

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT DE PROJET PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE DIPLÔME DE
MAITRISE EN TECHNOLOGIE DE L'INFORMATION

PAR
ABDELKRIM ALI-BEY

ARCHITECTURE ORIENTÉE PATRON, DIRIGÉE PAR DES MODÈLES ET ORIENTÉE
SERVICES (SOPOMA) POUR LES SYSTÈMES INTERACTIFS

MONTRÉAL, LE 06 OCTOBRE 2014



Abdekrim Ali-Bey, 2014



Cette licence [Creative Commons](https://creativecommons.org/licenses/by-nc-nd/4.0/) signifie qu'il est permis de diffuser, d'imprimer ou de sauvegarder sur un autre support une partie ou la totalité de cette œuvre à condition de mentionner l'auteur, que ces utilisations soient faites à des fins non commerciales et que le contenu de l'œuvre n'ait pas été modifié.

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Alain April, Directeur de maîtrise
Département de génie logiciel & technologie de l'information
École de technologie supérieure

M. Mohamed Taleb, Co-directeur de maîtrise
Département de génie logiciel & technologie de l'information
École de technologie supérieure

M. Alain Abran, Jury
Département de génie logiciel & technologie de l'information
École de technologie supérieure

REMERCIEMENTS

Je tiens à remercier tout d'abord mon directeur de recherche, Dr. **Alain April**, à qui je dois toute ma gratitude de m'avoir accepté en tant qu'étudiant de Maîtrise, et surtout de m'avoir fait confiance à la réalisation de ce mémoire. Alain April, un homme très accueillant et chaleureux. Je lui dis encore mille fois merci pour ses précieux conseils.

Je tiens également à remercier Dr. Mohamed Taleb d'avoir supervisé mon mémoire de maîtrise tout au long de son développement. Je voudrais lui témoigner ma gratitude pour sa patience, ses disponibilités et son soutien qui m'a été précieux afin de mener à bien mon travail.

Enfin, je tiens à remercier ma famille et mes amis pour leur soutien et leur patience durant ces deux longues années d'études.

ARCHITECTURE ORIENTÉE PATRON, DIRIGÉE PAR DES MODÈLES ET ORIENTÉE SERVICES (SOPOMA) POUR LES SERVICES,

Abdelkrim Ali-Bey

RÉSUMÉ

La recherche concernant les systèmes interactifs et de l'ingénierie des interfaces utilisateurs (UI) a publié plusieurs modèles d'architectures. Les développeurs et les concepteurs de logiciels ne rencontrent pas de grandes difficultés lors de la création d'un simple système interactif qui n'est pas sujet à de fréquents changements. Cette situation se complique lorsqu'il est question de logiciels de grande envergure contenant des processus d'affaires critiques, et la modification de ces logiciels patrimoniaux devient souvent un handicap. Un certain nombre de problèmes de conception se posent dans l'architecture, tels que : la gestion des environnements hétérogènes et multiplateformes, les faiblesses des langages de programmation et la prise en charge d'autres aspects de dialogues.

Ce projet de maîtrise vise à définir une architecture pour la conception des systèmes interactifs, que nous appelons SOPOMA (Service-Oriented Pattern-Oriented Model Driven Architecture ou tout simplement Service-Oriented POMA : architecture POMA Orientée Services). SOPOMA est basée sur la combinaison des différents concepts de deux architectures déjà existantes telles que : (1) l'Architecture Orientée Patron et dirigée par des Modèles (POMA), et (2) l'Architecture Orientée Services (AOS). SOPOMA vise à faciliter le développement et la migration des systèmes interactifs tout en améliorant leur utilisabilité et leur qualité en général. En d'autres termes, SOPOMA identifie une liste de services visant à fournir, sous forme d'une architecture orientée services, orientée patterns et dirigée par des modèles, une solution globale et intégrative. L'architecture SOPOMA montre comment plusieurs services peuvent être combinés à différents niveaux d'abstraction dans des structures hétérogènes, qui peuvent alors être utilisés comme éléments de base dans le développement des systèmes interactifs.

Ce document est divisé en trois chapitres. Le premier chapitre présente l'état de l'art sur l'architecture « POMA » incluant ses composants tels que ses patrons et ses modèles, et « l'architecture orientée services (AOS) » incluant entre autres son plus important composant appelé « service ». Le deuxième chapitre introduit la problématique de la recherche avec ses objectifs, ses limites, la méthodologie de la recherche et ses étapes de la recherche. Le troisième chapitre décrit principalement les parties les plus importantes de la recherche qui est de développer une nouvelle architecture appelée architecture orientée pattern et dirigée par des modèles orientés services (**SOPOMA**) pour faciliter le développement des systèmes interactifs incluant ses fondements et ses concepts clés, sa vue d'ensemble, ses justifications versus les architectures POMA et AOS, et ses spécifications. Le quatrième chapitre présente une étude de cas exploratoire appliquée à l'architecture proposée dans cette recherche. Le

VI

dernier chapitre présente une conclusion sur les travaux de recherche et son évolution dans le futur.

Mots-clés : SOPOMA, POMA, SOA, service, ESB, patron, modèle, PIM, PSM, architecture, système interactif.

SERVICE-ORIENTED PATTERN-ORIENTED AND MODEL-DRIVEN ARCHITECTURE (SOPOMA) FOR INTERACTIVE SYSTEMS,

Abdelkrim, Ali-Bey

ABSTRACT

Interactive systems and engineering user interfaces (UI) research has released several models of architectures. Developers and software designers do not face major difficulties when dealing with a simple interactive system which is not subject to frequent changes. However, this situation is more complex when it comes to large software containing critical business processes, and changing these legacy software often becomes a burden. A number of design issues continue to arise in the architecture, such as the management of heterogeneous environments and multiplatform, weaknesses in programming languages and support other aspects of dialogues.

This master degree project proposes an architecture for the design of interactive systems, which we call SOPOMA (Service-Oriented, Pattern-Oriented and Model-Driven Architecture or simply Service Oriented POMA). SOPOMA is based on the combination of concepts of two existing architectures: (1) Pattern-Oriented and Model-Driven Architecture (POMA), and (2) Service-Oriented Architecture (SOA). SOPOMA aims to facilitate the development and migration of interactive systems while improving their usability and quality in general. This means, SOPOMA identifies a list of services for providing, in the form of service-oriented, patterns oriented and model-driven architecture, a comprehensive and integrative solution. SOPOMA architecture shows how multiple services can be combined at different levels of abstraction into heterogeneous structures, which can then be used as building blocks in the development of interactive systems.

This document is divided into three chapters: the first chapter presents a background and related work on "POMA architecture including its components such as "Patterns" and "Models", and Service-Oriented Architecture (SOA) including, among others, the most important component called the "service". The second chapter introduces the research topic with its objectives, its limits, the research methodology, and research steps. The third chapter describes primarily the most important parts of the research: the development of a new architecture called **Service-Oriented Pattern-Oriented and Model-Driven Architecture (SOPOMA)** for facilitating the development of interactive systems, including fundamentals and key concepts, an overview, justifications versus POMA and SOA architectures and its specifications. The fourth chapter presents an exploratory case study applied to the architecture proposed in this research. The last chapter presents a conclusion on this research work and its expected evolution in the future.

Key words : SOPOMA, POMA, SOA, service, ESB, Pattern, Model, PIM, PSM, architecture, interactive system.

TABLE DES MATIÈRES

INTRODUCTION	1
CHAPITRE 1 ÉTAT DE L'ART	11
1.1 Architecture orientée patron et dirigée par des modèles (POMA)	11
1.1.1 Patrons.....	14
1.1.2 Modèle	16
1.1.3 Composition de l'architecture POMA	18
1.2 Architecture orientée services.....	22
1.2.1 Vue globale de l'architecture orientée services	24
1.2.2 Services Web	43
1.3 Pourquoi combiner les deux architectures ?	47
1.4 Sommaire du chapitre	51
CHAPITRE 2 PROBLÉMATIQUE DU PROJET	53
2.1 But et objectifs du projet.....	54
2.2 Limites du projet.....	54
2.3 Méthodologie du projet.....	55
2.4 Résumé du chapitre.....	56
CHAPITRE 3 ARCHITECTURE ORIENTÉE PATRON, DIRIGÉE PAR DES MODÈLES ET ORIENTÉE SERVICES	59
3.1 Concepts clés de SOPOMA	59
3.2 Vue d'ensemble de SOPOMA	60
3.3 Justification de SOPOMA	73
CHAPITRE 4 ÉTUDE DE CAS EXPLORATOIRE.....	75
4.1 Services de composition	75
4.2 Services de conversion « mapping ».....	77
4.3 Services de transformation.....	79
CONCLUSION	83
ANNEXE I SCHÉMA DE CLASSIFICATION DES PATRONS PAR CATÉGORIES	89
BIBLIOGRAPHIE.....	XCIII

LISTE DES FIGURES

Figure 0.1 – Combinaison de POMA et de la SOA.....	8
Figure 1.1 – POMA : Architecture pour le développement des systèmes interactifs.....	13
Figure 1.2 – Composition du patron	16
Figure 1.3 – Les couches d'une SOA.....	25
Figure 1.4 – Les couches de SOA.....	27
Figure 1.5 – Définition des termes clés de l'architecture orientée services.	28
Figure 1.6 – Différence entre l'orchestration et la chorégraphie.....	30
Figure 1.7 – Collaboration.	31
Figure 1.8 – Autonomie de service.	35
Figure 1.9 – Les composants de l'Entreprise Service Bus.	38
Figure 1.10 – Modèle d'ESB basé sur la SOA.....	39
Figure 1.11 – Les différents participants de la SOA.....	40
Figure 1.12 – Les principales limites de l'architecture orientée services.....	43
Figure 1.13 – Scénario d'utilisation des services Web.....	45
Figure 1.14 – Principe de fonctionnement d'une architecture basée sur les services Web.	46
Figure 1.15 – Résumé des standards ouverts et des plateformes de SOA.	50
Figure 2.1 – Méthodologie du projet.	56
Figure 3.1 – Architecture SOPOMA.	60
Figure 3.2 – Exemple de patrons similaire.	62
Figure 3.3 – Conception d'une page d'accueil en utilisant les patrons.	63
Figure 3.4 – Apparence d'un site pour différentes plateformes..	64
Figure 3.5 – Diagrammes de classe du modèle de domaine PIM.....	66

Figure 3.6 – Diagramme de classe du modèle de tâches PIM.	66
Figure 3.7 – Modèles de domaine PIM et modèle de tâches PIM unifiés pour la plateforme Laptop.	69
Figure 3.8 – Exemple d’une architecture N-tiers.	72
Figure 4.1 – Exemple de services de composition de patrons.	75
Figure 4.2 – Diagramme de classe du modèle de domaine PIM.	77
Figure 4.3 – Exemple d’utilisation des services de conversion « Mapping » du modèle de domaine PIM vers PSM pour PDA.	78
Figure 4.4 – Résultat de la conversion « Mapping » du modèle de domaine PIM à PSM pour la plateforme PDA.	79
Figure 4.5 – Exemple de services de transformation.	80

LISTE DES TABLEAUX

Tableau 1.1 – Changements dans les TI dirigés par les affaires	49
Tableau 1.2 – Évaluation des deux architectures et leurs caractéristiques	52

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

CRM	Customer Relationship Management
MDA	Model-Driven Architecture
MVC	Modèle-Vue-Contrôleur
OASIS	Advancing Open Standards for the Information Society
OMG	Object Management Group
PIM	Platform Independent Model
POD	Pattern-Oriented Design
POMA	Pattern-Oriented and Model-driven Architecture
PSM	Platform Specific Model
QOS	Qualité de Service (Quality of Service)
PDA	Personal digital assistant
SAP	Systems, Applications and Products for data processing
SCA	Service Component Architecture
SDO	Service Data Objects
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOPOMA	Service-Oriented POMA

UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
UI	User Interface
WSDL	Web Services Description Language
XML	Extensible Markup Language

INTRODUCTION

La recherche concernant les systèmes interactifs et de l'ingénierie des interfaces utilisateurs (UI) a publié plusieurs modèles d'architectures (Taleb *et al.*, 2009). Ces modèles d'architectures contribuent non seulement pour faciliter le développement et la maintenance des systèmes interactifs, mais aussi pour soutenir et améliorer la normalisation, la portabilité et la facilité d'utilisation des systèmes interactifs développés (Taleb *et al.*, 2013).

Ce projet de recherche se base sur les travaux réalisés par (Taleb *et al.*, 2009) sur les architectures de développement des systèmes interactifs. Il vise à améliorer l'Architecture Orientée Patrons et dirigée par des Modèles (POMA) afin de pallier à de nombreux problèmes de conception rencontrés tels que : la gestion des environnements hétérogènes, les faiblesses concernant les langages de programmation et la prise en charge d'autres aspects de dialogues en lui introduisant l'approche orientée services basée sur l'Architecture Orientée Services (AOS ou SOA pour Service-Oriented Architecture).

Dans cette introduction, une vue d'ensemble des systèmes interactifs et le concept d'architecture sur lequel ils sont basés, une description générale de l'architecture POMA, le concept de service, les motivations et les objectifs du projet et finalement l'organisation du document sont présentés.

A. À propos des systèmes interactifs

Un système interactif est une application avec laquelle l'utilisateur dialogue avec le système au moyen des interfaces dans le but d'accomplir une tâche donnée. Un système interactif est composé de deux parties (Taleb *et al.*, 2009) :

- La partie matérielle du système interactif est composée des périphériques d'entrée et sortie et des différents gestionnaires (pilotes de périphériques) qui fournissent le support physique à l'exécution de l'application interactive ;

- La partie logicielle (application interactive), qui est composée de deux sous-parties : l'interface utilisateur et la sous-partie algorithmique qui constitue la sémantique du système interactif.

L'interface utilisateur peut être vue comme un moyen par lequel transitent des données entre l'utilisateur et la machine. L'affichage des données sur l'écran (par exemple) constitue un support pour l'interaction et la rétroaction aux actions de l'utilisateur. Une interface utilisateur est donc la partie d'une application interactive qui : (a) présente les sorties à l'utilisateur ; (b) collecte les entrées de ce dernier pour les transmettre aux procédures de l'application qui les traitent et ; (c) se charge de l'enchaînement des dialogues.

Le terme « architecture » étant un concept crucial sur lequel s'appuient les systèmes interactifs. La section suivante présente un aperçu de ce concept dans le cadre de ce projet.

B. À propos des architectures

Plusieurs définitions ont été proposées concernant le concept d'architecture, parmi lesquelles :

(M. Fowler, 2002) a indiqué que Ralph Johnson a publié une remarquable annonce dans sa liste de courriers proposant de définir le terme « architecture » comme étant : « l'ensemble des compréhensions partagées de la conception d'un système par des développeurs et experts qui travaillent ensemble sur un projet logiciel » (traduction libre). Ces compréhensions représentent les principaux composants du système et la manière dont ils interagissent. Elles concernent également les décisions que les développeurs prennent.

(Perry et Wolf, 1992) proposent qu'une architecture logicielle est composée de trois types d'éléments, ayant chacun un objectif spécifique. À savoir :

- Éléments de traitements : ce type de composants effectue des transformations sur les éléments de données ;

- Éléments de données : ce type de composants contient les informations utilisées dans le processus de transformation ;
- Éléments de liaison : ce type de composant vise à lier les différentes pièces des éléments de l'architecture du logiciel.

Plusieurs définitions de modèles architecturaux des systèmes interactifs ont été proposées telles que :

(Buschmann *et al.*, 1996) proposent que le modèle architectural est « la structure des sous-systèmes et des composants d'un système et la relation entre eux typiquement représentés dans différentes vues pour démontrer les propriétés fonctionnelles et non fonctionnelles pertinentes » (traduction libre). Cette définition introduit les principaux composants d'architecture (en occurrence les sous-systèmes, les composants et les connecteurs) et couvre les méthodes pour les représenter, incluant à la fois les exigences fonctionnelles et non fonctionnelles, au moyen d'un ensemble de vues.

(Bass *et al.*, 2003) rejoint la définition de (Buschmann *et al.*, 1996) en indiquant que :

« L'architecture logicielle d'un programme ou d'un système informatique est la structure ou les structures du système, qui comprennent des composants logiciels, les propriétés visibles de l'extérieur de ces composants, ainsi que les relations entre eux. »

Cette définition introduit les principaux composants d'un logiciel et couvre la manière avec laquelle ils sont reliés (tels que les objets, les sous-systèmes et les relations : « appelle », « envoie », « utilise », etc.)

La section précédente a introduit les architectures logicielles de façon générale, et la section suivante décrit, de façon plus générale, l'architecture POMA tout en mentionnant ses forces et faiblesses.

C. À propos de l'architecture orientée patron et dirigée par des modèles (POMA)

POMA (Pattern-Oriented and Model-Driven Architecture) est une architecture permettant de faciliter le développement des systèmes interactifs proposée par (Taleb *et al.*, 2009). Elle repose essentiellement sur les concepts des architectures orientées patrons telles que POD (Pattern-Oriented Design) PSA (Pattern Supported Approach) et l'architecture dirigée par des modèles (MDA pour Model-Driven Architecture), et les concepts de certaines architectures n-tiers tels que : architecture MVC (« Model-View-Controller ») à 3-tiers (Goldberg, 1984), l'architecture à 5-tiers J2EE appelée «Core J2EE Patterns Architecture» et l'architecture de Zachman à multi-tiers (Taleb *et al.*, 2009).

L'architecture POMA fournit des solutions éprouvées à de multiples problèmes tels que :

- Offrir la possibilité aux développeurs de concevoir des interfaces utilisateurs sans se soucier de leur implémentation ;
- Définir des règles de conversion (« mapping ») permettant aux développeurs de convertir (« mapper ») un type de modèle vers un autre type et définir la manière de les combiner à différents niveaux d'abstraction dans des structures hétérogènes ;
- Définir des règles de transformation permettant de créer un modèle à partir d'un autre modèle de la même catégorie ;
- Créer des interfaces utilisateurs qui peuvent s'adapter simultanément sur plusieurs plateformes.

Parmi les faiblesses de l'architecture POMA, on peut mentionner :

- Le besoin de définir des mesures pour évaluer objectivement l'applicabilité des patrons qui peuvent être utilisés dans POMA ;

- POMA ne facilite pas l'utilisation des contraintes de conception ou des descriptions des interfaces qui sont d'une grande importance pour le concepteur (Booch *et al.*, 1999) (Myers, 1986) (Myers et Buxton, 1986) (Myers, 1990).

Cette section a permis d'introduire les concepts clés de l'architecture POMA, ses solutions à de multiples problèmes, ainsi que ses faiblesses. La section suivante introduira le concept de services qui est un concept essentiel utilisé par la nouvelle architecture SOPOMA.

D. À propos des services

L'architecture orientée services (AOS) (Jerstad *et al.*, 2005) a été proposée pour soutenir l'informatique orientée services. Pour la plupart des entreprises, la SOA offre une flexibilité considérable dans l'alignement des fonctions des technologies de l'information (TI), des processus d'affaires et des objectifs (A. Arsanjani *et al.*, 2007).

Le World Wide Web Consortium (W3C) (Booth *et al.*, 2004) définit l'architecture orientée services comme une forme de l'architecture des systèmes distribués caractérisée par les propriétés suivantes :

- Le service est une vue logique abstraite du programme réel ;
- La communication se fait par échange de messages entre les agents fournisseurs de services et les agents demandeurs de services, et non les propriétés des agents eux-mêmes ;
- La description des services n'inclut que les propriétés publiques et importantes à l'utilisation des services ;
- Le nombre d'opérations que les services utilisent a tendance à être petit avec des messages relativement importants et complexes ;
- Les services ont tendance à être orientés vers l'utilisation sur un réseau ;

- Les messages envoyés sont dans un format normalisé sous le langage XML (eXtensible Markup Language).

Finalement, l'idée derrière l'architecture orientée service est que tout composant du système doit être considéré en tant qu'un service identifiable, documenté, fiable, autonome et indépendant des autres services, accessible, et réalisant un ensemble de tâches parfaitement définies (Booth et al., 2004).

Cette section a fourni un aperçu sur le concept de service dans le cadre de l'architecture orientée services (AOS). Cette dernière est décrite de façon détaillée dans le chapitre 1.

E. Nouvelle génération de plateforme

Ces dernières années, les systèmes interactifs ont mûri en offrant les fonctionnalités de la simple page statique pour fournir à des processus complexes tels que des transactions financières bout-en-bout. Plusieurs techniques plus sophistiquées ont été mises à la disposition des utilisateurs pour interagir avec des services et l'information disponibles en utilisant différents types d'ordinateurs. Ces différents types d'ordinateurs et de dispositifs (comprenant, mais non limité, les ordinateurs traditionnels de bureau, les ordinateurs portables, les palmtops, le PDAs avec et sans claviers, smartphone, tablettes et les télévisions interactives) sont utilisés pour interagir avec de telles applications. Une des caractéristiques principales et majeures de tels systèmes interactifs de différentes plateformes est qu'ils permettent à un utilisateur d'interagir avec les services et le contenu du côté serveur de diverses manières. Les systèmes interactifs de petits dispositifs mobiles sont des ressources contraintes et ne peuvent pas soutenir la gamme complète des dispositifs des systèmes interactifs et de l'interactivité en raison du manque de l'espace d'écran ou de basse largeur de bande (Taleb, 2008).

La mosaïque des systèmes interactifs et des plateformes a mené à l'apparition et l'émergence de l'ingénierie des systèmes interactifs comme sous-discipline du génie logiciel avec quelques défis spécifiques. Une question importante est de savoir comment développer et déployer la même application pour différentes plateformes - sans « architecturer » et écrire

spécifiquement le code pour chaque plateforme, ou apprendre les différents langages de programmation et les nombreuses directives de conception des systèmes interactifs sont disponibles pour chaque plateforme.(Taleb, 2008).

F. Motivations et objectifs du projet

Les motivations clés pour ce projet sont :

- Le besoin d'adapter et de raffiner (1) les concepts de l'architecture POMA et (2) les concepts de l'architecture SOA afin de répondre à certains des défis de conception et développement de systèmes interactifs multiplateformes ;
- La nécessité de soutenir les utilisateurs novices et les experts en développement de systèmes interactifs afin de faciliter l'utilisation des services tant par les experts que par les novices.

Le but de ce projet est de définir une nouvelle architecture pour faciliter le développement et la migration des systèmes interactifs tout en améliorant leur utilisabilité et leur qualité en général.

Pour atteindre ce but, l'objectif est de définir une nouvelle architecture supportée par les outils CASE (Computer Aided Software Engineering), afin de bénéficier des avantages qu'offrent ces outils tels que : la rapidité de réalisation de certaines tâches, notamment celle qui sont liées à la réalisation de diagrammes et les spécifications associées, une meilleure précision, la réduction de la durée de maintenance, une meilleure documentation (PhoenixUniversity, 2011). De ce fait, la Figure 1.1 résume la combinaison de deux architectures : l'architecture POMA et l'architecture orientée services (AOS) pour obtenir la nouvelle architecture appelée architecture POMA orientée services (**SOPOMA** pour **S**ervice-**O**riented **POMA**).

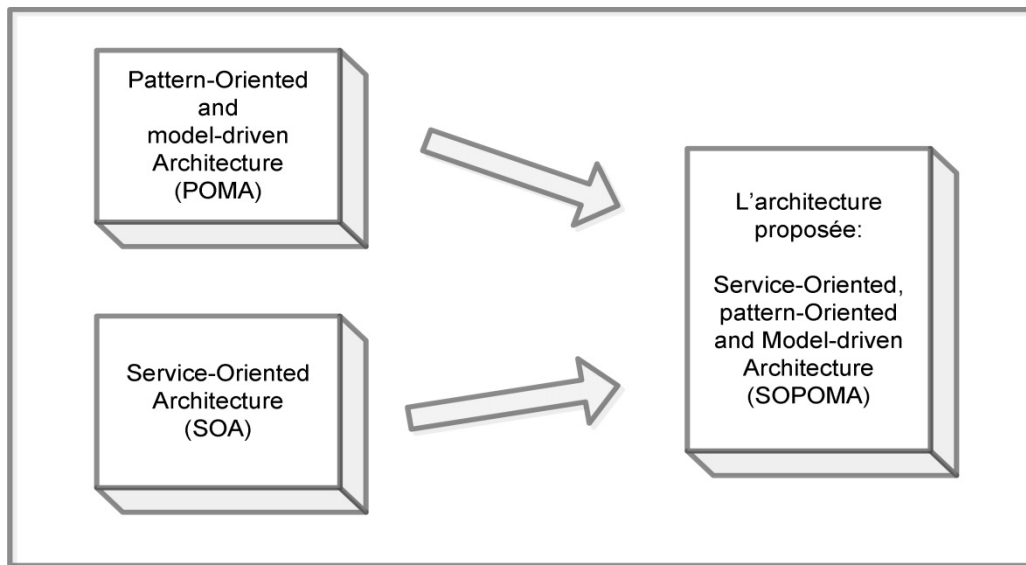


Figure 1.1 – Combinaison de POMA et de la SOA.

Selon la Figure 1.1, l'architecture SOPOMA doit principalement s'appuyer sur les concepts de l'architecture POMA et ceux de l'architecture SOA.

G. Organisation du mémoire

L'organisation de ce mémoire est comme suit :

Le CHAPITRE 1 présente l'état de l'art sur les concepts de l'architecture POMA, et les concepts de l'architecture orientée services (SOA).

Le CHAPITRE 2 introduit la problématique du projet, notamment le but et les objectifs du projet, ses limites et la méthodologie du projet.

Le CHAPITRE 3 présente les résultats du projet, en décrivant les concepts clés de la nouvelle architecture orientée patron, dirigée par des modèles et orientée services (SOPOMA), une vue d'ensemble de cette architecture et ces justifications.

Le CHAPITRE 4 présente une étude de cas pour illustrer et clarifier l'utilisation possible de la nouvelle architecture proposée et de quelle manière l'orientation service facilitera le développement des systèmes interactifs.

Finalement, le dernier chapitre de ce mémoire présente la conclusion concernant ce projet de recherche en résumant ses contributions principales, les limites de la proposition SOPOMA, et les travaux futurs.

CHAPITRE 1

ÉTAT DE L'ART

Ce chapitre présente l'état de l'art sur l'architecture orientée patron et dirigée par des modèles, appelée POMA (« **P**attern-**O**riented and **M**odel-driven **A**rchitectures ») ainsi que sur l'architecture orientée services (AOS ou SOA pour « Service-Oriented Architecture »).

La première section introduit l'architecture POMA et ses composants. La deuxième section introduit l'architecture orientée services (SOA) et ses principaux concepts. La troisième section discute de la combinaison de l'architecture POMA et l'architecture SOA. La quatrième section présente une évaluation de ces deux architectures de développement et leurs caractéristiques, et identifie les différences de ces architectures et la problématique du projet.

1.1 Architecture orientée patron et dirigée par des modèles (POMA)

Comme présenté dans l'introduction, l'architecture POMA (« Pattern-Oriented and Model-Driven Architecture») (Figure 1.1) (Taleb *et al.*, 2009) repose essentiellement sur les concepts des patrons, des modèles et des architectures n-tiers. L'architecture POMA facilite le développement des systèmes interactifs en améliorant leur utilisabilité et leur qualité en général. Plus particulièrement elle facilite l'ingénierie des applications multiplateformes telles que sur un ordinateur traditionnel, un PDA (Personal digital assistant), un téléphone mobile, une télévision interactive, un ordinateur portable et un ordinateur de poche.

L'architecture POMA (Taleb *et al.*, 2009) a été développée en se basant sur des concepts déjà existants dans d'autres architectures telles que :

- Le concept de niveaux d'architecture utilisé dans les architectures N-tiers telles que architecture à 3-tiers (MVC pour Modèle-Vue Contrôleur) (Taleb *et al.*, 2009) et (Goldberg, 1984), l'architecture à 5-tiers « Core Patterns J2EE » (Taleb *et al.*, 2009), (Goldberg, 1984), et (Alur, 2004), et finalement l'architecture multi-tiers de Zackman (Taleb, 2008; Zachman, 1987) ;

- Le concept de techniques de composition évoqué par l'architecture « Design orienté patrons ou POD pour Pattern-Oriented Design » (Taleb *et al.*, 2009) et (Yacoub et Ammar, 2003);
- Le concept de catégorisation de patrons utilisé par l'architecture « Pattern-Supported Approach » (Taleb *et al.*, 2009) et (Granlund *et al.*, 2001) ;
- Le concept de transformation des modèles et le concept de conversion (« Mapping ») utilisés par l'architecture dirigée par des modèles (MDA pour « Model-Driven Architecture ») (Taleb *et al.*, 2009) et (Miller et Mukerji, 2003).

Afin de simplifier la compréhension de la Figure 1.1, voici quelques instructions :

- Verticalement, est représentée la composition des patrons pour produire dix modèles PIM et PSM ;
- Horizontalement, sont représentées la composition et la conversion (« Mapping ») des patrons pour produire cinq modèles PIM et cinq modèles PSM et de la génération du code source pour l'ensemble du système interactif.

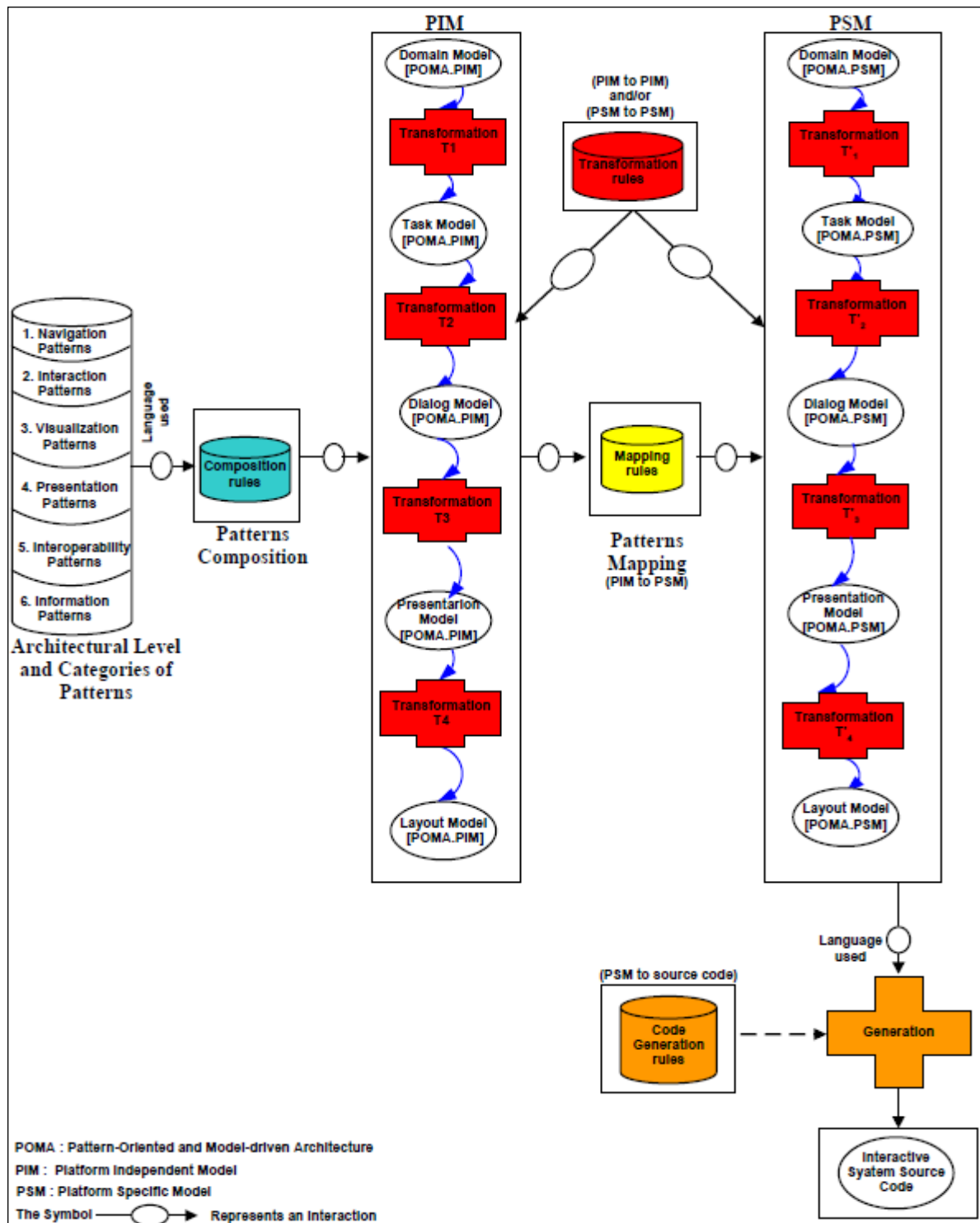


Figure 1.1 – POMA : Architecture pour le développement des systèmes interactifs.
Tirée de (Taleb *et al.*, 2009)

Une vue d'ensemble de l'architecture POMA a été présentée dans la section précédente, la description de ses composants est présentée dans les sections 1.1.1 et 1.1.2 :

1.1.1 Patrons

Cette section introduit certaines définitions des patrons suggérées par plusieurs auteurs, la composition des patrons et leurs objectifs.

Les patterns ont été introduits par l'architecte en bâtiment Christopher Alexander en 1977 (Alexander *et al.*, 1977). Il s'est basé sur la remarque suivante : « chaque modèle décrit un problème qui se manifeste constamment dans notre environnement, et donc décrit le cœur de la solution à ce problème, d'une façon telle que l'on puisse réutiliser cette solution des millions de fois, sans jamais le faire deux fois de la même manière » (Alexander *et al.*, 1977). À partir de cette remarque, il a organisé un catalogue de descriptions de problèmes et de solutions à ces problèmes. D'après (Meijers, 1996), C. Alexander définit les patterns comme la description d'un problème récurrent et les phases d'une solution générale à ce problème.

(Coplien, 1998) propose une définition plus générique d'un patron logiciel comme : « ...la chose et les instruments pour effectuer la chose » (traduction libre).

Les patrons de conception utilisés lors de développement de logiciels peuvent être vus comme un moyen d'organiser plusieurs modules en les combinant afin d'apporter une solution à un problème de conception de logiciels (Mall, 2009).

D'autres définitions ont été suggérées :

- « Les patrons décrivent une chose et comment y parvenir à la réaliser en utilisant les instruments appropriés » (Coad, 1992) (traduction libre) ;
- « Un patron décrit un problème à résoudre, une solution, et le contexte dans lequel cette solution est considérée. Un patron désigne une technique et décrit son coût et ses bénéfices, permettant à une équipe d'avoir un vocabulaire commun pour décrire les modèles » (Johnson, 1997) (traduction libre);

- « Les patrons capturent la connaissance dont les experts appliquent afin de résoudre leurs problèmes récurrents » (Rising, 1996) (traduction libre) ;
- « Le patron est une idée utilisée en pratique dans un contexte et qui peut être réutilisée par d'autres » (M Fowler, 1997) (traduction libre) ;
- « Le patron est la solution à un problème dans un contexte » (Lea, 1997) (traduction libre).

Chaque patron est principalement composé de trois éléments (Figure 1.2) (Taleb, 2008) et (Kohls et Wedekind, 2010)

1. Contexte : représente un ensemble de situations dans lesquelles les patrons s'adaptent et utilisent des méthodes, des outils et des technologies de manière adéquate afin d'arriver à une solution.
2. Problème : c'est un ensemble de forces, d'objectifs et de contraintes qui se produisent dans un contexte spécifique décrivant les problèmes de conception à résoudre. Selon (Kohls et Wedekind, 2010, beaucoup de personnes pensent que les patrons sont créés et/ou proposés pour résoudre des problèmes rencontrés. Alors qu'en réalité, le vrai pouvoir des patrons se trouve dans sa capacité de solutionner des problèmes qui n'ont pas encore été identifiés. Plusieurs types et niveaux de problèmes existent d'après (Kohls et Wedekind, 2010) :
 - a) Le cœur du problème : décrit ce qui est la problématique dans la situation actuelle et ce qui devrait être changé pour régler la situation.
 - b) Problème de trouver une solution : consiste rechercher des solutions qui peuvent être utilisées dans cette situation.
 - c) Problème de création de la solution : à ce niveau, la question « comment générer la solution ? » est abordée.
3. Solution : Elle désigne une conception d'un modèle ou de règles. Elle décrit un ensemble d'éléments organisés de manière à répondre à un ou plusieurs problèmes.

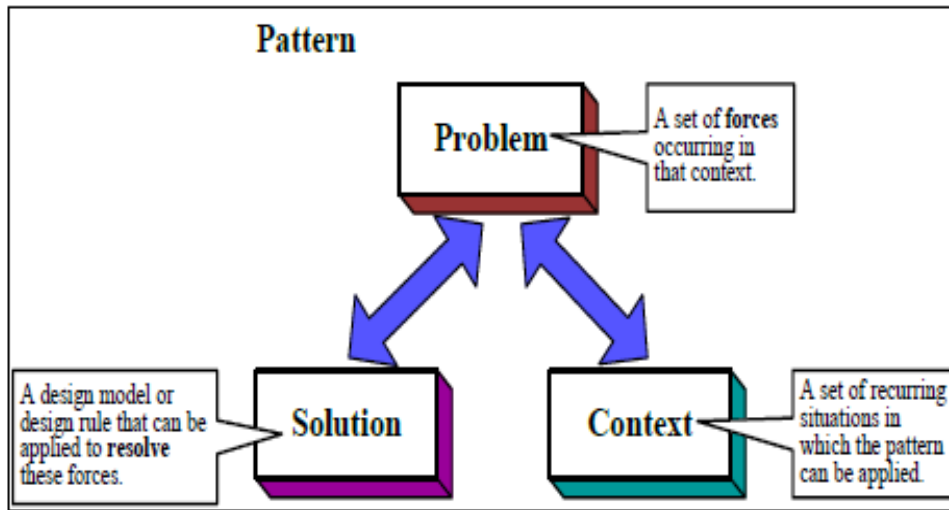


Figure 1.2 – Composition du patron.
Tirée de (Taleb, 2008)

L'objectif des patrons est de (Taleb *et al.*, 2006) :

- Partager les meilleures solutions de conception de logiciels interactifs auprès des professionnels HCI (Human-Computer Interaction) ;
- Fournir une base commune pour toutes les personnes impliquées dans la conception, le développement et les tests d'utilisabilité ;
- Fournir un système hautement interactif aux utilisateurs incluant les différents types d'applications.

1.1.2 Modèle

Le modèle est l'élément principal des systèmes interactifs. Il joue un rôle important dans le processus de développement et de maintenance des systèmes. Dans cette section, un modèle représente une description formelle de certains aspects clés d'un système interactif à partir d'un point de vue spécifique (Taleb *et al.*, 2009), (Taleb *et al.*, 2010c) et (Koch, 2006). Par exemple, un modèle d'architecture d'un système interactif complexe pourrait se concentrer

sur les aspects de concurrence, alors qu'un modèle financier d'une entreprise peut se concentrer sur les revenus projetés. La syntaxe du modèle inclut les notations graphiques ou tabulaires et le texte.

D'autres définitions sont données pour le terme « modèle » :

- « Dans l'architecture dirigée par des modèles, appelée MDA (Model-Driven Architecture), le modèle est une représentation d'une partie d'une fonction, d'une structure et/ou du comportement du système. » (OMG, 2008) (traduction libre) ;
- « Un modèle d'un système interactif est une description ou une spécification de ce système interactif et son environnement à certaines fins. Un modèle est souvent présenté comme une combinaison de dessins et de textes. Le texte peut être dans un langage de modélisation ou dans un langage naturel. » (Taleb, 2008) (traduction libre).

L'architecture POMA est basée sur la combinaison de deux approches :

- Orientée patrons : Cette approche décrit les niveaux d'architecture et les catégories de patrons en appliquant des règles de composition et de conversion (« mapping ») (Taleb *et al.*, 2006) (Taleb *et al.*, 2007b) (Taleb *et al.*, 2007c). Ces règles sont utilisées pour combiner et convertir (« mapper ») plusieurs types de patrons pour créer une conception orientée patron pour les systèmes interactifs, et pour générer des implémentations spécifiques pour différentes plateformes à partir de la même conception orientée patron.
- Dirigée par des modèles : la deuxième approche propose cinq (5) catégories de modèles (Taleb *et al.*, 2007a) (Taleb *et al.*, 2010b) en appliquant les règles de transformation (Taleb *et al.*, 2010c) et les règles de génération (Taleb *et al.*, 2010a) de ces modèles. Cela permet de résoudre divers problèmes tels que le découplage de divers aspects des systèmes (par exemple : la logique métier (« Business Logic »),

l'interface utilisateur, la navigation et l'architecture de l'information), et la séparation des aspects spécifiques liés aux plateformes des caractéristiques communes à tous les systèmes.

Dans cette section des définitions des concepts de patrons et de modèles ont été donnés, ainsi que les principales approches sur lesquelles POMA s'est basée. La section suivante détaillera les éléments constituant l'architecture POMA.

1.1.3 Composition de l'architecture POMA

L'architecture POMA consiste en six niveaux d'architecture de modèles utilisant des patrons afin de spécifier et de créer des systèmes interactifs. Cette architecture permet de supporter les utilisateurs novices et les experts du développement des systèmes interactifs.

Selon la (Figure 1.1), POMA est une architecture à 6-tiers, incluant ce qui suit :

- 1) six niveaux d'architecture et catégories de patrons (Taleb *et al.*, 2006) et (Taleb *et al.*, 2010b), à savoir :
 - La catégorie des patrons d'informations qui fournit des solutions à des questions telles que l'information qui peut ou doit être présentée sur quel périphérique (Ex. Renderer, etc.) ;
 - La catégorie des patrons d'interopérabilité qui décrit le découplage des niveaux d'un système interactif, en particulier, entre le contenu, le dialogue et les vues ou la présentation (Ex. patrons de Gamma (Bridge, Adapter, etc.) ;
 - La catégorie des patrons de visualisation qui décrit les différentes représentations visuelles de l'information (Ex. Bookmark, etc.) ;
 - La catégorie des patrons de présentation qui décrit les solutions pour savoir de quelle façon le contenu ou les services sont organisés, de façon visuelle (Ex. List, Graph, Map, Table, etc.) ;

- La catégorie des patrons d'interaction qui se focalise sur les mécanismes d'interaction qui peuvent être utilisés pour accomplir les tâches et les effets visuels (Ex. Search, etc.) ;
 - La catégorie des patrons de navigation qui décrit des techniques éprouvées pour la navigation avec un ensemble de pages et une partie de l'information (Ex. BreadCrumb (Path), Index Browser, etc.).
- 2) dix modèles dont cinq (5) sont des modèles indépendants de plateformes (PIM pour Platform Independent Model) et cinq (5) autres sont des modèles spécifiques de plateformes (PSM pour Platform Specific Model), à savoir :
- Le modèle de domaine qui encapsule les entités importantes de domaine du système en même temps que leurs attributs, leurs méthodes et leurs relations et définit les objets et les fonctionnalités accessibles par l'utilisateur via l'interface ;
 - Le modèle de tâches qui décrit la façon dont les activités peuvent être réalisées pour atteindre les objectifs de l'utilisateur lors de l'interaction avec le système interactif. Le modèle de tâche est constitué d'un ensemble de tâches que les utilisateurs ont besoin d'effectuer sur le système interactif ;
 - Le modèle de dialogue qui fournit des styles de dialogue pour accomplir les tâches et définit la structure de navigation de l'interface utilisateur ;
 - Le modèle de présentation qui décrit l'apparence visuelle de l'interface utilisateur ;
 - Le modèle de « Layout » qui définit le « Look-and-Feel » des systèmes interactifs.
- 3) quatre types de relations utilisés dans l'architecture POMA qui sont :
1. La composition : est utilisée pour combiner les différents patrons pour produire les modèles PIM en appliquant les règles de composition (Taleb *et al.*, 2006; Taleb *et al.*, 2007c). La composition des patrons est le processus de création d'un design de

modèles indépendants des plateformes (PIM) à partir d'un ensemble de patrons et de règles de composition. Cinq types de règles ont été utilisés dans POMA :

- (a) « Similar » : c'est une règle appliquée sur tous les patrons de la même catégorie ;
- (b) « Competitor » est une règle qui s'applique sur deux patrons de la même catégorie ;
- (c) « Super-ordinate » est une règle basique qui compose plusieurs patrons de différentes catégories ;
- (d) « Sub-ordinate » est une règle importante pour convertir («mapper») le processus de la conception orientée patron d'une plateforme à une autre ;
- (e) « Neighboring » s'applique sur des patrons qui découlent de la même catégorie parente.

2. La conversion « Mapping » est utilisée pour construire un modèle PIM qui deviendra un modèle PSM en utilisant les règles de conversion « mapping » (Taleb *et al.*, 2006). La conversion « mapping » des patrons est le processus de création d'un design de modèles spécifiques des plateformes (PSM) à partir des PIM et de règles de conversion « mapping ». Quatre types de règles ont été utilisés dans POMA pour construire un [POMA.PIM] qui deviendra un [POMA.PSM] en utilisant les règles de conversion « mapping » (PIM à PSM), à savoir :

- (a) « Identical » est une règle qui ne permet pas de changement à la conception originale. Exemple : Menus déroulants ;
- (b) « Scalable » est une règle qui permet des changements apportés à la taille de la conception originale ou au nombre d'items dans la conception originale. Exemple : long menu horizontal → réduction le nombre d'éléments du menu (PDA) ;

- (c) « Multiple » est une règle qui permet de répéter la conception originale, soit simultanément ou séquentiellement. Exemple : long menu (Laptop) → série de petits menus (PDA) ;
- (d) « Fundamental » est une règle qui permet de changer la nature de la conception originale. Exemple : menus verticaux permanents de gauche (Laptop) → menu déroulant (PDA).

3. La transformation est utilisée pour établir la relation entre deux modèles [POMA.PIM] et/ou deux modèles [POMA.PSM] en utilisant les règles de transformation (Taleb *et al.*, 2010c). La transformation de modèles est le processus de conversion d'un ou de plusieurs modèles appelés modèles sources en modèles cibles d'un même système interactif en tenant compte des règles de transformation. Six (6) types de règles existent : « *Tracker* », « *Sequencer* », « *Parser* », « *Linker* », « *Controller* », « *Atomicity* ». Elles sont utilisées pour permettre aux cinq (5) modèles PIM et aux 5 modèles PSM de se transformer entre eux dans le même système, c.-à-d. en établissant la relation entre deux modèles du même type ([POMA.PIM] → [POMA.PIM] et/ou [POMA.PSM] → [POMA.PSM]), et également pour créer de nouveaux modèles à partir d'autres modèles existants.

4. La génération est utilisée pour générer le code source de tout le système interactif à partir de tous les modèles PSM en appliquant les règles de génération (Taleb *et al.*, 2010a). « La génération de code des modèles est le processus de création d'un code source à partir d'un modèle en utilisant les règles de génération » (traduction libre). Il existe quatre (4) types de règles qui sont : « *Analyser* », « *Interpreter* », « *Parser* », « *Transmitter* », elles sont utilisées pour supporter la génération automatique du code source de tout le système interactif à partir de tous les modèles [POMA.PSM].

Dans cette section, les composants essentiels de l'architecture POMA ont été donnés et les quatre types de règles utilisées. La section suivante porte sur l'architecture orientée service (SOA pour Service-Oriented Architecture) et ses concepts clés qui sont essentiels pour la création de la nouvelle architecture POMA orientée services.

1.2 Architecture orientée services

Cette section présente une revue de la littérature de l'architecture orientée services (SOA), incluant ses définitions, ses concepts clés, ses avantages et ses limites.

Le développement des applications interactives dans le cas des architectures orientées services nécessite des composants combinés et réorganisés de manière cohérente et flexible qui doivent travailler ensembles, appelés « Services ». « L'architecture orientée services (SOA) peut être différentes choses pour différentes personnes en fonction du rôle des personnes et du contexte (affaires, architecture, implémentation, opérationnel) » (Ganci *et al.*, 2006) (traduction libre).

Du point de vue métier, (Sandy, 2007) définit la SOA comme une approche architecturale TI orientée métier supportant l'intégration du système d'information de l'entreprise et permet de l'aligner rapidement avec les besoins d'affaires changeants. (Ganci *et al.*, 2006) rejoint la définition de Sandy en indiquant que « SOA defines a set of business services composed to capture the business design that the enterprise wants to expose internally, as well as its customers and partners ». Cette définition indique que la SOA est une composition d'un ensemble de services métiers afin de capturer la conception d'affaires que l'entreprise désire exposer en interne, ainsi que ses clients et partenaires.

Du point de vue informatique (Rouillard *et al.*, 2007, p. 5) définit la SOA comme

Une manière de réorganiser les applications logicielles qui sont isolées et de supporter une infrastructure au sein d'un ensemble de services interconnectés, chacun accessible à travers des standards et des protocoles d'échange de messages.

(Papazoglou, 2003, p. 3) définit la SOA comme suit :

« La SOA est un moyen de réorganiser un portefeuille d'applications logicielles anciennement en silo et de supporter l'infrastructure dans un ensemble de services interconnectés, chacun accessible à travers des interfaces standards et des protocoles de messagerie. »

(Barry, 2010) rejoint ces définitions en indiquant que la SOA est «...un ensemble de services. Ces services communiquent entre eux. La communication peut impliquer soit le passage de données simple ou il pourrait impliquer deux ou plusieurs services coordonnant certaines activités » (traduction libre).

De son côté, (Bachman *et al.*, 2005, p. 5) définit la SOA comme suit :

« SOA is an evolution of distributed computing designed to allow the interaction of software components, called “services”, across a network. Applications are created from a composition of these services — and importantly, the services can be shared among multiple applications. »

Cette définition précise que la SOA découle des systèmes distribués. Elle est conçue pour permettre l'interaction des services à travers un réseau. La création des applications dans la SOA se fait par la composition des services partagés.

Du point de vue architecture, « La SOA pourrait aussi être vue comme un style d'architecture pour la création d'applications logicielles qui utilisent des services disponibles dans le réseau. » (Haines et Rothenberger, 2010) et (Papazoglou, 2003) (traduction libre).

(Ganci *et al.*, 2006, p. 5), à son tour, définit la SOA comme suit :

« La SOA est une architecture qui prend en charge l'orientation service. Au niveau de l'implémentation, la SOA est accomplie en utilisant une infrastructure basée sur des standards, un modèle de programmation et des technologies telles que les services Web » (traduction libre).

Du point de vue opérationnel, la « SOA comprend un ensemble de contrats entre les consommateurs et les fournisseurs de services qui spécifient la qualité du service, ainsi que

l'établissement des rapports sur les activités d'affaires clés et les métriques IT. » (Ganci *et al.*, 2006) (traduction libre).

D'autres définitions ont été données comme celle de (MacKenzie *et al.*, 2006) qui a défini la SOA comme étant « un paradigme pour organiser et utiliser des capacités distribuées qui peuvent être sous le contrôle de différents domaines propriétaires » (traduction libre). Ces capacités sont créées par différentes entités (personnes et organisations) afin de résoudre ou de soutenir une solution pour les problèmes rencontrés dans le cadre de leurs activités.

La section suivante présentera deux différentes classifications en couches des catégories de services selon (Erl, 2005) et (OMG, 2009).

1.2.1 Vue globale de l'architecture orientée services

Dans une perspective plus globale, (Erl, 2005) classe les services en trois catégories ou couches principales qui sont définies à partir de la réutilisation du service, du modèle logique que représentent le service et le domaine fonctionnel.

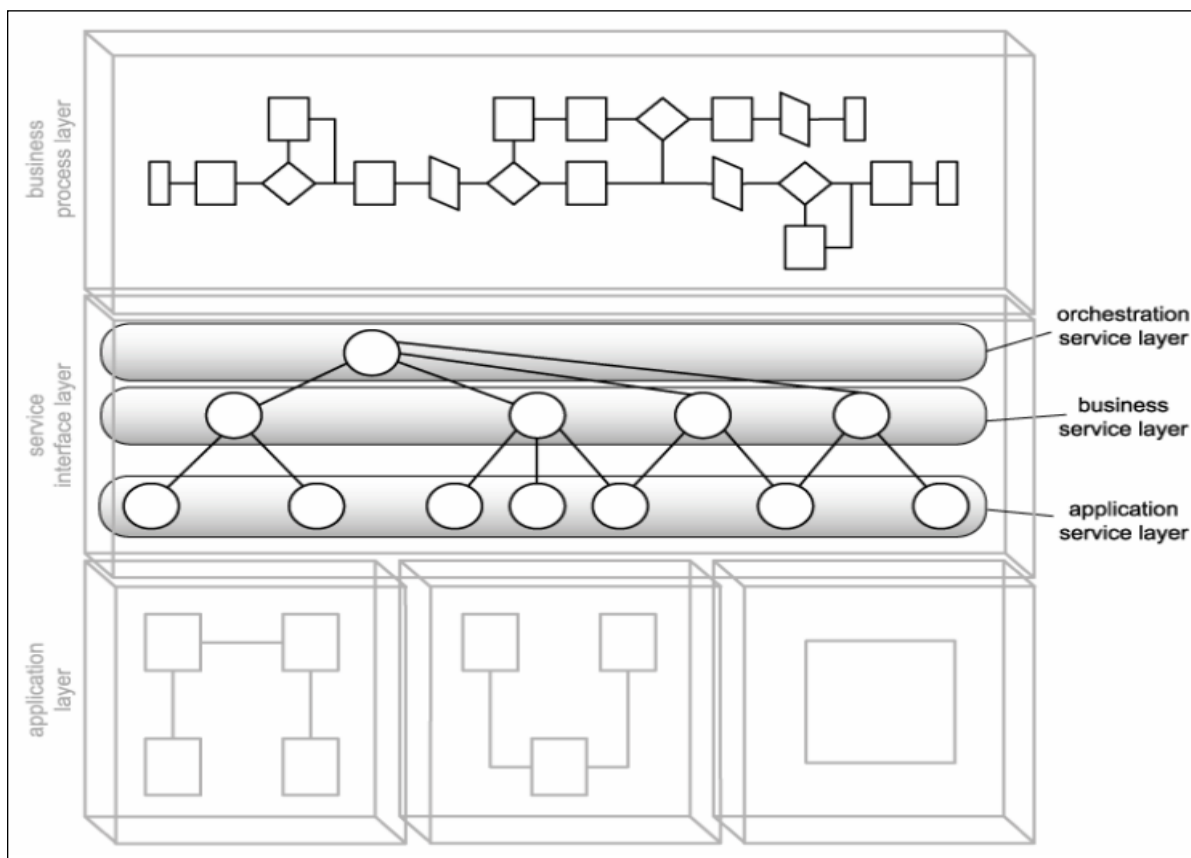


Figure 1.3 – Les couches d'une SOA.
Tirée de (Erl, 2005, p. 337)

La (Figure 1.3) présente les trois couches primaires de la SOA (Erl, 2005), à savoir:

- 1) La « couche application » qui présente la logique métier (« Business Logic ») de l'entreprise. Elle contient des applications et des bases de données qui composent le système d'information de l'entreprise.
- 2) La « couche services » qui permet d'exprimer les fonctionnalités spécifiques à la technologie de l'entreprise. Cette couche se subdivise, à son tour, en trois sous-couches :
 - a) La « sous-couche service d'application » qui expose les fonctionnalités de base adaptées au système d'information de l'entreprise. Les services de cette

sous-couche présentent des fonctionnalités dans un contexte de traitement spécifique, en permettant leur réutilisation et leur intégration avec d'autres services. La création de ces fonctionnalités peut s'effectuer à partir d'un mélange de services développés de manière personnalisée et d'autres services achetés ou loués.

b) La « sous-couche des services d'affaires » : permet d'exprimer la logique métier (« Business Logic ») de l'entreprise à travers l'orientation du service et la possibilité de représenter son modèle d'affaires sous forme de services Web. Les activités des processus d'affaires décrites par ces services sont indépendantes de la technologie existante. L'abstraction de cette sous-couche de service permet de distinguer deux sous-couches des services d'affaires (Couturier, 2011) et (Erl, 2005):

- La « sous-sous-couche des services d'affaires orientée entités » : représente généralement les entités du domaine d'affaires. En d'autres mots, les services métiers orientés entités englobent la logique de fonctionnement de l'entité telles que le client, le fournisseur, etc.
- La « sous-sous-couche des services d'affaires orientés tâches » : regroupe les opérations faisant interagir des services d'affaires des entités ou d'autres services de tâches. En d'autres mots, les services d'affaires orientés tâches sont spécifiques à une ou plusieurs activités telles que la gestion des commandes.

c) La « sous-couche de service d'orchestration » :

« La sous-couche de service d'orchestration présente un haut niveau d'abstraction qui pallie le besoin d'autres services pour gérer les détails d'interaction nécessaires pour s'assurer que les opérations de service soient exécutées dans un ordre spécifique (Erl, 2005) »
(Traduction libre).

3) La « couche processus d'affaires » présente les différentes activités et processus d'affaires de l'entreprise.

Une autre façon d'organiser les différentes couches de l'architecture orientée services est proposée par le groupe OMG (« Object Management Group »). Ce dernier divise l'environnement SOA en quatre principales couches telles que montrées dans la Figure 1.4 :

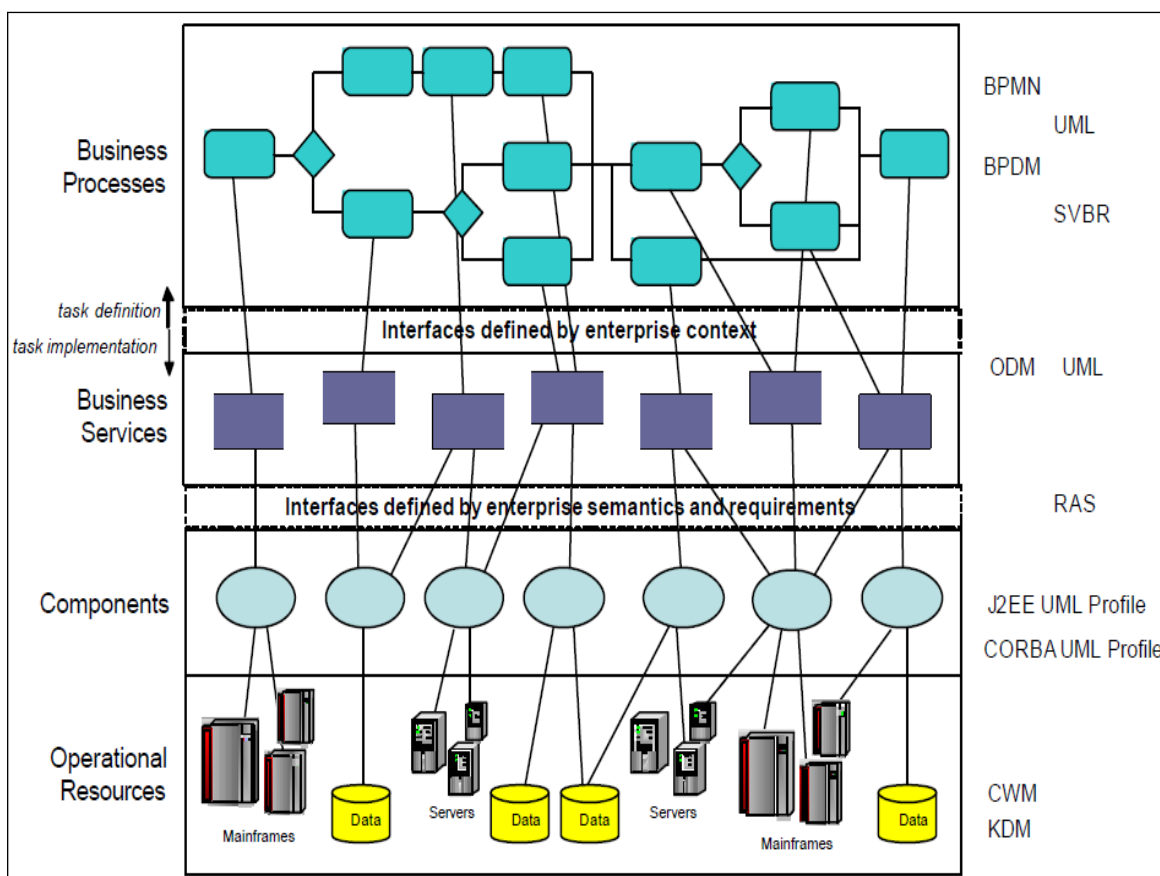


Figure 1.4 – Les couches de SOA.
Tirée de (OMG, 2009)

- 1) La couche « ressources opérationnelles » représente le niveau le plus bas de la SOA et illustre les applications, les packages et les bases de données qui pourraient être appelés par les différents composants.
- 2) La couche « composants » définit des composants logiciels et des orchestrations permettant aux services d'affaires d'être reliés aux ressources partagées au niveau des entreprises et de les appeler au besoin.

- 3) La couche « services d'affaires » définit les services d'affaires capables d'automatiser les activités spécifiques des processus d'affaires.
- 4) La couche « processus d'affaires » décrit les processus d'affaires constitués d'une séquence d'activités d'affaires.

Deux façons d'organiser les services en couches dans l'architecture orientée services ont été vue selon les auteurs (Erl, 2005) et (OMG, 2009). La section suivante détaille les principaux concepts clés utilisés dans la SOA tout en apportant une valeur ajoutée à l'architecture POMA.

1.2.1.1 Principes de conception d'une SOA

La Figure 1.5 montre les principaux concepts de l'architecture orientée services (Ganci *et al.*, 2006) :

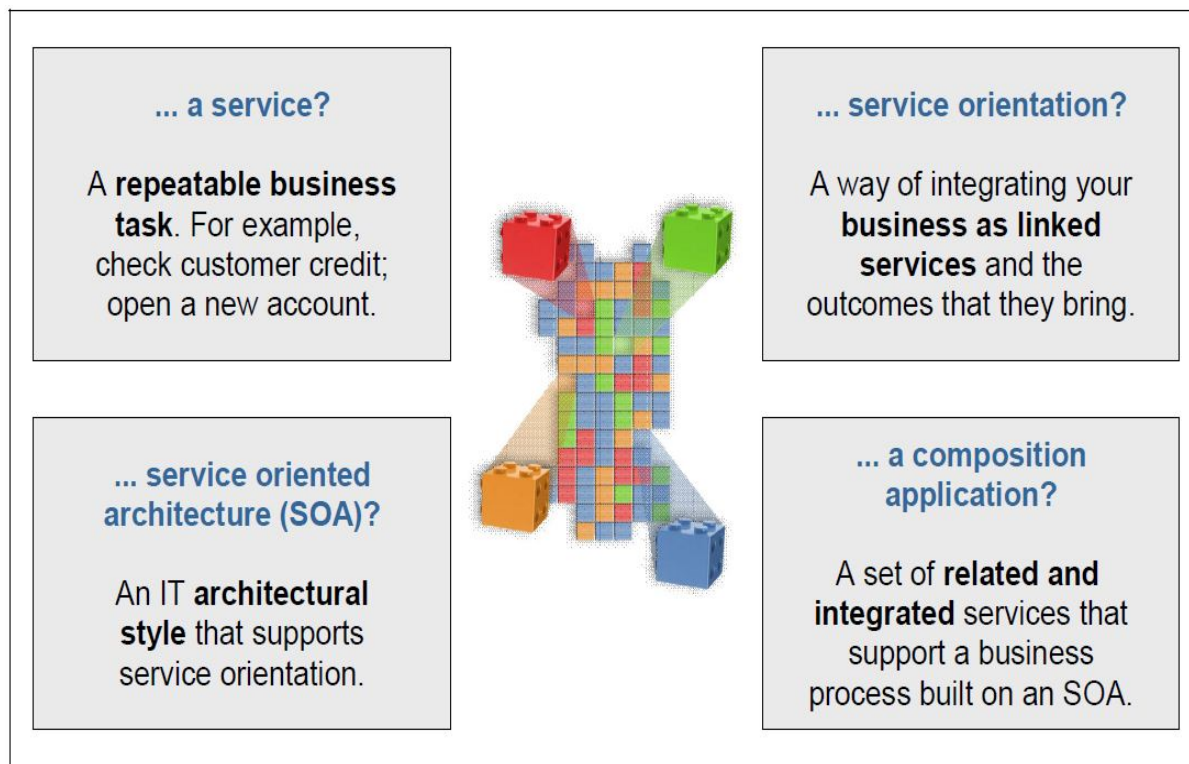


Figure 1.5 – Définition des termes clés de l'architecture orientée services.
Tirée de (Ganci *et al.*, 2006)

a. Service

Le service étant le composant le plus important dans une architecture orientée services. (Rouillard *et al.*, 2007) ont introduit la notion de service comme étant la mise à la disposition d'agents humains ou logiciels des processus métiers.

De même, (Papazoglou, 2003) a défini un service comme étant une fonction d'affaire implémentée dans un logiciel ayant une interface documentée et formelle. Cette fonction est connue par les concepteurs et les utilisateurs et facilement accessible en sachant où elle se trouve.

Un service représente une tâche d'affaire répétable (Ganci *et al.*, 2006). Il permet d'encapsuler les unités fonctionnelles d'une application en fournissant des interfaces faciles et une implémentation indépendante.

Deux types de services existent avec ou sans état, suivant (Lemrabet, 2012) : (1) le terme sans état (« *Stateless* ») désigne une condition d'exécution du service. Un service sans état ne traite pas l'état des données associé à une activité. Autrement dit, il ne conserve pas en mémoire son exécution précédente ; (2) les services avec état (« *Stateful* ») désignent les services qui traitent et conservent l'état des données.

b. Types de processus

(White et Miers, 2008) ont décrit trois types de processus pouvant être modélisés dans une architecture orientée services :

1. Orchestration : l'orchestration des processus permet de décrire la manière dont les processus d'une entreprise sont exécutés en interne. Dans le cas de l'architecture orientée services, l'orchestration permet de coordonner plusieurs services pour réaliser les besoins d'affaires de l'entreprise (Couturier, 2011) ;

2. *Chorégraphie* : la chorégraphie permet de gérer les échanges des flux de travail entre différents intervenants, tels que les partenaires, les clients et les fournisseurs (Couturier, 2011) et (Guilbault, 2007) ;

La Figure 1.6 permet de distinguer la différence entre ces deux types de processus. Dans l'exemple de la Figure 1.6, deux entreprises ayant chacune son propre processus d'affaires en interne (Orchestration). La chorégraphie consiste à échanger des flux d'informations entre les deux entreprises ce qui permet d'intégrer leur processus d'affaires.

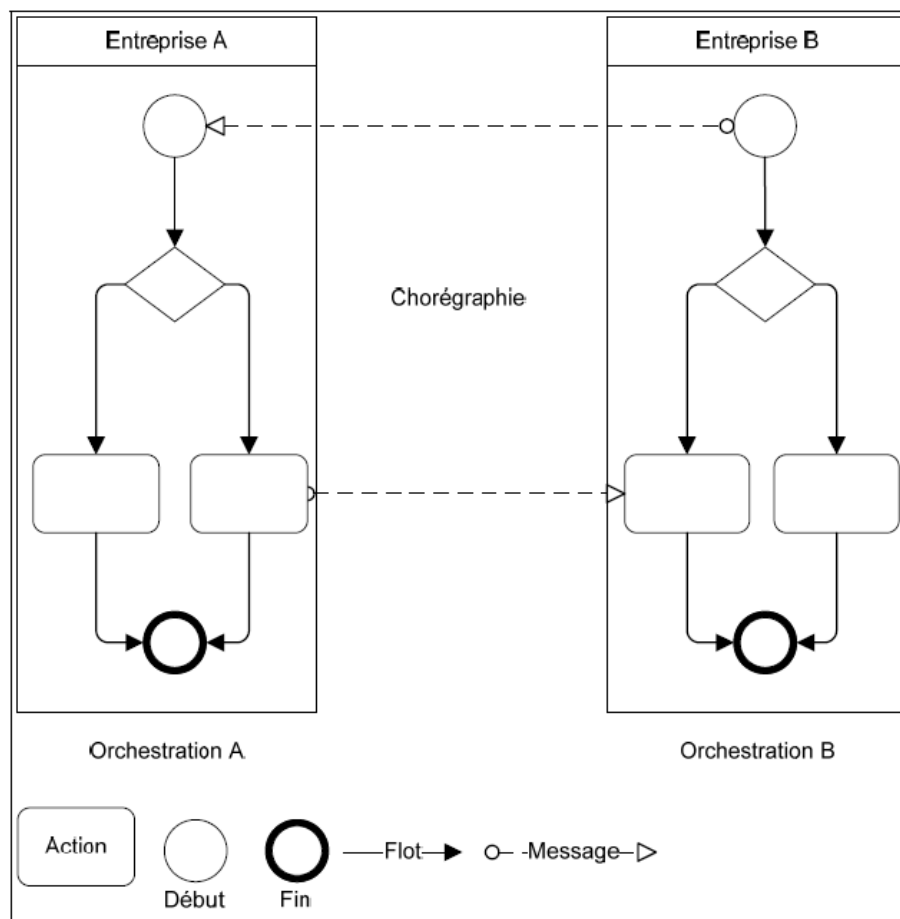


Figure 1.6 – Différence entre l'orchestration et la chorégraphie.
Tirée de (Couturier, 2011)

3. *Collaboration* : « La collaboration (Figure 1.7) permet de représenter les interactions entre les différents participants d'un processus, sans nécessairement avoir une connaissance détaillée d'un des processus » (Couturier, 2011). le terme « participants » dans cette définition rejoint le même terme « intervenants » introduit par (Couturier, 2011) et (Guilbault, 2007).

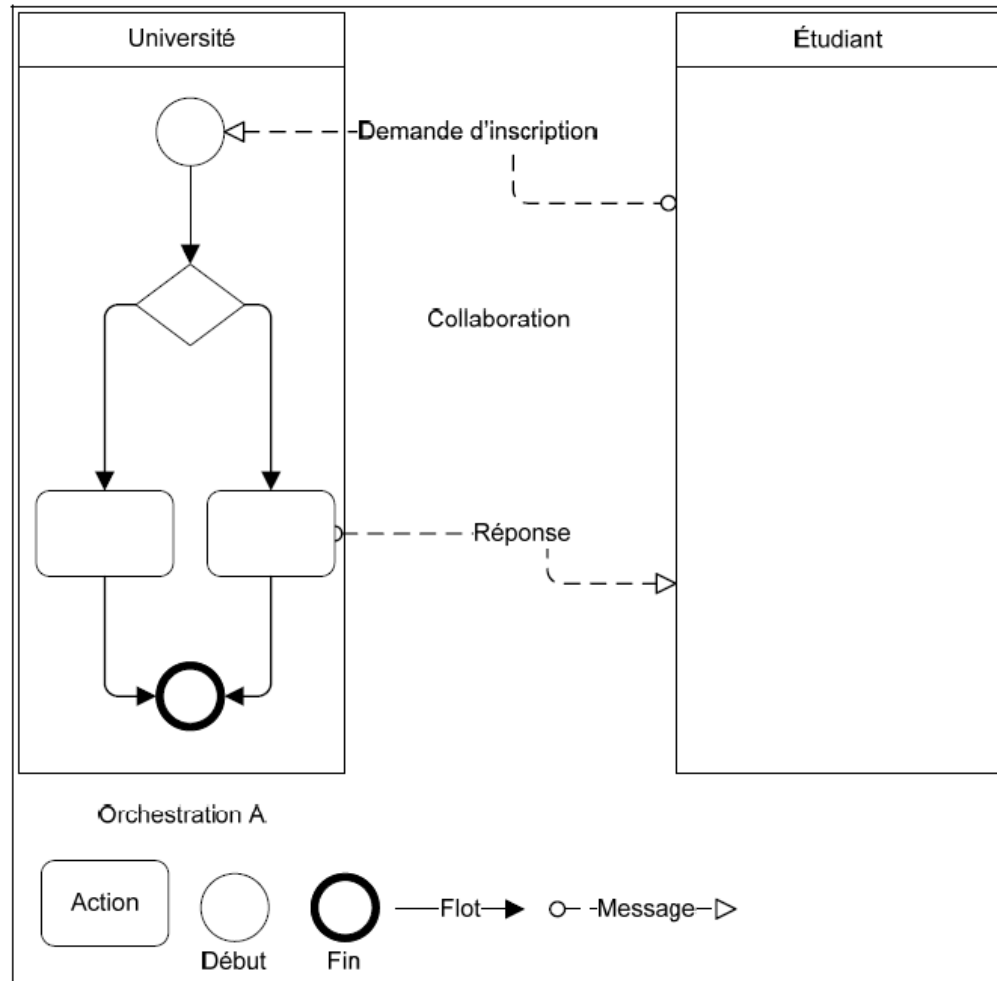


Figure 1.7 – Collaboration.
Tirée de (Couturier, 2011)

c. Politique de service

(MacKenzie *et al.*, 2006) proposent trois aspects de la politique de service, à considérer lors de la mise en œuvre d'une architecture orientée services, à savoir :

- 1) *L'assertion de la politique* : décrit la façon dont le service est réalisé en faisant le lien entre le service et son contexte d'exécution.
- 2) *Le titulaire de la politique* : désigne un participant (intervenant) qui adopte une assertion comme sa propre politique.
- 3) *L'application de la politique* : nécessite des techniques qui dépendent de la nature de la politique de service. Cela permet d'assurer que l'assertion de la politique est conforme et que son exécution se fera dans de bonnes conditions. Elle permet aussi d'effectuer des actions, en cas de violation de la politique.

d. Contrat de service

Un contrat représente un accord permettant de gérer les exigences et les attentes entre deux ou plusieurs participants (fournisseurs du service, registraire de services et consommateurs de services), contrairement à l'application de la politique qui est associée à un point de vue d'un individu et habituellement est la responsabilité du titulaire du contrat. L'exécution des contrats peut impliquer la résolution des différends entre les différentes parties du contrat (MacKenzie *et al.*, 2006). Un contrat de service (Heubès, 2009) est composé d'un contrat syntaxique qui propose une représentation technique du service, d'un contrat sémantique qui fournit une description informelle du traitement et d'un contrat de niveau de service (QoS pour Qualité de Service et SLA pour Service Level Agreement) qui précise les engagements du service.

e. Couplage

En génie logiciel, le terme couplage désigne la manière et les degrés auxquelles différents modules logiciels dépendent les uns des autres (Lemrabet, 2012) et (Radatz *et al.*, 1990).

Deux composants basés sur la SOA peuvent être fortement couplés s'ils échangent beaucoup d'informations ou bien faiblement couplés si la quantité d'informations échangée entre les deux est très faible.

(Webopedia) a défini le couplage faible en informatique, la manière dont plusieurs systèmes informatiques utilisant différentes technologies peuvent se réunir pour exécuter les mêmes transactions indépendamment du matériel, du logiciel et d'autres composants fonctionnels.

Deux catégories de couplage existent selon (Lemrabet, 2012) (Schmelzer, 2007) et (Erl, 2005), soit le couplage interne concernant le contrat de service ou le couplage externe concernant le consommateur du service.

f. Abstraction

Le terme « abstraction » est défini par (Radatz *et al.*, 1990) comme « une vue d'un objet qui se concentre sur l'information pertinente à un usage particulier et ignore le reste de l'information » (traduction libre). L'abstraction d'un service se concentre sur la quantité d'informations utiles disponibles sur ce dernier, en cachant tous les détails inutiles du contrat de service. De cette manière, le couplage entre les utilisateurs et le contrat de service sera de plus en plus faible.

g. Réutilisabilité

Plusieurs ouvrages, articles scientifiques, études de cas et des recherches sont axés sur l'architecture orientée services (SOA), plus particulièrement sur le principe de la réutilisation des services. Ils permettent de discuter des avantages de transformer les capacités TI de l'entreprise autant que services afin de les réutiliser et les combiner pour créer de nouvelles applications (Shah, 2007).

(Iheb, 2011), à son tour, définit la réutilisabilité comme suit :

La réutilisabilité des services fait allusion à la capacité d'un service d'être utilisé par plusieurs intervenants pour plusieurs objectifs. Ceci, afin d'assurer un meilleur retour sur l'investissement et augmenter l'agilité d'affaires de l'entreprise, assurant ainsi une réponse rapide aux futures exigences métiers de cette dernière. .

h. Autonomie d'un service

Le principe d'autonomie de service s'applique dans la conception orientée services et permet aux services de s'autogouverner en ayant la possibilité de contrôler leurs propres ressources et logique métier. Cela assure une meilleure qualité de service avec une plus grande fiabilité et performance étant donné que les services deviennent moins dépendants des ressources partagées. (Iheb, 2011) et (Erl, 2008) ont proposé quatre niveaux d'autonomie d'un service, à savoir :

- 1) Autonomie partagée : dans ce cas d'autonomie, les services partagent leur logique et leurs ressources (Figure 1.8-a).
- 2) Autonomie de la logique du service : cette catégorie de services permet de partager uniquement les ressources utilisées. Par contre, la logique métier du service est isolée (Figure 1.8-b).
- 3) Autonomie de contrat de service : aussi appelée « autonomie par isolation fonctionnelle » (Couturier, 2011). Les contrats de services dans ce cas-ci sont alignés les uns à côté des autres afin d'éviter le croisement des fonctionnalités de chacun des services (Iheb, 2011) (Figure 1.8-c).
- 4) Autonomie pure : ce type d'autonomie est le plus fort qu'on peut trouver dans une architecture orientée services. Les services dans ce cas ont un contrôle total sur les ressources et la logique métier de l'entreprise. Ce type d'autonomie est le plus haut dans une architecture orientée services. Un service, dans ce cas, a un contrôle total sur les ressources ainsi que la logique métier de l'entreprise (Figure 1.8-d).

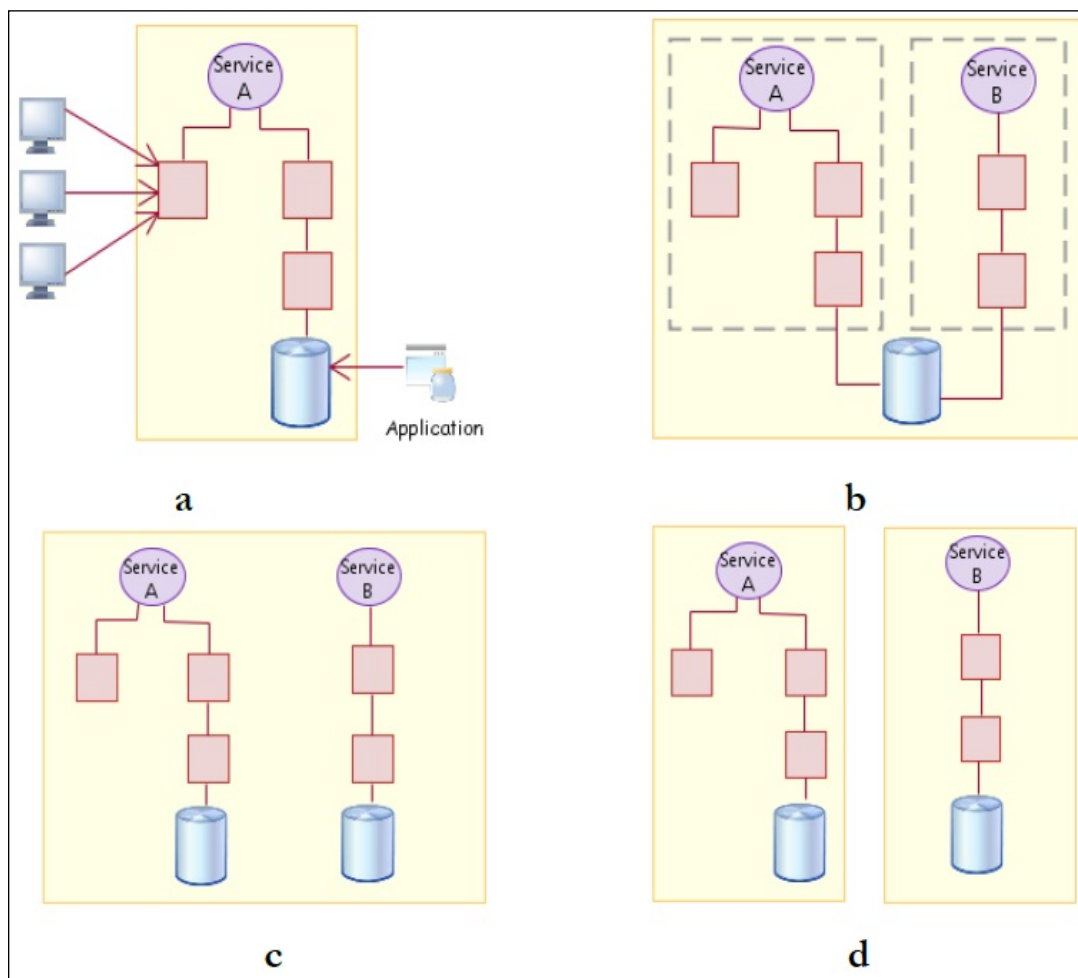


Figure 1.8 – Autonomie de service.
Tirée de (Erl, 2008; Lemrabet, 2012)

i. Découverte d'un service

Le principe de découverte permet à la logique d'un service d'être accessible par de nouveaux demandeurs (consommateurs) recherchant des services qui correspondent à leurs besoins.

C'est pourquoi la possibilité de découvrir des services est étroitement liée aux principes orientés services suivants (Erl, 2005) :

- Le contrat de services est l'élément le plus recherché à découvrir par les demandeurs de services ;
- La réutilisabilité des services est le principe le plus recherché par les demandeurs lors de la phase de découverte. C'est ce qui rend un service très utile lors de sa découverte. Autrement, il n'aura aucune raison d'être recherché et sera créé pour un demandeur de service spécifique.

j. Composition de services

La composition des services est fondamentale pour la conception de nouvelles solutions orientées services. (Lemrabet, 2012) a indiqué qu'un service peut-être sous deux formes (composable ou composé), tout dépend de sa capacité :

- Composé : si sa capacité fait appel aux capacités d'autres services ;
- Composable : dans le cas où sa capacité est appelée par d'autres services.

Un service peut être composable et composé en même temps. (Lemrabet, 2012) a rajouté à cela qu'un service peut-être composable non réutilisable si le service est créé pour une composition fortement couplée à un consommateur, ou réutilisable non composable si le service est utilisé par plusieurs consommateurs, mais participe uniquement à des activités de type point à point.

k. Entreprise Service Bus (ESB)

Le concept de l'ESB (Entreprise Service Bus) ne fait pas partie intégrante de la SOA, car les applications basées sur la SOA peuvent être implémentées sans nécessairement utiliser ce concept. L'ESB joue un rôle important dans la SOA, car il permet d'interconnecter plusieurs applications basées sur cette architecture et fournit des services sécurisés de transfert de messages et d'interopérabilité en utilisant les services Web et les technologies liées. Cela permet à différents systèmes de communiquer et de partager leurs informations (Alghamdi *et al.*, 2010).

D'après (Kumara et Gamage, 2010, p. 25) et (Rosen *et al.*, 2008)

« L'ESB est une infrastructure middleware extensible à l'échelle de l'entreprise fournissant la virtualisation et la gestion des interactions de service, y compris le support pour les technologies de la communication, de médiation, de transformation et d'intégration requises par les services ».

La virtualisation des services dans cette définition consiste à exécuter un ou plusieurs services dans le réseau comme une simple application.

(Maréchaux, 2006) a identifié les principaux concepts pour l'implémentation d'un ESB. Ce dernier doit au moins fournir les services suivants :

- Services de transports : ces services doivent s'assurer que les messages soient livrés entre les différents processus d'affaires qui sont interconnectés. Ils permettent également de router les messages de manière sécurisée entre les destinataires ;
- Services des évènements : ces services ont la capacité de détecter, déclencher et de distribuer les évènements. Ces derniers sont liés à la notion de traitement des évènements, une technique d'analyse et de contrôle de séries complexes d'évènements inter-reliés. « Un évènement est un changement significatif de l'état d'un système ou de son environnement » (Mani Chandy) (traduction libre) ;
- Services de médiation : ces services permettent d'abord de déterminer les différents protocoles utilisés pour intégrer les systèmes hétérogènes concernés. Ensuite, ils effectuent la conversion d'un protocole à un autre afin d'établir la communication entre les services participants. Selon (IBM), le flux de médiation permet d'intercepter et de modifier les messages transmis entre des services existants (ex : fournisseurs et clients) qui souhaitent utiliser les services.

(Hérault *et al.*, 2005) ont rajouté à la fonctionnalité de transformation des services de médiation, les éléments suivants (Figure 1.9) :

- Sécurité : cet élément est primordial lorsque plusieurs entreprises possédant des systèmes hétérogènes désirent interagir (ex. : cryptographie, autorisation) ;
- Routage dynamique : cela consiste à calculer les routes de manière automatique en utilisant des protocoles de routage afin acheminer les demandes à un ou plusieurs destinataires dans un réseau ;
- D'autres actions non fonctionnelles sont incluses dans les services de médiation. Ces actions concernent la gestion de la qualité du service QoS, telles que la mesure de la qualité, la détection de défaillance et la récupération, la gestion de données incomplètes, etc.

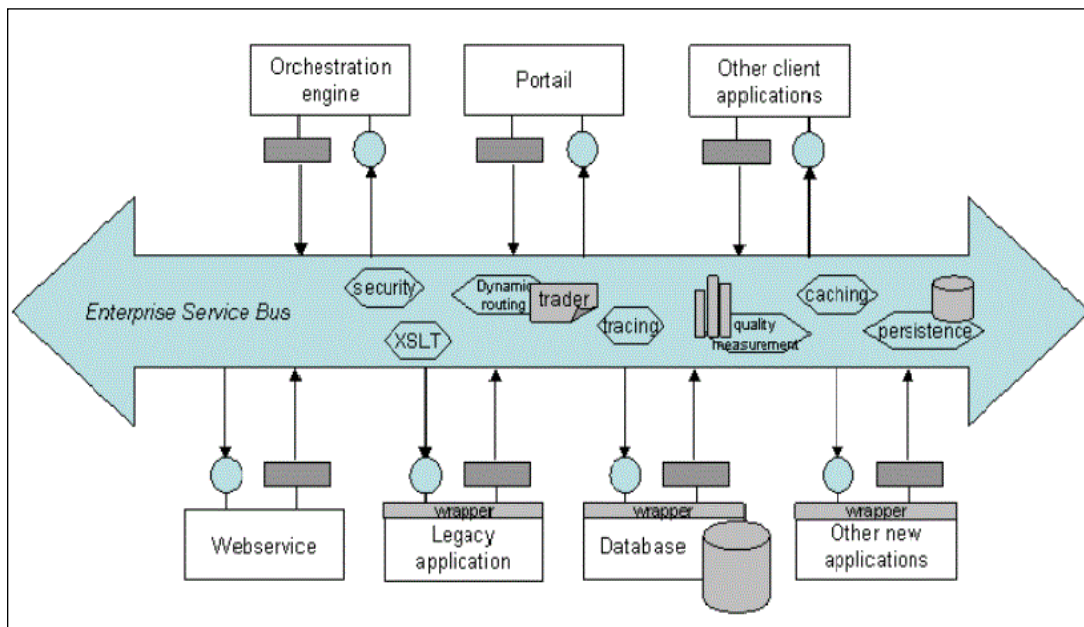


Figure 1.9 – Les composants de l'Entreprise Service Bus.

Tirée de (Hérault *et al.*, 2005)

La Figure 1.10 montre un modèle d'ESB basé sur la SOA.

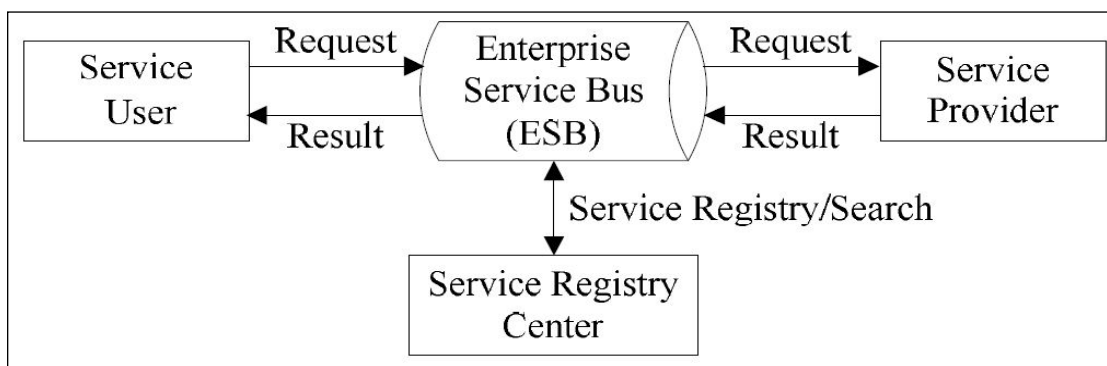


Figure 1.10 – Modèle d'ESB basé sur la SOA.
Tirée de (Pingshun, 2011)

Les détails des principaux concepts de l'architecture orientée services ont été présentés dans cette section, la suivante portera sur les relations entre les trois types d'intervenants de l'architecture orientée services et les opérations qu'ils utilisent.

1.2.1.2 Relation et opérations dans la SOA

L'architecture orientée services (SOA) met en relation trois types d'intervenants qui sont : le fournisseur de services, l'agent de découverte de services et le demandeur de service (Papazoglou, 2003). La Figure 1.11 montre les interactions et opérations (publier « *publish* », trouver « *find* », relier « *bind* ») entre les différents intervenants de la SOA qui sont :

- Fournisseur de service : est un agent permettant de fournir des services en définissant leurs descriptions et en les publiant à un agent de découverte de service ou directement à un client en utilisant l'opération publier (« *publish* ») ;

- Registraire de services (agent de découverte) : fournit les interfaces de publication et de découverte des services. Il met le fournisseur et le demandeur de service en contact ;
- Le demandeur de service (client) : recherche une description de service qui répond à ses besoins en utilisant l'opération « trouver » (« *find* ») les services à partir d'un agent de découverte. Il utilise l'opération « relier » (« *bind* ») pour se relier avec le fournisseur de ce service, ce qui lui permettrait d'invoquer et exécuter le service, et retourner les résultats.

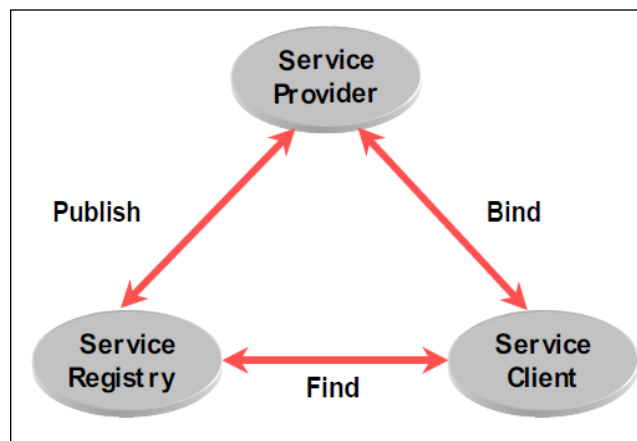


Figure 1.11 – Les différents participants de la SOA.
Tirée de (Papazoglou, 2003)

(Papazoglou, 2003) a défini certaines contraintes que les services doivent vérifier afin de pouvoir bénéficier des avantages de l'orienté services :

- Technologie neutre : les services doivent être invocables par la majorité des technologies standardisées, quel que soit l'environnement informatique. « Cela implique que les mécanismes d'invocation (protocoles, des descriptions et des mécanismes de découverte) doivent se conformer aux normes généralement reconnues » (Papazoglou, 2003)(traduction libre).

- Couplage lâche (faible) : la création de services ne doit pas exiger aucune connaissance sur le demandeur de services ni sur le service en question.
- Prise en charge de la transparence de l'emplacement : les services doivent partager leurs descriptions et les informations concernant leur emplacement dans un répertoire comme UDDI. Ils doivent être accessibles par des demandeurs de services indépendamment de leur emplacement.

La section suivante décrit les avantages de l'architecture orientée services qui peuvent être bénéfiques pour l'architecture POMA.

1.2.1.3 Avantages des architectures orientées services

Les principaux avantages de SOA (MacKenzie *et al.*, 2006) permettent de :

- Faciliter la croissance des systèmes informatiques à l'échelle des entreprises. Cela permet de rendre plus simple l'utilisation des services au niveau d'Internet et de réduire les coûts de la coopération qui relie les clients aux fournisseurs de services afin d'assurer les besoins qui les intéressent ;
- L'utilisation des principes de la SOA permet aux architectes de développer des systèmes évolutifs et facilement gérables. Cela permet de faciliter l'intégration des nouvelles fonctionnalités au-delà des limites des systèmes d'informations des entreprises (CRM (Customer Relationship Management), SAP (Systems, Applications and Products for data processing), JEdwards, etc.) ;
- Grâce à la capacité évolutive de la SOA, cette architecture permet un portefeuille TI de s'adapter aux besoins d'un domaine spécifique ou à l'architecture des processus.

D'autres avantages de la SOA ont été identifiés par (Erl, 2008) (Iheb, 2011) :

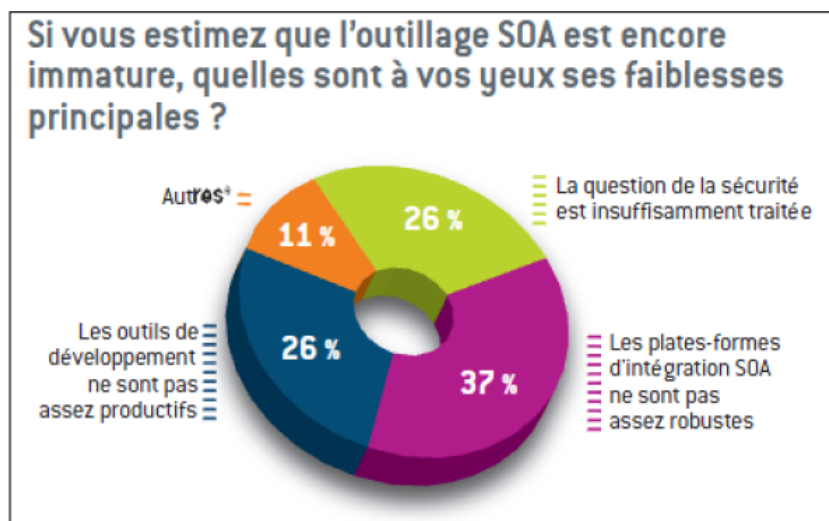
- Assurer une interopérabilité intrinsèque : les principes de la SOA permettent de développer une variété de solutions informatiques en assurant un échange de données et de fonctionnalités, quel que soit le langage de programmation ;
- Assurer un alignement entre le métier et les TI : l'introduction de la couche des services de la SOA permet une abstraction des détails techniques des différentes fonctionnalités et assure une meilleure communication entre les TI et le métier en utilisant des représentations graphiques des processus d'affaires sous forme de services ;
- Minimiser l'investissement initial et permettre la réutilisation : la SOA permet la réutilisation des composants existant, ce qui facilite la création de nouveaux services tout en réduisant les coûts et le temps de développement et des tests ;
- Accroître l'agilité de l'organisation : la SOA permet à l'entreprise d'être plus efficace et flexible, vue qu'elle pourra répondre rapidement aux changements ;
- Minimiser les risques de défaillance : la réutilisation des services déjà existants, testés et approuvés permet de réduire les risques de défaillance des nouveaux services.

Bien que l'architecture POMA bénéficie des avantages considérables de l'architecture orientée services, POMA hérite également les limites de la SOA. La section suivante résume ces limites selon un sondage réalisé par (Plouin et Pascal, 2007).

1.2.1.4 Limites de l'architecture orientée services

Un sondage réalisé en 2006 sur 85 entreprises (Plouin et Pascal, 2007) (Figure 1.12) montre que les principales limites de l'architecture orientée services sont :

- La sécurité dans la SOA est un élément insuffisamment traité par les éditeurs et manque de performance ;
- Les plateformes d'intégration de la SOA ne sont pas assez robustes ;
- Les outils de développement utilisés pour le développement de solutions orientées services ne sont pas assez productifs ;
- D'autres éléments ont été signalés par ces éditeurs.



**Figure 1.12 – Les principales limites de l'architecture orientée services.
Tirée de (Plouin et Pascal, 2007)**

Dans les sections précédentes, une revue littérature a été présentée sur les principaux concepts de la SOA, les différents intervenants et les opérations qu'ils utilisent et les avantages et les limites de la SOA. Les sections suivantes introduiront le concept des services Web, les protocoles qu'ils utilisent avec des scénarios illustrant le fonctionnement des services Web.

1.2.2 Services Web

« Lorsque les services encapsulent les fonctionnalités d'affaires d'une entreprise, une certaine forme d'infrastructure interservices est nécessaire pour faciliter les interactions et la communication » (Papazoglou, 2003) (traduction libre). Les services Web représentent un cas particulier de cette infrastructure. Ils permettent aux services de communiquer en utilisant Internet comme un moyen de communication et des standards ouverts sur Internet.

Du point de vue informatique, (Haas et Brown, 2004) de la W3C (World Wide Web Consortium) ont défini le service Web comme étant un système logiciel permettant de

supporter les interactions entre des machines interopérables sur un réseau. Un service Web possède une interface décrite avec un langage connu (WSDL). Les autres systèmes qui désirent interagir avec un service Web utilisent les messages SOAP qui sont généralement transmis en utilisant le protocole HTTP et doivent respecter sa description.

1.2.2.1 Protocoles utilisés

- **WSDL** : Le protocole universel de langage de description des services Web (WSDL pour « *Web Service Description Language* ») est un fichier permettant de décrire le service et la manière d'interagir avec lui (Christensen *et al.*, 2001). La communication entre plusieurs services se fait par échange de messages sous une façon structurée en utilisant le langage XML (Parveen et Tilley, 2008).
- **SOAP** : Le protocole d'accès aux objets simple (SOAP pour « *Simple Object Access Protocol* ») est le standard de base utilisé pour les services Web. Il permet de construire des requêtes et des réponses entre différents services Web indépendamment de la structure du réseau (Parveen et Tilley, 2008) (Gudgin *et al.*, 2007). La communication entre le client et le service est assurée via le protocole SOAP (Iheb, 2011).
- **UDDI** : Le protocole universel de découverte et de description (UDDI pour « *Universal Description, Discovery, and Integration* ») est un répertoire permettant d'enregistrer les descriptions des services créés par le fournisseur afin qu'ils puissent être découverts par les clients (utilisateurs de services). Il permet de décrire les services avec un langage compréhensible par l'humain et d'autres programmes en se basant sur : le langage de formalisation XML pour la description des données, le protocole HTTP pour le transfert et DNS pour la recherche d'autres services et application (Parveen et Tilley, 2008).

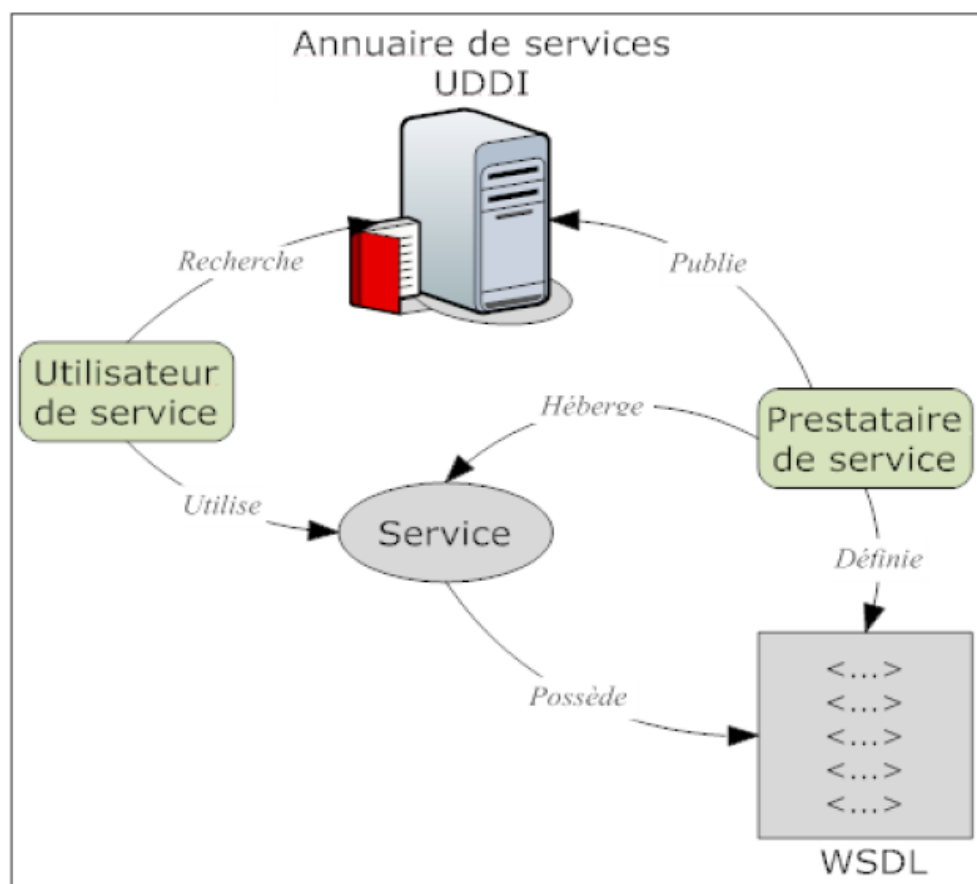


Figure 1.13 – Scénario d'utilisation des services Web.
Tirée de (Iheb, 2011)

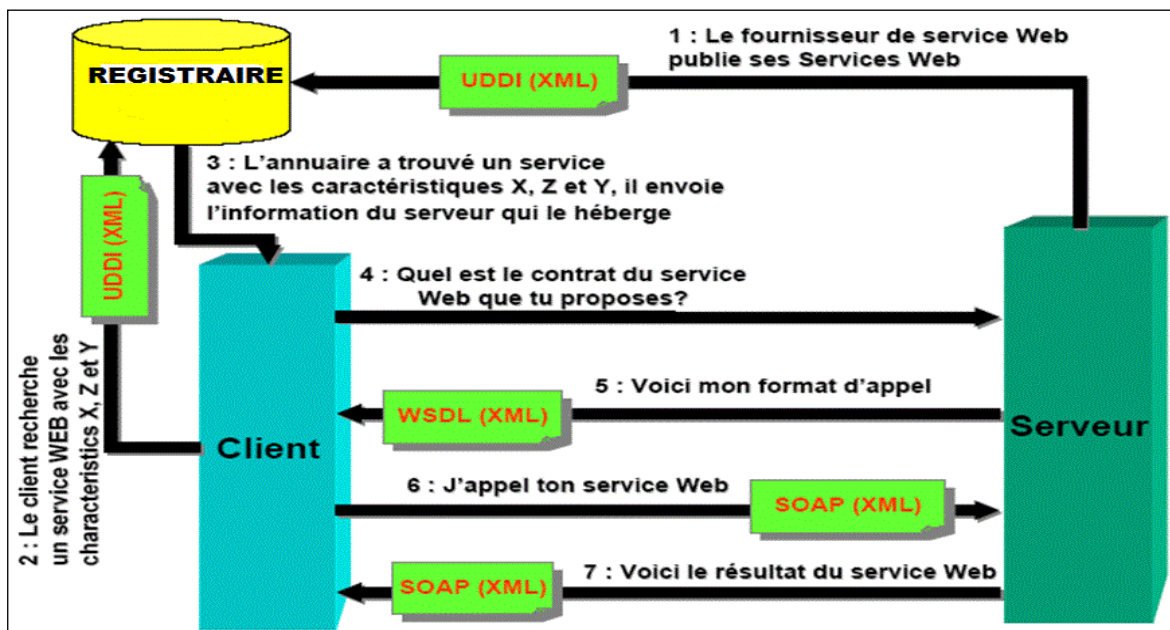
La Figure 1.13 schématise un scénario à base des services Web dans une architecture orientée services.

1. Le fournisseur commence par créer son service Web qui est composé d'un contrat rédigé avec le langage WSDL (Web Services Description Language) décrivant les fonctionnalités offertes par ce service et une logique métier « Business Logic ».
2. Le fournisseur publie son service dans un registre en utilisant le protocole standard le plus utilisé dans la découverte des services appelé UDDI (Universal Description

Discovery, and Integration) afin que ce service puisse être recherché et consulté par les clients (utilisateurs du service) en parcourant le registre de services.

- Une fois que le client aura trouvé la description du service qui lui convient, il obtient son contrat lui décrivant la manière avec laquelle il va pouvoir l'utiliser.

La Figure 1.14 donne plus de détails concernant le scénario basé sur les services Web en



apportant plus de précision dans chacune des sept (7) étapes suivantes.

Figure 1.14 – Principe de fonctionnement d’une architecture basée sur les services Web.

Tirée de (Cottin et Agour, 2007, p. 11)

Étape 1. Le fournisseur de services Web qui correspond au « Serveur » dans la Figure 1.13 crée et publie son service Web dans un registre de services UDDI en décrivant ses caractéristiques dans un fichier WSDL en XML à travers des messages SOAP ;

Étape 2. Le client (le consommateur de services Web) formule des requêtes (messages SOAP) et les met dans un fichier WSDL en XML afin de les envoyer au registre de services en spécifiant ses critères de recherches ;

Étape 3. le registre recherche les services correspondant aux critères spécifiés par le client et lui retourne toutes les URL des services disponibles et en décrivant la manière d'accéder à ces services et leur emplacement dans le réseau ;

Étape 4. Le client (demandeur de services) se relie au fournisseur, ce qui lui permettrait d'invoquer et exécuter le service ;

Étape 5. Le fournisseur de services envoie au client le contrat du service ;

Étape 6. Le client invoque le service Web en spécifiant son URL ;

Étape 7. Le fournisseur de service lui retourne les résultats souhaités.

1.3 Pourquoi combiner les deux architectures ?

(Taleb, 2008; Taleb *et al.*, 2010b) ont mentionné l'intérêt de combiner les patrons et les modèles, l'intérêt de segmenter l'architecture afin de réduire le couplage avec les spécifications très sévères ainsi que la gestion des environnements hétérogènes. Les limites actuelles de POMA (Taleb, 2008) incluent :

- Les patrons ne fournissent pas les dispositions suffisantes leur permettant de prendre en charge la spécification indépendante des plateformes des interfaces, les formulaires spécifiques des plateformes de ces interfaces, ni l'implémentation éventuelle de ces interfaces ;
- POMA n'encourage pas les concepteurs à considérer d'autres aspects de dialogue (ex. : les fonctions d'aides ou la gestion des erreurs), ce qui est essentiel aux utilisateurs ;
- Les patrons souffrent de grandes faiblesses au niveau des langages de programmation ;

- La recherche et l'application des patrons architecturaux appropriés en pratiques reste encore largement un processus ad hoc et non systématique, c'est-à-dire, il y'a un manque de consensus dans la communauté avec respect à la « philosophie » et la granularité des patrons architecturaux et un manque d'un langage mature des patrons ;
- Le besoin d'opportunités éducatives abordables qui focalisent sur les directives de style et de patrons pour la conception et l'utilisation de métalangage pour la spécification des interfaces (ISML pour « Interface Specification Meta-Language ») ;
- L'utilisation limitée de la norme ISO / CEI 9126-1 (2001) et d'autres attributs de qualité, tels que la communicabilité, la facilité d'apprentissage, de la maintenabilité et de l'utilisabilité.

De plus l'attrait de la réutilisation a incité de nombreuses entreprises à évaluer et à adopter la SOA comme un déclencheur pour de larges initiatives de transformation ou comme un catalyseur de projets de développement ou d'intégration de systèmes (A Arsanjani *et al.*, 2008). D'après les expériences de (A Arsanjani *et al.*, 2008) sur les projets et les dirigeants du secteur des affaires et des TI, le principal objectif est d'assurer la plus grande flexibilité et agilité en réponse aux demandes de changements constants des besoins de différents types d'utilisateurs, tels que les experts, les professionnels, les concepteurs, etc.

Cependant, pour remédier aux faiblesses de l'architecture POMA, il est nécessaire et pertinent de considérer l'architecture orientée services dans le cadre du développement des systèmes interactifs. De ce fait, la combinant de la SOA avec POMA est essentiel, afin de bénéficier de tous les avantages qu'offre l'orientation services plutôt que d'acheter de nouvelles solutions ou de perdre du temps précieux et de l'argent avec le codage et l'intégration.

L'intégration de la notion de services à l'architecture POMA permet une meilleure flexibilité entre les systèmes informatiques et le métier en représentant les applications en tant que services avec des interfaces standardisées. Ceci facilite le transfert et l'échange de tous types d'informations, ce qui la rend plus facile à exposer en externe. La flexibilité mène à une

meilleure visibilité des fonctionnalités que peut avoir un système interactif (Ganci *et al.*, 2006).

Les développeurs et les concepteurs ne rencontrent pas de grandes difficultés lors de la création d'un simple système interactif qui n'est pas sujet à de fréquents changements. Mais lorsqu'il s'agit d'un gros système interactif contenant de gros processus d'affaires, le changement des grands programmes monolithiques devient souvent un fardeau (TheOpenGroup, 2013). L'architecture POMA orientée services (SOPOMA) facilite ce changement et permet aux entreprises de s'aligner avec les nouvelles technologies et de profiter des opportunités d'affaires. Lorsque ce changement survient, il est important de comprendre les vrais besoins. Le Tableau 1.1 présente un résumé des besoins d'affaires qui représentent le moteur du changement dans les TI en allant de l'orientée fonction vers l'orienté processus et services afin de parvenir à la flexibilité et l'agilité (Ganci *et al.*, 2006).

Tableau 1.1 – Changements dans les TI dirigés par les affaires.
Tirée de (Ganci *et al.*, 2006).

De l'orientée fonction	Vers l'orienté services et processus
Crée pour les performances	Crée pour les changements
Un long cycle de développement	Cycle de développement incrémental
Applications en silos	Orchestration de solutions qui collaborent
Fortement couplées	Faiblement couplées
Utilisation des composants et des objets	Utilisation des services
Implémentation connue	Abstraction de l'implémentation

La SOA est le meilleur choix pour l'intégrer avec POMA, car elle basée sur des standards et des plateformes ouvertes. La Figure 1.15 résume les différentes plateformes ouvertes et standards de la SOA qui peuvent être utilisées par POMA.

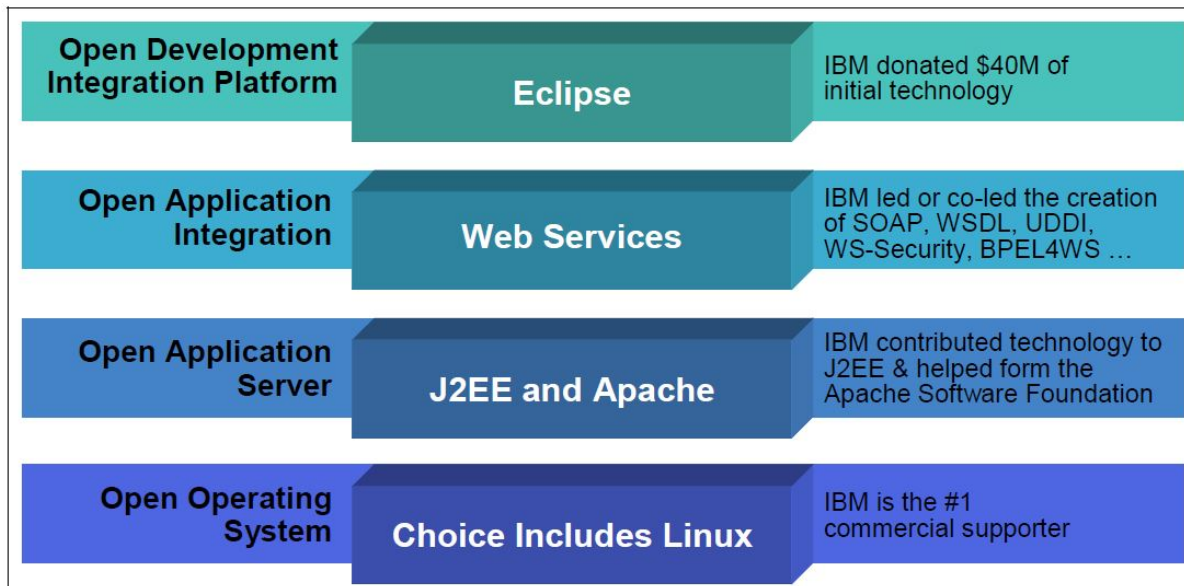


Figure 1.15 – Résumé des standards ouverts et des plateformes de SOA.
 Tirée de (Ganci *et al.*, 2006)

IBM est le leader des technologies, des produits et des solutions basées sur la SOA et le partenaire le plus important qui peut aider à définir les spécifications et les technologies utilisées pour implémenter une architecture orientée services, tels que les services Web, Service Component Architecture (SCA) et le Service Data Objects (SDO) (Ganci *et al.*, 2006).

La SOA utilise les meilleures pratiques pour délivrer des résultats particuliers, ces derniers permettront le développement des systèmes interactifs avec moins de problèmes et d'être déployés plus rapidement, par exemple (Ganci *et al.*, 2006):

- Le développement dirigé par les modèles permet de générer des artefacts logiciels à partir d'autres artefacts. Un modèle représente une description du système d'un point de vue particulier qui permet de cacher les détails non pertinents afin de rendre les caractéristiques d'intérêt plus claires.
- L'utilisation des patrons comme point d'entrée est une forme spécifique pour la capture et la réutilisation des éléments de conception.

L'architecture orientée services permet d'éviter les silos d'informations qui sont très difficiles à échanger et réduit les coûts de développement et d'intégration qui sont généralement très élevés.

L'architecture POMA orientée services (SOPOMA) héritera tous les avantages de la SOA comme indiqué précédemment. Par exemple : La réutilisation des patrons et des modèles plutôt que d'en créer de nouveau. La messagerie permet aux patrons d'invoquer directement d'autres patrons à travers les échanges de messages. La fonctionnalité de découverte des patrons offre une meilleure performance, car la sélection se fait par l'utilisateur en fonction de ses propres critères.

1.4 Sommaire du chapitre

Ce chapitre a donné un état de l'art sur les travaux qui ont été réalisés concernant l'architecture POMA (Pattern-Oriented and Model-Driven Architecture) et l'architecture orientée services communément appelée la SOA (Service-Oriented Architecture) et également leurs caractéristiques et critères pertinents sur ce projet (Tableau 1.2).

Les caractéristiques et les critères d'évaluation de ces architectures citées ci-dessus sont : le niveau architectural, le modèle, le concept éprouvé, la composition, l'intégration des services, la transformation, la conversion « Mapping », la génération, le couplage, la réutilisation, l'interopérabilité, l'échange de messages. Le (Tableau 1.2) récapitule l'évaluation des architectures présentées dans le CHAPITRE 1 selon les caractéristiques et les critères établis pour définir la problématique du projet.

Tableau 1.2 – Évaluation des deux architectures et leurs caractéristiques.

		Architectures		
		POMA	SOA	Patrons (n'est pas une architecture)
Caractéristiques	Modèle	Pleinement	Non	Non
	Niveau architectural	Oui	Non	Non
	Concept éprouvé	Partiellement	Oui	Oui
	Composition	Pleinement	Pleinement	Pleinement
	Transformation	Oui	Non	Non
	Conversion «Mapping»	Oui	Non	Oui
	Génération	Oui	Non	Non
	Couplage	Partiellement	Oui	Non
	Réutilisation	Partiellement	Pleinement	Non
	Interopérabilité	Pleinement	Partiellement	Oui
	Échange de messages	Non	Pleinement	Non
	Service	Non	Pleinement	Non
	Service Web	Non	Pleinement	Non
	Intégration	Non	Pleinement	Non

Légende :

- **Oui** : signifie que ce critère est pris en considération
- **Non** : signifie que ce critère n'est pas pris en considération
- **Partiellement** : signifie que ce critère est partiellement pris en considération
- **Pleinement** : signifie que ce critère est pleinement pris en considération et représente un élément fondamental

Dans le CHAPITRE 2, la problématique du projet est introduite et mène à la proposition du projet, le but et les objectifs du projet incluant les étapes du projet, les limites du projet et la méthodologie du projet utilisée.

CHAPITRE 2

PROBLÉMATIQUE DU PROJET

Il est nécessaire de choisir une méthodologie adaptée à ce projet. La première section de ce chapitre présente le but et les objectifs du projet. La deuxième section introduit les limites du projet. La troisième section décrit la méthodologie utilisée pour effectuer la conception de la nouvelle architecture de développement des systèmes interactifs.

L'architecture POMA présentée au CHAPITRE 1 n'utilise pas la notion de services et, conséquemment, ne profite pas des avantages qu'offre la SOA. Par exemple :

- La facilitation de développement des systèmes interactifs par la combinaison des patrons, les modèles et l'utilisation des services ;
- La réutilisation des services offre un avantage majeur. Elle permet de réduire les coûts de développement et des tests en utilisant des modules logiciels existants au lieu d'en créer de nouveau ;
- La possibilité aux patrons et les services d'invoquer d'autres qui tournent sur un même processeur par des échanges de messages. La surcharge est contrôlée, car les services doivent être faiblement couplés. Ceci permet aux entreprises d'utiliser facilement des services externes pour remplacer ceux à l'interne en utilisant la messagerie. Ce contrôle de la surcharge permet également d'effectuer rapidement le changement et s'aligner aux besoins d'affaires. L'analyse des flux de ces messages permet d'obtenir des informations précieuses sur les activités de l'entreprise, et permet de contrôler les politiques de gestion et de sécurité mises en place ;
- Une meilleure communication entre différentes plateformes grâce à la messagerie (tel que les ordinateurs de bureaux, les Smartphones, etc.) par la conversion du format de données ;

- La découverte permet aux services et aux patrons d'être transparents lors de leur création, mais facilement découverts lors de l'exécution. D'autres systèmes appellent ses services et patrons d'après leurs fonctionnalités et leurs coûts, ce qui permet une augmentation considérable au niveau des performances des systèmes interactifs ;
- Parmi les fonctionnalités très intéressantes qu'offre la SOA, la création des façades aux applications précédentes afin de les rendre accessibles. Par exemple : un système de gestion de bases de données peut fournir un service de gestion de bases de données. L'avantage pour les développeurs et aux entreprises c'est qu'ils gardent leurs anciens systèmes et applications, aucun frais supplémentaire ne sera nécessaire pour le développement et le remplacement du système existant.

2.1 But et objectifs du projet

Le but de ce projet est le suivant :

« Définir une nouvelle architecture pour faciliter le développement et la migration des systèmes interactifs tout en améliorant leur utilisabilité et leur qualité en général ».

Afin d'atteindre le but du projet, l'objectif est de définir une nouvelle architecture POMA orientée services, appelée **SOPOMA (Service-Oriented POMA)** en se basant sur l'architectures POMA et la SOA. Et ce, en combinant les deux architectures en rajoutant une couche de services à l'architecture POMA afin de lui permettre d'hériter tous les avantages de la SOA. Les autres objectifs tels que la migration, l'utilisabilité et la qualité ne sont pas inclus dans ce projet.

2.2 Limites du projet

Les points suivants délimitent la portée de ce projet :

- Ce travail se limite essentiellement à proposer conceptuellement parlant une nouvelle architecture de POMA orientée services ;
- Ce travail ne couvrira pas la réalisation de cette nouvelle architecture proposée incluant la génération du code source pour les systèmes interactifs.

2.3 Méthodologie du projet

La Figure 2.1 décrit les différentes étapes de la méthodologie du projet conçues pour atteindre les objectifs comme suit :

- 1) Élaborer une revue de littérature sur l'architecture POMA et l'architecture orientée services (SOA) (détails dans le CHAPITRE 1) ;
- 2) Définir la problématique du projet incluant ses objectifs (détails dans le CHAPITRE 2) ;
- 3) Proposer la nouvelle architecture SOPOMA (détails dans le CHAPITRE 3) ;
- 4) Concevoir un cas exploratoire sur l'architecture SOPOMA (détails dans le CHAPITRE 4).

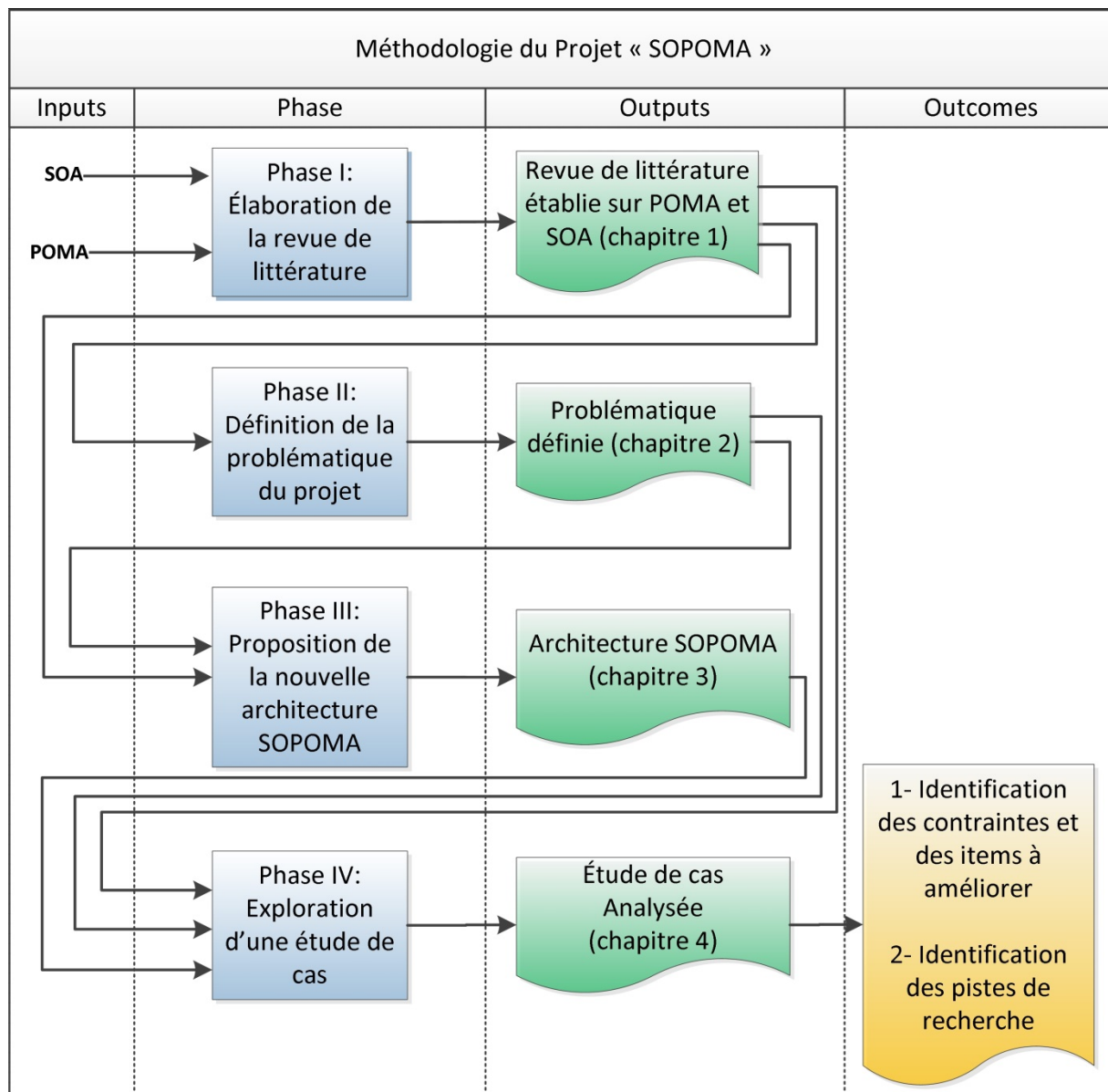


Figure 2.1 – Méthodologie du projet.

2.4 Résumé du chapitre

La nouvelle architecture proposée orientée patron, dirigée par des modèles et orientée services (SOPOMA) est une combinaison de l'architecture POMA et de l'architecture SOA.

Le but est de combiner et d'intégrer tous les concepts des deux architectures précédentes afin de permettre de :

- Faciliter le développement des systèmes interactifs en intégrant et en utilisant les trois approches : (a) orientée services, (b) orientée patron et (c) dirigée par des modèles ;
- Favoriser la communication et la réutilisation de l'expertise individuelle concernant les bonnes pratiques de conception.

Atteindre rapidement les objectifs en répondant aux réels besoins des concepteurs, développeurs et experts. Le chapitre 3 présente et décrit les principes et concepts clés, une vue d'ensemble, les justifications et les spécifications de l'architecture orientée patron et dirigée par des modèles, et orientée services (SOPOMA pour Service-Oriented and Pattern-Oriented et Model-Driven Architecture).

CHAPITRE 3

ARCHITECTURE ORIENTÉE PATRON, DIRIGÉE PAR DES MODÈLES ET ORIENTÉE SERVICES

Ce chapitre présente l'architecture proposée SOPOMA pour le développement des applications interactives selon la méthodologie décrite dans le chapitre précédent. Cette architecture combine l'architecture POMA et l'architecture SOA. La première section décrit les concepts clés de POMA. La deuxième section décrit une vue d'ensemble de l'architecture SOPOMA. La dernière section fournit les justifications de SOPOMA.

3.1 Concepts clés de SOPOMA

Quatre concepts clés sont utilisés dans SOPOMA (Figure 3.1), à savoir :

- Les concepts utilisés par POMA (six niveaux d'architecture et catégories de patrons, deux modèles PIM et PSM, la composition des patrons, la conversion (« Mapping ») des patrons, la transformation des modèles, la génération du code source) ;
- Le concept de service utilisé dans la SOA (vus dans la section 1.2.1.1) ;
- Le concept de l'ESB ;
- Le concept des services Web.

3.2 Vue d'ensemble de SOPOMA

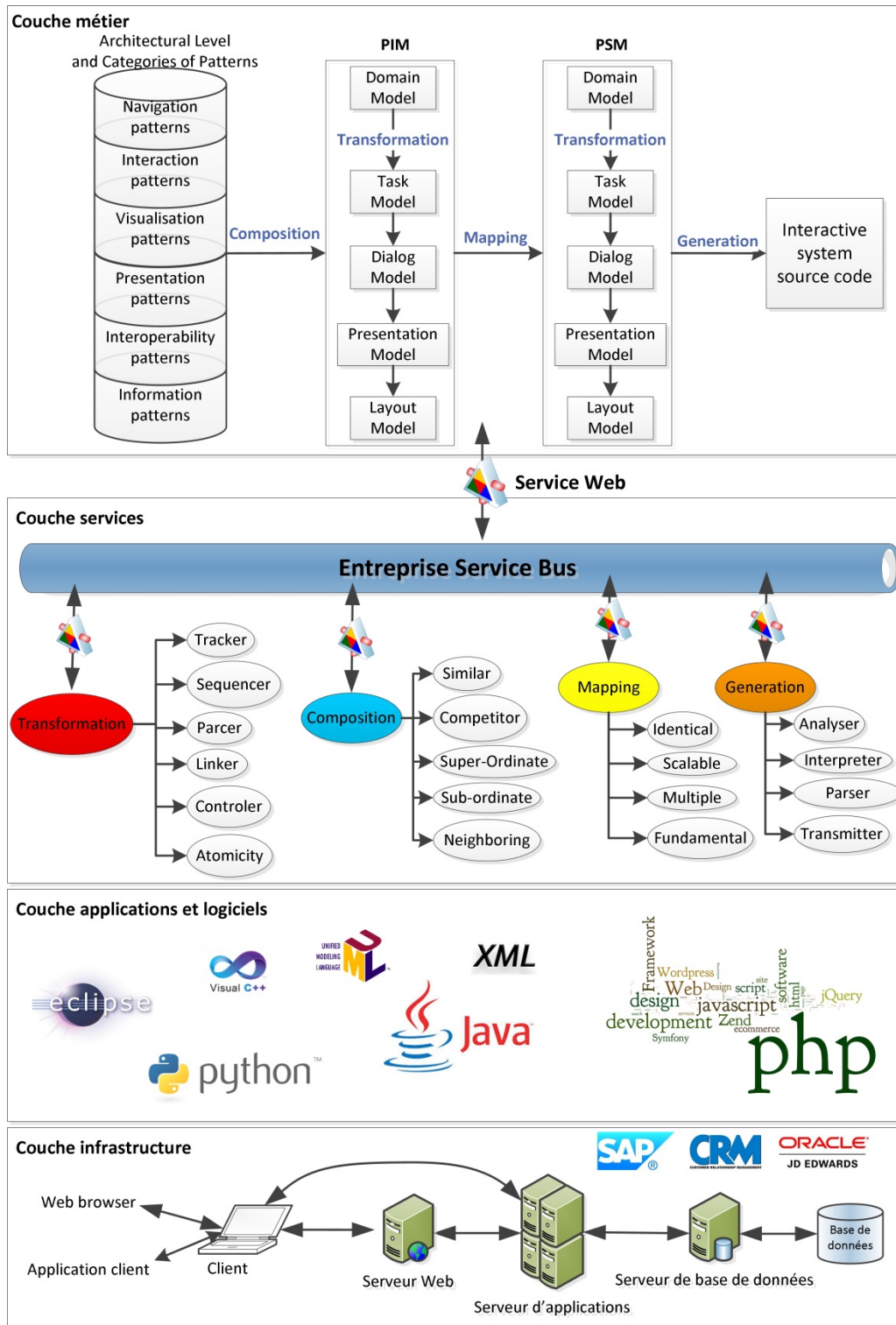


Figure 3.1 – Architecture SOPOMA.

Bien que les concepts vus dans la section précédente (section 3.1) soient des concepts fondamentaux pour l'implémentation de l'architecture SOPOMA, ils ne représentent que la partie théorique. La Figure 3.1 présente la nouvelle architecture SOPOMA pour le développement des systèmes interactifs qui combine les concepts de la SOA et ceux de POMA, comme suit :

L'architecture SOPOMA compte quatre (4) couches, à savoir :

1. **Couche métier** : elle regroupe les connaissances, les données et les pratiques permettant de créer l'architecture POMA. Cette couche est constituée de six (6) niveaux architecturaux et catégories de patrons. La composition de ces patrons permet de générer cinq (5) types de modèles PIM qui seront transformés en cinq (5) types de modèles PSM. L'application des règles de génération de code permet de créer le code source de toute l'application. Cette couche interagit essentiellement avec les services Web par l'envoi de message par la couche métier. Les services Web, à leur tour, communiquent avec l'intergiciel (« Middleware ») appelé le bus des services d'entreprise (ESB pour (Entreprise Service Bus). Cette communication s'effectue avec le protocole HTTP par exemple et le format de données XML pour l'échange des messages.
2. **Couche service** : Cette couche regroupe tous les services définis dans l'architecture SOPOMA. Elle communique via l'intergiciel ESB.

Quatre (4) familles de services existent dans SOPOMA :

- a) **Service de composition** : ce service permet de créer un modèle PIM à partir de la composition de plusieurs patrons en se basant sur les services ci-dessous :

- **Service Similar** : permet de sélectionner des patrons similaires de la même catégorie. Comme illustré dans la Figure 3.2 le service Similar va sélectionner les patrons « Convenient Toolbar » et « Browsing Index » qui offrent le même service de navigation dans le contexte des systèmes interactifs

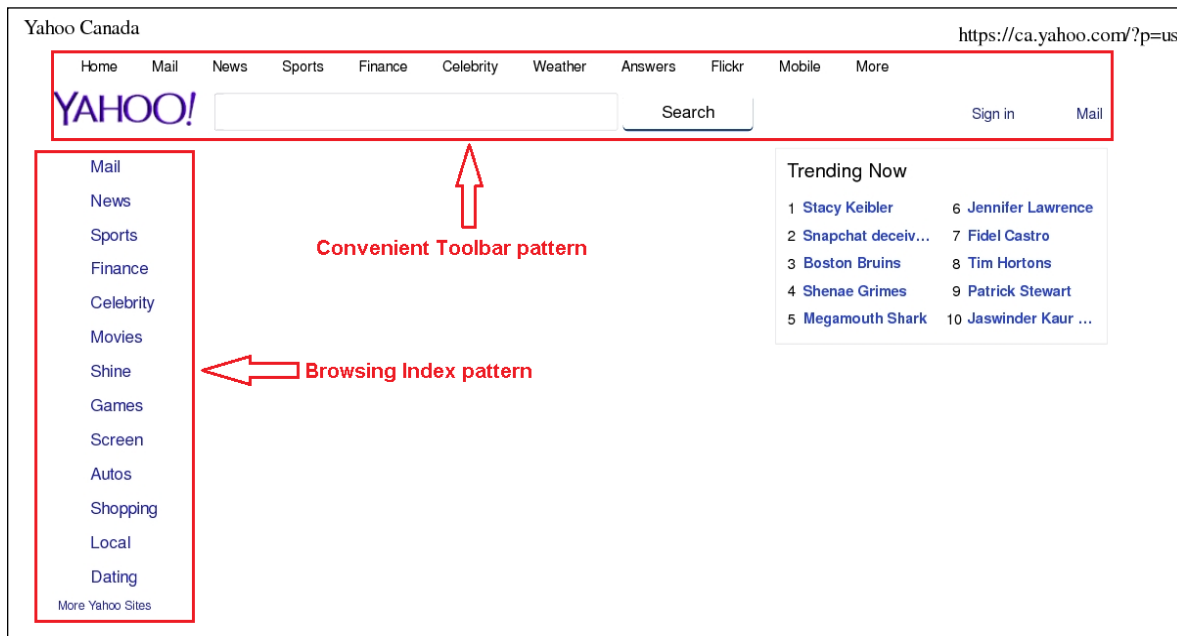


Figure 3.2 – Exemple de patrons similaire.
Tirée du (site Web de Yahoo Canada)

- **Service Competitor** : permet de sélectionner un patron parmi plusieurs qui sont de la même catégorie, mais qui ne peuvent pas s'appliquer en même temps. Comme illustré dans la Figure 3.2, le service Competitor va proposer les patrons « Index Browsing » et « Convenient Toolbar » qui fournissent la même solution, mais le patron « Index Browsing » prend beaucoup plus d'espaces que le patron « Convenient Toolbar » ce qui n'est pas adéquat pour certaines plateformes telles que les PDA et les smartphones. Toutefois le patron « Convenient Toolbar », qui fournit la même solution, est généralement considéré beaucoup plus approprié ;

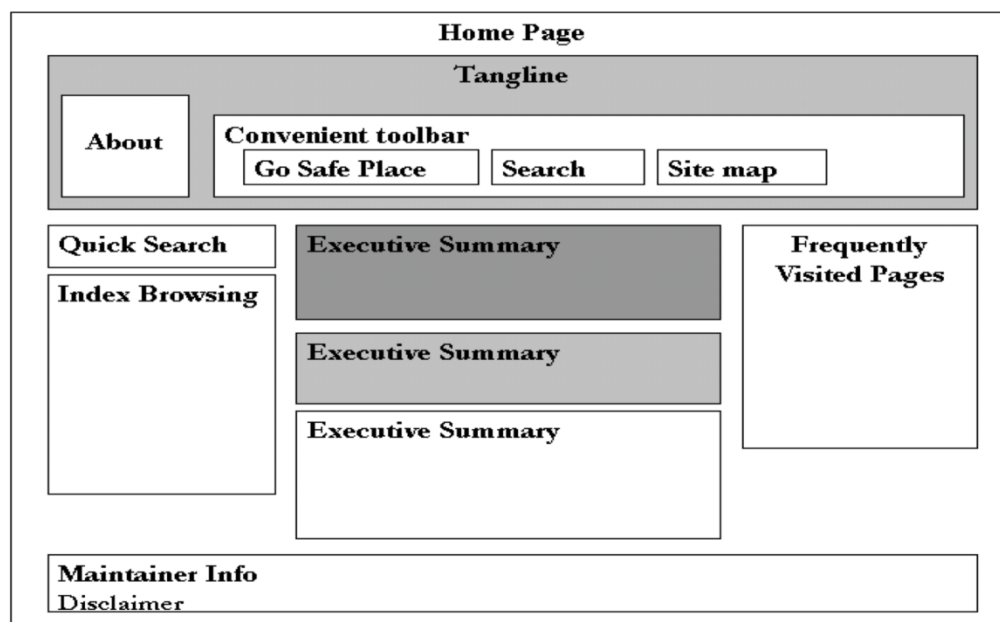


Figure 3.3 – Conception d’une page d’accueil en utilisant les patrons.
Tirée de (Taleb *et al.*, 2006)

- **Service Super-ordinate** : permet à plusieurs patrons de différentes catégories de se combiner. Le patron ‘A’ est considéré Super-ordinate à ‘B’ si et seulement s’il contient le patron ‘B’. Par exemple : le patron « Tangline » de la Figure 3.3 est super-ordonné du patron « Convenient toolbar ». Ce dernier est également super-ordonné des patrons « Go Safe Place », « Search » et « Site map » ;
- **Service Sub-ordinate** : ce service comme le précédent permet de composer différents patrons. Le service ‘B’ est Sub-ordinate à ‘A’, car il est utilisé par le patron ‘A’. Par exemple les patrons « Go Safe Place », « Search » et « Site Map » sont subordonnés du patron « Convenient toolbar » ;
- **Service Neighboring** : ce service permet de sélectionner un patron voisin à un autre service appartenant à la même catégorie. Par exemple les patrons « Search » et « Login » dans la Figure 3.2 sont voisins, car il appartient à la même catégorie de patrons qui est « Convenient toolbar » ;

Par exemple les patrons « Go Safe Place », « Search » et « Site Map » dans la Figure 3.3 sont voisins, car il appartient à la même catégorie de patrons qui est « Convenient toolbar ».

- b) **Service de conversion « Mapping »** : Ce service permet de reconcevoir les applications des systèmes interactifs pour d'autres plateformes, en effectuant la conversion « Mapping » du modèle indépendamment des plateformes (PIM pour Platform Independent Model) au modèle spécifique des plateformes (PSM pour Platform Specific Model), tels que le passage de la plateforme des ordinateurs traditionnels en passant par des tablettes vers finalement la plateforme des téléphones intelligents (Smartphones) et/ou PDAs pour « Personal Digital Assistant » Figure 3.4.

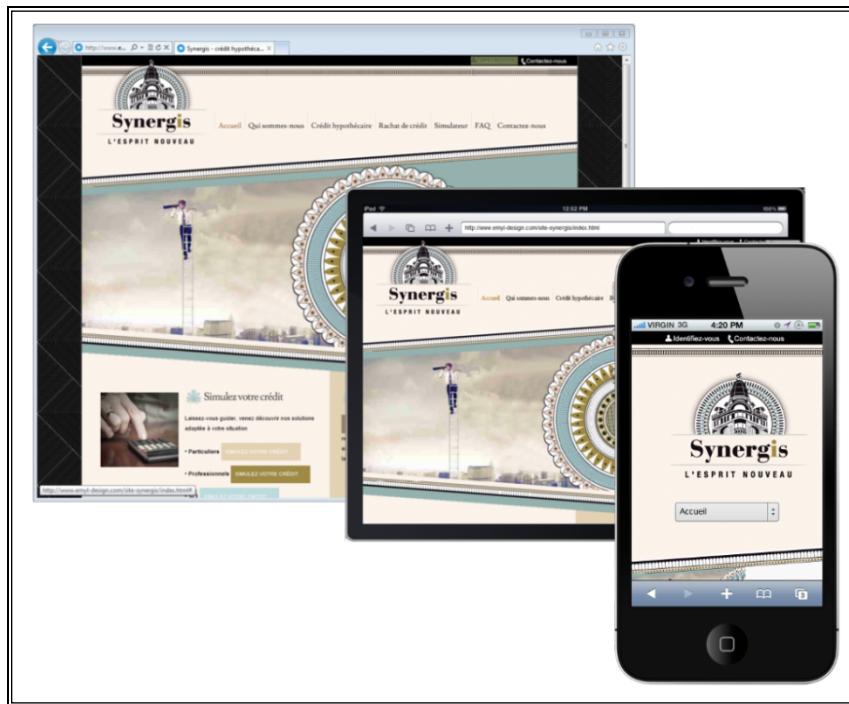


Figure 3.4 – Apparence d'un site pour différentes plateformes. Tirée de (Olvani) .

- **Service « Identical »** permet de reproduire le même type de patrons sur une plateforme différente sans effectuer aucune transformation. Comme le montre la Figure 3.4, le site Web reste le même sur la plateforme PC et sur la tablette ;

- *Service « Scalable »* permet de redimensionner la conception originale de telle sorte qu'elle s'adapte à la nouvelle plateforme ;
 - *Service « Multiple »* permet de reproduire la conception originale plusieurs fois sur la même conception. Par exemple, un site peut contenir plusieurs petits menus dans la même page, au lieu d'un menu général ;
 - *Service « Fundamental »* permet de remplacer la conception originale d'un système qui n'est plus utilisable par autre conception qui s'adapte à la nouvelle plateforme. Comme montré dans la Figure 3.4 la conception du même site Web a changée dans le téléphone intelligent (Smartphone).
- c) **Service de transformation** : ce service permet de convertir les modèles sources PIM en modèle plus spécifique. Six (6) types de services de transformation sont présentés ci-dessous.

La Figure 3.5, la Figure 3.6 et la Figure 3.7 illustrent un exemple de transformation d'un modèle vers un autre (PIM→PIM et/ou PSM→PSM). Cet exemple est tiré d'une étude de cas sur un système interactif sur « la gestion environnementale » (Taleb *et al.*, 2010c). Cet exemple permet de montrer la façon dont les services de transformation peuvent être appliqués entre les modèles de domaine et modèles de tâches pour la plateforme de l'ordinateur portable (« Laptop »).

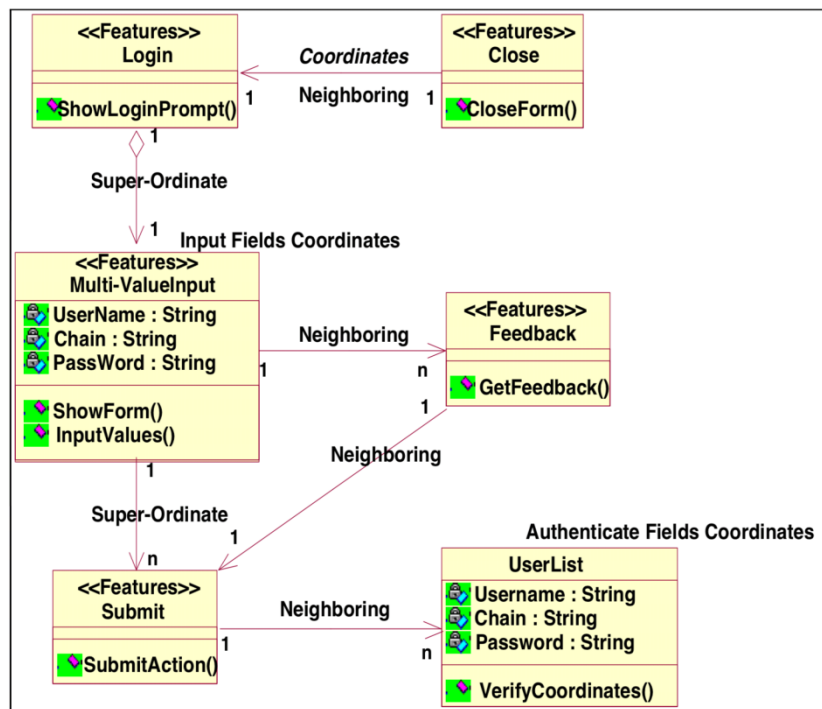


Figure 3.5 – Diagrammes de classe du modèle de domaine PIM.
Tirée de (Taleb *et al.*, 2010c)

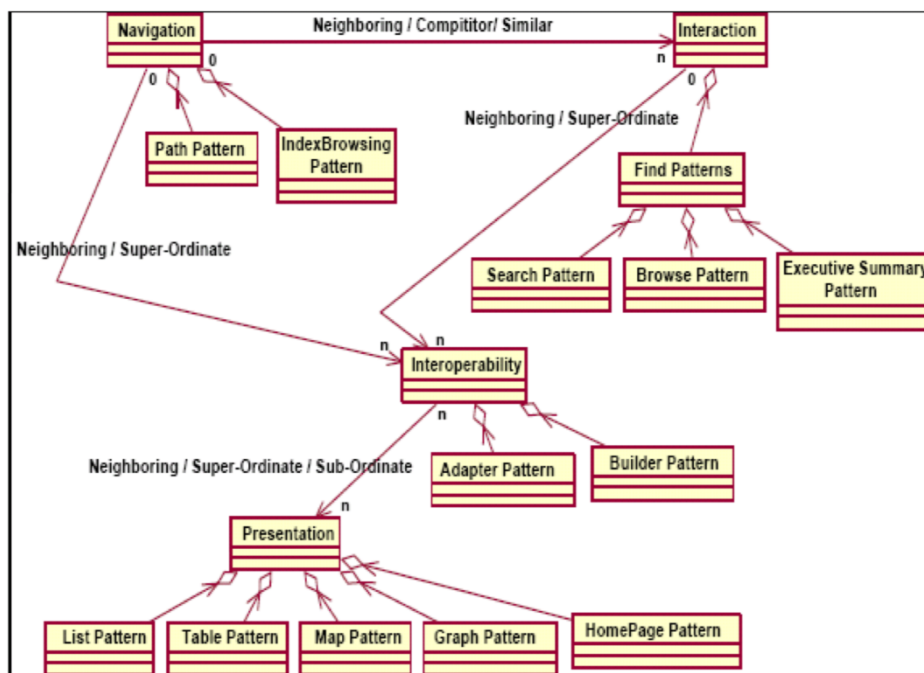


Figure 3.6 – Diagramme de classe du modèle de tâches PIM.
Tirée de (Taleb *et al.*, 2010c)

- ***Service Tracker*** : ce service permet de détecter les liens potentiels pour relier les différents modèles. Ce service exige les entrées (inputs) des objets. En d'autres mots, ce service trace les dépendances qu'il y a entre les objets. Par exemple, le service Tracker détecte la classe « Login » du modèle de domaine PIM (Figure 3.5) et la classe « Navigation » du modèle de tâches PIM (Figure 3.6) ;
- ***Service Sequencer*** : ce service génère des valeurs numériques qui constituent une liste de références pour tous les objets de chaque modèle pour renforcer la séquence des événements. Ces valeurs sont également utilisées par le service « Tracker » pour déterminer quels objets peuvent être liés à d'autres par le service « Linker ». Par exemple, le service Sequencer crée une liste de références et détermine le séquençement de tous les objets sélectionnés (Figure 3.7). Il attribue la référence (1) à la classe « Login » et la référence (2) à la classe « Navigation ». Le résultat sera envoyé au service Parser ;
- ***Service Parser*** : ce service vérifie, analyse la syntaxe et s'assure de l'existence des objets tels que la classe, classe associative, attributs, l'héritage, événement, l'état, transition, super-état, sous-état, service de classe, association, domaine, le domaine de service avant de les transmettre au service « Linker ». Par exemple, le service Parser s'assure de l'existence des deux classes « Login » et « Navigation » et que leur syntaxe sont bien conforme (Figure 3.7) ;
- ***Service Linker*** : ce service permet de créer des liens entre les différents modèles PIM et PSM en se basant sur les informations fournies par le service Parser et la liste des valeurs de références fournie par le service « Sequencer ». Par exemple, le service Linker permet d'établir les liens entre les classes sélectionnées « Login » et « Navigation » (Figure 3.7) ;

- *Service Semantic Modeller* : permet de contrôler et de vérifier si les sémantiques de la machine à états peuvent être supportées ;
- *Service Atomicity* : permet de contrôler les actions des modèles s'exécutant sans interruption et s'assure que les résultats obtenus sont cohérents et fiables. Autrement dit, le tout ou rien.

La Figure 3.7 représente le résultat de l'application des services de transformation sur le modèle de domaine PIM et le modèle de tâches PIM.

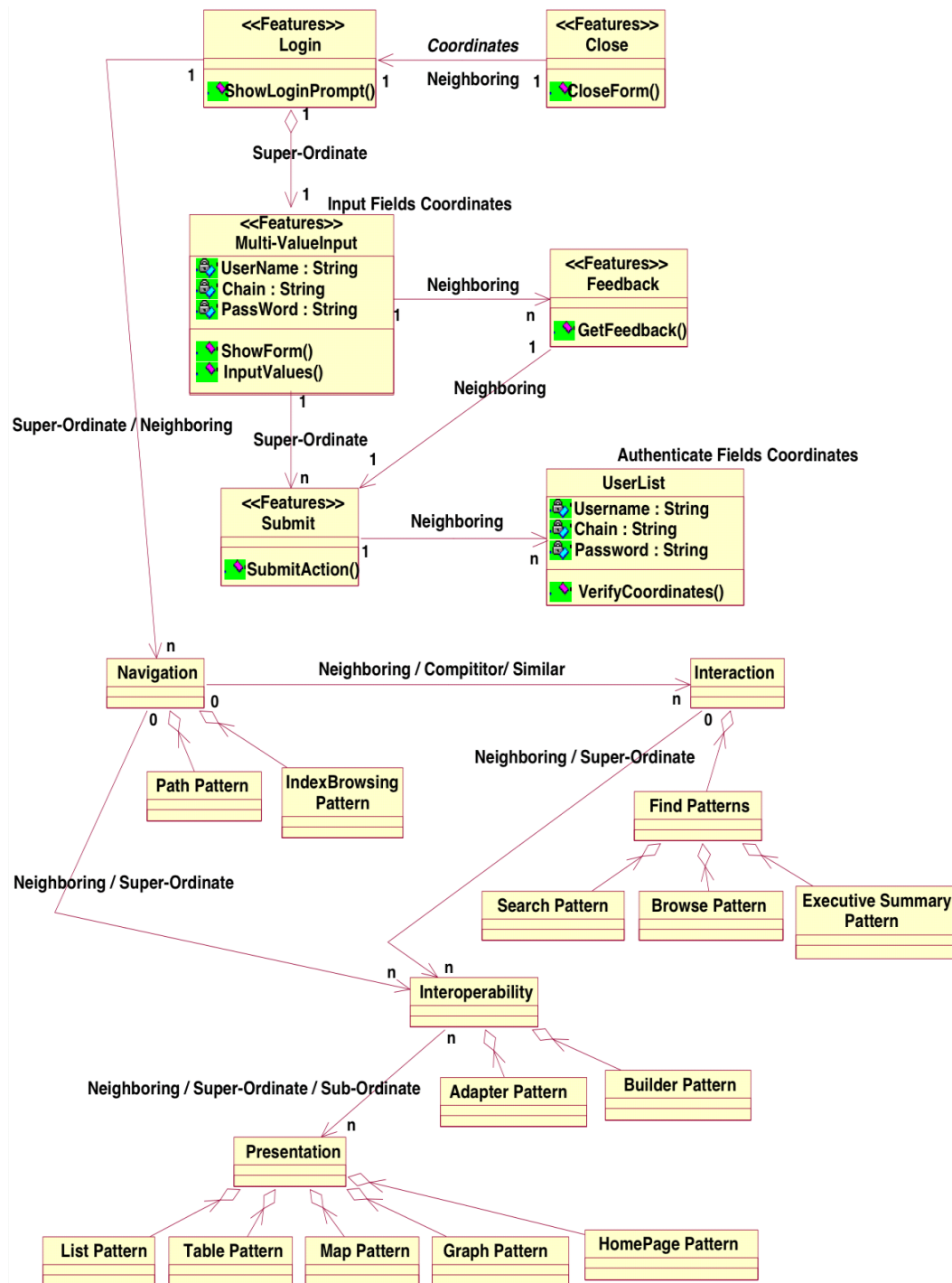


Figure 3.7 – Modèles de domaine PIM et modèle de tâches PIM unifiés pour la plateforme Laptop.
Tirée de (Taleb et al., 2010c)

d) Service de génération : ce service permet de générer le code source du système interactif en appliquant les services suivants :

- ***Service Analyser*** permet la découverte du modèle objets du système et les éventuelles erreurs qu'il contient et permet d'écarter toutes les erreurs et les fautes de conception ;
- ***Service Interpreter*** permet d'interpréter le modèle pré-analysé par le service Analyser et effectue les opérations de traduction de langage en code XML. Le résultat sera envoyé au service Parser ;
- ***Service Parser*** comme a été expliqué plus haut (dans « services de transformation »), le service Parser permet de vérifier et de compiler le code XML obtenu en effectuant le service Analyser et Interpreter et l'envoi au service Transmitter ;
- ***Service Transmitter*** permet de fournir le modèle objet PSM au générateur de code afin de convertir le code XML du modèle PSM en code source du système interactif dans un langage spécifique pour une certaine plateforme ;

3. Couche Application et logiciels :

Cette couche permet de répertorier et d'intégrer tous les outils et langages de développement des systèmes interactifs supportés par SOPOMA, tels que « Eclipse », UML (Unified Modeling Language), Rational Rose, Java. XML. Il est à noter que SOPOMA doit intégrer également d'autres systèmes les plus connus tels que CRM, JD Edwards, SAP.

4. Couche infrastructure :

Cette couche décrit l'infrastructure de l'architecture SOPOMA : elle correspond à une architecture N-tiers composée d'un serveur Web, serveur d'application, serveur de base

de données, la base de données et le client. Le modèle d'architecture N-tiers permet de développer des applications flexibles et réutilisables sur différentes plateformes.

- ***Serveur Web*** : un serveur Web peut être à la fois (1) un ordinateur et (2) un ensemble de logiciels. La première définition (ordinateur) décrit un ordinateur ou un ensemble d'ordinateurs pouvant contenir ou pas des écrans et un système de stockage. Tous ces ordinateurs sont connectés à Internet et permettent d'héberger des ressources (ex. : sites Web). La deuxième définition (ensemble de logiciels), aussi appelée serveur HTTP, permet de faire fonctionner le site Web et le rendre accessible par les internautes en assurant la communication entre le client utilisant un navigateur Web ou son application Web et le serveur Web. D'autres outils sont utilisés tels que le gestionnaire de base de données (ex : MySQL), le langage PHP pour la création, et la gestion des sites dynamiques.
- ***Serveur d'application*** : le serveur d'application permet de regrouper un ensemble d'applications dans un grand système. Il permet d'implémenter les opérations que le système doit effectuer sur les données en fonction de la demande du client. Le serveur d'application traite les demandes et renvoie les résultats au client via le serveur Web. L'avantage d'utiliser un serveur d'application est qu'il permet aux développeurs de se concentrer uniquement sur la logique métier (« Business Logic ») au lieu de perdre du temps sur des aspects comme le réseau, la sécurité, la gestion des transactions, etc., car tout cela sera géré par le serveur d'application.
- ***Serveur de base de données*** : ce serveur est un programme utilisé pour le stockage et la gestion des bases de données qui respecte l'architecture client-serveur. Ce logiciel représente la partie « Back-end » du serveur, il permet de stocker les données dans plusieurs bases de données plutôt que de tout regrouper dans une seule. Il permet également d'analyser et de combiner les données en respectant les requêtes du client (la partie Front-end).

La Figure 3.8 décrit un exemple de fonctionnement des services dans une architecture N-tiers en plusieurs étapes :

- 1) Le client désire utiliser un service. Il formule une requête avec tous les paramètres nécessaires et envoie la demande au serveur Web ;
- 2) Le serveur Web reçoit la requête, recherche le service souhaité dans le registre et renvoie son URL et son emplacement en lui indiquant qu'il doit solliciter le serveur d'application ;
- 3) Le client se connecte au serveur d'application et invoque le service désiré ;
- 4) Le serveur d'application reçoit la demande d'invocation du service et fait appel au serveur de base de données pour lui envoyer les données dont il a besoin pour l'exécution du service ;
- 5) Le serveur de base de données reçoit les données et les achemine au serveur d'applications ;
- 6) Le serveur d'application exécute le service et achemine les résultats au client.

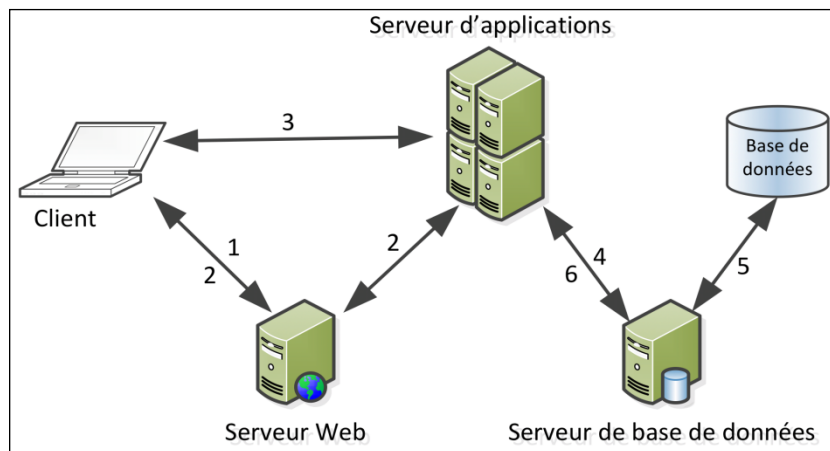


Figure 3.8 – Exemple d'une architecture N-tiers.

3.3 Justification de SOPOMA

Les justifications de SOPOMA sont les suivantes :

- Tous les concepts de l'architecture POMA tels que les niveaux architecturaux, la catégorisation des patrons, les techniques de composition, les types de modèles PIM et PSM, la transformation et la conversion « Mapping » sont hérités par SOPOMA.
- L'architecture orientée services (SOA) permet à l'architecture SOPOMA d'hériter tous ses concepts, tels que le service, le fournisseur de services, le demandeur de services, le registraire, les protocoles UDDI, WSDL, les principes de la réutilisation, couplage, composition, etc.
- La notion de services Web permet à SOPOMA d'hériter les protocoles SOAP de communication tels que HTTP, FTP, HTTPS, etc.
- L'ESB permet à SOPOMA d'hériter les concepts et les composants de l'ESB.

CHAPITRE 4

ÉTUDE DE CAS EXPLORATOIRE

Cette section présente une étude de cas sur la conception d'un prototype d'une interface utilisateur fonctionnel d'un système interactif de gestion de l'environnement. Ce cas est tiré de (Taleb *et al.*, 2009) (Taleb *et al.*, 2010c). Le but de cette étude de cas est de mettre en relief les avantages de la SOA dans POMA, et de décrire de quelle façon l'utilisation des services faciliteront la réalisation des différents modèles.

4.1 Services de composition

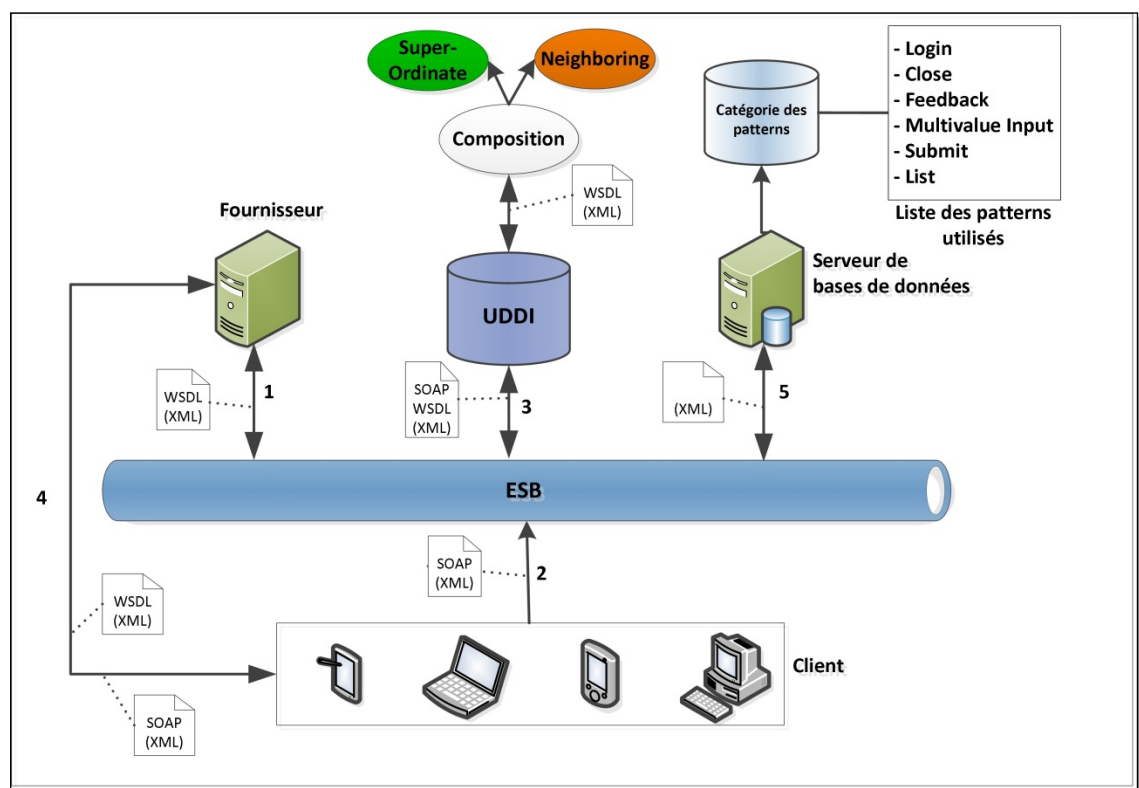


Figure 4.1 – Exemple de services de composition de patrons.

L'exemple de la Figure 4.1 traite l'utilisation des services de composition sur certains patrons, tels que « Close », « Login », « Feedback », « Multi-Value Input » et « Submit » afin de produire une page d'accueil d'un site Web.

Étape 1. Le fournisseur de services crée les services de composition, de transformation, de conversion « Mapping », et de génération. Il enregistre leur description dans des fichiers WSDL en XML et les publie dans le registre de services (registre UDDI). L'ESB reçoit les messages du fournisseur, les transforme afin qu'ils soient compatibles avec le récepteur et les achemine vers le registre « UDDI ».

Étape 2. Le demandeur de service (client) désire réaliser la composition des patrons pour produire un modèle PIM. Pour ce faire, il recherche les services qu'ils lui conviennent en interrogeant le registre des services en formulant une requête sous forme d'un message SOAP. Ce dernier contient les noms des méthodes appelées telles que le service Super-ordinate, le service Sub-ordinate et le service Neighboring, ainsi que les paramètres appliqués de ces méthodes.

Étape 3. Le registre utilise des technologies standards telles que : UDDI, SOAP, WSDL pour rechercher les services qui correspondent à la requête du client. Une fois le client aura trouvé la description des services qui lui convient, il télécharge leur fichier WSDL correspondant depuis le registre. Ce fichier contiendra l'URL vers le service Web et comment y accéder. Par exemple, la description des services concerne les services de composition (services Super-ordinate ou Sub-ordinate) qui fournissent le même résultat dans ce cas-ci, ainsi que le service Neighboring.

Étape 4. Une fois que le client aura reçu les fichiers WSDL, il pourra invoquer le service auprès du fournisseur en demandant son contrat de service. Le fournisseur lui retourne le contrat sous format WSDL. À ce moment-là, le client pourra exécuter le service ou lui demander d'exécuter certaines fonctionnalités. Le fournisseur de services enverra le résultat après exécution des services dans un fichier SOAP.

Étape 5. L'exécution des services sollicitera le serveur de base de données afin de lui envoyer l'information concernant les patrons utilisés. Le service Super-ordinateur recevra les données des patrons « Login », « Multi-Value Input » et « Submit ». Le service Neighboring recevra les données sur les patrons « Close » et « Feedback ».

Le résultat de la composition des patrons produira le modèle de domaine indépendant [SOPOMA.PIM] (Figure 4.2). Les couleurs des liens de la Figure 4.2 correspondent aux types de services de la Figure 4.1 utilisés.

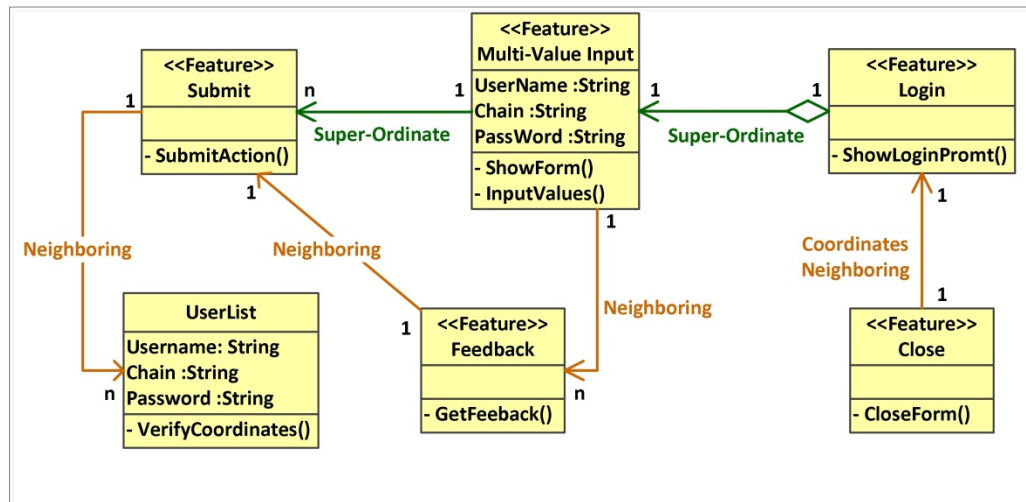


Figure 4.2 – Diagramme de classe du modèle de domaine PIM.
Tirée de (Taleb *et al.*, 2010c)

4.2 Services de conversion « mapping »

La Figure 4.3 est un exemple pour illustrer l'utilisation des services de conversion « Mapping » pour convertir (ou « mapper ») le modèle de domaine PIM vers un modèle de domaine PSM pour la plateforme PDA afin de voir le changement. Le résultat de la conversion « Mapping » est montré dans la Figure 4.4. Les flèches colorées dans la Figure 4.3 correspondent aux services utilisés.

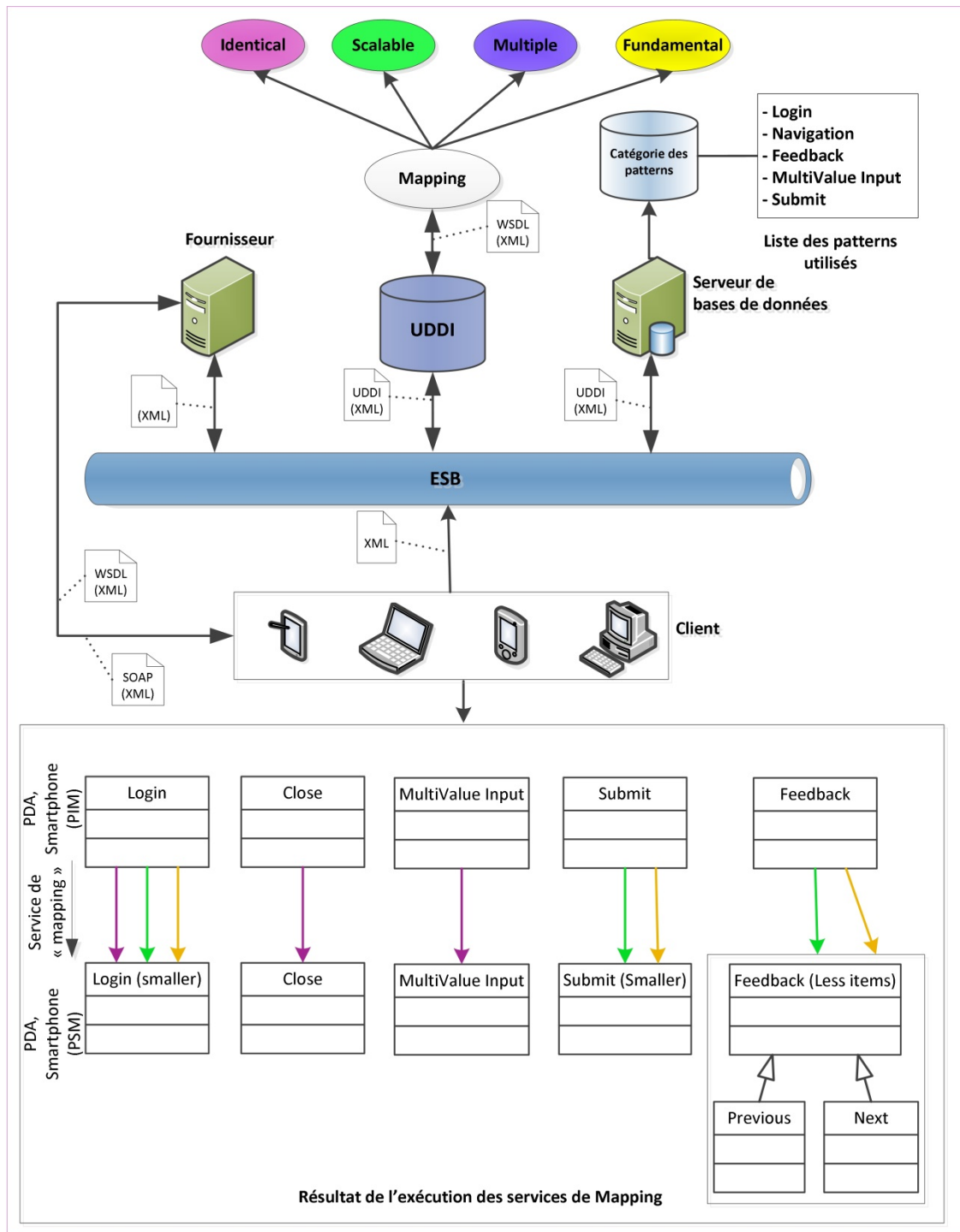
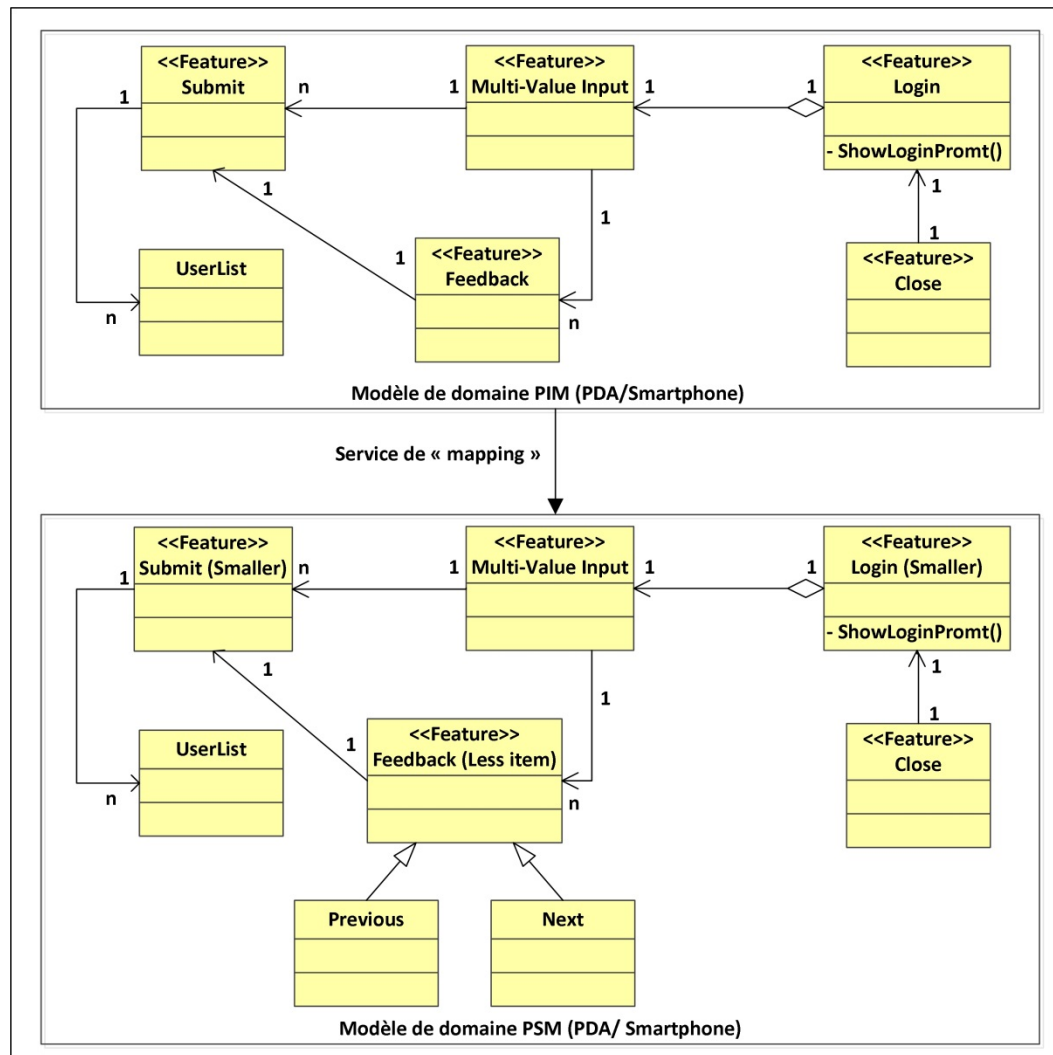


Figure 4.3 – Exemple d'utilisation des services de conversion « Mapping » du modèle de domaine PIM vers PSM pour PDA.



**Figure 4.4 – Résultat de la conversion « Mapping » du modèle de domaine PIM à PSM pour la plateforme PDA.
Tirée de (Taleb *et al.*, 2010c)**

4.3 Services de transformation

La Figure 4.5 résume les différentes étapes de transformation du modèle de domaine PIM et le modèle de tâche PIM afin d'établir un modèle unifié PIM, tel qu'expliqué dans les Figure 3.5, Figure 3.6 et Figure 3.7. Les couleurs dans la Figure 4.5 correspondent à l'exécution des services de transformation et leurs résultats correspondants.

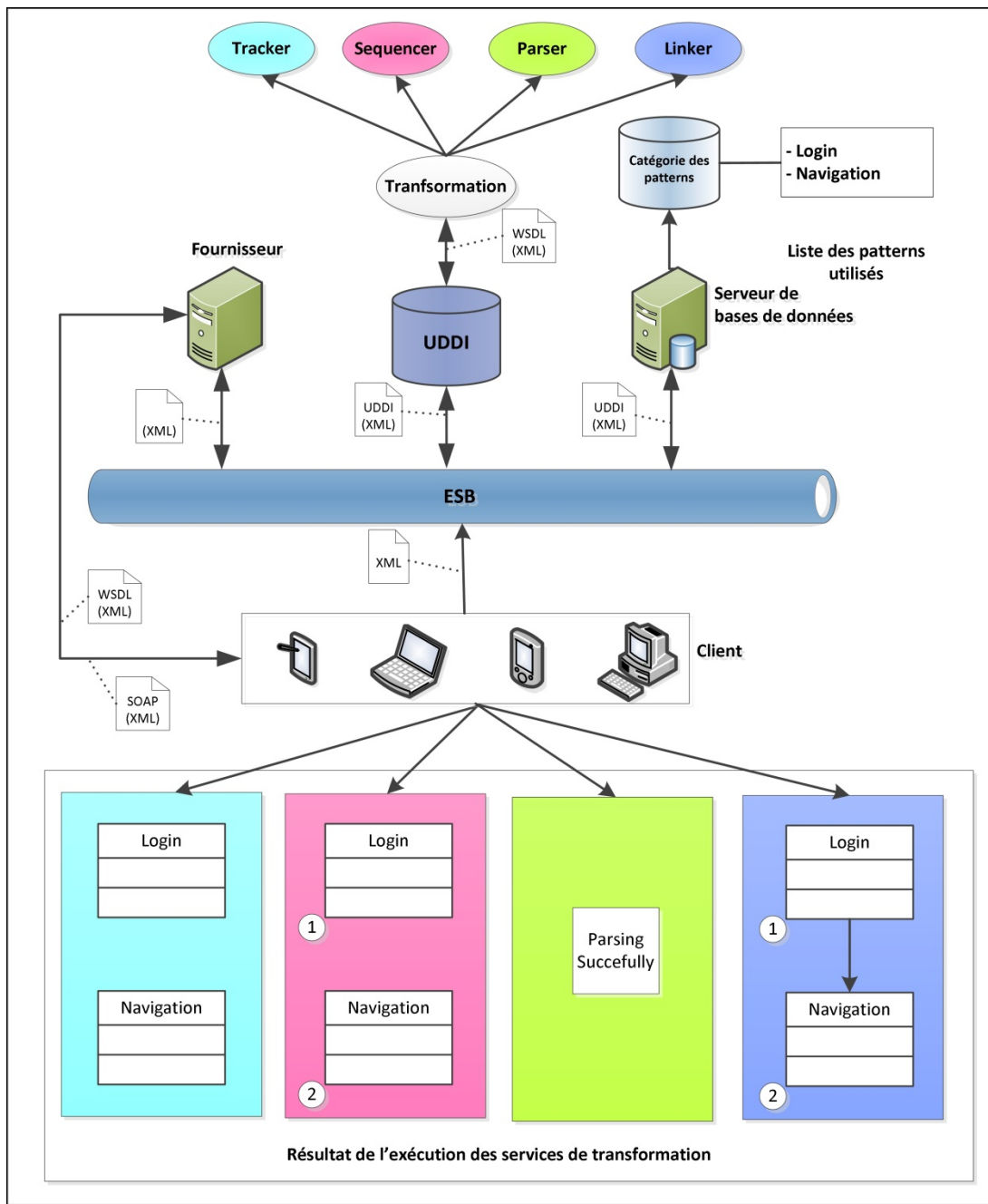


Figure 4.5 – Exemple de services de transformation.

Étape 1. Le service « Tracker » permet de détecter les classes « Login » et « Navigation » et envoie le résultat au service « Sequencer » ;

Étape 2. Le service « Sequencer » crée la liste de référence et détermine le séquencement des objets. Il affecte la référence (1) à la classe « Login » et (2) à la classe « Navigation » et envoie le résultat au service Parser.

Étape 3. Le service « Parser » analyse l'existence d'erreurs de syntaxe des classes « Login » et « Navigation » et envoie la confirmation des classes au service « Linker ».

Étape 4. Le service « Linker » établit les liens entre la classe « Login » et la classe « Navigation » avec leurs cardinalités.

CONCLUSION

A. Résumé du projet

Ce projet a permis de proposer une nouvelle architecture pour le développement des systèmes interactifs. Cette architecture assemble tous les concepts de POMA et celle de la SOA pour définir l'architecture SOPOMA. L'architecture POMA combine les patrons et les modèles alors que la SOA apporte plus de flexibilité en offrant tous les avantages de l'orienté services. La nouvelle architecture SOPOMA facilite le développement des systèmes interactifs multiplateformes sur l'ordinateur traditionnel, le laptop, la tablette, le PDA, Palmtop (ordinateur de poche), téléphone intelligent et télévision interactive.

Pour atteindre l'objectif de recherche, une revue de littérature a été établie sur l'architecture orientée patron et dirigée par des modèles, en particulier sur les niveaux architecturaux et les catégories des patrons et les types de modèles, et sur l'architecture orientée services, en particulier, sur le concept service et les services Web.

Les différents composants de la nouvelle architecture orientée patron, dirigée par des modèles et orientée services (appelée **SOPOMA** pour **S**ervice-**O**riented and **P**attern-**O**riented and **M**odel-**D**riven **A**rchitecture) sont les suivants :

- Les niveaux architecturaux et catégories de patrons ;
- Les types de modèles PIM et PSM ;
- Les services, et les concepts clés de la SOA intégrés dans SOPOMA ;
- Les services de compositions de patrons permettant pour produire un modèle PIM ;
- Les services de transformation d'un modèle PIM à PIM ou PSM à PSM ;
- Les services de conversion « Mapping » de modèles PIM aux modèles PSM pour différentes plateformes ;
- L'Entreprise Service Bus est à la base de SOA permettant d'intégrer plusieurs applications et également des processus comme des services basés sur les normes

existantes et d'offrir des services de transfert de messages et d'interopérabilité sécurisés.

Une étude de cas a été effectuée sur la conception d'un prototype d'une interface utilisateur fonctionnel d'un système interactif de gestion de l'environnement. Le but est d'expliquer et de clarifier de quelle façon l'orienté services facilite le développement des systèmes en utilisant la nouvelle architecture SOPOMA.

B. Contributions essentielles

Ce projet a permis de proposer une architecture de développement des systèmes interactifs appelée : architecture orientée patron, dirigée par des modèles et orientée services (SOPOMA). Les principales contributions dans ce projet sont :

- Ajout d'une couche de services à l'architecture (POMA) pour lui offre plus de flexibilité et d'agilité ;
- L'utilisation des fonctionnalités des services Web (publier « *publish* », trouver « *find* », relier « *bind* ») pour exposer plus facilement la nouvelle architecture SOPOMA au monde extérieur ;
- L'utilisation du service de messagerie dans SOPOMA pour faciliter la communication et l'échange de messages ;
- L'intégration du concept ESB dans SOPOMA pour intégrer plusieurs applications utilisant l'architecture orientée services en se basant sur les technologies des services Web ;
- La conversion des règles de POMA en services dans SOPOMA pour leur donner une interface facilement accessible de l'extérieur ;
- Une étude de cas exploratoire est proposée pour illustrer les avantages de la SOA dans POMA, et de décrire de quelle façon l'utilisation des services facilite la conception des différents modèles.

C. Limitations et forces

Bien que l'architecture orientée services apporte beaucoup de facilité et d'avantages pour l'architecture POMA, SOPOMA inclut les limitations suivantes :

- Il est nécessaire de définir des mesures permettant d'évaluer objectivement l'applicabilité des patrons, des modèles et des services qui pourraient être utilisés dans SOPOMA. Les tests sur les services sont relativement faciles. Mais ils se compliquent lorsqu'ils commencent à se combiner.
- Les services ne permettent pas la flexibilité entre la spécification de la plateforme indépendante des interfaces, la forme de ces interfaces spécifique à la plateforme, et l'éventuelle l'implémentation de ces interfaces.
- Les services ne constituent pas une approche d'intégration du cycle de vie complet et l'interopérabilité des systèmes d'informations des entreprises comprenant les logiciels, le matériel, les humains et les pratiques commerciales.
- Le besoin de connaissances avancées dans certains domaines techniques utilisées dans la SOA telles que : l'ESB, services Web, etc. ;
- Dans le cas où les applications sont sujettes à de fréquents changements, dans un environnement homogène de l'entreprise, SOPOMA ne serait pas la meilleure solution. Particulièrement dans le cas de petites entreprises ayant des objectifs très précis, le coût d'implémentation et du changement risque d'être élevé ;
- Dans le cas des systèmes qui nécessitent des réponses en temps réel (tels que les systèmes embarqués utilisés dans les voitures), l'architecture SOPOMA ne conviendrait peut-être pas, car elle est basée sur une communication asynchrone pour assurer le faible couplage entre les consommateurs de services et les fournisseurs de

services. Le problème s'empire si le système tourne sur un logiciel appartenant à un seul vendeur.

Les forces de l'architecture SOPOMA sont les suivants :

- L'architecture SOPOMA encourage les concepteurs à considérer d'autres aspects de dialogue importants aux utilisateurs en utilisant les fonctionnalités de la SOA ;
- L'architecture SOPOMA facilite le développement des systèmes interactifs en utilisant les outils de développement de l'architecture orientée services ;
- L'utilisation des fonctionnalités publier « *publish* », trouver « *find* » et relier « *bind* » permet de faciliter la recherche et l'application des services appropriés dans la pratique.

D. Futurs travaux

La prochaine étape pour de ce projet sera d'essayer d'étudier les limites de cette architecture en :

- Standardisant l'architecture SOPOMA afin qu'elle soit exploitable pour le développement de n'importe qu'elle types de systèmes ;
- Décrire le processus de génération de code source ;
- Effectuer une évaluation de l'efficacité et la productivité de SOPOMA ainsi que la qualité des applications développées en utilisant cette architecture ;
- Établir un système de documentation des détails de chaque composant de l'architecture SOPOMA pour démontrer sa validité et sa bonne organisation ;
- Migrer l'architecture SOPOMA vers le nuage informatique (Cloud Computing) pour mieux gérer l'aspect sécurité et le bon fonctionnement des ressources informatiques en se détachant des problèmes quotidiens de la gestion informatique.

ANNEXE I

SCHÉMA DE CLASSIFICATION DES PATRONS PAR CATÉGORIES. Tiré de

(Taleb *et al.*, 2007b)

Niveau Architectural	Catégories de patrons	Exemple de patrons
Navigation	Cette catégorie de patrons implémente des techniques éprouvées pour naviguer à l'intérieur et/ou entre un ensemble de pages et de blocs d'informations.	Shortcut pattern Bread Crumb pattern Index Browsing pattern
Interaction	Cette catégorie de modèles se concentre sur les mécanismes d'interaction qui peuvent être utilisés pour réaliser des tâches et les effets visuels qu'ils ont sur la scène, en tant que tels ils se rapportent principalement à des transformations graphiques et de rendu.	Search pattern Executive Summary pattern
Presentation	Cette catégorie de patrons fournit des solutions sur la façon dont le contenu ou les services connexes sont visuellement organisés dans les surfaces de travail, la mise en page efficace de plusieurs espaces d'informations et la relation entre eux. Ces patrons définissent la structure logique et physique adaptée pour des pages Web spécifiques tels que la page d'accueil, des listes et des tableaux.	Home Page pattern List pattern Table pattern
Visualization	Cette catégorie de patrons suggère	Favourite Collection

	différentes représentations visuelles et des métaphores pour regrouper et afficher des informations en blocs cognitivement accessibles. Ils définissent principalement le format et le contenu de la visualisation, c'est à dire, la scène graphique et, en tant que telle sont essentiellement liées aux données et les transformé de conversion « mapping ».	pattern Bookmark pattern Frequently Visited Page pattern Navigation Space Map pattern
Interoperability	Cette catégorie de patrons a pour but de découpler les différentes couches d'une application Web. En particulier, entre le contenu, la boîte de dialogue et les vues ou les couches de présentation. Ces patrons sont généralement des extensions des patrons de conception Gamma tels que MVC (Modèle, Vue et Contrôleur) observer, les actions de commande patrons. Les patrons de communication et d'interopérabilité sont des patrons utiles pour faciliter la conversion « mapping » de la conception entre les plateformes.	Adapter pattern Bridge pattern Builder pattern Decorator pattern Façade pattern Factory pattern Method pattern Mediator pattern Memento pattern Prototype pattern Proxy pattern Singleton pattern State pattern Strategy pattern Visitor pattern
Information	Cette catégorie de patrons décrit différents modèles conceptuels et architectures pour organiser le contenu sous-jacent sur plusieurs pages, serveurs et ordinateurs. Ces patrons fournissent des solutions à des questions telles que quelles informations peuvent être ou devrait être présenté sur quel	Sequence pattern Hierarchy pattern Grid pattern

	périphérique.	
--	---------------	--

BIBLIOGRAPHIE

- Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiskdahl-King, I., et Angel, S. (1977). *A Pattern Language*. Oxford University Press. New York. USA.
- Alghamdi, A., Nasir, M., Ahmad, I., et Nafjan, K. A. (2010). *An Interoperability Study of ESB for C4I systems*. Paper presented at the International Symposium in Information Technology (ITSim), 2010.
- Alur, D. (2004). Core J2EE Patterns, Frameworks and Micro Architectures. Consulté le 14 avril, 2014, à partir de <http://www.corej2eepatterns.com/presos/CoreJ2EEPatterns2.pdf>
- Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., et Holley, K. (2008). SOMA: A Method for Developing Service-Oriented Solutions. *IBM systems journal*, 47(3), 377-396.
- Arsanjani, A., Zhang, L.-J., Ellis, M., Allam, A., et Channabasavaiah, K. (2007). S3: A Service-Oriented Reference Architecture. *IT professional*, 9(3), 10-17.
- Bachman, J., Kline, S., et Soni, B. (2005). A New Service-Oriented Architecture (SOA) Maturity Model. *4550/2007*, 1198-1205.
- Barry, D. K. (2010). Service-Oriented Architecture (SOA) Definition. *Web Services and Service-Oriented Architectures and Cloud Computing*. Consulté le 06 avril, 2014, à partir de www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html
- Bass, L., Clements, P., et Kazman, R. (2003). *Software Architecture in Practice*. Boston: Addison-Wesley Professional.
- Booch, G., Rumbaugh, J., et Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Reading, PA: Addison-Wesley.
- Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., et Orchard, D. (2004). W3C Working Group Note 11: Web Services Architecture. Consulté le 07 juin, 2014, à partir de <http://www.w3.org/TR/ws-arch/#stakeholder>
- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., et Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons Ltd.

- Christensen, E., Curbera, F., Meredith, G., et Weerawarana, S. (2001). Web Services Description Language (WSDL) 1.1. . Consulté le 16 novembre, 2013, à partir de <http://www.w3.org/TR/wsdl>
- Coad, P. (1992). Object-Oriented Patterns. *Communications of the ACM*, 35(9), 152-159.
- Coplien, J., O. (1998). The Patterns Handbook : Techniques, Strategies, and Applications, chapter Software Design Patterns: Common Questions and Answers. *Cambridge University Press, New York*, 311–320.
- Cottin, N., et Agour, F. (2007). XML-Family :Web Services Description W.S.D.L. Consulté le 12 janvier, 2014, à partir de <http://frendag.free.fr/AWSIA/Presentation%20WSDL/WSDL.pdf>
- Couturier, J. F. (2011). *Élaboration et Expérimentation d'une Méthodologie Agile Permettant la Migration vers une Architecture Orientée Services en PME à l'aide d'OpenUP*. (Mémoire de maitrise), École de Technologie Supérieure.
- Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design* : Pearson Education India.
- Erl, T. (2008). *SOA : Principles of Service Design*. New Jersey: Prentice Hall PTR. Upper Saddle River
- Fowler, M. (1997). *Analysis patterns: reusable object models* : Addison-Wesley Professional.
- Fowler, M. (2002). *Patterns of Enterprise Application Architecture* : Addison-Wesley Longman Publishing Co., Inc.
- Ganci, J., Acharya, A., Adams, J., De Eusebio, P. D., Rahi, G., Strachan, D., Utsumi, K., et Washio, N. (2006). *Patterns: SOA Foundation Service Creation Scenario* : International Technical Support Organization.
- Goldberg, A. (1984). *SMALLTALK-80: The Interactive Programming Environment* : Addison-Wesley Longman Publishing Co., Inc.
- Granlund, A., Lafrenière, D., et Carr, D. A. (2001, August 5-10). *A Pattern-Supported Approach to the User Interface Design Process*. Paper presented at the Proceedings of HCI International on Human-Comuter Interaction, New Orleans.
- Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H. F., A, K., et Y, L. (2007). SOAP Version 1.2 Part 1: Messaging Framework. Consulté le 10 avril, 2014, à partir de <http://www.w3.org/TR/soap12-part1/>

- Guilbault, M. (2007). *Application de l'approche de Modélisation par archétype selon les normes et concepts de l'Architecture Orientée Services*. (Mémoire de maîtrise), École de Technologie Supérieure, Montréal.
- Haas, H., et Brown, A. (2004). Web Services Glossary. Consulté le 16 novembre, 2013, à partir de <http://www.w3.org/TR/ws-gloss/>
- Haines, M. N., et Rothenberger, M. A. (2010). How a Service-Oriented Architecture May Change the Software Development Process. *Communications of the ACM*, 53(8), 135-140.
- Hérault, C., Thomas, G., et Lalanda, P. (2005). *Mediation and Enterprise Service Bus: A Position Paper*. Paper presented at the Proceedings of the First International Workshop on Mediation in Semantic Web Services (MEDIATE 2005).
- Heubès, C. (2009). SOA : Du composant au service : Le contrat standardisé Consulté le 16 juin, 2013, à partir de <http://blog.xebia.fr/2009/03/04/soa-du-composant-au-service-le-contrat-standardise/>
- IBM. Centre de documentation IBM Business Process Manager: Création de Flux de Médiation. Consulté le 14 avril, 2014, à partir de <http://pic.dhe.ibm.com/infocenter/dmndhelp/v7r5m1/index.jsp?topic=%2Fcom.ibm.wbpm.wid.integ.doc%2Ftopics%2Ftblmed.html>
- Iheb, A. (2011). *Vers une Démarche d'Aide à la Décision pour l'Identification des Services d'une Architecture Orientée Services*. (Mémoire de maîtrise), École de technologie supérieure.
- Jerstad, I., Dustdar, S., et Thanh, D. V. (2005). *A Service Oriented Architecture Framework for Collaborative Services*. Paper presented at the 14th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise IEEE
- Johnson, R. E. (1997). Frameworks=(Components+ Patterns). *Communications of the ACM*, 40(10), 39-42.
- Koch, N. (2006). *Transformation Techniques in the Model-Driven Development Process of UWE*. Paper presented at the Workshop Proceedings of the Sixth International Conference on Web Engineering, Palo Alto.
- Kohls, C., et Wedekind, J. (2010). *Investigations of E-learning Patterns: Context Factors, Problems, and Solutions*. Hershey: Information Science Reference - IGI Global.

- Kumara, I., et Gamage, C. (2010). *Towards Reusing ESB Services in Different ESB Architectures*. Paper presented at the Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual.
- Lea, D. (1997). Patterns-Discussion FAQ. Consulté le 14 novembre, 2013, à partir de <http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html>
- Lemrabet, Y. (2012). *Proposition d'une Méthode de Spécification d'une Architecture Orientée Services Dirigée par le Métier dans le Cadre d'une Collaboration Inter-Organisationnelle*. (Mémoire de maitrise), Ecole Centrale de Lille, Lille.
- MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., et Metz, R. (2006). Reference Model for Service Oriented Architecture 1.0. *OASIS Standard*, 12.
- Mall, R. (2009). *Fundamentals of Software Engineering* : PHI Learning Pvt. Ltd.
- Mani Chandy, K. (2005). What is Different About Event Driven Architecture? Consulté le 07 Mai 2014, à partir de http://www.gartner.com/it/content/616700/616710/mani_chandy_research.pdf
- Maréchaux, J.-L. (2006). Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus. *IBM Developer Works*, 4-6.
- Meijers, M. (1996). Tool support for object-oriented design patterns. *Dept. of Computer Science INF-SCR-96-28, Utrecht University (August 1996)*.
- Miller, J., et Mukerji, J. (2003, June 01). MDA Guide Version 1.0. 1. *Object Management Group*.
- Myers, B. A. (1986). *Visual programming, programming by example, and program visualization: A taxonomy*. Paper presented at the ACM SIGCHI Bulletin, New York.
- Myers, B. A. (1990). Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing*, 1(1), 97-123.
- Myers, B. A., et Buxton, W. (1986). *Creating highly-interactive and graphical user interfaces by demonstration*. Paper presented at the ACM SIGGRAPH Computer Graphics.
- Olvani. Web design multi-supports. Naviguer facilement sur le web, les tablettes et les mobiles avec le même site. Consulté le 04 septembre, 2014, à partir de <http://www.olvani.com/sites-web/>
- OMG. (2008). Model-Driven Architecture Home Page. Consulté le 23 décembre, 2013, à partir de <http://www.omg.org/mda/index.htm>

- OMG. (2009). The OMG and Service-Oriented Architecture. Consulté le 10 juin, 2013, à partir de <http://www.omg.org/attachments/pdf/OMG-and-the-SOA.pdf>
- Papazoglou, M. P. (2003, December, 10-12). *Service-Oriented Computing: Concepts, Characteristics and Directions*. Paper presented at the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.
- Parveen, T., et Tilley, S. (2008). *A Research Agenda for Testing SOA-Based Systems*. Paper presented at the Systems Conference, 2008 2nd Annual IEEE.
- Perry, D. E., et Wolf, A. L. (1992). Foundations for the study of software architecture. *SIGSOFT Softw. Eng. Notes*, 17(4), 40-52. doi: 10.1145/141874.141884
- PhoenixUniversity. (2011). Advantages and Limitations of CASE Tools. Consulté le 08 septembre, 2014, à partir de <http://www.petruska.com/xoops1/modules/AMS/print.php?storyid=15>
- Pingshun, D. (2011). *Design and Implementation of ESB based on SOA in Power System*. Paper presented at the 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT).
- Plouin, G., et Pascal, G. (2007). SOA Perception des Entreprises Françaises Consulté le 10 avril, 2014, à partir de <http://www.youscribe.com/Product/Download/213297?embed=false&format=1>
- Radatz, J., Geraci, A., et Katki, F. (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std*.
- Rising, L. (1996). Reuse at AG communication systems= patterns. *Multiuse express*, 4(3).
- Rosen, M., Lublinsky, B., Smith, K. T., et Balcer, M. J. (2008). *Applied SOA*. Indianapolis: Wiley.
- Rouillard, J., Vantroys, T., et Chevrin, V. (2007). Architectures Orientées Service. Une Approche Pragmatique des SOA. *Coll. « Série génie logiciel »*. Vuibert, 317.
- Sandy, C. (2007). The New Language of Business: SOA & Web 2.0. *IBM Press*, 320.
- Schmelzer, R. R. (2007). The Seven Levels of Loose Coupling. Consulté le 19 janvier, 2014, à partir de <http://searchsoa.techtarget.com/tip/The-seven-levels-of-loose-coupling>

- Shah, M. (2007). SOA Reusability: Shrinking the Lag between Business and IT. *Java.net The Source for Java Technology Collaboration*. Consulté le 01 novembre, 2013, à partir de <https://today.java.net/pub/a/today/2007/07/24/soa-reusability-shrinking-lag-time.html>
- Taleb, M. (2008). *A Pattern-Oriented and Model-Driven Architecture for Interactive Systems*. (Doctoral thesis), Ecole de Technologie Supérieure Montreal.
- Taleb, M., Javahery, H., et Seffah, A. (2006, July 26-28). *Pattern-Oriented Design Composition and Mapping for Cross-Platform Web Applications*. Paper presented at the The XIII International Workshop on Design, specification and verification of interactive systems, Trinity College Dublin.
- Taleb, M., Seffah, A., et Abran, A. (2007a, July 22-27). *Model-Driven Architecture for Web Applications*. Paper presented at the 12th International Conference on Human Centered Interaction International, Beijing International Convention Center.
- Taleb, M., Seffah, A., et Abran, A. (2007b, March 3-6). *Pattern-Oriented Architecture for Web Applications*. Paper presented at the 3rd International Conference on Web Information Systems and Technologies (WEBIST), Barcelona.
- Taleb, M., Seffah, A., et Abran, A. (2007c, August 13-15). *Patterns-Oriented Design Applied for Cross-Platform Web-based Interactive Systems*. Paper presented at the International Conference on Information Reuse and Integration, 2007.
- Taleb, M., Seffah, A., et Abran, A. (2009, July 13-16). *Interactive Systems Engineering: A Pattern-Oriented and Model-Driven Architecture*. Paper presented at the The 2009 International Conference on Software Engineering Research and Practice (SERP'9)), Las Vegas.
- Taleb, M., Seffah, A., et Abran, A. (2010a). Generation Rules in POMA Architecture. *Journal of Software Engineering and Applications*, 3.
- Taleb, M., Seffah, A., et Abran, A. (2010b). Investigating Model-Driven Architecture for Web-based Interactive Systems. *Journal of eMinds : International Journal of Human-Computer Interaction*, 2(6).
- Taleb, M., Seffah, A., et Abran, A. (2010c, July 12-15). *Transformation Rules in POMA Architecture*. Paper presented at the Software Engineering Research and Practice (SERP'10).
- Taleb, M., Seffah, A., et Abran, A. (2013). *A UsiXML Proposal for a Pattern-Oriented and Model-Driven Architecture for Interactive Systems*. Paper presented at the

ADVCOMP 2013, The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences.

TheOpenGroup. (2013). Service Oriented Architecture : SOA Features and Benefits. Consulté le 19 septembre, 2014, à partir de http://www.opengroup.org/soa/source-book/soa/soa_features.htm

Webopedia. Loose Coupling. Consulté le 15 juin, 2013, à partir de http://www.webopedia.com/TERM/L/loose_coupling.html

White, S. A., et Miers, D. (2008). BPMN Modeling and Reference Guide: Understanding and using BPMN. *Future Strategies Inc. Lighthouse Point, Fla*(1).

Yacoub, S., et Ammar, H. (2003). Composition of Design Patterns : Addison Wesley.

Zachman, J. A. (1987). A Framework for Information Systems Architecture. *IBM systems journal*, 26(3).