles de cas d'utilisation établis seront utilisés pour définir les services (Kim et Doh, 2007).

B.2. Effectuer une analyse de domaine : l'analyse de domaine définit un vocabulaire de domaine pour le système en cours de développement, ex. information concernant des concepts d'affaires qui doivent être pris en considération par le système, avec leurs attributs et leurs relations (Stojanovic et al., 2004). Suite à cette analyse, qui peut être menée avec différentes techniques[18], les types d'information nécessaires à l'accomplissement des BUC définis durant l'activité précédente[19] sont définis.

B.3. Identifier les services : chaque BUC peut être décomposé en plusieurs tâches, chacune possédant un ensemble d'étapes de traitement particulier. Les services sont identifiés par l'analyse de ces tâches (Kim et Yun, 2006) en utilisant des techniques[20] qui ont pour objectif d'identifier des services élémentaires réutilisables ayant une granularité optimale.

---

[18] La technique des « *Class Responsabilities Collaborators Cards* » (CRC *Cards*) constitue un exemple de technique d'analyse de domaine (Brown et al., 2007). Elle consiste à établir les classes d'analyse en fonction de leurs responsabilités et de leurs collaborations. Les cartes (simplement des papiers cartonnés de format A5 par exemple) sont disposées sur une table, en fonction des collaborations entre classes. Cette technique insiste sur la communication entre les membres de l'équipe. Une session CRC est, en pratique, une réunion où les participants sont réunis autour d'une table, focalisés sur le sujet de l'étude, en l'occurrence un modèle basé sur le concept de classes.

[19] Il est clair que ce « précédent » n'est pas un absolu et que dans certains cas l'analyse du domaine peut précéder le BUC.

[20] Kim et Doh (2007) parlent d'une technique qui consiste à générer des arbres de tâches (*task trees*) à partir des modèles de cas d'utilisation établis dans l'activité précédente. Des règles de réusinage

B.4. Définir les interfaces de services : les interfaces de services sont définies dans le but de raffiner les services identifiés et de clarifier le rôle et les fonctions de chaque service (Kim et Doh, 2007). Elle inclut le nom, une explication simple du service, sa stratégie de mise en œuvre, ses opérations, ses exigences non fonctionnelles et la technologie qui sera adoptée par le service (Kim et Doh, 2007). L'analyse des interactions et de la collaboration entre les différents services peut mener à la modification ou à l'élimination de certains services et à la création de nouveaux services. L'application répétée de ce raffinement produit une liste finale de services.

B.5. Effectuer une analyse de coût-bénéfice : les services identifiés pour le projet SOA peuvent être construits ou achetés. Avant de prendre la décision d'acheter des services, l'entreprise doit conduire une analyse de coût-bénéfice afin d'évaluer le pour et le contre de la décision d'achat ou de construction de nouveaux services (M. Perepletchikov et al., 2005).

B.6. Analyser les systèmes hérités : cette activité a pour objectif d'identifier les systèmes hérités qui peuvent être capables d'accomplir la fonctionnalité des services identifiés. Si des candidats existent, les services concernés seront réalisés à partir des systèmes hérités (voir activité C.4).

B.7. Élaborer un document d'architecture SOA : définir et documenter l'architecture SOA dans son ensemble, en incluant les composants matériels et logiciels (Mittal, 2006).

III.2.3 Activités SOA de développement (Groupe C)

C.1. Concevoir l'architecture des composants de services : l'architecture des composants de services peut être construite en analysant les dépendances entre les services identifiés

---

(décomposition, équivalence, généralisation, fusion et suppression) sont par la suite appliquées sur les arbres de tâches, dans le but de raffiner les modèles des cas d'utilisation et d'en dériver des services avec une granularité optimale et un bon niveau d'abstraction.

(Kim et Doh, 2007). L'architecture ainsi générée est par la suite raffinée en utilisant des patrons de conception.

C.2. Élaborer un document de conception détaillée : pour les services à construire, décrire dans un document de conception détaillée la façon avec laquelle ces derniers seront conçus et construits (Mittal, 2006).

C.3. Spécifier les interfaces/contrats de services : l'interface de service peut, par exemple, être décrite avec CORBA IDL, WSDL, ou un format XML sur-mesure (Krafzig et al., 2004). Elle doit contenir toute l'information utile à l'utilisation du service. Généralement, ceci inclut les pré-conditions pour l'utilisation du service, les cas particuliers, les erreurs potentielles, les informations de gestion des versions, les droits d'accès et les opérations. De plus, pour chaque opération de l'interface de service il faut spécifier les pré-conditions, les post-conditions, le profil des données en entrée, le profil des données en sortie, le profil d'interaction et les SLA (*Service Level Agreement*) (Newcomer et Lomow, 2004). Les SLA définissent par exemple la disponibilité, le temps de réponse, le délai moyen entre les pannes, etc.

C.4. Réaliser les services à partir des systèmes hérités : cette activité peut être accomplie par deux différentes techniques : le *wrapping* ou le *refactoring* (réusinage). La première technique consiste à décomposer l'application monolithique existante, à sélectionner les parties du code qui mettent en œuvre la fonctionnalité du service visé et à les exposer en tant que composant de services en utilisant des classes enveloppantes (*wrappers*) (Chen et al., 2005). Le réusinage des systèmes hérités, quant à lui, modifie[21] la structure interne du code pour qu'elle soit axée sur les services, alors que le comportement externe du code reste le

[21] Les modifications peuvent inclure la réécriture des définitions de données, la suppression des redondances des données, la consolidation et la migration des données vers des bases de données relationnelles, la simplification du code, la suppression des redondances du code, les améliorations de performance, etc. (Kavianpour, 2007).

même (Perepletchikov et al., 2005). De cette façon, les systèmes hérités seront restructurés en composants de services.

C.5. Développer de nouveaux composants de services : les développeurs sont guidés par les contrats de services pour le développement des composants de services. Le développement axé sur le service a une nature fondamentalement itérative. Les développeurs doivent effectuer des itérations continues sur le code pour satisfaire les besoins spécifiés par les contrats et les changements qui y sont apportés (Krafzig et al., 2004). Des IDE proposés par des fournisseurs de solutions logicielles offrent de l'assistance, des meilleures pratiques et des normes pour la construction des composants de services.

C.6. Créer un registre[22] de services : le registre SOA détient l'information concernant chaque service qui a été approuvé pour l'utilisation et qui est passé en production selon les règles de gouvernance SOA. Il est important de s'assurer que le répertoire est l'unique source d'information concernant les services. Chaque fois qu'une nouvelle application est conçue et développée, la consultation des descriptions de services dans le registre doit être suffisante pour décider si des services existants peuvent être utilisés pour construire la nouvelle fonctionnalité et de la manière de les utiliser (Krafzig et al., 2004). Les fonctions principales du registre sont de publier les services, de permettre la découverte des services (durant la conception et durant l'exécution) et de gouverner l'utilisation en temps-réel des services. Son contenu varie, mais ce qui suit représente l'information que devrait contenir un registre concernant un service qui y est enregistré (Hurwitz et al., 2007) :

- Une description des fonctionnalités offertes par le composant ;
- Une description complète de l'interface ou des interfaces du service ;
- La localisation du programme/composant exécutable ;

---

[22] Dans la littérature SOA, une ne grande confusion existe entre les deux expressions « répertoire de service » et « registre de service ». D'après Hurwitz et al. (2007), le répertoire constitue le système d'enregistrement au complet, et le registre l'enregistrement en temps réel d'un domaine SOA.

- Une spécification des niveaux de service (disponibilité, temps de réponse, etc.) définis pour le service ;
- Une définition des droits d'utilisation du service – essentiellement qui possède quel type d'accès au service et sous quelles conditions ;
- Une définition des règles de sécurité qui gouvernent l'utilisation du service.

C.7. Composer les services : durant cette activité, les services sont assemblés pour construire l'application (ou les applications) composée(s) qui constituent le produit final du projet SOA. L'assemblage ou la composition des services est effectué grâce au mécanisme d'orchestration, qui permet de faire collaborer de manière efficace plusieurs services en définissant des services composites (Joffroy et al, 2007). L'orchestration des services peut être définie en utilisant par exemple le langage BPEL (*Business Process Execution Language*).

C.8. Utiliser une infrastructure d'intégration des services : lors de leur composition, les services sont connectés à travers un bus abstrait d'interconnexion, ex. un ESB - *Enterprise Service Bus*. L'ESB constitue une infrastructure d'intégration qui permet la communication entre les services. Il n'assure pas seulement l'interconnectivité et la liaison, mais aussi la gestion et le contrôle des interactions entre les services pour assurer une qualité de service conforme aux contrats de services (Papazoglou, 2007 ; Tsai et al., 2006). Les industriels proposent des ESB incluant des outils d'orchestration de services et d'exécution d'orchestration basés sur BPEL (Papazoglou, 2007).

III.2.4 Activités SOA de post-développement (Groupe D)

D.1. Élaborer des diagrammes de déploiement de l'infrastructure : ces diagrammes décrivent les systèmes d'exécution et les plateformes nécessaires au déploiement des applications du projet SOA (Bloomberg et Schmelzer, 2006 ; Mittal, 2006). Les plateformes incluent le matériel (le réseau, les serveurs, etc.) ainsi que les systèmes d'exploitation, les

middlewares et les autres logiciels d'infrastructure. Durant cette activité, les services représentent des métadonnées que les systèmes doivent gérer et maintenir (Bloomberg et Schmelzer, 2006).

D.2. Déployer l'application (ou les applications) composée(s) : l'application composée est déployée pour l'exécution dans l'environnement de production (Tsai et al., 2006).

D.3. Effectuer la maintenance : durant l'exécution, des points d'intérêts prédéterminés sont contrôlés et des données sont collectées de façon à ce que la performance du système et le comportement de l'application puissent être évalués (Tsai et al., 2006). Si la performance n'est pas satisfaisante ou que des erreurs sont détectées, une reconfiguration et une recomposition peuvent être accomplies en changeant les liaisons vers de meilleurs services ou en raffinant la spécification de la composition (Tsai et al., 2006). L'application recomposée doit être vérifiée et validée de nouveau.

III.2.5 Activités SOA de support (Groupe E)

E.1. Tester les composants de services : durant cette activité, chaque composant est testé séparément avant son intégration dans un ou plusieurs services. Cette activité a pour objectif d'évaluer le code écrit — afin de s'assurer que ce dernier est conforme aux normes de l'organisation et d'identifier des erreurs potentielles au niveau de la performance et de la sécurité — et de s'assurer que les fonctionnalités des composants sont conformes aux exigences spécifiées dans les contrats (Bloomberg et Schmelzer, 2006 ; Harris, 2007).

E.2. Tester des interfaces de services[23] (Harris, 2007) : cette activité focalise sur les interfaces de services. Elle a pour objectif de déterminer si le comportement de l'interface et le partage de l'information entre les services fonctionnent conformément aux spécifications

---

[23] Nous avons remplacé l'expression proposée par l'auteur « *Integration test* » par « Test des interfaces de services » car nous pensons que cette dernière correspond plus à la description de l'activité.

(normes, formats, données, etc.). Les scénarios de test destinés aux tests des interfaces de services prennent aussi en considération les couches de communication et les protocoles de réseaux, ainsi que le test des services externes à l'organisation.

E.3. Tester l'orchestration des services : ces tests assurent que les services opèrent collectivement conformément aux spécifications (Harris, 2007). Durant cette activité, la cohérence des données (validité et intégrité) est vérifiée, le modèle de composition des services est évalué et des simulations sont accomplies afin d'évaluer le comportement de l'application composée (Tsai et al., 2006).

E.4. Élaborer des documents de présentation (whitepapers) : les documents de présentation constituent un bon point de départ pour disséminer l'information relative à l'introduction d'une nouvelle technologie ou d'une nouvelle méthode dans une entreprise. Idéalement, il devrait y avoir plusieurs documents de présentation, chacun traitant d'un aspect particulier de l'initiative SOA (Krafzig et al., 2004).

- *Un document de présentation de la stratégie*, qui explique l'objectif de l'initiative SOA. Les aspects à couvrir incluent, par exemple, l'intégration avec l'architecture existante, les principaux facteurs de changement résultant de l'initiative SOA et son potentiel d'intégration au sein de l'entreprise (intégration avec les fournisseurs, les partenaires et les clients).

- *Un document de présentation d'affaires*, qui focalise sur les bénéfices en affaires qui résultent de l'introduction de SOA. Il devrait contenir une analyse RCI et démontrer les bénéfices de l'augmentation de la réutilisation de la fonctionnalité d'affaires mise en œuvre et un potentiel de développement efficace de nouveaux services pour les clients, les employés et les partenaires. Il pourrait de même souligner les fonctionnalités d'affaires qui conviennent idéalement à la réutilisation.

- *Un document de présentation technique*, qui couvre les aspects techniques impliqués dans la mise en œuvre de SOA. D'une part, il décrit comment

l'intégration de SOA avec l'infrastructure technique existante est envisagée. D'autre part, il décrit la réalisation technique de SOA elle-même. Souvent, une plateforme spécifique sera utilisée ou développée pour réaliser l'infrastructure technique de SOA (bus des services d'entreprise, voir activité C.8). Ce document peut servir à spécifier l'étendue de cette plateforme et un plan pour sa mise en œuvre. Le registre représente un des ingrédients clé de SOA. Le document de présentation technique devrait aussi décrire la structure du registre, ainsi que les processus pour l'utiliser et l'améliorer.

E.5. Définir un modèle de programme d'application : Fournir des directives concernant la façon avec laquelle le développement sera structuré, incluant le processus, les technologies, les normes de codage et les procédures de déploiement (Mittal, 2006). En particulier, la réutilisation de la fonctionnalité fournie par les services existants devrait être préférée à la mise en œuvre d'une nouvelle fonctionnalité, même si ceci nécessite la modification de la mise en œuvre du service existant. Un tel processus pourrait être pris en charge par un comité technique ou architectural dédié (Krafzig et al., 2004).

E.6. Élaborer un document d'applicabilité SOA : Ce document explique comment les projets à venir peuvent être construits au-dessus de l'infrastructure SOA (Mittal, 2006). De cette façon, les nouveaux projets peuvent être sûrs d'adhérer aux principes de l'axé sur le service.

E.7. Planifier et exécuter les activités de formation : les projets SOA requièrent l'acquisition de nouvelles compétences par le personnel impliqué dans le CVP. Le personnel client peut aussi avoir besoin d'être formé pour pouvoir se servir des nouvelles applications (Mittal, 2006). Les activités de formation nécessaires doivent être planifiées tôt dans le CVP, bien avant que le personnel ne commence à appliquer l'expertise requise.

Dans ce qui suit, nous présentons une figure récapitulative qui illustre l'ensemble des activités SOA nous venons de présenter réparties selon les cinq groupes d'activités de la norme IEEE/EIA std 1074.

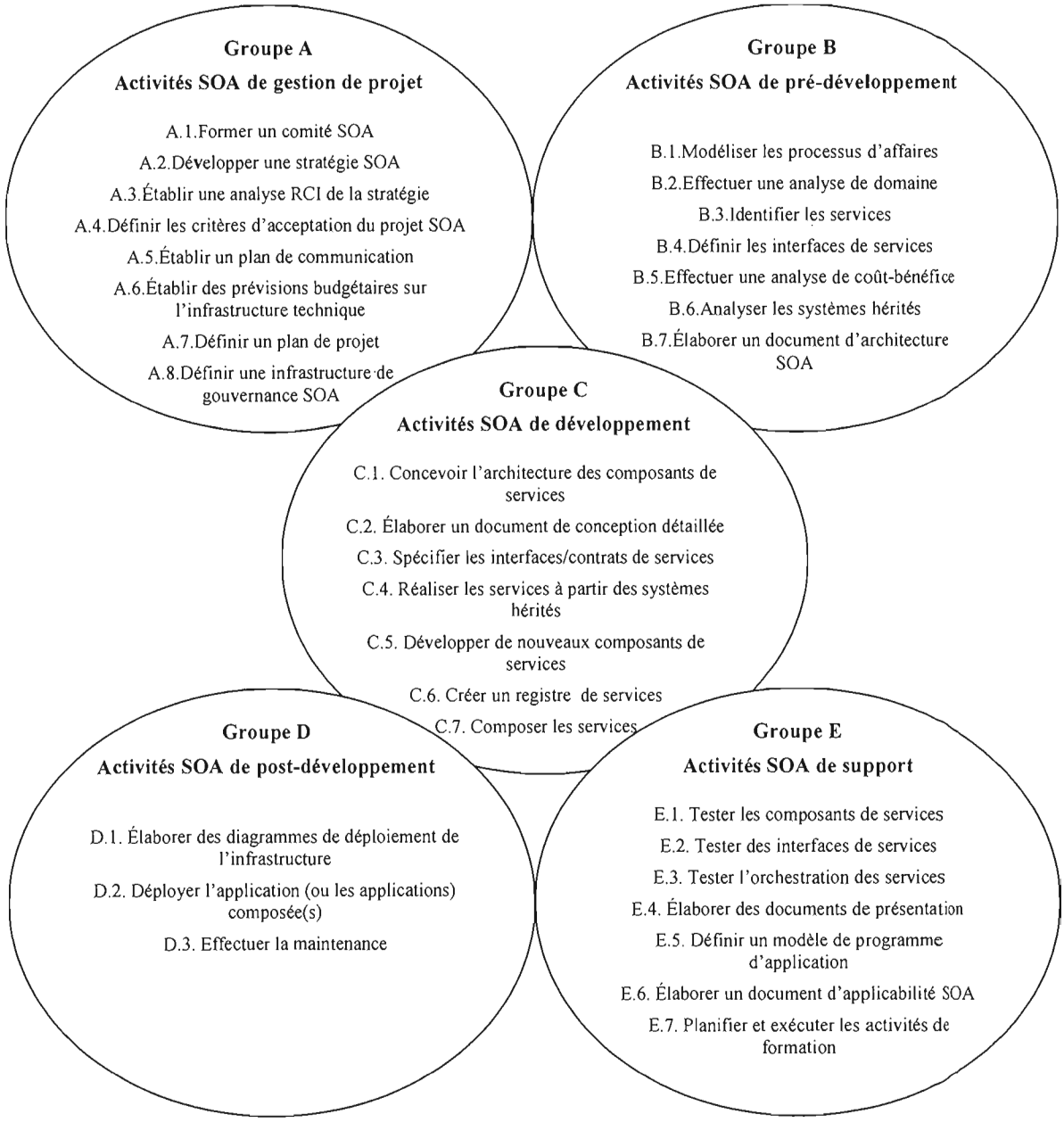


Figure 3.2 Activités SOA réparties selon les groupes d'activités de la norme IEEE/EIA std 1074

III.3 Rôles et tâches SOA

En partant de la littérature analysée, cette section effectue une synthèse des *rôles* impliqués dans une initiative SOA et des *tâches* qui leurs sont associées. Une activité étant un ensemble de *tâches*, comprendre les activités à accomplir durant un projet SOA (section III.2) nous a permis de mieux appréhender les *tâches* et de faciliter ainsi leur assignation aux rôles identifiés.

Parmi les rôles cités dans la littérature, certains sont plutôt des activités camouflées en rôles (ex. gestionnaire de gouvernance SOA, gestionnaire de mesures SOA, etc.). Ces rôles n'ont pas été retenus dans notre synthèse. Les rôles SOA retenus sont au nombre de sept (7) : Gestionnaire de projet, Architecte d'entreprise, Architecte SOA, Analyste d'affaires, Concepteur de services, Développeur de services et Administrateur du registre des services. Un groupe de rôles a de même été retenu en raison de son importance. Il s'agit du Comité SOA.

Dans ce qui suit, nous allons décrire les tâches qui sont attribuées à chacun des sept rôles retenus, ainsi qu'au groupe de rôles du comité SOA, selon notre analyse de la littérature.

Gestionnaire de projet : ce rôle est responsable de la gestion et du contrôle du projet SOA, ainsi que de la définition des plans de projet et de leur mise en œuvre (Kajko-Mattsson et al., 2007). Dans un environnement axé sur les services, le rôle du gestionnaire de projet subit des changements, parmi lesquels nous retenons :

- La planification des cycles de développement courts, en raison de la courte durée des cycles de livraison des services (Bieberstein et al., 2005 ; Mittal, 2006b)
- L'établissement de nouveaux modèles d'acceptation pour les projets SOA (Bieberstein et al., 2005).
- La communication avec les utilisateurs et la collaboration avec les rôles concernés pour la définition des contrats de services (Bieberstein et al., 2005 ; Mittal, 2006b).

- L'assurance que les spécifications des contrats de services ont bien été respectées (Mittal, 2006b).

Architecte d'entreprise[24] **:** il est responsable de l'architecture logique et physique de l'initiative SOA et approuve les architectures d'applications (Strnadl, 2007). Les fonctions attribuées à ce rôle sont (Mittal, 2006b) :

- Promouvoir et encourager l'adoption de SOA.
- Communiquer les principes de l'axé sur les services aux architectes d'application et aux groupes de développement
- Collaborer avec les architectes SOA dans la traduction des exigences d'affaires en services.
- Définir les mécanismes de mesure et de suivi des SLA.
- Confirmer que les services construits sont capables de rencontrer les SLA établis.
- Définir des politiques et des procédures de gouvernance, de sécurité.

Architecte SOA[25] **:** lors d'un projet SOA, il constitue un intermédiaire entre les groupes d'affaires et les groupes techniques pour la conception des aspects techniques des services (Kajko-Mattsson et al., 2007). Son rôle est de (Mittal, 2006b) :

- Traduire les exigences d'affaires en services.

---

[24] L'architecte d'entreprise constitue un rôle émergent pour plusieurs entreprises (Bloomberg et Schmelzer, 2006). Dans les organisations traditionnelles, l'architecte d'entreprise définit les politiques de gouvernance, les meilleures pratiques et les procédures que chaque groupe d'architecture d'application devrait suivre (Mittal, 2006b).

[25] Dans la littérature SOA, l'architecte SOA est aussi appelé « Architecte d'affaires SOA » ou « Architecte de processus d'affaires ».

- Savoir quel service exposer, avec quel niveau de granularité et de quelle façon l'exposer.
- Définir les exigences non fonctionnelles des services.
- S'assurer que le code écrit adhère aux principes de l'axé sur les services.
- Travailler de concert avec les architectes d'entreprise et les autres architectes SOA pour s'assurer que toute opportunité de réutilisation de service est bien exploitée et que les services sont proprement construits, découverts, sécurisés, utilisés et mesurés.

Analyste d'affaires : L'analyste d'affaires est un intermédiaire entre les groupes d'affaires et les groupes techniques pour la conception des aspects d'affaires des services (Mittal, 2006b). Il doit avoir une bonne compréhension du domaine d'affaires et posséder les aptitudes nécessaires pour transmettre les connaissances reliées à ce domaine à l'équipe de projet (Bieberstein et al., 2005). Dans un environnement axé sur le service, l'individu responsable de ce rôle a pour principales fonctions de :

- Communiquer avec les gestionnaires et les utilisateurs pour comprendre les exigences d'affaires. De ce fait, il représente les clients potentiels des services.
- Travailler avec l'équipe de développement au complet pour que ses membres puissent commencer à penser en termes de services.
- Travailler avec l'équipe technique pour la conception et la construction des services nécessaires, et potentiellement la réutilisation des services existants.

Concepteur de services : il modélise les services (données, fonction, états, interfaces, etc.) (Kajko-Mattsson et al., 2007) et travaille conjointement avec l'architecte SOA pour la création des contrats de services à partir des exigences d'affaires (Bieberstein et al., 2005).

Concepteur du registre des services : il est responsable de la planification, de la conception et de la construction du registre des services, (Bieberstein et al., 2005). Le registre peut utiliser un modèle de données générique et normalisé tel qu'UDDI[26], ou un modèle de données propriétaire à l'organisation.

Développeur de services : ce rôle concerne le développement les interfaces et les mises en œuvre des services, ainsi que le code d'invocation des services (Bieberstein et al., 2005). Les personnes ayant ce rôle devraient être aptes à comprendre les processus et la fonctionnalité d'affaires et à développer les services conformément aux principes de l'axé sur les services, notamment celui de la réutilisabilité (Mittal, 2006b). Si un service existe déjà, il faut l'utiliser, non le réinventer. Un développeur de services devrait avoir deux sous-rôles[27] (Bieberstein et al., 2005) :

- Développeur de services, possédant de l'expertise avec XML et les normes des services Web et ayant des habilités de mise en œuvre sous des plateformes telles que J2EE, C++ ou .NET.

- Adaptateur de systèmes hérités, ayant des habilités avec l'adaptation des systèmes hérités en services.

Administrateur du registre des services[28] : il est responsable du registre des services dans son ensemble. Il est chargé de l'enregistrement des services, approuve les ajouts et les changements au contenu du registre et s'assure de la cohérence des données des services publiés (Kajko-Mattsson et al., 2007 ; Strnadl, 2007).

---

[26] Universal Description, Discovery and Integration directory

[27] Selon Mittal (2006b), en raison de la courte durée des cycles de développement utilisés par les équipes SOA, la division des développeurs en fonction de la technologie, habituellement appliquée dans les groupes TI traditionnels, est impraticable. Dans les groupes TI traditionnels, les développeurs œuvrent typiquement dans un segment de l'application. Ces segments sont déterminés en fonction de la fonctionnalité (ex. module d'établissement de rapports, d'enregistrement, etc.) ou de la technologie (JSP, EJB, etc.). Ceci explique peut être la division des rôles des développeurs par Bieberstein et al. (2005) que nous citons.

[28] Appelé aussi « Bibliothécaire de services » et « Bibliothécaire SOA », dans la littérature SOA.

Comité SOA : ce groupe de rôles a une responsabilité globale de l'initiative SOA et du contrôle de son application dans les projets en-cours. Il a pour tâches de[29] :

- Définir les rôles, les responsabilités et les procédures pour la gestion des processus et des activités SOA.
- Définir un processus qui gouverne le cycle de vie des services.
- Établir des politiques et des processus pour la gestion de la qualité des services.
- Élaborer un corps de directives SOA incluant les directives métiers et les normes industrielles et organisationnelles.
- Définir et renforcer les directives qui vont assurer la conformité aux objectifs et aux standards SOA, ainsi qu'aux objectifs du processus SOA.

Dans ce qui suit, nous présentons une figure récapitulative qui illustre les *rôles* et *tâches* SOA qui viennent d'être définis.

---

[29] Le comité SOA, en tant que tel, n'a pas été bien défini dans la littérature. Seulement, nous avons pu comprendre d'après les quelques définitions qui en ont été données que les tâches associées à ce groupe de rôle correspondent, en fait, aux pratiques de la gouvernance SOA.

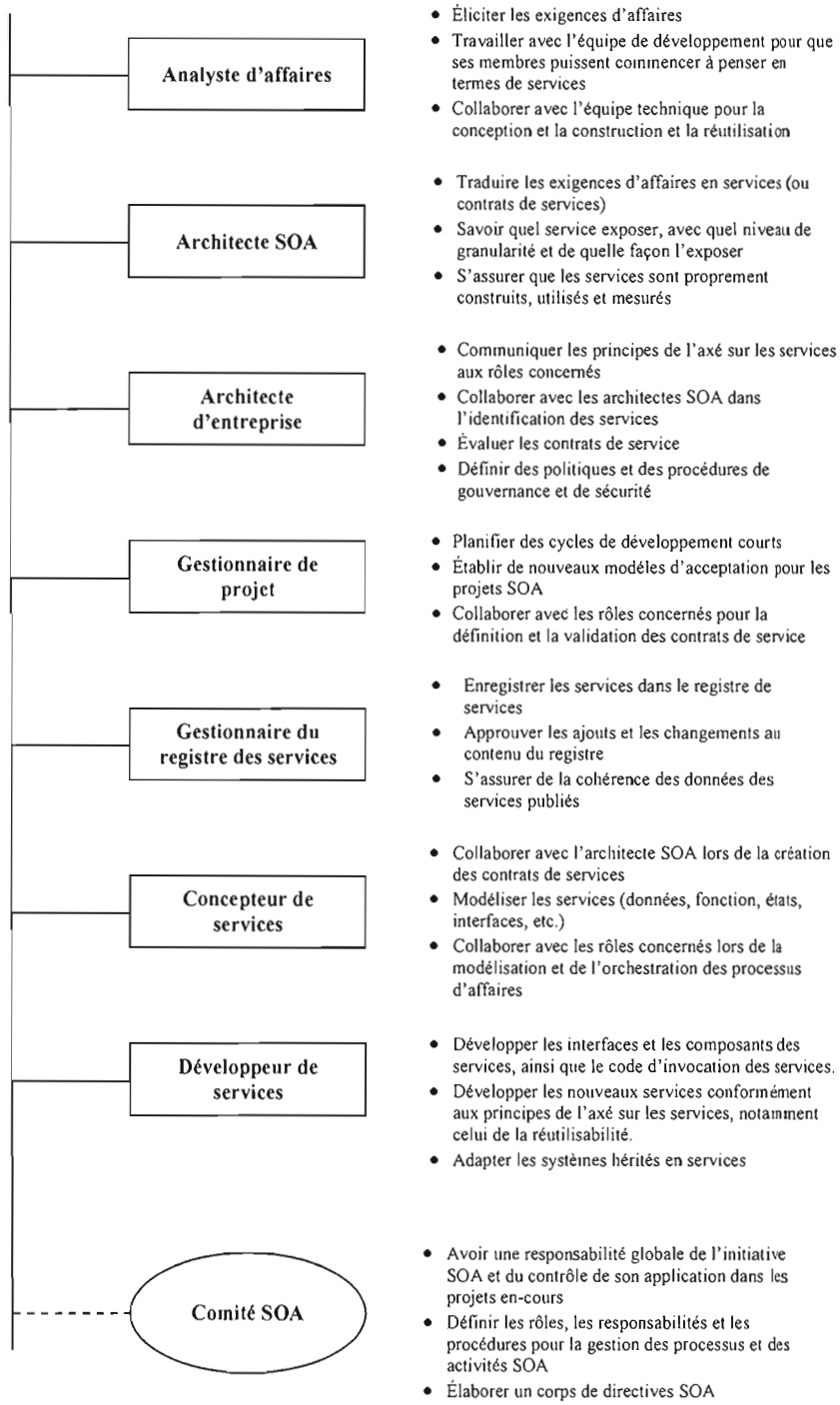


Figure 3.3 *Rôles* et *tâches* SOA

III.4 Conclusion

Suite à cette synthèse, nous pouvons conclure que ce sont les rôles du gestionnaire de projet et des architectes qui jouissent des plus grandes responsabilités dans le déroulement des projets SOA. Le gestionnaire de projet doit veiller à la planification et au bon déroulement du projet SOA et les architectes sont responsables du raffinement et de la validation de tâches telles que l'identification des services et la définition des contrats de services. Le comité SOA a de même de grandes responsabilités dans le déroulement de l'initiative SOA qui vont de la définition des rôles impliqués dans le CVP, à l'instauration des règles de gouvernance SOA. Ce travail de synthèse nous aura permis de :

- Identifier les rôles qui possèdent le plus d'autorité et de décision formelle quant au déroulement du projet SOA et qui, par conséquent, sont les meilleurs candidats pour les intervenants de notre étude de cas.

- Identifier les points qui n'ont pas été assez traités par la littérature et dont nous pourrons dériver les questions d'investigation relatives à notre étude de cas (voir section IV.2.3).

- Élaborer les questionnaires spécifiques qui seront adressées à chaque répondant, dépendamment des tâches qui se trouvent dans les limites de son rôle et des activités auxquelles il participe (voir section IV.2.3).

CHAPITRE IV

ÉTUDE DE CAS

Tel que déjà énoncé dans notre méthode de recherche (section I.4), notre recherche suit une approche positiviste de type exploratoire et n'a donc aucune prétention d'apporter des preuves. Notre étude devrait permettre d'approfondir certaines hypothèses (ex. l'importance des nouveaux rôles dans le développement axé sur les services), hypothèses qui devraient être étudiées et insérées dans un contexte théorique plus complet dans des travaux futurs. Qu'elle soit basée sur un ou plusieurs cas, une étude de cas exploratoire a pour objectif de définir des questions, des construits, des propositions ou des hypothèses qui feront l'objet d'une étude empirique subséquente (Yin, 1993). Au départ, en raison de la période de temps disponible pour l'étude et afin de maximiser ce qui peut être appris durant ce délai, nous avions choisi de mener une étude de cas unique. Seulement, suite à l'étude du cas choisi, il s'est présenté à nous une opportunité pour un deuxièmes cas dont l'étude nous semblait très intéressante. En effet, le deuxième cas présente une architecture SOA décentralisée, différente de celle du premier cas, qui était complètement centralisée. En incluant ce deuxième cas à notre recherche nous avions la possibilité d'étudier l'impact de SOA sur le CVP dans deux contextes totalement différents.

Les questions de recherche que nous avons élaborées, et qui ont été énoncées dans la section I.3, définissent un objectif clair pour notre étude et permettent de spécifier le type de données à collecter. Dans ce qui suit, nous allons détailler le processus de sélection de nos cas d'étude (section IV.1), ainsi que le protocole que nous avons élaboré pour mener notre investigation (section IV.2).

IV.1 Sélection des cas

À présent, beaucoup d'entreprises sont en train d'adopter SOA. Peu importe le domaine d'activité des organisations que nous avons choisi pour mener notre étude – qu'elles soient des entreprises manufacturières, des institutions financières ou des entreprises spécialisées dans un tout autre domaine – l'essentiel c'est que les organisations en question aient déjà entamé une initiative SOA. La sélection de nos cas d'étude a de même dépendu des ressources qui ont pu nous être accordées pour mener notre investigation (nombre de répondants, nombre d'entrevues accordées, nombre de sources de preuves qualitatives, etc.), ainsi que de la grande différence entre les environnements axés sur les services des deux cas choisis.

Le premier cas présente un environnement complètement centralisée, où une seule organisation est responsable du développement des différents composants de l'architecture SOA. Il s'agit d'une institution financière, dénommée X, qui a entamé son initiative SOA depuis trois années. L'objectif de la mise en place d'une SOA dans cette institution financière est principalement de permettre de livrer des fonctionnalités qui pourront être réutilisées de projets en projets.

Le deuxième cas présente une initiative de mise en œuvre d'une SOA dans un environnement axé sur les services décentralisé, où certaines organisations constituent des fournisseurs de services, d'autres sont des demandeurs de services et d'autres sont chargées de la mise en œuvre de l'infrastructure SOA. Ce deuxième cas est celui du Dossier de Santé du Québec (DSQ). Le DSQ est un projet pilote pour l'initiative du Dossier de Santé Électronique (DSÉ) – une initiative d'instauration d'un système de santé interopérable à travers tout le Canada. La solution proposée pour le DSÉ, nommée « infostructure du DSÉ » (iDSÉ) a été conçue en se basant sur une architecture SOA. Pour connaitre plus en détails le projet DSQ, veuillez consulter l'appendice A.

IV.2 Le protocole de l'étude de cas

La fiabilité est un concept très important dans la recherche par étude de cas positiviste (Yin, 2003). L'objectif de la fiabilité est de minimiser les erreurs et les biais dans l'étude. En

général, une façon d'atteindre la fiabilité est de conduire la recherche de façon à ce qu'un autre investigateur puisse répéter les mêmes procédures et arriver aux mêmes conclusions. Une condition préalable pour permettre à d'autres investigateurs de répéter une ancienne étude de cas est la documentation des procédures qui ont été utilisées dans l'étude (Paré, 2004). Yin (2003) propose de créer un protocole d'étude de cas pour accroître la fiabilité.

Un protocole d'étude de cas ne contient pas seulement les instruments d'entrevues ou d'étude. Il devrait aussi inclure des procédures et des règles générales à suivre lors de l'utilisation des instruments et qui doivent être crées avant la phase de collecte des données. Un protocole d'étude de cas devrait contenir quatre composants, qui sont (Paré, 2004) :

1. Un aperçu du projet de l'étude de cas, décrivant l'objectif de l'étude de cas et les questions d'investigation.

2. Le domaine des procédures, qui s'intéresse aux questions reliées à la collecte des données telles que l'accès à l'organisation en question, la possession des ressources suffisantes à l'intérieur de l'organisation et la planification des activités de collecte des données.

3. Le guide des entrevues (ou des entretiens), qui contient les questions spécifiques à discuter avec chaque répondant.

4. Le rapport de l'étude de cas, qui contiendra les données collectées, énoncées sous un format approprié, et qui doit être révisé par les participants.

IV.2.1 Aperçu du projet de l'étude de cas

L'objectif de notre recherche est de savoir si le passage vers un environnement axé sur les services pour une organisation demande des changements aux *rôles* et aux *tâches* du CVP. En effectuant une analyse et une synthèse de la littérature SOA, nous avons constaté ces changements identifiés dans la section III.3.

Les questions d'investigation de notre étude de cas ont donc pour finalité :

1. de vérifier si les changements (dans les *rôles* et les *tâches* du CVP) identifiés sont présents dans la pratique des entreprises étudiées.

2. de vérifier si des changements introduits dans les entreprises étudiées n'ont pas été considérés dans la littérature que nous avons analysée.

3. de clarifier les aspects qui n'ont pas été assez investigués par les recherches.

IV.2.2 Méthodes de collecte des données

La recherche par étude des cas combine plusieurs méthodes de collecte de données. Collecter différents types de données par différentes méthodes, de plusieurs sources, produit une couverture plus importante et peut résulter en une vision plus étendue sur le phénomène étudié (Bonoma, 1985 ; Paré, 2004). Yin (2003) identifie diverses sources de preuves qualitatives, parmi lesquelles la documentation, les documents d'archives, les entrevues, l'observation directe et les artefacts physiques.

Durant notre étude, nous avons essayé d'avoir accès au maximum des sources d'information qui viennent d'être citées. Cependant, les entrevues, la documentation et les questionnaires élaborés pour nos répondants ont constitué notre principale méthode de collecte de données.

Dans ce qui suit, nous allons donner les caractéristiques de répondants sollicités pour nos deux cas d'étude (section IV.2.2.1), fournir des détails sur la façon avec laquelle les entretiens se sont déroulés (section IV.2.2.2) et définir la nature du contenu des documents auxquels nous avons eu accès (section IV.2.2.3). Ces derniers étant de nature confidentielle, ils ne seront pas présentés en annexe. Les questionnaires destinés à nos répondants sont décrits dans la section IV.2.3.

IV.2.2.1 Répondants

Lors de l'étape de synthèse de la littérature, nous avons remarqué que ce sont les rôles des gestionnaires de projets et des architectes qui subissent le plus de changements et qui se voient accorder les tâches les plus importantes lors d'une initiative SOA. De plus, ce sont les

individus responsables de ces rôles qui possèdent le plus d'autorité et qui prennent les décisions pouvant avoir le plus d'impact sur le déroulement du projet et qui, par conséquent, sont les plus aptes à nous communiquer l'information recherchée.

Le tableau ci-dessous présente, pour chaque cas, le profil des répondants sollicités, leur unité d'affiliation à l'organisation ou au projet, leurs années d'expérience (avec des projets SOA ou d'autres projets) et le nombre de rencontres. Pour ce qui est du cas 1, R1 a été notre unique répondant. Néanmoins, R1 combine deux rôles à la fois (celui de directeur de comité SOA[30] et celui de gestionnaire de projet) et a été impliqué dans tous les projets élaborés dans le cadre de l'initiative SOA de l'organisation X.

En analysant de près le projet DSQ, nous nous sommes aperçus du nombre important d'intervenants dans le projet et du grand degré d'interaction entre ces derniers. Il ne s'agit plus du même contexte que celui de l'organisation X, où la mise en œuvre de l'architecture SOA est sous la responsabilité d'une organisation unique et où le nombre d'utilisateurs est relativement restreint. En raison de sa complexité et de sa grande envergure, le projet DSQ a été découpé en plusieurs sous-projets qui ont pour finalité de fournir les composantes technologiques de la solution de DSQ. L'un de ces sous-projets est le projet SRC-Labo dont font partie R3, R4 et R5 (les raisons du choix de ce sous-projet sont expliquées dans la section IV.2.2.2). Mis à part les groupes chargés des sous-projets, il existe un groupe d'intégration incluant une équipe d'architecture (dont fait partie R2), une équipe de déploiement, une équipe d'exploitation et une équipe d'infrastructure et de télécommunications. Toutes ces dernières équipes travaillent au niveau du projet DSQ pour intégrer la solution avant de la déployer en région.

Il est à noter que le choix de ces répondants, en particulier pour Cas 2, n'a pas été totalement de notre ressort. R2 a constitué notre premier point de contact pour le projet DSQ et s'est avéré être un bon candidat comme premier répondant pour le cas 2 étant donné son rôle d'architecte au sein du groupe d'intégration. D'autres personnes affiliées au Cas 2, dont

---

[30] Appelé « centre de compétences SOA » (CC SOA) par l'organisation X.

un architecte (R4) et un directeur de projet (R5) et, se sont portées volontaires pour participer à note recherche, mais n'ont pas pu par la suite se rendre disponibles pour un entretien.

Tableau 4.1 Spécificités des répondants pour les deux cas d'étude

Cas d'étude	Rép	Rôle(s)	Années d'expérience SOA	Années d'expérience Autre	Unité d'affiliation	Nbre de rencontres (durée)	Commentaires
Cas 1	**R1**	Directeur comité SOA Gestionnaire de projet	1 à 2	+ de 15	Comité SOA et projets SOA	1 (1h)	R1 combine deux rôles à la fois.
Cas 2	**R2**	Architecte-intégrateur	2 à 4	+ de 15	Groupe d'intégration du DSQ	1 (1h)	-
	R3	Architecte de solutions	2 à 4	+ de 15	Projet SRC-Labo du DSQ	1 (1h)	-
	R4	Architecte de solutions	-	-	Projet SRC-Labo du DSQ	0	Pas de rencontre
	R5	Gestionnaire de projet	-	-	Projet SRC-Labo du DSQ	0	Pas de rencontre

IV.2.2.2 Déroulement des entretiens

Avant chaque entretien, le répondant a été introduit aux objectifs de notre recherche et a reçu le questionnaire qui lui était destiné quelques jours à l'avance. Les entretiens se sont déroulés selon cet ordre : entretien avec R1, avec R2 et finalement avec R3.

Une fois l'entretien avec R1 accompli et les échanges de documents (voir Tableau 4.2 **Documentation relative aux deux cas d'étude**) effectués, nous avons pu collecter les données désirées du cas 1, à savoir les *rôles* impliqués dans l'initiative SOA avec leurs *tâches* associées.

En ce qui concerne le cas 2, les choses n'ont pas été aussi faciles, essentiellement en raison de l'indisponibilité des répondants du projet SRC-Labo. Néanmoins, les données synthétisées du cas 1 ont servi à l'élaboration des questionnaires du cas 2. Les entretiens avec

R2 et R3 se sont déroulés à la date souhaitée. Ceux avec R3 et R4 n'ont finalement pas eu lieu, après une attente d'environ 3 mois. C'est ce qui explique l'interruption de ce cas.

L'entretien avec R2 a permis d'appréhender le projet DSQ dans son ensemble, de comprendre le rôle du répondant dans ce projet, ainsi que celui de l'équipe dont il fait partie. Il nous a de même permis de réaliser la complexité et l'étendue du projet DSQ qui peuvent poser problème pour son étude. Suite à cette rencontre, il a été décidé que si nous voulions continuer à travailler sur ce cas, il serait préférable de choisir un de ses sous-projets et de s'y intéresser en particulier afin d'éliciter les *rôles* et les *tâches* associées. R2 nous a conseillé le choix du projet SRC-Labo[31], puisque ce dernier touche au développement des 3 composantes de l'architecture SOA du DSÉ (services et registres, applications frontales et bus de services), qu'il est bien documenté et qu'il pourrait nous mettre en contact avec ses membres.

Pour l'étude du projet SRC-Labo, nous avons eu accès à de la documentation (voir Tableau 4.2 **Documentation relative aux deux cas d'étude**) et nous avons été mis en contact avec R3, R4 et R5. Tel que déjà énoncé, les entretiens avec R4 et R5 n'ont finalement pas eu lieu en raison de l'indisponibilité de ces derniers. Néanmoins, l'entretien avec R3 nous a permis de collecter des informations concernant les impacts de SOA sur le rôle et les tâches de l'architecte en particulier, surtout que pour R3, le projet SRC-Labo a constitué une première expérience avec SOA.

IV.2.2.3 Documents

Le tableau ci-dessous présente les documents auxquels nous avons eu accès pour nos deux cas. En raison de leur grand nombre (surtout pour le cas 2), seuls les principaux documents utilisés sont listés.

---

[31] Le projet SRC-Labo (Services Régionaux de Conservation et Laboratoires) fait partie de l'ensemble des projets travaillant pour la réalisation du DSQ.

Tableau 4.2 Documentation relative aux deux cas d'étude

Cas d'étude	Type de document	Nombre de documents	Nature du contenu
Cas1	Présentation	2	Objectifs et stratégie SOA de X
	Présentation	1	Stratégie de tests SOA de X
	Modèle	1	Processus de développement SOA décrivant les activités, les rôles et les artefacts
Cas 2	Rapport	1	Plan d'implantation du DSQ
	Rapport	1	Appel d'offres du projet CAIS
	Rapport	1	Appel d'offres du projet SRC-Labo
	Bulletin	11	Bulletins du projet DSQ décrivant l'avancement du projet et les changements apportés

IV.2.3 Guide des entrevues

Le guide des entretiens contient les questions spécifiques à discuter avec les répondants et les questions à se rappeler durant chaque entrevue (Paré, 2004). Étant donné qu'il s'agit d'une étude de cas exploratoire, les entrevues que nous avons planifiées sont semi-structurées, c'est-à-dire que les questions élaborées sont ouvertes et laissent place à l'argumentation du répondant.

Les questionnaires élaborés (voir Appendice B) contiennent des questions commues, qui étaient destinées à tous les répondants et des questions spécifiques pour chaque répondant. La liste des questions spécifiques dépend de plusieurs facteurs tels que l'unité d'affiliation et le rôle joué dans le projet par l'intervenant. D'une façon générale, les questions ont été élaborées de façon à ne pas influencer la perception et le jugement des répondants face à la problématique.

Pour tous les questionnaires, les questions I et II (questions communes) ont eu pour objectif de spécifier le nombre d'années d'expérience avec les projets SOA et avec d'autres

types de projets de développement logiciel de chaque répondant, ainsi que les postes occupés par ce dernier.

Quant aux questions spécifiques à chaque répondant, elles sont explicitées dans les tableaux ci-dessous. Pour chaque questionnaire, nous présentons un tableau qui défini les objectifs des questions spécifiques et leur origines. L'origine d'une question peut correspondre à un élément de la littérature (chapitre 2), du travail de synthèse effectué sur la littérature (chapitre 3), de la documentation qui nous été accordée pour les deux cas (section IV.2.2.3), des données synthétisées du Cas1 (section IV.2.4) ou a une combinaison de ces derniers éléments.

Il est à noter qu'un tableau n'a pas été présenté pour Q4, étant donné que les questionnaires Q3 et Q4 sont identiques. Ceci est du au fait que R3 et R4 jouent le même rôle au niveau du projet DSQ.

Tableau 4.3 Objectif(s) et origine(s) des questions de Q1

Question(s)	Objectif(s)	Origine(s)	Remarques
III	Savoir si le centre de compétences SOA (CC SOA) possède le même rôle que celui du comité SOA	Tâches du comité SOA (section III.3)	Q1 a eu pour objectif principal de d'identifier les activités du CC SOA et de son directeur étant donné que les seules informations disponibles lors de son élaboration sont que : – X est une institution financière qui a entamé une initiative SOA – La personne avec qui nous avons obtenu un entretien est le directeur du CC SOA de X.
IV et V	Identifier les membres du comité SOA	Rôles SOA (section III.3)	
VI	Définir les tâches associées au rôle du directeur du CC SOA	N/A	
VII et VIII	Tâches du CC SOA	Gouvernance SOA (section II.4.4) Tâches du comité SOA (section III.3)	
VIII et X	Identifier la méthode de développement utilisée par l'organisation X pour ses projets SOA	Les différentes méthodes de développement (section II.3.1)	

Tableau 4.4 Objectif(s) et origine(s) des questions de Q2

Question(s)	Objectif(s)	Origine(s)	Remarques
III et IV	Connaître l'affiliation de R2 au projet DSQ	Documentation publique du DSQ	Les seules informations portant sur le projet DSQ auxquelles nous avions eu accès avant l'élaboration de Q2 sont celles rendues publiques sur le site Web du Ministère le la Santé et des Services Sociaux. En ce qui concerne R2, nous savions uniquement qu'il était à la fois affilié à une grande compagne de consultation canadienne, dont nous ne citerons pas le nom et qu'il occupait un poste d'architecte au sein du projet DSQ.
V	Définir les principales composantes de l'architecture SOA du DSQ étant donné la complexité de cette dernière	Éléments d'une SOA (section II.2.4)	
VI	Identifier les différentes organisations impliquées dans le développement des composantes de l'architecture SOA du DSQ	Éléments d'une SOA (section II.2.4)	
VII	Identifier l'entité qui joue le rôle du comité SOA	Gouvernance SOA (section II.4.4) Tâches du comité SOA (section III.3)	
VIII, IX et X	Identifier la méthode de développement utilisée pour le DSQ et pour les projets SOA initiés par la firme de consultation à laquelle est affilié R2	Les différentes méthodes de développement (section II.3.1)	

Tableau 4.5 Objectif(s) et origine(s) des questions de Q3

Question(s)	Objectif(s)	Origine(s)	Remarques
III et IV	Connaître l'affiliation de R3 au projet DSQ et au sous-projet SRC-Labo	Documentation relative au DSQ	Étant donné que le rôle de l'architecte de solutions n'a pas été abordé dans la littérature nous avons utilisé dans les questions VII et VIII les informations synthétisées du Cas 1.
V	Définir les principales composantes de l'architecture SOA du DSQ étant donné la complexité de cette dernière	Éléments d'une SOA (section II.2.4)	
VI	Situer les objectifs du projet SRC-Labo par rapport à ceux du projet DSQ en identifiant les composantes SOA dont le projet SRC-Labo possède la responsabilité de développement	Éléments d'une SOA (section II.2.4)	
VII et VIII	Définir les tâches accomplies par R3	Tâches du rôle d'architecte de solutions de X (section IV.2.4)	

Tableau 4.6 Objectif(s) et origine(s) des questions de Q5

Question(s)	Objectif(s)	Origine(s)	Remarques
III et IV	Identifier les rôles présents dans le projet SRC-Labo et vérifier la pertinence des tâches qui leurs sont associées	Rôles SOA de la littérature (section III.3) et rôles SOA présents dans X (section IV.2.4)	
V	Approfondir les tâches qui sont sous la responsabilité de R5	Tâches associées au rôle de gestionnaire de projet identifiées à partir de la littérature (section III.3) et du Cas 1 (section IV.2.4)	
VI et VII	Identifier la méthode de développement utilisée pour le projet SRC-Labo	Les différentes méthodes de développement (section II.3.1)	

IV.2.4 Rapport de l'étude de cas

Les données collectées proviennent de trois sources : (1) documentation reliée aux projets SOA des deux cas d'étude, (2) questionnaires remplis par les répondants et (3) informations résultant des entretiens. Les entretiens ont principalement consisté à faire le tour des questionnaires individuels avec les répondants. Certains répondants ont préalablement rempli le questionnaire dans sa totalité avant l'entretien, d'autres l'ont fait partiellement et d'autres ont préféré le faire durant l'entretien même.

Les informations collectées ont été analysées et synthétisées. Le tableau ci-dessous présente le résultat du travail de synthèse accompli sur ces dernières. Nous avons délibérément choisi de ne pas les rédiger dans un document et de ne pas les soumettre aux participants à des fins de validation en raison de l'indisponibilité de ces derniers pour cette tâche. C'est pour cette raison que nous avons essayé de nous assurer de l'exactitude des informations collectées durant les entretiens mêmes.

Tableau 4.7 Synthèse des données collectées à partir du cas 1 et du cas 2

	Cas 1	Cas 2
Mise en œuvre de SOA		
Développement des composantes SOA[32]	– Les trois composants (applications frontales, services et bus de services) sont mis en œuvre par l'organisation X. – X collabore avec une firme de consultants pour la définition d'un nouveau processus de développement	– La mise en œuvre des 3 composantes SOA est sous la responsabilité de plusieurs organisations – Plus de 10 organisations se partagent le développement des composants SOA : certaines sont en charge du développement des services, d'autres des applications frontales et d'autres du bus de services
Impacts de SOA sur le CVP		
Rôles & tâches	– Attribution de nouvelles tâches aux rôles existants – Apparition de nouveaux rôles avec de nouvelles tâches	– N'ont pas pu être entièrement déterminés en raison de l'interruption du cas – Existence du rôle d'architecte de solutions avec les tâches correspondantes, tel que présent dans Cas 1
Autres	– Adoption d'un nouveau processus de développement plus agile, qui offre des livraisons plus rapides	– Utilisation d'une méthode de documentation unifiée pour tous les projets impliqués dans l'initiative SOA
Correspondance avec la littérature		
Rôles & tâches SOA	– Grande similitude avec les rôles et tâches SOA de la littérature – Tous les rôles SOA de la littérature ont été retrouvés dans Cas 1 – Pas de correspondance pour les rôles d'architecte de solutions et d'architecte de données qui sont présents dans le cas 1 mais pas dans la littérature	– N'a pas pu être déterminée en raison de l'interruption de l'étude de ce cas – Le rôle de l'architecte de solutions identifié à partir du Cas 1 et les tâches qui lui sont associées ont été retrouvés
Autres	– Existence d'un CC SOA qui joue le rôle de comité SOA, tel que présenté dans la littérature	– Instauration d'un groupe (nommé groupe d'intégration) jouant le rôle du comité SOA, , tel que présenté dans la littérature
Aspects non abordés dans la littérature	– Les rôles d'architecte de solutions et d'architecte de données sont présents dans le	– Utilisation de méthodes de développement différentes pour les projets de l'initiative SOA

[32] Applications frontales, services et bus de services

	cas 1 mais n'ont pas été abordés par la littérature – Établissement de nouvelles procédures de tests et d'assurance qualité	– Utilisation d'une méthode de documentation unifiée pour tous les projets impliqués dans l'initiative SOA

Tel que déjà énoncé dans la section IV.2.2.2 et dans le tableau ci-dessus, les changements introduits par SOA sur les *rôles* et les *tâches* du CVP n'ont pu être identifiés que pour le cas 1. Nous les avons, par la suite, confrontés aux *rôles* et aux *tâches* identifiés à partir de la littérature.

Le tableau ci-dessous présente le résultat de cette confrontation. Il montre qu'il y a une forte correspondance entre les *rôles* et les *tâches* SOA de la littérature et ceux présents dans le cas 1. Même si les appellations des *rôles* changent de peu, tous les *rôles* de la littérature ont été retrouvés dans le cas 1. De plus, les *tâches* à l'intérieur des limites de chaque *rôle* sont très similaires. Seuls deux rôles (architecte de solutions et architecte de données) sont présents dans le cas 1 mais n'ont pas été abordés par la littérature.

Tableau 4.8 Correspondance entre rôles de la littérature vs. rôles du cas 1

Rôles (Littérature)	Tâches	Rôles (Cas 1)	Tâches
Analyste d'affaires	• Éliciter les exigences d'affaires • Travailler avec l'équipe de développement au complet pour que ses membres puissent commencer à penser en termes de services • Travailler avec l'équipe technique pour la conception et la construction des services nécessaires, et potentiellement la réutilisation des services existants	**Analyste d'affaires SOA**	• Détailler les besoins en interactions entre services • Effectuer la revue des contrats de services • Rédiger la spécification du service
Architecte d'application	• Traduire les exigences d'affaires en services (ou contrats de services) • Savoir quel service exposer, avec quel niveau de granularité et de quelle façon l'exposer • S'assurer que le code écrit adhère aux principes de l'axé sur les services • S'assurer que toute opportunité de réutilisation de service est bien exploitée • S'assurer que les services sont proprement construits, utilisés et mesurés	**Architecte d'affaires SOA**	• Identifier et classer les services • Déterminer les caractéristiques du service • Déterminer si et somment SOA s'applique • Produire l'architecture de service d'intégration[33]

[33] L'architecture de service d'intégration est une liste des points d'intégration requis pour la réalisation ou la mise à jour de la (ou des) application(s) interpellées par le projet, avec une recommandation du mode optimal d'intégration pour chacun de ces points

Architecte d'entreprise	• Promouvoir et encourager l'adoption de SOA • Communiquer les principes de l'axé sur les services aux architectes SOA et aux groupes de développement • Collaborer avec les architectes SOA dans l'identification des services • Évaluer les contrats de service • Définir les mécanismes de mesure et de suivi des SLA • Définir des politiques et des procédures de gouvernance et de sécurité	**Architecte d'entreprise**	• Approuver la classification des services • Raffiner la classification des services • Approuver les entités d'affaires spécifiées au contrat de service, afin d'assurer qu'elles soient cohérentes avec l'AE[34] • Approuver certaines sections de l'architecture de service d'intégration, afin d'assurer qu'elles soient cohérentes avec l'AE
Comité SOA	• Avoir une responsabilité globale de l'initiative SOA et du contrôle de son application dans les projets en-cours • Définir les rôles, les responsabilités et les procédures pour la gestion des processus et des activités SOA • Définir un processus qui gouverne le cycle de vie des services • Établir des politiques et des processus pour la gestion de la qualité des services • Élaborer un corps de directives SOA incluant les directives métiers et les normes industrielles et organisationnelles • Définir et renforcer les directives qui vont assurer la conformité aux objectifs SOA	**Centre de Compétences SOA (CC SOA)**	• Avoir une responsabilité globale de l'atteinte des objectifs SOA par l'organisation • S'assurer que les différents processus et responsabilités sont performants, claires et adéquatement assumés • Valider l'applicabilité SOA aux projets • Approuver l'architecture de service d'intégration • Approuver les contrats de service

[34] Architecture d'entreprise

Gestionnaire de projet	• Planifier des cycles de développement courts, en raison de la courte durée des cycles de livraison des services • Établir de nouveaux modèles d'acceptation pour les projets SOA • Communiquer avec les utilisateurs et collaborer avec les rôles concernés pour la définition des contrats de service • S'assurer que les spécifications des contrats de service ont bien été respectées	**Directeur de projet SOA**	• Approuver les livrables du projet (contrats de service) • Faire ajuster un contrat de service non approuvé • Inclure les jalons SOA au plan de projet
Administrateur du registre de services	• Enregistrer les services dans le registre de services • Approuver les ajouts et les changements au contenu du registre • S'assurer de la cohérence des données des services publiés	**Administrateur de Service SOA**	• Intégrer les services au registre des services • Publier le registre des services
Architecte SOA	• Définir les exigences non fonctionnelles des services. • Définir les contrats de service en collaboration avec le concepteur de service	**Architecte SOA**	• Détailler le contrat de service • Rechercher les normes de l'industrie reliées au contrat de service • Détailler l'architecture de service d'intégration • Raffiner la classification des services • Raffiner les efforts de réalisation des services • Évaluer les efforts de réalisation des services
N/A	Aucune correspondance dans la littérature	**Architecte de solutions**	• Identifier les points d'intégration avec les systèmes d'information • Collaborer avec les architectes SOA dans les activités spécifiquement SOA du projet

N/A	Aucune correspondance dans la littérature	**Architecte de données**	• Définir les messages XML (Schémas XSD) • Gérer la traçabilité des données • Étudier les standards de données et les domaines de valeurs • Normaliser les domaines de valeurs • Collaborer dans l'évaluation des efforts de réalisation des services SOA • Collaborer dans le raffinement des efforts de réalisation des services
Concepteur de service	• Collaborer avec l'architecte d'applications lors de la création des contrats de services • Modéliser les services (données, fonction, états, interfaces, etc.) • Collaborer avec les rôles concernés lors de la modélisation et de l'orchestration des processus d'affaires, ainsi que dans la liaison entre processus d'affaires et services	**Analyste-Concepteur de service**	• Identifier les éléments de conception • Concevoir le service • Produire la description du service

CHAPITRE V

DISCUSSION

Comme un travail sur la littérature et une étude de cas ont fait l'objet de cette recherche, la discussion se présente comme suit : les sections V.1 et V.2 répondent à nos questions de recherche secondaires en partant, respectivement, de la littérature et des résultats de notre étude de cas et la section V.3 présente une réponse à notre question de recherche principale.

V.1 Littérature

L'abondance de la littérature concernant SOA montre l'importance du phénomène et l'intérêt suscité. Elle a constitué une bonne base pour notre travail sur l'étude de cas, en nous introduisant au « jargon » utilisé pour décrire les pratiques reliées au développement axé sur les services et en nous permettant de dégager les *rôles* et les *tâches* à vérifier dans nos deux cas d'étude. Cependant, nous avons été confrontés à beaucoup d'incohérences dans le contenu proposé par certains livres et articles publiés. Ceci pourrait s'expliquer par le manque de normalisation de la terminologie SOA ou par la nouveauté du concept.

Ainsi, suite à notre travail sur la littérature et en réponse à nos questions de recherche secondaires qui étaient :

- « Dans *la littérature SOA, existe-t-il une conceptualisation cohérente des rôles impliqués dans le CVP et des tâches qui leurs sont associées ?* »,
- « *Quels sont les changements aux rôles* ? »,
- « *Quels sont les changements aux tâches associées aux rôles* ? »,

Nous répondons que, dans la littérature, il existe une conceptualisation cohérente des *rôles* et des *tâches* caractérisant le CVP dans un environnement axé sur les services, mais qui n'a

cependant pas été facile à identifier. En effet, dégager un ensemble de *rôles* et de *tâches* SOA que nous avons jugé cohérent a nécessité une analyse des activités à accomplir durant un projet SOA (section III.2 Activités SOA) et l'élimination de certaines activités camouflées en *rôles*. Les changements aux *rôles* et aux tâches associées aux *rôles* sont présentés dans la section III.3 Rôles et tâches SOA.

V.2 Étude de Cas

V.2.1 Cas 1

Le constat le plus important que nous avons fait concernant le cas I est qu'il existe une forte correspondance entre les *rôles* et les *tâches* identifiés à partir de la littérature et ceux présents dans le premier cas. Le Tableau 4.8 **Correspondance entre rôles de la littérature vs. rôles du cas 1** montre que tous les *rôles* SOA identifiés à partir de la littérature avec leurs *tâches* associées ont été retrouvés dans le cas 1. Seuls deux rôles (architecte de solutions et architecte de données) sont présents dans le cas 1 mais n'ont pas été abordés par la littérature.

À nos deux dernières questions de recherches secondaires qui étaient :

- « *Quels sont les changements aux rôles* ? »,
- « *Quels sont les changements aux tâches associées aux rôles* ? »,

Nous répondons que, dans le cas 1, les changements aux *rôles* et aux *tâches* associées aux *rôles* sont les mêmes que ceux identifiés à partir de la littérature.

Un changement introduit par SOA qui a été constaté dans les pratiques de développement du cas 1, mais qui n'a cependant pas été abordé dans la littérature, se situe au niveau des procédures de tests et d'assurance qualité. Voici comment l'équipe de développement de l'organisation X procède, pour chaque nouveau service identifié :

1. Modélisation détaillée du service.

2. Création de « *stub* » du service[35].

3. Création d'un jeu de tests unique, qui sera enrichi et approuvé par le bureau d'assurance qualité.

4. Mise en œuvre du service et automatisation des tests.

Cette méthode apporte plusieurs avantages à l'organisation X. D'abord, elle génère moins d'anomalies, étant donné que les tests sont conçus avant le développement et que les impacts d'un changement dans un module sur un autre sont détectés dans l'exécution des tests. Ensuite, grâce à cette méthode, le bureau d'assurance qualité peut orienter ses efforts sur la validation et l'analyse des tests, non sur l'exécution de ces derniers. Tous ces avantages offrent une économie d'effort à plus long terme.

Un autre changement introduit par la mise en œuvre de SOA pour le cas 1 est celui de l'adoption d'un nouveau processus de développement logiciel. Il s'agit de *Scrum* ; une nouvelle méthode qui, d'après R1, offre plus d'agilité et des livraisons plus rapides.

V.2.2 Cas 2

Tel que déjà énoncé dans la section IV.2.2.1, le projet DSQ inclut une équipe d'intégration chargée d'intégrer la solution de l'iDSÉ avant de la déployer en région. Le groupe d'intégration a de même la responsabilité de définir les règles de gouvernance au niveau de l'architecture. Ici aussi, il y a concordance avec la littérature qui recommande l'instauration d'un tel groupe jouant le rôle du comité SOA, spécialement dans les initiatives SOA à grande envergure

Une des pratiques de la gouvernance SOA intéressantes à citer pour le cas du projet DSQ est l'utilisation d'une méthode de documentation unifiée. En effet, tous les projets du DSQ utilisent *Macroscope*, une méthode de documentation (développée par la firme DMR) qui a été adoptée comme norme par le ministère de la santé et qui, dans le cadre du DSQ, permet de normaliser la documentation des différents projets.

---

[35] En se basant sur la modélisation détaillée du service (en WSDL) et grâce à l'introduction d'un nouvel outil (LISA), l'équipe de développement peut simuler le service, avant sa mise en œuvre, en générant un « Stub de service ».

Un autre constat important pour notre le cas 2 – qui n'a cependant pas été soulevé par la littérature SOA – est que des méthodes de développement différentes peuvent être utilisées pour les différents projets qui sont impliqués dans une initiative SOA ; mais, ce qui importe le plus, c'est d'avoir une méthode commune pour la documentation de tous les projets. Tel est le cas des projets du DSQ. Cependant, d'après R2, la méthode de développement la plus utilisée par ces derniers est UP ; ce qui confirme les dires de certains auteurs de la littérature SOA, qui voient en UP un des meilleurs candidats pour le développement axé sur les services.

Pour ce qui est des *rôles* et *tâches* du cas 2, nous ne pouvons tirer de conclusion quant à leur concordance ou discordance avec ceux identifiés à partir de la littérature. Cependant, nous pouvons affirmer que R3 accomplit toutes les *tâches* associées à son *rôle* d'architecte de solutions, telles que présentées dans le questionnaire Q3[36]. Nous ne pouvons donc avoir de réponse aux deux dernières sous questions de recherche pour le cas 2.

V.3 Réponse à la question principale

Notre question de recherche principale était :

« Est-ce que le passage vers un environnement axé sur les services pour une organisation demande des changements aux rôles et aux tâches du CVP ? ».

Il est clair qu'en ce qui concerne la littérature que nous avons consultée, la réponse est « Oui ». Cette réponse est évidente puisque tous les auteurs sont d'accord que le passage vers un environnement axé sur les services amène vers des changements organisationnels. Ces changements se traduisent par l'apparition de nouvelles pratiques qui refaçonnent les *rôles* existants et les *tâches* qui leurs sont allouées. Aussi, le fait que l'on ait introduit un nouvel environnement avec de nouveaux principes (principes de l'axé sur les services) amène vers cette évidence.

En ce qui concerne notre étude de cas, nous n'avons pas de réponse univoque à cette question. La « faiblesse » théorique de notre approche (étude exploratoire avec seulement deux cas) ne nous permet pas de généraliser.

---

[36] Les tâches de l'architecte de solutions présentées dans le questionnaire Q3 sont celles identifiées à partir du 1er cas d'étude, étant donné que ce rôle n'a pas été cité dans la littérature.

CHAPITRE VI

CONCLUSION

VI.1 Survol

Dans ce travail de recherche, nous nous sommes intéressés à la pratique SOA en adoptant un point de vue théorique. Ce que l'on a remarqué, c'est que les considérations théoriques ne sont pas si loin de la réalité. Le jargon utilisé par les répondants, ainsi que leurs pratiques sont les mêmes que ceux introduits par les auteurs de la littérature SOA.

On oublie parfois que la théorie est elle-même tirée de la pratique et *vice-versa*. En effet, la plupart des publications prises en considération dans ce mémoire appartiennent à des auteurs, qui sont en même temps des praticiens ayant de l'expérience avec des projets SOA. De plus, le contenu (présentations, *whitepapers*, etc.) utilisé pour introduire les fondements de SOA aux organisations est tiré des articles et des livres publiés. Les influences sont les mêmes pour tout le monde. SOA peut donc être vue comme une évolution normale ; une façon de nommer les nouvelles pratiques qui sont entrain de façonner les environnements TI actuels, suite à de nombreuses influences, parmi lesquelles :

- La volonté d'intégrer les informations de différentes sources qui crée un souci d'interopérabilité – interopérabilité, qui est de même née de la volonté des entreprises et organisations de se découpler des vendeurs spécifiques.

- La prédominance d'Internet et de ses technologies (XML, services Web, etc.) dans les échanges inter-organisationnels et entre organisations et clients – qui a aussi engendré le passage du modèle économique de la chaîne de valeurs à celui du réseau de valeurs. Désormais, pour l'entreprise, la fourniture de valeur ajoutée dépend plus que jamais des sous-traitants, fournisseurs et partenaires, rendant ainsi le modèle de la chaîne de valeurs obsolète.

En particulier, dans ce mémoire, les nouvelles pratiques auxquelles nous nous sommes intéressées sont celles des *rôles* et des *tâches* associées aux rôles dans les environnements axés sur les services. Deux environnements axés sur les services ont été considérés : un environnement centralisé et un environnement décentralisé. Le constat majeur que nous avons fait est que les *rôles* ne sont pas tellement différents de ceux retrouvés dans les autres types d'environnements. Même si les appellations changent de peu, les rôles de l'ingénierie logicielle[37] (gestionnaire de projet, architecte, analyste, etc.) sont toujours là, à la différence que ces derniers sont introduits à de nouvelles *tâches* qui sont en relation avec les principes SOA (identification des services selon une granularité optimale, définition des contrats de services, etc.).

À l'heure où nous rédigeons cette conclusion, un nouvel acronyme est apparu : WOA pour *Web Oriented Architecture*. WOA est vu comme un style architectural ; un sous-style de SOA basé sur l'architecture du Web avec les contraintes additionnelles suivantes : « *globally linked, decentralized, and uniform intermediary processing of application state via self-describing messages* » (Schmelzer, 2008).

WOA ne possède pas encore d'entrée dans Wikipédia – peut être en raison de sa nouveauté ou du faible intérêt qu'il suscite – mais est classé par Gartner comme l'une des dix technologies de l'année 2008. C'est à croire que le phénomène SOA n'est pas encore prêt de s'éteindre. Peut-être est-ce en raison des principes qu'il regroupe ? Interopérabilité, abstraction, autonomie, faible couplage et forte cohésion des composants logiciels (ou services) ne sont pas des concepts nouveaux, mais peut-être que SOA a permis de les regrouper ensemble ?

VI.2 Apprentissage

En entamant cette recherche, j'ai[38] pu appréhender un domaine de connaissances nouveau qui m'a ouvert de nouveaux horizons quant à un éventuel travail de recherche au niveau d'une thèse de doctorat et qui m'a permis d'acquérir de nouveaux outils dont je pourrais me servir en tant que future enseignante.

Grâce au travail sur la littérature, j'ai appris à juger de la qualité des articles scientifiques et des divers types de publications. Avant d'entreprendre ce mémoire, j'avais tendance à croire que

---

[37] Même si, jusque-là aucun consensus n'existe en ce qui concerne les *rôles* et les *tâches* à inclure dans le cycle de vie du projet, nous allons considérer que c'est le cas. En effet, juger de la validité des *rôles* du génie logiciel et des *tâches* qui leurs sont associées n'est pas du ressort de ce mémoire.

toutes les publications se valaient de part la fiabilité des informations qui y sont disséminées. Ce travail m'a permis de développer un certain nombre de critères (ex. utilisation d'une terminologie normalisée, description de la démarche de recherche adoptée, etc.) dont je pourrai me servir dans l'élaboration de mes propres publications.

L'étude de cas menée dans le cadre de ce mémoire ayant été une première expérience pour moi avec les études de cas exploratoires, j'ai pu acquérir et maitriser les notions reliées à ce type de méthodes de recherche qualitatives (définition d'un protocole de l'étude de cas, combinaison de différentes sources de collecte de données qualitatives, conduite des entretiens, etc.). De plus, grâce à cette expérience, l'entrée en contact avec des praticiens, particulièrement dans le cadre du projet DSQ, m'a permis de connaître la façon avec laquelle les projets TI à l'échelle gouvernementale sont abordés et de pouvoir réaliser l'importance des enjeux reliés à ce type de projets.

VI.3 Difficultés rencontrées

Étant donné que notre projet de recherche est basé sur la littérature et sur une étude de cas exploratoire, les difficultés que nous avons rencontrées se situent sur deux plans.

Au niveau de la littérature, la principale difficulté rencontrée a résidé dans le manque de normalisation au niveau de la terminologie SOA – manque de normalisation qui, à notre avis, a engendré des incohérences dans le contenu proposé par certains livres et articles publiés. Actuellement, seul un modèle de référence SOA existe. Il s'agit d'un modèle très pauvre, proposé par OASIS. C'est pour cette raison que lors de la définition des *activités*, *rôles* et *tâches* SOA (chapitre 3), nous avons préféré introduire notre propre terminologie.

En ce qui concerne notre étude de cas, des difficultés ont été rencontrées dans le deuxième cas. En effet, il a été très difficile, voir impossible de rencontrer certains répondants, même si ces derniers avaient préalablement accepté de participer à notre étude. Ceci peut s'expliquer par le fait que le projet (projet DSQ, voir Appendice A) auquel participent les répondants est à sa phase d'exécution, ce qui a causé l'indisponibilité de ces derniers.

---

[38] L'emploi du pronom personnel « je » rend cette section plus personnelle.

VI.4 Limites et contraintes

Ce projet de recherche présente deux principales limites : la première est que notre étude de cas inclut uniquement deux cas et la deuxième est que le l'étude du deuxième cas n'a pas pu être finalisée. Ces dernières limites confèrent à la partie de notre recherche concernant la littérature un poids plus grand que la partie étude de cas et supposent un manque de validation pour les résultats et la conclusion que nous proposons.

Au fur et à mesure de son avancement, notre recherche a fait l'objet de plusieurs modifications au niveau son objectif et de ses questions de recherche. Le passage d'une version à une autre (objectif, questions de recherche, raisons du changement et travail effectué sur chaque version) est expliqué dans l'appendice C. L'évolution de notre démarche nous semble relativement intéressante car elle met en évidence certaines difficultés qui sont loin d'être exclusives à notre travail. L'engouement pour SOA (comme pour tout nouveau champ de recherche qui a des retombées économiques immédiates) crée une floraison d'articles et de concepts où il est parfois difficile de différencier les vraies nouveautés des concepts qui reviennent sous de nouveaux noms[39]. C'est à cause de cette « richesse » conceptuelle que la partie de notre étude concernant la littérature a un poids plus grand que la partie étude de cas.

VI.5 Contribution et travaux futurs

La contribution de cette recherche est qu'elle constitue un pas de plus dans l'étude du phénomène SOA et de son impact sur le CVP. Elle permet d'affirmer que la mise en œuvre d'une SOA entraîne l'apparition de nouvelles pratiques qui refaçonnent les *rôles* existants et les *tâches* qui leurs sont allouées.

Étant donné le manque de validation de la conclusion que nous proposons, il serait intéressant, dans un travail futur, d'étendre notre recherche à plus d'un cas afin de pouvoir, éventuellement, tirer des conclusions plus générales concernant les impacts de SOA sur le CVP.

---

[39] Et souvent, hélas de nouveaux acronymes !

RÉFÉRENCES

Anaya V. et A. Ortiz (2005). « How enterprise architectures can support integration ». Proceedings of the first international workshop on Interoperability of heterogeneous information systems, pp. 25-30.

Arsanjani A. (2004). « Service-oriented modeling and architecture: how to identify, specify, and realize services for your SOA ». IBM whitepaper. URL : ftp://www6.software.ibm.com/ software/developer/library/ws-soa-design1.pdf

Balzer Y. (2004). « Improve your SOA project plans ». IBM DeveloperWorks Library. URL : http://www.ibm.com/developerworks/library/ws-improvesoa/

Bennett K., P. Layzell, D. Budgen, P. Brereton, L. Macaulay et M. Munro (2000). « Service-based software: The future for flexible software ». Proceedings of the 7th Asia-Pacific Software Engineering Conference, pp. 214.

Bieberstein N., S. Bose, M. Fiammante, K. Jones et R. Shah (2005). « Service-Oriented Architecture Compass: Business Value, Planning and Enterprise Roadmap ». IBM Press.

Bloomberg J. et R. Schmelzer (2006). « Service Orient or Be Doomed! ». Wiley Editions, 258 pages.

Bonnet P. (2005). « Cadre de référence Architecture SOA - partie1 ». Orchestra Networks. URL : http://pie.bonnet.ifrance.com/ON-guideSOA-2005-02-23%20(part1).pdf

Bonoma T. V. (1985). « Case research in marketing: Opportunities, problems, and a process ». Journal of Marketing Research, vol. 22, 199-208.

Chang S. H. et Kim S. D. (2007) « A Systematic Approach to Service-Oriented Analysis and Design ». Springer-Verlag Berlin Heidelberg.

Chen F., S. Li, H. Yang, C-H.Wang, W. C-C. Chu (2005). « Feature analysis for service-oriented reengineering ». Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05).

Denzin N. K. et Y. S. Lincoln (2003). « Collecting and interpreting qualitative materials ». SAGE publications, second edition.

Erl T. (2005). « Service-Oriented Architecture: Concepts Technology and Design ». Prentice Hall.

Gibbs R. D. (2006). « Project Management with the IBM Rational Unified Process - Outsourcing and the IBM Rational Unified Process ». IBM Press.

Huang Y., S. Kumaran et J.Y. Chung (2004). « A service management framework for service-oriented enterprises ». Proceedings of the IEEE International Conference on E-Commerce Technology, pp. 181-186.

Hurwitz J., R. Bloor, C. Baroudi, M. Kaufman (2007). « Service Oriented Architecture for Dummies ». Wiley Publishing, pp, 131-140, pp. 167-180.

Ivanyukovich A., G. R. Gangadharan, V. D'Andrea, M. Marchese (2005). « Towards a service-oriented development methodology ». Journal of integrated design & process science. Vol. 9, No. 3, pp. 53-62.

Jacobson I., G. Booch et J. Rumbaugh (2000). « Le processus unifié de développement logiciel ». Éditions Eyrolles.

Joffroy C., S. Mosser, M. Blay-Fornarino et C. Nemo (2007). « Des Orchestrations de Services Web aux Aspects ». JFDLPA'07.

Johnston S. (2005). « UML 2.0 Profile for Software Services ». IBM DeveloperWorks Library. URL: www.ibm.com/developerworks/rational/library/05/419_soa/

Kajko-Mattsson M., G. A. Lewis et D. B. Smith (2007). « A Framework for Roles for Development, Evolution and Maintenance of SOA-Based Systems ». International Workshop on Systems Development in SOA Environments (SDSOA'07).

Karhunen H., M. Jantti et A. Eerola (2005).« Service-oriented software engineering (SOSE) framework ». Proceedings of the Services Systems and Services Management (ICSSSM '05). Vol. 2, pp. 1199-1204.

Kavianpour M (2007). « SOA and Large Scale and Complex Enterprise Transformation ». Service-Oriented Computing – ICSOC 2007, Volume 4749/2007, pp. 530-545.

Kim Y. et H. Yun (2006). « An Approach to Modeling Service-Oriented Development Process ». IEEE International Conference on Services Computing (SCC'06), pp. 273-276.

Kim Y. et K.-G.Doh (2007). « The Service Modeling Process Based on Use Case Refactoring ». Lecture notes in computer science - Springer. URL : http://www.springerlink.com/content/a4wv33g2u6v3463v/

Krafzig D., K. Banke et D. Slama (2004). « Enterprise SOA: Service-Oriented Architecture Best Practices ». Prentice Hall PTR.

Krogdahl P., G. Luef et C. Steindl (2005). « Service-oriented agility: An initial analysis for the use of agile methods for SOA development ». Proceedings of the IEEE International Conference on Services Computing (SCC'05). Vol. 2, pp. 93-100.

Kruchten (2000). « The Rational Unified Process: An Introduction, Second Edition ». Addison Wesley, 320 pages.

Marks E. A. et M. Bell (2006). « Service-Oriented Architecture: A planning and Implementation Guide for Business and Technology ». Wiley Editions, 375 pages.

Mittal K. (2006). « Build your SOA, Part 3: The Service-Oriented Unified Process ». IBM developerWorks. URL : www.ibm.com/developerworks/library/ws-soa-method1.html

Mittal K. (2006b). « Create the Ideal SOA Team ». IBM developerWorks. URL : http://www.ibm.com/developerworks/library/ar-soateam/index.html

Morse J. M. (1994). « Designing Funded Qualitative Research ». Handbook of Qualitative Research, pp. 220-235..

Newcomer E. et G. Lomow (2004). « Understanding SOA with Web Services ». Independent Technology Guides. Addisson Wesley, 444 pages.

Niclot S. (2005). « Alternative à RUP: dX ». Conservatoire National des Arts et Métiers : Centre Enseignement Principal de Tours. URL: http://tbrouard.univ-tours.fr/t/IMG/pdf/dX.pdf

Office Québécois de la Langue Française (OQLF). Dernière consultation : 14 Avril 2008. URL : http://www.granddictionnaire.org

Papazoglou M. P. (2003). « Service -Oriented Computing: Concepts, Characteristics and Directions ». Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03).

Papazoglou M. P. (2007). « What's in a Service ». Springer-Verlag Berlin Heidelberg.

Paré G. (2004). « Investigating Information Systems with Positivist Case Study Research ». Communications of the AIS, 13, 2004, pp.231-264.

Perepletchikov M. et al. (2005) «The impact of software development strategies on project and structural software attributes in SOA ». Proceedings of the 2nd INTEROP Network of Excellence Dissemination Workshop (INTEROP). URL: http://goanna.cs.rmit.edu.au/~caspar/downloads/INTEROP2005paper.pdf

Pulier E. et H. Taylor (2006). « Understanding Enterprise SOA ». Manning Publications, pp. 59-72.

Schmelzer R. (2008). « WOA is Me - Another Acronym? WOA and SOA ». ZapFlash Document. URL : http://www.zapthink.com/report.html?id=ZAPFLASH-20085 16

Stojanovic Z. et A. Dahanayake (2005). « Service-Oriented Software System Engineering: challenges and practices ». Idea Group Publishing, pp. 27-47.

Stojanovic Z., A. Dahanayake et H. Sol (2004). « Modeling and design of service-oriented architecture ». IEEE International Conference on Systems, Man and Cybernetics. Vol. 5, pp. 4147-4152.

Strnadl C. F. (2007). « Bridging Architectural Boundaries Design and Implementation of a Semantic BPM and SOA Governance Tool ». Lecture notes in computer science, Springer.

Tsai T., X. Wei, Z. Cao, R. Paul, Y. Chen et J. Xu (2007). « Process Specification and Modeling Language for Service-Oriented Software Development ». Proceedings of the 11th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'07), pp. 181-188.

Tsai W.T., Y-H. Lee, Z. Cao, Y. Chen et B. Xiao (2006). « RTSOA: Real-Time Service-Oriented Architecture ». Proceedings of the Second IEEE International Symposium on Service-Oriented System Engineering (SOSE'06).

Yin R. K. (2003). « Case Study Research: Design and Methods ». SAGE Publications, pp.57-97.

Yukyong K. et Y. Hongran (2006). « An Approach to Modeling Service-Oriented Development Process ». IEEE International Conference on Services Computing (SCC'06), pp. 273-276.

Zhang T., S. Ying, S. Cao et X. Jia (2006). « A Modeling Framework for Service-Oriented Architecture ». Proceedings of the Sixth International Conference on Quality Software (QSIC'06), pp. 219-226.

Zimmermann O., N. Schlimm, G. Waller et M. Pestel (2005). « Analysis and Design Techniques for Service-Oriented Development and Integration ». Informatik 2005 SOA Workshop.

Zimmermann O., P. Krogdahl et C. Gee (2004). « Elements of Service-Oriented Analysis and Design. An interdisciplinary modeling approach for SOA projects ». IBM DeveloperWorks. URL : http://www.ibm.com/developerworks/library/ws-soad1/

APPENDICE A

LE DOSSIER DE SANTÉ DU QUÉBEC (DSQ)

Le Dossier de Santé Électronique (DSÉ)

Inforoute Santé du Canada (Inforoute) est une organisation indépendante à but non lucratif, financée par le gouvernement fédéral du Canada (Site web Inforoute). En collaboration avec des partenaires du secteur public et privé à travers le Canada (ministères de la Santé, régies régionales de la santé, autres organismes de soins de santé et fournisseurs de systèmes d'information), elle travaille pour la mise en œuvre et la réutilisation de systèmes d'information de santé compatibles qui supportent un système de santé plus efficace. L'objectif est d'intégrer les différents prestataires et les différentes administrations de la santé (ex. hôpitaux, laboratoires, médecins, pharmacies et agences gouvernementales) dans chaque province et de les connecter afin de former un réseau national de la santé avec des formats de données et des protocoles de communication normalisés et un fichier unique de l'historique de la santé du patient (Jahnke-Weber et al., 2007). Ce fichier, constituant le dossier de santé électronique (DSÉ) du patient, contient l'information de santé du patient, accessible d'une façon omniprésente, en utilisant des services communs selon différents privilèges d'accès pour les patients et les fournisseurs. La solution proposée pour le DSÉ, nommée infostructure du DSÉ (iDSÉ) est :

- Compatible avec des standards et des technologies de communication prédéfinis ;
- Inclut l'information reliée à la santé actuelle et historique, aux conditions médicales et aux tests médicaux des sujets ;

- Peut être accédée à travers les différents points de service du réseau de la santé (pharmacies, laboratoires, hôpitaux, etc.).

L'iDSÉ a été conçue en se basant sur une architecture SOA. Les principaux composants de l'architecture SOA de l'iDSÉ sont :

1. Les applications de points de service (dans les systèmes de pharmacies, de laboratoires, d'hôpitaux, etc.) et les visualiseurs de DSÉ qui sont considérés comme des applications frontales.

2. Les services, qui sont publiés dans trois (3) principaux registres de services (« données et services de registre », « données et services auxiliaires » et « données et services de DSÉ »).

3. La couche d'accès à l'information sur la santé (CAIS) qui connecte les applications frontales aux services.

Le Dossier de Santé du Québec (DSQ)

À compter du printemps 2008, la région de la Capitale-Nationale sera la première région du Québec et du Canada où la solution de l'iDSÉ sera utilisée[40]. Ce projet pilote, nommé DSQ (pour Dossier de Santé du Québec) est en-cours de réalisation et est conforme à l'architecture SOA établie pour la solution iDSÉ. Notre deuxième cas d'étude pour cette recherche est celui du DSQ.

---

[40] Source : Ministère de la Santé et des Services Sociaux du Québec. URL : http://www.dossierdesante.gouv.qc.ca/

APPENDICE B

GUIDE DES ENTRETIENS

Cet appendice regroupe les questionnaires élaborés pour les répondants de nos deux cas d'étude. Q1, Q2 et Q3, les questionnaires qui ont été élaborés respectivement pour R1, R2 et R3 sont remplis par ces derniers. Q4 et Q5, les questionnaires destinés à R4 et R5 ne sont pas remplis, étant donné que les répondants n'ont finalement pas pu se rendre disponibles pour notre recherche.

QUESTIONNAIRE Q1 (rempli par R1)

I. Combien d'années d'expérience possédez-vous avec les projets SOA[41] et quels postes avez-vous occupé durant ces projets ?

Années d'expérience	
Moins de 1 an	
1 à 2 ans	X
2 à 4 ans	
4 à 6 ans	
Plus de 6 ans	

Postes occupés :

R1 : Gestionnaire de projet, directeur du centre de compétences SOA. La mise en œuvre d'une SOA au niveau de notre organisation constitue ma première expérience avec SOA----------

II. Combien d'années d'expérience possédez-vous en informatique, dans d'autres projets, et quels postes avez-vous occupé durant ces projets ?

Années d'expérience	
1 à 5 ans	
5 à 10 ans	
10 à 15 ans	
Plus de 15 ans	x

[41] *Service-Oriented Architecture*

Postes occupés :

R1 : Analyste-développeur, Architecte d'applications, gestionnaire de projet----------

---

III. L'adoption des principes SOA implique tous les secteurs de l'organisation. Ces principes doivent être compris et acceptés par tous. C'est pour cette raison que l'on préconise dans la littérature SOA la mise en place d'un comité SOA. Le comité SOA est une entité organisationnelle responsable de l'initiative SOA et du contrôle de son application dans les projets en-cours.

Le centre de compétences SOA de X constitue-t-il une entité organisationnelle similaire au comité SOA ?

R2 : Oui----------

Sinon, existe-t-il, à X, une entité organisationnelle jouant le rôle du comité SOA ?

IV. Parmi la liste suivante, quels sont les rôles qui font partie de votre centre de compétences SOA et y a-t-il plusieurs personnes avec le même rôle ?

Membres	Oui	Nombre	Commentaires

Architecte d'entreprise	x	2	
Architecte d'application			
Gestionnaire de projet	x	1	
Analyste d'affaires	x	1	
Concepteur de services			
Développeur de services			
Autres : – Subject Matter Expert (SME) – Spécialistes SOA (Test et *Versionning*) – Architecte SOA			Les SME font partie de l'équipe de nos consultants/partenaires SOA

V. Dans la liste précédente, y a-t-il des personnes qui possèdent plus d'un rôle ?

R1 : *Oui, moi. Je suis le directeur du centre de compétences SOA et gestionnaire de projet*----------

---

VI. Quel est l'ensemble des tâches et des responsabilités qui vous sont accordées dans le cadre de votre poste de directeur du centre de compétences SOA de X ?

R1 : *M'assurer que le comité SOA rempli sa mission qui est d'avoir une responsabilité globale dans l'atteinte des objectifs SOA fixés par l'organisation. Notre travail consiste à valider l'applicabilité SOA aux projets et à nous assurer que les différents*

processus et responsabilités sont performants, claires et adéquatement assumés -----

---

VII. Parmi la liste suivante, quels sont les objectifs du centre de compétences SOA dans le cadre des objectifs de haut niveau de l'initiative SOA de X ?

Objectifs	Oui	Commentaires
Avoir une responsabilité globale de l'initiative SOA et du contrôle de son application dans les projets en-cours	x	
Définir les rôles, les responsabilités et les procédures pour la gestion des processus et des activités SOA	x	
Définir un processus qui gouverne le cycle de vie des services	x	
Établir des politiques et des processus pour la gestion de la qualité des services	x	
Élaborer un corps de directives SOA incluant les directives métiers et les normes industrielles et organisationnelles	x	
Définir et renforcer les directives qui vont assurer la conformité aux objectifs SOA	x	

VIII. Existe-t-il d'autres objectifs du centre de compétences SOA qui n'ont pas été cités dans la liste précédente ? Si oui, lesquels ?

IX. Quelle est la méthode de développement utilisée pour les projets SOA de BNC (voir Liste) ?

Méthode de développement	
Modèle en cascade	
Modèle en spirale	
Rapid Application Development (RAD)	
Unified Process (UP)	
Dynamic Systems Development Method (DSDM)	
Microsoft Solution Framework (MSF)	
Catalysis	
Feature Driven Development (FDD)	
Adaptive Software Development (ASD)	
Extreme Programming (XP)	
Crystal	
Scrum	x
dX	
Autre :	

Commentaires : ----------

X. Cette méthode est-elle différente de la méthode ou des méthodes utilisée(s) pour les autres types de projets ? Si oui, en quoi est-elle différente et quelles sont les raisons de ce choix ?

R2 : *Il s'agit d'une nouvelle méthode qui offre plus d'agilité et des livraisons plus rapides*----------

---

QUESTIONNAIRE Q2 (rempli par R2)

I. Combien d'années d'expérience possédez-vous avec les projets SOA[42] et quels postes avez-vous occupé durant ces projets ?

Années d'expérience	
Moins de 1 an	
1 à 2 ans	
2 à 4 ans	X
4 à 6 ans	
Plus de 6 ans	

Postes occupés :

R2 : Architecte-intégrateur – volet normes et standards.

II. Combien d'années d'expérience possédez-vous en informatique, dans d'autres projets, et quels postes avez-vous occupé durant ces projets ?

Années d'expérience	
1 à 5 ans	
5 à 10 ans	
10 à 15 ans	
Plus de 15 ans	x

Postes occupés :

R2 : Architecte-intégrateur, Spécialiste HL7, concepteur logiciel, chargé de projet, Analyste, Analyste-programmeur.

---

[42] *Service-Oriented Architecture*

Selon la documentation publique[43] du Ministère de la Santé et des Services Sociaux du Québec, la mise en œuvre du Dossier de Santé Électronique (DSÉ) implique des équipes de projet régionales. Chaque équipe (voir Figure 1), accompagnée par l'équipe du Dossier de santé, assurera la pleine réalisation sur son territoire d'une infostructure DSÉ (voir annexe A) qui sera interopérable avec les infostructures DSÉ des autres régions.

III. Faites-vous partie de l'équipe de projet DSÉ de la région de Montréal. Si oui, quel est votre rôle au sein de cette équipe ?

R2 : Non. Je fais partie du groupe d'intégration au niveau du Programme DSQ (le projet qui chapeaute l'ensemble des projets). Ce groupe d'intégration inclus une équipe d'architecture dont je fais partie, une équipe de déploiement, une équipe d'exploitation, d'infrastructure et de télécoms; toutes ces équipes travaillent au niveau du programme DSQ pour intégrer la solution avant de la déployer en région.

---

[43] « Le Dossier de Santé du Québec : Plan d'affaires 2007-2010 »
http://www.dossierdesante.gouv.qc.ca/

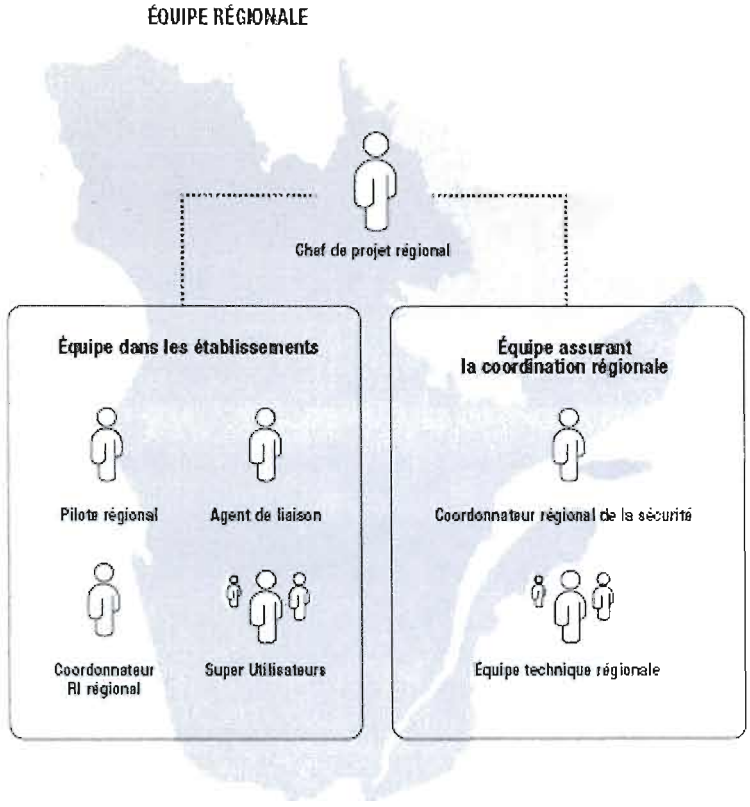


Figure 1 : Rôles type au sein de l'équipe régionale de projet

IV. Sinon, de quelle façon participez-vous au projet DSÉ ?

R2 : En tant qu'architecte-intégrateur, je suis responsable des normes et standards. Je m'assure que le développement utilise les normes du MSSS et celles d'Inforoute Santé Canada. Je suis aussi responsable de la mise en place d'un processus de conformité des solutions.

Selon une perspective de haut niveau, un système fondé sur une architecture SOA est constitué de :

1. **Services** et **registre de services**, où les services sont publiés
2. **Applications frontales**, qui découvrent et utilisent les services publiés
3. **Infrastructure SOA**, qui connecte les applications aux services

V. Êtes-vous d'accord avec les énoncés du tableau suivant ?

Énoncé : Dans le cas du DSÉ, et selon l'annexe A, ...	**Oui**	**Non**	**Commentaires**
1. Les applications de points de service (dans les systèmes de pharmacies, de laboratoires, d'hôpitaux, etc.) et les visualisateurs de DSÉ sont considérés comme des **applications frontales**	x		Notez que les applications frontales ne découvrent pas dynamiquement les services publiés.
2. Les **services** sont distribués sur différentes parties de cette architecture et il existe 3 principales catégories de **registres de services** (« données et services de registre », « données et services auxiliaires » et « données et services de DSÉ »)	x		Chaque catégorie de registre contient des services d'affaires et des données d'affaires. Les services d'affaires sont mis en œuvre par des services Web
3. La couche d'accès à l'information sur la santé (CAIS) est responsable de la fonctionnalité de l'**infrastructure SOA**	x	x	Oui et non. En fait, CAIS fourni l'infrastructure, mais la fonctionnalité est fournie par plusieurs composantes de l'infostructure.

VI. Dans le contexte du DSÉ, quelles sont les organisations chargées du développement des trois composants[44] de l'architecture SOA, cités dans la question précédente. Afin de mieux les identifier, nous proposons pour chaque développeur de composant[45], un ensemble de tâches spécifiques :

	Principales tâches[46]	Organisation(s)	Commentaires
Développeur d'infrastructure	• Sélectionner les standards et les produits qui seront mis en œuvre dans l'infrastructure • Développer un ensemble de services d'infrastructure tel que la découverte, la communication et la sécurité • Identifier et développer des mécanismes de lien qui pourront satisfaire un grand ensemble d'utilisateurs potentiels de services • Fournir des outils pour les développeurs de services et d'applications • Documenter et maintenir l'infrastructure	Ministère de la Santé et des Services sociaux (MSSS) Équipe d'intégration du DSQ Contracteur de l'appel d'offre du projet CAIS	Les normes et standards sont sélectionnés par le MSSS et l'équipe d'intégration DSQ. Les services et outils sont développés par l'attributaire de la CAIS.
Développeur d'applications	• Comprendre l'infrastructure SOA • Découvrir quels sont les services appropriés qui doivent être incorporés aux applications • Invoquer les services identifiés dans les applications (incluant la conversion des données, le traitement des erreurs, etc.) • Tester l'exactitude des services dans le contexte de l'application en cours de développement	Contracteurs de l'appel d'offres des projets de domaines et applications frontales (DSE, SRC-Labo, RID-PACS, DCI) Pharmacies (4 compagnies) Médecins, hôpitaux	

[44] (1) Services et registres de services, (2) Applications frontales et (3) Infrastructure SOA

[45] Ici, « développeur de composant » se réfère à l'organisation responsable du composant (conception, mise en œuvre, maintenance, etc.), qui peut être assistée par un ensemble de collaborateurs dans l'accomplissement de ses tâches.

[46] Source : Kajko-Mattsson et al. (2007)

Développeur de services	• Définir les exigences des utilisateurs potentiels de services • Comprendre l'infrastructure SOA • Développer le code qui reçoit la requête de service, la traduit en appels destinés à de nouveaux systèmes ou à des systèmes existants et produit une réponse • Décrire et publier les services dans les registres • Développer le code d'initialisation et les procédures fonctionnelles du service	Contracteurs de l'appel d'offres des projets de domaines et registres (DES, SRC-Labo, RID-PACS, Médicaments, RU, RI, ODS-LDS)	

Dans la littérature SOA, on conseille la mise en place d'un comité spécifique chargé de définir les pratiques de la gouvernance SOA. Par exemple, il aurait pour responsabilité de :

- Déterminer la supervision de l'architecture SOA (ex. Qui possède les standards et le contrôle de la conformité aux directives SOA? Qui détermine les niveaux appropriés de granularité des services ?).
- Établir des directives SOA (ex. directives de conformité aux objectifs et aux standards SOA, directives de sécurité).
- Mettre en œuvre le processus de gouvernance SOA (ex. gestion des interdépendances entre les services partagés, gestion des conflits entre les organisations).
- Gouverner la définition, la création et la publication des services (ex. comment les services seront définis, développés et modifiés? Qui aura l'autorité de conception? En se basant sur quelles exigences? Qui possède les services?)
- Établir des politiques et des processus pour la gestion de la qualité des services (ex. Qui se charge de renforcer les contrats de service ? Quelles sont les technologies qui pourront permettre de renforcer les directives et de mettre en œuvre la gestion des services?).

VII. Dans le cas du projet DSÉ, y a-t-il un comité spécifique chargé de définir les principes de la gouvernance SOA ? Si oui, qui sont les membres de ce comité ?

R2 : C'est le groupe d'intégration qui est responsable de définir les règles de gouvernance, au niveau de l'architecture; les décisions sont prises par le comité de gestion du projet. Pour l'exploitation et l'opération de la solution, les règles seront définies par un autre groupe (exploitation, opérations).

VIII. Quelle est la méthode de développement (ou la méthode à laquelle s'apparente la méthode de développement) employée par la compagnie de consultants dont vous faites partie pour les projets SOA ?

Méthode de développement	
Modèle en cascade	
Modèle en spirale	
Rapid Application Development (RAD)	
Unified Process (UP)	
Dynamic Systems Development Method (DSDM)	
Microsoft Solution Framework (MSF)	
Catalysis	
Feature Driven Development (FDD)	
Adaptive Software Development (ASD)	
Extreme Programming (XP)	
Crystal	
Scrum	
dX	
Autre : Concert	x

R2 : Concert est la méthode de développement la plus préconisée par la compagnie de consultants dont je fais partie pour les projets SOA.

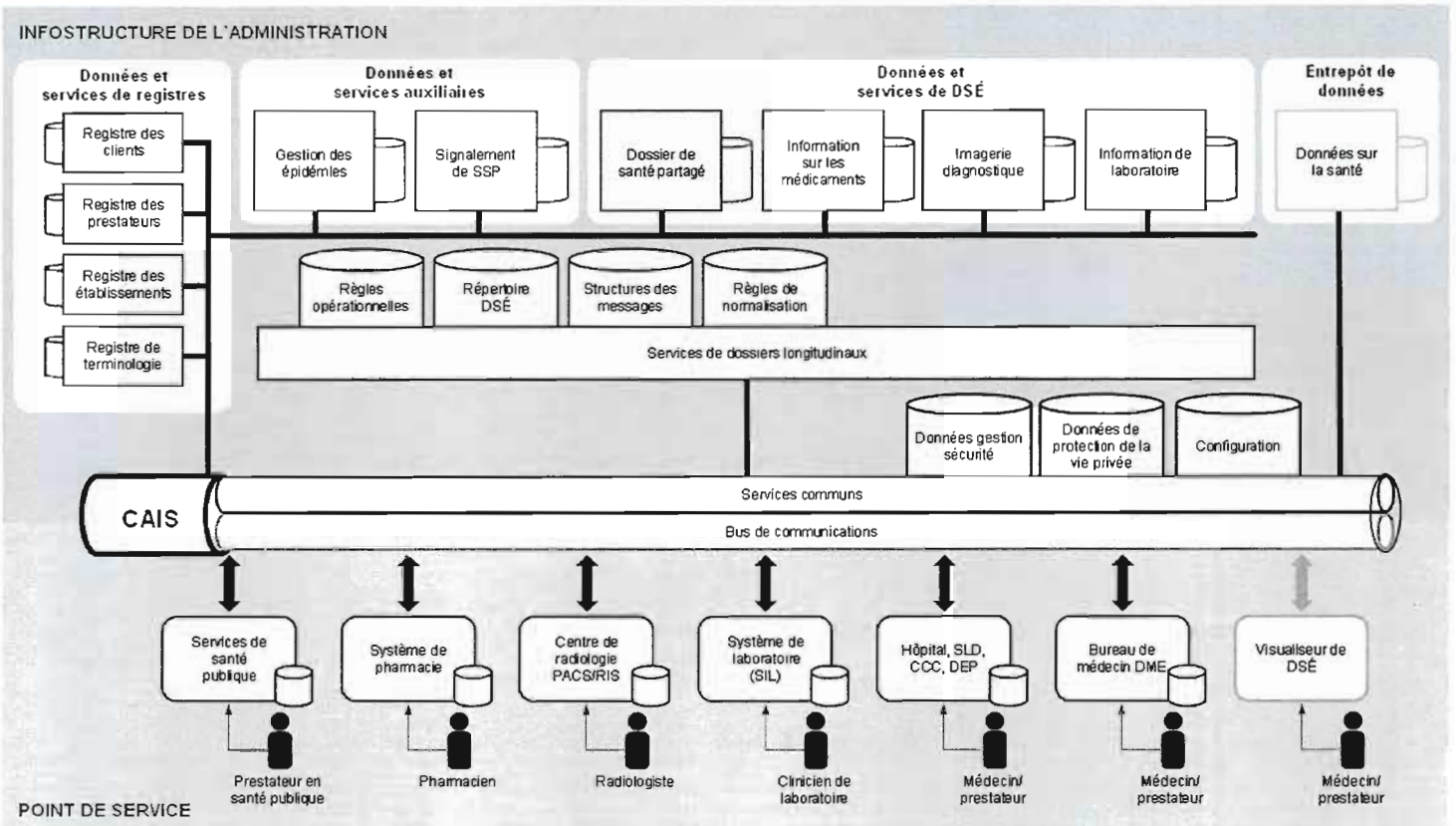
IX. Cette méthode est-elle différente de la méthode ou des méthodes utilisée par la firme de consultation à laquelle vous êtes affilié pour les autres types de projets ? Si oui, en quoi est-elle différente et quelles sont, d'après vous, les raisons de ce choix ?

R2 : Une firme de consultation aide les clients à développer et implanter des solutions. La méthode varie donc d'un client à l'autre.

X. S'agit-il de la même méthode employée dans les projets du DSQ ? Sinon, quelle est la méthode employée pour les projets du DSQ et en quoi est-elle différente de la méthode employée par CGI pour les projets SOA ?

R2 : Les projets du DSQ utilisent Macroscope, une méthode de documentation développée par la firme DMR. Il s'agit d'une norme adoptée par le ministère de la santé qui, dans le cadre du DSQ permet de normaliser la documentation des différents projets. Des méthodes de développement différentes peuvent être utilisées, mais ce qui importe le plus, c'est d'avoir une méthode commune pour la documentation des différents projets qui sont impliqués dans une initiative SOA. Tel est le cas des projets du DSQ, qui utilisent différentes méthodes de développement. Cependant la méthode de développement la plus utilisée est UP.

Annexe A : Infostructure de DSÉ - Architecture Conceptuelle[47]



[47] Source : Inforoute Santé du Canada http://www.infoway-inforoute.ca/fr/home/home.aspx

Annexe B : Principaux nouveaux rôles pour le développement des composants d'une architecture SOA

Rôles	Tâches
Analyste d'affaires SOA	• Détailler les besoins en interactions entre services • Effectuer la revue des contrats de services • Rédiger la spécification du service
Architecte d'affaires SOA	• Identifier et classer les services • Déterminer les caractéristiques du service • Déterminer si et somment SOA s'applique
Architecte d'entreprise	• Approuver la classification des services • Raffiner la classification des services • Approuver les entités d'affaires spécifiées au contrat de service
Comité SOA	• Responsabilité globale de l'atteinte des objectifs SOA • S'assurer que les différents processus et responsabilités sont performants, claires et adéquatement assumés • Valider l'applicabilité SOA aux projets • Approuver l'architecture de service • Approuver les contrats de service
Directeur de projet SOA	• Approuver les livrables du projet (contrats de service) • Contester une recommandation SOA • Faire ajuster un contrat de service non approuvé • Inclure les jalons SOA au plan de projet
Administrateur de Service SOA	• Intégrer les services au registre de services • Gérer le registre des services
Architecte SOA	• Détailler l'architecture de service • Détailler le contrat de service • Raffiner les efforts de réalisation des services • Raffiner la classification des services • Rechercher les normes de l'industrie reliées au contrat de service • Évaluer les efforts de réalisation des services SOA
Analyste-Concepteur de service	• Concevoir le service • Identifier les éléments de conception • Produire le WSDL du service

QUESTIONNAIRE Q3 (rempli par R3)

I. Combien d'années d'expérience possédez-vous avec les projets SOA[48] et quels postes avez-vous occupé durant ces projets ?

Années d'expérience	
Moins de 1 an	
1 à 2 ans	
2 à 4 ans	x
4 à 6 ans	
Plus de 6 ans	

Postes occupés : *Architecte d'affaires, Architecte de solutions. Ma participation à la phase 1 du DSQ a constitué ma première expérience SOA* ----------

II. Combien d'années d'expérience possédez-vous en informatique, dans d'autres projets, et quels postes avez-vous occupé durant ces projets ?

Années d'expérience	
1 à 5 ans	
5 à 10 ans	
10 à 15 ans	
Plus de 15 ans	x

Postes occupés : *Chargé de projet, analyste d'affaires et architecte* ----------
---
---

[48] *Service-Oriented Architecture*

III. Selon la documentation publique[49] du Ministère de la Santé et des Services Sociaux du Québec, la mise en œuvre du Dossier de Santé du Québec (DSQ) implique plusieurs projets qui ont pour finalité de fournir les composantes technologiques de la solution de DSQ. Faites-vous partie de l'équipe du projet SRC-Labo[50] ? Si oui, quel est votre rôle au sein de cette équipe ?

Actuellement, je suis architecte de solutions pour le projet SRC-Labo. Je suis affilié au Ministère de la Santé et des Services Sociaux et je collabore avec l'organisation bénéficiaire de l'appel d'offres du projet SRC-Labo. J'ai également été architecte intégrateur durant la phase 1 du DSQ et architecte intégrateur pour le projet Imagerie-Diagnostique ----------
---

IV. Sinon, de quelle façon participez-vous au DSQ ?

---

[49] « Le Dossier de Santé du Québec : Plan d'affaires 2007-2010 » http://www.dossierdesante.gouv.qc.ca/
[50] Le projet SRC-Labo (Services Régionaux de Conservation et Laboratoires) fait partie de l'ensemble des projets travaillant pour la réalisation du DSQ.

Selon une perspective de haut niveau, un système fondé sur une architecture SOA est constitué de :

1. **Services**, dont la mise en œuvre fournit la logique d'affaires et **registre de services**, où les services sont publiés
2. **Applications frontales**, qui découvrent et utilisent les services
3. **Bus de services**, qui connecte les applications aux services

V. Êtes-vous d'accord avec les énoncés du tableau suivant ?

Énoncé : Dans le cas du DSQ, et selon l'annexe A, ...	**Oui**	**Non**	**Commentaires**
1. Les applications de points de service (dans les systèmes de pharmacies, de laboratoires, d'hôpitaux, etc.) et les visualiseurs de DSÉ[51] sont considérés comme des **applications frontales**	x		
2. Les **services** sont distribués sur différentes parties de cette architecture et il existe 3 principaux **registres de services** (« données et services de registre », « données et services auxiliaires » et « données et services de DSÉ »)	x		
3. La couche d'accès à l'information sur la santé (CAIS) est responsable de la fonctionnalité du **bus de services**	x		

[51] Dossier de Santé Électronique

VI. Parmi les composants de l'architecture SOA[52] cités dans la question précédente, quels sont ceux dont le projet du DSQ auquel vous êtes affilié possède la responsabilité de développement. Afin de mieux les identifier, nous proposons pour chaque composant un ensemble de tâches spécifiques de développement :

	Principales tâches de développement[53]	Oui	Non	Commentaires
Bus de services	• Sélectionner les standards et les produits qui seront mis en œuvre dans le bus de services • Développer un ensemble de services d'infrastructure tel que la découverte, la communication et la sécurité • Identifier et développer des mécanismes de lien qui pourront satisfaire un grand ensemble d'utilisateurs potentiels de services • Fournir des outils pour les développeurs de services et d'applications • Documenter et maintenir le bus de services			
Applications frontales	• Comprendre l'infrastructure SOA • Découvrir quels sont les services appropriés qui doivent être incorporés aux applications • Invoquer les services identifiés dans les applications (incluant la conversion des données, le traitement des erreurs, etc.) • Tester l'exactitude des services dans le contexte de l'application en cours de développement	x		La compagnie avec laquelle nous collaborons et qui a bénéficié de l'appel d'offres du projet SRC-Labo est responsable de ces tâches. Mon rôle est de valider le travail exécuté pour ces tâches.
Services et registres de services	• Définir les exigences des utilisateurs potentiels de services • Comprendre l'infrastructure SOA • Développer le code qui reçoit la requête de service, la traduit en appels destinés à de nouveaux systèmes ou à des systèmes existants et produit une réponse • Décrire et publier les services dans les registres • Développer le code d'initialisation et les procédures fonctionnelles du service	x		Même commentaire que pour « Applications frontales »

[52] (1) Services et registres de services, (2) Applications frontales et (3) Bus de services
[53] Source : Kajko-Mattsson et al. (2007)

Selon la littérature SOA, et d'après les résultats d'une étude de cas que nous avons mené auprès d'une institution financière ayant entamé une initiative SOA, différents ensembles de rôles et de tâches associées à ces rôles sont nécessaires au développement des composants d'une architecture SOA. Le tableau suivant présente, les principales nouvelles tâches identifiées pour le rôle d'architecte.

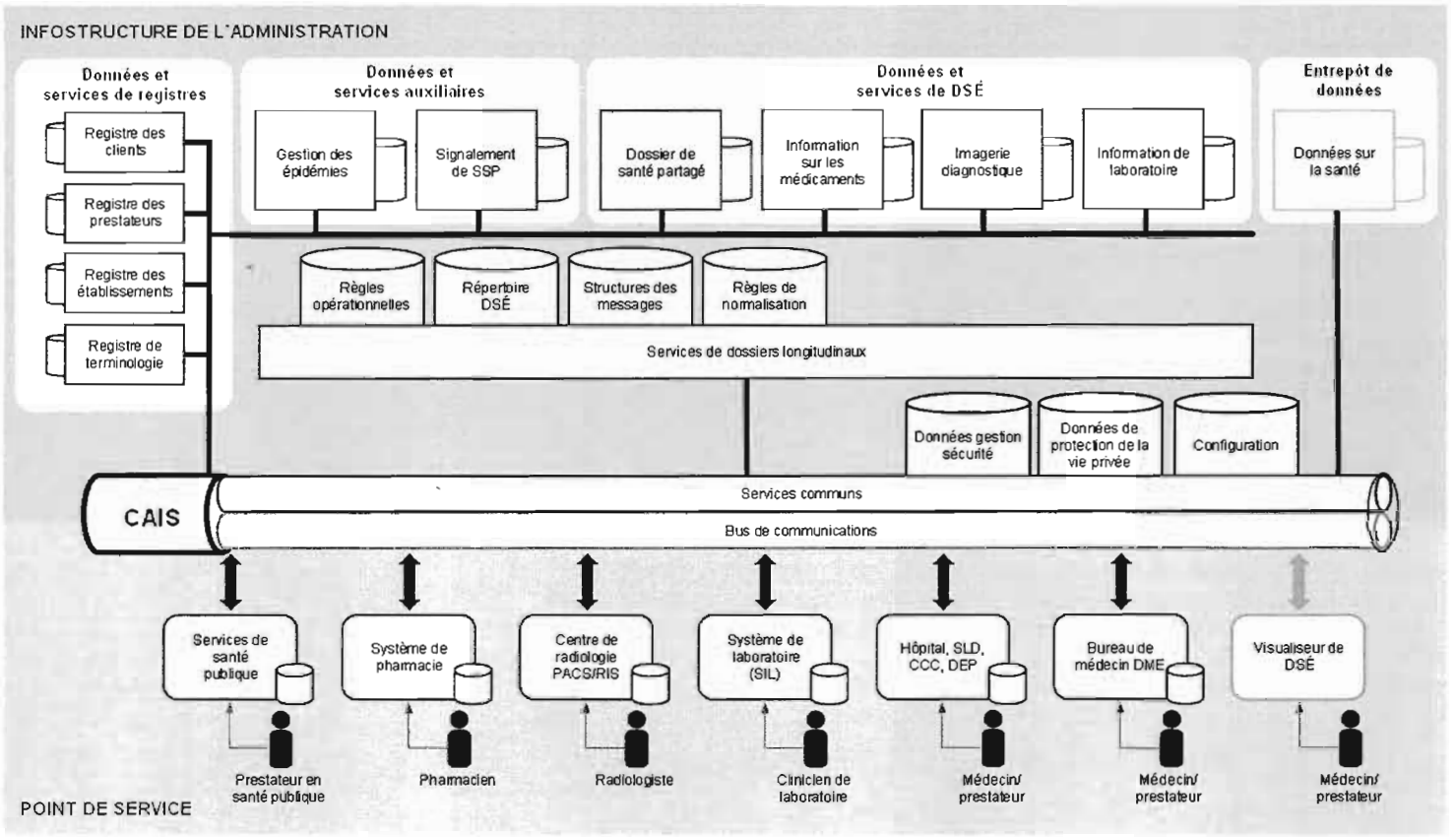
VII. Étant un architecte de solutions, quelles sont parmi les tâches suivantes celles qui sont sous votre responsabilité (R) ou auxquelles vous participez (P) dans un projet SOA ?

Nouvelles tâches	R	P
Identifier et classer les services		x
Déterminer les caractéristiques du service		x
Détailler le contrat de service		x
Rechercher les normes de l'industrie reliées au contrat de service		x
S'assurer que les services sont proprement construits, découverts, sécurisés et mesurés		x
Raffiner la classification des services		x
Raffiner les efforts de réalisation des services		x
Évaluer les efforts de réalisation des services SOA		x
Identifier les points d'intégration avec les systèmes d'information		x

VIII. Mis à part les tâches mentionnées dans le tableau précédant, y a-t-il d'autres nouvelles tâches dont vous êtes responsable ou auxquelles vous participez dans un projet SOA ?

Responsable de ...
-
-
-
-
-
-
Participe à ....
-
-
-
-
-
-

Annexe A : Infostructure de DSÉ - Architecture Conceptuelle[54]



[54] Source : Inforoute Santé du Canada http://www.infoway-inforoute.ca/fr/home/home.aspx

QUESTIONNAIRE Q4 (non rempli)

I. Combien d'années d'expérience possédez-vous avec les projets SOA[55] et quels postes avez-vous occupé durant ces projets ?

Années d'expérience	
Moins de 1 an	
1 à 2 ans	
2 à 4 ans	
4 à 6 ans	
Plus de 6 ans	

Postes occupés : ----------

II. Combien d'années d'expérience possédez-vous en informatique, dans d'autres projets, et quels postes avez-vous occupé durant ces projets ?

Années d'expérience	
1 à 5 ans	
5 à 10 ans	
10 à 15 ans	
Plus de 15 ans	

Postes occupés : ----------

---

[55] *Service-Oriented Architecture*

III. Selon la documentation publique[56] du Ministère de la Santé et des Services Sociaux du Québec, la mise en œuvre du Dossier de Santé du Québec (DSQ) implique plusieurs projets qui ont pour finalité de fournir les composantes technologiques de la solution de DSQ. Faites-vous partie de l'équipe du projet SRC-Labo[57] ? Si oui, quel est votre rôle au sein de cette équipe ?

IV. Sinon, de quelle façon participez-vous au DSQ ?

---

[56] « Le Dossier de Santé du Québec : Plan d'affaires 2007-2010 » http://www.dossierdesante.gouv.qc.ca/

[57] Le projet SRC-Labo (Services Régionaux de Conservation et Laboratoires) fait partie de l'ensemble des projets travaillant pour la réalisation du DSQ.

Selon une perspective de haut niveau, un système fondé sur une architecture SOA est constitué de :

1. **Services**, dont la mise en œuvre fournit la logique d'affaires et **registre de services**, où les services sont publiés
2. **Applications frontales**, qui découvrent et utilisent les services
3. **Infrastructure SOA**, qui connecte les applications aux services

V. Êtes-vous d'accord avec les énoncés du tableau suivant ?

Énoncé : Dans le cas du DSQ, et selon l'annexe A, ...	Oui	Non	Commentaires
1. Les applications de points de service (dans les systèmes de pharmacies, de laboratoires, d'hôpitaux, etc.) et les visualiseurs de DSÉ[58] sont considérés comme des **applications frontales**			
2. Les **services** sont distribués sur différentes parties de cette architecture et il existe 3 principaux **registres de services** (« données et services de registre », « données et services auxiliaires » et « données et services de DSÉ »)			
3. La couche d'accès à l'information sur la santé (CAIS) est responsable de la fonctionnalité de l'**infrastructure SOA**			

[58] Dossier de Santé Électronique

VI. Parmi les composants de l'architecture SOA[59] cités dans la question précédente, quels sont ceux dont la mise en œuvre entre dans le cadre du projet du DSQ auquel vous êtes affilié ? Afin de mieux les identifier, nous proposons pour chaque composant un ensemble de tâches spécifiques dont votre équipe de projet pourrait être responsable (R) ou auxquelles elle pourrait participer (P) :

	Principales tâches [60]	R	P	Commentaires
Infrastructure SOA	Sélectionner les standards et les produits qui seront mis en œuvre dans l'infrastructure SOA			
	Développer un ensemble de services d'infrastructure tels que la découverte, la communication et la sécurité			
	Identifier et développer des mécanismes de lien qui pourront satisfaire un grand ensemble d'utilisateurs potentiels de services			
	Documenter et maintenir l'infrastructure SOA			
Applications frontales	Identifier les services appropriés qui doivent être incorporés aux applications			
	Invoquer les services identifiés dans les applications (incluant la conversion des données, le traitement des erreurs, etc.)			
	Tester l'exactitude des services dans le contexte de l'application en cours de développement			
Services et registres	Définir les exigences des utilisateurs potentiels des services			
	Décrire et publier les services dans les registres de services			
	Développer le code d'initialisation et les procédures fonctionnelles des services			

[59] (1) Services et registres de services, (2) Applications frontales et (3) Infrastructure SOA
[60] Source : Kajko-Mattsson et al. (2007)

Selon la littérature SOA, et d'après les résultats d'une étude de cas que nous avons mené auprès d'une institution financière ayant entamé une initiative SOA, différents ensembles de rôles et de tâches associées à ces rôles sont nécessaires à la mise en œuvre des composants d'une architecture SOA. Le tableau suivant présente, les principales nouvelles tâches identifiées pour le rôle d'architecte.

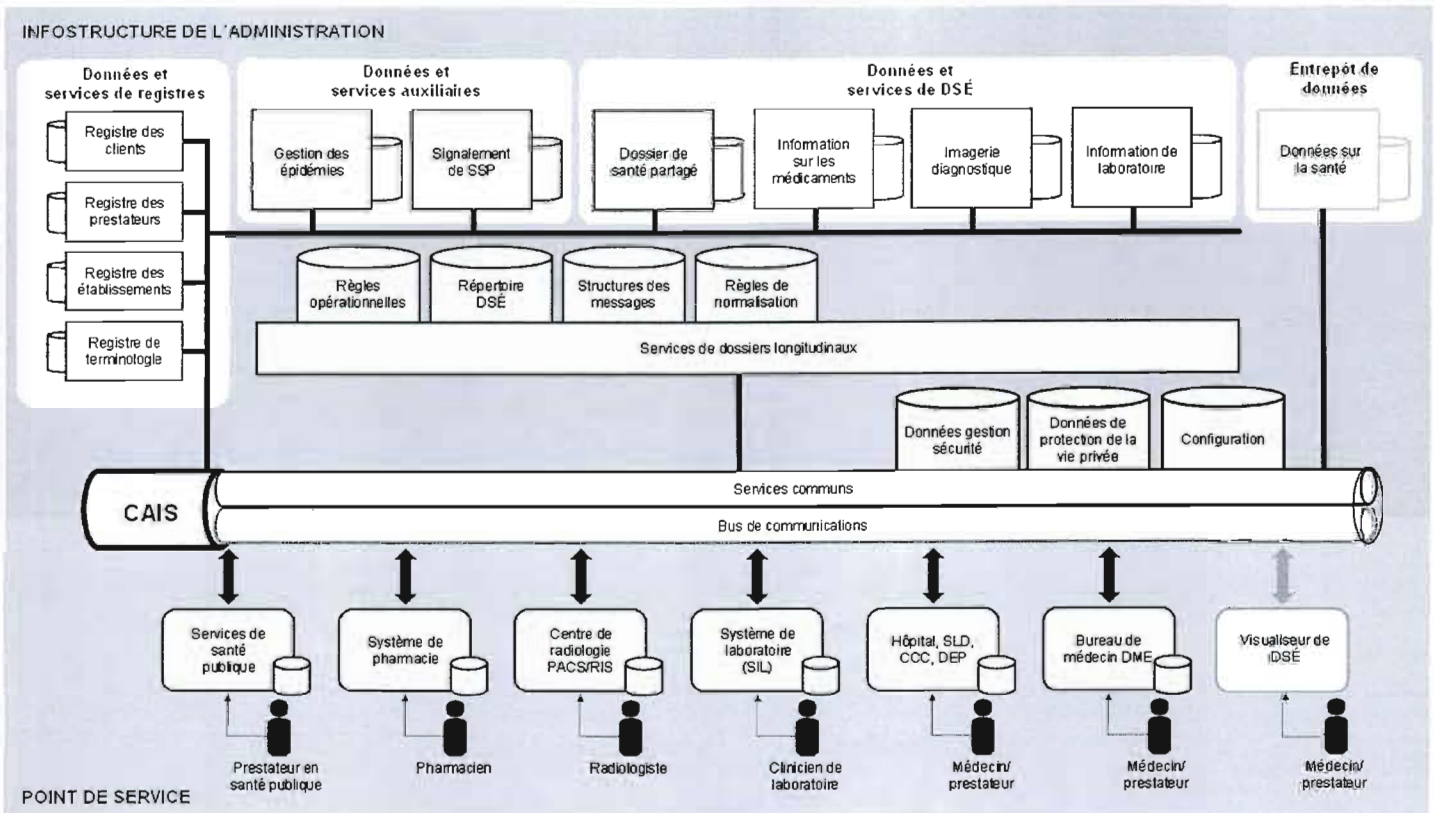
VII. Quelles sont parmi les tâches suivantes celles qui sont sous votre responsabilité (R) ou auxquelles vous participez (P) dans un projet SOA ? Ces tâches vous semblent-elles pertinentes ?

Nouvelles tâches pour le rôle d'architecte	R	P
Identifier et classer les services		
Détailler le contrat de service		
Rechercher les normes de l'industrie reliées au contrat de service		
S'assurer que les services sont proprement construits, découverts, sécurisés et mesurés		
Raffiner la classification des services		
Raffiner les efforts de réalisation des services		
Évaluer les efforts de réalisation des services		

VIII. Mis à part les tâches mentionnées dans le tableau précédant, y a-t-il d'autres nouvelles tâches dont vous êtes responsable ou auxquelles vous participez dans un projet SOA ?

Responsable de ...
- - - - - -
Participe à ....
- - - - - -

Annexe A : Infostructure de DSÉ - Architecture Conceptuelle[61]



[61] Source : Inforoute Santé du Canada http://www.infoway-inforoute.ca/fr/home/home.aspx

QUESTIONNAIRE Q5 (non rempli)

I. Combien d'années d'expérience possédez-vous avec les projets SOA[62] et quels postes avez-vous occupés durant ces projets ?

Années d'expérience	
Moins de 1 an	
1 à 2 ans	
2 à 4 ans	
4 à 6 ans	
Plus de 6 ans	

Postes occupés : ----------

II. Combien d'années d'expérience possédez-vous en informatique, dans d'autres projets, et quels postes avez-vous occupés durant ces projets ?

Années d'expérience	
1 à 5 ans	
5 à 10 ans	
10 à 15 ans	
Plus de 15 ans	

[62] *Service-Oriented Architecture*

Postes occupés : ----------

Selon la littérature SOA, et d'après les résultats d'une étude de cas que nous avons mené auprès d'une institution financière ayant entamé une initiative SOA, différents ensembles de rôles sont nécessaires au développement des composants d'une architecture SOA. L'annexe A présente les principaux nouveaux rôles identifiés et les tâches qui leurs sont allouées.

III. Parmi les rôles suivants, quels sont ceux qui sont présents dans le projet SRC-Labo ? Les tâches qui sont sous la responsabilité de chaque rôle (voir Annexe A) sont-elles pertinentes (P) ?

Rôles	Oui	Non	P	Commentaires
Analyste d'affaires SOA				
Architecte d'affaires SOA				
Architecte d'entreprise				
Comité SOA				
Directeur de projet SOA				
Administrateur de service				
Architecte SOA				
Analyste-Concepteur de service				

IV. Mis à part les rôles mentionnés dans l'annexe A, y a-t-il d'autres nouveaux rôles qui sont présents dans le projet SRC-Labo et qui ont des responsabilités majeures dans le déroulement du projet ? Si oui, qui sont-ils et quelles sont leurs tâches ?

Rôles	Tâches
	•
	•
	•
	•
	•
	•
	•

V. En ce qui concerne votre rôle de directeur de projet, estimez-vous que les nouvelles tâches que vous exécutez dans le cadre d'un projet SOA – en comparaison avec des projets non SOA – se limitent à celles citées dans le tableau de l'annexe B. Si non, quelles seraient les autres nouvelles tâches qui s'ajoutent à votre rôle, lors d'un projet SOA ?

VI. Quelle est la méthode de développement (ou la méthode à laquelle s'apparente la méthode de développement) employée pour le projet SRC-Labo ?

Méthode de développement	
Modèle en cascade	
Modèle en spirale	
Rapid Application Development (RAD)	
Unified Process (UP)	
Dynamic Systems Development Method (DSDM)	
Microsoft Solution Framework (MSF)	
Catalysis	
Feature Driven Development (FDD)	
Adaptive Software Development (ASD)	
Extreme Programming (XP)	
Crystal	
Scrum	
dX	
Autre :	

Commentaires : ----------

VII. S'agit-il de la même méthode de développement employée pour les autres projets du DSQ ? Sinon, pouvez-vous citer quelques une des autres méthodes de développement utilisées ?

Annexe A : Principaux nouveaux rôles pour le développement des composants d'une architecture SOA[63]

Rôles	Tâches
Analyste d'affaires SOA	• Détailler les besoins en interactions entre services • Effectuer la revue des contrats de services • Rédiger la spécification du service
Architecte d'affaires SOA	• Identifier et classer les services • Déterminer les caractéristiques du service • S'assurer que les services sont proprement construits, découverts, sécurisés, utilisés et mesurés
Architecte d'entreprise[64]	• Approuver la classification des services • Raffiner la classification des services • Approuver les entités d'affaires spécifiées au contrat de service, afin d'assurer qu'elles soient cohérentes avec l'architecture d'entreprise
Comité SOA	• Responsabilité globale de l'atteinte des objectifs SOA • S'assurer que les différents processus (de gouvernance, de gestion de la qualité des services, etc.) et responsabilités sont performants, claires et adéquatement assumés • Valider l'applicabilité SOA aux projets • Approuver les contrats de service
Directeur de projet SOA[65]	• Approuver les livrables du projet (ex. contrats de service) • Faire ajuster un contrat de service non approuvé • Inclure les jalons SOA au plan de projet
Administrateur de service	• Intégrer les services au registre de services • Gérer le registre des services
Architecte SOA	• Détailler le contrat de service • Rechercher les normes de l'industrie reliées au contrat de service • Raffiner la classification des services • Raffiner les efforts de réalisation des services • Évaluer les efforts de réalisation des services SOA
Analyste-Concepteur de service	• Concevoir le service • Identifier les éléments de conception • Produire la description WSDL[66] du service

[63] Le tableau présente une combinaison des informations recueillies suite à une analyse de la littérature SOA et d'une étude de cas menée auprès d'une institution financière ayant entamé une initiative SOA.

[64] Même si pour certaines entreprises, il ne s'agit pas vraiment d'un nouveau rôle – introduit lorsque cette dernière entame une initiative SOA – nous le considérons ainsi. En effet, actuellement, l'architecture d'entreprise est considérée comme une discipline émergente.

[65] Ici aussi, le chef de projet ne constitue pas un nouveau rôle. Seulement, nous le considérons ainsi car, dans le contexte d'un projet SOA, ce rôle clé acquiert de nouvelles responsabilités.

[66] *Web Service Description Language*

APPENDICE C

ÉVOLUTION DU PROJET DE RECHERCHE

En raison d'évènements qui sont survenus, notre projet de recherche a subi plusieurs changements. Dans le tableau ci-dessous, nous présentons l'objectif, les questions de recherche, les raisons du changement et le travail effectué sur chaque version.

Tableau A.1 Évolution du travail de recherche

Vers.	Date[67]	Objectif	Questions de recherche (QP[68] et QS[69])	Raisons du changement[70]	Travail accompli par la suite
V1	02/15/07	Identifier les changements organisationnels qui caractérisent la transition de l'entreprise vers une architecture axée sur les services	**QP :** Quels sont les changements organisationnels qui caractérisent la transition d'une entreprise vers une architecture axée sur les services ?	- Objectif de recherche encore mal cerné - Question de recherche trop vague	- Clarification du paradigme SOA à travers des lectures, notamment concernant sa relation avec l'architecture d'entreprise - Intérêt porté sur l'impact de SOA sur les méthodes de gestion de projet - Redéfinition de l'objectif et des questions de recherche
V2	03/08/07	Identifier les meilleurs pratiques pour la gestion de projets de développement dans un environnement axé sur les services	**QP :** Quels sont les meilleures pratiques de la gestion de projets de développement dans un environnement axé sur les services **QS1 :** Quels sont les rôles, les responsabilités, les compétences, les procédures et les livrables nécessaires au développement axé sur les services ? **QS2 :** Quels sont les principaux artefacts impliqués dans la planification, la conception, le développement et la gestion de solutions axées sur les services dans une entreprise ?		- Aperçu général sur les méthodes de développement logiciel - Définition de la méthode de recherche
V2.1	03/29/07	Pas de changement	Pas de changement	Il s'est avéré utile de s'intéresser, en particulier, aux méthodes de développement agiles car ces dernières semblent être préconisées pour le développement dans les environnements axés sur les services	- Intérêt pour SOSE - Intérêt pour les méthodes agiles (approfondissement de UP et RUP) - Approfondissement des pratiques et des défis du développement dans les environnements axés sur les services
V2.2	04/23/07	Pas de changement	Pas de changement	- L'approfondissement de la revue de la littérature a permis de raffiner l'objectif et	- Redéfinition de l'objectif et de la question de recherche principale

[67] Date de finalisation de la version
[68] Question principale
[69] Question secondaire
[70] Raisons qui ont conduit à la modification de l'objectif et/ou des questions de recherche

Vers.	Date[67]	Objectif	Questions de recherche (QP[68] et QS[69])	Raisons du changement[70]	Travail accompli par la suite
				les questions de recherche - Intervention de la part de Mr I. Maffezzini pour une définition plus détaillée des étapes de la méthode de recherche	- La méthode de recherche a été détaillée
V3	04/29/07	Établir une ébauche de méthode appliquée aux projets de développement dans un environnement axé sur les services. L'ébauche de méthode consiste en la définition des principaux artefacts, activités, rôles et outils qui caractérisent les projets de développement dans un environnement axé sur les services.	**QP :** Quelles sont les principales caractéristiques d'une méthode de développement logiciel adaptée aux environnements axés sur les services ? **QS1 :** Quels sont les rôles, les responsabilités, les compétences, les procédures, les méthodes, les outils et les livrables nécessaires au développement axé sur les services ? **QS2 :** Quels sont les principaux artefacts impliqués dans la planification, la conception, le développement et la gestion de solutions axées sur les services dans une entreprise ?	Intervention de Mr. E. Elia pour l'ajout d'une étude de cas exploratoire à la méthode de recherche et pour détailler le processus de validation de la méthode. Argument : Il est mieux de combiner les connaissances tirées de la littérature et d'un cas observé pour proposer une ébauche de méthode de développement destinée aux environnements axés sur les services.	- Introduction d'une étude de cas exploratoire dans la méthode de recherche - Processus de validation de la méthode détaillé
V3.1	05/11/07	Pas de changement	Pas de changement		Accomplissement des étapes 1 et 3 de la méthode de recherche : - Faire une analyse approfondie des principaux articles et livres publiés durant les cinq dernières années sur SOA. - Synthétiser les spécificités et les caractéristiques de SOA ayant un impact sur la gestion de projet et sur les méthodes.
			Période d'inactivité (3 mois)		
V3.2	09/04/07	Pas de changement	Pas de changement		- Définition des activités et des rôles de la méthode de développement destinée aux environnements axés sur les services
V3.3	10/23/07	Pas de changement	Pas de changement	- L'élaboration de l'ébauche de méthode a donné un modèle surchargé (grand nombre d'activités, de tâches et de rôles impliqués) - La validation du modèle par les répondants, lors de l'étude de cas, ne serait pas facile	- Le problème a été abordé autrement. On a essayé de ne pas perdre tout le travail synthétisé de la littérature, tout en essayant de rendre l'étude de cas la plus intéressante possible. - Modification aux questions de recherche

Vers.	Date[67]	Objectif	Questions de recherche (QP[68] et QS[69])	Raisons du changement[70]	Travail accompli par la suite
				- Soumettre le modèle aux répondants les aurait sûrement influencé dans leurs réponses	
V4	11/14/07	Savoir si l'adoption des principes de l'axé sur le service par une organisation, dans le but de créer une SOA, exerce réellement un impact sur le CVP	**QP :** La présence d'un environnement axé sur les services exerce-t-elle un impact sur le CVP ? **QS1 :** Le CVP admet-il de nouvelles phases ? **QS2 :** Les phases du CVP admettent-elles de nouvelles activités ? **QS3 :** Existe-t-il de nouveaux rôles qui sont impliqués dans les phases du CVP ?	- Question de recherche principale relativement dangereuse selon Mr. Maffezzini et impliquerait un certain nombre de cas pour notre étude	- Réorientation des questions de recherche - Élaboration d'une ébauche pour les questionnaires destinés aux entretiens de l'étude de cas - Recherche du cas d'étude unique
V5	01/10/07	Définir les changements qui sont apportés au CVP dans un environnement axé sur les services, en termes d'activités et de rôles	**QP :** Quels sont les changements encourus au CVP, en présence d'un environnement axé sur les services ? **QS1 :** Quels sont les changements en termes d'activités qui sont apportés au CVP ? **QS2 :** Quels sont les changements en termes de rôles qui sont apportés au CVP ?		- Cas d'étude sélectionné : Institution financière ayant entamé ne initiative SOA - Élaboration du questionnaire 1 pour le 1er cas d'étude et accomplissement de l'entretien avec R1
V5.1	01/17/08	Pas de changement	Pas de changement	Les résultats du premier entretien et les échange d'information avec le répondant, nous ont permis de recueillir les données dont nous avions besoin. Après l'analyse des données, nous nous sommes aperçus que : 1. Les activités sont en fait une succession de tâches bien définies, accomplies par des rôles. Identifier les rôles et les tâches qui leurs sont allouées revient à définir les activités impliquées dans le CVP. 2. La SOA étudiée est complètement centralisée. L'organisation est en même temps le développeur pour les trois composants de l'infrastructure SOA. Un deuxième cas d'étude, comportant une SOA décentralisée, s'est présenté à nous	- Changement au niveau de l'étude de cas. Nous sommes passés d'une étude de cas exploratoire comportant un cas unique vers une étude de cas exploratoire comportant deux cas : le 1er étant caractérisé par une SOA centralisée et le 2ème par une SOA décentralisée. - Changement au niveau du type de données à collecter pour le deuxième cas. Focalisation sur les rôles et les tâches associées aux rôles. - Ajustement de l'objectif et des questions de recherche suite à l'introduction d'un deuxième cas dans notre étude - Ajout d'une QS concernant les rôles et les tâches SOA de la littérature
V6	02/12/08	Savoir si le passage vers un environnement axé sur les services pour une organisation	**QP :** Est-ce que le passage vers un environnement axé sur les services pour		- Préparation du questionnaire 1 pour le 2ème cas d'étude et déroulement de

Vers.	Date[67]	Objectif	Questions de recherche (QP[68] et QS[69])	Raisons du changement[70]	Travail accompli par la suite
		demande des changements aux rôles et aux tâches du CVP	une organisation demande des changements aux rôles et aux tâches du CVP ? **QS1** : Dans la littérature SOA, existe-t-il une conceptualisation cohérente des rôles impliqués dans le CVP et des tâches qui leurs sont associées ? **QS2** : Quels sont les changements aux rôles ? **QS3** : Quels sont les changements aux tâches associées aux rôles ?		l'entretien avec R2 - Préparation des questionnaires 2 et 3pour le 2ème cas d'étude et attente pour les entretiens avec R3 et R4
V7	05/03/08	Pas de changement	Pas de changement		- Déroulement de l'entretien avec R3 - Interruption du 2ème cas d'étude en raison de l'indisponibilité de R3 pour l'entretien - Finalisation du mémoire (synthèse des connaissances tirées des cas d'étude, discussion, conclusion, etc.)