ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE, CONCENTRATION PERSONNALISÉE
M.Ing.

PAR
Jean-François COUTURIER

ÉLABORATION ET EXPÉRIMENTATION D'UNE MÉTHODOLOGIE AGILE PERMETTANT LA MIGRATION VERS UNE ARCHITECTURE ORIENTÉE SERVICES EN PME À L'AIDE D'OPENUP

MONTRÉAL, LE 7 OCTOBRE 2011

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. François Coallier, directeur de mémoire
Département de génie logiciel et des T.I à l'École de technologie supérieure

Roger Champagne, président du jury
Département de génie logiciel et des T.I à l'École de technologie supérieure

Ghizlane El Boussaidi, membre du jury
Département de génie logiciel et des T.I à l'École de technologie supérieure

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

LE 7 SEPTEMBRE 2011

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

J’aimerais remercier mon directeur, le professeur François Coallier. Merci pour ses judicieux conseils et sa patience. Ma famille pour la patience et le support qu'ils m’ont témoigné pendant ces nombreuses années d’études. Le département de génie logiciel et des TI de l'ÉTS pour leur soutien. Daniel Hallé pour son aide et son soutien dans la réalisation des projets pilotes et la Corporation d’Hébergement du Québec pour m’avoir offert un environnement pour expérimenter mes travaux.

ÉLABORATION ET EXPÉRIMENTATION D'UNE MÉTHODOLOGIE AGILE PERMETTANT LA MIGRATION VERS UNE ARCHITECTURE ORIENTÉE SERVICES EN PME À L'AIDE D'OPENUP

Jean-François COUTURIER

RÉSUMÉ

Une architecture orientée services (*Services-Oriented Architecture* ou SOA) se base fréquemment sur les services Web. Souvent, les progiciels et les systèmes patrimoniaux ne supportent pas nativement les services Web. Une organisation aura la plupart du temps un historique technologique avec lequel elle devra composer. Pour ces raisons, l'implantation d'une SOA impliquera généralement la réingénierie de plusieurs composantes logicielles. Elle impliquera également une connaissance des processus et de l'architecture de l'entreprise.

Les plus petites organisations n'ont pas toujours une documentation très élaborée des processus d'affaire de l'entreprise. Ce sont des organisations qui réagissent rapidement au changement. Afin de les encourager à utiliser une approche plus rigoureuse, il faut leur offrir des outils efficaces et légers.

L'objectif de cette maîtrise est de concevoir une méthode légère et ouverte qui permettra aux PME d'implanter une architecture orientée services de manière itérative et incrémentale à l'aide des services Web et d'un cadre de développement agile. Cette méthode inclut des activités au niveau de l'entreprise afin d'aligner les projets SOA avec les besoins d'affaires et l'environnement technologique de l'entreprise. La stratégie utilisée pour réaliser cette méthode est d'identifier un ensemble de méthodes existantes qui abordent ces différents aspects, de les intégrer en une seule méthode documentée et de l'expérimenter.

Dans le cadre de ce travail, un partenaire industriel a été sollicité afin de tester et valider la méthode. Des projets spécifiques ont été sélectionnés afin de faire la réingénierie de certaines composantes logicielles dans le but de migrer graduellement vers une SOA. Ces tests ont permis d'identifier des artéfacts particulièrement importants dans la méthode.

De ces travaux découlent une nouvelle méthode ouverte intégrant les activités d'architecture d'entreprise, les activités orientées services et les activités de développement logiciel classique. Bien que plusieurs améliorations soient toujours possibles, la méthode élaborée dans le cadre de ces travaux est intégrée et documentée, que ce soit directement ou dans la littérature des méthodologies originales.

Mots-clés : architectures orientées services, méthodologies agiles, services web, architecture d'entreprise

ÉLABORATION ET EXPÉRIMENTATION D'UNE MÉTHODOLOGIE AGILE PERMETTANT LA MIGRATION VERS UNE ARCHITECTURE ORIENTÉE SERVICES EN PME À L'AIDE D'OPENUP

Jean-François COUTURIER

ABSTRACT

Service-Oriented Architecture (SOA) is frequently based on Web services. Often, business software and legacy systems do not natively support Web services. An organization typically has a technological history with which it must deal. For these reasons, the implementation of an SOA will generally involve re-engineering of several software components. It will also involve knowledge of processes and enterprise architecture.

Smaller organizations may not have very elaborate documentation of their business processes. These are organizations that respond quickly to change. They often have neither the time nor the interest to produce this documentation.

The objective of this thesis is to design a lightweight and open methodology which will enable SMEs to improve their competitiveness and their capacity for change by iteratively and incrementally implementing an SOA using Web services and an Agile development framework. This methodology includes activities at the enterprise level to align SOA with business needs and the technological environment of the company. The strategy used to achieve this method is to identify a set of existing methods that address these different aspects, to integrate them into one documented method and test it.

As part of this work, an industrial partner is sought to test and validate the methodology. Specific projects have been selected to re-engineer software components in order to gradually migrate to an SOA. We also derive a set of recommendations on the methodology used to implement an SOA. These tests have identified particularly important artefacts in the method.

The result of this work is a new open method that integrates the activities of enterprise architecture, service-oriented activities and the activities of traditional software development. Although many improvements are always possible, the method developed as part of this work is integrated and documented, either directly or in the original literature.

Keywords : Service-oriented architecture, agile methodologies, web services, enterprise architecture.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ADM	Architecture Development Method
B2B	Business-to-Business
B2C	Business-to-Customer
BPEL	Business Process Execution Language
CHQ	Corporation d'Hébergement du Québec
EPF	Eclipse Process Framework
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertexte Transfer Protocol
MSSS	Ministère de la Santé et des Services sociaux
OASIS	Organization for the Advancement of Structured Information Standards
ODBC	Open DataBase Connectivity
PME	Petite Moyenne Entreprise
REST	Representational State Transfer
RPC	Remote Procedure Call
RUP	Rational Unified Process
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
TOGAF	The Open Group Architecture Framework
UDDI	Universal Description Discovery and Integration
XML	Extensible Markup Language

XSD	XML Schema Definition
W3C	World Wide Web Consortium
WCF	Windows Communication Foundation
WSDL	Web Services Definition Language
WSE	Web Service Enhancement
WS-CDL	Web Services Choreography Description Language
WS-I	Web Services Interoperability

INTRODUCTION

L'architecture orientée services (*Service Oriented Architecture* ou SOA) est un paradigme architectural récent. Peu de personnes dans l'industrie des TI en comprennent réellement tous les aspects (Erl, 2005, p. 2). Malgré cela, de plus en plus d'entreprises effectuent le saut vers une SOA, motivées par les nombreux avantages annoncés.

L'économie québécoise repose essentiellement sur les PME, c'est-à-dire les entreprises de moins de 500 employés. 99,4% des entreprises québécoises comptent moins de 500 employés. 90% ont moins de 20 employés (Paradis, 2004). Au niveau canadien, ce sont 99,7% des entreprises qui se situent dans la catégorie des PME (Dellazizzo, 2006). C'est un immense bassin d'entreprises qui peuvent bénéficier des avantages stratégiques d'une SOA dans leurs architectures d'affaires et technologique.

Les PME sont peut-être moins sensibles aux problèmes d'intégration des systèmes puisque ceux-ci sont moins nombreux. Par contre, le besoin d'intégrer leurs processus avec des fournisseurs ou des clients externes prend de l'ampleur. Les PME ont moins de moyens et moins de ressources disponibles, ils doivent faire plus avec moins. Les PME exploitent leurs matériels informatiques, où résident leurs systèmes patrimoniaux, plus longtemps. (Bloomberg, 2006).

Il existe plusieurs méthodologies décrivant les étapes nécessaires à la réalisation d'un service ou d'un ensemble de services. La méthodologie de (Erl, 2005) présente les phases d'analyse et de conception du cycle de développement d'un service. Mais l'auteur ne fournit pas beaucoup de détail sur la réalisation de ces activités à l'intérieur d'un cycle de développement. Il y a également les méthodologies SOAF (Erradi, Anand et Kulkarni, 2006) et SOMA (Arsanjani, 2004) qui abordent ces mêmes thèmes, sans jamais offrir une solution intégrée avec une vue de l'entreprise. Ces méthodologies sont soient propriétaires, soient non

intégrés dans un outil de modélisation des processus. Elles abordent toutes l'analyse et la conception d'un service. Cependant, la perspective de ces méthodologies reste souvent au niveau d'un projet. Il est difficile d'associer ces activités de projet à des activités plus globales, à des activités d'entreprise.

Pour répondre à ce besoin, il existe des cadriciels d'entreprise. Pour l'architecture d'entreprise, TOGAF (TOGAF, 2009) est un cadriciel ouvert et reconnu pour l'architecture d'entreprise. Le cadre de Zachman (Zachman, 1987) est également très populaire et a été à l'origine de plusieurs déclinaisons. Cependant, ce sont tous des cadriciels assez complexes, qui ne sont pas appropriés aux plus petites organisations. Le processus unifié d'entreprise de (Scott W.Ambler, 2005) est moins connu. Mais sa similitude avec certaines méthodologies de développement est intéressante. Ces cadres abordent typiquement le développement logiciel à très haut niveau.

Problématique

Il n'existe pas à notre connaissance de méthode ouverte et légère permettant d'assister les PME dans leurs efforts pour migrer vers une SOA qui combine les activités d'architecture d'entreprise et le développement des systèmes dans le contexte particulier d'une SOA. Si les objectifs d'une SOA sont d'augmenter la réutilisabilité des composantes ou encore d'automatiser la réalisation de certains processus d'affaires, il est nécessaire d'avoir une vue globale arrimée avec le cycle de développement logiciel.

Il existe plusieurs raisons de développer une méthode facilitant la réingénierie de l'architecture applicative d'une PME à l'aide de méthodes ouvertes et légères afin de migrer vers une SOA. Les PME possèdent plusieurs avantages face aux grandes organisations. Généralement, la communication entre les personnes d'une petite organisation se fait plus rapidement. Les employés d'une même organisation sont géographiquement très proches.

Pour ces mêmes raisons, la gouvernance des systèmes et des données est souvent plus facile à établir (Meehan, 2007).

Objectifs

L'objectif de ce projet de maîtrise est de développer, de documenter et de tester une méthode qui va intégrer un cycle de vie constitué d'activités d'entreprise offrant une perspective plus globale et un cycle de développement logiciel incluant les phases d'analyse et de conception d'une SOA. Ces deux cycles seront ensuite arrimés pour assurer une meilleure cohérence.

Le livrable principal est donc une méthode intégrée, modélisée et documentée dans un outil de composition des processus. Cette méthode intègrera des activités d'architecture, de documentation et de modélisation de l'entreprise afin d'aligner les projets SOA avec les besoins d'affaires de l'organisation. Cette méthode a été testée dans une PME, de la phase d'analyse à la phase d'élaboration. Elle inclut également une itération complète des activités d'entreprise.

Hypothèses

Dans le cadre de ce mémoire, on pose certaines hypothèses. On présume que les plus petites organisations n'ont pas une documentation très élaborée des processus d'affaire de l'entreprise. Ce sont des organisations qui réagissent rapidement au changement. Elles ont cependant un plus grand contrôle sur leur environnement technologique.

Le défi est d'offrir à ces organisations une méthodologie légère et agile qui permettra d'accroître leur compétitivité et leur capacité de changement en implantant une architecture orientée services. Cette agilité doit se retrouver tant au niveau de la modélisation, de l'architecture, qu'au niveau de l'implémentation.

Une SOA apporte des avantages compétitifs significatifs. Ces avantages peuvent justifier la mise en place d'une SOA en PME, de manière itérative et incrémentale. Dans le cadre de cette expérience, nous émettons les hypothèses suivantes :

- On peut utiliser les méthodes agiles pour faciliter la mise en place d'une architecture SOA dans les PME.

- Les PME ont des avantages liés à leur petite taille. À l'aide d'une approche agile, il est facile de réaliser certaines activités d'architecture d'entreprise et de mettre en place une SOA adaptée aux PME.

Trois grandes catégories d'utilisateurs sont susceptibles d'être touchées par les résultats de ces travaux. Dans une PME, il est possible que ces trois rôles soient réalisés par les mêmes individus.

Les architectes sont concernés, car la réingénierie des systèmes va impacter l'architecture technologique de l'entreprise. Les architectes identifieront les systèmes susceptibles d'être portés vers une SOA et feront la promotion de cette migration. Ce sont eux qui assureront la cohérence de l'architecture applicative à travers les différents travaux de réingénierie.

Les analystes d'affaires sont sollicités dans la compréhension et la modélisation des processus d'affaires et dans leur participation à un projet de réingénierie SOA. Ils sont particulièrement interpellés par les travaux précédents le développement de la solution; la définition des besoins d'affaires et la décomposition des processus d'affaires afin d'y intégrer les services.

Les développeurs seront intéressés par la méthodologie de développement inspirée d'OpenUP et son utilisation dans le cadre d'une implantation SOA. OpenUP est un cadre de développement agile et la documentation disponible offre aux développeurs un ensemble de

guides sur lesquels ils peuvent se référer. L'arrimage avec les activités et les artéfacts d'entreprises leur fourniront des outils pour débuter ou orienter leurs travaux.

Approche

La Figure 1 illustre la démarche utilisée dans ce mémoire. La revue de littérature nous permet de confirmer l'absence de méthodologie intégrant des activités d'architecture d'entreprise et d'architecture orientées services dans un cycle de développement agile. On identifie trois méthodologies candidates susceptibles d'êtres intégrées et documentées.

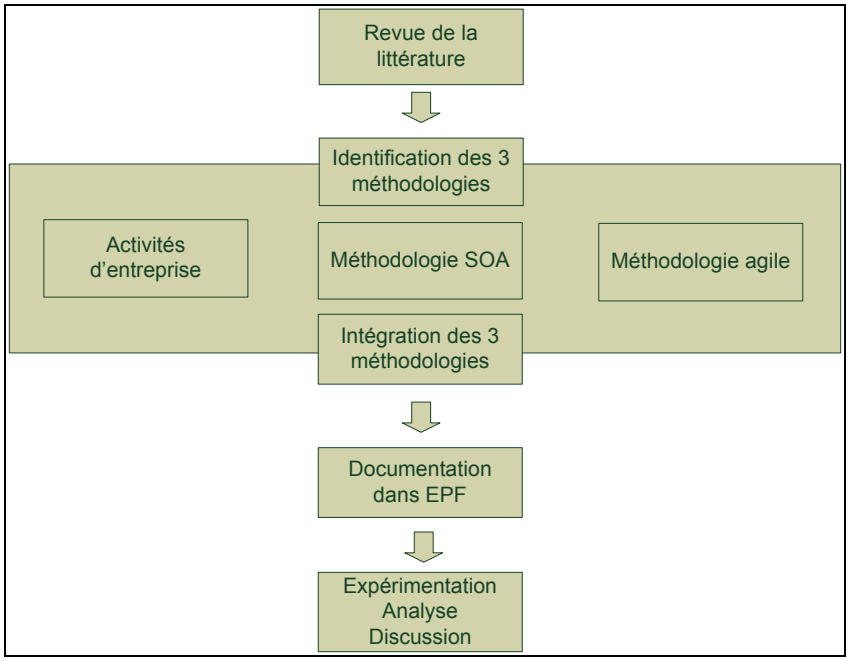


Figure 1 Démarche du mémoire

Finalement, la méthode est testée sur des petits projets avec l'aide d'un partenaire industriel. Une approche itérative et incrémentale est utilisée. On effectuera la réingénierie des systèmes d'informations existants pour les transposer graduellement dans une SOA selon les besoins d'affaires du partenaire impliqué. Les artéfacts réalisés dans le cycle continu d'entreprise alimenteront le cycle de développement logiciel, assurant ainsi une meilleure cohérence avec

l'architecture de l'entreprise. Les artéfacts générés pendant le cycle de développement logiciel permettront d'ajouter ou de mettre à jour les artéfacts d'entreprise.

Partenaires industriels

Ce projet de recherche inclut un partenariat industriel avec la Corporation d'Hébergement du Québec (CHQ). La CHQ est une société d'État provincial existant depuis 1999. La mission de la CHQ est d'offrir des services en génie-conseil aux établissements du réseau de la santé.

Les projets de construction touchant les hôpitaux doivent respecter des normes très spécifiques et ce sont les spécialistes de la CHQ (architectes, ingénieurs) qui vérifient la conformité des travaux. La CHQ assure aussi le financement à court et long terme des travaux, au nom du ministère de la Santé et des Services sociaux du Québec (MSSS). Finalement, la CHQ offre un service d'expertise en gestion et suivi de contrat de construction. La CHQ emploie approximativement 130 employés. Elle s'est dotée d'une équipe en TI de cinq personnes (Corporation d'hébergement du Québec, 2010).

Les analystes programmeurs de la CHQ sont expérimentés. Ils possèdent plusieurs années d'expérience en programmation, mais également en analyse et en gestion des bases de données. Ils connaissent bien la plateforme et les outils de développement de Microsoft. Puisque c'est une petite équipe, les processus liés au développement sont légers. Leur imputabilité couvre toutes les étapes du développement dans lesquelles ils sont impliqués.

La CHQ possède plusieurs systèmes d'information qui ont été développés ou acquis au cours des dernières années. Ces systèmes communiquent entre eux de différentes façons. Dans certains cas, ce sont des liens ODBC. Autrement, ce sont des modules programmés et intégrés directement dans les applications. Certains systèmes propriétaires majeurs, comme le système de comptabilité, ne sont pas intégrés aux autres systèmes. Il n'existe pas de connectique permettant cette interopérabilité.

La CHQ a orienté ses choix technologiques vers les produits Microsoft au cours des dernières années, que ce soit pour le développement ou les systèmes d'exploitation des postes de travail et des serveurs ainsi que des bases de données. C'est un environnement relativement homogène. Cependant, certains systèmes patrimoniaux importants, comme le système de comptabilité, reposent sur des technologies propriétaires différentes.

Elle impartit certains services, comme l'hébergement du site Web ainsi que l'hébergement d'un portail extranet. L'infrastructure réseau est administrée par le MSSS alors que la CHQ gère ses serveurs applicatifs et ses serveurs de bases de données.

La CHQ n’applique pas un processus spécifique pour son développement. L'équipe des TI a élaboré au fil du temps une approche qui s’apparente à un développement itératif et incrémental.

Les propositions de projets sont soumises au coordonnateur des TI. Le coordonnateur assigne un analyste au projet. Les exigences logicielles sont relevées par l’analyste. Ensuite, un prototype non fonctionnel est réalisé afin de reproduire le comportement attendu. Le client valide les exigences à l'aide du prototype. Une fois les exigences raffinées, le processus métier est programmé. Lorsque la fonctionnalité est approuvée, un nouveau prototype est construit pour la fonctionnalité suivante, jusqu'à la réalisation complète du projet. Les tests sont effectués manuellement tout au long du développement par les clients et les programmeurs. Les tests unitaires ne sont pas utilisés à la CHQ.

Le suivi des demandes de modifications se fait à l'aide d'un outil développé par le MSSS. Les utilisateurs peuvent y accéder et saisir leurs demandes de changements. C'est l'analyste et le client qui gère les priorités de ces demandes. Le coordonnateur n'intervient qu'en dernier recours.

Présentation de la structure du mémoire

Le chapitre 1 est une recherche bibliographique ayant comme objectifs de confirmer l'hypothèse selon laquelle il n'existe aucune méthodologie ouverte et légère intégrant les méthodologies d'architecture d'entreprise, les méthodologies de développement et les méthodologies SOA. Le chapitre 1 donne également les définitions préalables et décrit l'état de l'art sur les différentes méthodologies. On y explique notamment ce qu'est une SOA et ce que sont les services web. À la fin de ce chapitre, on doit être en mesure d'identifier les trois méthodologies candidates.

Le chapitre 2 présente la méthodologie utilisée pour ce travail de maîtrise. On y explique la démarche pour réaliser la nouvelle méthode ainsi que pour valider les hypothèses.

Le chapitre 3 explique en détail la réalisation de la méthode. On y explique les raisons justifiant le choix des trois méthodologies candidates. On explique les composantes de la méthodologie, tant pour les activités d'entreprises que pour les activités de développement. Vient ensuite l'expérimentation de la méthodologie par la réalisation de deux études de cas dans le chapitre 4. Quelques itérations pour chaque étude de cas sont décrites.

Le chapitre 5 contient l'interprétation des résultats et une discussion. On identifie les différentes variables qui ont influencé l'évaluation de la méthode et on donne une appréciation de la méthodologie, puis on en expose les limites. La conclusion trace un bilan de ce mémoire et propose plusieurs ouvertures sur des travaux subséquents.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Introduction

Il existe plusieurs concepts, protocoles, standards et organismes concernant l'architecture SOA ainsi que sur les technologies qui implémentent ce concept. On définit dans ce chapitre ce qu'est une SOA et comment l'architecture d'entreprise joue un rôle dans son implantation. Puis sont décrits les organismes de normalisation qui œuvrent autour de SOA et des technologies qui permettent son implémentation. Sont ensuite définis les concepts permettant la réalisation des services Web qui contribueront à la réalisation d'une SOA.

1.2 L'architecture orientée services

L'architecture orientée services (SOA) est un paradigme architectural relativement récent. Cette architecture se base sur les avancées de la conception orientée objet pour créer une couche d'abstraction supplémentaire entre les besoins d'affaires et le code applicatif. SOA offre une architecture décentralisée où des clients peuvent rechercher et consulter des services pour ensuite les consommer. Ces services sont programmés dans un langage de programmation et utilisent un protocole normalisé pour communiquer entre eux, habituellement par l'intermédiaire d'un intergiciel.

Pour mettre en place une SOA, les processus d'affaires doivent être préalablement bien modélisés. À partir de ce modèle et des applications qui sont en place, on identifie et on crée des services. Puis on les publie pour qu'ils soient consommés par des applications conçues selon le paradigme SOA. Ces services exécutent une tâche bien précise. C'est pourquoi on dit qu'un service doit être faiblement couplé (avoir le moins de dépendances possible avec d'autres services) et avoir une cohésion très forte (exécuter une action précise).

La Figure 1.1 illustre que le SOA a atteint une certaine maturité puisqu'elle se retrouve dans la phase de « grâce » (*Slope of Enlightenment*) qui correspond à l'industrialisation du concept. On doit ainsi s'attendre à une adoption globale de cette architecture logicielle d'ici deux à cinq ans.

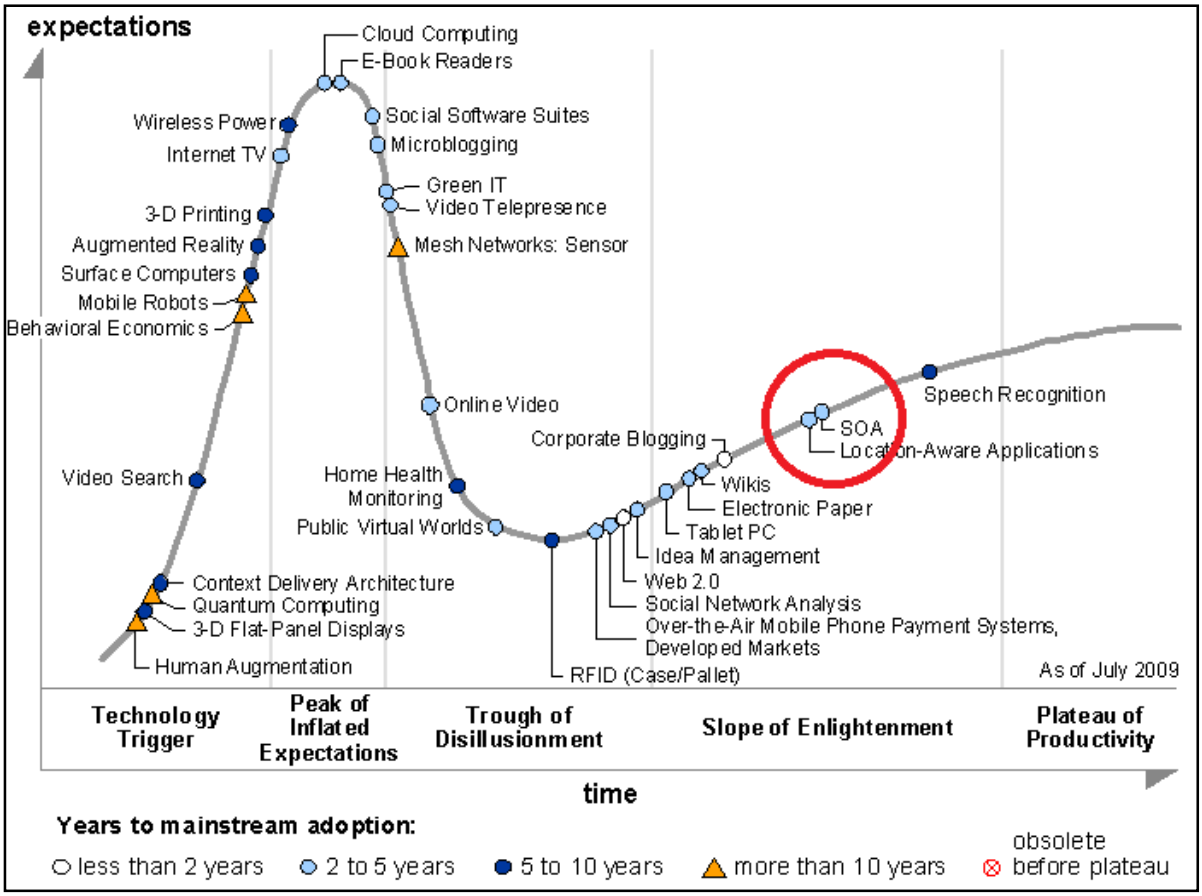


Figure 1.1 Le « hype cycle » de SOA.
Tirée de (Gartner, 2009)

Voici quelques avantages d'une SOA selon (Natis, 2003, p. 5) :

- Développement incrémental et déploiement d'application d'affaires;
- Réutilisation de composantes d'affaires;
- Coût réduit dans la création de nouveau processus d'affaires;
- Clarté dans l'architecture applicative de l'entreprise.

Les avantages d'une SOA selon (Erl, 2004, p. 461) :

- Augmente l'interopérabilité entre les systèmes patrimoniaux. Cela permet de réévaluer les nombreuses procédures d'affaires qui reposent sur de multiples applications ou bases de données;
- Produit un ensemble de services d'affaires utilitaires qui permettront l'automatisation de certaines procédures d'affaires.

Une SOA est une architecture qui se veut indépendante technologiquement. Mais les technologies qui implémentent cette architecture ne sont évidemment pas neutres. Un environnement utilisant le standard Corba, par exemple, peut être la base technologique d'une SOA. Une SOA doit, pour être reconnue comme telle, respecter certaines caractéristiques. Le Tableau 1.1 regroupe ces caractéristiques.

Tableau 1.1 Les caractéristiques d'une SOA
Tiré de (Wing et Venky, 2007)

Caractéristiques	Description
Exposé	Les applications exposent leurs services pouvant être accédés par d'autres applications ou services.
Distribué	Les services sont distribués et peuvent être installés à l'intérieur ou à l'extérieur des pare-feu de l'organisation.
Faible couplage	Les services sont faiblement couplés.
Propriété	Un ensemble de services n'a pas nécessairement un seul propriétaire.
Standard ouvert	Les services utilisent des standards ouverts plutôt que propriétaires.
Neutralité	Les services sont indépendants de la plateforme, du langage et du vendeur.
Création	De nouvelles applications, de nouvelles transactions ou de nouveaux services peuvent être créés par l'assemblage, la composition et le chaînage de services de plus bas niveau.
Transparence	Le détail de l'implémentation d'un service (transport de bas niveau, communication, plateforme, langage) est dissimulé à l'application qui utilise le service.
Réutilisabilité	La réutilisation des services est encouragée autant que possible.

1.2.1 Les couches d'abstraction d'une SOA

La Figure 1.2 illustre qu'une SOA est composée de plusieurs couches d'abstraction. La première couche représente l'ensemble des systèmes patrimoniaux et des bases de données auxquelles on souhaite donner accès. La seconde couche est l'abstraction des données. Toutes les données doivent être ramenées dans un format standardisé. La troisième couche est la messagerie. Cette couche assure le transport des données. Les services composent la quatrième couche et permettent la consommation (l'utilisation) de ces données. Une cinquième couche permet de regrouper plusieurs services entre eux par composition, orchestration ou chorégraphie pour réaliser une activité ou un processus d'affaires particulier.

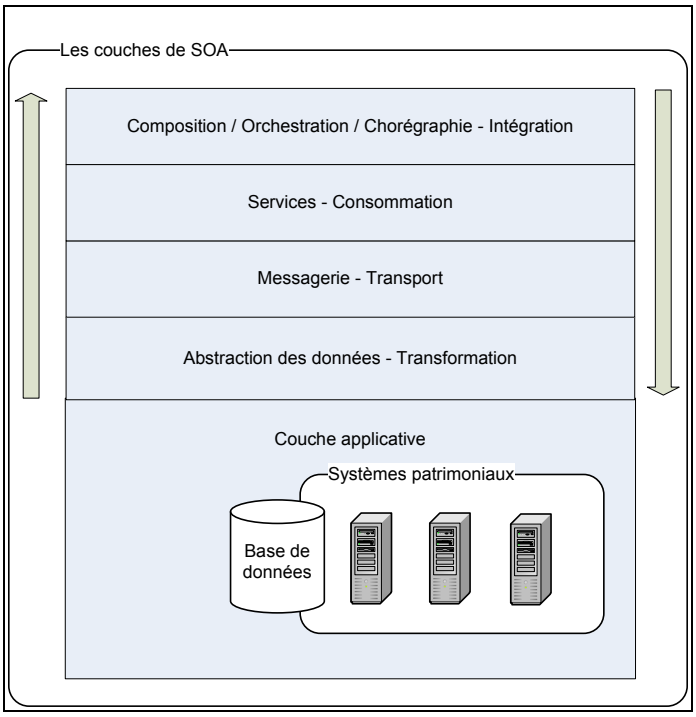


Figure 1.2 Les couches d'une SOA.
Adaptée de (Arsanjani, 2004)

Notons qu'aucune technologie n'est mentionnée dans la Figure 1.2. Que ce soit par le biais de services Web, de Corba ou de toute autre technologie, il faut que la technologie choisie couvre minimalement ces couches d'abstraction.

Pour la conception des services d'une SOA, (Erl, 2005) suggère quant à lui trois couches d'abstraction qui incluent implicitement les couches illustrées dans la figure ci-dessus. Les couches d'abstraction présentées par (Erl, 2005) sont illustrées dans la Figure 1.3 et consistent en :

La couche d'affaires

La couche d'affaires représente les processus d'affaires et les activités qu'ils contiennent. Ces processus et ces activités sont agnostiques de la technologie qui les supporte. Cette couche n'est pas représentée dans la Figure 1.2. C'est la couche de plus haut niveau.

La couche applicative

La couche applicative contient l'ensemble des applications et des bases de données qui supportent la mission de l'entreprise. Cette couche est située en dessous de la couche de « transformation » de la Figure 1.2. C'est la couche de plus bas niveau.

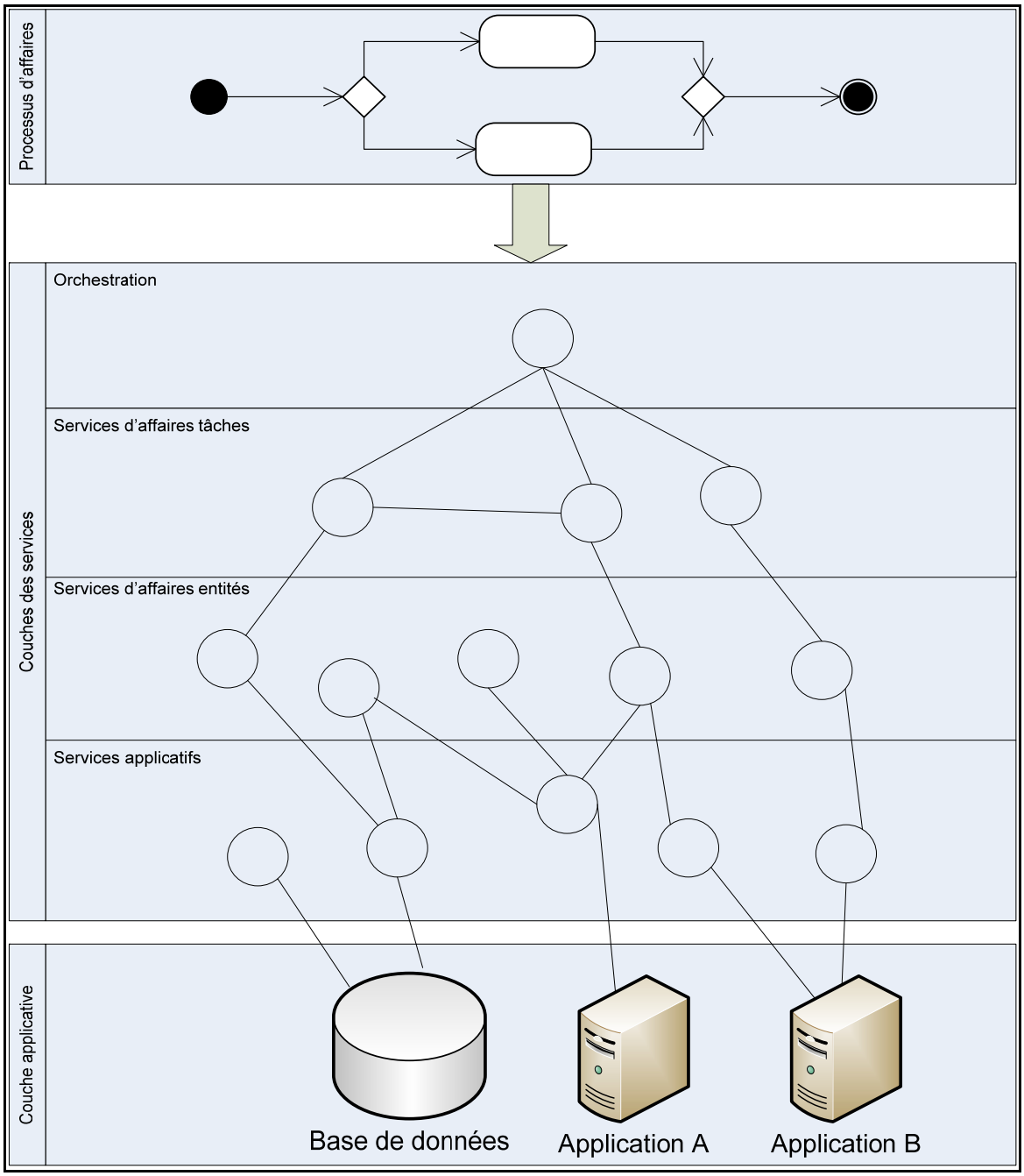


Figure 1.3 Les couches de services d'une SOA.
Adaptée de (Erl, 2005)

La couche des services

La couche des services se retrouve entre la couche d'affaires et la couche applicative. Elle correspond aux couches de « messagerie » et de « service » de la Figure 1.2

La couche des services peut elle-même être subdivisée en plusieurs sous-couches..Il y a tout d'abord la couche des services applicatifs. Les services applicatifs permettent d'interagir avec la couche applicative. Les services applicatifs seront utilisés par d'autres services de plus haut niveau. Ce sont les seuls services qui seront associés à une technologie particulière, celle de l'application avec laquelle ils interagissent.

La couche des services d'affaires permet de réaliser les activités automatisables des processus d'affaires. Ces services sont, grâce aux services applicatifs, indépendants de la technologie. On peut subdiviser la couche des services d'affaires en deux autres sous-couches. La couche des services d'affaires entités représente généralement les entités du domaine d'affaires. Les services de tâches regroupent des opérations faisant interagir des services d'affaires entités ou d'autres services de tâches.

Il pourra finalement y avoir une couche d'orchestration, optionnelle, qui aura pour mission de diriger le fonctionnement de plusieurs services d'affaires afin de réaliser un flux de travail plus complexe. Elle correspond à la couche « intégration » de la Figure 1.2.

1.3 Les types de processus

SOA est un paradigme architectural qui revoit la façon dont une entreprise organise ses processus. En effet, l’une des conséquences d’une SOA est l’automatisation et l’intégration de certains de ces processus avec des services.

Parfois, ce seront des processus internes à l'organisation. Parfois, ce seront des processus liant plusieurs organisations entres elles, tel l'achat de marchandise à un fournisseur. (White, 2008) définit trois types de processus qui sont susceptibles d'êtres modélisés: L'orchestration, la chorégraphie et la collaboration.

1.3.1 L'orchestration

L'orchestration se déroule dans un cadre privé, généralement pour un processus de l'entreprise. L'orchestration décrit comment le processus est réalisé. Par exemple, l'achat d'une pièce d'équipement. Il peut y avoir un ou plusieurs rôles impliqués dans la réalisation du processus. Par exemple, le processus d'ouverture d'un projet peut impliquer les départements de gestion des contrats, de gestion des projets et de comptabilité.

Dans le contexte d'une SOA, l'orchestration implique la coordination de plusieurs services pour réaliser un besoin d'affaires. Plusieurs besoins d'affaires peuvent être résolus par le biais de services déjà existants par composition (Erl, 2005, p. 200). Cette réutilisabilité est l'un des avantages d'une SOA.

Pour une PME, l'orchestration est particulièrement intéressante. C'est par le biais de l'orchestration qu'une intégration plus complète des systèmes d'information de l'entreprise avec les processus d'affaires est possible.

1.3.2 La chorégraphie

La chorégraphie permet la gestion des échanges entre plusieurs acteurs différents (partenaires, clients, fournisseurs). La chorégraphie inclut toutes les parties et offre une vue globale d'un flux de travail (Guilbault, 2007). Par exemple, le processus complet d'une commande, en incluant les actions et les messages échangés entre le client et le fournisseur.

Pour une PME, la chorégraphie peut être complexe à mettre en place. Cela implique que la PME interagit avec d'autres organisations et que ces participants ont un intérêt commun à chorégraphier leurs relations, par exemple, à l'aide de services Web. Dans le contexte d'une SOA, les services Web peuvent être utilisés aussi bien à l'intérieur d'une entreprise qu'avec d'autres entreprises (B2B ou *Business-to-Business*) ou encore avec ses clients (B2C ou *Business-to-Customer*). Mais implémenter une chorégraphie dans l'entreprise peut avoir un impact très important. Comme pour l'orchestration, on attendra d'avoir atteint une certaine expertise avec l'architecture SOA et les technologies utilisées.

Il n'est pas nécessaire de restreindre la chorégraphie à un processus entier. En effet, une organisation pourra offrir ou consommer des services provenant d'une autre entité pour l'aider à réaliser une certaine tâche sans pour autant chorégraphier l'ensemble du processus d'affaires.

La Figure 1.4 illustre la distinction entre la chorégraphie et l'orchestration. Dans ce cas-ci, il y a deux organisations ayant chacune un processus d'affaires distinct. Par exemple l'orchestration pour la commande d'une pièce et l'orchestration pour le traitement d'une commande de pièce. La chorégraphie serait l'intégration des deux processus. Si on suit cet exemple, l'action de commander une pièce déclenche automatiquement le processus de traitement d'une commande.

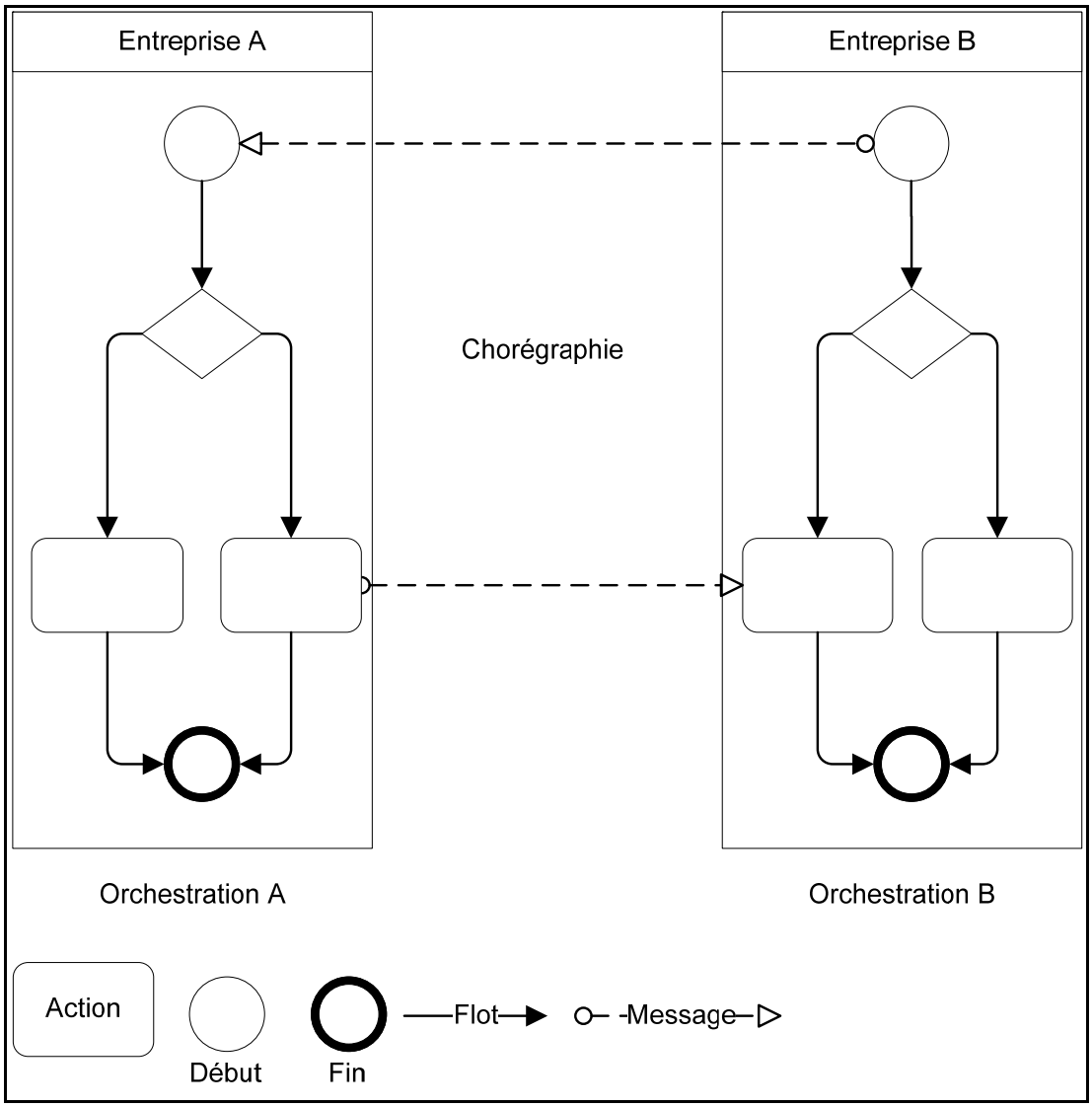


Figure 1.4 L'orchestration et la chorégraphie.

1.3.3 La collaboration

La collaboration permet de représenter les interactions entre les différents participants d'un processus, sans nécessairement avoir une connaissance détaillée d'un des processus (v*oir* Figure 1.5). Un processus impliquant deux participants, par exemple une université et un futur étudiant, est un exemple de collaboration. L'université a un processus d'inscription.

Mais ce processus n'a aucun contrôle sur le processus du futur étudiant. L'étudiant n'a pas de processus en soi.

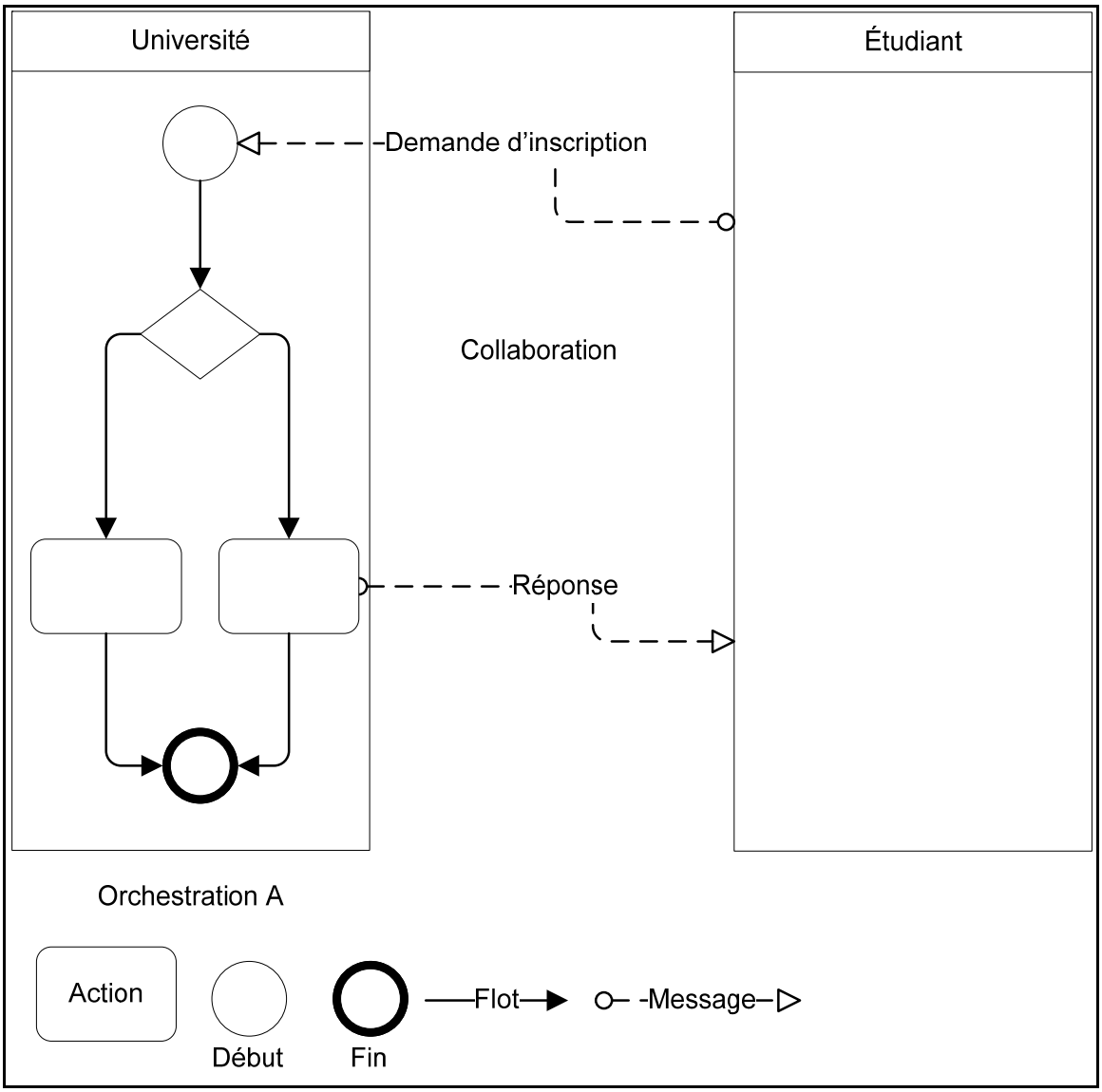


Figure 1.5 La collaboration.

1.4 Les services Web

Les servies Web sont une implémentation particulière et communément utilisée de SOA. Un service Web est un service logiciel auquel on accède par l'intermédiaire du Web (Daniel,

2003). Les services Web fournissent un standard d'interopérabilité entre différentes applications logicielles fonctionnant sur une variété de plateformes. Les services Web sont caractérisés par leur grande interopérabilité et leur grande extensibilité grâce à l'utilisation du standard XML.

Fondamentalement, un service Web exécute une tâche bien précise et permet à deux systèmes de communiquer entre eux. Les services Web peuvent être combinés afin de réaliser une opération plus complexe. On parle alors d'orchestration ou de chorégraphie selon que les opérations se déroulent dans une entreprise ou entre plusieurs entreprises. Des programmes offrant des services peuvent interagir entre eux afin d'offrir des services sophistiqués ayant une forte valeur ajoutée pour l'entreprise (W3C, 2002).

Les services Web sont préférablement faiblement couplés et ont une forte cohésion. Cela implique qu'un service exécute une action bien précise et que pour réaliser cette action, les dépendances à d'autres services ont été ramenées au minimum.

Les prochaines sections présentent les éléments, les standards et les extensions qui permettent la spécification et l'utilisation des services Web. On définit ce qu'est le langage XML et le document XSD. Ensuite on présente le contrat du service, le document WSDL. On décrit le protocole SOAP pour l'échange de messages ainsi que deux types différents de services Web, l'un qui utilise SOAP, l'autre qui utilise le protocole http. On conclut avec l'annuaire UDDI qui permet d'entreposer les services Web d'une organisation.

1.4.1 XML

XML (*eXtensible Markup Language* ou langage de balisage extensible) est un langage de balisage générique. C'est le W3C qui est responsable de ce format. XML est un langage qui est très intéressant pour une SOA, car il permet l'échange de messages dans un format compréhensible par tous. Un fichier XML permet de hiérarchiser l'information. Il peut être lu

par un individu, même si, à priori, les messages XML sont destinés à un échange entre machines (Daniel, 2003). Le langage XML est utilisé de façon intensive avec les services Web. On utilise XML pour définir l'interface du service, pour définir et valider le format des données qui transitent ainsi que pour les messages qui sont échangés entre les services.

1.4.2 XSD

Un document XSD (*XML Schema*) permet de définir la structure et le type des données qui peuvent être stockées dans un document XML. Un fichier XSD utilise la syntaxe XML. Les services Web utilisent les fichiers XSD afin de valider la structure et les types de données attendus à l'intérieur de l'enveloppe SOAP. Ainsi, les systèmes peuvent automatiser la validation d'un document XML (Daniel, 2003).

1.4.3 WSDL

WSDL (*Web Services Description Language*) est un standard utilisant le format XML pour décrire un service web. Il définit l'interface du service, permettant ainsi de connaître les paramètres nécessaires à l'appel du service. Un fichier WSDL se décompose en deux sections (Daniel, 2003; Erl, 2005).

Description abstraite

La première section constitue la partie abstraite de la description du service Web. Cette section donne des informations sur l'interface du service, sans référence à la technologie mise en place. Il y a trois principaux attributs dans cette section :

- L'attribut « *type* » permet de définir le format et la structure de la requête et de la réponse d'un service. On spécifie le schéma XSD utilisé afin de détailler la structure et le type des données qui va être transmis dans les messages.

- L'attribut « *message* » définit les messages qui seront transmis en entrées et en sorties, liés avec le format défini dans l'attribut « *type* ». Cet attribut peut également définir la séquence des échanges.
- L'attribut « *interface* » ou « *port-type* » définit les actions avec leurs paramètres, qui sont les « *messages* ».

Description concrète

La seconde section constitue la partie concrète de la description du service Web. Elle définit un moyen de communication concret permettant l'échange d'information :

- L'attribut « *binding* » définit le protocole de transport du message. Il existe plusieurs protocoles de transport possibles, par exemple HTTP, FTP, SMTP, etc.
- L'attribut « *port* » contient l'adresse physique d'où le service peut être utilisé à l'aide d'un protocole qui aura été spécifié dans l'attribut « *binding* ».
- L'attribut *« service* » est un regroupement de « *port* » associés.

1.4.4 SOAP

Le protocole SOAP a été défini par le W3C, suite aux travaux d'IBM et de Microsoft. L'objectif de ce protocole est d'assurer l'échange de messages XML (Daniel, 2003).

SOAP est un protocole permettant l'échange d'information dans un environnement décentralisé et distribué. C'est un protocole contenant trois parties :

1) L'enveloppe qui définit ce que contient le message et comment il doit être traité.

2) Un ensemble de règles d'encodage permettant de représenter des formats de données définis par des applications tierces.

3) Une convention pour se représenter les appels et les réponses distants (à l'aide de message RPC (*Remote Procedure Call*) ou de message de type document).

SOAP peut être transporté avec une variété de protocoles. Généralement, on l'utilise conjointement avec le protocole HTTP (W3C, 2000). SOAP peut être utilisé avec d'autres protocoles de transport, tels SMTP ou FTP.

1.4.5 Les extensions WS-*

Plusieurs extensions sont développées afin de pouvoir étendre les possibilités des standards WSDL et SOAP. Ce sont les extensions WS-*. Il en existe plusieurs. Certaines sont développées par le W3C, d'autres par l'OASIS.

Par exemple, l'extension *WS-Security* permet de définir l'utilisation du cryptage ou de signer les messages échangés entre deux services. De cette façon, l'intégrité du message est assuré d'un service à l'autre, contrairement aux services REST qui n'assurent la confidentialité des données que lors du transport.

WS-Transaction permet de définir dans les échanges le format d'une transaction, avec des facilités pour annuler (*rollback*) ou confirmer (*commit*) la transaction. Les extensions WS-* permettent également l'orchestration et la chorégraphie des services.

Contrairement au services Web REST, les extensions WS-* permettent plus de flexibilité. Ils peuvent notamment êtres utilisés dans un plus grand nombre de protocoles de communication. En contrepartie, cette flexibilité amène une plus grande complexité dans la mise en place de l'architecture, ce qui peut nuire au déploiement d'une SOA dans une PME.

1.4.6 Services Web REST

Les services Web REST sont des services qui utilisent le protocole http pour échanger des messages. C'est souvent la méthode la plus simple à mettre en place. Selon les exigences et les besoins d'un projet, une petite organisation pourrait avoir intérêt à utiliser cette architecture.

La protection des données dans un service Web REST est assuré par SSL (*Secure Socket Layer*) ou TLS (*Transport Layer Security*). Une fois l'information transmise, il est cependant plus difficile d'assurer la confidentialité des données si cette information doit parcourir d'autres étapes plus complexes. Les extensions WS-* peuvent alors offrir des solutions plus intéressantes.

1.4.7 UDDI

UDDI (*Universal Description, Discovery and Integration*) est un annuaire permettant de décrire un service, de le découvrir et de le consommer via le réseau Internet. On accède à l'UDDI via le protocole SOAP. Le standard UDDI a été développé par l'OASIS (OASIS, 2010).

La Figure 1.6 illustre le fonctionnement de l'UDDI. Le client demande à l'annuaire UDDI la description des services. Cette description est fournie grâce au fichier WSDL. Ce service a été préalablement enregistré par le fournisseur de service. Le client peut ensuite consommer le service via le protocole SOAP.

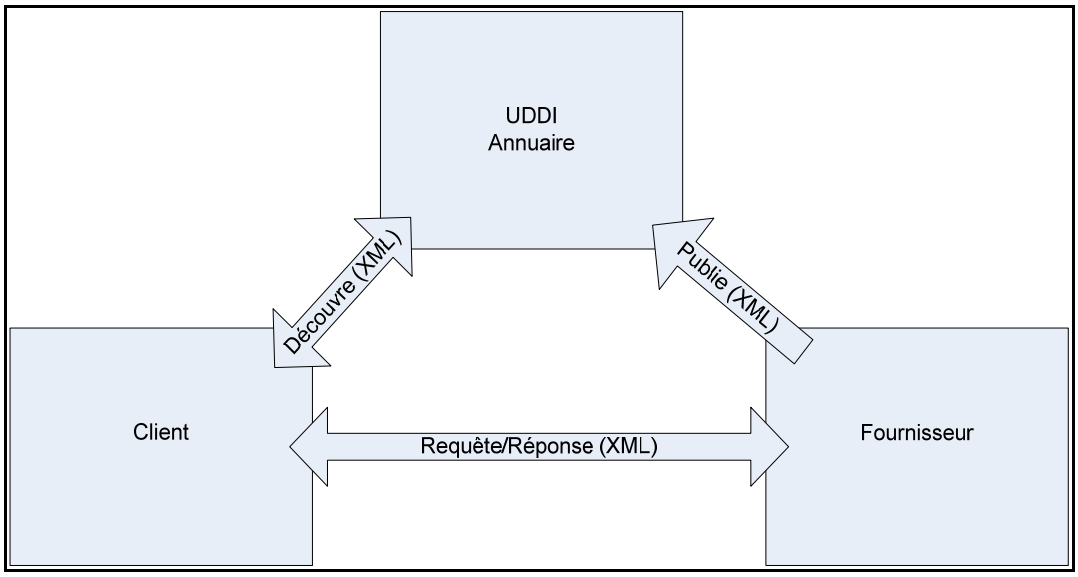


Figure 1.6 Le modèle d'UDDI.

La publication d'un service dans un annuaire UDDI requiert ces informations (Boatto, 2003; Erl, 2005) :

- Entités d'affaires ou *« Business entities »* : chaque service appartient à une organisation, qui doit être identifiée par un nom, une description, des personnes de contact, une ou plusieurs catégories, etc.

- Services d'affaires ou *« Business Services »* : incluent les informations non techniques relatives aux services offerts par l'entité d'affaires, comme le nom et une brève description.

- Gabarit de connexion ou « *Binding Template »* : définit où (URL) et comment (protocole) accéder au service.

- *« tModels »* : comprend des liens vers les informations techniques du service, par exemple un fichier WSDL décrivant les différentes opérations disponibles.

Un annuaire UDDI est intéressant lorsqu'une entreprise possède plusieurs services Web et qu'un référentiel devient nécessaire pour emmagasiner et retracer ces services. Lorsqu'on débute un travail pour implanter une SOA à l'aide des services Web, l'implantation d'un annuaire UDDI est prématurée.

Puisque l'expérience se déroule dans une entreprise n'ayant aucune expérience avec SOA et les services Web, il est peu probable qu'un annuaire soit utilisé lors des premières itérations. Il sera important de réévaluer ce besoin lors d'une utilisation systématique des services Web dans l'organisation.

1.5 Les organismes de standardisation

Il existe plusieurs organismes qui participent à l'élaboration des standards utilisés dans la mise en place d'une SOA à l'aide des services Web. Il est important de connaître les plus importants, car on y fait régulièrement référence. Des mises à jour des normes et des spécifications sont publiées par ces organismes et les standards évoluent rapidement.

1.5.1 L'OASIS

L'OASIS (*Organization for the Advancement of Structured Information Standards*) est un consortium regroupant 5000 personnes à travers 600 organisations. Il est à l'origine de plusieurs standards. L'OASIS a développé les normes UDDI, BPEL, plusieurs spécifications WS-* touchant à la sécurité ainsi que la norme BPEL4WS (*Business Process Execution Language for Web Services*) qui sera particulièrement utile pour l'intégration des services entre eux pour réaliser un processus d'affaires complexe (OASIS, 2010).

1.5.2 Le W3C

Le W3C (*World Wide Web Consortium*) est un organisme qui développe les standards touchant le Web et Internet. Le W3C est à l'origine des normes HTML, XHTML et XML. Il a aussi développé des standards important pour les services Web comme WSDL pour la définition de l'interface, SOAP pour l'échange d'information et quelques standards WS-*, entres autres pour l'adressage des messages (W3C, 2008).

La coordination des différents services entre deux organisations est possible grâce au langage WS-CDL (*Web Services Choreography Description language*) développé par le W3C (Erl, 2005, p. 208).

1.5.3 Le WS-I

Le WS-I (*Web Services Interoperability organization*) se donne pour mission d'établir de meilleures pratiques (regroupées dans des profils) à travers la multitude de normes et de standards qui sont développés autour des services web. Elle publie ces profils qui regroupent un ensemble de meilleures pratiques pour assister les développeurs. Elle offre aussi des exemples d'applications utilisant ces profils et développe des outils de tests pour assurer la conformité des services Web avec les standards et les meilleures pratiques de l'industrie (WS-I, 2008).

1.6 Les cadres d'architecture d'entreprise

L'architecture d'entreprise permet de structurer les processus d'affaires et les actifs informationnels d'une organisation. L'architecture d'entreprise englobe, entre autres, l'architecture des données, l'architecture applicative et l'architecture technologique.

L'architecture d'entreprise permet d'avoir une vue d'ensemble des différents éléments constituant l'organisation. L'objectif de l'architecture d'entreprise est d'améliorer l'efficience de l'entreprise. On y arrive généralement en documentant l'architecture actuelle, en identifiant les objectifs à moyen et long terme et en travaillant sur les écarts entre la vision actuelle et la cible visée (ou *gap analysis* en anglais).

Plusieurs éléments de l'architecture d'entreprise sont utilisés dans le cadre de ce mémoire. Les méthodologies orientées services se réfèrent aux processus d'affaires ainsi qu'aux actifs informationnels (processus d'affaires, actifs applicatifs, actifs technologiques, données). Cependant, elles intègrent rarement des activités d'architecture d'entreprise liées au développement d'une SOA. Cette lacune est comblée dans ce mémoire.

Il existe un grand nombre de cadriciels d'entreprise sur le marché. Dans ce mémoire, seuls les cadres de TOGAF, Zachman et *Enterprise Unified Process* (EUP) sont présentés. TOGAF a été retenu, car il est disponible, ouvert et très connu. De plus, TOGAF a été documenté dans un outil de modélisation des processus. Le cadre de Zachman est également l'un des cadriciels les plus importants et il a inspiré un grand nombre de cadres d'architecture d'entreprise, notamment aux États-Unis. EUP est intéressant, car il utilise une terminologie et une symbolique très proche de la méthodologie UP.

1.6.1 TOGAF

TOGAF (*The Open Group Architecture Framework*) est un cadriciel d'architecture d'entreprise développé par l'organisme « The Open Group » (*The Open Group : Making Standards Work*, 2010). Ce cadriciel regroupe un ensemble de bonnes pratiques afin de développer et maintenir une architecture d'entreprise.

TOGAF couvre quatre domaines d'architecture. L'architecture d'affaires définit la stratégie d'affaires et la gouvernance. L'architecture des données définit la structure et la gouvernance

des données logiques et physiques de l'organisation. L'architecture applicative offre une vision des applications, leurs relations avec les processus d'affaires réalisés et leurs interrelations avec les autres applications. L'architecture technologique représente l'infrastructure sur laquelle reposent les applications d'affaires et les données.

Pour permettre son implémentation, TOGAF décrit une méthodologie en plusieurs phases, l'« *Architecture Development Method* » (ADM), illustrée dans la Figure 1.7. Chaque phase est ensuite brièvement expliquée (Josey et al., 2009).

La phase préliminaire

C'est une phase de préparation. On identifie les intervenants. On définit un cadre d'architecture en prévision de l'élaboration d'une nouvelle architecture. On s'assure également du support de la direction dans cette démarche.

Gestion des exigences

Cette phase centrale dirige toutes les autres phases du cycle. Elle a pour objectif d'identifier les exigences. En effet, la nature évolutive de l'architecture amène à avoir un mécanisme central qui documente les exigences au fur et à mesure qu'elles sont identifiées. Ces exigences sont ensuite transmises comme intrants aux phases appropriées. L'impact de ces exigences sera étudié puis elles seront priorisées. C'est également dans cette phase que les conflits entre les exigences sont réglés.

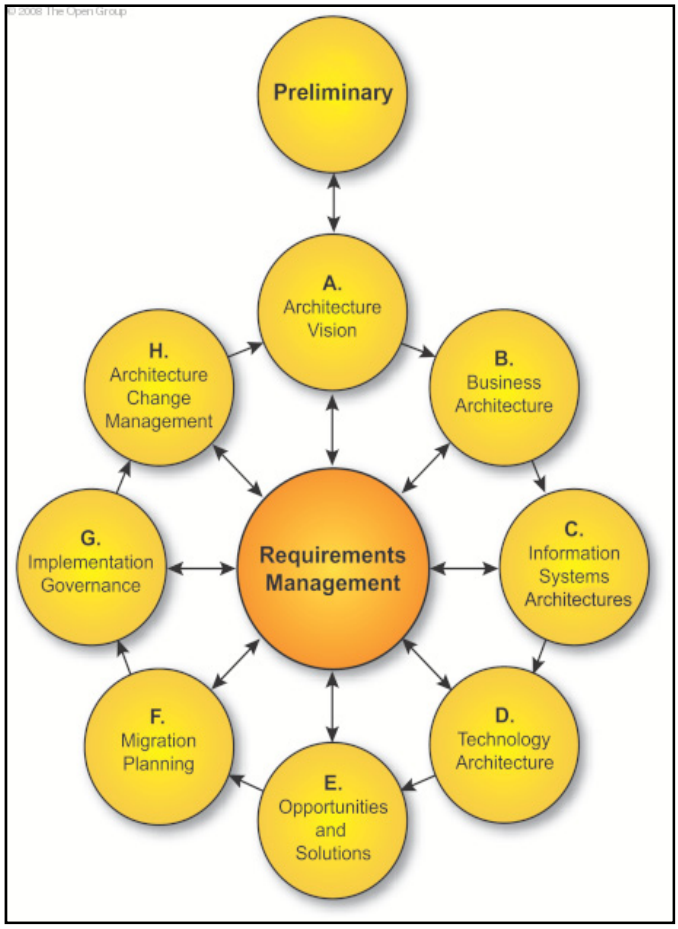


Figure 1.7 Les phases de TOGAF.
Tirée de (TOGAF, 2009)

A. La vision

C'est la première itération du cycle de développement de l'architecture d'entreprise. Lors de cette phase, on définit la vision, la portée, la mission, les stratégies utilisées et les buts de l'entreprise. On définit les tâches architecturales qui seront à réaliser.

B. L'architecture d'affaires

Lors de cette phase, on définit les produits et services offerts par l'organisation, on documente à l'aide de modèles d'activité, de cas d'utilisation et de modèles du domaine l'architecture d'affaires actuelle. On définit ensuite une architecture d'affaire cible. On compare la situation actuelle et future et on évalue l'écart. On rencontre les parties prenantes concernées.

C. L'architecture des systèmes d'information

La phase C documente les architectures applicative et informationnelle. On identifie les systèmes actuels et les systèmes cibles et on évalue l'écart. On définit un budget pour les systèmes à acquérir ou à rénover. On identifie les types d'applications. On documente les contraintes auxquelles il faut faire face, par exemple des contraintes de temps, des contraintes environnementales ou encore d'interopérabilité. On rencontre les parties prenantes concernées.

D. L'architecture technologique

La phase D documente l'architecture des composantes technologiques associées à l'architecture des systèmes d'information. Cela inclut le matériel et les logiciels. Cette phase comprend également l'infrastructure supportant les communications dans l'organisation. On documente l'architecture initiale et l'architecture visée et on étudie l'écart. On rencontre les parties prenantes concernées.

E. Opportunités et solutions

Lors de la phase E, on identifie les véhicules qui permettront la livraison des projets, programmes ou portefeuilles. On documente les transitions entre l'architecture actuelle et

future. On écrit un plan de mise en œuvre qui sera approuvé. On entreprend la réflexion sur l'implémentation. On consolide les différentes études d'écart des phases précédentes. L'architecte doit considérer les dépendances entre les différents éléments de l'architecture. Cela amènera peut-être la définition d'une ou plusieurs architectures de transitions.

F. Planification de la migration

Cette phase permet d'assurer le bon déroulement de la transition vers une nouvelle architecture d'entreprise. On identifie les dépendances, les coûts, les délais possibles. Les projets sont priorisés. On rédige un plan de migration. On fixe une chronologie dans l'exécution du plan de mise en œuvre. Les spécifications qui ont été rédigées plus tôt (contraintes, exigences, architectures) sont finalisées à cette étape.

G. Gouvernance de la mise en œuvre

La phase G valide la conformité entre la vision et la mise en œuvre. On prépare les équipes de support. On formule des recommandations quant aux déploiements des solutions. Tout au long du déploiement, on utilise des métriques de conformité. Lorsqu'un déploiement est terminé, on effectue une analyse rétrospective et on en tire des leçons pour le prochain déploiement.

H. Gestion du changement

La phase H assure le suivi de l'architecture pour préserver sa pertinence. Elle suit l'évolution des besoins d'affaires et des technologies. On évalue la performance de la nouvelle architecture et on propose des corrections ou des ajustements. Le cycle peut alors recommencer depuis le début.

1.6.2 Le cadre de Zachman

Le cadre de Zachman a été développé dans les années 80 par John Zachman (Zachman, 1987). Ce dernier a diviser l'architecture d'une organisations en plusieurs vues distinctes. Ce cadre, qui se présente sous la forme d'un tableau (voir Figure 1.8), vise à organiser et analyser les données selon deux dimensions présentant les différentes vues d'une entreprise. Le cadre de Zachman est propriétaire. S'il existe beaucoup de documentation sur les principaux concepts, le cadre n'est pas en soit une méthodologie ni un processus mais plutôt une structure qui décrit et organise les différentes vues de l'organisation (Zachman, 2008). Chaque élément du tableau de la Figure 1.8 peut contenir plusieurs vues permettant de documenter l'architecture de l'entreprise.

	Why	How	What	Who	Where	When
Contextual	Goal List	Process List	Material List	Organizational Unit & Role List	Geographical Locations List	Event List
Conceptual	Goal Relationship	Process Model	Entity Relationship Model	Organizational Unit & Role Rel. Model	Locations Model	Event Model
Logical	Rules Diagram	Process Diagram	Data Model Diagram	Role relationship Diagram	Locations Diagram	Event Diagram
Physical	Rules Specification	Process Function Specification	Data Entity Specification	Role Specification	Location Specification	Event Specification
Detailed	Rules Details	Process Details	Data Details	Role Details	Location details	Event Details

Figure 1.8 Le cadre de Zachman
Tirée de (Le Framework de Zachman, 2011)

Voici une brève description de la Figure 1.8 tirée de (*Le Framework de Zachman*, 2011) et de (Zachman, 2008).

La vue contextuelle

- Pourquoi: identifier les types d'objectifs de l'organisation,
- Comment: identifier la liste des principales fonctions d'affaires de l'organisation et les différents types de processus.
- Quoi: identifier la définition de l'organisation, de ses différentes filiales et leurs interrelations. Identifier les actifs de l'organisation.
- Qui: identifier le type des unités d'affaires et des différents département.
- Où: identifier le type de localisation géographique de l'organisation et de ses entités
- Quand : identifier les principaux types d'évènement qui activent les principales fonctions d'affaires.

La vue conceptuelle

- Pourquoi: identifier les principaux objectifs d'affaires de l'organisation, la mission de l'organisation. Identifier les liens qui unissent les différents objectifs de l'organisation.
- Comment: créer le modèle des processus d'affaires, en prenant soin d'identifier les intrants et les extrants de ces processus.
- Quoi: identifier les éléments physiques de l'organisation et leurs relations. Identifier les entités d'affaires et leurs interrelations.
- Qui: identifier les rôles dans l'organisation et leurs interrelations. Assignation des rôles et des responsabilités.
- Où: identifier les lieux physiques et leurs interrelations.
- Quand: identifier les évènements ponctuels et cycliques qui agissent sur l'organisation et sur les affaires.

La vue logique

- Pourquoi: identifier les règles qui appliquent une contrainte sur les processus d'affaires, sans égard à l'implémentation physique.
- Comment: décrire plus en détail la réalisations des différents processus dans le système et la transformation des intrants, sans égard à l'implémentation physique.
- Quoi: identifier et décrire les entités du systèmes et leurs interrelations, sans égard à l'implémentation physique.
- Qui: identifier les rôles et leurs interrelations dans les systèmes, grouper par types de livrables, sans égard à l'implémentation physique.
- Où: identifier où sont situés les systèmes et les données, sans égard à l'implémentation physique.
- Quand: identifier les évènements ponctuels et cycliques et leurs interrelations sur les processus et leurs activités dans le système, sans égard à l'implémentation physique.

La vue physique

- Pourquoi: les règles et politiques s'appliquant aux technologies, dans un langage formel.
- Comment: implémente les fonctions à l'aide d'une technologie permettant la réalisation d'un processus et la transformation des intrants.
- Quoi: implémente les entités de données et des éléments technologiques, incluant les attributs et leurs interrelations.
- Qui: définir les rôles des responsables de l'exécution du travail et du flux de travail. Rôles des technologies.
- Où: localiser les infrastructures physiques où sont hébergées les technologies et où se déroulent les activités.
- Quand: identifier les évènements liées aux technologies.

1.6.3 Le processus unifié d'entreprise

Le processus unifié d'entreprise (*Enterprise Unified Process* ou EUP) est une méthodologie développée par (Scott W.Ambler, 2005). Inspiré de RUP, cette méthodologie propose un ensemble de disciplines d'entreprise qui précèdent les phases de développement classique. EUP ajoute également deux nouvelles phases, « production » et « retrait », qui suivent la phase de transition.

L'auteur aborde, par le biais de sept nouvelles disciplines, la modélisation de l'entreprise, la gestion du portefeuille, l'architecture d'entreprise, la réutilisabilité stratégique, la gestion des ressources humaines, l'administration de l'entreprise et les processus d'améliorations logicielles. La Figure 1.9 illustre la réalisation de ces disciplines à travers les phases de RUP. Ces disciplines consistent en :

- **Modélisation d'entreprise**
  Cette discipline documente le vocabulaire, l'environnement, les processus et les règles d'affaires ainsi que la structure de l'organisation. On rédige la vision et l'on définit les grandes lignes de la stratégie d'exécution.

- **Gestion du portefeuille**
  On inventorie les systèmes existants, on crée un portefeuille des systèmes et des projets. On regroupe les systèmes partageant les mêmes objectifs d'affaire en programme. On priorise les projets selon la stratégie et la vision énoncées dans la modélisation d'entreprise. On gère le risque au niveau de l'entreprise.

- **Architecture d'entreprise**
  On se réfère aux travaux précédents pour définir l'architecture d'entreprise. Ce travail est itératif. Après avoir identifié les exigences, on définit une architecture candidate. Après plusieurs raffinements, on obtient une architecture d'entreprise complète, intégrant autant

de vues que nécessaire. On définit aussi des architectures de références qui sont des patrons génériques sur lesquels les développeurs peuvent se référer pour accélérer le développement de nouveaux systèmes.

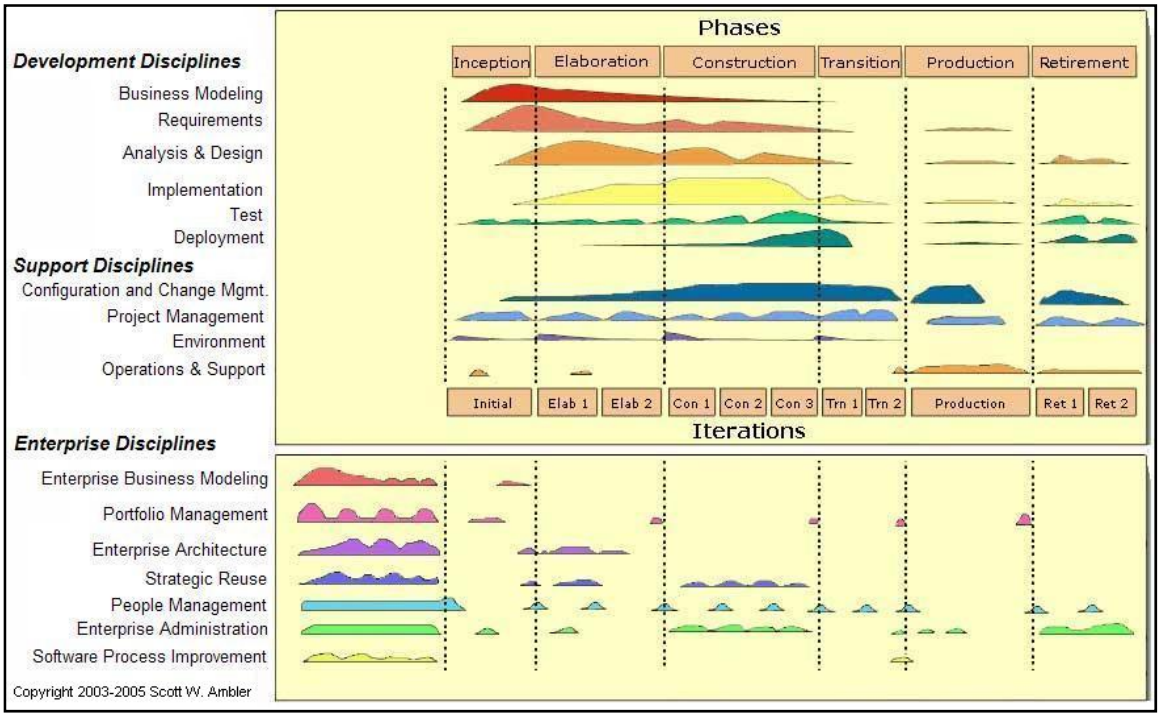


Figure 1.9 Le processus unifié d'entreprise.
Tirée de (Scott W.Ambler, 2005)

- **Réutilisation stratégique**

Cette discipline permet d'élaborer un plan afin de réaliser une réutilisation stratégique des actifs d'une organisation à travers plusieurs projets.

- **Gestion des ressources humaines**

Cette discipline se concentre sur la gestion des ressources humaines. Elle inclue des activités reliées à la gestion de carrière et au plan de relève.

- **Administration de l'entreprise**

Cette discipline aligne les différents actifs de l'entreprise (infrastructure, information, sécurité). Elle s'assure du respect des politiques internes.

- **Processus d'améliorations logicielles**

On identifie les besoins en processus de développement. On récupère les éléments dans les méthodologies existantes. Si ceux-ci ne satisfont pas le besoin, on peut aussi en créer pour les besoins plus spécifiques. Finalement, on déploie ces processus logiciels.

1.7 Les méthodologies SOA

Il existe un certain nombre de méthodologies dédiées aux SOA. La plupart sont propriétaires et la description de leur contenu est souvent fragmentaire. Il existe également quelques méthodologies publiques. Les prochaines sections décrivent sommairement les principales méthodologies SOA identifiées dans la revue de la littérature.

1.7.1 SOMA et SOAD d'IBM

Les méthodologies SOMA (*Service Oriented Modeling and Architecture*) et SOAD (*Service Oriented Analysis and Design*) d'IBM sont des méthodologies SOA propriétaires. La documentation disponible est donc limitée au moment d'écrire ces lignes.

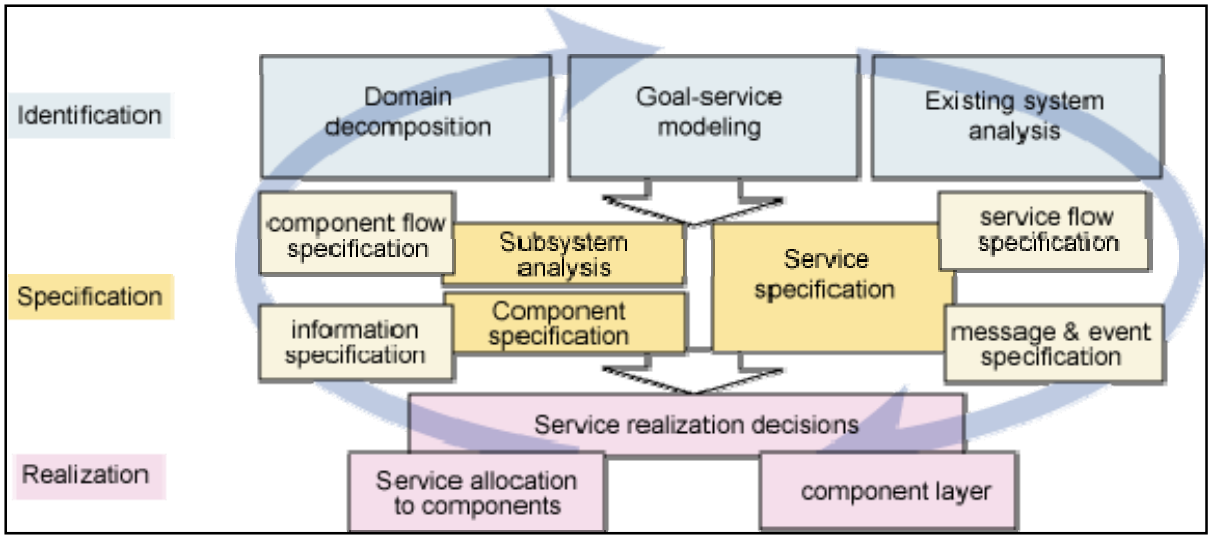


Figure 1.10 La méthodologie SOMA.
Tirée de (Arsanjani, 2004)

SOMA fractionne la modélisation d'un service en trois étapes : identification, spécification et réalisation. Comme l'illustre la Figure 1.10, l'identification incorpore des activités de haut niveau, comme la décomposition du domaine et l'analyse des systèmes existants. Cela implique la connaissance du portefeuille d'applications. La phase de spécification analyse les systèmes touchés et définit les spécifications du service, des composantes, de l'information, des messages, etc. Les flux sont également détaillés. Finalement, la phase de réalisation permet le développement du service (Arsanjani, 2004).

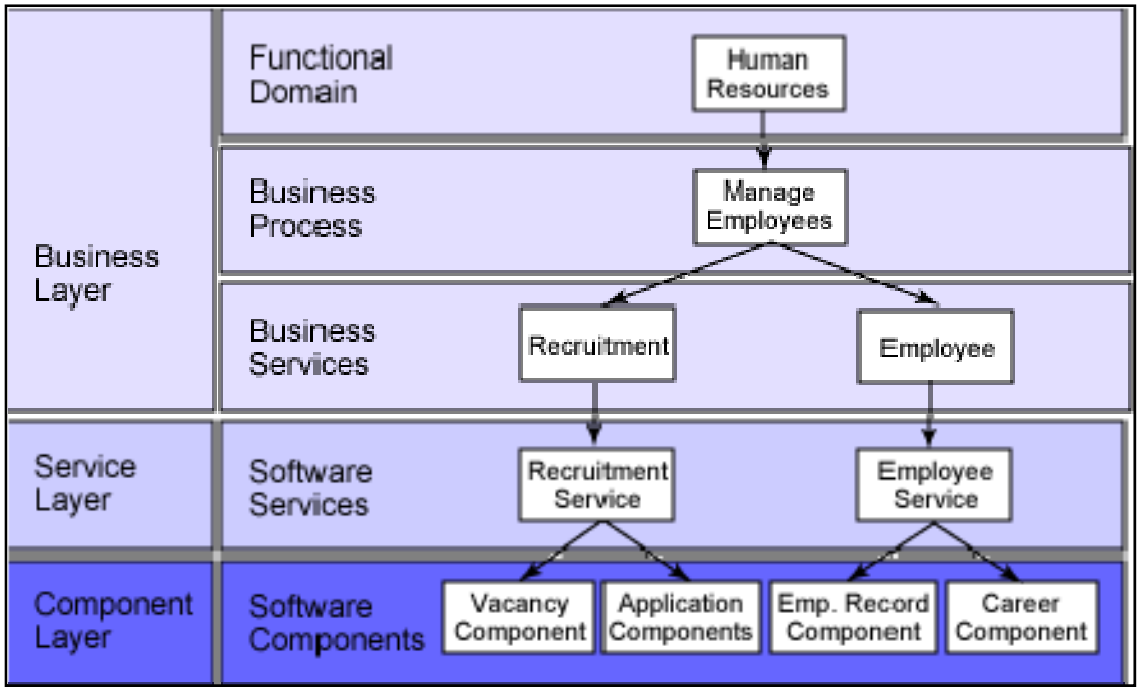


Figure 1.11 La méthodologie SOAD.
Tirée de (Zimmermann, Krogdahl et Gee, 2004)

Dans la Figure 1.11, on voit comment SOAD décompose un problème en plusieurs couches, la couche supérieure faisant l'abstraction de la couche inférieure. On fait clairement la distinction entre la couche d'affaires qui représente les processus et les services d'affaires, la couche de service qui offre le service logiciel, et finalement la couche composante qui implémente la logique métier dans les systèmes patrimoniaux.

Les auteurs recommandent d'intégrer les activités d'architecture d'entreprise avec les autres activités de modélisation incluse dans le cadriciel TOGAF. Par contre, aucune activité d'architecture d'entreprise n'est arrimée directement avec SOAD.

1.7.2 SOA RQ de SUN

SOA RQ est une méthodologie propriétaire développé par SUN (SOA RQ Methodology : A Pragmatic Approach, 2006). Elle est annoncée comme itérative et incrémentale. Puisque SOA RQ est propriétaire, la documentation disponible est limitée. Elle se décompose en cinq phases : initialisation, élaboration, construction, transition et conception (*Voir* Figure 1.12).

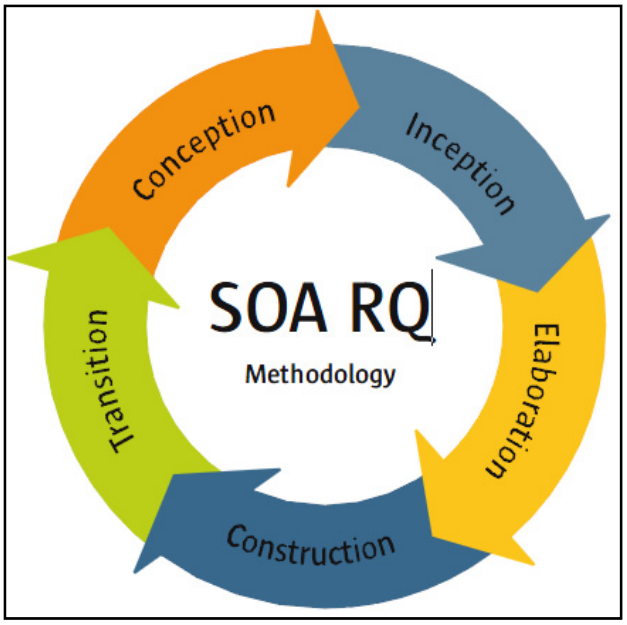


Figure 1.12 La méthodologie SOA RQ.
Tirée de (SOA RQ Methodology : A Pragmatic Approach, 2006).

Ces phases semblent très similaires dans la terminologie aux phases de RUP et on présume que les activités associées sont également similaires. Selon l'étude comparative réalisée par (Ramollari, Dranidis et Simons, 2007), SOA RQ est utilisée à grande échelle dans l'industrie. Aucune autre information ne vient appuyer cette affirmation. Aucune publication n'a été trouvée qui détaille l'approche, les différentes couches d'abstractions utilisées ou les artéfacts produits.

1.7.3 SOAF

La méthodologie SOAF (*Service Oriented Architecture Framework*) est une méthodologie réalisée par (Erradi, Anand et Kulkarni, 2006). Contrairement aux méthodes propriétaires précédentes, cette méthodologie détaille les artéfacts entrants, les activités et les artéfacts sortants pour chacune des phases. C'est à première vue la méthodologie disponible la plus complète. SOAF se décompose en cinq phases :

1. **Extraction de l'information**

Cette phase se concentre sur l'étude et la documentation de l'architecture d'affaires. On modélise les processus d'affaires, on fait l'inventaire des applications et on associe les processus d'affaires avec leurs applications respectives.

2. **Identification des services**

Cette phase décompose les processus d'affaires qui ont été préalablement modélisés afin d'identifier des services. Dans ce cas, on crée différents types de services; les services d'affaires sont associés à des tâches d'affaires alors que les services applicatifs communiquent avec les applications qui supportent les processus.

3. **Définition des services**

Cette phase modélise les services identifiés précédemment. On spécifie les interfaces des services, ainsi que les standards, qui seront utilisés. Les règles et politiques que devront respecter les futurs échanges de messages sont établies.

4. Réalisation des services

Cette phase définit la stratégie pour mettre en place les services identifiés à la phase d'identification. On identifie les transformations qui devront être faites sur les données ou sur les applications qui supportent le processus d'affaires. On développe les services selon la stratégie choisie.

5. Feuille de route et planification

Cette phase permet d'identifier les risques et la stratégie pour mettre en place les services qui ont été développés sans nuire au déroulement des activités de l'organisation. C'est également dans cette phase qu'est établie la gouvernance et que sont identifiés les propriétaires des services. Cette phase embrasse une vision plus large en analysant les risques et les plans de contingences de plusieurs projets.

La Figure 1.13 SOAF se réfère à des artéfacts liés à des activités d'architecture d'entreprise (applications existantes, portefeuille, modèles et bonnes pratiques) sans pour autant donner davantage de détails sur la réalisation de ces activités. Cette méthodologie n'est enchâssée ni dans une méthodologie de développement complète ni dans un cycle incluant des disciplines d'entreprises.

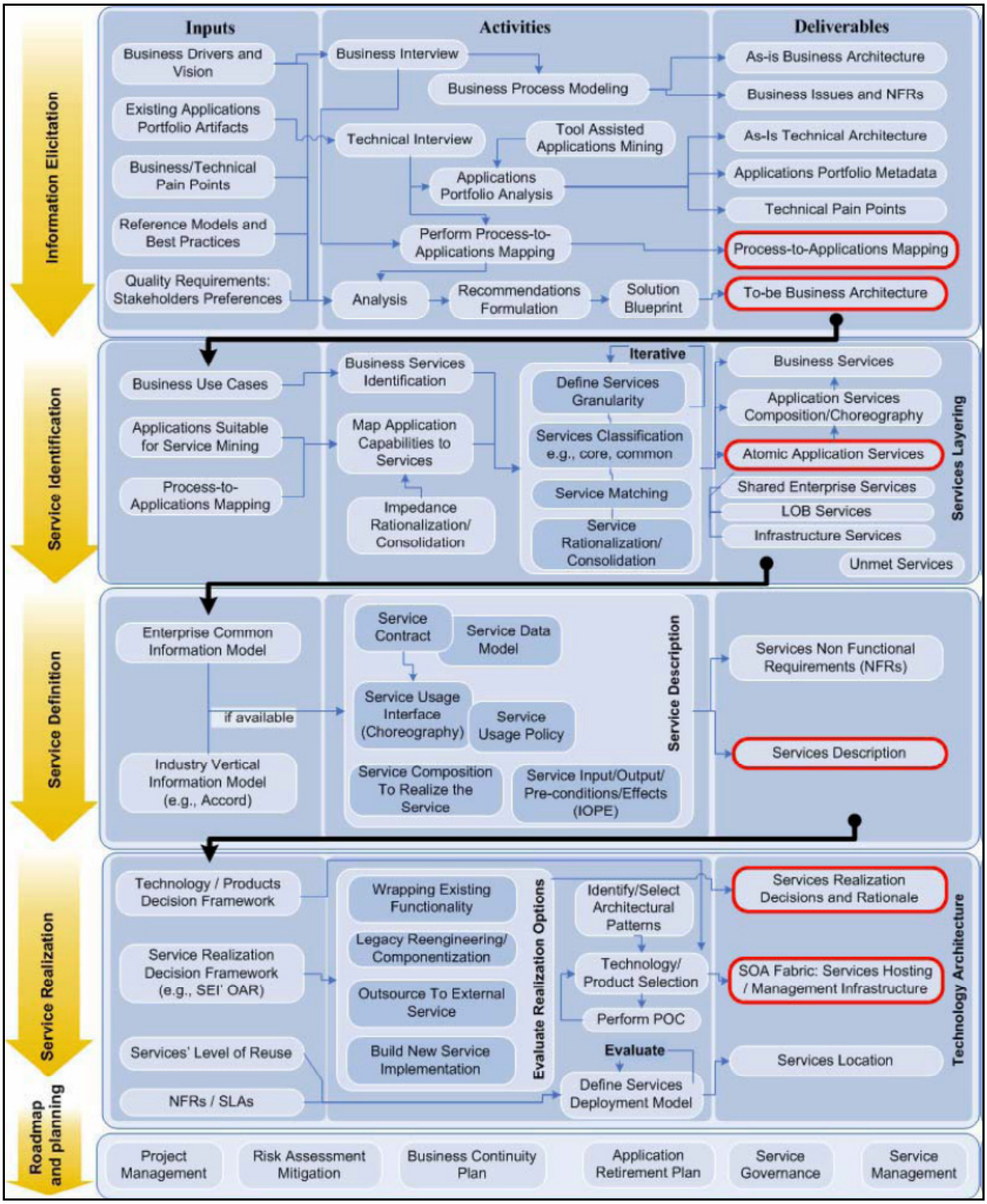


Figure 1.13 La méthodologie SOAF.
Tirée de (Erradi, Anand et Kulkarni, 2006)

1.7.4 La méthodologie SOA de Thomas Erl

(Erl, 2005) propose six phases dans le cycle de vie d'un développement SOA (*Voir* Figure 1.14). Selon l'auteur, un projet SOA diffère des méthodes de développement standard uniquement dans les phases d'analyse et de conception alors que les phases subséquentes sont très similaires.

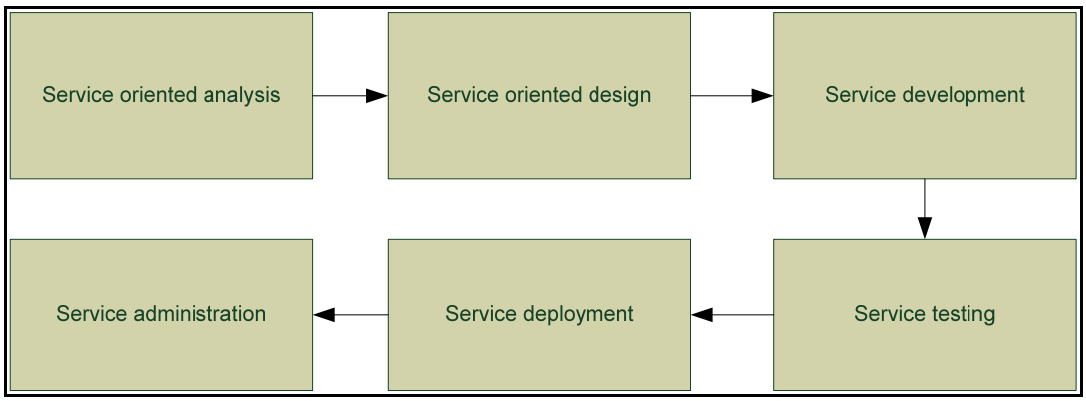


Figure 1.14 La méthodologie de Thomas Erl.
Adaptée de (Erl, 2005)

Les six phases de ce cycle de vie sont :

- **Analyse orientée service**

Ensemble des activités d'analyse permettant d'identifier les processus qui peuvent être représentés comme des services. On identifie les services candidats.

- **Conception orientée services**

Ensemble des activités regroupant les services identifiés précédemment en un ensemble cohérent. Association avec les processus d'affaires existants. Séparation par couches selon que le service agisse au niveau de la coordination globale des services, du processus d'affaires ou de l'infrastructure.

- **Développement du service**

Reprend globalement les activités de développement des méthodologies classiques. Utilisation d'un cadriciel de développement quelconque afin d'implémenter le service.

- **Test du service**

Reprend globalement les activités de développement des méthodologies classiques. Utilisation des techniques de test afin de vérifier et valider le fonctionnement des services. Il faut noter que la nature distribuée des services complique les activités de test.

- **Déploiement du service**

Reprend globalement les activités de transition des méthodologies classiques. Test d'intégration et intégration du service dans l'environnement de production.

- **Administration du service**

Reprend globalement les activités de maintenance des méthodologies classiques. Gestion de l'évolution du service, gestion des changements et des demandes de modification.

1.8 Les méthodologies de développement

Ce travail porte sur l'implantation d'une SOA à l'intérieur d'une PME à l'aide de méthodologies agiles. Cette méthodologie doit donc inclure un cycle de développement agile et les pratiques qui lui sont associées. Ce choix est justifié par la taille des PME et, plus spécifiquement, la taille et la complexité des projets de développement que l'on y trouve. Nous allons présenter succinctement dans les prochaines sections quelques méthodologies de développement agiles répandues avec leurs principales caractéristiques. Une de ces méthodologies sera ensuite sélectionnée pour l'intégration de processus et pratiques de développement SOA.

1.8.1 UP et RUP

UP (*Unified Process*) est une méthodologie publique qui a été développée et publiée dans un livre en 1998 (Jacobson, Booch et Rumbaugth, 1998). RUP (*Rational Unified Process*) est une méthodologie propriétaire très connue développée et maintenue par IBM et qui est largement inspirée d'UP. C'est pourquoi on regroupe ces deux méthodologies dans cette section. RUP, contrairement à UP, bénéficie du support d'IBM, qui a intégré sa variante de la méthodologie dans un gestionnaire de processus qui permet de modifier le processus. De plus, RUP a continué d'évoluer au cours des dernières années.

Elles se décomposent en quatre grandes phases : initialisation, élaboration, construction et transition. Ces phases correspondent à l'avancement du projet. Elles détaillent aussi un ensemble de disciplines qui seront exécutées à différents niveaux pendant chacune des phases. (IBM, 2007).

L'initialisation consiste en l'identification des besoins d'affaires et des exigences. Pendant la phase d'élaboration, on réalise l'analyse et la conception du logiciel et on identifie les éléments les plus risqués. Dans la phase de construction, on implémente les fonctionnalités. À cette étape, les éléments les plus risqués ont été mitigés. Finalement, la transition permet le déploiement du logiciel (Larman, 2005).

UP et RUP sont des méthodologies itératives et incrémentales. Mais la grande quantité de processus et d'artéfacts qui composent ces méthodologies les exclut des méthodologies agiles. Elles n'en restent pas moins parmi les méthodologies les plus répandues sur le marché du développement logiciel.

Dans la Figure 1.15, on peut voir que certaines disciplines seront davantage sollicitées pendant la phase d'initialisation (*inception*), comme la modélisation d'affaires et

l'explicitation des exigences alors que le déploiement se fera plutôt entre les phases de construction et de transition.

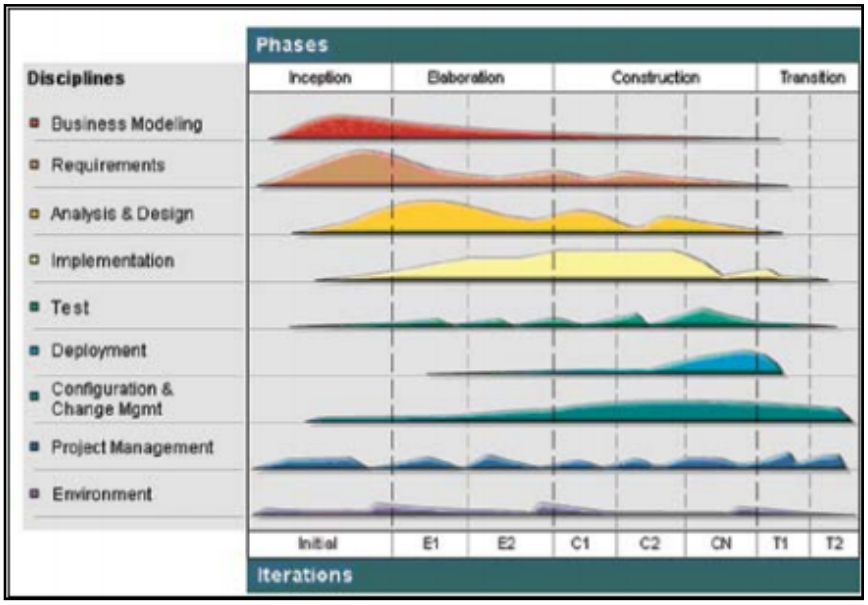


Figure 1.15 Les phases et disciplines de RUP.
Tirée de (IBM, 2007)

Bien que ni UP ni RUP ne soient pas des méthodologies agiles et qu'elles ne soient pas nécessairement adaptées aux besoins des plus petites organisations, elles restent des méthodologies auxquelles on fait régulièrement référence, que ce soit pour les concepts ou encore la terminologie.

1.8.2 SCRUM

SCRUM est une méthode agile publiée en 1996. Dans SCRUM, un projet est subdivisé en plusieurs petites itérations. Chaque itération doit être réalisée dans un laps de temps relativement court (plus ou moins 30 jours pour certains). C'est ce qu'on appelle le sprint. Ce même sprint se décompose en bloc de 24 heures où les développeurs vont coder, tester et

publier leur travail. Chaque itération doit se conclure par un ou plusieurs livrables (le but). On regroupe les sprints en « *release* » (Alliance, 2009). La Figure 1.16 illustre ce cycle.

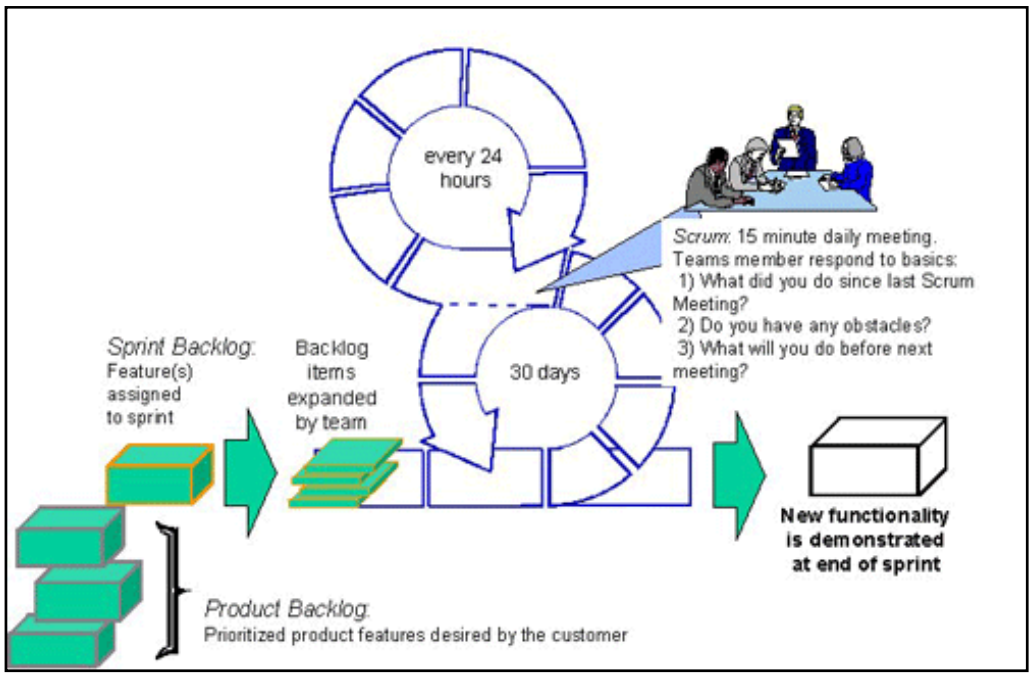


Figure 1.16 Le cycle de Scrum.
Tirée de (Alliance, 2009)

SCRUM adhère aux principes du manifeste agile (Kent *et al.*, 2001) en intégrant une participation active du client pour déterminer les priorités de chaque sprint et but. Ces besoins sont consignés dans ce qu'on appelle une liste de tâches ou « *backlog* ». On retrouve le cahier de charge du produit qui consigne l'ensemble des fonctionnalités du logiciel à produire alors que la liste de tâches du sprint contient plutôt la planification et la répartition en temps nécessaire pour sa réalisation.

Le cycle inclut aussi des activités quotidiennes de suivi, d'où l'importance de réaliser le code, les tests et la publication tous les jours. Cela assure ainsi un meilleur contrôle du risque advenant des retards dans la réalisation d'un sprint (Alliance, 2009).

1.8.3 Extreme Programming

L' « *Extreme Programming* » (XP) est une méthode agile qui pousse à l'extrême la simplicité du cycle de développement. Cette méthode de développement se veut itérative et incrémentale. Dans cette approche, le client est très sollicité pendant toutes les étapes du projet. C'est le client qui oriente les priorités. Dans sa forme la plus pure, on retrouve les pratiques suivantes (Beck, 1999) :

- **Client sur le site**

Dans XP, le client et le programmeur sont physiquement au même endroit. Cela facilite la communication et une compréhension commune du problème et de la solution.

- **Jeu du planning**

Un projet est d'abord divisé en versions, puis en itérations et finalement en tâches. Chaque tâche représente de deux à trois jours de travail alors que chaque itération dure en moyenne de deux à trois semaines. De cette façon, les rétroactions sont nombreuses et le client peut rapidement constater l'avancement des travaux.

- **Intégration continue**

Les développeurs intègrent leur code en continu. Cette intégration inclut les tests unitaires, les tests fonctionnels et le déploiement. Le client peut ainsi voir en tout temps l'avancement des travaux. Le système peut être livré à tout moment, à condition que les tests unitaires et fonctionnels soient réalisés

- **Petites livraisons**

Toujours dans le but de recueillir une rétroaction rapide du client, les versions doivent être relativement courtes et ne pas dépasser quatre mois.

- **Rythme soutenable**

Dans XP, les heures supplémentaires sont fortement déconseillées. Généralement, l'accumulation d'heures supplémentaires diminue le moral de l'équipe et augmente sa fatigue. Le nombre d'erreurs augmente. Il faut alors consacrer du temps à les corriger. Ce n'est pas efficace.

- **Dirigé par les tests**

Pour supporter l'intégration continue, il faut que le développement soit dirigé par les tests. Les tests doivent être écrits avant le code et toujours réussir à 100% dans le cas des tests unitaires. Le temps investi à écrire ces tests permet de réaliser un développement plus sûr. Le développeur sait à tout moment si une modification a altéré le fonctionnement du système.

- **Conception simple**

La conception dans XP doit être la plus simple possible. Cela implique que tous les tests passent avec succès, que le code n'est pas dupliqué, que les développeurs tentent de créer le moins de méthodes et de classes possibles et que le code soit le plus clair possible.

- **Réusinage du code**

Au fur et à mesure que le code évolue, il faut consacrer du temps à remanier le code afin de conserver une bonne conception. Les tests unitaires permettent de réaliser un remaniement plus sécuritaire du code.

- **Appropriation collective du code**

Tout développeur devrait être en mesure de retoucher n'importe quelle partie du code. Le code appartient à tout le monde. Ainsi, le savoir est partagé entre les membres de l'équipe et les segments de codes plus complexes sont généralement simplifiés. Encore ici, les tests unitaires sont importants pour préserver l'intégrité du système.

- **Convention de nommage**

L'équipe de développement doit définir un standard dans la façon de programmer et utiliser une politique de nommage commune. Ainsi, l'équipe a plus de facilité à retravailler les sections développées par leurs collègues. Cela facilite également le remaniement de code.

- **Programmation en binôme**

Tout développement se fait en équipe de deux. Une personne rédige le code pendant que l'autre apporte des suggestions et fait une revue par les pairs. Les rôles sont régulièrement inversés. Cette technique améliore la conception et le savoir est mieux partagé.

La Figure 1.17 illustre le cycle de développement de XP. Les développeurs explicitent avec le client des histoires (*User Stories*) qui déterminent les exigences du projet. Au même moment est élaborée une architecture initiale (*Architectural Spike*). Le regroupement et la priorisation de ces histoires permettent la planification des versions (*Release planning*) et la planification des itérations. Les itérations sont développées et sont validées par le client avec les tests d'acceptation.

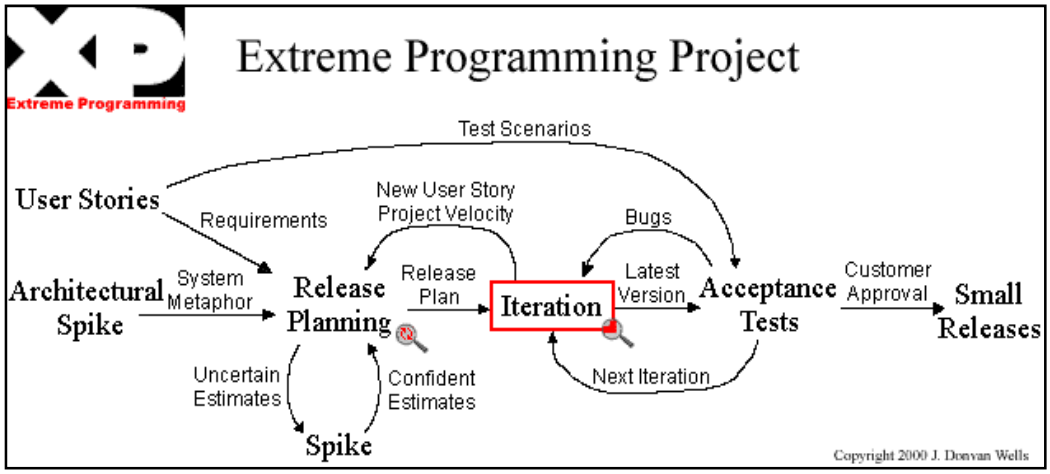


Figure 1.17 Le cycle de XP.
Tirée de (Wells, 2006, 17 février).

1.8.4 OpenUP

OpenUP est une version allégée et agile du processus unifié (*Unified Process* ou UP) d'IBM. OpenUP adopte une approche itérative et incrémentale à l'intérieur d'un cycle de vie structuré. OpenUP se concentre sur la nature collaborative du développement logiciel. (Eclipse, 2008).

OpenUP est une méthodologie basée sur RUP qui est très connue. La terminologie ainsi que la méthodologie sont les mêmes. Comme l'illustre la Figure 1.18, le cycle de développement d'OpenUP se décompose en 4 grandes phases (*OpenUP*, 2010):

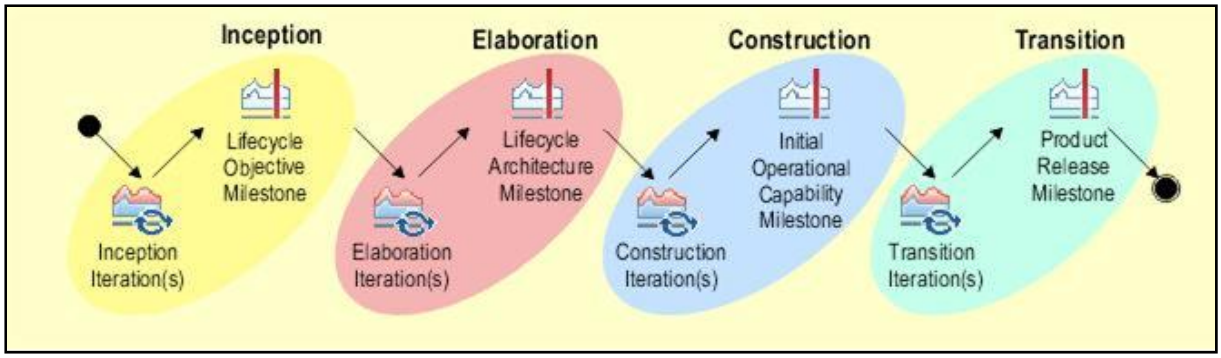


Figure 1.18 Les 4 phases d'OpenUP.
Tirée de (Eclipse, 2008)

L'initialisation

Dans cette phase on évalue la faisabilité du projet. On identifie les bénéfices du projet et son retour sur l'investissement. On identifie les principales fonctionnalités du système et on s'accorde sur la solution. On évalue sommairement les coûts, la durée et les risques du projet.

L'élaboration

Dans cette phase, on s'assure de mieux comprendre les exigences du système et on les documente. On développe l'architecture du système. On évalue les risques les plus importants afin de contrôler leurs impacts. On conçoit les composantes logicielles. On commence les premières itérations de développement.

La construction

Dans cette phase, on développe les éléments restants. Normalement, l'architecture est stable et les risques les plus importants ont été mitigés. On est en mesure d'évaluer avec plus de précision les délais et les coûts. La solution est suffisamment avancée pour permettre le déploiement de versions bêta.

La transition

Dans cette phase, on réalise le déploiement final. Cela inclut de petits ajustements afin d'améliorer la performance, corriger les dernières erreurs et améliorer la qualité de la solution dans son ensemble.

1.9 Eclipse Process Framework Composer

L'« *Eclipse Process Framework Composer* » (EPF) est un logiciel permettant de faire l'ingénierie des processus. C'est un logiciel libre. EPF permet d'élaborer des librairies de processus qui peuvent ensuite être publiées sous la forme d'un site Web.

EPF regroupe un cadriciel de développement dans un objet plugiciel. Un plugiciel contient deux dossiers, le dossier « *Method Content »* et le dossier « *Processes »*. Le dossier « *Method Content »* regroupe tous les éléments du cadriciel de développement, comme les produits

livrables, les tâches, les rôles, etc. L'objet « *Processes* » sépare les processus en deux sous-répertoires :

- Le répertoire « *Capability patterns* » dans lequel on retrouve des patrons de processus qui sont réutilisables. Par exemple, la réalisation des tests unitaires qui peut-être réalisé dans différentes étapes du développement.
- Le répertoire « *Delivery processes* » dans lequel on retrouve la définition du cycle de vie complet, incluant les différentes phases, itérations et activités.

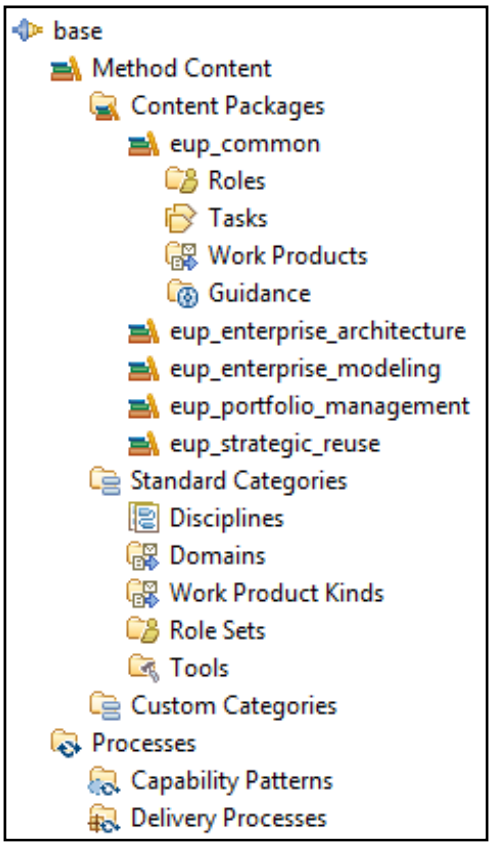


Figure 1.19 L'arborescence détaillée dans EPF.

Voici une description des éléments importants, illustrés dans la Figure 1.19, d'une méthode dans EPF (Eclipse, 2008) :

Rôle (*Role*)

Définit qui fait le travail. Le rôle définit le comportement et les responsabilités d'un individu. Dans le cas d'une petite organisation, il ne sera pas rare de voir une même personne prendre plusieurs rôles à la fois, tout comme il est possible de changer de rôle tout au long du projet.

Tâche (*Task*)

Définit comment réaliser le travail. Une tâche est un travail réalisé par une personne détenant un rôle. Une tâche peut se décomposer en plusieurs étapes qui peuvent inclure la création ou la mise à jour de plusieurs artéfacts.

Document (*Work product*)

Cet artéfact représente ce qui est produit. Il spécifie quel est l'intrant ou l'extrant d'une tâche. Cela peut-être un document, du code source, des cas d'utilisation ou un modèle quelconque. On nomme souvent ces objets des artéfacts. Dans ce mémoire, les documents de travail seront utiles pour lier les disciplines d'entreprise avec le cycle de développement d'une SOA.

Conseil (*Guidance*)

Les conseils sont des exemples, des listes de vérification, des directives et tout autre supplément permettant de faciliter la compréhension de la méthode. Ces conseils sont liés à des éléments spécifiques. OpenUP possède une vaste librairie de conseils.

Disciplines

Les disciplines sont des catégories auxquelles les tâches peuvent être associées. Cela peut faciliter la compréhension de regrouper les tâches associées à l'explicitation des exigences ou à la définition de l'architecture, par exemple, dans une même discipline.

Domaines (*Domains*)

Les domaines sont des catégories auxquelles les artéfacts peuvent être associés. Cela peut faciliter la compréhension de regrouper tous les artéfacts associés à l'explicitation des exigences ou encore à la définition de l'architecture dans un même domaine.

Type de document de travail (*Work Product Kind*)

Une autre catégorie auxquelles les artéfacts peuvent êtres associés. Cette fois, c'est le type de document qu'on souhaite spécifier.

Groupe de rôles (*Role sets*)

Les groupes de rôles permettent de regrouper des rôles ayant des points en communs. Par exemple, un groupe de rôles « Analyste » qui regrouperait les rôles d'analystes fonctionnels et d'analystes d'affaires.

Patron de processus (*Capability Patterns*)

Ce sont des patrons de processus particuliers qui sont réutilisables. Ces blocs contiennent généralement un ensemble d'activités qui sont associées à une discipline en particulier. Ce sont des blocs d'activités qui permettront de construire le processus de livraison.

Processus de livraison (*Delivery Process*)

Définit la répartition du travail et le flux de travail (*workflow*) pour une phase ou un cycle de vie complet. Un processus de livraison regroupe en un tout cohérent les artéfacts, les tâches et les rôles afin de créer une séquence d'actions amenant à la réalisation du projet.

1.10 Conclusion

Le premier objectif de la revue de la littérature est de réaliser l'état de l'art sur les cadres d'architecture d'entreprise, sur les méthodologies SOA et sur les méthodologies de développement agile. On explique les différents types de processus d'affaires qui sont susceptibles d'êtres automatisés dans une SOA et on fait référence à l'outil de modélisation des processus qui sera utilisé dans ce mémoire.

Dans une architecture orientée services, l'organisation modélise et développe des services afin de réaliser sa mission. Ces services permettent de réaliser certains processus internes (orchestration) ou certains processus sollicitant la collaboration avec une entité externe (chorégraphie). La Figure 1.20 illustre cette architecture entre deux organisations. Ces services sont suffisamment génériques pour permettre une forte réutilisation et ils seront utilisés pour réaliser plusieurs processus d'affaires distincts.

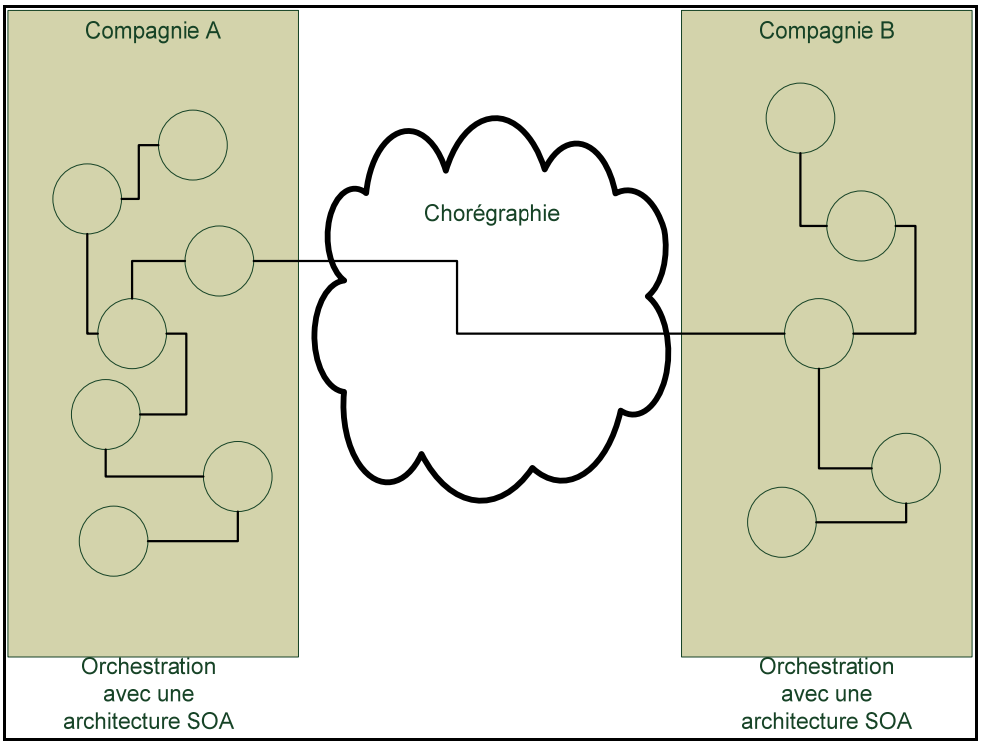


Figure 1.20 Architecture SOA et processus d'affaires.

Les plus importants standards dans l'utilisation des services Web sont le langage XML, les fichiers WSDL pour la définition de l'interface et le format SOAP pour les échanges d'informations. Le langage XML est utilisé autant pour l'envoi des messages que pour la définition des contrats ou la découverte de ces services à l'aide d'un annuaire UDDI. La figure Figure 1.21 nous rappelle que XML est utilisé dans tous les aspects de l'utilisation d'un service Web. Ces standards seront utilisés dans ce mémoire lors des expérimentations de la méthodologie.

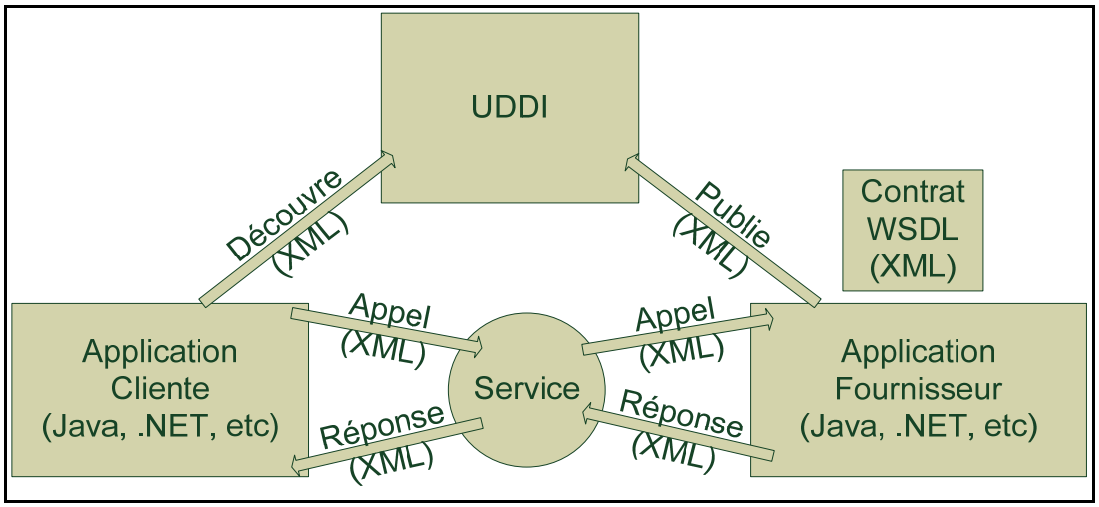


Figure 1.21 Les services Web.

Plusieurs organismes participent à la définition des différentes normes entourant les services Web qui permettent la mise en place d'une SOA. Les principaux organismes sont le W3C, l'OASIS et le WS-I. Le respect des profils proposés par le WS-I permet de garantir l'interopérabilité des services entre les différentes plateformes. Le Tableau 1.2 résume les contributions, non exhaustives, de chaque organisation. Pour ce mémoire, le profil de base défini par le WS-I sera utilisé lors de la création des services Web.

Tableau 1.2 Les organisations et leurs contributions

Organisations	Contributions
WS-I	Profil BASIC, Security, Attachement
OASIS	UDDI, BPEL, certains standards WS-*
W3C	XML, XSD, WSDL, certains standards WS-*

Le second objectif de la revue de littérature est de confirmer qu'aucune des méthodologies n'intègre dans une seule méthode ouverte l'architecture d'entreprise, les activités de développement d'une SOA et les phases de développement logiciel. On constate dans le Tableau 1.3 qu'aucune des méthodologies ne répond à ces critères. Il y a généralement absence des activités d'entreprise nécessaires à la vue d'ensemble de l'organisation, de ces processus et de ses systèmes. De plus, la plupart des méthodologies orientées services sont propriétaires.

Tableau 1.3 Les contributions de chaque méthodologie

Méthodologie	Ouverte/Disponible?	Entreprise?	Orientée services?	Développement?
Cadres d'entreprise				
TOGAF	Oui	Oui	Non	Non
Zachman	Non	Oui	Non	Oui
EUP	Oui	Oui	Non	Non
Méthodologies de développement				
RUP	Non	Non	Non	Oui
OpenUP	Oui	Non	Non	Oui
XP	Oui	Non	Non	Oui
SCRUM	Oui	Non	Non	Oui

Méthodologie	Ouverte/Disponible?	Entreprise?	Orientée services?	Développement?
Méthodologies SOA				
Erl	Oui	Incomplet	Oui	Incomplet
SOMA	Non	Non	Oui	Oui
SOAD	Non	Non	Oui	Oui
SOA RQ	Non	Non	Oui	Oui
SOAF	Non	Incomplet	Oui	Oui

Dans le chapitre qui suit, on identifie les méthodologies candidates et on explique l'approche qui permettra de les intégrer en une seule méthode.

CHAPITRE 2

APPROCHE POUR LA RÉALISATION DE LA MÉTHODE

2.1 Introduction

Il est nécessaire d'établir une approche adaptée aux particularités de ce projet de maîtrise. L'objectif de ce chapitre est d'expliquer l'approche utilisée pour l'identification et l'intégration des trois méthodologies qui seront retenues à la suite de la revue de la littérature.

Il faut rappeler que l'objectif de ce mémoire n'est pas de démontrer l'efficacité d'une méthode de développement intégrant l'architecture d'entreprise, les pratiques orientées services et le développement agile. Le nombre d'échantillons et le temps nécessaire pour tirer des conclusions allant bien au-delà de la portée de ce mémoire.

La revue de littérature nous a permis de constater qu'il y a un réel problème d'intégration entre l'architecture d'entreprise, les pratiques orientées services et le développement logiciel. On définit dans les prochaines sections comment on crée et intègre une nouvelle méthode qui réaliserait cette intégration.

2.2 Approche

2.2.1 Identification des trois méthodologies

La nouvelle méthode doit intégrer un cadre d'architecture d'entreprise ouvert. Ce cadre d'entreprise est nécessaire afin d'avoir une vue globale de l'organisation, qui est un élément essentiel dans la réalisation d'une SOA. Elle doit également intégrer les activités de développement d'une SOA. Ces activités utilisent d'une part les artéfacts créés dans les activités d'entreprise. Une SOA se concrétise à travers plusieurs projets à l'aide des phases

d'une méthodologie de développement logicielle. Le choix des méthodologies candidates se fera donc sur des critères de complémentarité, d'ouverture, d'agilité et sur l'existence d'un corpus significatif.

2.2.2 Intégration des méthodologies candidates

La seconde étape consiste à intégrer les trois méthodologies choisies en une seule méthode cohérente. Cette étape se divise elle-même en deux phases.

- La première phase d'intégration se base sur une analyse individuelle des trois méthodologies sélectionnées. On réalise ensuite une analyse comparative entre la méthodologie SOA et le cadre d'architecture d'entreprise choisie afin d'identifier les liens qui existent entre les disciplines d'entreprise et le développement SOA. Cette analyse comparative se poursuit avec les activités de réalisation d'une SOA et le cycle de développement logiciel. Tout au long de cette analyse, on documente la nouvelle méthode. Cette analyse est réalisée dans le prochain chapitre.

- La seconde phase d'intégration consiste à tester la méthode et à identifier les artéfacts qui sont susceptibles d'êtres communs aux méthodes. Deux exemples réalisés par l'expérimentateur permettent d'identifier les activités, les disciplines et les artéfacts qui ont été utilisés. Cette expérience est détaillée dans le chapitre 4.

2.2.3 Documentation des méthodologies candidates

La troisième étape est de documenter les méthodologies dans un outil de modélisation des processus. Le logiciel EPF a été retenu pour ce travail. C'est un logiciel gratuit et libre.

La documentation des méthodologies candidates et de la méthode intégrée ne sont pas réalisées à un moment précis dans le mémoire, et il n'y a pas de chapitre dédié à ce travail.

La documentation de la méthode est réalisée à la suite de plusieurs itérations successives. Les premières itérations permettent surtout d'ajouter les disciplines d'architecture d'entreprise et les activités orientées services, qui ne sont pas documentées dans EPF. Les itérations subséquentes sont surtout dédiées à l'intégration des disciplines et des activités au cycle de développement logiciel.

2.3 Conclusion

Ce chapitre explique l'approche utilisée pour réaliser ce mémoire. À la suite de la revue de la littérature, il a été conclu qu'il n'existe aucune méthodologie ouverte et agile intégrant l'architecture d'entreprise, les pratiques orientées services et le développement agile. Parmi toutes les méthodologies étudiées dans le chapitre précédent, trois seront retenues pour bâtir une méthode qui répond à ce critère. Dans le chapitre qui suit, on sélectionne et on analyse en détail ces trois méthodologies. Ceci est fait individuellement dans un premier temps, puis on réalise une analyse comparative afin d'identifier les activités et les artéfacts susceptibles d'être interreliés. Finalement, la nouvelle méthode est documentée dans EPF.

L'approche choisie a ses limites et ses contraintes. Il est nécessaire d'évaluer sa validité, sa fiabilité et sa possible généralisation. Ces éléments seront abordés dans la discussion ainsi que dans les recommandations qui suivent l'expérimentation.

CHAPITRE 3

RÉALISATION DE LA MÉTHODE

3.1 Introduction

Ce chapitre décrit la réalisation de la méthode selon l'approche décrite dans le chapitre précédent. Nous allons donc identifier, documenter et modéliser la nouvelle méthode en intégrant les activités d'architecture d'entreprise, les phases de développement et les activités propres aux architectures orientées services.

Dans la section 3.3, nous analysons en détail les activités d'architecture d'entreprise du cadre sélectionné. Dans la section 3.4, nous analysons en détail les activités propres à la réalisation d'une SOA. Dans la section 3.5, nous analysons en détail les phases qui composent le cycle de développement logiciel retenu. Finalement, dans la section 3.6, nous réalisons une analyse comparative entre les activités d'architecture d'entreprise et les activités orientées services, ainsi qu'une analyse comparative entre ces mêmes activités orientées services et les phases d'initialisation et d'élaboration d'OpenUP.

3.2 Identification des méthodologies candidates

Cadre d'architecture d'entreprise

Pour les activités d'entreprise, le cadre de Zachman ne peut être retenu. La première raison est que ce cadre n'est pas ouvert, mais également parce que le cadre de Zachman n'est pas une méthodologie, mais plutôt une taxonomie des différents artéfacts. TOGAF et EUP sont deux cadres très intéressants qui proposent un processus. TOGAF est reconnu, contrairement à EUP, et il est déjà documenté dans un outil de modélisation des processus. Mais c'est un cadre plus lourd et complexe, alors que l'objectif est d'obtenir un outil adapté aux plus petites organisations. Le processus unifié d'entreprise de (Scott W.Ambler, 2005) propose

une extension au processus unifié, mieux connu sous le nom de RUP (pour *Rational Unified Process*) (IBM, 2007), elle-même dérivée de UP (pour *Unified Process*). EUP offre un ensemble de disciplines d'entreprise qui précède et suit les phases d'un projet. C'est un ensemble d'activités se déroulant en continu et fournissant les informations de l'entreprise aux projets de développement. Ces disciplines sont utilisées afin de construire le cycle continu d'entreprise. EUP partage une même terminologie et une même représentation graphique qu'avec OpenUP, ce qui le rend particulièrement intéressant. EUP offre plusieurs disciplines susceptibles d'aider le développement des services, par exemple la discipline de réutilisation. On peut présumer que si EUP peut étendre les disciplines de RUP, il le peut également pour OpenUP, étant lui-même un dérivé de RUP. Mais EUP n'est pas documenté dans EPF. Il faut donc documenter EUP dans EPF si nous décidons de choisir ce cadre.

Les trois premières disciplines sont des disciplines de haut niveau et de vision alors que la discipline de la réutilisation stratégique est dans une classe à part. Ce découpage est justifié, car les premières disciplines permettent de documenter et modéliser les actifs informationnels de l'entreprise alors que la réutilisation stratégique permet d'identifier les éléments réutilisables de l'entreprise et est le déclencheur des phases d'un projet de développement.

Les trois dernières disciplines d'EUP, la gestion du personnel, l'administration de l'entreprise et l'amélioration du processus logiciel ont été volontairement exclues de ce projet de recherche. Ces disciplines portent davantage sur le support aux opérations. Cela va au-delà de la portée de ce mémoire qui se concentre sur les disciplines ayant une valeur ajoutée pour la réalisation des projets de développement SOA.

Méthodologie de développement logiciel

OpenUP ayant déjà été modélisé à l'intérieur de l'EPF, cela simplifie considérablement le travail nécessaire à sa publication et à sa diffusion. De plus, OpenUP est constitué d'un corpus considérable de fichiers d'aide, d'exemples, de gabarits, de guides et de listes de vérification qui peuvent faciliter sa compréhension et son adoption en entreprise. On ne retrouve pas d'équivalent aussi documenté dans les autres méthodologies modélisées dans EPF, comme Scrum et XP. Pour ces raisons, OpenUP est utilisé comme cadriciel principal de développement pour ce mémoire. OpenUP est suffisamment large pour permettre d'intégrer à l'intérieur de ses phases les activités particulières à la modélisation et à l'implantation d'une architecture SOA. En somme, OpenUP est utilisé comme méthodologie de développement logiciel.

Méthodologie SOA

Pour les méthodologies orientées services, aucune sauf celle de Thomas Erl (Erl, 2005) n'est ouverte. Les méthodologies orientées services sont propriétaires et la documentation n'est pas disponible. Bien que toutes les méthodologies orientées services semblent aborder les phases de développement, il est impossible de vérifier si tel est le cas. La méthodologie de Thomas Erl possède la documentation la plus exhaustive qui a été trouvée à ce jour. C'est pour ces raisons que cette méthodologie a été retenue pour la réalisation de ce mémoire. Néanmoins, les phases de développements abordés dans la méthodologie sont très sommaires et incomplètes. De plus, la méthodologie n'est pas documentée dans un outil de modélisation des processus comme EPF. Les spécificités de l'analyse et de la conception d'une SOA seront greffées à OpenUP. La méthodologie présentée par (Erl, 2005) définissant l'analyse et la conception d'un service sera intégrée à OpenUP dans les phases d'initialisation et d'élaboration, puisque les phases de construction et de transition sont semblables.

Le Tableau 3.1 illustre comment les trois méthodologies vont s'intégrer les unes aux autres. Premier constat, elles sont toutes ouvertes et disponibles. Une documentation importante existe et peut être utilisée pour chacune d'elles. EUP aborde l'architecture d'entreprise alors que la méthodologie de Thomas Erl l'aborde partiellement et qu'OpenUP ne l'aborde pas du tout. La méthodologie de Thomas Erl est la seule à aborder spécifiquement le développement d'une SOA à l'intérieur d'un cycle de développement, qui est très peu élaboré. Ce cycle de développement est entièrement défini dans OpenUP

Tableau 3.1 Les contributions des trois méthodologies retenues

Méthodologie	Ouvert?	Entreprise?	Orientée services?	Développement?
EUP	Oui	Oui	Non	Non
Erl	Oui	Incomplet	Oui	Incomplet
OpenUP	Oui	Non	Non	Oui

3.3 Description des disciplines d'entreprise

Tel que mentionné dans la section 3.2, quatre des sept disciplines d'entreprise selon (Scott W.Ambler, 2005) ont été retenues. La Figure 3.1 présente les quatre disciplines d'entreprise et leurs activités respectives. Les prochaines sections décrivent ces disciplines et leurs activités respectives en détail. L'ensemble de ces quatre disciplines, leurs activités et leurs étapes respectives, les rôles associés ainsi que les artéfacts ont dû être modélisés dans EPF.

Presentation Name	Index	Predecessors
EUP Lifecycle	0	
Enterprise Phase	1	
Enterprise Iteration [1..n]	2	
Enterprise modeling	3	
Define enterprise strategy	*4*	
Model business process	*10*	*4*
Identify process implementation options	*15*	*10*
Model the domain	*18*	*4*
Model the organization	*22*	*4*
Support project teams	*27*	*4*
Portfolio management	32	59
Inventory current systems	*33*	
Plan portfolio and programs	*35*	*34*
Manage portfolio	*43*	*35*
Manage programs	*46*	*35*
Manage contracts	*50*	*35*
Manage enterprise risk	*55*	*35*
Enterprise architecture	59	3
Define architectural requirements	*60*	
Define candidate architecture	*64*	
Refine enterprise architecture	*69*	*64*
Define reference architecture	*74*	
Support the project team	*78*	
Strategic reuse	82	32
Plan reuse program	*83*	
Harvest existing asset	*91*	
Obtain External asset	*97*	
Develop asset	*103*	
Publish asset	*108*	*91,97,103*
Support project teams	*111*	*83*
Lifecycle Objectives Milestone	116	2

Figure 3.1 Les disciplines d'entreprise dans EPF.
Adaptée de (Scott W.Ambler, 2005)

3.3.1 La discipline de modélisation d'affaires

L'objectif de la modélisation d'affaires est de documenter la vision, les buts et les objectifs ainsi que la stratégie pour réaliser la mission de l'entreprise. C'est dans cette discipline que sont identifiés les principaux processus d'affaires et que le modèle du domaine et le modèle de l'organisation sont documentés. Les extrants de cette discipline seront particulièrement intéressants pour la gestion du portefeuille et la priorisation des projets. La modélisation des processus d'affaires permet d'identifier les actifs nécessaires à la réutilisation stratégique. La Figure 3.2 illustre les activités de cette discipline qui ont été documentés dans EPF.

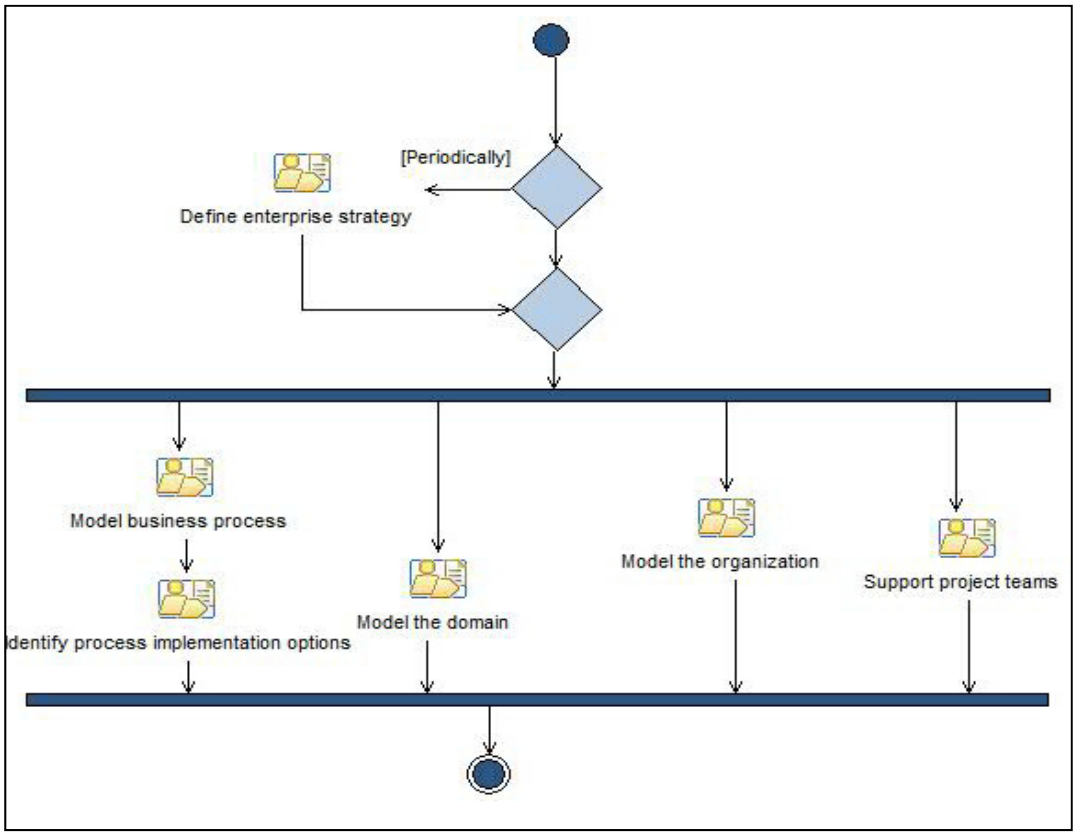


Figure 3.2 Les activités de la modélisation d'affaires de l'entreprise.
Adaptée de (Scott W.Ambler, 2005)

- **Définir une stratégie d'entreprise**

La définition de la stratégie d'entreprise regroupe un ensemble de tâches permettant d'identifier et de documenter la mission de l'entreprise. Cela inclut les objectifs et les buts à court et à long terme ainsi que la stratégie globale qui sera appliquée pour atteindre ces objectifs. On met en évidence l'écart qui existe entre la situation présente et la vision de ce que l'on veut réaliser. On retrouve fréquemment cette information dans le plan directeur de l'organisation, lorsqu'il existe.

La définition de la stratégie d'entreprise permet une meilleure priorisation des projets, en fonction des objectifs de l'entreprise. Les activités subséquentes de modélisation seront également influencées par la stratégie d'entreprise. Les principaux livrables sont la vision, les objectifs et la mission de l'entreprise.

- **Modéliser les processus d'affaires**

La modélisation des processus d'affaires de l'entreprise consiste à documenter l'environnement de l'entreprise, les processus d'affaires et les règles d'affaires. On identifie dans l'environnement de l'entreprise les clients et leurs besoins. On identifie également les fournisseurs et les partenaires de l'organisation.

Les processus d'affaires sont les différentes activités réalisées au sein d'une entreprise afin de produire un service ou un produit ayant de la valeur pour un client (Le grand dictionnaire terminologique, 2010). La modélisation de ces processus permet d'identifier les liens entre chaque activité et d'identifier les rôles qui sont sollicités dans leur réalisation. On peut également documenter les principaux intrants et extrants de chaque activité à l'intérieur d'un processus.

(Scott W.Ambler, 2005) insiste sur l'importance de ne pas trop modéliser en détail cette étape. En effet, les détails dans la réalisation d'un processus d'affaires sont susceptibles de changer, rendant le modèle rapidement imprécis ou désuet. Il faut identifier des règles

d'affaires de haut niveau qui sont susceptibles de rester inchangées pour de longues périodes de temps. Les principaux livrables sont un modèle des processus d'affaires de l'entreprise et une spécification des règles d'affaires de l'entreprise.

- **Identifier les options d'implémentation de processus**

À partir du modèle des processus d'affaires ainsi que du modèle d'architecture de l'entreprise, il est possible d'identifier les options d'implémentation de processus. Cette étape est particulièrement importante, car c'est ici qu'on évalue ce qui doit être fait manuellement et ce qui peut être automatisé.

(Scott W.Ambler, 2005) met l'emphase sur la nécessité de considérer autant les automatisations possibles que le travail manuel réalisé par des individus. Il n'est pas toujours intéressant (ou justifiable) de vouloir tout automatiser.

L'objectif à cette étape est de fournir un maximum d'information aux individus pour faciliter la réalisation de leurs tâches manuelles. Ensuite, utiliser les systèmes d'informations pour réaliser les tâches répétitives. Finalement, évaluer la rentabilité d'une automatisation à l'intérieur d'un laps de temps assez court. On suggère deux ans maximum pour le retour sur l'investissement (ROI) d'un projet d'automatisation. Ce court délai est justifié par le changement qu'un environnement d'affaires est susceptible de vivre, rendant un long projet d'automatisation plus difficile à rentabiliser.

- **Modéliser le domaine**

La modélisation du domaine de l'entreprise permet de documenter les principales entités d'affaires de l'organisation et leurs interrelations. Le modèle du domaine est souvent complémentaire au glossaire, car il associe les éléments entre eux. Ces associations sont des sources d'informations importantes dans la compréhension du domaine d'affaires.

L'un des objectifs de cette activité est d'éviter que les éléments communs soient modélisés à chaque nouveau développement. Au lieu de cela, ils sont réutilisés de projet en projet. Lors de la réalisation d'un projet de développement, l'équipe peut concentrer ses efforts sur les éléments qui sont uniques au domaine du projet.

Un autre objectif important est la réalisation d'un glossaire pour l'entreprise. Il n'est pas rare de voir dans les organisations plusieurs termes pour désigner le même document ou le même concept. Ce sont autant de possibilités de confusion ou d'ambiguïté. Les deux principaux livrables sont un glossaire d'affaires de l'entreprise ainsi qu'un modèle du domaine de l'entreprise.

- **Modéliser l'organisation**

La modélisation de l'organisation permet de documenter les principales structures administratives de l'entreprise. On modélise les différents départements et si nécessaire, on indique les relations qui existent entre eux. On peut ensuite aligner le modèle de l'organisation avec le modèle des processus d'affaires.

La modélisation de l'organisation peut comprendre également la modélisation du territoire desservi par l'entreprise, qu'il soit physique (géographique) ou virtuel (Internet). Le principal livrable est le modèle organisationnel de l'entreprise.

- **Supporter l'équipe de projet**

Cette activité est critique dans l'utilisation des différents artéfacts réalisés pendant la modélisation d'affaires de l'entreprise. On supporte les équipes de projet afin qu'elles connaissent l'existence de ces artéfacts ainsi que leur utilité.

Afin d'atteindre cet objectif, les modeleurs qui ont participé à la réalisation de ces artéfacts seront souvent intégrés aux équipes de projet afin d'intégrer leurs travaux et donner la formation nécessaire aux autres membres de l'équipe.

Le

Tableau 3.2 donne les principaux livrables de la discipline de modélisation d'affaires.

Tableau 3.2 Les principaux artéfacts de la discipline de modélisation d'affaires
Adapté de (Scott W.Ambler, 2005)

Artéfacts	Description
Glossaire	Le glossaire contient une définition de la terminologie d'affaires de l'entreprise. Ce glossaire permet l'utilisation d'un même vocabulaire par tous les départements de l'organisation. Ce glossaire est fréquemment mis à jour.
Modèle des processus d'affaires	Le modèle des processus d'affaires contient la définition des principaux processus d'affaires de l'organisation. Ce modèle contient les activités, les participants, les intrants et les extrants.
Modèle du domaine	Le modèle du domaine vient compléter le glossaire en documentant les liens unissant les principaux concepts d'affaires de l'organisation. Ce modèle est souvent représenté comme un diagramme de classe simplifié.
La vision, la mission et les objectifs de l'entreprise	Ces informations se trouvent généralement dans le plan directeur de l'entreprise. La vision indique où l'entreprise souhaite se positionner au cours des prochaines années alors que les objectifs détaillent la façon de réaliser la vision. La mission définit sa raison d'être.

3.3.2 La discipline de gestion du portefeuille

La gestion du portefeuille de projets est une discipline qui assure le suivi des systèmes pendant leur cycle de vie, de la création au retrait. On regroupe les projets de domaines connexes dans des programmes. Il y a dans cette discipline des activités de gestion des contrats avec les fournisseurs externes. On y trouve également des activités de gestion du risque au niveau de l'entreprise, du portefeuille et des programmes. Les listes des projets et

des programmes seront réutilisées afin de documenter l'architecture de l'entreprise. La Figure 3.3 illustre les activités de cette discipline qui ont été documentés dans EPF.

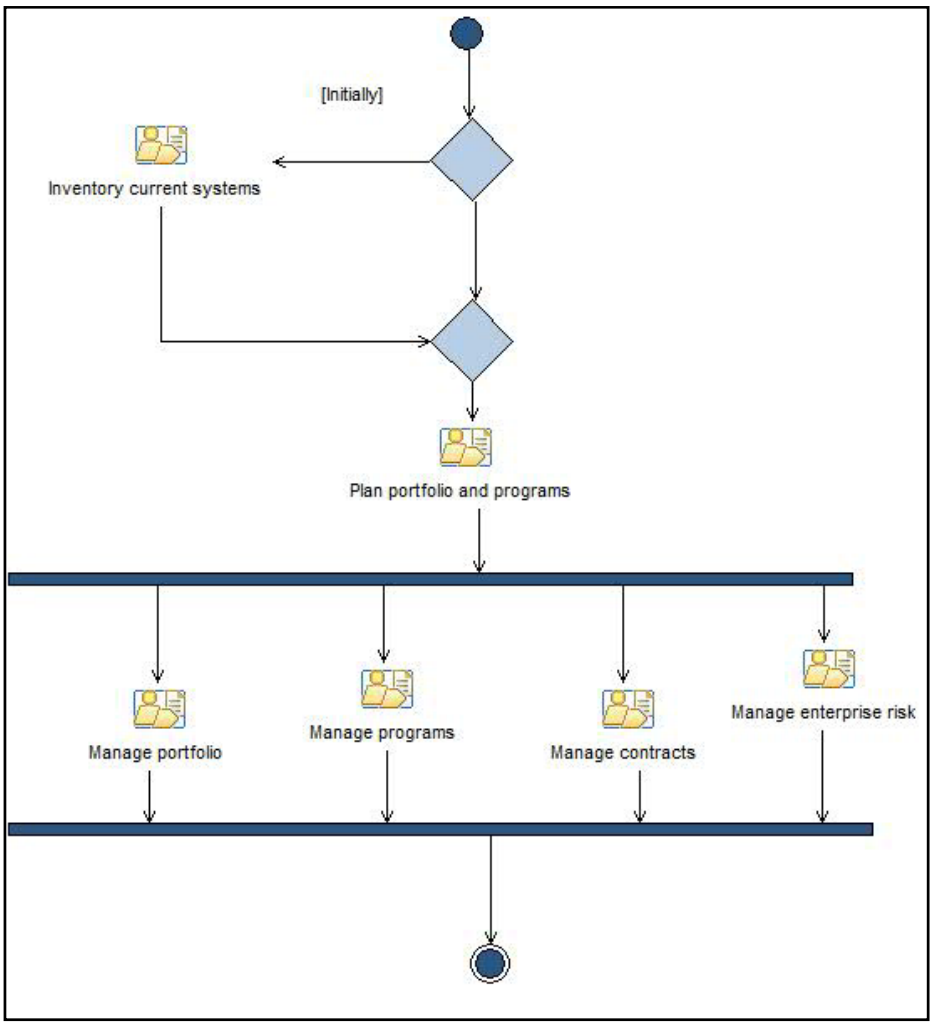


Figure 3.3 Les activités de la gestion du portefeuille de projets.
Adaptée de (Scott W.Ambler, 2005)

- **Inventorier les systèmes actuels**

L'inventaire des systèmes actuels permet d'identifier ce que l'entreprise possède déjà comme systèmes d'information. Cet exercice sera particulièrement utile pour identifier les logiciels utilisés dans plusieurs directions ou départements. L'inventaire permet d'évaluer avec une

meilleure précision les besoins de l'entreprise en nouveaux systèmes. Le principal livrable est un inventaire des systèmes actuels de l'entreprise.

- **Planifier le portefeuille et les programmes**

La planification du portefeuille et des programmes permet de prévoir les travaux qui seront à faire pour une période de temps plus ou moins longue, généralement un trimestre ou une année. Les programmes regroupent des projets ayant la même portée ou visant un même objectif d'affaires.

La planification du portefeuille est réalisée à l'aide de l'inventaire des systèmes actuels, mais également à l'aide de la mission et de la vision de l'entreprise ainsi qu'avec le modèle d'architecture de l'entreprise. À l'aide de ces artéfacts, le responsable sera en mesure d'évaluer et de prioriser les projets à venir.

Les dépendances entre les projets sont identifiées. Ces dépendances peuvent être des dates de livraison, des livrables pour l'atteinte d'un objectif architectural, le remplacement ou la mise à jour d'un système. Les principaux livrables sont le portefeuille et la liste des programmes. Il y a également la planification du portefeuille et des programmes.

- **Gérer le portefeuille**

Cette activité correspond à l'exécution du plan. Tout projet qui ne se retrouve pas dans un programme spécifique peut être géré à partir de cette activité. La gestion du portefeuille permet, par exemple, d'identifier des métriques de performances par projet pour ensuite les rapporter sur l'ensemble du portefeuille.

C'est à cette étape que la première version du document de vision d'un projet est rédigée. Cette activité précède l'initialisation du projet, tel que vu par RUP ou, par extension, par OpenUP.

- **Gérer les programmes**

C'est une activité similaire à la gestion du portefeuille. La gestion des programmes permet de contrôler les différents projets d'un programme et de reporter ces mesures au programme lui-même. L'objectif de cette activité est de s'assurer que les projets sont toujours alignés avec leur programme et d'apporter des corrections si un programme connait des problèmes.

- **Gérer les contrats**

La gestion des contrats permet d'identifier les fournisseurs externes qui réaliseront certaines tâches pour des projets de l'entreprise. La gestion des contrats définit les principales étapes menant à l'attribution d'un contrat, en passant par l'identification des besoins et l'identification des différents fournisseurs. La gestion des contrats s'assure aussi de contrôler le travail des fournisseurs.

- **Gérer les risques au niveau de l'entreprise**

La gestion des risques permet d'identifier les risques qui pourraient avoir un impact négatif sur les objectifs d'affaires ou la mission de l'organisation. Chaque projet a généralement des éléments plus ou moins risqués. L'agrégation de tous les problèmes au niveau du portefeuille amène une perspective plus globale de ces risques.

Par exemple, est-ce que l'utilisation d'une même plateforme technologique pour l'ensemble des projets place l'entreprise dans une situation risquée? Est-ce que le départ à la retraite de plusieurs ressources met en péril certains programmes? Est-ce qu'il y a des dépendances entre certains projets et/ou certaines ressources?

Il y a généralement plusieurs façons de traiter un risque. On peut éliminer le risque, par exemple en ne développant pas un élément jugé trop risqué. On peut minimiser le risque en mettant en place un plan de contingence. On peut également transférer le risque en utilisant l'impartition. On peut finalement ignorer le risque et espérer qu'il ne se matérialise pas. Les principaux livrables de cette activité sont une liste des risques d'entreprise et un plan de

gestion des risques d'entreprise. Le Tableau 3.3 résume les principaux livrables de la discipline de gestion du portefeuille.

Tableau 3.3 Les principaux artéfacts de la discipline de gestion du portefeuille
Adapté de (Scott W.Ambler, 2005)

Artéfacts	Description
Liste des risques d'entreprise	Les risques d'entreprises correspondent aux risques de chaque projet ramenés à une perspective plus globale. Cette liste inclut également les risques liés aux dépendances entre les projets ou des programmes ou encore à tout autre risque qui peut avoir un impact sur plusieurs projets.
Portefeuille	Le portefeuille contient la liste des projets en cours ainsi que la liste des systèmes déployés dans l'entreprise.
Planification du portefeuille	La planification du portefeuille contient le plan de réalisation des projets et des programmes. Analogue à un plan de projet, mais pour l'ensemble du portefeuille.

3.3.3 La discipline d'architecture d'entreprise

Cette activité modélise et documente l'architecture de l'entreprise. On utilise plusieurs extrants des disciplines précédentes. On aborde l'architecture sous différentes vues, selon ce que l'on souhaite documenter : une vue logique, une vue du matériel, une vue des technologies, des applications, de l'information, etc. Ce travail est réalisé de manière itérative et incrémentale par la définition d'architectures candidates. Ces architectures candidates sont par la suite raffinées au cours du temps. Finalement, des architectures de référence faisant office de patron d'architecture sont produites afin d'accélérer le travail des développeurs. La Figure 3.4 illustre les activités de cette discipline qui ont été documentées dans EPF.

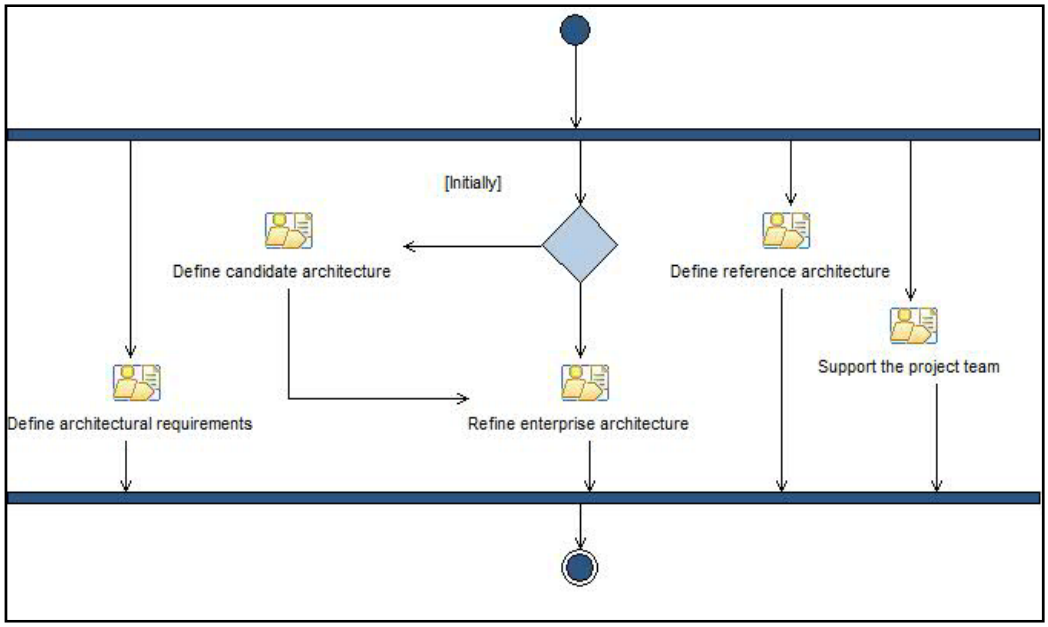


Figure 3.4 Les activités d'architecture d'entreprise.
Adaptée de (Scott W.Ambler, 2005)

- **Définir les exigences architecturales**

Cette activité permet d'identifier les exigences architecturales de l'entreprise. L'architecture d'entreprise dans EUP comprend autant les exigences d'affaires que les exigences applicatives.

À l'aide des artéfacts créés précédemment, on obtient le modèle d'affaires de l'entreprise. Ce modèle contient le modèle des processus d'affaires, le modèle du domaine, le modèle de l'organisation, le glossaire d'affaires de l'entreprise et les règles d'affaires.

Les exigences architecturales vont toucher l'ensemble des applications de l'entreprise, par exemple, l'exigence que toutes les applications journalisent les violations de règles de sécurité. Ou encore que toutes les applications soient contrôlées aux cinq minutes pour vérifier leur opérationnalité.

En croisant le modèle d'affaires de l'entreprise avec le plan du portefeuille, il est possible d'identifier les exigences architecturales d'affaires et techniques. Les principaux livrables sont les exigences techniques de l'entreprise et les exigences architecturales d'affaires de l'entreprise.

- **Définir l'architecture candidate**

La définition de l'architecture candidate permet de documenter l'évolution souhaitée de l'architecture actuelle. Il n'est pas souhaitable d'intégrer toutes les exigences en un seul effort. L'auteur suggère plutôt une progression par étapes successives, limitant ainsi le risque. Par la suite, l'architecture candidate sera mise en place dans le cadre d'un projet ou validée avec le développement d'un système utilisant cette nouvelle architecture. Les principaux livrables sont le plan de l'architecture candidate et une définition des technologies qui seront utilisées.

- **Raffiner l'architecture d'entreprise**

Cette activité permet d'intégrer l'architecture candidate au modèle d'architecture de l'entreprise et de documenter cette dernière. On retrouve dans cette activité plusieurs vues permettant d'illustrer les différents aspects de l'architecture d'entreprise (données, processus, domaine, logiciel, infrastructure, sécurité). C'est un travail itératif et incrémental. (Scott W.Ambler, 2005) a identifié cinq vues particulièrement importantes pour apprécier l'architecture d'une entreprise : la vue logicielle, la vue réseau, la vue flux de travail, la vue du domaine et la vue des données.

La vue logicielle permet de se représenter les applications, les technologies qu'elles utilisent et les liens d'interopérabilité qui les unissent. La vue réseau représente l'infrastructure du réseau ainsi que les principales pièces de matériel comme les serveurs et les routeurs. La vue des flux de travail représente les différents processus d'affaires de l'organisation. La vue par domaine représente les différents aspects de l'organisation. Souvent, des applications importantes seront attachées à un ou plusieurs domaines, comme la gestion de la relation

client ou encore la comptabilité. Finalement, la vue des données décrit l'infrastructure qui supporte l'exploitation des données, comme les entrepôts de données. Le principal livrable est le modèle d'architecture de l'entreprise qui est mis à jour.

- **Définir des référentiels architecturaux**

Les référentiels architecturaux sont des patrons architecturaux qui pourront être réutilisés par les équipes de développement. Par exemple, le patron architectural de l'entreprise pour implémenter la journalisation ou la sécurité. Ces patrons sont généralement accompagnés d'exemples.

- **Supporter les équipes de projet**

La compréhension et l'adoption de l'architecture d'entreprise par les équipes de projets dépendent du support apporté aux équipes de projet. L'architecte d'entreprise doit travailler avec les architectes applicatifs afin de les aider à concevoir leurs applications et à utiliser correctement les référentiels architecturaux. L'adoption et la compréhension de l'architecture de l'entreprise passent par des séances de formation et de mentorat. Le
Tableau 3.4 donne les principaux livrables de la discipline d'architecture d'entreprise.

Tableau 3.4 Les principaux artéfacts de la discipline d'architecture d'entreprise
Adapté de (Scott W.Ambler, 2005)

Artéfacts	Description
Les exigences architecturales d'affaires	Cet artéfact permet d'identifier les exigences d'affaires qui sont susceptibles d'influencer l'architecture. Par exemple, un besoin d'affaires qui justifierait l'achat d'un entrepôt de données.
Les exigences techniques d'entreprise	Les exigences techniques d'entreprise définissent les contraintes de conception, d'un point de vue plus globale. Par exemple, des contraintes forçant la journalisation des activités d'une application dans une autre application dédiée à cette tâche.

Artéfacts	Description
L'architecture candidate	L'architecture candidate offre une vue de l'architecture visée. Cet artéfact fournit beaucoup information sur les orientations d'affaires et technologiques de l'organisation.

3.3.4 La discipline de réutilisation stratégique

La réutilisation stratégique est la discipline charnière entre les disciplines d'entreprise et les phases de développement d'une SOA. À l'aide des extrants des disciplines précédentes, la réutilisation stratégique permet de mettre à profit les actifs existants dans d'autres projets de l'entreprise. La Figure 3.5 illustre les activités de cette discipline qui ont été documentés dans EPF.

La réutilisation stratégique est un élément clef de la méthodologie présentée dans ce projet de recherche. C'est souvent suite à la discipline de réutilisation stratégique qu'un projet de développement va être lancé. Selon le cas, le projet en sera un de développement lorsqu'un nouvel actif est à développer ou à bonifier ou d'intégration lors de l'acquisition et l'intégration d'un actif externe. Les principaux livrables de cette discipline sont le plan de réutilisation et les actifs réutilisables.

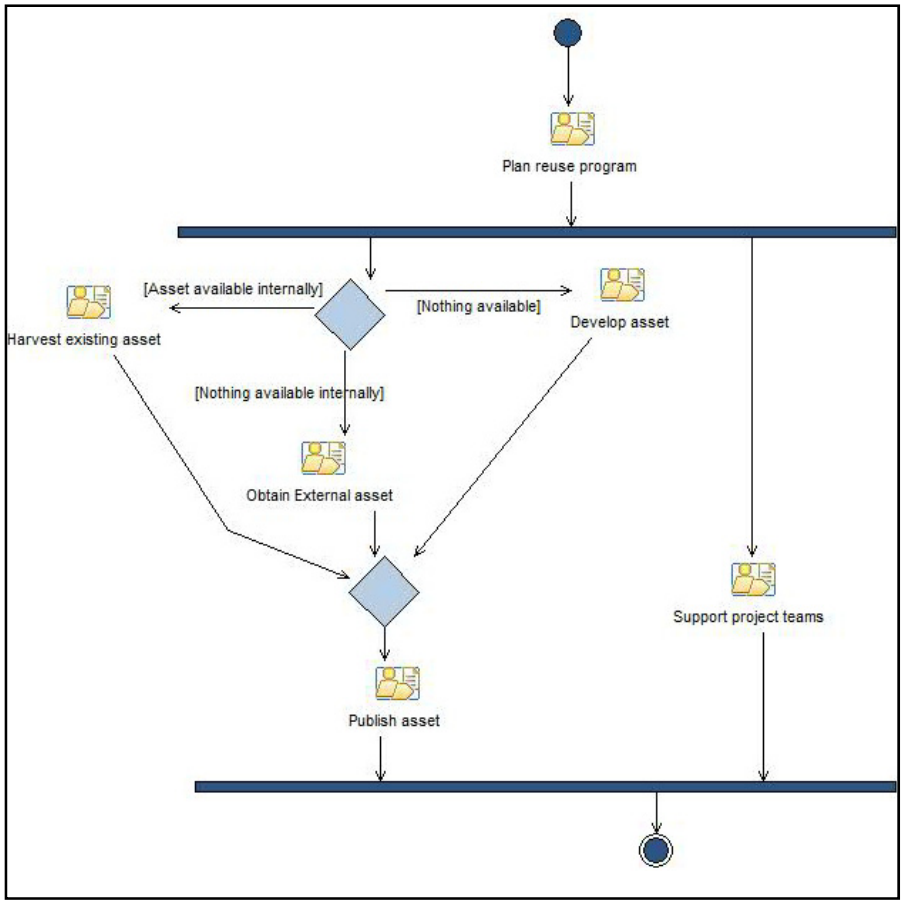


Figure 3.5 Les activités de la réutilisation stratégique.
Adaptée de (Scott W.Ambler, 2005)

- **Planifier le programme de réutilisation**

La planification du programme de réutilisation est nécessaire afin d'en faciliter sa réussite au sein de l'entreprise. Il faut prévoir du temps et des ressources. Autrement, la pression exercée par les projets prendra le dessus et le programme de réutilisation sera mis de côté.

Le plan permet de déterminer les buts de la réutilisation stratégique dans l'entreprise et les personnes qui seront impliquées dans ce programme. Il permet également de déterminer la stratégie à appliquer pour réaliser les objectifs de la réutilisation et les outils qui seront utilisés afin de maintenir les actifs.

Le plan identifie le potentiel de réutilisabilité de l'entreprise et la stratégie de communication qui sera utilisée afin de promouvoir la réutilisation des actifs. Le plan clarifie comment seront maintenus les actifs réutilisables et comment ils évolueront. Le principal livrable de cette activité est le plan de réutilisation.

- **Identifier les actifs existants**

Cette activité permet d'identifier au sein de l'entreprise les actifs existants qui pourraient être réutilisables. Cette activité inclut une évaluation de l'actif afin de déterminer si oui ou non, l'actif est réutilisable. Dans l'affirmative, la seconde étape serait de généraliser cet actif. La généralisation permet de sortir l'actif de son contexte initial (pour un projet spécifique) et de le rendre suffisamment générique pour une utilisation dans d'autres projets. La généralisation demande un plus grand effort que le développement pour une utilisation unique.

L'identification des actifs pendant la réalisation d'un projet permet à un responsable de la réutilisation de généraliser l'actif et de l'intégrer au bénéfice du projet courant. Cependant, cela peut gêner l'équipe de projet si la généralisation cause des délais. Identifier les actifs à la fin du projet permet au responsable de la réutilisation de généraliser un actif sans nuire à l'équipe de projet. Il y a cependant un risque que cet actif ne soit jamais utilisé. On peut également généraliser un actif existant au début d'un nouveau projet, s'assurant ainsi d'avoir au moins deux projets réutilisant le même actif. Par contre, il n'est pas garanti que l'équipe originale soit disponible pour travailler avec l'actif réutilisable et qu'ils intègreront la version généralisée dans leur système.

- **Obtenir des actifs externes**

Il est possible d'acquérir des actifs provenant de l'extérieur de l'entreprise. Il n'est alors pas nécessaire de développer cet actif et de le généraliser, il ne reste qu'à l'intégrer et/ou à l'utiliser.

Il existe plusieurs sources pour obtenir des actifs réutilisables. Il y a les solutions libres (« *Open Source* ») ou encore des produits commerciaux. L'auteur (Scott W.Ambler, 2005) mentionne qu'il est préférable dans un premier temps d'évaluer ce qui se fait de gratuit et de l'évaluer. Après cela, regarder du côté des systèmes commerciaux et finalement, si nécessaire, penser au développement à l'interne.

- **Développer des actifs**

Le développement des actifs réutilisables doit se faire lorsqu'il y a un fort potentiel de réutilisation. Autrement, le risque est élevé qu'un actif développé ne soit finalement jamais utilisé.

Il y a plusieurs avenues possibles. Développer une composante avec la réutilisabilité en tête sans la généraliser tout de suite. Si plusieurs équipes manifestent un intérêt pour cette composante, il est alors plus facile de la retravailler. S'il y a un intérêt potentiel d'utilisation à l'extérieur de l'entreprise, il est possible d'utiliser la communauté du code source libre qui sera susceptible de modifier et développer le projet avec l'équipe de l'entreprise. Dans ce cas, il y a des aspects légaux qu'il faut s'assurer de vérifier. Si l'utilisation des communautés de code source libre n'est pas permise, il est également possible de faire appel à la communauté à l'intérieur de l'entreprise.

- **Faire évoluer les actifs**

L'évolution des actifs est nécessaire. Les lois de Lehman (Lehman, 1980) énoncent que l'évolution des actifs est nécessaire, au risque de voir ces actifs péricliter et disparaître. Les besoins et les exigences des utilisateurs changent. Il peut y avoir des erreurs ou encore l'arrivée d'une nouvelle plateforme qui force la mise à jour d'un actif.

Dans cette activité, la propriété de l'actif doit être déterminée afin de connaître qui aura la responsabilité de le maintenir. C'est peut-être l'équipe qui a développé l'actif, ou les clients qui l'utilisent.

- **Publier des actifs**

Que l'actif ait été développé, récupéré ou acheté, il faut le rendre disponible aux personnes susceptibles d'en avoir besoin. Cela fait partie du plan de communication de la stratégie de réutilisation. L'actif est généralement inscrit dans un registre et rendu disponible d'une quelconque façon (répertoire partagé, gestionnaire de code source, etc.). Une annonce est ensuite faite sur les différents canaux de communication disponibles.

- **Retirer un actif**

Cette activité permet de préparer le retrait d'un ancien actif. Généralement, cet évènement survient lorsqu'un actif plus récent est disponible. Dans pareil cas, une date limite sera alors communiquée aux équipes utilisant l'ancien actif. Cela a pour but d'offrir un délai avant la fin du support et la possibilité d'intégrer la nouvelle version.

- **Supporter les équipes de projet**

Le support aux équipes de projet est une activité déterminante dans la réussite d'une stratégie de réutilisation des actifs d'une entreprise. Il doit y avoir une volonté de vouloir réutiliser des composantes, sachant que l'effort pour généraliser un actif est considérable. Des ressources doivent donc accompagner les équipes de projets dans l'atteinte de cet objectif.

De plus, plusieurs facteurs vont influencer le degré de succès de la réutilisation des actifs par les équipes de projets. L'un de ces facteurs est la confiance des équipes de projet envers la qualité de l'actif, sa facilité d'utilisation ainsi qu'une documentation claire incluant des exemples.

- **Mesurer le programme de réutilisation**

Pour valider le programme de réutilisation stratégique, il faut identifier des métriques permettant de mesurer les économies réalisées. Les exemples de métriques sont le nombre d'heures net sauvé par actif, le coût de base de chaque actif, le bénéfice net d'un actif ou

encore le bénéfice total du programme de réutilisation. Le Tableau 3.5 donne les principaux livrables de la discipline de réutilisation stratégique.

Tableau 3.5 Les principaux artéfacts de la discipline de réutilisation stratégique
Adapté de (Scott W.Ambler, 2005)

Artéfacts	Description
Les actifs actuels	On peut voir cet artéfact soit comme l'ensemble des actifs réutilisables, soit comme la liste des actifs réutilisables.
Le plan de réutilisation	Cet artéfact énonce la politique de réutilisation de l'organisation.

3.4 Description des activités pour le développement d'une SOA

Maintenant que nous avons un cadre d'architecture d'entreprise, il faut y ajouter les activités de développement SOA. Dans le chapitre 2, nous avons sélectionné la méthode de (Erl, 2005) comme étant la source de ces pratiques et processus. Une des raisons pour laquelle les travaux de (Erl, 2005) ont été retenus est la possibilité de faire un lien entre ces phases et les phases classiques de développement d'OpenUP. Cela est illustré dans le Tableau 3.6.

Tableau 3.6 Les phases d'OpenUP et les activités de Thomas Erl

OpenUP	Thomas Erl
Initialisation	Analyse orientée services
Élaboration	Conception orientée services
Construction	Développement Test
Transition	Déploiement
N/A	Administration
N/A	

Les phases qui diffèrent du développement classique sont surtout les phases d'analyse et de conception (Erl, 2005, p. 358). Ce sont ces deux phases qui seront intégrées dans les phases d'initialisation et d'élaboration d'OpenUP. Les phases de développement, de test et de déploiement seront substituées par leurs équivalents bien plus étoffés dans OpenUP. Finalement, la phase d'administration qui correspond aux nouvelles phases de production et de retrait proposé par (Scott W.Ambler, 2005) ne fait pas partie de ce mémoire pour les raisons expliquées dans le chapitre 2.

L'analyse et la conception orientée services ont dû êtres modélisés dans EPF. Les sections qui suivent décrivent en détail les activités orientées services que nous allons intégrer dans notre nouvelle méthode. Nous en avons un aperçu à la Figure 3.6.

Presentation Name	Index	Predecessors
SOA Inception phase	0	
SOA Analysis	1	
Define business requirements	2	
Identify automation systems	3	2
Model candidate services	4	3
Presentation Name	**Index**	**Predecessors**
SOA Elaboration phase	0	
SOA Design	1	
Compose SOA	2	
Design entity-centric business services	3	
Design application services	4	
Design task centric business services	5	3,4
Design service oriented business process	6	5

Figure 3.6 Les phases de SOA dans EPF.
Adaptée de (Erl, 2005)

3.4.1 L'activité d'analyse orientée services

La phase d'analyse orientée services permet d'identifier quels sont les services à créer. Elle contient trois tâches (*Voir* Figure 3.7). On définit lors de cette phase les différentes couches d'abstractions qui sont propres à une SOA. (Erl, 2005) précise que la phase d'analyse orientée services propose des services, sans tenir compte des contraintes, notamment les contraintes environnementale ou technologique. Ce sont les services candidats. Lors de la conception des services, il est possible qu'il y ait plusieurs changements afin de concilier la modélisation faite en analyse aux contraintes. Les services candidats sont les principaux livrables de l'analyse orientée services. Ces services candidats sont ensuite proposés et soumis à l'activité de conception orientée services.

Presentation Name	Index	Predecessors
SOA Inception phase	0	
SOA Analysis	1	
Define business requirements	2	
Identify automation systems	3	2
Model candidate services	4	3

Figure 3.7 Les tâches de l'activité Analyse orientée services**.**
Adaptée de (Erl, 2005)

- **Définir les exigences d'affaires**

Cette tâche permet d'identifier les processus d'affaires visées par les travaux.

- **Identifier les systèmes automatisés**

Avant de pouvoir planifier la création des services, il faut identifier les applications qui supportent les processus d'affaires identifiés précédemment.

- **Modéliser les services candidats**

Lorsque les processus d'affaires et les applications qui les supportent sont identifiés, il est temps d'identifier le ou les services candidats. Il est possible de regrouper ces services selon le contexte. Éventuellement, ces services seront peut-être assemblés afin de créer un service composite. Cette tâche est décomposée en douze étapes :

Étape 1. Décomposer le processus d'affaires

À cette étape, on décompose un processus d'affaires en une série d'étapes plus petites. Il est possible que lors de cette décomposition, nous obtenions une granularité plus fine que celle qui a été documentée. On profite de cette activité pour alimenter ou valider le glossaire et le modèle du domaine de l'entreprise.

Étape 2. Identifier les opérations candidates d'affaires

Avec les éléments du processus d'affaires identifié précédemment, on identifie les opérations qui sont susceptibles d'être réalisées par un service. Peuvent êtres exclues les opérations manuelles (faite par un individu et non automatisable) ainsi que les opérations déjà réalisées par un système patrimonial et où la création d'un service n'est pas envisageable.

Étape 3. Définir la logique d'orchestration

Cette étape est spécifique à l'orchestration des services. Si une couche d'orchestration a été prévue, il faut identifier les éléments qui participent à la logique de réalisation du service, par exemple les règles d'affaires, les conditions, les exceptions et les séquences d'actions.

Étape 4. Créer les services candidats d'affaires

Il faut séparer en groupes logiques les différentes opérations du processus d'affaires. Chaque groupe ou contexte représente un service candidat. Pour chacun de ces services, on identifie les opérations susceptibles d'être implémentées.

Étape 5. Raffiner et appliquer les principes orientés services

On étudie plus en détail les services candidats créés précédemment et on révise deux des principes clefs de l'architecture orientée services: la réutilisabilité et l'autonomie.

Étape 6. Identifier la composition des services candidats d'affaires

Pour l'ensemble des services identifiés et leurs opérations respectives, recenser les différents scénarios possibles dans la réalisation du processus d'affaires. Cette étape a pour but d'identifier les possibles compositions de services, ce qui inclut peut-être la création de services d'orchestration. Elle permet également de valider le regroupement des différents contextes identifiés plus tôt.

Étape 7. Réviser les groupes d'opérations des services candidats d'affaires

Revoir les opérations et les services suite à l'identification de la composition des services candidats. Lors de cette étape, des services sont parfois créés, des opérations sont fusionnées, etc.

Étape 8. Analyser les exigences de traitement

Les étapes précédentes permettent d'analyser en détail les constituants du processus d'affaires ainsi que sa logique de fonctionnement. On identifie dans cette étape les applications existantes (ou à développer) qui supportent actuellement le processus d'affaires.

Étape 9. Identifier les opérations des services candidats applicatifs

La suite logique est d'identifier les opérations réalisées par les applications qui seront appelées par un service.

Étape 10. Créer les services candidats applicatifs

On regroupe les opérations en service candidat selon le contexte. Ce regroupement peut se faire sur la base d'une association avec une application patrimoniale, selon le type de fonction, etc. Cette étape est similaire à l'étape 4.

Étape 11. Réviser la composition des services

Reprendre ce qui a été fait aux étapes 5 et 6. Appliquer les principes orientés services. Réviser les scénarios d'exécution en intégrant cette fois les services applicatifs candidats. Représenter les liens associant les services candidats d'affaires et les services candidats applicatifs

Étape 12. Réviser les groupes d'opérations des services applicatifs candidats

Revoir les opérations et les services suite à l'identification de la composition des services candidats. Lors de cette étape, des services sont parfois créés, des opérations sont fusionnées, etc.

Les extrants de cette phase sont des services et des opérations d'affaires ainsi que des services et des opérations applicatifs. S'il y a lieu, des informations sur l'orchestration sont identifiées. Les services sont regroupés selon leur contexte. Ce regroupement peut se faire sur la base de leur fonction, de leur appartenance à un système, etc.

3.4.2 L’activité de conception orientée services

La conception orientée services utilise les services candidats identifiés lors de l'analyse orientée service afin d'implémenter une solution concrète. Elle contient cinq tâches (*voir* Figure 3.8). C'est lors de la phase de conception que sont déterminées les technologies et les plateformes qui seront utilisées pour développer les services. C'est également pendant la phase de conception que sont choisis les standards et les extensions requis pour la SOA.

Presentation Name	Index	Predecessors
SOA Elaboration phase	0	
SOA Design	1	
Compose SOA	2	
Design entity-centric business services	3	
Design application services	4	
Design task centric business services	5	3,4
Design service oriented business process	6	5

Figure 3.8 Les tâches de l'activité Conception orientée services
Adaptée de (Erl, 2005)

- **Composer la SOA**

Cette tâche définit les bases technologiques et architecturales sur lesquelles reposeront les services développés. C'est pendant la composition que sont choisies les caractéristiques et les extensions qui répondront aux exigences. La composition se décompose en trois étapes :

Étape 1. Choisir les couches d'abstraction des services

Lors de cette étape, l'équipe d'architecture définit le nombre de couches d'abstraction utilisé. On parle ici de la couche applicative, de la couche des services et de la couche d'affaires. On pourra retrouver dans la couche de service d'autres couches comme la couche des services applicatifs, la couche des services d'affaires ou encore la couche d'orchestration.

Si un nombre élevé de couches permet une meilleure décomposition et une meilleure réutilisabilité des services, cela peut également induire davantage de traitement afin de recevoir, traiter et expédier les multiples messages à travers les différentes couches. Ce trafic peut conduire à une congestion et à un ralentissement. Il peut être alors nécessaire de réaliser des tests de performance.

Étape 2. Identifier les principaux standards à utiliser pour les services

Cette étape sert à définir les principaux standards qui seront utilisés, par exemple, choisir de respecter le profil de base proposé par le WS-I, choisir le type de service ou identifier les versions de SOAP, de WSDL et de XSD qui seront utilisées.

Étape 3. Choisir les extensions SOA à utiliser avec les services

Cette étape sert à définir les principales extensions qui seront utilisées, par exemple, choisir d'intégrer des fonctionnalités de cryptages avancées avec WS-Encryption.

- **Concevoir les services d'affaires entités**

Les services d'affaires entités représentent les données du domaine d'affaire avec lesquelles le système travaille. Le modèle du domaine, le glossaire ou le modèle de base de données sont des sources d'inspiration intéressantes afin d'identifier les services d'affaires entités. Ces services réalisent peu d'opérations et seront plutôt réutilisés dans les services composites. La conception des services d'affaires entités se décompose en sept étapes :

Étape 1. Identifier les services d'affaires entités existants

Avant de développer un nouveau service d'affaire entité, il faut vérifier qu'il n'existe pas déjà dans les actifs de l'organisation. Dans ce cas, vérifier si les opérations candidates sont présentes. Il faut également vérifier si d'autres services entités fournissent certaines fonctionnalités des opérations candidates. Dans la discipline de réutilisation stratégique, le registre des actifs existants est un artéfact qui peut être très utile.

Étape 2. Définir le schéma des messages

On définit à cette étape le schéma des messages qu'utilisent les services d'affaires entités. Certains outils génèrent cette partie automatiquement. Le modèle du domaine peut fournir de précieuses informations pour réaliser cette étape.

Étape 3. Dériver l'interface abstraite

Il faut analyser les opérations candidates identifiées lors de l'analyse, confirmer que ces opérations sont suffisamment génériques et réutilisables et définir les opérations concrètes alignées avec le schéma des messages.

Étape 4. Appliquer les principes orientés services

Puisque les services d'affaires entités vont êtres composés par des services parents dans la réalisation d'une tâche ou d'un processus, il faut que le principe d'autonomie soit assez fort et que les dépendances avec d'autres services soient ramenées au minimum. Les services d'affaires entités sont généralement sans état.

Étape 5. Standardiser l'interface du service d'affaires entité

S'il y a des contraintes de conception particulières, comme le respect d'un profile du WS-I, on les intègre et on raffine l'interface.

Étape 6. Étendre la conception du service d'affaires entité

La phase d'analyse se concentre surtout à identifier les opérations d'affaires. Lors de la conception, il est important d'inclure les principales opérations utilitaires d'un service d'affaire entités. Ces opérations sont généralement l'ajout, la suppression, la modification et la récupération.

Étape 7. Analyser les exigences de traitement requis

Identifier dans cette étape si des services applicatifs sont nécessaires pour supporter les services d'affaires entité. Si oui, est-ce que ces services applicatifs existent ou doivent-ils être créés?

- **Concevoir les services applicatifs**

La conception des services applicatifs est le pont entre l'infrastructure applicative identifiée dans les disciplines gestion de portefeuille et architecture d'entreprise et les services entités.

C’est à cette étape que sont effectuées les transformations nécessaires pour assurer l’interopérabilité entre les systèmes hétérogènes, par exemple la conversion en XML d’une information binaire ou textuelle. La conception des services applicatifs se décompose en sept étapes :

Étape 1. Identifier les services applicatifs existants

Avant de développer un nouveau service applicatif, il faut vérifier qu'il n'existe pas déjà dans les actifs de l'organisation. Dans ce cas, vérifier si les opérations candidates sont présentes. Il faut également vérifier si d'autres services applicatifs fournissent certaines fonctionnalités des opérations candidates.

Étape 2. Confirmer le contexte

Dans la phase d'analyse, c'est surtout le contexte d'affaires des opérations et des services qui oriente les regroupements. Dans la phase de conception, la connaissance des applications existantes force à revoir le contexte de certains de ces services.

Étape 3. Dériver une interface initiale

La première version de l'interface est créée lors de cette étape. On utilise les services applicatifs candidats et leurs opérations respectives. On intègre les intrants et les extrants de ces opérations.

Étape 4. Appliquer les principes orientés services

Comme lors des étapes précédentes du même nom, l'application des principes de l'architecture orientée services est un exercice de révision. Cette révision permet de renforcer les principes de réutilisabilité et d'autonomie.

Étape 5. Standardiser et raffiner l'interface du service applicatif

S'il y a des contraintes de conception particulières, comme le respect d'un profil du WS-I, on les intègre et on raffine l'interface. On applique les standards utilisés pour les autres types de services.

Étape 6. Ajouter des fonctionnalités

Afin d'augmenter la valeur du service applicatif, il peut être intéressant de profiter de cette occasion pour ajouter des fonctionnalités qui pourraient être utiles ultérieurement. C'est ce que (Erl, 2005) appelle des caractéristiques spéculatives. Bien qu'allant à l'encontre de la philosophie agile, cette étape peut s'avérer nécessaire dans le cadre d'une architecture d'entreprise, où une vision à long terme est requise.

Étape 7. Identifier les contraintes techniques des services applicatifs

La dernière étape dans la conception des services applicatifs est l'identification des contraintes techniques. Les services applicatifs sont dépendants des technologies qu'ils exposent. Est-ce que l'application a des contraintes de sécurités particulières? Est-ce qu'il y a des délais de réponses? Quelles sont les performances? Il faut déterminer également quelle technique le service va utiliser pour communiquer avec le système, surtout si c'est un vieux système.

- **Concevoir les services de tâche d'affaires**

La conception des services de tâche d'affaires assure l'interaction des services entités afin de réaliser une tâche précise ou un processus d'affaires. Ce sont peut-être les services les moins réutilisables, car l'interaction des services entités est généralement unique à une tâche spécifique. À cette étape on identifie et modélise les processus d'affaires et les flots de travail. Selon le cas, cette information proviendra de la discipline « Modélisation d'affaires » ou alimentera cette dernière. Il y a cinq étapes :

Étape 1. Définir la logique du flux de travail

Cette étape permet d'identifier tous les scénarios de réalisation du processus d'affaires. Si lors de l'analyse, il y a eu identification des services candidats de composition, une partie du travail a été réalisé. Les services de tâches d'affaires combinent généralement des services d'affaires entités et des services applicatifs. On suggère l'utilisation de diagramme d'activité pour représenter les différentes possibilités.

Étape 2. Dériver l'interface initiale du service de tâches d'affaires

Créer une interface à partir des opérations candidates identifiées plus tôt. Dans ce cas-ci, les différents scénarios et les diagrammes d'activités seront des sources d'informations complémentaires importantes.

Étape 3. Appliquer les principes orientés services

Dans ce cas-ci, le principe de réutilisabilité est plus difficile à appliquer. La tâche regroupe des services applicatifs et des services d'affaires entités pour réaliser une tâche un peu plus précise, moins générique.

Les services d'affaires tâches, en tant que service composite, partagent l'autonomie de leurs constituants. En d'autres mots, si les principes orientés services ont été bien appliqués en amont, le service d'affaires tâches en bénéficiera.

Le service d’affaires tâche exécute un flux de travail et doit parfois conserver en mémoire un état. (Erl, 2005) suggère de préserver cet état dans les messages qui sont échangés. Cela empêche d'augmenter le couplage en déléguant cette fonction. En contrepartie, cela risque de générer plus de trafic sur le réseau, tout en augmentant le traitement pour traiter les états.

Étape 4. Standardiser et raffiner l'interface du service d'affaires tâche

S'il y a des contraintes de conception particulières, comme le respect d'un profile du WS-I, on les intègre et on raffine l'interface. On applique les standards utilisés pour les autres types de services.

Étape 5. Identifier le traitement requis

Le service d'affaires tâches, en tant que service composite, utilisera des services applicatifs. Ces services applicatifs sont attachés à une ou plusieurs applications distinctes. Il faut réévaluer les dépendances aux applications pour ce service tâche et voir si d'autres services sont nécessaires.

- **Concevoir les services d'affaires processus**

Finalement, la conception des services de processus d'affaires permet la mise en commun des différents services de tâches afin de réaliser les scénarios plus complexes de processus d'affaires. C'est ici qu'est faite la conception de l'orchestration ou de la chorégraphie des services. Il y a cinq étapes :

Étape 1. Faire le lien entre les scénarios d'interactions

Identifier les messages qui relient les différents scénarios identifiés précédemment, les services d'affaires disponibles et tous les participants aux scénarios.

Étape 2. Concevoir l'interface du service processus

La conception de l'interface du service processus débute avec les différentes opérations ainsi qu'avec les intrants et les extrants trouver à l'étape 1. À ce stade, les outils génèrent souvent la définition à partir des services existants qui composent le service processus.

Étape 3. Formaliser la conversation entre les participants au processus

À cette étape, on utilise les normes existantes pour la conversation entre les participants du processus, par exemple WS-BPEL. Cette étape permet de déterminer comment va se dérouler

la conversation entre les services qui sont mis à contribution. On spécifie le rôle joué par chaque service dans ce service processus.

Étape 4. Définir la logique du processus

On utilise les éléments définis dans les étapes précédentes pour définir la logique du flux de travail. Cette définition inclut le scénario nominal ainsi que tous les scénarios alternatifs. S'il y a des blocs conditionnels, c'est à cette étape qu'ils sont intégrés. On ajoute également la gestion des défauts et des erreurs dans le processus.

Étape 5. Aligner les interactions des scénarios et raffiner la définition du processus

Cette étape optionnelle est une révision du service processus avec les scénarios identifiés à l'étape 1. On peut ainsi documenter les ajouts qui ont pu s'introduire pendant le processus de conception. Si un diagramme d'activité a été utilisé pour documenter la solution, il faut le mettre à jour.

3.5 Description des phases d'OpenUP

Une fois que les premières itérations des disciplines d'entreprise sont faites, on est en mesure d'initialiser des projets de développements SOA alignés avec les besoins d'affaire de l'organisation. Comme le mentionne (Erl, 2005), les activités qui sont particulières à la SOA sont l'analyse orientée services et la conception orientée services. Les phases subséquentes sont identiques aux méthodologies classiques.

3.5.1 La phase d'initialisation

Dans la phase d'initialisation, on identifie les besoins d'affaires, les systèmes impliqués ainsi que les services exigés. Parmi ces activités, on retrouve les activités d'OpenUP comme la planification de l'itération. À la fin de cette phase, l'équipe devrait être en mesure de dire si le

projet est réalisable ou non. C'est pourquoi cette phase doit éviter d'être trop longue. La Figure 3.9 illustre les activités de cette phase.

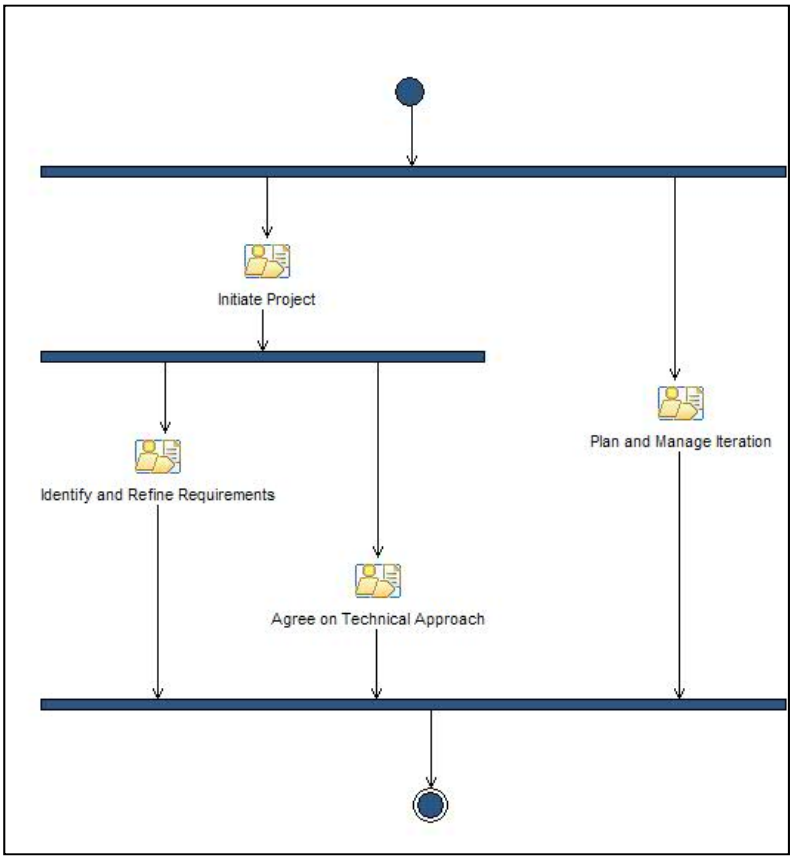


Figure 3.9 Les activités de la phase d'initialisation.
Tirée de (OpenUP, 2010)

- **Amorcer le projet**

Le projet est lancé. C'est lors de cette activité que sont identifiés les intervenants du projet et qu'un document de vision est rédigé. On rédige le plan de projet. On crée le glossaire du projet. Si un glossaire d'entreprise existe, on l'utilise et on le met à jour.

- **Planifier et gérer l'itération**

On retrouve cette activité à toutes les phases du développement et elle se déroule pendant tout le cycle de vie d'une itération. Elle permet d'identifier en début d'itération quels sont les risques et les priorités. Le travail est assigné aux différents membres de l'équipe. Pendant le

déroulement de l'itération, les membres de l'équipe se rencontrent régulièrement pour évaluer ce qui a été réalisé, ce qu'il reste à faire et quels sont les problèmes qui perturbent le bon déroulement de l'itération. À la toute fin, on compare la planification de l'itération à sa réalisation et on en tire les leçons appropriées.

- **Identifier et raffiner les exigences**

C'est lors de cette activité que sont identifiées les exigences. Dans la phase d'initialisation, l'analyste cherche à comprendre le problème et à s'entendre avec les parties prenantes sur les principales exigences du système. Les cas d'utilisations les plus importants sont rédigés et le glossaire est mis à jour.

- **Entente sur l'approche technique**

Cette activité aligne une approche technologique aux informations tirées de l'activité d'identification et de raffinement des exigences. L'architecte identifie les contraintes touchant l'architecture du système. Les interfaces avec les autres systèmes sont recensées. Une première réflexion portant sur le déploiement du système a lieu. C'est également à ce moment que l'architecte identifie les opportunités de réutilisation. C'est une étape importante dans le cadre de cette méthode puisque l'une des propriétés d'une SOA est son potentiel de réutilisation des services développés.

3.5.2 La phase d’élaboration

Dans la phase d’élaboration, l’équipe de projet se concentre à comprendre les exigences et a créer les bases architecturales du système. Les risques les plus élevés sont identifiés et minimisés. Les premiers incréments de la solution sont développés. La Figure 3.10 illustre les activités de cette phase. Toutes les activités sont réalisées en parallèle.

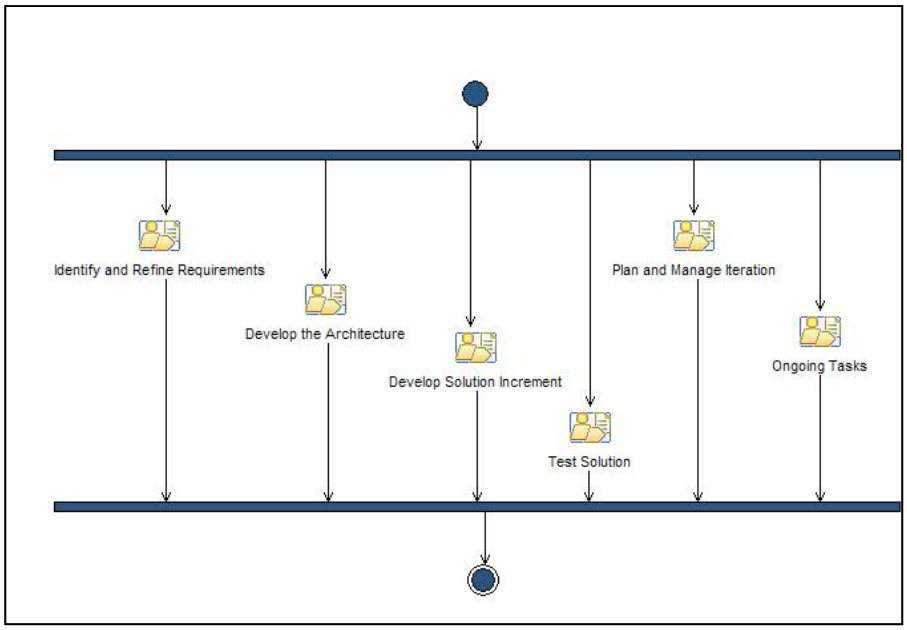


Figure 3.10 Les activités de la phase d'élaboration.
Tirée de (OpenUP, 2010)

- **Planifier et gérer l'itération**

Se référer à La phase d'initialisation à la page 101

- **Identifier et raffiner les exigences**

C'est lors de cette activité que sont identifiées les exigences. Dans la phase d'élaboration, on identifie les exigences ayant le plus de valeur pour les parties prenantes du projet. L'analyste s'attarde maintenant à identifier les solutions. Les cas de test sont rédigés afin de valider les exigences lors de leur développement.

- **Développer l'architecture**

On raffine l'architecture telle qu'elle a été esquissée lors de l'entente sur l'approche technique. Dans cette activité, l'architecte et les développeurs travaillent de concert afin de produire un produit stable conforme aux exigences. On s'assure que les décisions architecturales sont bien communiquées aux développeurs, que l'équipe a suffisamment d'information pour développer

le produit et que les exigences qui ont été priorisées pour l'itération en cours sont correctement implémentées.

- **Développer un incrément de la solution**

Cette activité regroupe l'ensemble des étapes permettant la création d'un incrément de la solution. Cela inclut la conception, l'implémentation, les tests et l'intégration de l'incrément à la solution.

- **Tester la solution**

Lorsque l'incrément a été intégré à la solution, on teste le système pour s'assurer que la nouvelle fonctionnalité réponde aux exigences. Le résultat des tests est communiqué aux membres de l'équipe. Selon le résultat des tests, des actions appropriées sont prises.

- **Tâches non planifiées**

Cette activité permet de recevoir à tout moment lors d'une itération, des requêtes de changement. Ces requêtes proviennent généralement des parties prenantes. Dans le contexte d'un projet itératif, ces demandes sont courantes et témoignent de l'évolution de la perception des besoins des intervenants.

3.5.3 La phase de construction

La phase de construction inclut le développement des services et les tests qui sont associés à ce développement. Lorsque débute la phase de construction, les principaux risques ont été traités. L’architecture des composantes logicielles est pratiquement terminée. Même si on raffine encore les exigences, l’accent est mis sur le développement et le test des incréments. La Figure 3.11 illustre les activités de cette phase. Toutes les activités sont réalisées en parallèle.

- **Planifier et gérer l'itération**

Se référer à La phase d'initialisation à la page 101

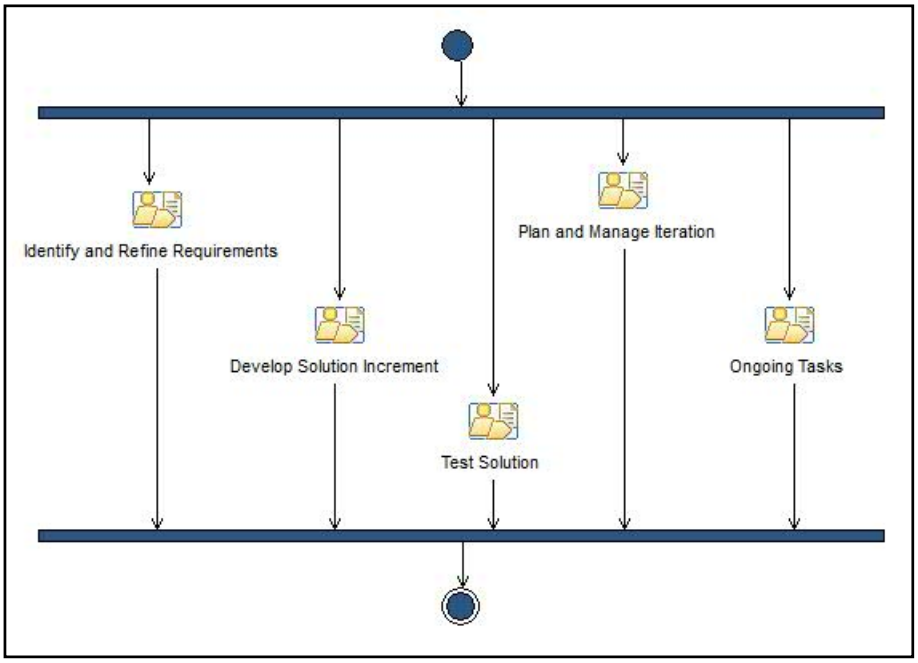


Figure 3.11 Les activités de la phase de construction.
Tirée de (OpenUP, 2010)

- **Identifier et raffiner les exigences**

Se référer à La phase d'élaboration à la page 103

- **Développer un incrément de la solution**

Se référer à La phase d'élaboration à la page 103

- **Tester la solution**

Se référer à La phase d'élaboration à la page 103

- **Tâches non planifiées**

Se référer à La phase d'élaboration à la page 103

3.5.4 La phase de transition

La phase de transition réalise les tests d'acceptation, le déploiement et la publication des services dans l'environnement de production. On profite de cette phase pour apporter les derniers ajustements avant la livraison finale de cette version. La Figure 3.12 illustre les activités de cette phase. Toutes les activités sont réalisées en parallèle.

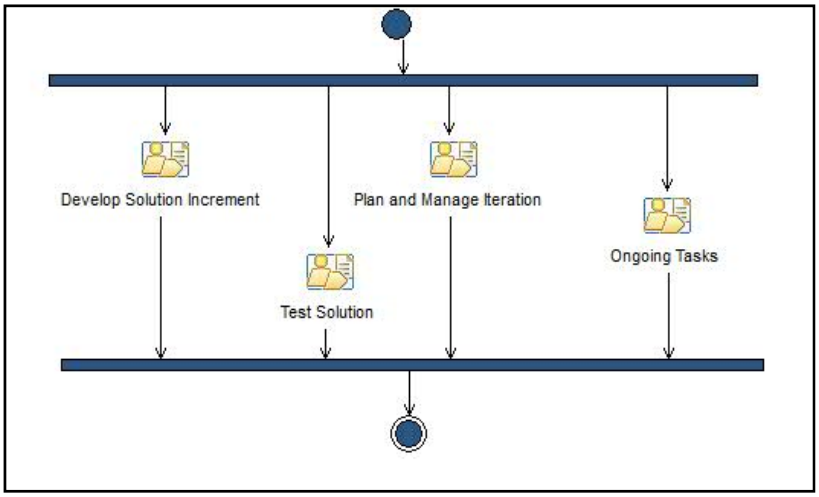


Figure 3.12 Les activités de la phase de transition.
Tirée de (OpenUP, 2010)

- **Planifier et gérer l'itération**

Se référer à La phase d'initialisation à la page 101

- **Développer un incrément de la solution**

Cette activité regroupe l'ensemble des étapes permettant la création d'un incrément de la solution. Cela inclut la conception, l'implémentation, les tests et l'intégration de l'incrément à la solution. Normalement, les incréments les plus risqués ont été traités dans la phase

d'élaboration. Dans la phase de transition, on s'affaire à ajouter les dernières demandes de changements.

- **Tester la solution**

Se référer à La phase d'élaboration à la page 103

- **Tâches non planifiées**

Se référer à La phase d'élaboration à la page 103

3.6 Intégration des méthodologies

Dans les sections précédentes, trois méthodologies ont été sélectionnées. Deux de ces trois méthodologies, EUP et la méthodologie de (Erl, 2005) ont dû être documentés dans EPF. Dans les prochaines sections, on réalise une analyse comparative de ces trois méthodologies afin de les intégrer en une seule et même méthode. On réalise dans un premier temps une analyse entre EUP, l'analyse orientée service et la conception orientée service. Puis une analyse entre OpenUP, l'analyse orientée service et la conception orientée service. À chaque fois, on décrit comment les artéfacts ou les activités de l'un viennent alimenter ceux de l'autre.

3.6.1 EUP et SOA

3.6.1.1 L'analyse orientées services et EUP

Pour réaliser une SOA, il faut avoir une vision globale de l'organisation. (Erl, 2005) a décrit les principales étapes permettant de bâtir une SOA, mais sa méthodologie n'inclut pas en détail les activités nécessaires pour bâtir cette vision globale. C'est ici qu'EUP prend tout son intérêt. Plusieurs activités d'entreprises permettent d'alimenter cette étape dans le développement d'une SOA.

Lors de l'analyse orientée services, il faut analyser et décortiquer les processus d'affaires qui doivent avoir été préalablement documentés. Il faut également connaître les différentes applications existantes (ou à développer) associées à ces processus d'affaires (Erl, 2005, p. 416). Cette connaissance est d'autant plus nécessaire si l'on souhaite développer des services ayant un fort potentiel de réutilisabilité. Si cette documentation existe, elle fait partie des artéfacts d'entreprise. Si cette documentation n'existe pas, il faut la rédiger. Ces artéfacts réalisés lors du projet pourront ainsi devenir des artéfacts d'entreprise.

Définir les exigences d'affaires

Dans l'analyse orientée services, l'une des tâches est de définir les exigences d'affaires. Le modèle des processus d'affaires d'EUP est un artéfact important à cette étape. C'est un artéfact qui provient de la discipline de modélisation d'affaires. La documentation peut également être mise à jour lors de l'exécution de cette tâche. Dans la discipline EUP de modélisation d'affaires, on propose de modéliser les processus d'affaires, le domaine et l'organisation. On se retrouve donc avec un ensemble d'artéfacts susceptibles d'êtres récupérés dans l'analyse orientée services. Le modèle du domaine et le glossaire permettent notamment de standardiser le langage utilisé par les experts métiers et de récupérer cette terminologie à travers tous les projets SOA.

Toujours dans la discipline de modélisation d'affaires, on identifie les options d'implémentation de processus. Pour planifier une SOA, il faut connaître les processus qui sont réalisés, autant manuellement qu'à l'aide de technologie de l'information. Ces artéfacts seront notamment utiles dans la modélisation des services candidats, qui demande de décomposer les processus d'affaires et de définir la logique d'orchestration des services.

Identifier les systèmes automatisés

Dans la discipline de gestion du portefeuille d'EUP, on réalise l'inventaire des systèmes actuels. Cet inventaire est nécessaire afin d'identifier les applications et les systèmes d'information existants. Cela permet d'avoir une vue de l'information et de son traitement dans l'organisation. Avec cet inventaire en main, il est possible d'évaluer avec plus de précision les projets SOA qui sont susceptibles d'apporter un bénéfice rapidement pour l'ensemble de l'organisation et de ses différents départements.

La gestion du portefeuille permet également d'identifier les dépendances entre les applications et l'information qui y circule. Ce sont autant d'éléments qui faciliteront l'analyse orientée services dans laquelle il est justement question d'identifier les systèmes à automatiser. L'artéfact « portefeuille d'application » provenant de la discipline de gestion du portefeuille fournit cette information.

Dans la discipline de réutilisation stratégique, il faut également réaliser l'inventaire des éléments réutilisables existants afin de pouvoir planifier une bonne réutilisation des actifs. Voici un autre excellent exemple de complémentarité entre EUP et SOA, illustré dans la Figure 3.13.

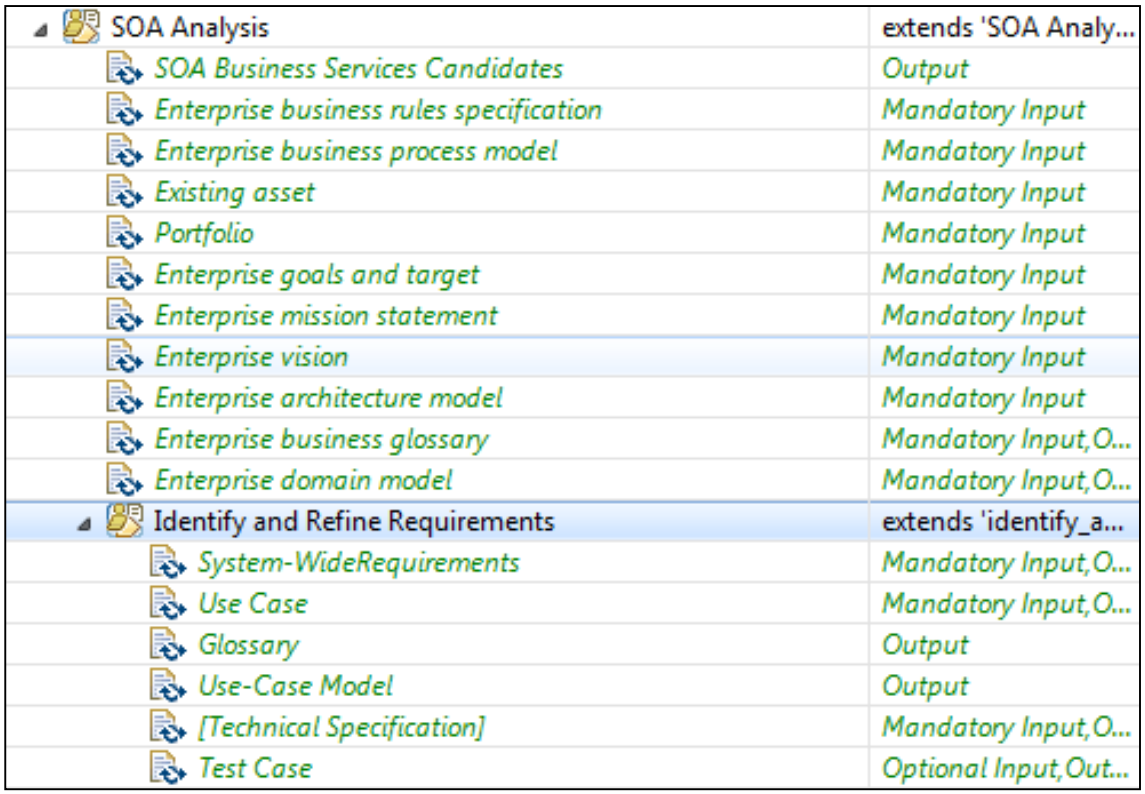


Figure 3.13 Artéfacts entrants et sortants dans l'analyse SOA
Adaptée de (*OpenUP*, 2010)

3.6.1.2 La conception orientée services et EUP

Composer la SOA

La conception orientée services comprend notamment l'identification des principaux standards et le choix des extensions SOA à utiliser avec les services. Bien que ces choix soient alignés en partie par des contraintes liées aux projets, il est nécessaire d'élaborer une vision plus large pour bénéficier d'une architecture technologique cohérente au sein de l'organisation.

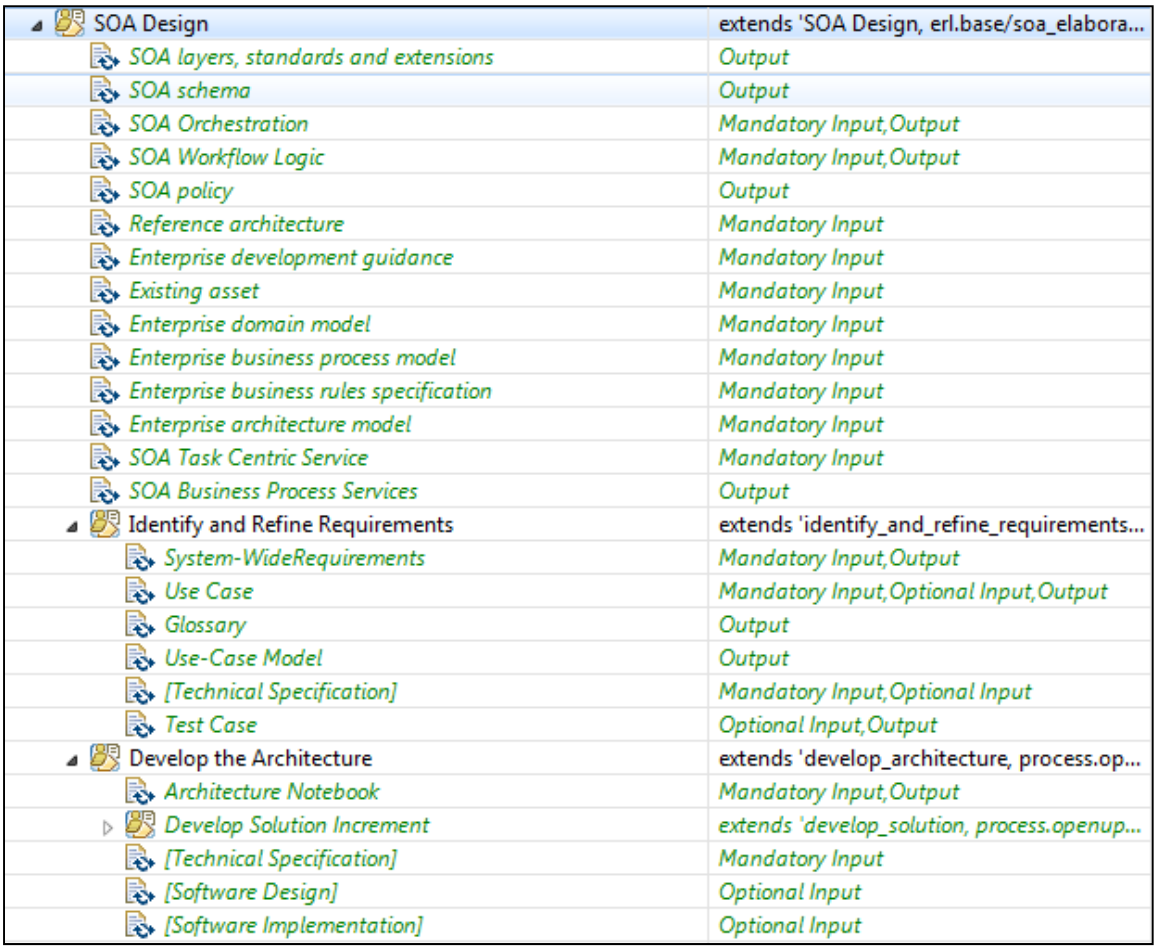
EUP vient bonifier la conception orientée services avec sa discipline d'architecture d'entreprise. En définissant les exigences architecturales pour l'ensemble de l'organisation, en suggérant des architectures candidates et en tenant à jour des référentiels architecturaux, la discipline d'architecture d'entreprise pose des balises pour les développements SOA futurs. Comme par exemple lors de la composition de la SOA, où on doit identifier les principaux standards à utiliser pour les services.

Le transfert d'information entre EUP et la conception orientée services n'est pas unilatéral. Chaque projet SOA est susceptible d'alimenter les artéfacts dans EUP. Un projet SOA va changer l'architecture technologique et cela doit se refléter dans la documentation d'entreprise et l'inventaire des systèmes. C'est aussi dans le cadre d'un projet qu'une organisation est amenée à identifier des référentiels architecturaux qui pourront être réutilisés dans d'autres projets. Finalement, tous les services développés dans le cadre d'un projet sont susceptibles de devenir des actifs important dans la stratégie de réutilisation de l'organisation, notamment les services entités et les services applicatifs plutôt génériques.

Une autre documentation susceptible d'être réutilisée dans plusieurs projets est l'ensemble des standards ainsi que les guides de modélisation définis dans la discipline d'architecture d'entreprise d'EUP. Au fur et à mesure que des projets orientés services sont réalisés, cette documentation pourra être raffinée et réutilisée. Comment l'organisation compte découper les processus et combien de couches d'abstraction seront utilisées sont autant d'informations qui pourront alimenter cette documentation.

Conception des services

Lorsqu'on réalise la conception des services entités et des services applicatifs, les artéfacts d'entreprise utilisés lors de l'analyse orientée services sont encore utilisés, que ce soit pour connaître les technologies en place, l'information qui y circule et les dépendances qui les relient. Les actifs existants, le modèle du domaine et le glossaire d'affaires sont des artéfacts susceptibles d'êtres utilisés encore ici, comme on peut le voir dans la Figure 3.14.



SOA Design	extends 'SOA Design, erl.base/soa_elabora...
SOA layers, standards and extensions	*Output*
SOA schema	*Output*
SOA Orchestration	*Mandatory Input,Output*
SOA Workflow Logic	*Mandatory Input,Output*
SOA policy	*Output*
Reference architecture	*Mandatory Input*
Enterprise development guidance	*Mandatory Input*
Existing asset	*Mandatory Input*
Enterprise domain model	*Mandatory Input*
Enterprise business process model	*Mandatory Input*
Enterprise business rules specification	*Mandatory Input*
Enterprise architecture model	*Mandatory Input*
SOA Task Centric Service	*Mandatory Input*
SOA Business Process Services	*Output*
Identify and Refine Requirements	extends 'identify_and_refine_requirements...
System-WideRequirements	*Mandatory Input,Output*
Use Case	*Mandatory Input,Optional Input,Output*
Glossary	*Output*
Use-Case Model	*Output*
[Technical Specification]	*Mandatory Input,Optional Input*
Test Case	*Optional Input,Output*
Develop the Architecture	extends 'develop_architecture, process.op...
Architecture Notebook	*Mandatory Input,Output*
Develop Solution Increment	*extends 'develop_solution, process.openup...*
[Technical Specification]	*Mandatory Input*
[Software Design]	*Optional Input*
[Software Implementation]	*Optional Input*

Figure 3.14 Artéfacts entrants et sortants dans la conception SOA.
Adaptée de (*OpenUP*, 2010)

3.6.2 OpenUP et SOA

(Erl, 2005) propose une méthodologie pour développer une SOA. Il s'intéresse particulièrement à l'analyse et à la conception orientée services et il considère que les phases subséquentes du projet sont très similaires aux cycles de développement habituel.

Cette méthodologie est intéressante, mais incomplète dans la perspective d'un projet de développement. Plusieurs activités de planification et de gestion de projets n'y figurent pas,

sans doute avec raison, puisque ce n'est pas l'objectif de sa méthodologie. Il convient alors d'essayer d'arrimer ces activités orientées services à une méthodologie de développement plus complète, comme OpenUP.

Il existe des similarités importantes entre les phases d'OpenUP et les phases de la méthodologie de (Erl, 2005). OpenUP décompose un projet logiciel en quatre grandes phases. (Erl, 2005) quant à lui décompose le développement d'une SOA en cinq phases. Il est possible de faire un lien entre les phases SOA de (Erl, 2005) et les phases classiques de développement d'OpenUP, tel qu'illustré précédemment dans le Tableau 3.6 de la page 88.

Les phases qui diffèrent du développement classique sont surtout les phases d'analyse et de conception (Erl, 2005, p. 358). Ce sont ces deux phases qui seront intégrées dans les phases d'initialisation et d'élaboration d'OpenUP. Les phases d'analyse et de conception orientées services sont converties en activités qui sont intégrées à OpenUP, car elles ne se suffisent pas à elles seules pour gérer un projet logiciel. Les phases de développement, de test et de déploiement sont substituées par leur équivalent dans OpenUP. Finalement, la phase d'administration qui correspond aux nouvelles phases de production et de retrait proposé par (Scott W.Ambler, 2005) ne fait pas partie de ce mémoire.

3.6.2.1 La phase d'initialisation et l'analyse orientée services

Dans la phase d'initialisation, les activités de démarrage du projet, de planification de l'itération et d'entente sur l'approche technique sont des activités complémentaires qui n'entrent pas en conflit avec l'analyse orientée services.

Par contre, dans l'identification et le raffinement des exigences, certaines étapes pourrait laisser penser qu'il y a un chevauchement dans la définition des tâches. Si on regarde de plus près les artéfacts entrants et sortants pour ces deux activités dans la Figure 3.13 à la page 111, on constate que l'analyse orientée services va utiliser un ensemble d'artéfacts provenant

d'EUP alors que l'activité d'identification et de raffinement des exigences va plutôt utiliser et générer des artéfact qui seront locaux au projet.

Même le cas d'utilisation, qui est reconnu pour illustrer les interactions entre un utilisateur et un système, pourrait être utilisé, dans ce contexte, pour représenter les interactions entre plusieurs processus. C'est pourquoi, ces deux activités ont été conservées, comme on l'illustre dans la Figure 3.15 de la page 113. Par contre, il apparaît logique que l'analyse orientée services débute un travail de plus haut niveau avec les artéfacts d'entreprise et inclut par la suite l'identification et le raffinement des exigences pour le projet. C'est pourquoi cette dernière activité se retrouve sous l'analyse orientée services, tel qu'illustré sur la Figure 3.15.

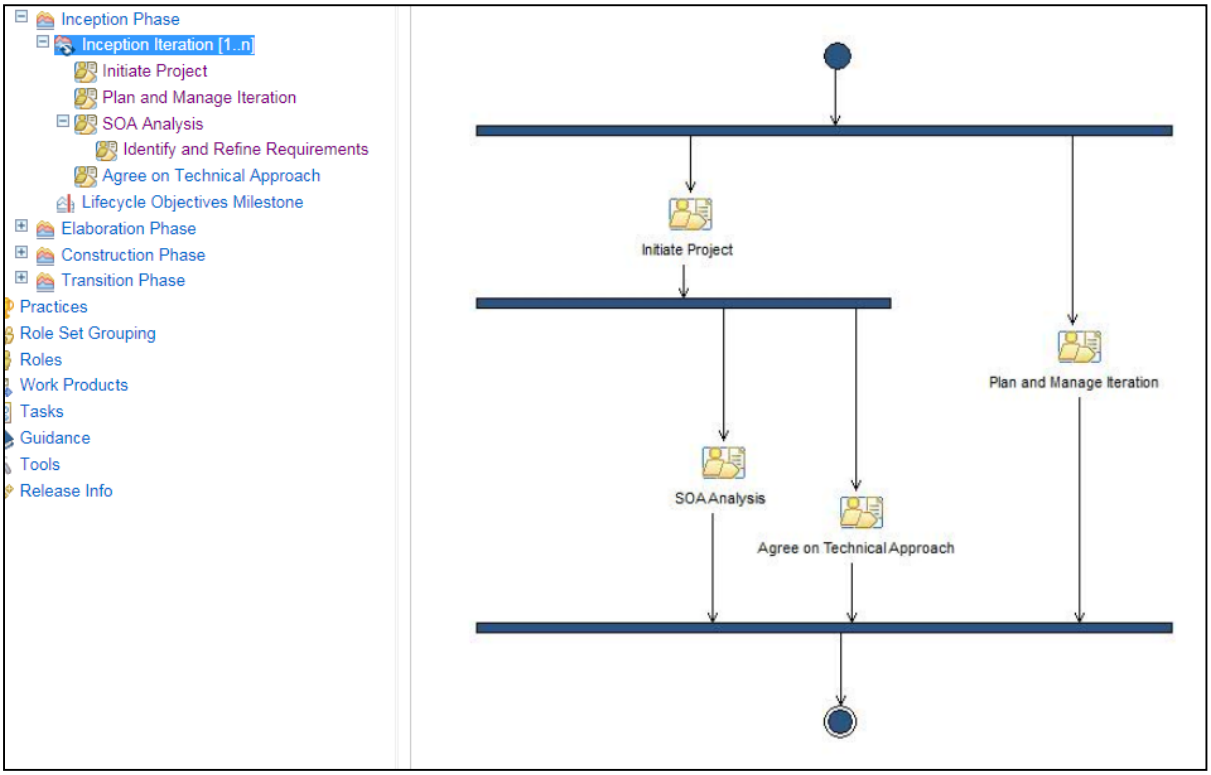


Figure 3.15 La phase d'initialisation avec SOA.
Adaptée de (*OpenUP*, 2010)

L'analyse orientée services est une analyse d'affaires et ne correspond pas seulement à l'analyse comme on l'entend généralement en développement et qui a plutôt lieu pendant la phase d'élaboration. Puisque la phase d'initialisation dans OpenUP correspond également à l'étude des exigences d'affaires, il est tout à fait logique d'y retrouver l'analyse orientée services qui se concentre également sur les exigences d'affaires, mais du point de vue d'une SOA.

3.6.2.2 La phase d'élaboration et la conception orientées services

À l'image de la phase précédente, certaines activités sont complémentaires à la conception orientée services. Ainsi, le développement d'un incrément, la planification de l'itération, la gestions des tâches non-planifiés et le test de l'incrément sont toutes des activités qui n'entrent pas en conflit avec l'activité de conception orientée services.

Par contre, il y a toujours l'identification et la raffinement des exigences ainsi que le développement de l'architecture qui sont susceptibles de chevaucher certaines activités de la conception orientée services. Encore une fois, l'analyse orientée services se distingue par l'utilisation d'artéfacts provenant d'EUP alors que les deux autres activités ont une perspective par projet, comme l'illustre la Figure 3.14.

On propose, comme dans la phase d'initialisation, d'imbriquer l'identification et le raffinement et des exigences, en plus du développement de l'architecture, dans la conception orientée services, tel qu'illustré dans la Figure 3.16.

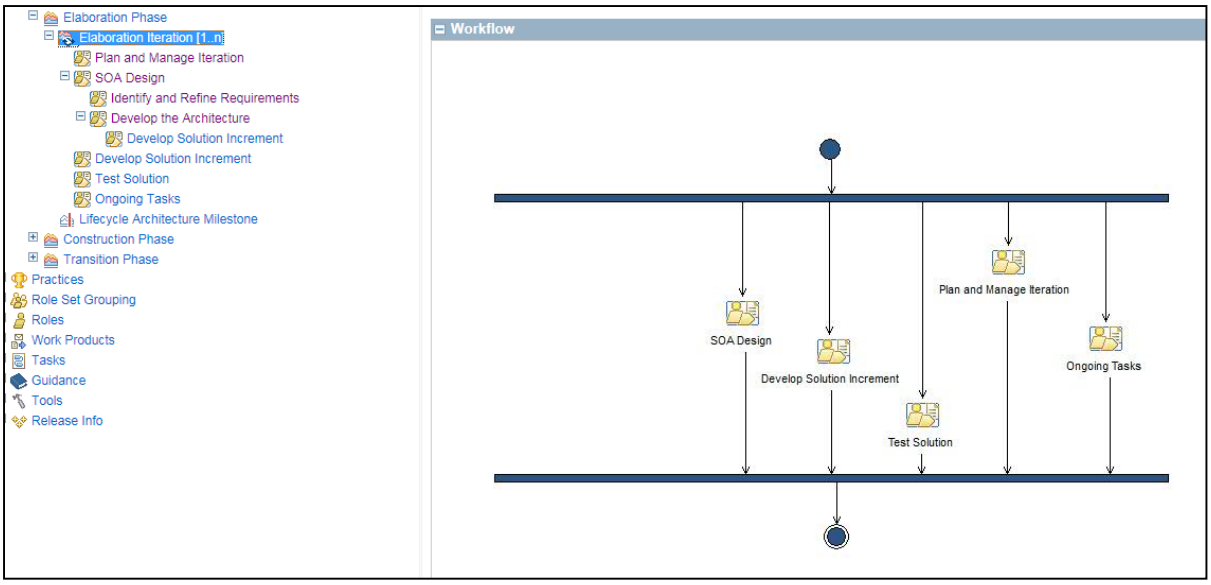


Figure 3.16 La phase d'élaboration avec la conception SOA.
Adaptée de (*OpenUP*, 2010)

3.7 Conclusions

Avec l'intégration des disciplines d'entreprises de (Scott W.Ambler, 2005), des activités orientées services d'analyse et de conception de (Erl, 2005) et des phases de développement d'OpenUP, on obtient une nouvelle méthode. Cette méthode ouverte et agile est documentée et publiée à l'aide de l'outil EPF. Un CD-ROM contenant la publication est joint en annexe de ce mémoire. À la suite d'une analyse comparative, plusieurs artéfacts d'entreprises se retrouvent liés aux activités d'analyse et de conception orientée services. De plus, OpenUP vient combler les manques de la méthodologie de Thomas Erl en ce qui a trait au cycle de développement logiciel.

Dans EPF, cette méthode se présente sous la forme de deux cycles distincts, puisqu'OpenUP a un cycle par projet alors qu'EUP a un cycle continue. La Figure 3.17 nous présente ces deux cycles avec leurs principales activités.

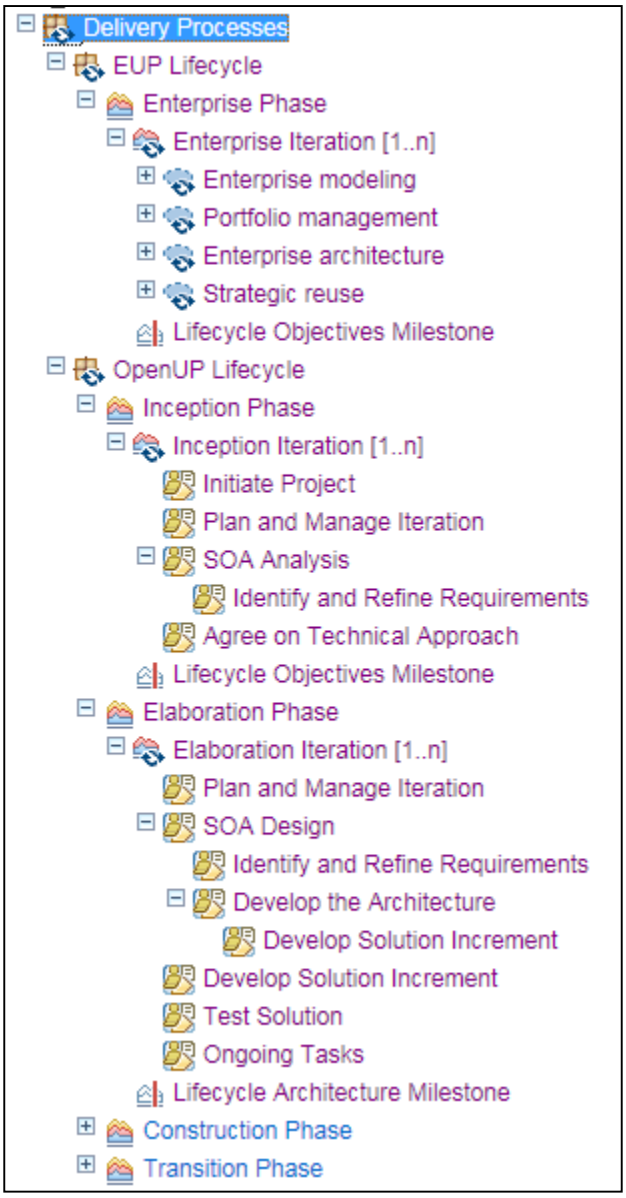


Figure 3.17 Cycles de la nouvelle méthode

Les Figure 3.18 et Figure 3.19, quant à elles, nous présente les artéfacts d'EUP et de SOA qui ont été ajoutés à OpenUP. Certains des artéfacts d'EUP, comme nous l'avons vu dans ce chapitre, sont utilisés pour analyser et concevoir la SOA, notamment le modèle des processus d'affaires.

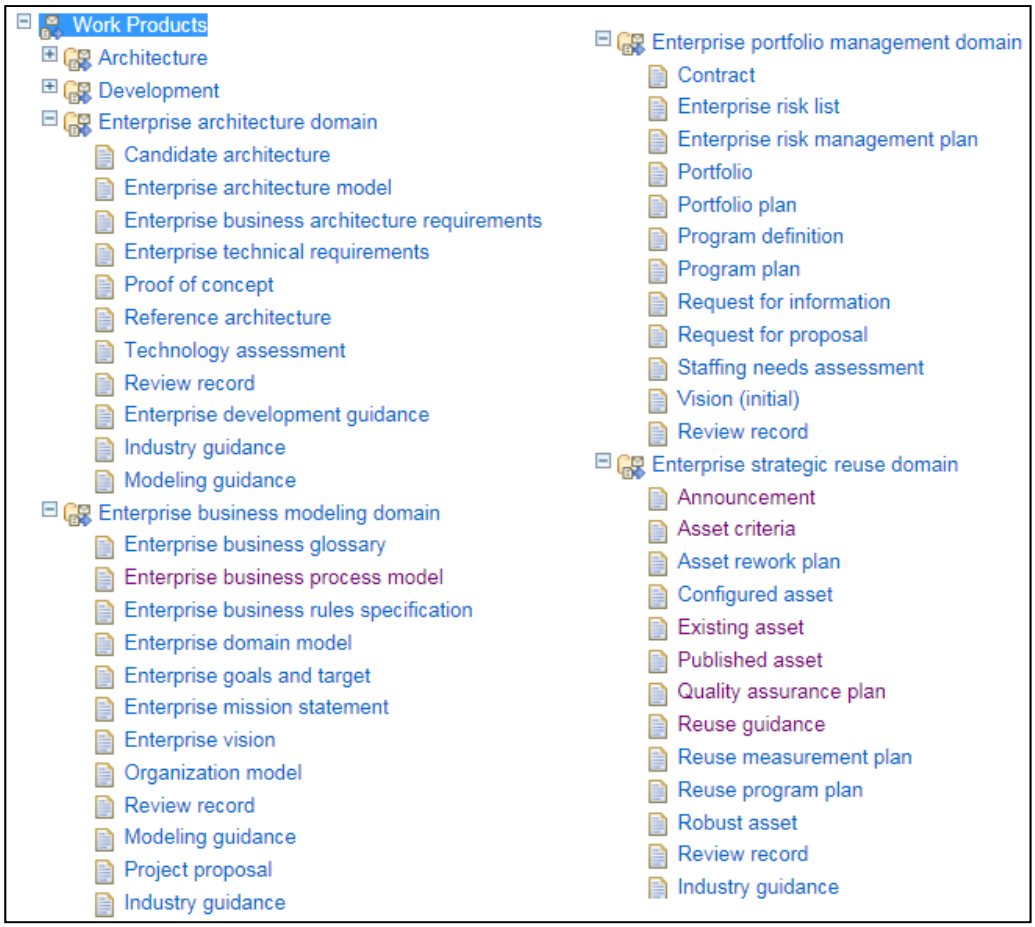


Figure 3.18 Les artéfacts d'EUP

Figure 3.19 Les artéfacts SOA

Avec EUP vient un ensemble de rôles qui a été divisé en deux groupes. Les rôles communs sont les rôles génériques qui participent aux différentes disciplines d'entreprise alors que les rôles du groupe entreprise sont les rôles principaux des quatre disciplines d'entreprise. Ces rôles sont illustrés dans la Figure 3.20.

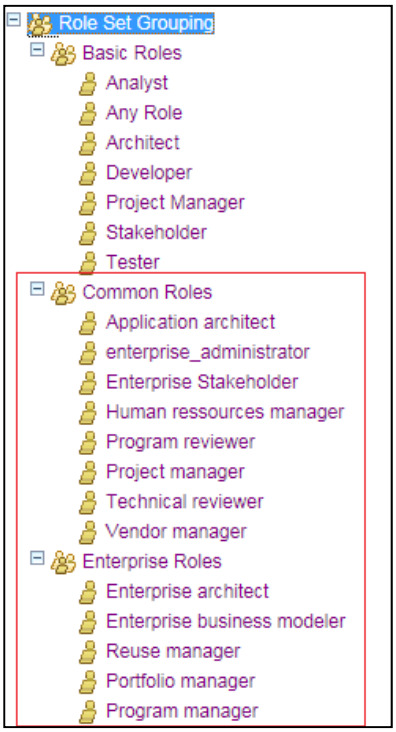


Figure 3.20 Les rôles provenant d'EUP

Cette méthode intégrée, présentée à la Figure 3.21, permet la réalisation d'une SOA en offrant une perspective plus globale de l'organisation. Cette perspective est nécessaire pour développer une solution SOA qui soit alignée avec les besoins d'affaires de l'organisation.

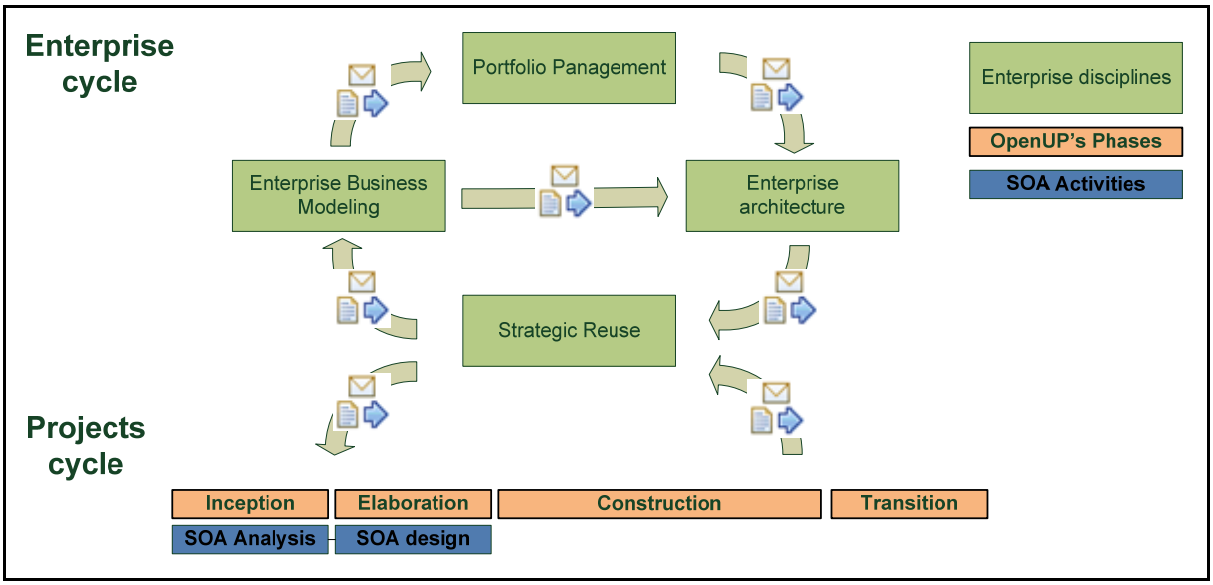


Figure 3.21 Méthode intégrée.

La prochaine étape est d'expérimenter cette méthode. Le prochain chapitre décrit deux petits projets qui ont été réalisés avec un partenaire industriel. Ces projets ne permettent pas de valider si la méthode permet de réaliser une SOA. Ils permettent cependant de valider si la nouvelle méthode permet de structurer une démarche menant à une SOA au sein d'une petite organisation.

CHAPITRE 4

EXPÉRIMENTATION DE LA MÉTHODOLOGIE

4.1 Introduction

Ce chapitre décrit l'utilisation de la méthodologie dans le cadre de deux projets pilotes. Dans le premier cas, on souhaite intégrer un nouveau service Web à une application développée par la CHQ pour ajouter des fonctionnalités de suivi de facture que le système comptable n'offre pas. Le système comptable *Virtuo* provient d'un fournisseur tiers et celui-ci a récemment un service Web pour la gestion des factures. À ce jour, il est impossible de créer des factures dans le système comptable autrement qu'en utilisant l'interface de saisie et par conséquent, il était impossible d'automatiser cette fonction.

Dans le second cas, la CHQ souhaite la création d'un service pour exploiter les données contenues dans un référentiel développé à l'interne. Ce référentiel contient les données sur tous les immeubles du réseau de la santé ainsi que sur les occupants qui logent dans ces immeubles. Il y a deux justificatifs à ce projet. Premièrement, les informations contenues dans ce référentiel sont utilisées par la plupart des systèmes d'information de la CHQ. Il y a un fort potentiel de réutilisation tout en normalisant la manière d'accéder à cette information. Deuxièmement, des partenaires externes de la CHQ souhaiteraient pouvoir accéder facilement à l'information contenue dans le référentiel. Les règles de sécurité actuelles du réseau rendent difficile un accès aux systèmes internes sans une intervention du MSSS. Un service Web répondrait au besoin puisque la sécurité du transport des données n'est pas critique et que l'information pourrait facilement circuler via le protocole HTTP.

La CHQ a participé au processus de sélection de ces projets pilotes. L'objectif est d'apprécier qualitativement la méthodologie développée dans le cadre de ce projet de recherche. Cette

expérimentation est réalisée lors de quelques itérations, sur deux projets et sur une courte période de temps d'environ deux mois.

Les disciplines d'entreprises sont exécutées lors des premières itérations afin de faire l'inventaire de la documentation et des artéfacts disponibles. Ces artéfacts permettent d'identifier des projets SOA candidats. Les phases d'initialisation et d'élaboration sont ensuite décrites pour les deux projets pilotes qui ont été identifiés à la suite des premières itérations. Les phases de construction et de transition sont exclues. Elles n'ont pas été réalisées. Mais puisque l'objectif de ce mémoire est d'intégrer un cadre d'architecture d'entreprise, les phases de développement et les activités propres aux architectures orientées services et que les phases de construction et de transition sont identiques, que ce soit pour une SOA ou un développement logiciel standard, on croit que la démarche couvre les éléments les plus critiques de la méthode.

4.2 Les outils utilisés

Pour la réalisation des projets pilotes, il fallait choisir une plateforme de développement pour programmer les services Web. Afin de respecter l'environnement technologique actuel, les outils sélectionnés sont ceux déjà utilisés par la CHQ.

4.2.1 Microsoft Visual Studio 2008

Visual Studio 2008 est une plateforme de développement de la compagnie Microsoft (*Microsoft Visual Studio 2008*, 2010). Le partenaire industriel utilise cette plateforme dans l'ensemble de ses développements logiciels. Les travaux seront réalisés sous cette plateforme afin de préserver les choix technologiques de l'organisation.

Rappelons que les outils utilisés pour mettre en place une SOA ne sont pas neutres technologiquement. Ce sont les standards utilisés par ces outils pour l'interfaçage et le transport qui assureront l'interopérabilité entre les différentes plateformes.

4.2.2 Windows Communication Foundation

Windows Communication Foundation (WCF) est le cadriciel de développement délivré par Microsoft pour la mise en place de son architecture orientée services (*Windows Communication Foundation*, 2010). WCF est destiné à remplacer la solution précédente de Microsoft basée sur les fichiers .asmx et la librairie WSE (*Web Services Enhancement*). Cette solution avait comme principal défaut de forcer l'utilisation des services à l'aide d'un serveur Web. WCF élimine cette contrainte en permettant de créer des services pouvant, en plus d'être hébergé sur un serveur web, être hébergés comme un service système, dans un fichier exécutable ou une application console (Klein, 2007).

Pour l'interopérabilité avec des plateformes différentes de Microsoft, le projet « Tango » a été mis sur pied. C'est le résultat d'une collaboration entre les ingénieurs de Microsoft et de la compagnie Sun qui se sont assuré d'implémenter les mêmes spécifications afin de permettre aux applications J2EE d'êtres compatibles avec celles de Microsoft (Carr, 2006). WCF utilise cependant une terminologie légèrement différente comme on peut le constater dans le Tableau 4.1.

Tableau 4.1 La terminologie utilisée dans WCF
Tiré de (Peiris et Mulder, 2007)

Standard	WCF	Description
Entity	*DataContract*	Définit les données et leur type (XSD)
Message	*MessageContract*	Définit le format du message (WSDL)
Interface	*ServiceContract*	Expose les opérations disponibles (WSDL)
Transport	*Binding*	Définit le protocole de transport utilisé (WSDL)
EndPoint	*Deployment*	Point d'entrée du service (WSDL)

4.3 Exécution des activités d'entreprises

Lorsque la majeure partie de la méthodologie a été documentée, le travail avec la CHQ a débuté. Ce sont les activités d'entreprises qui ont été abordées lors de la première itération. Les activités d'entreprises sont communes aux deux projets. Il a fallu faire l'inventaire de la documentation existante. Dans un contexte agile, il ne faut pas créer toute la documentation suggérée, mais utiliser ce qui existe et ne maintenir que ce qui peut être utile.

Les disciplines de modélisation d'affaires, d'architecture d'entreprise et de stratégie de réutilisation contiennent toutes une activité de support aux membres des équipes de projet. Même si cette activité fait partie de la méthodologie, elle ne fait pas partie des activités expérimentées. Aucun travail de support n'a été réalisé auprès des équipes du partenaire.

4.3.1 Modélisation d'affaires de l'entreprise

La modélisation d'affaires est la première discipline qui a été exécutée. Le premier effort a été d'identifier la documentation d'affaires qui était disponible. Pour la modélisation d'affaires de l'entreprise, l'auteur et le partenaire industriel ont recensé les artéfacts suivants :

L'organigramme

L'organigramme, bien qu'incomplet pour être considéré comme un modèle complet de l'organisation, fournit une information sur les structures administratives. On peut voir l'organigramme à la Figure 4.1.

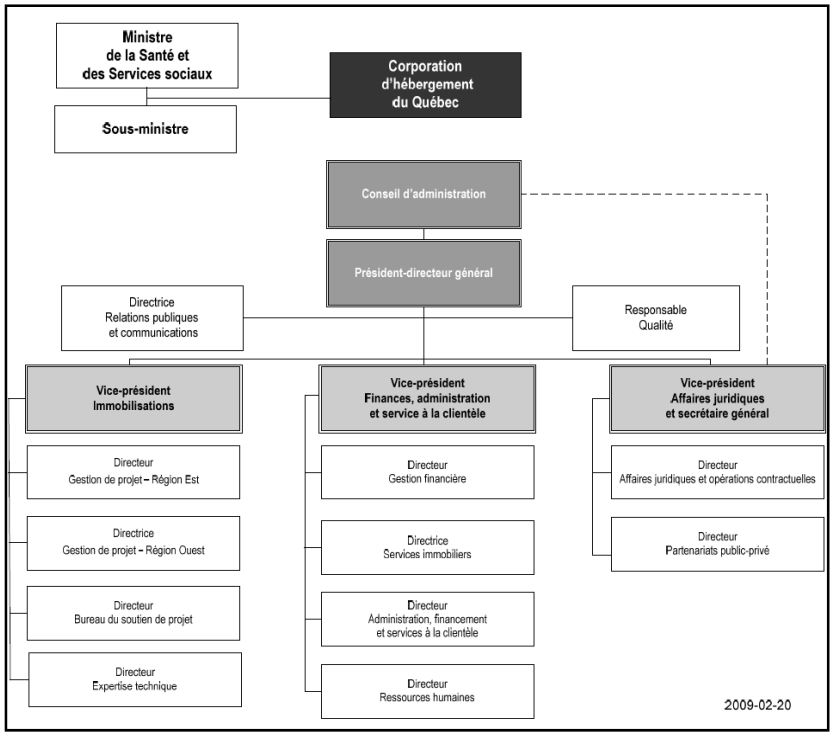


Figure 4.1 Organigramme de la CHQ.
Tirée de (*Corporation d'hébergement du Québec*, 2010; *OpenUP*, 2010)

Les processus d'affaires

Les principaux processus d'affaires sont déjà documentés dans des diagrammes d'activités. Ces diagrammes ont été réalisés lors d'un processus de normalisation ISO 9001 il y a quelques années. Cette documentation ne comprend pas les règles d'affaires, mais intègre les applications qui sont sollicitées dans la réalisation du processus ainsi que les rôles qui sont impliqués. Ces processus sont illustrés plus loin.

Le bilan annuel et le plan directeur

Le plan directeur et le bilan annuel fournissent d'importantes informations sur la vision, la mission et les buts de la CHQ pour les prochaines années. Ces documents sont disponibles en version électronique et accompagnent ce mémoire.

Il n'y avait pas, par contre, de modèle du domaine pour l'entreprise. Il n'y avait pas non plus de glossaire, ou d'autres supports permettant de partager la terminologie et les concepts propres à l'organisation. Cependant, un effort de normalisation de la terminologie a été fait il y a quelques années, lorsque l'organisation a développé ses premiers systèmes référentiels. Les systèmes existants utilisent donc tous la même terminologie.

À l'aide des modèles de processus d'affaires et du plan directeur, l'auteur et le partenaire industriel sont en mesure d'identifier avec beaucoup plus de précision quels sont les projets SOA susceptibles d'être alignés avec la stratégie de l'organisation. Par contre, il y a une lacune pour la définition de la terminologie. Il faut se référer aux personnes ressources pour connaître la signification des éléments du domaine d'affaires.

À la fin de cette itération, et surtout à l'aide des modèles de processus d'affaires, le partenaire industriel a déjà une bonne idée des options d'automatisations susceptibles d'avoir un intérêt d'affaires.

4.3.2 Gestion du portefeuille

Pour la discipline de gestion du portefeuille, il faut identifier les applications existantes. Si ces applications sont regroupées dans un domaine particulier, on peut également identifier les programmes s'y rattachant. Il est nécessaire d'identifier l'environnement technologique actuel ainsi que les projets qui sont en cours.

Le partenaire industriel avait en main un diagramme répertoriant les principaux systèmes existants (*voir* Figure 4.2). Plusieurs plus petites applications qui visent peu d'employés sont absentes de ce diagramme. Le partenaire avait également un diagramme de l'infrastructure technologique (*voir* Figure 4.3). Le système comptable Virtuo avec lequel nous devons interfacer est au centre des applications de la CHQ.

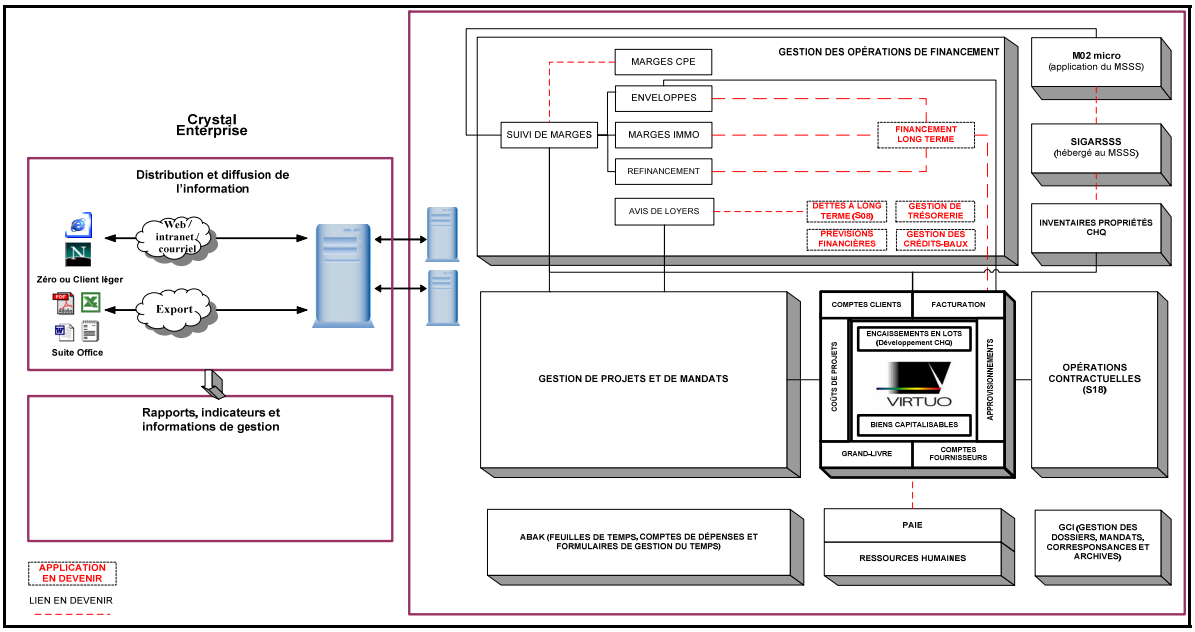


Figure 4.2 Vue sommaire des applications de la CHQ.
Tirée de (*Corporation d'hébergement du Québec*, 2010; *OpenUP*, 2010)

L'activité de gestion des contrats n'est pas pertinente dans ce cas-ci. Il n'y a pas d'audit formel pour les fournisseurs actuels. On a identifié deux principaux fournisseurs pour le système de la comptabilité et le système de gestion de projet. Les autres systèmes sont soit développés à l'interne ou ont une portée plus restreinte dans l'environnement technologique de l'organisation.

La gestion du risque au sein de l'entreprise n'est pas réalisée de manière formelle. Il n'y a donc pas de plan de gestion du risque ni d'audit de ces risques. L'organisation a pourtant connu régulièrement des changements importants dans la direction au cours des dernières

années. Plusieurs personnes ressources sont susceptibles de partir à la retraite prochainement. Ce type de risque touche l'ensemble des projets en cours sans être adressé.

4.3.3 Architecture d'entreprise

À l'aide des documents identifiés lors des disciplines précédentes, on peut dresser une vue assez complète de l'architecture technologique de l'organisation. L'infrastructure et les principales applications sont identifiées. Les technologies utilisées pour chacun des systèmes sont également identifiées. Les départements qui sont impliqués dans l'exploitation de ces systèmes viennent compléter l'information. La Figure 4.3 présente l'infrastructure technologique et les principales applications de la CHQ.

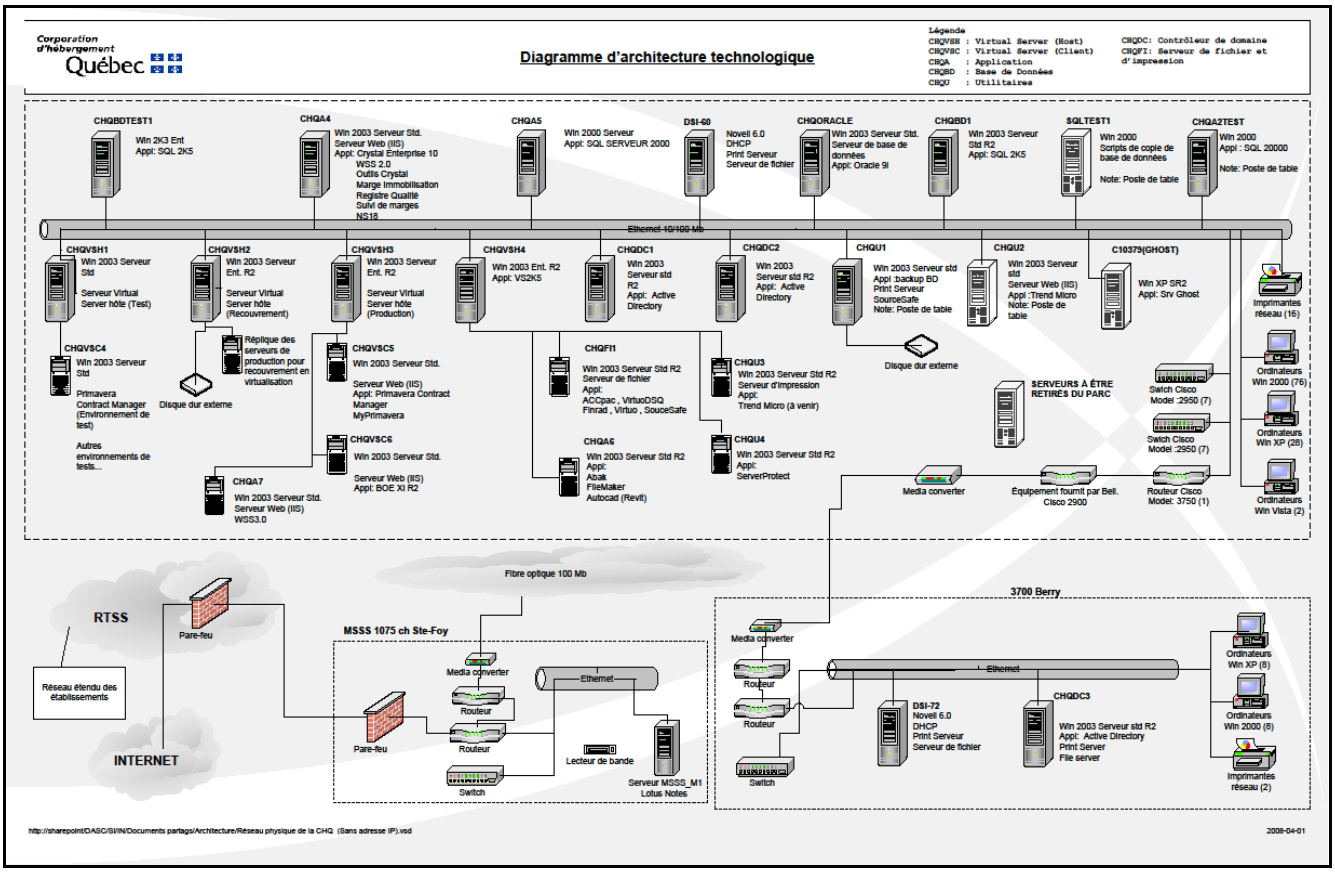


Figure 4.3 Architecture technologique de la CHQ.
Tirée de (Corporation d'hébergement du Québec, 2010)

Il n'y a pas de vue formelle de l'architecture candidate. Par contre, la gestion du portefeuille permet d'identifier les programmes et les projets à venir et d'avoir une idée des principales composantes de ces architectures candidates.

(Scott W.Ambler, 2005) a identifié cinq vues particulièrement importantes pour apprécier l'architecture d'une entreprise. La vue logicielle, la vue réseau, la vue des flux de travail, la vue du domaine et la vue des données. Lors de cette première itération, l'auteur a été en mesure d'identifier des documents existants pour toutes ces vues, sauf pour le domaine. Avec la Figure 4.3 on visualise les vues logicielles, réseau et données alors que la figure Figure 4.4 illustre la vue flux de travail pour le premier projet.

L'auteur a jugé qu'il était encore trop tôt pour identifier ou définir des architectures de références. Les architectures de références sont des éléments qui pourront être réutilisés pour réaliser l'architecture d'une nouvelle application.

4.3.4 Stratégie de réutilisation

L'itération portant sur les activités d'entreprises se conclut avec la discipline de stratégie de réutilisation. L'auteur et le partenaire industriel ont analysé les différents artéfacts identifiés dans les disciplines précédentes afin d'identifier des sources potentielles de réutilisation. Une SOA doit contenir des services qui sont facilement réutilisables. Cette volonté est cependant limitée par la difficulté, mentionnée par (Scott W.Ambler, 2005), d'identifier à priori les composantes qui seront réutilisables. Afin de contourner ce problème, il a été proposé que les projets de développement soient des initiatives de réingénierie des systèmes existants. Il est ainsi plus facile d'identifier les éléments ayant une forte valeur de réutilisation.

D'autres critères ont également été considérés. L'un des critères importants a été d'identifier un projet ayant une valeur d'affaires aux yeux du partenaire industriel. Autrement, son implication dans le projet risquait d'être limitée de par son manque de motivation. Il fallut également choisir un projet qui n'était pas trop complexe. Ce projet jouait le rôle de projet pilote et servait de base d'apprentissage pour des projets subséquents.

Lors de la réutilisation stratégique, on planifie pour le premier projet d'obtenir un actif à partir d'un fournisseur externe. Ce projet en est donc un d'intégration. Dans le second projet, on développe l'actif à l'interne. À ce stade, la CHQ n'est pas en mesure d'identifier d'autres actifs existants qui pourraient accomplir la tâche. Il est trop tôt pour mesurer le programme de réutilisation, tout autant que l'activité de retrait. L'activité de publication de l'actif se déroulera dans la phase de transition du cycle de développement.

4.4 Projet d'intégration d'un service provenant d'un fournisseur externe

Il y a à la CHQ un processus pour la création et le suivi des factures. Ce processus a été documenté et est présenté à la Figure 4.4. Un agent saisit une nouvelle facture dans le système comptable. Par la suite, les étapes 2 et 3 de vérifications et d'approbations sont réalisées dans un système de suivi des factures. Finalement, les étapes 4 et 5 sont réalisées dans le système comptable.

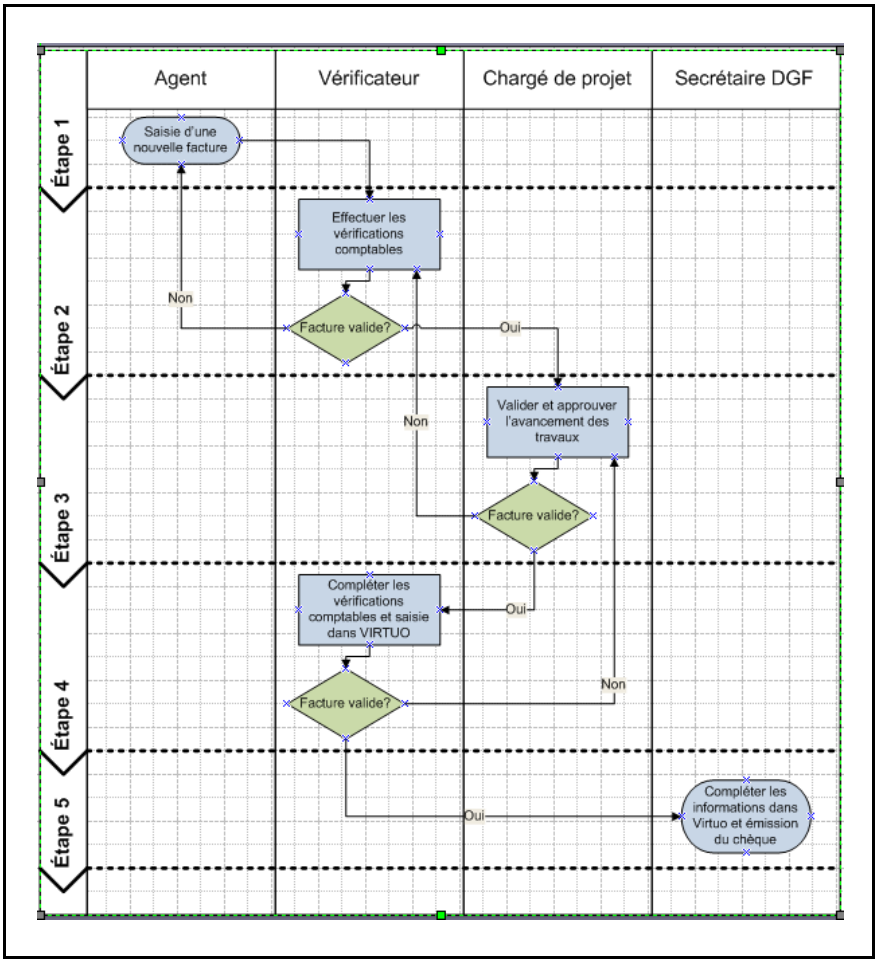


Figure 4.4 Processus de création et de suivi des factures.
Tirée de (Corporation d'hébergement du Québec, 2010)

La création de facture dans le système comptable ne peut se faire qu'à l'aide de l'interface graphique de l'application Virtuo. Il est impossible d'automatiser la gestion des factures dans les autres systèmes. Même si le personnel technique a des accès à la base de données Oracle du système comptable, ils n'ont pas le modèle de données ni les règles d'intégrité. Le risque de corruption de la base de données est trop élevé.

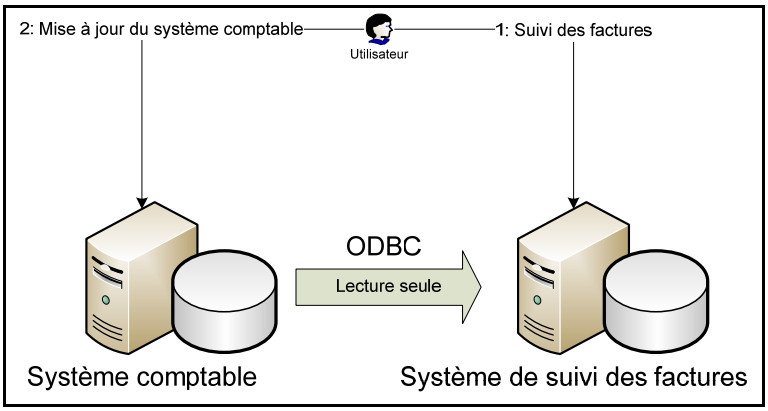


Figure 4.5 Projet d'intégration - Avant.

Un système interne a été développé par les programmeurs de la CHQ au cours des derniers mois afin d'assurer un suivi des factures, le système comptable n'étant pas suffisant pour répondre aux besoins. Ce système interne puise certaines informations provenant du système comptable. Mais il est impossible de mettre à jour le système comptable lorsqu'un utilisateur modifie la facture dans le système de suivi. Il faut alors ouvrir l'application comptable Virtuo et modifier une seconde fois les informations (*Voir* Figure 4.5). Il y a une double-saisie qui peut devenir une source d'erreur.

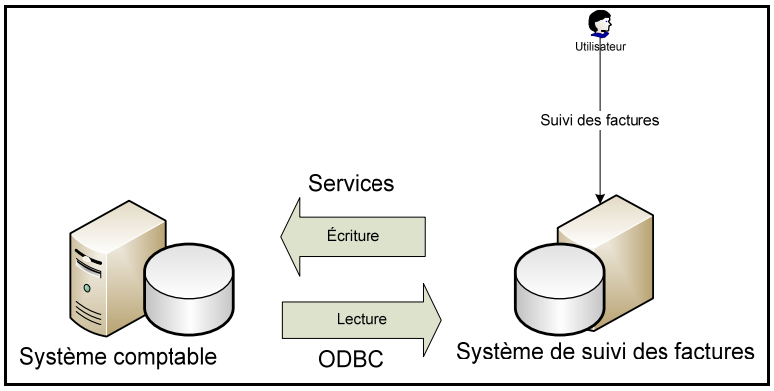


Figure 4.6 Projet d'intégration - Après.

Le fournisseur du système comptable a développé récemment un service Web pour la gestion des factures dans le cadre d'un projet d'implantation chez un autre client. Il a été entendu que le partenaire pourrait intégrer le service Web dans son environnement dans le cadre d'un projet pilote (*Voir* Figure 4.6).

4.4.1 La phase d'initialisation

Le projet a été lancé suite à l'entente d'utilisation du service avec le fournisseur. Le service a été installé sur un serveur Web par le fournisseur et une documentation pour l'utilisation du service a été fournie à la CHQ. Il n'y a aucun service candidat à identifier puisque c'est le fournisseur du système comptable qui fournit le service Web. Les intervenants sont le fournisseur, l'analyste du système de suivi des factures et le responsable du suivi des factures, relevant du département de la comptabilité.

4.4.2 La phase d'élaboration

Lors de la phase d'élaboration, la première étape a été d'identifier et de minimiser les principaux risques. Deux risques importants ont été identifiés. Il fallut dans un premier temps s'assurer que la CHQ maîtrisait la technique de consommation du service Web de gestion des

factures. Il fallut ensuite s'assurer que le service Web répondait à toutes les exigences d'affaires touchant les processus de facturation.

Des tests de fonctionnement du service ont été effectués avec succès afin de minimiser le risque. Des tests préliminaires avec le système de suivi des factures ont été faits également. La CHQ était assez sûre de pouvoir intégrer le service de gestion des factures à son infrastructure. Ce fut la preuve de concept. Le code de ces tests est disponible en ANNEXE X.

Malheureusement, la CHQ a constaté que le service Web de gestion des factures ne réalise pas certaines caractéristiques importantes. Le service ne peut pas être déployé par l'équipe dans ces conditions. Aucune entente n'est encore intervenue pour le développement des caractéristiques manquantes par le fournisseur. Malgré cela, la CHQ a décidé d'aller de l'avant avec ce projet pilote afin de valider la technologie. Il sera peut-être question par la suite de faire développer les caractéristiques manquantes par le fournisseur.

Dans cette itération de la phase d'élaboration, il est inutile de réaliser l'activité de conception orientée services puisque c'est une intégration dans un système existant. Il n'est pas nécessaire de développer davantage les couches de services, car le potentiel de réutilisabilité est minime et le service a déjà découplé la gestion des factures et l'accès à la base de données. Pour terminer, le service a été intégré dans une version pilote de l'application de suivi des factures avec succès.

4.5 Projet de développement d'un service à l'interne

La CHQ possède une application interne contenant de nombreuses informations nominatives sur le parc immobilier du réseau de la santé. On retrouve également des détails sur la condition de ces immeubles, sur leurs occupants, leur mission, etc.

Ce référentiel est utilisé dans la plupart des systèmes patrimoniaux de l'organisation; système de gestion des contrats, système de gestion de projet, système comptable et plusieurs autres petits systèmes internes. Selon le cas, des liens ont été créés afin de permettre aux bases de données de ces applications d'être lié à ce référentiel. Parfois ce sont des liens ODBC, parfois ce sont des procédures stockées et parfois l'intégration a été faite directement dans le code de l'application (*Voir* Figure 4.7).

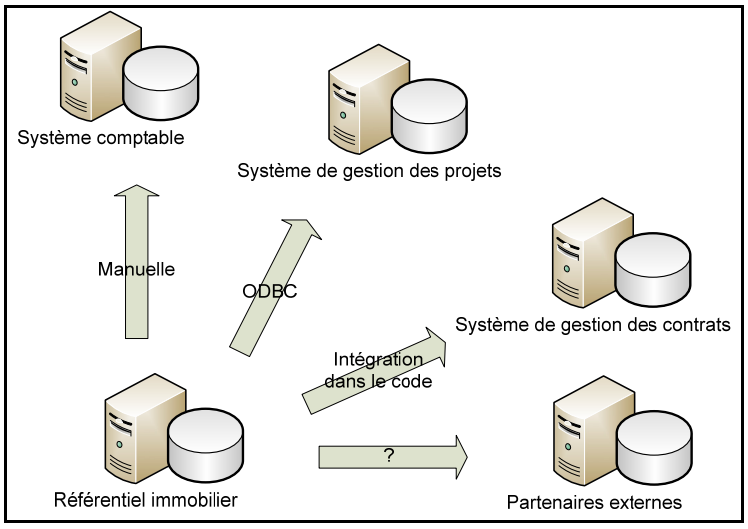


Figure 4.7 Projet de développement d'un service interne - Avant.

De plus, il a été reconnu que le référentiel maintenu par la CHQ était le plus à jour. Des organisations externes souhaitent pouvoir lier leurs systèmes à ce référentiel. Ce lien permettrait de limiter la double saisie et le risque d'erreur, en plus de réaliser des économies d'échelle importantes sur l'effort déployé pour maintenir ces informations à jour dans plusieurs bases de données distinctes.

Pour ces raisons, il est intéressant de développer un ensemble de services qui permettront au partenaire de faire une gestion du référentiel à partir des nombreuses applications qui l'utilisent. En outre, ces services permettront aux organisations externes d'interroger le référentiel et de l'intégrer dans leurs propres applications, tout en limitant la complexité que

ce genre de partage implique lorsqu'on utilise un autre canal que celui utilisé par les services Web (*Voir* Figure 4.8).

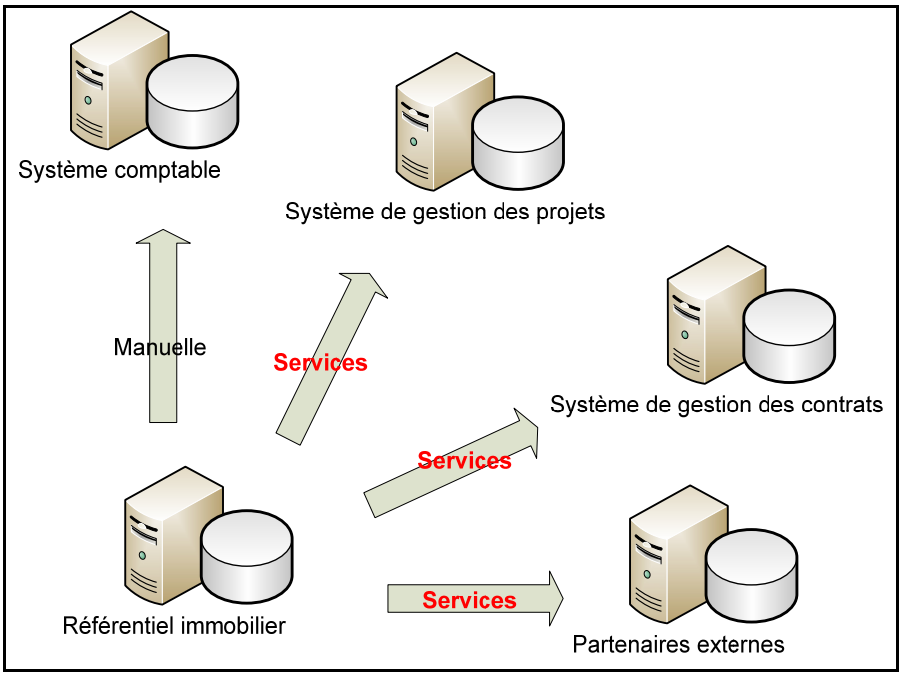


Figure 4.8 Projet de développement d'un service interne - Après.

4.5.1 La phase d'initialisation

Dans ce projet, les intervenants sont les analystes-programmeurs qui sont responsables des liens attachant le référentiel aux autres systèmes. Le résultat sera transparent pour les utilisateurs puisque la fonctionnalité est déjà présente à travers les liens qui ont été créés précédemment.

Cette fois, il y a une analyse orientée services à faire. Les différents processus d'affaires qui utilisent l'information du référentiel doivent être analysés afin d'identifier les opérations candidates du service. À priori, les services candidats qui sont identifiés se situent dans la couche des services applicatifs pour se connecter à la base de données du référentiel et dans

la couche des services d'affaires entités pour représenter les objets du domaine d'affaires. Dans ce cas-ci, l'entité d'affaires est l'objet du domaine « immeuble ».

Lors de l'analyse orientée services, un service entité « Immeuble » a été identifié. Pour ce service, quelques opérations ont été également identifiées :

- Chercher une liste d'immeubles par identifiant.
- Chercher une liste d'immeubles par adresse.

Un seul service applicatif a également été identifié, celui nécessaire à la connexion à la base de données et permettant de récupérer les différentes informations provenant de plusieurs tables. Dans sa première version, seules ces deux opérations ont été identifiées. Parce que ce besoin est simple, aucun service de tâches ni aucun service d'orchestration n'a été identifié.

La CHQ a décidé d'utiliser la plateforme de développement habituelle, afin de préserver l'homogénéité de son environnement. Les services sont développés à l'aide de Visual Studio 2008 et de la librairie WCF.

4.5.2 La phase d'élaboration

Lors de la phase d'élaboration, la CHQ a défini ses exigences d'affaires. Elle souhaite obtenir un service permettant d'obtenir l'information sur un immeuble. On doit pouvoir mettre à jour les informations de cet immeuble et finalement, on peut le supprimer de la base de données.

Le risque le plus important qui a été identifié est la possibilité de modifier ou supprimer les informations des immeubles à partir de ce service. Il a été décidé dans un premier temps d'offrir uniquement une opération de récupération des données. Dans une itération ultérieure, il faudra inclure des options de sécurité afin de contrôler les opérations d'édition.

Lors de la conception du service applicatif, la CHQ a constaté que WCF offrait des automatismes (à l'aide de classes générées par Visual Studio 2008) permettant de faire abstraction de la connexion à la base de données. Le service applicatif se contente donc d'interroger la base de données, déléguant la connexion à une classe générée.

On retrouve le contrat WSDL de gestion des immeubles à l'ANNEXE I. Dans l'ANNEXE II et l'ANNEXE III on retrouve les enveloppes SOAP utilisées pour chacune des options de recherche. Dans l'ANNEXE IV se trouve le contrat de données pour la gestion des immeubles (schéma XSD). L'ANNEXE V contient la définition de la classe alors que l'ANNEXE VI contient l'interface pour la gestion de l'immeuble. Finalement une interface et une classe pour la récupération des données sont documentées dans les ANNEXE VII et ANNEXE VIII. Cette dernière classe est utilisée pour interfacer avec la base de données.

Cela résulte en un service d'affaires permettant de récupérer les informations d'un immeuble. Ce service d'affaires utilise un service applicatif qui fait le lien entre le service d'affaires et la classe générée par la plateforme de développement pour se connecter à la base de données.

Pour ce projet pilote, aucune extension WS-* particulière n'est utilisée. Dans une itération ultérieure, des extensions pour la sécurité et le cryptage des données pour les fournisseurs externes devront être intégrées. Le profil de base du WS-I est une contrainte d'interopérabilité qui a été respectée. Les partenaires externes sont susceptibles d'utiliser des environnements différents de ceux de la CHQ. Ce projet pilote s'est limité à une seule itération. Le service, quoique fonctionnel, n'a pas été déployé. La personne à la CHQ qui assurait le lien avec ce projet est partie et le projet pilote a été abandonné.

4.6 Conclusions

Dans cette partie, une itération pour chacune des disciplines d'entreprises a été exécutée. Lors de cette itération, nous avons été en mesure d'identifier plusieurs artéfacts d'entreprise

existants qui facilitent l'alignement de projets SOA avec les besoins d'affaires de l'organisation.

Lors de la définition des exigences d'affaires, le plan directeur et le bilan annuel ont fourni les objectifs, les buts, la mission et la vision de l'entreprise. Ces informations permettent de s'assurer que les projets rejoignent les plans à court et moyen terme de l'organisation. Le modèle des processus d'affaires est également un artéfact très important. Il permet de comprendre le flux de travail et d'identifier les rôles qui participent à sa réalisation.

Lors de l'identification des systèmes automatisés et automatisables, le portefeuille des applications a été évidemment très utile. Dans ce cas-ci, le modèle des processus d'affaires de la CHQ documente les systèmes qui sont sollicités dans le processus. Ce modèle incluait certaines règles d'affaires en commentaires, sans pour autant être un document distinct.

Lors de la modélisation des services candidats, tous les artéfacts du Tableau 4.2 ont été utilisés. La décomposition du processus d'affaires a nécessité le modèle des processus d'affaires. Le portefeuille a permis d'identifier les services candidats applicatifs. Même s'il n'y a pas d'actif existant, on peut présumer qu'une vérification dans ce catalogue serait faite. Le glossaire et le modèle du domaine sont des artéfacts qui n'ont pas d'existence propre pour l'instant, mais plusieurs termes d'affaires ont dût êtres définis et associés pendant l'analyse.

Tableau 4.2 Les artéfacts d'entreprise utiles lors de l'analyse orientée services

Disciplines d'entreprises	**Artéfacts**
Gestion du portefeuille	Portefeuille d'application
Modélisation d'affaires	Glossaire
Modélisation d'affaires	Modèle des règles d'affaires
Modélisation d'affaires	Mission de l'entreprise
Modélisation d'affaires	Modèle du domaine
Modélisation d'affaires	Modèles des processus d'affaires
Modélisation d'affaires	Objectifs et buts de l'entreprise
Modélisation d'affaires	Vision de l'entreprise

Disciplines d'entreprises	Artéfacts
Réutilisation stratégique	Actifs existants

Lors de la composition, on a défini les couches d'abstractions, les standards et les extensions qui allaient être utilisés. Le modèle d'architecture d'entreprise, même s'il n'a pas été utilisé lors de cette itération, a fourni une information importante sur les pratiques actuelles de l'organisation.

Lors de la conception des services d'affaires entités, le glossaire et le modèle du domaine ont été réutilisés suite à leur création lors de l'analyse (voir Tableau 4.2). Le portefeuille des applications et le modèle d'architecture de l'entreprise ont permis d'identifier les applications impliquées, les technologies qui étaient en jeux et les dépendances avec les autres systèmes. C'est ce que présente le Tableau 4.3

Tableau 4.3 Les artéfacts d'entreprise utiles lors de la conception orientée services

Disciplines d'entreprises	Artéfacts
Architecture d'entreprise	Le modèle d'architecture de l'entreprise
Gestion du portefeuille	Portefeuille d'application
Modélisation d'affaires	Glossaire
Modélisation d'affaires	Le modèle des règles d'affaires
Modélisation d'affaires	Modèle du domaine
Modélisation d'affaires	Modèles des processus d'affaires
Réutilisation stratégique	Actifs existants

Puisqu'aucun des deux projets ne nécessitait la conception de services de tâche ou de la conception d'un processus orientée services, aucun artéfact particulier n'a pu être identifié formellement comme un intrant à ces tâches. Néanmoins, le modèle des processus d'affaires, le portefeuille et les actifs existants sont susceptibles d'êtres aussi utiles que pour les tâches précédentes.

CHAPITRE 5

DISCUSSION

5.1 Introduction

Nous présentons dans ce chapitre un rappel des contributions ainsi que les limites de la solution décrite dans ce mémoire. On met en évidence certains facteurs ayant influencé l'expérimentation de la méthodologie. On donne également une appréciation de la méthodologie en identifiant les artéfacts d'entreprises qui ont contribué à la réalisation des projets de développement SOA.

5.2 Rappel des contributions

Dans un premier temps, la revue de littérature nous a permis de constater qu'il y a un réel problème d'intégration entre l'architecture d'entreprise, les pratiques orientées services et le développement logiciel. Nous n'avons trouvé aucune méthodologie qui intégrait ces trois aspects. Dans un second temps, il fallait sélectionner des méthodologies que nous pourrions intégrer. Finalement, il fallait documenter cette nouvelle méthode et l'expérimenter.

Les défis étaient notamment d'identifier les méthodologies candidates dans la revue de littérature susceptibles de remplir ce rôle. Ensuite, l'une des difficultés a été de réaliser une analyse comparative afin de proposer une intégration des trois méthodes qui ont été choisies. Dans le cadre de cet exercice, il fallait documenter EUP et la méthodologie de Thomas Erl dans EPF. EUP étant similaire à OpenUP dans la terminologie, l'exercice fut long, mais relativement simple, les artéfacts et les rôles étant déjà identifiés. Mais dans le cas de la méthodologie de Thomas Erl, bien que chaque étape soit clairement identifiée, il n'en était rien pour les artéfacts.

Nos contributions se résument à une nouvelle méthode ouverte et légère qui intègre les disciplines d'architecture d'entreprise, les pratiques orientées services et le développement agile. Cette méthode est documentée dans l'outil de modélisation des processus EPF. Plusieurs artéfacts utilisés dans EUP sont judicieusement réutilisés dans OpenUP, à l'intérieur des activités d'analyse et de conception orientées services. Une expérimentation limitée a également été réalisée afin de tester cette nouvelle méthode.

5.3 Limites de l'expérimentation

Une expérimentation limitée a été réalisée dans le cadre de ce mémoire afin de tester la nouvelle méthode. L'objectif de cette expérimentation était de voir si cette nouvelle méthode permettait de structurer une démarche menant à une SOA.

Les deux projets pilotes ont été réalisés avec un certain succès. Les phases d'initialisation et d'élaboration ont permis de bâtir les premiers éléments d'une SOA. Les étapes de (Erl, 2005) lors de l'analyse et de la conception orientée services offrent des activités qui facilitent le découpage des couches d'abstraction ainsi que l'analyse et la conception des services.

La méconnaissance des outils utilisés influence la qualité des artéfacts développés. Si la plateforme .Net de Microsoft n'était pas inconnue, WCF et le cadre proposé par Microsoft pour le développement des services étaient nouveaux. Sans doute qu'une fois les outils maîtrisés, il sera plus facile de se guider avec la méthode, surtout lors des activités orientées services.

La CHQ a son siège social à Québec alors que l'expérimentation a été réalisée à partir des bureaux de Montréal. Les bureaux de Montréal étaient adéquats pour accéder à l'environnement de développement. Mais le travail à distance a gêné les échanges avec les intervenants, surtout lors de l'exécution des disciplines d'entreprise, où plusieurs intervenants étaient sollicités.

Il est difficile de tirer des conclusions sur l'efficacité de la méthode sur un si court laps de temps et pour un si petit échantillon de projets. Il est également difficile de déterminer si, en absence des artéfacts d'entreprise, la CHQ aurait effectué les mêmes choix de projets et si ces projets se seraient déroulés différemment. Finalement, la transition vers une SOA n'est pas complétée et plusieurs autres projets seraient à prévoir au cours des prochaines années afin de conclure à une migration significative de l'architecture actuelle vers une SOA.

5.4 Recommandations

Il reste beaucoup de travail pour faire évoluer cette méthode. L'une des modifications qui complèterait le cycle de développement serait l'ajout dans OpenUP des cycles de production et de retrait selon (Scott W.Ambler, 2005) afin d'inclure la maintenance tout au long de la vie du logiciel ainsi que l'éventuel remplacement ou retrait des actifs désuets.

La méthode est très générique et il faut documenter davantage les nouvelles activités afin d'y intégrer de la documentation en support, et ce, autant pour les disciplines d'entreprises que pour les activités orientées services.

Les disciplines d'entreprises d'EUP ont été utilisées pour leur terminologie très similaire à OpenUP. Cependant, EUP demeure un cadre d'entreprise très marginal qui n'est pas très reconnu par la communauté. Une intégration des phases de développement avec un cadre d'entreprise plus populaire, par exemple TOGAF, faciliterait peut-être son adoption.

L'expérimentation ne permet pas d'évaluer les impacts de la réingénierie des systèmes vers une SOA pour l'entreprise. La compilation de données suite à la réalisation de plusieurs projets sur une plus longue période de temps serait susceptible de donner une information intéressante sur ces impacts et de prouver l'intérêt d'utiliser une approche globale et intégrée pour développer une SOA.

CONCLUSION

Nous n'avons trouvé aucune méthodologie dans la revue de littérature qui intégraient l'architecture d'entreprise, les pratiques orientées services et le développement logiciel. Nous avons créer et documenter une méthode à partir des méthodologies existantes et proposer une intégration de ces méthodologies.

Le principal livrable de ce projet de recherche est une méthode intégrant quatre des sept disciplines d'entreprises de (Scott W.Ambler, 2005) aux phases de développement agile d'OpenUP auxquelles ont été intégrées les activités orientées services d'analyse et de conception de (Erl, 2005). La méthode a été modélisée et documentée dans l'outil de composition des processus EPF. Auparavant, il a fallu documenter entièrement les méthodologies EUP et SOA dans EPF afin de pouvoir par la suite les intégrer avec OpenUP. Une analyse individuelle suivie d'une analyse comparative a permis d'intégrer ces trois méthodologies. Les artéfacts d'entreprise qui sont utiles dans l'analyse et la conception d'une SOA ont été identifiés lors de cette analyse comparative ainsi que dans l'expérimentation qui a suivi.

La méthode élaborée embrasse une vision holistique de l'architecture orientée services. On ne se limite pas à la définition des phases nécessaires au développement des services qui implémenteront cette architecture. On y ajoute et intègre également plusieurs disciplines de nature opérationnelles qui sont à l'extérieur du cycle des projets et qui offrent une vision globale de l'entreprise et des artéfacts susceptibles d'aider au développement.

Le contenu de cette méthode n'est pas nouveau. Les trois constituants que sont les disciplines d'entreprise de (Scott W.Ambler, 2005), les quatre phases de développement d'OpenUP et la méthodologie orientée services de (Erl, 2005) en sont les bases. L'originalité de ce projet de recherche est d'intégrer ces trois constituants en un seul, tout en conservant une terminologie commune à OpenUP et à RUP.

Cette intégration offre un cycle de développement enchâssé dans un cycle continu où sont réalisées un ensemble d'activités d'architecture d'entreprise. Parfois, ce seront ces activités d'entreprise qui génèrent les artéfacts, parfois ce seront les projets qui alimenteront les artéfacts d'entreprise. L'expérimentation de la méthodologie nous porte à croire que ce principe de vase communicant permet une meilleure connaissance de l'architecture de l'organisation, une meilleure orientation des projets de développement et un potentiel de réutilisation supérieure.

Le cycle de développement réutilise une terminologie qui est déjà très connue par la communauté des développeurs. On bénéficie également de la grande quantité de documentation disponible à l'intérieur du cadre de développement et de la gratuité d'OpenUP.

Le développement de services inclut des activités distinctes pour leur analyse et leur conception. Ces activités orientées services sont insérées dans la méthodologie d'OpenUP, dans les phases d'initialisation et d'élaboration.

Pour conclure, bien que plusieurs améliorations soient toujours possibles, nous obtenons à la fin de cet exercice une méthode intégrée et documentée, que ce soit directement dans EPF ou dans la littérature des méthodologies originales. La nouvelle méthode, bénéficiant d'un tel corpus, pourra donc être utilisée dans des projets SOA par toute organisation souhaitant avoir un cycle de développement qui intègre les activités d'entreprise.

ANNEXE I

WSDL DU SERVICE WEB IManageImmeuble

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:tns="http://tempuri.org/"
      xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
      name="ManageImmeuble"   targetNamespace="http://tempuri.org/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <xsd:schema targetNamespace="http://tempuri.org/Imports">
      <xsd:import namespace="http://tempuri.org/" />
      <xsd:import
      namespace="http://schemas.microsoft.com/2003/10/Serialization/" />
      <xsd:import
      namespace="http://schemas.datacontract.org/2004/07/WCF_S25" />
    </xsd:schema>
  </wsdl:types>
  <wsdl:message
      name="IManageImmeuble_searchImmeubleByAdresse_InputMessage">
    <wsdl:part name="parameters" element="tns:searchImmeubleByAdresse" />
  </wsdl:message>
  <wsdl:message
      name="IManageImmeuble_searchImmeubleByAdresse_OutputMessage">
    <wsdl:part name="parameters"
      element="tns:searchImmeubleByAdresseResponse" />
  </wsdl:message>
  <wsdl:message name="IManageImmeuble_searchImmeubleById_InputMessage">
    <wsdl:part name="parameters" element="tns:searchImmeubleById" />
  </wsdl:message>
  <wsdl:message name="IManageImmeuble_searchImmeubleById_OutputMessage">
    <wsdl:part name="parameters" element="tns:searchImmeubleByIdResponse"
/>
  </wsdl:message>
  <wsdl:portType name="IManageImmeuble">
    <wsdl:operation name="searchImmeubleByAdresse">
      <wsdl:input
wsaw:Action="http://tempuri.org/IManageImmeuble/searchImmeubleByAdresse"

      message="tns:IManageImmeuble_searchImmeubleByAdresse_InputMessage" />
      <wsdl:output
      wsaw:Action="http://tempuri.org/IManageImmeuble/searchImmeubleByAdres
seResponse"

message="tns:IManageImmeuble_searchImmeubleByAdresse_OutputMessage" />
    </wsdl:operation>
    <wsdl:operation name="searchImmeubleById">
```

```
      <wsdl:input
      wsaw:Action="http://tempuri.org/IManageImmeuble/searchImmeubleById"

      message="tns:IManageImmeuble_searchImmeubleById_InputMessage" />
      <wsdl:output
wsaw:Action="http://tempuri.org/IManageImmeuble/searchImmeubleByIdResponse"

      message="tns:IManageImmeuble_searchImmeubleById_OutputMessage" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="BasicHttpBinding_IManageImmeuble"
      type="tns:IManageImmeuble">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="searchImmeubleByAdresse">
      <soap:operation
soapAction="http://tempuri.org/IManageImmeuble/searchImmeubleByAdresse"
                    style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="searchImmeubleById">
      <soap:operation
      soapAction="http://tempuri.org/IManageImmeuble/searchImmeubleById"
                    style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ManageImmeuble">
    <wsdl:port name="BasicHttpBinding_IManageImmeuble"
      binding="tns:BasicHttpBinding_IManageImmeuble">
      <soap:address
      location="http://localhost:8080/WCF_S25/ManageImmeuble/" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

ANNEXE II

SOAP - RECHERCHER UN IMMEUBLE PAR ADRESSE

Requête

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"

xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
            http://tempuri.org/IManageImmeuble/searchImmeubleByAdresse
    </Action>
  </s:Header>
  <s:Body>
    <searchImmeubleByAdresse xmlns="http://tempuri.org/">
      <adresse>1685 avenue Valois</adresse>
    </searchImmeubleByAdresse>
  </s:Body>
</s:Envelope>
```

Réponse

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header />
  <s:Body>
    <searchImmeubleByAdresseResponse xmlns="http://tempuri.org/">
      <searchImmeubleByAdresseResult
            xmlns:a="http://schemas.datacontract.org/2004/07/WCF_S25"
            xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <a:Immeuble>
          <a:idImmeuble>00001</a:idImmeuble>
          <a:adresse>1685 av valois</a:adresse>
        </a:Immeuble>
      </searchImmeubleByAdresseResult>
    </searchImmeubleByAdresseResponse>
  </s:Body>
</s:Envelope>
```

ANNEXE III

SOAP - RECHERCHER UN IMMEUBLE PAR IDENTIFIANT

Requête

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"

xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
            http://tempuri.org/IManageImmeuble/searchImmeubleById
    </Action>
  </s:Header>
  <s:Body>
    <searchImmeubleById xmlns="http://tempuri.org/">
      <id>00002</id>
    </searchImmeubleById>
  </s:Body>
</s:Envelope>
```

Réponse

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header />
  <s:Body>
    <searchImmeubleByIdResponse xmlns="http://tempuri.org/">
      <searchImmeubleByIdResult xmlns:i="http://www.w3.org/2001/XMLSchema-
instance"

      xmlns:a="http://schemas.datacontract.org/2004/07/WCF_S25">
        <a:Immeuble>
          <a:idImmeuble>00002</a:idImmeuble>
          <a:adresse>2260 bourbonniere</a:adresse>
        </a:Immeuble>
      </searchImmeubleByIdResult>
    </searchImmeubleByIdResponse>
  </s:Body>
</s:Envelope>
```

ANNEXE IV

SCHÉMA XSD - CONTRAT DE DONNÉES POUR IManageImmeuble

```
<xs:schema xmlns:tns="http://schemas.datacontract.org/2004/07/WCF_S25"
           elementFormDefault="qualified"

      targetNamespace="http://schemas.datacontract.org/2004/07/WCF_S25"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="ArrayOfImmeuble">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded"
                  name="Immeuble" nillable="true" type="tns:Immeuble" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfImmeuble" nillable="true"
type="tns:ArrayOfImmeuble" />
  <xs:complexType name="Immeuble">
    <xs:sequence>
      <xs:element minOccurs="0" name="idImmeuble" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" name="adresse" nillable="true"
type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Immeuble" nillable="true" type="tns:Immeuble" />
</xs:schema>
```

ANNEXE V

DÉFINITION DE LA CLASSE MANAGEIMMEUBLE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace WCF_S25
{
    class ManageImmeuble : IManageImmeuble
    {


        public List<Immeuble> searchImmeubleByAdresse(string adresse)
        {

            GetData g = new GetData();
            List<Immeuble> imm = g.GetImmeubleByAdresse(adresse);
            return imm;
        }


        public List<Immeuble> searchImmeubleById(string id)
        {
            GetData g = new GetData();
            List<Immeuble> imm = g.GetImmeubleById(id);
            return imm;
        }


    }
}
```

ANNEXE VI

DÉFINITION DE L'INTERFACE IMANAGEIMMEUBLE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace WCF_S25
{
    [ServiceContract]
    public interface IManageImmeuble
    {
        [OperationContract]
        List<Immeuble> searchImmeubleByAdresse(string adresse);

        [OperationContract]
        List<Immeuble> searchImmeubleById(string id);

    }
}
```

ANNEXE VII

DÉFINITION DE L'INTERFACE IGETDATA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace WCF_S25
{
    /*[ServiceContract]*/
    public interface IGetData
    {
        /*[OperationContract]*/
        List<Immeuble> GetImmeubleById(string id);

        /*[OperationContract]*/
        List<Immeuble> GetImmeubleByAdresse(string adresse);

    }

}
```

ANNEXE VIII

DÉFINITION DE LA CLASSE GETDATA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace WCF_S25
{
    public class GetData : IGetData
    {
        #region IGetData Members

        public List<Immeuble> GetImmeubleById(string id)
        {
            DataContext db = new DataContext();
            var matchingImmeubles = from imm in db.tS25Immeubles
                                    where imm.idImmeuble.StartsWith(id)
                                    select imm;
            return matchingImmeubles.ToList();
        }

        public List<Immeuble> GetImmeubleByAdresse(string adresse)
        {
            DataContext db = new DataContext();
            var matchingImmeubles = from imm in db.tS25Immeubles
                                    where imm.adresse.StartsWith(adresse)
                                    select imm;
            return matchingImmeubles.ToList();
        }


        #endregion
    }

}
```

ANNEXE IX

CODE GÉNÉRÉ POUR INTERFACER AVEC LA BASE DE DONNÉES

```
#pragma warning disable 1591
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.3603
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace WCF_S25
{
      using System.Data.Linq;
      using System.Data.Linq.Mapping;
      using System.Data;
      using System.Collections.Generic;
      using System.Reflection;
      using System.Linq;
      using System.Linq.Expressions;
      using System.Runtime.Serialization;
      using System.ComponentModel;
      using System;


      [System.Data.Linq.Mapping.DatabaseAttribute(Name="S25")]
      public partial class DataContext : System.Data.Linq.DataContext
      {

            private static System.Data.Linq.Mapping.MappingSource
mappingSource = new AttributeMappingSource();

    #region Extensibility Method Definitions
    partial void OnCreated();
    #endregion

            public DataContext() :

      base(global::WCF_S25.Properties.Settings.Default.S25ConnectionString,
mappingSource)
            {
                  OnCreated();
            }
```

```
            public DataContext(string connection) :
                        base(connection, mappingSource)
            {
                  OnCreated();
            }

            public DataContext(System.Data.IDbConnection connection) :
                        base(connection, mappingSource)
            {
                  OnCreated();
            }

            public DataContext(string connection,
System.Data.Linq.Mapping.MappingSource mappingSource) :
                        base(connection, mappingSource)
            {
                  OnCreated();
            }

            public DataContext(System.Data.IDbConnection connection,
System.Data.Linq.Mapping.MappingSource mappingSource) :
                        base(connection, mappingSource)
            {
                  OnCreated();
            }

            public System.Data.Linq.Table<Immeuble> tS25Immeubles
            {
                  get
                  {
                        return this.GetTable<Immeuble>();
                  }
            }
      }

      [Table(Name="dbo.tS25Immeuble")]
      [DataContract()]
      public partial class Immeuble
      {

            private string _idImmeuble;

            private string _adresse;

            public Immeuble()
            {
            }

            [Column(Storage="_idImmeuble", DbType="NChar(5)")]
            [DataMember(Order=1)]
            public string idImmeuble
            {
```

```
                get
                {
                        return this._idImmeuble;
                }
                set
                {
                        if ((this._idImmeuble != value))
                        {
                                this._idImmeuble = value;
                        }
                }
        }

        [Column(Storage="_adresse", DbType="NVarChar(50)")]
        [DataMember(Order=2)]
        public string adresse
        {
                get
                {
                        return this._adresse;
                }
                set
                {
                        if ((this._adresse != value))
                        {
                                this._adresse = value;
                        }
                }
        }
    }
}
#pragma warning restore 1591
```

ANNEXE X

CODE SOURCE POUR LA GESTION DES FACTURES AVEC UN SERVICE WEB

CONFIDENTIEL

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ServiceModel;
using ServiceReference;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    /**
     * Procédure qui ajoute la facture.
     * */
    protected void addInvoice()
    {
            //Step 1: Create an endpoint address and an instance of the WCF
Client.
            SupplierInvoiceServiceClient client = new
SupplierInvoiceServiceClient();
            SupplierInvoiceGeneralData dataContractValue = new
SupplierInvoiceGeneralData();
            // Utilisez la variable « client » pour appeler des opérations
sur le service.

            // string IsRequired=true
            //Doit être unique
            dataContractValue.InvoiceNumber = txtInvoiceNumber.Text;

            // String IsRequired=true
            // Ce numéro existe dans ffsup.id
            dataContractValue.SupplierNumber = txtSupplierNumber.Text;

            // DateTime IsRequired=true, Order=4

            int year = int.Parse(txtDueDate.Text.Substring(0, 4));
            int month = int.Parse(txtDueDate.Text.Substring(5,2));
            int day = int.Parse(txtDueDate.Text.Substring(8, 2));
```

```
            dataContractValue.DueDate = new DateTime(year, month, day);

            year = int.Parse(txtInvoiceDate.Text.Substring(0, 4));
            month = int.Parse(txtInvoiceDate.Text.Substring(5, 2));
            day = int.Parse(txtInvoiceDate.Text.Substring(8, 2));

            //obligatoire si AuxGeneralData.FiscalPeriodInputType = '4'
            dataContractValue.InvoiceDate = new DateTime(year, month, day);

            // string IsRequired=true, Order=5
            dataContractValue.Description = txtDescription.Text;

            // string Order=6
            dataContractValue.ChequeDescription =
txtChequeDescription.Text;

            // String IsRequired=true, Order=8
            // doit exister dans FFCTER.ID
            // modifié de la 1er version
            dataContractValue.PaymentTermCode =
ddlPaymentTermCode.SelectedValue;

            // string IsRequired=true, Order=9
            // doit exister dans FFCCUR.ID
            dataContractValue.ClassificationCode =
ddlClassificationCode.SelectedValue;

            //pas obligatoire
            dataContractValue.BalanceIndicator =
bool.Parse(ddlBalanceIndicator.SelectedValue.ToString());


            //obligatoire si AuxGeneralData.FiscalPeriodInputType = '1'
            //dataContractValue.FiscalPeriodInfo = new FiscalPeriodData();
            //dataContractValue.FiscalPeriodInfo.FiscalPeriod = 1;
            // dataContractValue.FiscalPeriodInfo.FiscalYear = 2009;

            // InvoiceAmountData object IsRequired=true, Order=14
            dataContractValue.InvoiceAmount = new InvoiceAmountData();
            dataContractValue.InvoiceAmount.GrossAmount =
decimal.Parse(txtGrossAmount.Text);
            dataContractValue.InvoiceAmount.NetAmount =
decimal.Parse(txtNetAmount.Text);

            // AuxGeneralData object IsRequired=true
            dataContractValue.AuxGeneralInfo = new AuxGeneralData();
            //char IsRequired=true ('2')
            dataContractValue.AuxGeneralInfo.OperationType =
char.Parse(ddlOperationType.SelectedValue);
            //char Order=1 'V'
            dataContractValue.AuxGeneralInfo.GLAccountInputType =
char.Parse(ddlGLAccountInputType.SelectedValue);
            //string Order=2
```

```
            //pDataContactValue.AuxGeneralInfo.MatchingManagementTable =;
            //char Order=3 '5'
            dataContractValue.AuxGeneralInfo.FiscalPeriodInputType =
char.Parse(ddlFiscalPeriodInputType.SelectedValue);
            //pDataContactValue.AuxGeneralInfo.ExtensionData = ;
//ExtensionDataObject


            // AccountingEntityData object IsRequired=true, Order=2
            dataContractValue.AccountingEntity = new
AccountingEntityData();
            dataContractValue.AccountingEntity.FirstElement =
txtFirstElement.Text;    //string IsRequired=true
            dataContractValue.AccountingEntity.SecondElement =
txtSecondElement.Text;   //string IsRequired=true
            dataContractValue.AccountingEntity.ThirdElement =
txtThirdElement.Text;    //string IsRequired=true

            AccountingDistributionData[] listAccountingDistributionData =
new AccountingDistributionData[1];

            AccountingDistributionData pAccountingDistributionData = new
AccountingDistributionData();
            //pAccountingDistributionData.CodificationBlock  = ;  //byte
nillable Order=4

            // Doit exister dans FGCHAR.NU _ACC
            string[] aGLAccountList = new string[] { txtAccountList.Text };
            //string[] IsRequired=true, Order=1
            pAccountingDistributionData.GLAccountList = aGLAccountList;
            //Decimal IsRequired=true, Order=2
            pAccountingDistributionData.GrossAmount =
decimal.Parse(txtGrossAmount.Text);
            //char IsRequired=true 'A'
            pAccountingDistributionData.TransactionCode =
char.Parse(ddlTransactionCode.SelectedValue);
            //  //string[] Order=5
            pAccountingDistributionData.CodificationBlock =
byte.Parse(txtCodificationBloc.Text);

            //Soit une série de string, soit une seule chaîne. Le service
concatène...
            //string[] aProjectAccount = new string[] { "1", "00147", "01",
"P", "00155", "50", "0", "92", "005" }a;
            string[] aProjectAccount = new string[] {
txtProjectAccount.Text };
            pAccountingDistributionData.ProjectAccount = aProjectAccount;


listAccountingDistributionData.SetValue(pAccountingDistributionData, 0);

            dataContractValue.AccountingDistributionList =
listAccountingDistributionData;
```

```
            NoteData[] listNoteData = new NoteData[1];
            NoteData noteData = new NoteData();
            noteData.Date = DateTime.Now;
            noteData.Description = txtNotes.Text;
            listNoteData.SetValue(noteData, 0);
            dataContractValue.Notes = listNoteData;

            // TimeStampData object IsRequired=true, Order=20
            dataContractValue.CreationTimeStamp = new TimeStampData();
            dataContractValue.CreationTimeStamp.Group = "PILOTE";
            dataContractValue.CreationTimeStamp.User = "CIO";

            try
            {
                client.Create(dataContractValue);
            }
            catch (Exception e)
            {

            }
            // Fermez toujours le client.
            client.Close();
    }

    /**
     * Action click sur bouton btnAddInvoice
     * Appelle de la fonction d'ajout
     * et redirection sur la page virtuo.aspx qui
     * affiche les 10 derniers enregistrements de la
     * table ffinvo par ordre décroissant de date de création
     * */
    protected void btnAddInvoice_Click(object sender, EventArgs e)
    {
        addInvoice();
    }
}
```

RÉFÉRENCES

Alliance, Scrum. 2009. « SCRUM : It's About Common Sense ». En ligne. <http://www.controlchaos.com/>. Consulté le 22 mars 2009.

Arsanjani, Ali. 2004. « Service-oriented modeling and architecture : How to identify, specify, and realize services for your SOA ». En ligne. <http://www.ibm.com/developerworks/library/ws-soa-design1/>. Consulté le 18 janvier 2009.

Beck, Kent 1999. *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

Bloomberg, Jason. 2006. « SOA for small and medium businesses (SMBs) ». In *searchSOA.com*. <http://searchsoa.techtarget.com/tip/0,289483,sid26_gci1168992,00.html>. Consulté le 26 mai 2008.

Boatto, Laurent. 2003. « École polytechnique fédérale de Lausanne : UDDI ». En ligne. <http://ditwww.epfl.ch/SIC/SA/SPIP/Publications/spip.php?article241>. Consulté le 13 avril 2008.

Carr, Harold. 2006. « Sun's Project Tango ». In *Sun Developer Network*. <http://java.sun.com/developer/technicalArticles/glassfish/ProjectTango/>. Consulté le 27 septembre 2008.

Corporation d'hébergement du Québec. 2010. En ligne. <http://www.chq.gouv.qc.ca/chq/index_f.aspx>. Consulté le 15 février 2010.

Daniel, Jérôme. 2003. *Services Web : Concepts, techniques et outils*. Vuibert, 341 p.

Dellazizzo, Lise 2006. « IT Business in Canada:2006 Mid Market and Large Enterprise Opportunity ». En ligne. <http://www.ipsos.ca/pdf/Whitepapers/Ipsos_WP_ITBusinessinCanada.pdf>. Consulté le 11 février 2010.

Eclipse. 2008. « Eclipse Process Framework Composer ». In *Eclipse Process Framework Project (EPF)*. En ligne. <http://www.eclipse.org/epf/>. Consulté le 24 juin 2008.

Erl, Thomas. 2004. *Service Oriented Architecture : A field Guide to Integrating XML and Web Services*. Prentice Hall Professional Technical Reference. <http://soabooks.com/fg>.

Erl, Thomas. 2005. *Service-Oriented Architecture : Concept, Technology, and design*. Prentice Hall Professional Technical Reference, 760 p.

Erradi, A., S. Anand et N. Kulkarni. 2006. « SOAF: An Architectural Framework for Service Definition and Realization ». In *Services Computing, 2006. SCC '06. IEEE International Conference on*. p. 151-158.

Gartner. 2009. « Gartner's 2009 Hype Cycle Special Report Evaluates Maturity of 1,650 Technologies ». En ligne. <http://www.gartner.com/it/page.jsp?id=1124212>. Consulté le 11 février 2010.

Guilbault, Martin. 2007. « Application de l'approche de modélisation par archétype selon les normes et concepts de l'architecture orientée services ». Mémoire de maîtrise, Montréal, École de technologie supérieure, 126 p.

IBM. 2007. « IBM Rational Unified Process ». En ligne. <ftp://ftp.software.ibm.com/software/rational/web/datasheets/RUP_DS.pdf>. Consulté le 23 mars 2009.

Jacobson, ivar, Grady Booch et James Rumbaugth. 1998. *The Unified Software Development Process*. Addison Wesley.

Josey, Andrew, et al. 2009. *TOGAF Version 9 : Guide de Poche*, Seconde éd. Wilco, Amersfoort - Hollande: Van Haren Publishing. Consulté le 17 février 2010.

Kent, Beck, Beedle Mike, van Bennekum Arie, Cockburn Alistair, Cunningham Ward, Fowler Martin, Grenning James, Highsmith Jim, Hunt Andrew, Jeffries Ron, Kern Jon, Marick Brian, C. Martin Robert, Mellor Steve, Schwaber Ken, Sutherland Jeff et Thomas Dave. 2001. « Manifesto for Agile Software Development ». En ligne. <http://agilemanifesto.org/>. Consulté le 11 février 2010.

Klein, Scott 2007. *Professional WCF Programming: .NET Development with the Windows Communication Foundation*. Wrox press, 450 p.

Larman, Craig. 2005. *Applying UML and patterns : An introduction to object-oriented analysis and design and iterative developement*, 3e edition. Prentice Hall, 702 p.

Le Framework de Zachman. 2011. En ligne. <http://en.wikipedia.org/wiki/Zachman_Framework>.

« Le grand dictionnaire terminologique ». 2010. In *Le grand dictionnaire terminologique*. En ligne. <http://w3.granddictionnaire.com/btml/fra/r_motclef/index800_1.asp>. Consulté le 15 février 2010.

Lehman, M. M. 1980. « Programs, life cycles, and laws of software evolution ». *Proceedings of the IEEE,* vol. 68, n° 9, p. 1060-1076.

Meehan, Michael. 2007. « Why SOA makes sense for SMBs ». In *SOA Talk: A SearchSOA.com blog*. <http://soa-talk.blogs.techtarget.com/category/small-midsize-business-smb/>. Consulté le 26 mai 2008.

Microsoft Visual Studio 2008. 2010. En ligne. <http://www.microsoft.com/visualstudio/en-us/default.mspx>. Consulté le 3 mars 2010.

Natis, Yefim V. 2003. « Service-Oriented Architecture Scenario ». En ligne. Gartner. <http://www.gartner.com/resources/114300/114358/114358.pdf>. Consulté le 26 mai 2008.

OASIS. 2010. « OASIS ». <http://www.oasis-open.org/home/index.php>. Consulté le 13 janvier 2010.

The Open Group : Making Standards Work. 2010. En ligne. <http://www.opengroup.org/>. Consulté le 15 février 2010.

OpenUP. 2010. En Ligne. <http://epf.eclipse.org/wikis/openup/>. Consulté le 22 février 2010.

Paradis, Pierre Emmanuel. 2004. « Québec : Regard sur la PME ». En ligne. Fédération Canadienne de l'Entreprise Indépendante. <http://www.fcei.ca/quebec/pdf/Qc_Primer_2004.pdf>. Consulté le 26 mai 2008.

Peiris, Chris , et Dennis Mulder. 2007. *Pro WCF: Practical Microsoft SOA Implementation*. Apress, 505 p.

Ramollari, Ervin , Dimitris Dranidis et Anthony J. H. Simons. 2007. « A Survey of Service Oriented Development Methodologies ». <http://www.dcs.shef.ac.uk/~ajhs/research/papers/soasurvey.pdf>.

Scott W.Ambler, John Nalbone, Michael J. Vizdos. 2005. *The enterprise unified process : extending the Rational Unified Process*. Prentice Hall, 384 p. <http://www.enterpriseunifiedprocess.com/>.

« SOA RQ Methodology : A Pragmatic Approach ». 2006. En ligne. Sun Microsystems. <http://www.sun.com/products/soa/soa_methodology.pdf>. Consulté le 23 mars 2009.

TOGAF. 2009. In *TOGAF Version 9*. En ligne. <http://www.opengroup.org/togaf/>. Consulté le 22 mars 2009.

W3C. 2000. « SOAP ». En ligne. <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>. Consulté le 7 avril 2008.

W3C. 2002. « Les services Web ». En ligne. <http://www.w3.org/2002/ws/Activity>. Consulté le 6 avril 2008.

W3C. 2008. *W3C* En ligne. <http://www.w3.org/>. Consulté le 24 juin 2008.

Wells, Don 2006, 17 février. « Extreme Programming : A gentle introduction ». En ligne. <http://www.extremeprogramming.org/>. Consulté le 22 mars 2009.

White, A. Stephen; Miers, Derek. 2008. *BPMN Modeling and Reference Guide : Understanding and Using BPMN*. Ligthhouse Point (FL.): Future Strategies, 225 p.

Windows Communication Foundation. 2010. En ligne. <http://msdn.microsoft.com/en-us/netframework/aa663324.aspx>. Consulté le 3 mars 2010.

Wing, Hong Lam, et Shankararaman Venky. 2007. *Enterprise Architecture and Integration: Methods, Implementation and Technologies* IGI Global, 344 p.

WS-I. 2008. « The Web Services Interoperability Organization (WS-I) ». En ligne. <http://www.ws-i.org/>. Consulté le 12 avril 2009.

Zachman, J. A. 1987. « A framework for information systems architecture ». *IBM Systems Journal,* vol. 26, n^{o} 3, p. 276-292.

Zachman, John A. 2008. *Zachman framework associates*. En ligne. <http://www.zachmanframeworkassociates.com/index.php/home-article/13#maincol>. Consulté le 2011-05-18.

Zimmermann, Olaf, Pal Krogdahl et Clive Gee. 2004. « Elements of Service-Oriented Analysis and Design : An interdisciplinary modeling approach for SOA projects ». En ligne. <http://www.ibm.com/developerworks/webservices/library/ws-soad1/>. Consulté le 18 janvier 2009.