République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université 8 Mai 1945-GUELMA

Faculté des Mathématiques et de l'Informatique et des Sciences de la Matière

Département d'Informatique

Mémoire

En vue de l'obtention du diplôme de Magister en informatique

Option : Génie logiciel

Présenté par : Mlle. MADI LEILA

THEME

D'une modélisation dans la conception d'un système d'information vers une architecture orientée services (SOA)

Soutenu publiquement le :23/05/2012

Devant le jury :

PRESIDENT	**Mr SERIDI HAMID**	**Pr**	**Université 08 MAI 1945 Guelma**
RAPPORTEUR	**Mme SERIDI HASSINA**	**MC**	**Université Baji Mokhtar d'Annaba**
EXAMINATEUR	**Mr YACINE LAFIFI**	**MC**	**Université 08 MAI 1945 Guelma**
EXAMINATEUR	**Mr TAREK KHADIR**	**Pr**	**Université Baji Mokhtar d'Annaba**

Année 2012

Remerciements

Ce travail n'aurait pas abouti sans mon directeur de mémoire : 'Dr SERIDI HASSINA' qui y a participé avec ardeur en me procurant sa précieuse aide et en m'éclairant par ses conseils. Je le remercie vivement.

J'exprime également ma gratitude et mon profond respect à l'égard du Pr 'SERIDI HAMID' de m'avoir fait l'honneur de présider le jury.

Je tiens également à exprimer ma reconnaissance aux jurys : 'Dr Yacine Lafifi' et 'Pr Tarek Khadir', d'avoir accepté de faire partie du jury et pour leurs critiques constructives.

Que Messieur 'Meadi Nadjib' ainsi que tous les membres de l'administration du département de l'Informatique, trouvent également l'expression de mon profond respect à travers ces lignes.

Résumé

Les répercussions de la mondialisation obligent les entreprises a modernisé leurs systèmes d'information basés sur des applications anciennes. Ces systèmes, connus par ' systèmes patrimoniaux', sont basés sur des logiciels et des sources de données hétérogènes, résultant de l'utilisation successive de technologies diverses, ou de l'acquisition d'autres sociétés d'où la naissance de la notion de EAI (Enterprise Application Intégration), qui rassemble les différentes applications au sein d'une même entreprise. Mais a cause des problèmes qui ont fait face à l'EAI, une nouvelle solution basés sur les services Web a vu le jour.

Dans ce mémoire, nous nous intéressons à l'annotation des services web extraits a partir d'un système patrimonial en adoptant l'approche boite grise. Par conséquent, nous enrichissons une approche qui englobe toutes les étapes nécessaire a l'extraction des services web en offrant leur annotation sémantique par le standard SAWSDL.

Cette annotation est basée sur une ontologie de domaine et contextuelle crée à partir d'un modèle UML reflétant le système étudié. L'annotation sémantique a pour objectif de surmonter le problème des conflits sémantiques ou d'une autre manière le problème qu'un concept peut avoir différents sens selon son contexte.

Mots clés : Système patrimonial, SOA , Annotation sémantique, SAWSDL, Ontologie.

Abstract

The impact of globalization is forcing companies to upgrade their information systems based on legacy applications. These systems, known as 'legacy systems', are based on software and heterogeneous data sources, resulting from the successive use of various technologies, or the acquisition of other companies where the birth of the concept of EAI (Enterprise Application Integration), that brings together different applications within an enterprise. In another way, problems that have faced the EAI, a new solution based on Web services has emerged.

In this paper, we focus on the annotation of web services which were extracted from a legacy system by adopting the gray box approch .Therefore, we enrich an approch that encompasses all steps necessary to retrieve web services by offering their semantic annotation by the standard SAWSDL.

This annotation is based on domain ontology and context ontology created from a UML model that reflects the system under study. The semantic annotation is intended to overcome the problem of semantic conflicts or otherwise the problem that a concept can have different meanings depending on context.

Keywords: legacy systems, SOA, Semantic Annotation, SAWSDL, Ontology.

الملخص

العولمة تجبر الشركات على تطوير انظمة المعلومات الخاصة بهم و المكونة على أساس التطبيقات القديمة. وتستند هذه الانظمة، والمعروفة باسم 'الانظمة الموروثة'، على البرمجيات وعلي مصادر البيانات غير المتجانسة، الناتجة عن الاستخدام المتعاقب للتقنيات المختلفة ، أو الاستحواذ على شركات أخرى حيث ولادة مفهوم EAI (تكامل تطبيقاث المؤسسة) الذي يجمع بين التطبيقات المختلفة داخل المؤسسة. ولكن بسبب المشاكل التي واجهت EAI، برز حل جديد يستند الى خدمات ويب.

نركز في هذه المذكرة على شرح للخدمات على شبكة الإنترنت المستخرجة من النظام الموروثة من خلال اعتماد نهج' العلبة الرمادية'. ولذلك، فإننا نثري نهجا يشمل جميع الخطوات اللازمة لاسترداد خدمات الويب من خلال تقديم شرح الدلالي لهم بستعمال SAWSDL.

ويستند هذا الشرح على الأنطولوجيا المجال والسياق المنشأة من طراز UML التي تعكس النظام المدروس. ويهدف هذا الشرح الدلالي للتغلب على مشكلة الصراعات الدلالية، بمعني اخرالتغلب علي مشكلة أن المفهوم يمكن أن يكون له معان مختلفة تبعا للسياق.

الكلمات الرئيسية : النظم القديمة، الخدمية ، الشرح الدلالي، الأنطولوجيا،

SAWSDL

Sommaire

Chapitre II : Les services web

Chapitre III : Approche proposée

Chapitre IV : Etude de cas

Table des illustrations

Liste des figures

Liste des tableaux

Introduction Générale

Introduction générale

Les répercussions de la mondialisation obligent les entreprises à modifier en profondeur leur organisation et à s'ouvrir davantage à leur environnement a fin de répondre aux exigences du marché et renforcer leur compétitivité en s'appuyant sur les technologies de l'information et de la télécommunication. Le Système d'Information (SI) d'une entreprise est généralement basé sur des logiciels et des sources de données hétérogènes, résultant de l'utilisation successive de technologies diverses, ou de l'acquisition d'autres sociétés.

Cette hétérogénéité a amené les directions informatiques à considérer la problématique d'intégration comme un enjeu majeur dès lors que les entreprises ont cherché à faire communiquer différents départements ou services entre eux pour optimiser leurs processus métiers. C'est à dire la possibilité d'accéder aux données importantes à partir des différents points, pour garantir la cohérence, l'actualité d'information, en d'autres termes, il faut intégrer toutes les sources de données avec les applications dans un seul système, c'est le but de l'EAI (Enterprise Application Integration) [3].

L'EAI signifie la construction du système par assemblage des systèmes préalablement réalisés, à travers un réseau reliant les machines d'une entreprise ou des entreprises impliquées, en vérifiant que leurs interactions confèrent au système les propriétés attendues. L'EAI fournit un cadre capable de prendre en compte la synchronisation des processus en question. Malgré ça, nous pouvons conclure qu'elle ne présente pas une solution finale aux défis de l'intégration a cause des problèmes qu'ils lui ont fait face, tels que les coûts du logiciel d'intégration et sa complexité, ceci exige un effort énorme et le grand problème résidant dans l'inflexibilité des solutions existantes. Ces problèmes ont guidé les entreprises à chercher et choisir d'autres solutions. Parmi ces nouvelles solutions nous trouvons l'implémentation des systèmes EAI basés sur les services Web.

Un service Web est un système logiciel conçu pour supporter l'interaction interopérable de machine à machine au-dessus d'un réseau. Il est considéré comme une interface placée entre le code d'application et l'utilisateur de ce code. Les services Web permettent les échanges des informations entre différents systèmes

d'informations à travers le Web en utilisant des standards tels que SOAP, UDDI, WSDL, HTTP...etc.

Nul doute que les systèmes patrimoniaux représentent aujourd'hui le cœur des applications informatiques de nombreuses sociétés qui dépendent de ceux-ci dans leurs activités quotidiennes et à l'intérieur desquels sont souvent enfouies les spécificités dont elles tirent leur avantage compétitif. Pour ces entreprises, l'accès aux nouvelles technologies par la mise en place de progiciels ou la réécriture de toutes leurs applications n'est souvent pas la meilleure solution, compte-tenu des coûts, risques et délais associés. Rester sur leur système actuel s'avère très coûteux et ne permet pas facilement d'intégrer les anciennes plates-formes (en particulier Mainframe) dans les Environnements modernes (l'Internet). Grâce à la technologie services Web, cette solution est passée à une autre meilleure qui est la migration des systèmes patrimoniaux vers les nouvelles technologies. Dans le domaine de migration des systèmes patrimoniaux, trois approches ont été définies dans différents travaux.

La première approche propose d'intégrer les applications via des adaptateurs (appelés Wrappers), pour que les applications puissent être invoquées comme des services Web. Dans ce type de migration, le système patrimonial est vu comme une « boîte noire » parce que seules les interfaces exposées par ces systèmes sont analysées et le noyau est ignoré. Cette approche présente une solution à court terme et peut effectivement compliquer la maintenance et la gestion du système au cours du temps.

La deuxième approche, appelée « boîte blanche », nécessite une phase de compréhension du système patrimonial qui permet d'extraire la logique de métier (Business Logic) et ensuite une deuxième phase d'implémentation de cette logique dans des nouveaux services. Cette approche permet la réutilisation des règles métier existantes dans le système patrimonial. L'inconvénient majeur de cette approche réside dans la difficulté de la récupération de la logique des métiers à cause des phases successives de maintenance et le manque de documentation du cycle de vie du système.

La dernière approche, dite approche « boîte grise », est une combinaison des deux approches précédentes. Cette approche vise à l'extraction des parties des systèmes patrimoniaux qui ont une logique de métier importante et les intégrer comme des services Web. Cette approche est économique et plus pratique par rapport aux deux approches précédentes, parce que l'extraction de quelques fonctionnalités

intéressantes d'un système patrimonial et leur réutilisation comme des services Web se font d'une manière aisée.

a. Objectif du travail

Notre travail de recherche s'inscrit dans le cadre de la migration des systèmes patrimoniaux vers une architecture orientée service. Dans notre mémoire nous nous intéressons à l'annotation des services web extraits a partir d'un système patrimonial en adoptant l'approche boite grise. Selon notre point de vue, tous les travaux existants ont abordé l'aspect syntaxique mais présentent un inconvénient majeur celui de l'hétérogénéité sémantique des services web résultants.

Pour pallier a cette insuffisance, nous enrichissons une approche générique en offrant une annotation sémantique des services web par le standard SAWSDL .cette annotation est basée sur une ontologie de domaine et contextuelle crée à partir d'un modèle UML reflétant le système.

Cette annotation a pour objectif de surmonter le problème des conflits sémantiques ou d'une autre manière le problème qu'un concept peut avoir différents sens selon son contexte.

b. Organisation du document

Le reste de ce mémoire sera subdivisé en trois parties, la première est consacrée à l'état de l'art, la deuxième contient notre solution de la problématique posée et nous terminerons avec la troisième partie où sera exposée une étude de cas pour valider notre solution.

Dans la première partie de ce mémoire, nous allons présenter un état de l'art qui est composé de deux chapitres.

Dans le premier chapitre, nous présentons une étude approfondie sur le domaine d'intégration des applications des entreprises (EAI) tout en exposant quelques définitions de l'EAI tirées de littératures, les différents objectifs du processus d'intégration, une architecture pour l'EAI suivie de ses types, les modèles d'intégration logique et le modèle de maturité pour l'intégration des applications.

Apres nous entamerons les systèmes patrimoniaux et les techniques utilisées pour les faire migrer vers une architecture orientée services Web, en présentant d'abord les différentes phases du cycle d'évolution de ces systèmes patrimoniaux, et

ensuite nous exposerons quelques travaux de différentes approches, boîte blanche et boîte grise, pour l'extraction et la réutilisation des fonctionnalités existantes dans les systèmes patrimoniaux comme des services Web.

Dans le deuxième chapitre, nous nous intéresserons par les services Web, les différents standards utilisés pour les déployés et les avantages apportés par les services Web au monde industriel. Apres nous dévions vers le web sémantique ou nous présentons l'architecture sémantique, les conflits sémantiques, notions d'ontologie et l'annotation des services web.

Dans la deuxième partie de ce mémoire, nous introduisons notre approche pour l'annotation des services Web, extraits à partir d'un système patrimonial.

La troisième partie, sera consacrée pour la validation de notre approche, où nous appliquerons cette approche sur une application patrimoniale qui a été développée en langage C standard.

Nous clôturons ce mémoire par une conclusion général.

Intégration des systèmes patrimoniaux

1. Introduction

Le Système d'Information (SI) d'une entreprise est généralement basé sur des logiciels et des sources de données hétérogènes, résultant de l'utilisation successive de technologies diverses, ou de l'acquisition d'autres sociétés.

Avec le développement rapide de l'informatique dans l'entreprise les conditions de l'échange de l'information et de la collaboration d'affaires entre différentes entreprises s'augmentent également, et les voix pour résoudre le problème des îlots des informations montent plus haut que jamais .

Cette variété dans les systèmes d'information a amené les directions informatiques à considérer la problématique d'intégration comme un enjeu majeur dès lors que les entreprises ont cherché à faire communiquer différents départements ou services entre eux pour optimiser leurs processus métiers, c'est à dire la possibilité d'accéder aux données importantes à partir des différents points, pour garantir la cohérence, l'actualité d'information, la bonne évolution de business d'une entreprise, et pour construire une base d'information complète et exhaustive.

De même pour les systèmes patrimoniaux ces besoins forcent les entreprises pour penser à remplacer leurs systèmes informatiques existants par d'autres plus avancés et capables d'améliorer leurs activités et rendement et être capables aussi de les diffuser dans le monde par l'intégration au Web.

Ces travaux ont donné naissance à un nouveau domaine de recherche, appelées EAI (Enterprise Application Integration), dont l'objectif est de définir tous les outils et les méthodes nécessaire pour accomplir l'intégration des applications dans une entreprise et à un temps très réduit. La technologie EAI peut être placée dans la catégorie des technologies informatiques d'intégration métier (Business Intégration) et d'urbanisation. Elle permet d'échanger des données en temps réel [2 ,7].

L'objectif de ce chapitre est d'explorer le domaine de l'intégration des applications dans l'entreprise, d'une part, et de s'aiguiller vers l'intégration des systèmes patrimoniaux au web, d'autre part. Alors notre premier chapitre est composé de deux sections.

Première section : Nous commençons cet état de l'art par l'introduction des différents systèmes d'information qui peuvent être existés dans une entreprise, dans ce cadre nous illustrons la signification du terme intégration, nous révélerons les

différents objectifs du processus d'intégration. Apres, Nous exposons une architecture pour l'EAI suivie de ses types. Nous entamons par la suite les modèles d'intégration logique en manifestant le modèle de maturité pour l'intégration des applications suivies des avantages apportés aux entreprises grâce à l'EAI et une synthèse.

Deuxième section : a pour objectif d'étudier les systèmes patrimoniaux et leurs approches de la modernisation et de la migration vers la SOA. Pour cela nous allons démarrer cette étude par définir le terme système patrimonial. En suite, nous exposons les différentes phases de cycle d'évolution d'un système patrimonial. Nous exposons dans la section qui suive les besoins qui imposent la phase de la modernisation et nous entamons aussi les défis qui gênent les agents de la maintenance. Et après nous énonçons les trois approches de la modernisation. Apres, nous arrivons à la section de la migration des systèmes patrimoniaux vers les services Web, on a entamé les approches existantes, qui sont essentiellement s'inspiré des approches vus dans la section précédente. On a vu des exemples sur les deux types des approches de migrations, Black et Grey Box. En fin, nous clôturons ce chapitre par une conclusion.

2. Intégration ou « homogénéité » pour les systèmes d'information des entreprises SIE

Nous avons évoqué, dans ce qui précède, que le SI d'une entreprise s'était construit au fil du temps, des besoins et des époques technologiques et le besoin de disposer d'un SI « homogène ». Les conséquences d'une « non homogénéité » sont nombreuses et présentes dans les entreprises, en voici quelques unes [9]:

- Ressaisie de données entre différents sous systèmes (îlots informatisés).
- Difficulté à consolider des informations en raison de structures de données ou de formats incompatibles.
- Impossibilité d'accéder à des données en raison de structures technologiques non ouvertes et/ou non communicantes.
- Processus métier à cheval sur deux sous-systèmes non communicants générant une rupture ou des délais d'exécution.
- Difficultés à regrouper des données reçues par des canaux différents et gérées par des systèmes physiques et applicatifs distincts.

- Contraintes pour faire évoluer un composant du système d'information sans remettre en questions d'autres éléments voisins en cascade.
- Manque de fluidité et de « temps réel » dans l'exploitation des informations stockées dans une perspective de pilotage d'activité.

Ces difficultés mettent en évidence le manque d'homogénéité, de cohabitation et d'échange entre les différentes parties en présence.

Le terme « intégration » viserait donc à proposer un SI évolutif, réactif et flexible, sans barrière technologique, répondant aux besoins de l'organisation et favorisant son évolution.

3. Notion de l'intégration des applications d'entreprises (EAI)

Dans les littératures, il existe plusieurs définitions de L'EAI parmi les quelles :

- « faire entrer un élément dans un ensemble de sorte qu'il en devienne une partie constitutive » (dictionnaire de l'académie française [10]).
- « assimiler, incorporer, faire entrer dans un ensemble en tant que partie intégrante » (dictionnaire Robert).

Ces définitions nous permettent de préciser que l'intégration consiste donc à incorporer un élément dans un ensemble plus vaste afin qu'il se confonde avec le tout.

Selon Meinadier [11], l'intégration des applications d'entreprise est :
« La phase de construction du système par assemblage des ses constituants préalablement réalisés en vérifiant que leurs interactions confèrent au système les propriétés attendues ».

Il existe d'autres définitions pour L'EAI selon deux perspectives [7, 11] :

- **Perspective technique**

La technologie EAI permet d'échanger des données entre deux ou plusieurs applications hétérogènes. Ces applications peuvent être développées indépendamment et peuvent utiliser des technologies différentes, l'EAI va s'occuper du transfert et de la conversion des données ».

L'EAI n'apporte pas une solution d'interfaçage mais un cadre d'intégration souple et robuste. Ainsi, les entreprises seront aptes à faire évoluer leur SI en

s'appuyant sur les solutions existantes. L'EAI se chargeant d'établir la communication entre les différentes applications existantes.

- **Perspective de Business**

Grâce à l'EAI, les entreprises peuvent réagir aux changements économiques tels que la fusion, l'élargissement de la concurrence ou de nouvelles acquisitions. La valeur de l'EAI se situe à ce niveau: pour relier un nouveau système, il suffit aux entreprises de « brancher » un connecteur allant du SI à l'EAI pour que le nouveau système puisse communiquer avec tous les autres composants du SI.

L'EAI facilite le suivi des échanges grâce aux flux fonctionnels. Autrement dit, si le SI présente un dysfonctionnement, il est possible de trouver la source du problème rapidement. Cette caractéristique permet de simplifier le suivi et la supervision. Ainsi le SI est plus robuste, et les dirigeants ont un meilleur contrôle des chiffres, ce qui leur permet de mieux anticiper les changements économiques.

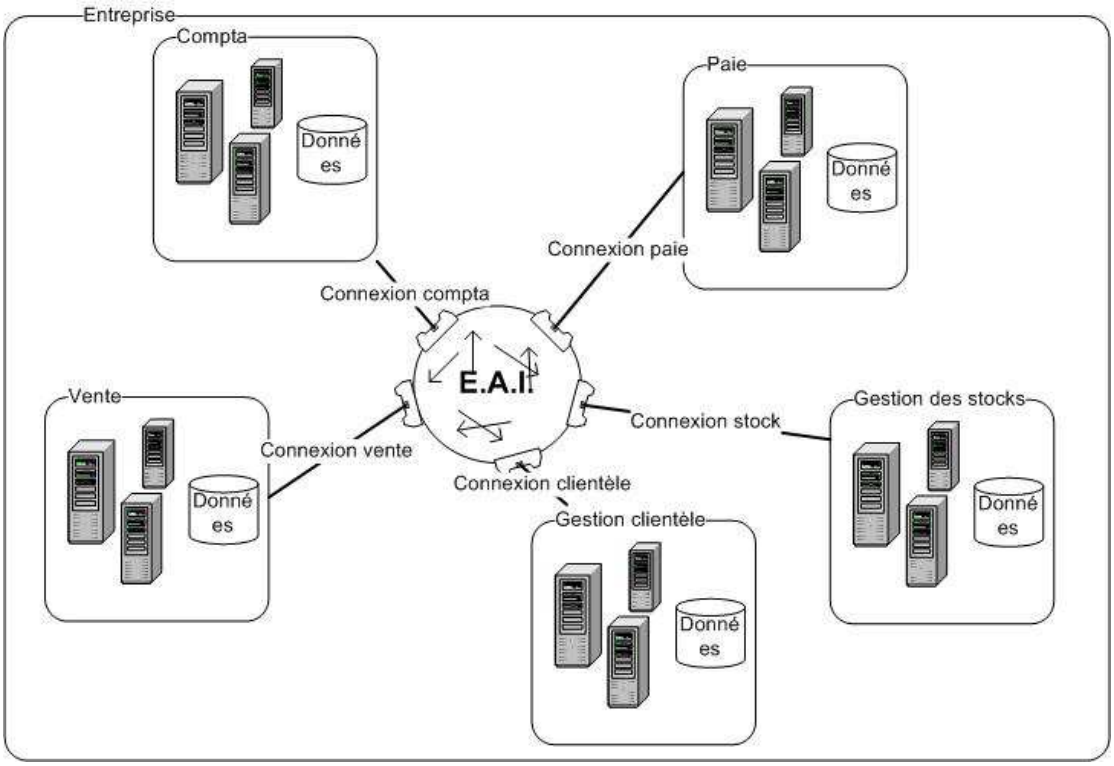


Figure I.1 : Le rôle de l'EAI [1]

Le nœud central, vu la [figure 1.1], va gérer les échanges entre les différentes applications du SI. Il apporte une notion de découplage. C'est-à-dire, tous les liens entre les applications sont désormais remplacés par une liaison unique partant de l'application X vers l'EAI. L'EAI peut aussi être vu comme le cœur de l'application, auquel chaque application se connecte indépendamment.

4. Les objectifs

Un système EAI cherche à faire interagir de manière optimale divers systèmes hétérogènes de l'entreprise afin de [3], [4] et [5]:

- Constituer un support efficace au déroulement des processus de l'entreprise en résolvant les problèmes de cohérence entre des systèmes qui interagissent,
- Fournir une structure capable d'évoluer en fonction des changements du marché (économiques, sociologiques, technologiques),
- L'intégration des nouvelles applications à l'existant par le biais de connecteur.
- La gestion des processus : grâce au BPM (Business Process Management), qui est une activité qui couvre la modélisation et l'exécution des processus métier, tous les processus de l'entreprise sont automatisés.
- La maîtrise de l'évolution du Système d'Information : toutes les applications sont connectées et contrôlées par l'EAI. Il est donc plus simple de maîtriser l'évolution du système d'information.
- La baisse des coûts de maintenance : comme toutes les applications ne peuvent communiquer que par le biais de l'EAI, les erreurs peuvent être focalisées rapidement.
- La baisse des coûts de traitement de l'information : l'EAI se charge de la conversion des données pour que celles ci soit compréhensibles par le destinataire.
- La traçabilité des échanges d'information entre les différentes applications : tous les échanges inter application sont enregistrés par l'EAI.
- La fusion du SI de plusieurs entreprises en cas de rachat: nous venons de voir que l'évolution du système d'information se réalise avec un connecteur. En cas de rachat, il suffira juste de relier le SI de la société achetée au SI de l'entreprise.

5. L'architecture de l'EAI

L'architecture, qui peut être considérée comme idéale, est une architecture qui permet à chaque application de se connecter au SI de façon indépendante et unique, sans avoir de connaissance a priori du domaine global. L'application obéit aux ordres du SI en traitant des tâches et en retournant des informations suivant l'exécution des processus métiers. Dans ce type d'architecture, le SI de l'entreprise peut être modulé de façon triviale, les ajouts, modifications ou suppressions d'applications n'ayant aucun impact sur le reste du domaine. L'architecture d'une EAI est composée de quatre couches [Figure I.2], qui permettant la liaison entre les processus de l'entreprise et ses applications [19] :

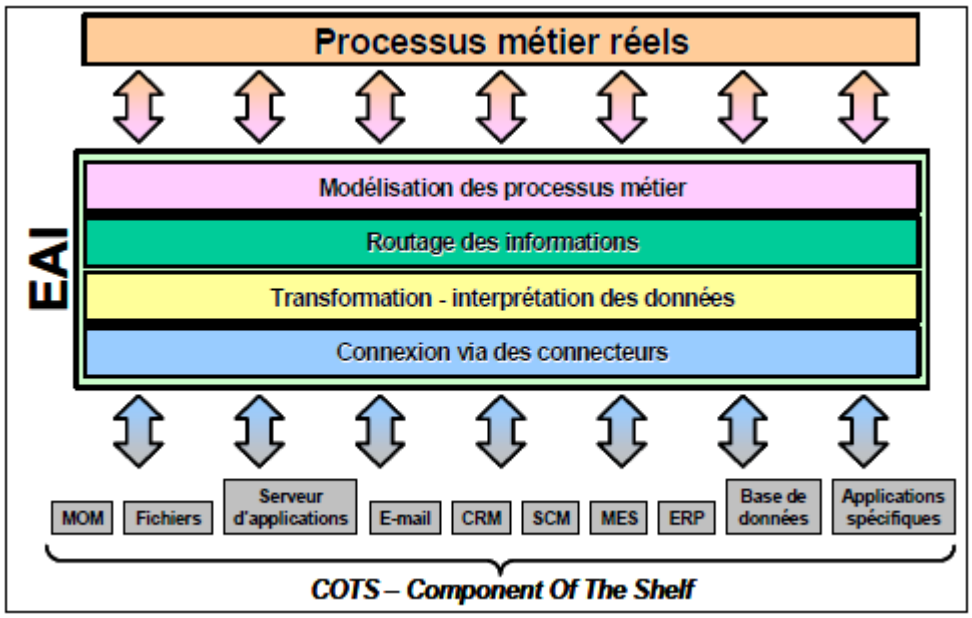


Figure I.2 L'architecture EAI [19]

6. Modélisation des processus métier

Les outils d'EAI sont généralement couplés à des outils de BPM18 (Business Process Management), qui automatisent les processus de l'entreprise. On définit alors les échanges entre les différents départements de l'entreprise. Cette couche illustre le positionnement stratégique des systèmes de workflow dans le contexte EAI. Les workflow permettent d'automatiser l'exécution d'une partie des processus de

l'entreprise. Cette automatisation consiste à traiter les activités nécessaires, les informations manipulées et les applications requises pour l'exécution de ces processus [19].

6.1 Routage des informations

Cette couche a la mission d'aiguiller les différents flux d'information vers les bonnes applications. Cet aiguillage est rendu possible grâce à la représentation des processus métier fournie par la couche modélisation décrite ci dessus. D'un point de vue technique, cet aiguillage peut notamment être rendu possible grâce à la mise en place d'outils logiciels permettant de gérer des files d'attentes de messages (les outils Middleware [19].

6.2 Transformation-interprétation des données

Cette couche sert à structurer le format des données véhiculées au sein de la plate forme EAI. Le but de cette couche est donc de transformer (dans les deux sens) le format des messages (données des diverses applications) en un format pivot propre à la plate-forme EAI. A l'heure actuelle, ce format pivot est généralement basé sur XML [19].

6.3 Connexion via des connecteurs

Cette couche propose des connecteurs spécifiques pour les différentes applications de SI afin de les faire communiquer dans une approche processus. Ces connecteurs permettent d'interroger les différentes applications pour récupérer et / ou insérer des données. Cette couche gère également la définition des protocoles de transport qui permettent d'acheminer et de distribuer les données.

Le principal reproche que l'on peut cependant faire aux outils d'EAI est leur aspect propriétaire. La logique d'intégration d'un EAI étant propriétaire, les éléments de l'outil (connecteurs, transformateurs de données, orchestrateur des processus) ne sont pas standardisés, ce qui lie l'entreprise à l'éditeur qu'elle aura choisi, avec les conséquences coûteuses (coût du conseil et des interventions, pérennité de la solution, etc.) [8].

7. Les types d'architecture de l'EAI

Il existe deux types d'architecture pour l'EAI sont L'architecture Hub and Spoke (Etoile) et L'architecture Network Centric (Bus applicative) .

7.1 Architecture Hub and Spoke (Etoile)

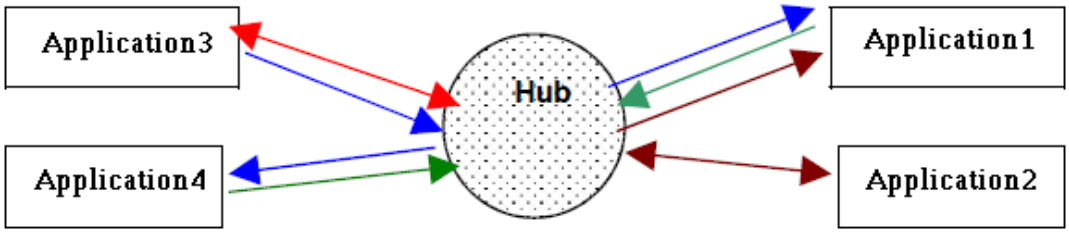


Figure I.3 : Architecture Hub and Spoke [66]

Il s'agit d'un modèle centralisé de l'EAI. C'est-à-dire, toutes les communications passent par un hub qui concentre les services sur un seul serveur. Les applications ne peuvent en aucun cas communiquer sans le hub. Quand une application envoie un message, ce dernier est expédié à destination du hub [65]. L'un des plus grands avantages de ce type d'architecture se situe au niveau de la supervision et de l'administration. Par contre, la gestion de la charge est complexe dans ce type d'environnement. Pour cela, il faudra multiplier les hubs sur les différents segments du réseau [8].

7.2 L 'architecture Network Centric (Bus applicative)

Il s'agit cette fois de la version décentralisée de l'implémentation de l'EAI [65].L'architecture bus applicative distribue les services sur plusieurs serveurs. Les référentiels de règle et des gestionnaires de messages sont dispersés sur tous les nœuds. Quand une application souhaite émettre un message vers une autre application, cette demande est traitée par le référentiel du nœud correspondant. L'un des avantages de ce type d'architecture se situe au niveau de la charge. Celui-ci est réparti sur l'ensemble des nœuds. Si on veut comparer les deux types d'architecture, il

est possible de dire que le modèle « bus » va offrir de meilleure performance que le modèle « hub », mais sa mise en œuvre est plus complexe [8].

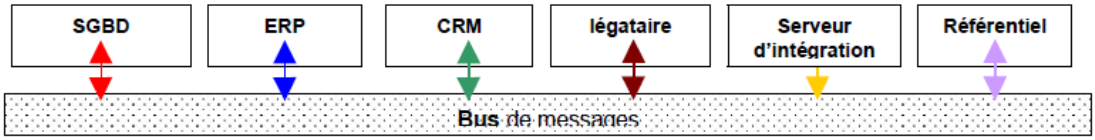


Figure I.4 : Architecture network centric [66]

7.3 Comparaison

Une différence pas uniquement technique. Une plate-forme de type "Network Centric" est facturée en premier lieu en fonction du nombre de nœuds; résultat, le client doit d'embler payer pour le périmètre des applications qu'il souhaite couvrir. En revanche, avec le modèle "hub and spoke", un client a la possibilité de n'utiliser qu'un seul hub dans un premier temps pour couvrir un large éventail d'applications. Autrement dit, l'architecture "hub and spoke" donne plus de souplesse que sa concurrente pour acheter des hubs au fur et à mesure de la montée en charge.

8. Le modèle de « maturité » pour l'intégration d'applications

Un organisme américain (American Management Systems) a proposé un modèle pour valider le degré d'intégration des applications des entreprises. Ce modèle a pour but de guider les entreprises dans les étapes conduisant à la gestion optimale de leurs capacités d'intégration d'applications en vue de satisfaire les objectifs de l'entreprise (économiques, sociologiques, technologiques) [6] . Le tableau suivant, synthétisé de plusieurs travaux, illustre les différents niveaux du modèle de « maturité »:

Tableau I.1 : le degré d'intégration atteint par les différents niveaux du modèle de maturité

Niveau d'intégration	Degré d'intégration atteint
Pré-intégration	• Systèmes indépendants ne disposant que de rares interfaces. • Processus peu réutilisables. • Synchronisation manuelle des données entre applications.
Intégration point à point	• Interfaces personnalisées pour des interactions point à point. • Utilisation d'interfaces programmables (API) ou d'outils de synchronisation de données. • Outils et bus permettant l'échange de messages. Systèmes faiblement couplés
Intégration structurelle	• L'architecture suit un modèle en étoile. • Le bus logiciel comprend des courtiers de messages ou des serveurs d'applications. • Caractéristiques des bus logiciels : transformation de données, intégrité des transformations. • Existence d'un modèle des interfaces des applications de l'entreprise.
Intégration des processus	• L'architecture des interfaces de niveau 2 est un fondement du niveau 3. • L'information entre les systèmes n'est plus simplement échangée mais elle est gérée. • Le bus logiciel comprend des outils permettant d'automatiser la modélisation des processus. • Caractéristiques des bus logiciels : modélisation des flux (workflow), automatisation des routages, automatisation des décisions. • Existence d'un modèle économique de l'entreprise (business model).
Intégration externe	• Cette intégration enrichit le niveau 3 et l'étend pour prendre en compte plusieurs organisations. • Infrastructure réseau commune (Internet, par exemple). • Adoption de standards pour l'échange des données (XML, par exemple). • Caractéristiques des bus logiciels : transactions sécurisées, agents intelligents, définition des sémantiques de l'application, adaptation des interfaces. • Application standard qui inclut le B2B et le B2C.

8.1 Synthèse

Ces niveaux ont pour objectif de structurer progressivement l'entreprise en définissant les processus et en mettant en œuvre des technologies capables de supporter leur déroulement. Cette structuration passe également par la maîtrise et l'amélioration des processus et des technologies sous-jacentes. Le dernier niveau de maturité (niveau 4) prend en compte la chaîne logistique d'une manière globale, ce qui implique l'adoption de standards et des protocoles de communication pour structurer les échanges .

9. Approches d'intégration

Traditionnellement, on distingue la typologie qui comprend principalement deux approches permettant de faire communiquer et coopérer des applications en pratique [14] :

- l'adhésion à des interfaces standardisées et l'utilisation d'intergiciels pour faire communiquer et faire coopérer diverses applications. Un exemple typique de cette approche est l'ensemble des standards industriels qui ont été développés tels que XML, EDI, STEP, etc.
- La seconde approche comprend les architectures et les technologies liées aux intergiciels telles que l'architecture CORBA, l'architecture DCOM, ou encore les intergiciels orientés messages (MOM – Message-Oriented Middleware) ou les outils EAI (Enterprise Application Integration).

La typologie du paragraphe précédent porte beaucoup plus sur les techniques d'intégration qui peuvent être mises en œuvre. Une analyse plus fine de l'état de l'art du domaine de l'intégration des applications d'entreprise permet de révéler en fait une multitude de dimensions permettant à la fois de caractériser la notion d'intégration et de classifier les approches d'intégration existantes. Le tableau suivant présente les principaux aspects de l'intégration synthétisée selon plusieurs travaux [14], [63] et [64]. Cette classification est à travers quatre dimensions fondamentales qui sont :
Les périmètres d'intégration, les points de vue d'intégration, les couches d'intégration et les niveaux d'intégration.

Tableau I.2 : Classification des approches d'intégration

Les critères	Approches d'intégration
les périmètres d'intégration	• **Intégration intra-entreprise** : implique les applications internes a l'entreprise et se décompose en deux sous classes : ➢ Intégration horizontale : concerne l'intégration logique et physique des processus métier. ➢ Intégration verticale : concerne des différents niveaux de gestion d'une entité organisationnelle. • **Intégration inter-entreprise** : implique les applications appartenant à des entreprises différentes.
les points de vue d'intégration	• **La vue externe (utilisateur)** : les différentes vues que possèdent des experts de domaine et les utilisateurs métier. • **La vue conceptuelle (concepteur)** : concerne les différents modèles utilisés lors de la conception des SI ou des applications. • **La vue interne (programmeur)** :
les couches d'intégration	L'intégration peut concerner une ou plusieurs couches du système d'information d'entreprise : • **Intégration par les données** : la couche données qui traite l'accès aux données manipulées. • **L'intégration par les traitements** : la couche de traitement d'application qui correspond aux règles métier implémentées. • **Intégration par les présentations** : la couche présentation qui gère l'interaction homme-machine • **Intégration par les processus** : la couche processus qui permet l'orchestration des applications.
les niveaux d'intégration	• **Intégration syntaxique** : permet de résoudre les hétérogénéités d'ordre syntaxique qui peuvent exister au niveau des modèles de données ou des signatures des fonctions et des processus.les approches principales de l'intégration syntaxique sont : ➢ L'approche par conversion: consiste à transformer la syntaxe d'un composant dans la syntaxe d'un autre composant. ➢ L'approche par modèle canonique: suppose l'existence d'un modèle pivot par lequel on doit transiter pour pouvoir faire interopérer deux composants du système d'information • **Intégration sémantique** : a pour but de résoudre les hétérogénéités sémantiques qui peuvent exister entre les applications. Parmi les solutions utilisées pour l'intégration sémantique, on peut citer : ➢ approches basées sur les ontologies qui permettent de définir une couche d'abstraction où tous les concepts d'un domaine y sont définis et où des mécanismes d'inférence peuvent être mis en œuvre en vue de résoudre les différences d'ordre sémantique.

10. Les difficultés d'intégration des applications

Il existe des difficultés qui font face à l'intégration des systèmes d'information et des applications dans les entreprises. Le tableau suivant illustre quelques problèmes de plusieurs niveaux :

Tableau I.3 : Illustration de quelques difficultés d'intégration des applications

Niveau de difficulté	Problèmes
Niveau du support technologique	-Les systèmes d'information des entreprises ont des niveaux d'avancement technologique différents, par exemple : les transactions et la sécurité : Plusieurs systèmes d'information sont des primitives et ne offrent pas des supports pour l'accès transactionnel, et il existe des autres plus avancés qui permettent un accès transactionnel aux ressources.
Les restrictions technologiques et administratives	l'utilisation des systèmes patrimoniaux est très compliqués, par exemple : l'ajout d'un nouveau client à une base de données d'un système patrimonial est très difficile. Donc, les entreprises qui possèdent ce type des systèmes exigent de l'adapter avec les nouveaux systèmes et il faut les intégrer avec les différentes applications de l'entreprise.
L'aptitude de l'intégration avec les autres systèmes	l'intégration des SIEs est difficile, parce que chacun de ces systèmes a été développé dans son propre environnement, qui ne peut pas être conforme avec les autres systèmes, et en plus n'ont pas les mêmes objectifs.
La difficulté de la compréhension des systèmes	Avant que le développeur démarre l'intégration des applications, il faut d'abord comprendre tous les détails de bas niveau de programmation des systèmes d'information. Alors il faut que le développeur étudie la bibliothèque du langage utilisé pour qu'il puisse utiliser les fonctionnalités de ce système.

11. Avantages et inconvénients de l'EAI

La technologie EAI permet de faire baisser les coûts d'intégration et d'augmenter le ROI (Return On Investment) c à d : la rentabilité du capital investi [7], [6].

- **Flux centralisés** : Avant l'arrivée de l'IAE, les entreprises devaient développer des interfaces spécifiques à chaque application et les connecter point à point. Il en résultait un réseau complexe (plat de spaghetti) de flux, difficile à maintenir et à faire évoluer. Maintenant, toutes les interfaces IAE convergent vers un serveur central (concentrateur ; en anglais, hub) qui traite et redistribue les flux vers les applications enregistrées.
- **Flux traités "au fil de l'eau"** : Les mises à jour des données sont effectuées au fil de l'eau, c'est-à-dire au fur et à mesure des événements des applications sources. Cela réduit les flots de donnée lors des transferts et propose une donnée "à jour" peu de temps après son éventuelle modification. Cela réduit aussi la perte de performance des applications due à l'extraction ou la mise à jour des données car on ne traite que des flots de petite taille et répartis dans le temps.
- **Flux réutilisable** : Si une nouvelle application veut accéder aux OM déjà présents dans l'EAI, toute la logique de récupération n'est plus à développer. En théorie elle n'a besoin d'ajouter au concentrateur EAI que sa collaboration (si elle a besoin d'un traitement spécifique), ses OMS, ses mappings et son connecteur.
- **Coût de migration des interfaces** : Lors du changement d'une des applications interfacées (migration, changement de produit), peu de modifications sont nécessaires. Seuls le connecteur, le mappage ou la collaboration spécifique à l'application doivent être modifiés.

Mais cette technologie risque de se heurter à de nombreux problèmes:

- **Flux massif** : Pour les flux massifs (par exemple : mise à jour de 10 000 articles en même temps), la logique du traitement unitaire de l'information est très lente.

- **Coût initial** : Le coût de mise en place de l'infrastructure est assez élevé. Mais il se réduit grandement au fur et à mesure de l'ajout de nouveaux flux.
- L'absence de référentiel commun peut être un frein devant cette technologie. Par exemple, si l'on souhaite faire une mise à jour pour 1000 articles en même temps, plus il y aura de SI différents, plus le traitement durera longtemps.
- L'EAI va permettre aux utilisateurs d'avoir à leur disposition l'ensemble des informations de l'entreprise en temps réel. Mais les projets EAI sont lourds car ils nécessitent l'interfaçage avec les applications concernées et ne garantissent pas la qualité des données

12. Intégration des systèmes patrimoniaux

Les systèmes patrimoniaux sont encore très présents dans la plupart des grandes entreprises et les grandes institutions des secteurs public et parapublic. Ces systèmes sont en général issus des technologies des ordinateurs centraux et ont été, pour la plupart, développés entre les années 1970 et 1995. De l'avis de plusieurs personnes consultées, il semble que les systèmes patrimoniaux qui possèdent les caractéristiques des technologies populaires des années 1980 sont là pour encore plusieurs années. Bien sûr, les organisations effectuent déjà, où sont en voie de le faire, d'importants investissements pour moderniser leurs portefeuilles d'applications de même que leurs infrastructures technologiques.

Alors, les entreprises sont obligées d'évoluer pour rester au premier rang du monde industriel. Elles doivent fréquemment fusionner avec d'autres entreprises, réorganisant leur structure interne, et adoptant de nouvelles technologies et plateformes pendant qu'elles essayent d'obtenir des avantages concurrentiels [67], [17]. D'une autre manière**,** les entreprises sont confrontées a la modernisation de leur système patrimonial a fin de pouvoir s'intégrer a d'autres entreprises dans le but de satisfaire au maximum les besoins des clients. On parle alors d'intégration inter entreprise et diffusion du système modernisé dans le monde par l'intégration au web.

Les propriétaires des entreprises, n'ont trouvé que deux solutions à faire pour leurs patrimoines applicatifs. Le premier est la modernisation pour les adapter avec les nouvelles technologies, mais cette solution est très compliquée. L'autre solution propose de traiter ces systèmes afin d'extraire le code qui implémente la logique

d'affaires et l'exposer comme des services Web ou même de le réutiliser dans le développement des autres systèmes d'information plus avancés et plus confiants.

12.1 Définitions des Systèmes patrimoniaux

Puisque l'origine de l'expression « système patrimoniaux » est l'expression anglaise « Legacy systems » parfois « heritage system », en Europe « vintage system, nous exposons d'abord une définition pour le terme « Legacy » extrait du dictionnaire anglais Oxford (Oxford English Dictionary) [49]: « Legacy : Une somme d'argent, ou un article spécifique, donné à l'autre près la volonté; quelque chose remis vers le bas ou reçu d'un ancêtre ou d'un prédécesseur. »

Les systèmes patrimoniaux sont des pièces du logiciel hérités, et sont précieuses. Nous pouvons dire aussi que les systèmes patrimoniaux sont des systèmes d'exploitation ou des programmes d'applications qui doivent être utilisés pour quelques raisons, telles que le coût énorme de remplacement et de reconception par rapport à sa compétitivité et compatibilité faibles.

Selon [48], le système patrimonial est:

« Tout système informatique hérité des années passées (en pratique 1970 à 1995), généralement développé sur mesure (« in-house ») pour supporter les fonctions importantes, sinon essentielles au fonctionnement des entreprises ».

Une autre définition propose que [50] : « Les Systèmes patrimoniaux sont des systèmes d'information ou logiciels développés dans l'époque avec des moyens et des techniques traditionnels par rapports ceux de nos jours et pour effectuer des tâches nécessaire aux activités des entreprises dans celles périodes. »

12.2 Caractéristiques des systèmes patrimoniaux

En général, le système patrimonial typique comporte les caractéristiques suivantes [48] :

- Une technologie centralisée de grande puissance (ordinateur « main frame »).
- Un système de gestion des transactions entre l'ordinateur central et les divers points d'entrée (terminaux), souvent distribués.
- Des langages de programmation de 2e et de 3e génération, tels COBOL, PL/1 ou d'autres moins répandus (v.g. Assembler) ;
- Un système de gestion de données (base de données) ;

- Une taille imposante en termes du nombre de « lignes de code » et du nombre de programmes / modules.
- Le résultat d'un développement maison (donc un produit personnalisé ou « sur mesure»).
- Un ensemble technologique (plate-forme, système d'exploitation, système de transactions, langages de programmation, système de gestion de données) arrivé au terme de son cycle de vie et dont le support soulève certaines incertitudes.
- Une documentation souvent incomplète ou absente, ce qui a pour conséquence que la contribution des personnes expérimentées est essentielle.
- l'évolution de ces systèmes se fait généralement au moyen de « versions » (implantations de 3 à 5 fois par année), regroupant en une livraison de multiples changements (souvent entre 20 et 30) demandés par les utilisateurs ou rendus nécessaires pour corriger certaines anomalies.

12.3 Les problèmes posés par le système patrimonial

L'identification du ou des problèmes permet de circonscrire les alternatives de solution possibles. Il existe essentiellement trois catégories de problèmes dans les systèmes patrimoniaux [60].

12.3.1 Le manque de documentation

Le manque de documentation pertinente et récente accompagnant un code source est un problème qui existe dans de nombreuses organisations. Dans ces dernières, il arrive souvent, lorsqu'on maintient un système, que l'on néglige l'entretien des documents de conception, pour ne maintenir à jour que le code source. Ainsi une entreprise peut se retrouver avec un système qui fonctionne mais qui n'est pas correctement documenté [60], [17].

12.3.2 Manque de compétence

Le manque de compréhension crée des difficultés à modifier et d'étendre le logiciel. Il peut être difficile de trouver des ingénieurs qui ont des compétences en technologie et en langage dont le quel le logiciel existant est développé.

12.3.3 Les problèmes techniques

a. Problèmes au niveau des traitements

Tableau I .4 Problèmes au niveau des traitements [60]

Problème	Description
Fragmentation	Une opération logique est effectuée en faisant appel à d'autres fonctions à travers divers systèmes distincts
Redondance	La même opération se retrouve plus d'une fois à travers un ou plusieurs systèmes
Diversité Technique	Des applications qui sont censés fonctionner ensemble sont sur des plates formes différentes, écrit dans des langages différents.
Diversité de conception	L'architecture d'un système ne suit pas une conception unique ou cohérente
Taille & Temps	L'application est trop volumineuse ou le temps d'exécution est trop long pour offrir des services adéquats .

b. Problèmes au niveau des données

Tableau I.5 Problèmes au niveau des données [60]

Problème	Description
Fragmentation	Des données liées sont distribuées à travers divers systèmes distincts
Redondance	Les mêmes données se retrouvent plus d'une fois à travers un ou plusieurs systèmes
Intégrité	Des informations contradictoires se retrouvent à travers un ou plusieurs systèmes
Sémantique	Des données homonymes (données ayant la même description) identifient des informations différentes Des données synonymes (données ayant une description distincte) identifient des informations identiques
Accessibilité	Les données ne sont pas accessibles aux usagers qui pourraient vouloir les manipuler
Flexibilité	Le système informatique ne peut pas être aisément modifié pour s'adapter aux besoins, changeants, des usagers dans un contexte d'affaires
Sécurité	Les données ne sont pas protégées contre une manipulation interdite ou erronée : lecture, ajout, modification, suppression. Les données ne sont pas encryptées pour empêcher de l'espionnage.

12.4 Evolution des systèmes patrimoniaux

Généralement, lorsqu'un projet de migration a été décidé, la préoccupation essentielle n'est pas de corriger ou d'améliorer l'application qui va disparaître mais de s'en débarrasser. Pourtant une nouvelle application doit s'intégrer dans l'environnement informatique de l'organisation. Cet effort d'intégration d'une

nouvelle application requiert souvent des améliorations de l'environnement informatique dans lequel il s'intègre et qui va demeurer présent. Il existe plusieurs manières de construire le système cible à partir du système patrimonial.

- Le système cible peut être construit en ayant une profonde compréhension des mécanismes du système patrimonial.
- Il est aussi possible de construire le système cible à partir du système patrimonial. Au minimum, pour modéliser cette construction il faut pouvoir identifier et garder la trace des nouveaux éléments logiciels qui vont apparaître, de ceux qui seront modifiés et de ceux qui seront supprimés.
- Construire le système cible sans porter atteinte au fonctionnement interne du système patrimonial.

12.5 Pourquoi la modernisation ?

Les raisons pour laquelle les entreprises se trouvent en besoin de la modernisation de leurs applications, sont résumées comme suit [12] :

- Les systèmes patrimoniaux ne sont pas capables de répondre à des objectifs de business, dus aux facteurs tels qu'une fonctionnalité est absente, le manque de support ou les cycles de développements est longs.
- Les systèmes patrimoniaux ne peuvent pas être intégrés avec d'autres systèmes internes même avec ceux du fournisseur ou du client
- Le système patrimonial souffre du manque d'habilité pour la maintenance ...etc.

12.6 Approches de modernisation

D'après [68], [12] il existe trois approches, l'approche « white box » et « black box » et « gray box » cette classification est basée sur le niveau de compréhension du code source.

a. L'approche boite noire

Modernisation sans analyser ou comprendre le code source ou « black box ». L'analyse se concentre sur la compréhension des interfaces de l'élément logiciel. La modernisation de l'élément logiciel correspond à une forme d'encapsulation. L'analyse ignore le code source et le fonctionnement interne de l'élément logiciel qui n'est pas modifié [60]. La (figure I.5) représente l'approche «boîte noire» pour faire migrer un système existant à un service Web.

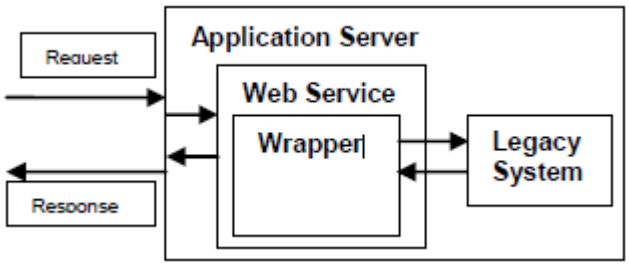


Figure I.5 approche boite noir (wrappers) [72]

Cette modernisation comporte des limites :

- **Limites**

La maintenance corrective se fait souvent en boite noire, sans comprendre le système patrimonial. Sans le savoir, du code peut être ajouté pour réaliser une fonction déjà programmée dans l'application. Cette nouvelle fonction constitue une redondance dans le système. Avec le temps, ces redondances peuvent s'accumuler. L'objet de la migration structurée est d'effectuer des changements contrôlés.

b. L'approche boite blanche

C'est un effort de modernisation en analysant et comprenant le code source ou « white box ». La compréhension du fonctionnement interne de l'application par une modélisation pointue est la base de l'effort de modernisation. Cela implique de produire ou analyser une modélisation de l'application et du domaine pour comprendre le code source.

c. Boîte grise

Cette approche est une combinaison de boîte noire et les approches de logique métier. Cette stratégie est la plus populaire à l'heure actuelle. Juste les parties du système patrimonial, ayant une logique métier pertinente, qui sont intégrés. Elle est basée sur l'analyse des besoins et la réingénierie du système. Il faut noter que cette approche est apparaît comme une approche rapide parce qu'elle s'intéresse à des parties du système qui disposent des valeurs précieuses dans la logique du business. L'approche boîte grise est également économique et pratique pour extraire quelques fonctionnalités comme composants ou services et pour profiter le support de la technologie de la programmation [73], [12].

d. Comparaison

Tableau I.6 : comparaison entre les différentes approches de modernisation

Approches de modernisation	Boite noire	Boite blanche	Boite grise
Degré de Compréhension du système	Faible	Fort (approfondie)	Moyen
Modélisation de l'application et du domaine	N'existe pas	Existe	Existe
Parties de système intéressantes	Les interfaces	Toutes les parties du système	Les fonctionnalités pertinentes
Modification apporté au Code source	Le gardé tel qu'il est et l'encapsulé par une interface moderne.	Restructuration en gardant le même niveau d'abstraction	. Garder les fonctionnalités pertinentes et suppression des inutiles. .Ajout de nouvelles fonctionnalités
Délais	Très courte durée	Langue durée	Courte durée
Cout	Economique	Grand investissement	Economique

13. Migration des systèmes patrimoniaux vers SOA

13.1 Redéveloppement (white box)

Le redéveloppement des anciennes applications implique la réécriture a partir de la base des applications héritées. Une approche générale pour le redéveloppement est proposée par Zhang et al :

- D'abord le code existant est analysé afin d'acquérir une compréhension de ses fonctionnalités.
- Cette analyse est utilisée pour identifier les processus d'affaires effectués par l'ancien système.
- Ces processus d'affaires sont ensuite mis en œuvre comme services [72].

Le logiciel hérité sera conçu à nouveau pour être utilisé dans une architecture SOA, par conséquent il sera plus facile à maintenir et à étendre dans les SOA. Cette

restructuration est généralement une transformation d'une représentation à une autre au même niveau d'abstraction pour améliorer la qualité du système, tout en conservant la fonctionnalité du système externe.

Bien que le redéveloppement produise des logiciels de qualités souhaitables, son coût en temps et en argent peut être prohibitif.

13.2 Emballage (black box)

La Modernisation en boîte noire concerne l'évaluation des entrées et sorties de l'ancien système, pendant le fonctionnement, pour acquérir des connaissances de l'interface. Cette tâche consiste à envelopper l'ancien système dans une couche logicielle qui masque la complexité inutile du système par une interface moderne.

Selon [72], [12] il ya deux modèles de base pour l'intégration des systèmes patrimoniaux comme des services Web les quels sont : modèle d'interface et modèle de passerelles (Gateway) présentés dans le tableau suivant :

Tableau I.7 les modèles de l'approche boite noire

	Interface	**Passerelle**
Les Modèles de l'approche boite noire	-les adaptateurs s'exécutent sur un ordinateur central. -utilise des interfaces qui permettent l'intégration parfaite. -n'exige pas de matériel additionnel. -Aucune modification n'est faite sur le système à émigré.	-l'exécution est faite indépendamment de l'ordinateur central -fonctionne dans des serveurs intermédiaires d'une architecture trois tiers. -L'emplacement des passerelles est très important parce qu'il ya des options pour accéder au serveur a partir du tiers intermédiaire, ce qui permet aux passerelles d'exiger une certaine forme de méthode *screen scraping.*

13.3 Approche boite grise

13.3.1 Approche basée sur les techniques de Clustering

Le clustering, en français regroupement ou partitionnement, est une tâche dont l'objectif est de trouver des groupes au sein d'un ensemble d'éléments (appelés par la

suite objets). Ces objets sont décrits par des caractéristiques, encore appelées attributs, qui décrivent les propriétés des objets. Les groupes recherchés, communément appelés des clusters, forment des ensembles homogènes d'objets du jeu de données partageant des caractéristiques communes. Il existe de très nombreuses méthodes de clustering permettant de créer ces clusters de manière automatique, chacune utilisant une stratégie et un objectif propre pour construire les clusters. Pour réaliser cette opération de regroupement, on fait fréquemment appel à la notion de similarité entre les objets. En effet, cette notion de similarité prend tout son sens en clustering car il s'agit d'évaluer à quel point deux éléments sont similaires.

a. Méthodes de clustering

Dans ce qui suit, nous présentons les principales familles de méthodes de regroupement des données en clusters. D'après [82] Les méthodes peuvent être séparées en quatre groupes :

Les méthodes basées sur une distance, basées sur une grille, Les méthodes probabilistes et Les méthodes hiérarchiques

a.1 Méthodes basées sur une distance

Les méthodes basées sur les distances sont parmi les premières méthodes de clustering à avoir été proposées et sont toujours très populaires aujourd'hui. Ces méthodes se basent sur la notion de distance entre objets du jeu de données, en posant que si deux objets sont proches suivant cette distance, ils doivent être regroupés ensemble dans un même cluster. Les principales méthodes utilisant une distance sont les méthodes à base de prototypes, les méthodes neuronales et les méthodes à base de densité.

a.2 Méthodes basées sur une grille

Les méthodes à base de grille sont fondées sur le principe de la discrétisation de l'espace des données. Celui-ci est décomposé en un ensemble de cellules qui forment l'unité de la grille. Ces méthodes ont été proposées pour réduire l'explosion combinatoire des méthodes à base de densité qui fait suite à l'augmentation du nombres d'objets. La densité d'une cellule est basée sur le rapport entre le nombre de points présents dans cette cellule et son volume. Ainsi, la relation de voisinage qui servait dans les méthodes à base de densité est remplacée par le voisinage entre les cellules, ce qui permet de réduire le nombre d'objets à regrouper.

Le processus de clustering dans les méthodes à base de grille consiste à regrouper les cellules denses les plus proches. Nous citons quelques algorithme de cette methode : L'algorithme bang et L'algorithme clique.

a.3 Méthodes probabilistes

Les méthodes probabilistes supposent que les données suivent une certaine loi de probabilité. L'objectif est d'estimer les paramètres de cette loi et de définir un modèle de mélange de lois pour représenter les différents clusters. Ces méthodes font l'hypothèse qu'à chaque cluster *Ci* est associée une loi de probabilité *P*(*x, Θi*) de paramètres *Θi* qui permet de déterminer la probabilité d'appartenance de *x* à *Ci*. Si on note *Θi* la proportion de la *i*-ème loi dans le mélange, les paramètres du modèle sont : Φ= Π1, . .. , *ΠK, Θ*1, . . . , *ΘK*) et la fonction de densité est :

$$P(x, \Phi) = \sum_{i=1}^{K} \pi_i P(x, \theta_i)$$

Á noter qu'en général, les lois considérées sont supposées gaussiennes.

a.4 Méthodes hiérarchiques

Les méthodes hiérarchiques construisent une hiérarchie de clusters, c'est-à-dire un arbre de clusters pouvant se présenter sous la forme d'un dendrogramme. Chaque nœud contient ses clusters enfants, et les nœuds frères partitionnent les objets contenus dans leurs parents. Ce type d'approche permet d'explorer les données à différents niveaux de granularité. Les méthodes de clustering hiérarchique sont décomposées en deux types d'approches, les approches ascendantes et les approches descendantes. Dans les approches ascendantes, l'algorithme part d'un grand nombre de clusters et ceux-ci sont ensuite fusionnés jusqu'à n'obtenir plus qu'un unique groupe contenant tous les objets du jeu de données. Les approches descendantes partent, de l'ensemble des données, et le divisent en clusters qui sont ensuite divisés récursivement.

a.5 Méthodes spectrales

Le coeur du clustering spectral consiste à considérer le problème de clustering comme un problème de coupe de graphe normalisée et à utiliser le Laplacien de la matrice d'adjacence du graphe qui représente alors les données. Une matrice de similarité est utilisée qui représente la similarité entre les couples d'objets. Ces algorithmes utilisent les vecteurs propres du Laplacien et sont de fait peu appropriés à de gros volumes de données en raison du coup important de son calcul. Cependant, il

suffit de déterminer une matrice de noyau pour l'appliquer, ce qui facilite le traitement de données hétérogènes (définition de différentes matrices de noyau).

13.3.2 Approche basée sur les règles métier

H.Sneed propose une approche pour la création des services Web à partir des Systèmes patrimoniaux basée sur la détection du code qui implémente la règle métiers. La règle métier est définie comme un algorithme est utilisé pour calculer un résultat donné. Les étapes de cette approche sont les suivantes :

a. La récupération du code de service

Cette étape est essentielle pou illustrer comment détecter le code qui implémente la règle métier. L'auteur utilise une approche qui détermine le résultat produit par la règle métier pour détecter le code qui l'implémente. Ceci est réalisé par l'identification des variables qui sont retournées par les fonctions implémentant la règle métier et aussi par l'identification des fonctions. La difficulté qui s'impose ici est qu'un bloc de code (fonction, subroutine, paragraphe,…etc.) peut contenir plusieurs règles métiers. Pour remédier ce problème, cette approche propose de faire une analyse de flux de donnée basée sur le résultat final, en retournant vers toutes les instructions qui ont contribué dans le calcul de ce résultat, en suite les blocs du code qui contient ces instructions seront recopiés à partir du code original avec les variables utilisées en formant un nouveau module séparé.

La portée pratique de cette phase est la documentation de la règle métier existante par un diagramme des flux des données. Cette documentation est utile pour décider si une règle existante mérite d'être réutiliser comme un service dans une SOA. Cette décision nécessite la compréhension totale de cette règle métier et aussi sa valeur économique. Les candidats principaux pour la réutilisation sont Les règles qui ont des valeurs économiques élevées.

b. Adaptation du Code

Le but de la phase d'adaptation est de fournir la discription WSDL du service extrait du Systèmes patrimonial. Pour accomplir la tâche d'adaptation Mr H. Sneed utilise un outil logiciel appelé SoftWrap. Cet outil a été développé essentiellement pour automatiser le processus de génération d'une interface WSDL pour un service écrit dans les langages (PL/I, COBOL ou C/C++), et aussi génère pour chaque service

deux module supplémentaire, un pour analyser le message entrant et extraire les données et l'autre pour créer le message de retour contenant les résultats produits par le service.

c. Liaison de Service Web au Code

Dans cette étape, Il a crée un nouveau composant appelé « Proxy » (procuration) dans la même adresse que la définition de processus. Ce Proxy vérifie les paramètres et génére l'interface WSDL qui est expédiée par le message de SOAP au serveur d'application. Sur le serveur d'application, il existe un planificateur « Scheduler », qui reçoit le message SOAP, détermine par quel Web service doit être exécuté et l'envoie l'interface WSDL. Après l'exécution du service, les résultats sont transformés par l'adaptateur en un document XML, qui va retourner vers le « Scheduler » pour le retransmit au client.

De cette façon, le processus d'affaires peut s'exécuté sur n'importe quel client (n'importe où), et il est capable d'accéder à toutes les fonctions d'un Systèmes patrimonial sur le serveur d'application [12].

18.3.3 Approche basée sur les ontologies (SOSR)

Zhuopeng et all [75, 76] proposent une méthode pour l'identification du code hérité réutilisables pour soutenir la réingénierie de logiciel orientée service (Service-Oriented Software Reengineering 'SOSR').Ils ont défini et développer des ontologies les quelles sont:

a. Ontologie de Concept du domaine (ACD) :

A pour objectif de fournir un ensemble de concepts, dans le domaine d'application, qui permettent la classification des différents types d'objets selon leurs structures internes, tels que les fichiers textes, des diagrammes, cartes, images, tableaux, documents.

b. Ontologie de fonctionnalités (FO) :

Vise à décrire les fonctionnalités de l'application et les relations de ces fonctionnalités de point de vue du monde réel. FO est principalement le résumé des exigences. Les fonctionnalités sont la capacité d'une application logicielle de satisfaire les exigences du domaine d'application.

c. Ontologie de composants logiciels (SCO) :

Comprend des composants qui résument la compréhension du code. SCO a pour but de décrire la conception de logiciel, de représenter des modèles de conception de logiciels et des concepts en liaison à l'aide des concepts développés dans cette ontologie. SCO définit principalement la structure du code et comprend également des objets au niveau schéma tels que des schémas conceptuel de base de données et de schémas fonctionnels.

Ces ontologies stockent les connaissances correspondantes au domaine d'application et aux entités du code, pour soutenir la poursuite de l'analyse. L'identification des services candidats dans les systèmes patrimoniaux est atteinte par le mapping de FO et SCO via une nouvelle méthode combinant l'analyse formelle des concepts (CAF) et l'Analyse des concepts relationnels (RCA).

Ces ontologies se concentrent sur la fermeture de l'écart conceptuel existant entre les modes des processus de réingénierie et les ressources. Ces ontologies peuvent être appliquées à de nombreux processus de réingénierie de logiciel, qui comprend l'analyse de code source, l'analyse de documents, l'analyse et la modélisation des processus de traçabilité.les étapes de cette approche sont les suivantes :

a. Evaluation du système patrimonial:

Une évaluation de l'héritage systèmes est effectuée pour confirmer l'applicabilité de cette approche SOSR et de déterminer d'autres activités de la réingénierie.

b. Analyse du domaine:

La technique d'analyse orientée service sera appliqué afin de déterminer quels sont les services nécessaires selon les besoins de l'entreprise. Si les fonctions de service nécessaire sont implémentées dans le système existant, ces fonctions seront identifiées et extraites comme des composants hérités.

c. Identification des Services candidats:

Premièrement, les techniques d'analyse statique et dynamique appliquées à cette étape et leurs résultats sont à la base des travaux de réingénierie et l'entrée de l'identification des services candidats. Deuxièmement, le système patrimonial est décomposé en composant candidats via des techniques d'analyse formelle de concept. Les résultats de cette étape sont des segments de code et des états clés, qui contiennent l'implémentation des fonctions de service.

L'identification des services candidats nécessite une décomposition du système en termes des principes suivants :

- la réutilisation de la fonctionnalité.
- sa valeur commerciale.
- le degré de granularité du service qui en résulte.
- son indépendance.

d. Extraction des composants hérités:

Basé sur les résultats de l'identification des services candidats, un programme statique des techniques de décomposition sont appliquées pour mieux comprendre ces composants candidats. L'objectif de cette étape est d'extraire les composants hérités exécutable, qui ont une haute cohésion et un faible couplage.

e. Intégration des Services:

Au cours du processus d'intégration des services, les composants existants et les nouvellement construits sont concaténés afin de construire le service cible.

14. Conclusion

Les entreprises ont aujourd'hui la nécessité de faire coopérer l'ensemble de leurs applications pour rester compétitives. Cependant, ces applications sont hétérogènes et celles-ci doivent interagir suivant des processus « métiers » bien définis. L'EAI fournit un cadre capable de prendre en compte la synchronisation de ces processus. Malgré ça, nous pouvons conclure que l'EAI ne présente pas une solution finale aux défis de l'intégration.

Les services Web sont la prochaine extension logique de l'EAI parce qu'ils standardisent la communication, la description et la découverte. Nous trouvons que les vendeurs de l'EAI ont étendu ou transformé leurs produits pour être conforme aux technologies récentes telles que les services Web en remplaçant les protocoles de communication par les normes de services Web.

Le chapitre suivant sera destiné à l'illustration le domaine des services Web et signalé leurs impactes sur le business des entreprises et le monde industriel en entier.

Les services web

1. Introduction

L'évolution permanente du marché marquée par l'arrivée de derniers canaux de ventes -Internet ou utilisateurs nomades- et l'évolution des exigences du client a montré les difficultés rencontrées dans l'intégration de nouvelles technologies dans les systèmes patrimoniaux des entreprises. Les entreprises font face à des nombreuses pressions

introduites par la personnalisation et l'amélioration qualitative des produits et des services et par la réduction du délai de mise sur le marché (time-to-market). A cet effet, les entreprises font évoluer en continu, leurs processus métiers. En conséquence, ces évolutions engendrent des coûts élevés de maintenance ou d'évolution des applications (basées sur des développements spécifiques ou sur des progiciels).

Une grande partie des dépenses concernant les outils informatiques est consommée dans la maintenance et le support de ces systèmes, laissant peu de marge pour l'innovation et l'investissement sur les nouvelles technologies. En plus, les problèmes manifestant ces dernières années devant les développeurs des systèmes d'information est quels sont les outils technologiques nécessaires pour offrir aux utilisateurs de ces systèmes un accès rapide, intégré et d'une manière standard aux informations pertinentes. Pour arriver à ce but plusieurs solutions sont proposées comme par exemple les objets CORBA de l'OMG et la programmation orientée composants, mais le problème reste toujours se blottit dans la standardisation.

Dernièrement, un nouveau paradigme est apparu comme la solution la plus adéquate des problèmes cités ci-dessus, c'est l'architecture orientée services (SOA) dont le but est de permettre la réalisation rapide et efficace de systèmes d'information distribués, en intégrant des applications existantes et nouvelles. Enfin, les services Web ont arrivé pour permettre les échanges des informations entre différents systèmes d'informations à travers le Web et en utilisant des standards tels que SOAP, UDDI, WSDL, HTTP...etc.

Un service web est une fonction métier autonome et sans état et est considéré comme une interface placée entre le code d'application et l'utilisateur de ce code.

Un service peut être invoqué par une ou plusieurs requêtes et retourne une ou plusieurs réponses à travers une interface standard. Les services implémentent des règles de business, des calculs et se chargent de l'exécution de transactions, etc. Ces services sont indépendants : ils ne dépendent pas de l'état des autres fonctions ou processus.

La technologie utilisée pour implémenter le service, tel que le langage de programmation, ne fait pas partie de la définition du service. Un service ne doit pas

dépendre d'une condition provenant d'un autre service . Il doit recevoir toute l'information dont il a besoin pour fournir une réponse à la requête qui a permis son activation. L'absence de dépendance entre services et consommateurs de ces services permet de les impliquer dans de nombreux flux réalisant la logique applicative des processus métiers [13], [14] .

Ce chapitre est consacré à l'état de l'art des services Web Donc, nous allons présenter d'abord la notion du Architecture Orientée Service (SOA), qui est considérée comme la meta classe des services Web, ensuite nous exposons quelques définitions tirés des littératures des services Web une comparaison entre POO et SOP. Dans la section suivante, nous introduisons les différents standards des services Web qui sont les clés de succès de ce paradigme, XML, SOAP, WSDL, UDDI. Ensuite, nous allons exposer la pile du protocole des services Web qui composée de six couches. Nous discutons un peut sur les effets des services Web sur le monde industriel et les systèmes d'information dans la section suivante. Après nous allons présenter le notion du l'orchestration ou la chorographie des services Web et à la fin de cette section nous prenons un exemple des systèmes de gestion des processus métiers qui est BPEL4WS (Business Processus Execution Language For Web Services), et en terminant par les avantages et les inconvénients des services Web, et une conclusion.

2. Architecture orientée service

Le terme SOA est évoqué mais la traduction de cet acronyme « Service-Oriented Architecture » par Architecture Orientée Service ne permet pas de comprendre exactement ce qu'il signifie. Une définition simple pourrait être « la notion d'intégrer et de manipuler les différents composants d'un système informatique en tant qu'ensembles fonctionnels appelés services » [13].

Une autre définition selon [18] « Une façon de concevoir et d'implémenter les applications de l'entreprise qui possèdent un couplage faible, sont à grandes mailles et sont des services réutilisables, accessibles par des interfaces bien définies et indépendantes de toute plateforme. » Cette définition n'inclut donc pas de protocole de communication obligatoire et il n'y a aucune restriction évoquée : pas d'obligation d'envoyer des messages SOAP en HTTP. SOA est beaucoup plus que seulement des Web Services et sa vraie force intervient réellement quand il n'y a plus ces restrictions et autorise les services à être invoqués par une multitude de protocoles.

La programmation orientée service (SOP) est la prochaine étape qui suive la programmation orientée objet (POO) et le développement basé sur les composants (D.B.C) [13] .Cette architecture à la mode répond aux problèmes d'intégration et de réutilisation d'outils (ou produits) des entreprises. Pour mieux comprendre sa définition, il faut voir cette architecture comme une philosophie. C'est une approche permettant de réutiliser et d'organiser des ressources existantes, dans une solution autorisant une interopérabilité entre plateformes et environnements, une évolutivité des modules applicatifs et une flexibilité autorisant l'utilisation dynamique d'applications. Cette solution permet donc d'intégrer divers systèmes :

- chaque ressource peut être accessible en tant que service possédant une interface.
- L'implémentation du fournisseur de service est donc libre de changer sans qu'il y ait un impact sur son utilisation.
- On peut voir ce service comme une boîte noire : on sait qu'elle va rendre le service voulu sans savoir comment est faite la boite noire.
- On peut choisir de la remplacer par un autre service implémenté différemment mais répondant aussi à la même fonctionnalité.

Une architecture orientée service consiste donc en une collection de services, indépendants les uns des autres, qui interagissent et communiquent entre eux. L'avantage premier de l'approche basée sur les services Web est l'adoption à une large échelle de ses technologies. Cela permet aux entreprises d'effectuer une transition incrémentale vers une architecture orientée service avec des risques minimaux et donc de limiter les coûts [13].

2.1 Comparaison entre SOA et OOA

Il est intéressant de comparer l'architecture SOA à une autre architecture qui met en avant des différences importantes. Cette architecture est l'architecture orientée objet. Dans cette architecture, les données manipulées sont directement associées au mode de traitement qui leur est appliqué. La programmation orientée objet (POO) implique deux choses : liaison forte à un modèle spécifique et un nombre d'appels important entre les deux couches de

présentation et métier (contenant les objets métiers).

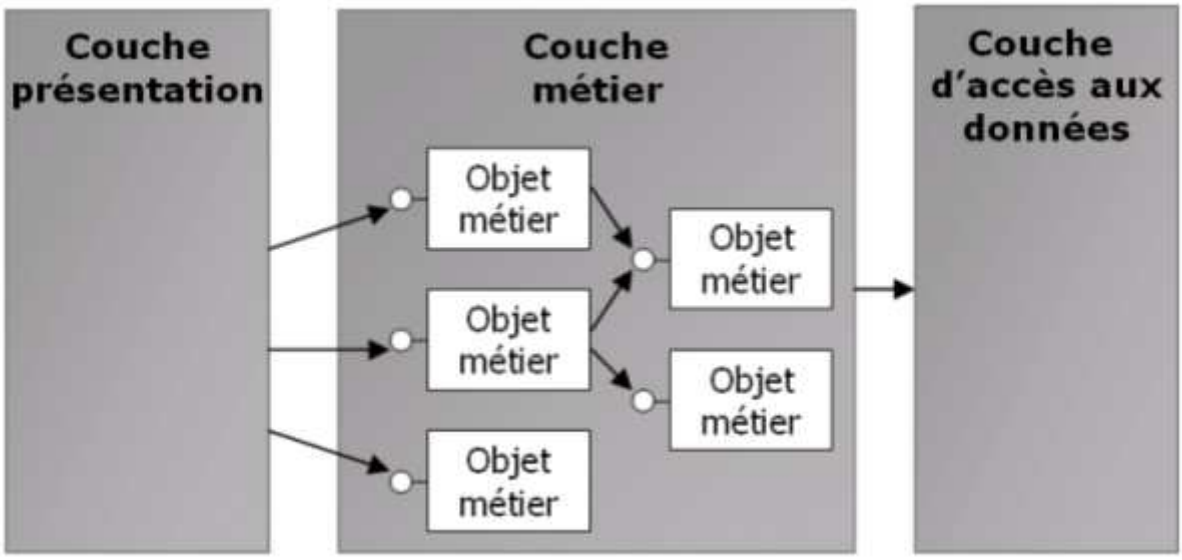


Figure II.1 Architecture orientée objet [13]

Au contraire, l'architecture SOA permet à la couche de présentation de passer par des services pour manipuler les objets métier sans avoir besoin de les connaître explicitement. Les services agissent ainsi en tant que boites noires faisant abstraction de la complexité du modèle objet.

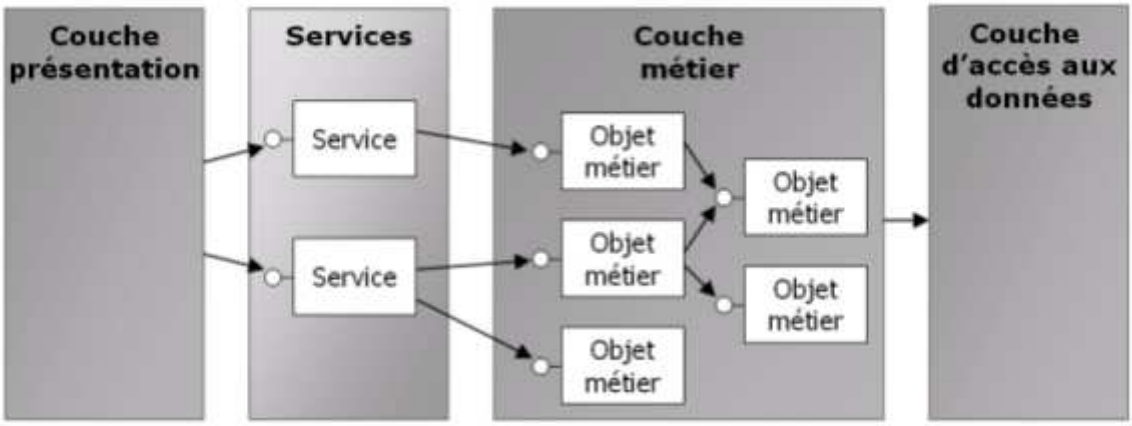


Figure II.2 Architecture Orientée Services [13]

Pour mettre en pratique la théorie (SOA), une implémentation s'est vite imposée : les Web Services. Même si ce n'est pas l'unique choix d'implémentation, elle est souvent associée à SOA.

2.2 Les avantages des architectures SOA

Les architectures orientées services offrent une flexibilité et une meilleure résilience, (notamment en termes de disponibilité) au système d'information. Leur rôle est de mutualiser la gestion des processus en permettant l'accès aux processus des applications internes, et en assurant l'interopérabilité entre les systèmes de façon indépendante des technologies sous-jacentes, grâce aux standards applicatifs des architectures orientées services [17]. Ces architectures apportent trois avantages majeurs :

a. Réactivité : la création des processus métiers interentreprises obtenu par la composition de services distribués permet d'accélérer la mise en œuvre d'une solution pour répondre à un nouveau besoin métier engendré par l'évolution des conditions de marché.

b. Evolutivité : la possibilité de mettre en place des processus métiers évolutifs qui sont recomposables selon les modifications des besoins de l'entreprise.

c. Flexibilité : mise en œuvre de nouveaux processus métiers à partir de services. Ces processus métier peuvent devenir à leur tour de nouveaux services utilisables pour construire de futurs processus métiers plus complexes.

En proposant une véritable transformation du système d'information , les architectures orientées services permettent à l'entreprise de diversifier ses canaux de vente et de distribution mais aussi ses sources d'approvisionnement indépendamment de la technologie de systèmes d'information patrimoniaux [17].

d. Intégration

Dans le passé, les données et la logique d'une application ne peuvent pas être utilisées dans une autre application, cette problématique donne la naissance à un nouveau domaine appelé EAI (Enterprise Application Intégration). Dans les solutions traditionnelles d'EAI, une application communique avec une autre application par un bus ou un hub qui enveloppe et traduit l'information dans une application selon des règles spécifiques fournies par l'application de réception. EAI est un grand pas en avant, permettant à des compagnies d'améliorer leur investissement dans ses applications critiques, plutôt que de réécrire l'application ou les émigrer vers une nouvelle technologie. Cependant, l'EAI traditionnel a présenté un ensemble de problèmes, qui sont : coûts de logiciel d'intégration, de la

complexité, de l'effort et de l'inflexibilité des solutions. Ces problèmes ont guidé les compagnies pour choisir des autres solutions.Parmi ces solution nous trouvons l'intégration des applications basée sur les services Web, qui attaque les trois problèmes identifié au-dessus avec un nouveau modèle qui est à prix réduit, plus facile à apprendre et se déployer, et plus adaptable aux business changeantes à besoin. Le résultat est l'intégration des applications plus rapide et plus facile, permettant à des compagnies de relier beaucoup plus des applications au sein de leur compagnie et ouvrant la porte pour des processus métiers plus efficaces au sein et à travers des entreprises [12].

2.3 Principe du SOA

Dans le modèle SOA, chaque service est défini par un fournisseur. Le fournisseur de services publie la description de son service dans des registres de services spécialement conçus en vue d'être interrogés par des clients.

Les clients de services (applications clientes) localisent leurs besoins en termes de services en effectuant des recherches sur le registre de services. Une fois le service localisé, le client récupère sa description du registre et sur la base des informations fournies dans la description du service, le client interagit (bind) alors avec le service en vue de l'exécuter [14].

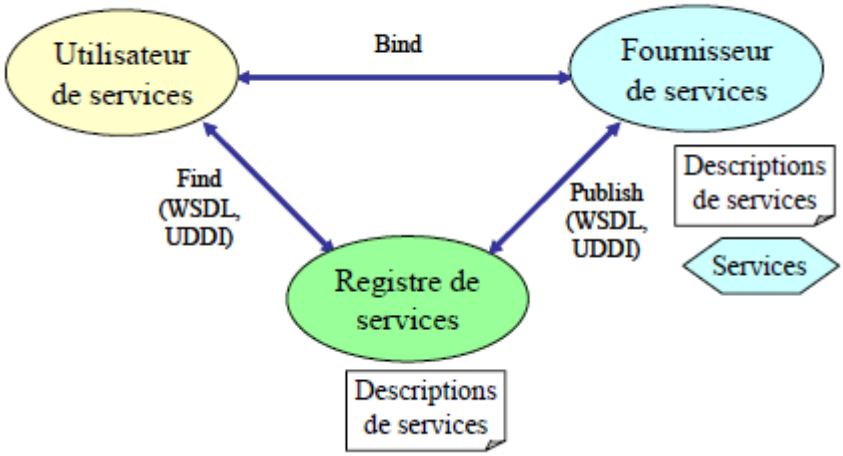


Figure II.3 : Principe de SOA [14]

3. Notion des services web

Les Web Services représentent l'implémentation la plus utilisée pour appliquer l'architecture SOA. Le concept de Web Service regroupe un ensemble de technologies basées sur XML, permettant de créer des composants logiciels distribués, de décrire leurs interfaces et de les utiliser en mettant en oeuvre des standards tels que SOAP, XSD, WSDL et UDDI. Les Web Services respectent donc des protocoles et des normes informatiques utilisés pour échanger des données entre les applications : ils ont été définis, pour la plus grande part, par le W3C (World Wide Web Consortium), qui a été fondé pour promouvoir la compatibilité des technologies du Web et émet des recommandations à valeur de standards industriels.

La notion de service logiciel selon [14] correspond à « une fonctionnalité offerte par un composant applicatif et dont l'interface est indépendante de la plate-forme ou de la technologie utilisée pour le développement du composant ». Les principales caractéristiques d'un service logiciel sont les suivantes:

- Il est auto-descriptif et permet la réutilisation dans le sens où il fournit une description de son interface;
- Il est autonome, complet et cohérent dans la mesure où le service contient toute sa logique d'exécution (self-contained) ce qui lui permet de s'exécuter indépendamment des autres services;
- Le service peut être localisé dans la mesure où il possède une adresse ce qui permet de l'invoquer dynamiquement

Selon [15] « Un service web est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet, par et pour des applications ou machines, sans intervention humaine, et en temps réel ».

Le groupe W3C [16] propose que : « Un service Web est un système logiciel conçu pour supporter l'interaction interopérable de machine à machine au-dessus d'un réseau. Il a une interface est décrit dans un format exploitable par machine, spécifiquement WSDL (Web Service Description Language). D'autres systèmes agissent avec le Service Web en quelque sorte exigé par sa description en utilisant des messages de SOAP (Simple Object Access Protocol), typiquement transmis en utilisant le HTTP (HyperText Transfer Protocol) avec une publication en XML et d'autres standards Web ».

4. Les standards

Une caractéristique majeure déjà évoquée des Services Web est qu'ils se basent sur des protocoles industriels standardisés. Ces derniers sont basés sur XML comme le standard SOAP (Simple Object Access Protocol) utilisé comme middleware pour le transport des messages, le standard WSDL (Web Service Description Language) utilisé comme langage pour la description des services et le standard UDDI (Universal Description, Discovery and Integration) utilisé pour le stockage et la découverte de services à travers le Web [Figure II.4], [14].

Un Web Service peut, dans un exemple simple, laisser un client communiquer avec lui en utilisant des messages **XML** qui répondent au standard **SOAP** (Simple Object ou Service Oriented Architecture Protocol). Il représente donc le format d'échange de données dont les protocoles primaires sont HTTP et HTTPS. On y trouve un en-tête et un corps. Dans l'en-tête, on peut faire référence à un fichier XSD (XML Schema Definition), permettant de définir les types des différentes données échangées. Dans le corps, les données indiquant la méthode utilisée et les bons paramètres sont présents.

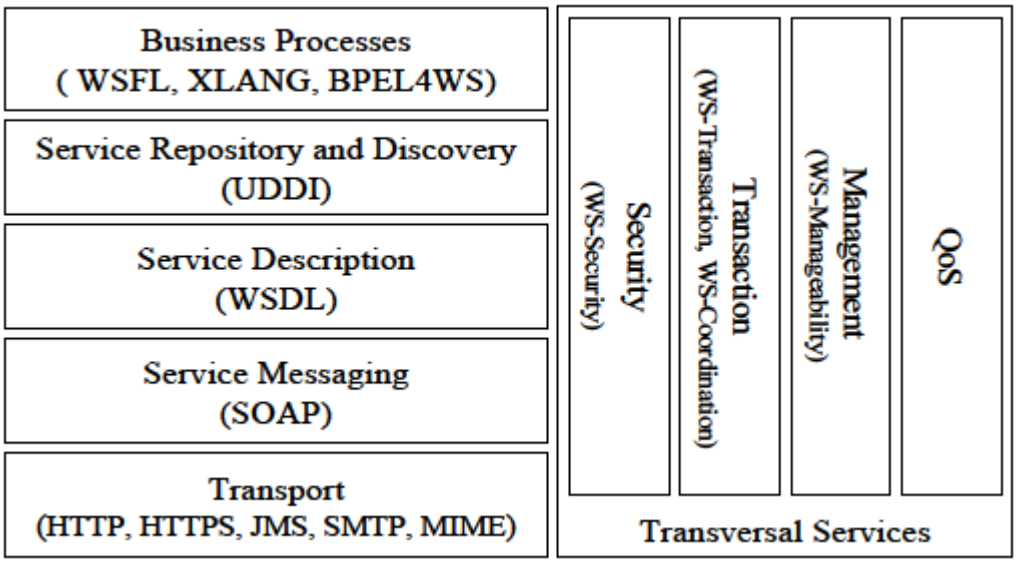


Figure II.4 : Principaux standards des Services Web [14]

4.1 XML (Extensible Markup Language)

XML (eXtensible Markup Language) a été développé, à l'origine, pour combler les lacunes du langage HTML, initialement conçu pour décrire les pages Web, et qui s'est très vite avéré limité dans la mesure où il ne distinguait pas les informations de présentation et de contenu d'un document. XML, dérivé du langage normalisé SGML (Standard Generalized Markup Language -ISO 8879), a été conçu comme un langage de structuration de données permettant de distinguer les informations de présentation et de structure d'un document, est basé sur l'utilisation de balises définissant un format universel de représentation des données [16].

XML comporte des standards associés facilitant soit la mise en œuvre d'XML soit le développement d'applications verticales. Dans la première classe, nous citons :

- Schéma XML pour la définition des structures de document,
- XLink (XML Linking Language) pour la gestion d'hyperlien entre les documents,
- XQuery pour les requêtes sur des collections des documents,
- DOM (Document Oject Model) et SAX (Simple API for XML) pour l'interface avec les langages de programmation,
- SOAP (Simple Object Access Protocol) pour l'invocation des services distants.

Les standards appartenant à la classe des applications verticales sont multiples. Il s'agit de langage particuliers crées en suivant les directives du métalangage XML. Nous citons par exemple :

- RDF (Resource Description Framework) pour la description de ressources Web.
- CXML (Commerce XML) pour la commerce électronique.
- XMI (XML Metadata Interchange) pour l'échange des modèles UML.
- DSML (Directory Service Markup Laguage) pour les annuaires.

Donc, selon la spécification du W3C, le langage XML décrit :

- Une classe d'objets de données appelés documents XML.
- le comportement des programmes qui les traitent [12].

Les services web communiquent en utilisant XML (interfaces et messages de XML interprétables facilement par les applications) pour décrire leurs interfaces et encoder leurs messages et en s'appuyant sur les protocoles standards de web comme les protocoles SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery and Integration), et WSDL (Web Services Description Language) pour réaliser l'interaction.

Ces standards utilisent également XML .De par son ouverture et sa flexibilité, XML est un langage aux possibilités d'évolution, d'expression et d'adaptation exceptionnelles [17].

4.2 SOAP (Simple Object Access Protocol)

SOAP est un standard qui représente une « enveloppe » légère contenant la charge utile "payload" des messages échangés entre les producteurs et consommateurs de services. C'est un standard basé sur XML qui décrit le contenu du message échangé ainsi que la façon de traiter ce contenu. Ce protocole permet en outre une association avec les protocoles de transport. SOAP offre aussi un groupe de règles d'encodage pour exprimer les instances des types de données définies par les applications et une convention pour représenter les invocations à distance et les réponses des applications. Voici les éléments principaux de ce standard [17] **:**

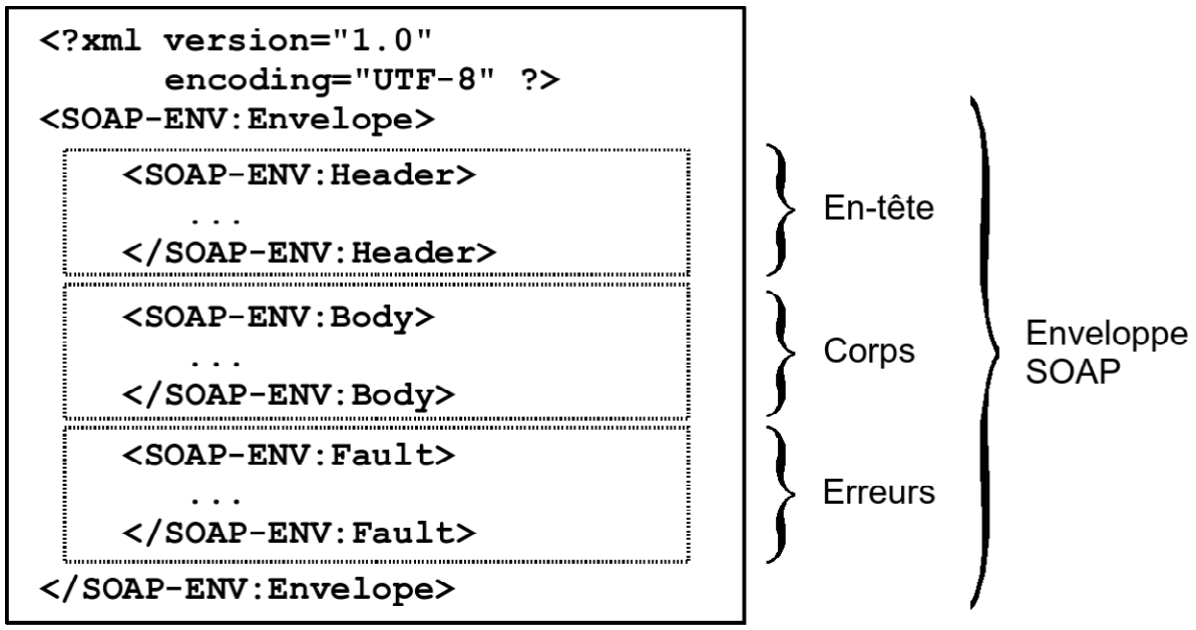


Figure II.5 : Format d'un message SOAP [71]

- L'enveloppe de SOAP (SOAP Envelop) contenant des informations sur le message lui-même afin de permettre son acheminement et son traitement. Il contient des détails supplémentaires ainsi que les informations liées à la sécurité ou à la destination finale du message [70].
- Un en-tête optionnel (**Header**) pour stocker les informations métiers spécifiques à La transaction (authentification, jeton d'autorisation, état de la transaction, ...).
- Un corps (**Body**) contenant les données à transporter,

- Une gestion d'erreur (**Fault**) qui identifie la condition d'erreur (si applicable).
- Et enfin des attachements optionnels (format **DIME**) [71].

4.2.1 Types de message SOAP

Le protocole SOAP a définit trois types de messages utilisés dans l'échange entre le fournisseur et le client du service Web :

a. Requête:

Ce message est obligatoire, émis par le client du service Web. Il est utilisé pour invoquer un service distant. Il contient les arguments d'appel c'est à dire les paramètres d'entrées de la fonction appelée. L'exemple suivant présente un appel à une opération 'echo'. Pour appeler l'opération "echo", on précise dans le Body SOAP de la requête l'identifiant de l'objet distant, l'opération à exécuter et les éventuels paramètres :

- <m:echo xmlns:m="urn:Echo"> : l'espace de nommage défini a pour URN l'identifiant du service (« urn :Echo »). Le nom de l'élément (echo) correspond au nom de la méthode à exécuter ;
- Les paramètres d'appel de la méthode sont ensuite ajoutés les uns à la suite des autres. Ici on n'a qu'un seul paramètre : « expression », de type string [18 ,12].

```
<SOAP-ENV : Body>
    <m:echo xmlns: m = "urn:Echo">
        <expression type= "xsd:string">
            Hello World
        </expression >
    </m:echo>
<SOAP-ENV : Body>
```

b. Réponse

Message optionnel, émis par le fournisseur du service. Il est crée dans le cas où il y a des informations de retours doit être renvoyées au demandeur de service. Un exemple présentant un message SOAP répondant sur l'appel de l'exemple précédent, est le suivant :

Après l'appel de méthode echo. Un message SOAP est alors envoyé au service SOAP. Le service exécute la méthode précisée dans la requête et retourne ensuite un message SOAP dont le Body contient le résultat de l'opération, le résultat est contenu dans un élément <return> :

- L'attribut type précise le type de retour de la méthode exécutée.
- Le contenu de l'élément est le résultat de l'exécution de la méthode [18 ,12].

```
<SOAP-ENV : Body>
    <m : echo  xmlns : m = "urn : Echo">
      <return type="string">
            message reçu : Hello World
      </return>
    </m:echo>
</SOAP-ENV : Body>
```

c. Erreur (Fault)

En cas d'erreur lors du traitement de la requête, le serveur renvoie un message SOAP donnant les raisons de l'erreur, dans un message HTTP dont le header commence par :HTTP/1.1 500 Server Error**.**

L'erreur est détaillée dans le Body SOAP, dans un élément Fault, donnant :

- Faultcode : le code de l'erreur, destiné à un traitement informatique ;
- Faultstring : une explication textuelle, à destination des opérateurs humains ;
- Faultactor (optionnel) : en cas d'erreur dans le transport, l'acteur mis en cause peut être précisé : firewall, serveur, proxy, etc;
- detail (optionnel) : un détail de l'erreur (par exemple en Java la trace de l'exception renvoyée).

Par exemple, dans le cas où la signature de la méthode de la requête ne correspond pas à signature de la méthode du service [18], [12] :

```
<SOAP-ENV : Body>
    <SOAP-ENV: Fault>
          <Faultcode> SOAP-ENV : Client </Faultcode>
          <Faultstring>
              La signature de la méthode ne correspond pas.
          </Faultstring>
    </SOAP-ENV : Fault>
</SOAP-ENV : Body>
```

4.3 WSDL

WSDL est un langage au format de type XML utilisé pour la description des services offerts sur un réseau. Un fichier WSDL (Web Services Description Language) décrit un web service : c'est la fameuse interface évoquée précédemment dans la description d'un service.

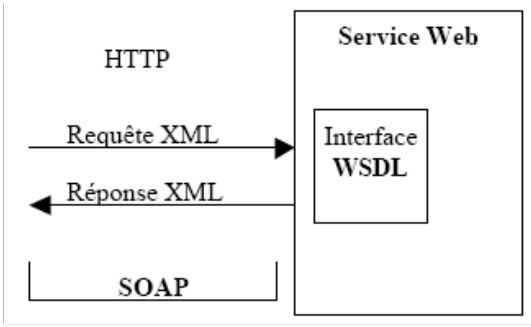


Figure II.6 Le Web Service et WSDL[18]

Pour faire un rapprochement, on peut voir ce fichier comme une interface en langage JAVA : on connaît les méthodes mais par leur implémentation. Elle contient aussi d'autre informations comme le fichier XSD (si jamais il y a des types complexes) mais surtout les informations permettant d'accéder au service (protocole, port et "EndPoint" - l'endroit où se trouve le service).

Ce langage décrit les interfaces des messages contenants des information orientées documents ou orientées procédures. Les opérations et les messages sont décrits de manière abstraite. Ces descriptions sont ensuite couplées au système de transport (spécifiant le format des messages et les protocoles réseau) pour définir les points de connexion aux services afin de définir une extrémité. Les messages sont associés au protocole SOAP et au protocole de transport HTTP.

La nature abstraite de WSDL (défini comme un outil de description pour les services) fournit la flexibilité nécessaire pour décrire des applications complexes des services web [17].

4.3.1 Structure d'un document WSDL

Un document WSDL contient une ou plusieurs définitions de Web Services reprenant, pour chacun d'entre eux, les différents composants (types de données, messages, types de port, liaisons et ports). Il est également possible d'utiliser la balise <Import> pour fragmenter et rassembler les documents WSDL. Comme pour SOAP, la définition de WSDL mentionne

des Namespaces de référence où se trouvent les définitions XML Schema des balises WSDL elles-mêmes. Le squelette général d'un document WSDL est illustré par le schéma suivant [17]:

```
<definitions>
        <message>
        …
        </message>
        <portType>
            <operation> …</operation>
            <operation> … </operation>
            …
        </portType>
        <binding>
        …
        </binding>
        <service>
            <port> … </port>
            <port> … </port>
        </service>
</definitions>
```

On utilise un ensemble de balises pour définir un service, et ces différents paramètres (entrée et de sortie).

- <**definitions**> : C'est la racine de tout document WSDL. Cet élément contient la définition du service. Cette balise peut contenir les attributs précisant le nom du service, et les Namespaces. Elle contient trois types d'éléments :
- <**message**> et <**portType**> : Ces éléments définissent les opérations offertes par le service, leurs paramètres d'entrée et de sortie, etc. En particulier, un <message> correspond à un paramètre d'entrée ou de sortie d'une <operation>. Un <portType> définit un ensemble d'opérations. Une <**operation**> définit un couple message entrée / message sortie. Par exemple, dans le monde Java, une opération est une méthode et un portType une interface.
- <**binding**> : Cet élément associe les <portType> à un protocole particulier. Les bindings possibles sont SOAP, CORBA ou DCOM. Actuellement seul SOAP est utilisé. Il est possible de définir un binding pour chaque protocole supporté.
- <**service**> : Cet élément précise les informations complémentaires nécessaires pour invoquer le service, et en particulier l'URI du destinataire. Un <service> est modélisé

comme une collection de ports, un <**port**> étant l'association d'un <binding> à un URI.

Chaque élément WSDL peut être documenté à l'aide de l'élément <documentation>. Cet élément contient des informations liées à la compréhension du document par les utilisateurs humains du service. Il est aussi possible de définir des types de données complexes dans une balise <types> juste avant la balise <message>.

Exemple :

L'exemple suivant est la description du service Echo défini par la classe java suivante :

```
public class Echo {
    public String echo( String message ) {
        return "message recu : " + message ;
        }
}
```

```
1. <definition
2. xmlns : xsd = "http://www.w3.org/1999/XMLSchema"
3. xmlns : soap = "http://schemas.xmlsoap.org/wsdl/soap/">
4.
5.          <message name = "echoInput">
6.                  <part name = "expression" type = "xsd:string"/>
7.                  </message>
8.
9.                  <message name = "echoOutput">
10.                 <part name = "expression" type = "xsd:string"/>
11.                 </message>
12.
13.                 <portType name = "EchoPortType">
14.             <operation name = "echo">
15.                     <input   message = "echoInput"/>
16.                     <output message = "echoOutput"/>
17.             </operation>
18.             </portType>
19.
20.             <binding  name = "EchoSoapBinding" type = "tns:EchoPortType">
21.         <soap:binding style = "document"
22.                 transport = "http://schemas.xmlsoap.org/soap/http"/>
23.             <operation name = "echo">
24.                 <soap:operation soapAction = "urn:ServiceEcho"/>
25.                     <input>
26.                         <soap:body use = "encoded"
27.                             encodingStyle =
28.                             "http://schemas.xmlsoap.org/soap/encoding/"/>
29.                     </input>
30.                     <output>
31.                         <soap:body use = "encoded"
32.                             encodingStyle =
33.                             "http://schemas.xmlsoap.org/soap/encoding/"/>
34.                     </output>
35.             </operation>
36.             </binding>
37.
38.         <service name = "EchoService">
39.                 <port name = "EchoSoap" binding = "tns:EchoSoapBinding">
40.                     <soap:address
41.                         location = "http://www.improve.fr/ServiceEcho"/>
42.                 </port>
43.         </service>
44.</definition>
```

1

2

3

4

Explication :

1 : La première partie du fichier définit les paramètres d'entrée et de sortie des opérations du service : echoInput et echoOutput.

2 : la définition abstraite du Web Service est faite par définition du **portType**, qui encapsule la définition de l'opération echo. On fait ici référence aux messages définis précédemment (paramètres d'entrée et de sortie de l'opération). On obtient une description abstraite du service, indépendante de tout protocole de communication. C'est l'interface du service définissant les méthodes exportées, et leurs paramètres d'entrée et de sortie.

3 : Il est ensuite possible d'associer ce service à un protocole existant, par la définition d'un **binding** (pour l'exemple qui nous intéresse : SOAP). Le binding définit les paramètres d'invocation du service spécifiques au protocole utilisé. On définit ici les paramètres nécessaires à l'utilisation du service via SOAP (lien vers les spécifications du transport utilisé, règles d'encodage pour la sérialisation des messages échangés, etc.).

4: La définition du service se termine par la définition des paramètres restants. Par exemple pour un binding SOAP, il reste à définir l'adresse URL du service à invoquer. Notons qu'il est possible de définir plusieurs bindings, et d'associer ces bindings à plusieurs URL, et ce en utilisant la même définition abstraite du service. Ce service peut être représenté avec le schéma UML suivant (sauf que ce dernier ne permet pas de donner des détails sur les protocoles utilisés et le point d'accès du service) .

4.3.2 Les limites de WSDL

Indispensable pour connaître les informations essentielles à l'interaction entre un client et un service, la description WSDL montre vite ses limites [69] :

a. Uniquement un langage d'interface

La seule chose que connaît un utilisateur d'un service est donc cette description WSDL. On y trouve toutes les informations décrivant ses différentes opérations : nom des opérations, les messages à envoyer, les messages à réceptionner et les types des différents messages. Mais, aucune description n'est faite concernant le comportement même du service, qui permettrait de répondre aux questions :

- Doit-on appeler l'opération "authentification" avant l'opération "effectuer un virement" ?
- Peut-on utiliser indifféremment l'opération "annuler" et l'opération "terminer" ?

➢ Bien évidement, on peut trouver une solution à ce problème en adoptant un système de nommage évitant les ambiguïtés, mais ceci ne fonctionne que dans le cadre d'une utilisation par un programmeur de cette description. Donc, pour permettre la réutilisation des services dans le cade d'un processus métier composable, des informations supplémentaires sont indispensables ! Encore plus dans le cadre d'une composition automatique (ou semi-automatique) de services.

b. Interaction sans mémoire

De même, une fois le problème précédent énoncé, la gestion de l'état côté serveur n'est pas présente : en effet, à aucun moment, le service n'explique s'il conserve les

informations issues d'une précédente opération, ou s'il a besoin de faire une comparaison entre deux valeurs. On se retrouve donc bien dans un modèle d'interaction sans mémoire.

4.4 UDDI

Un registre UDDI Universal Decription Discovery and Integration) est un annuaire basé sur XML qui permet de publier des services et facilite leur découverte par d'autres services en définissant comment ils interagissent. Un scénario d'utilisation possible est donc la publication d'un fournisseur de service (donc de son WSDL) auprès d'un registre en créant tout d'abord une entreprise et une catégorie de service. Un client demande ensuite à un registre UDDI la localisation d'un service qui correspond au service venant d'être ajouté à l'annuaire. Le WSDL du service demandé est alors reçu par le client qui peut communiquer avec le fournisseur de services en SOAP.

1- Service fournit le WSDL.

2 - Le Client veut un service et reçoit son WSDL.

3- Le Client peut communiquer avec le service.

UDDI se comporte lui-même comme un Web Service dont les méthodes sont appelées via le protocole SOAP. UDDI est composé de deux parties :

- l'UDDI Business Registry, annuaire d'entreprise et de Web Services.
- les interfaces d'accès à ces annuaires, et les modèles de données.

L'objectif de standardisation UDDI a pour fin de construire un annuaire global qui soit pour les services Web et les entreprises qui les offrent ce que les noms de domaine et les adresses IP sont aux sites Web. Cet annuaire universel (UDDI Business Registry) rassemble l'équivalent des "pages blanches" (adresses, points de contact, identités des services), des "pages jaunes" (classement des services selon des taxinomies standardisées) et des "pages vertes ou roses" donnant des informations complémentaires et le mode d'emploi de l'annuaire.

4.4.1 Recherche de services Web avec UDDI

La recherche et la sélection dans UDDI reposent sur la publication préalablement décrite du service et de son fournisseur. Le futur client peut connaître par l'intermédiaire de UDDI : Les fournisseurs d'un service, les services proposés par un fournisseur donné, les moyens d'invoquer un service. Pour apporter aux clients la réponse à ces questions, UDDI

organise l'ensemble des informations qu'il contient en trois parties, spécifiées en XML. Chacune d'elles peut être utilisée pour faire une recherche via UDDI. Ces parties sont illustrées dans le tableau suivant:

Tableau II.1 Les différents types d'utilisation d'un UDDI

Information	**Utilité**
Page blanches : Cette description contient toutes les informations jugées pertinentes pour identifier l'organisation: nom, prénom, téléphone…	Permet de connaître les informations à propos de l'organisation proposant le service.
Page jaunes : Décrivent la catégorie de l'entreprise, le secteur d'activité dans lequel exerce l'entreprise, les services offerts par cette organisation, le type de services et les conventions d'utilisation : prix, qualité de service, etc	- Détaillent la description de l'organisation faite dans les pages blanches en répertoriant les services proposés - permet de chercher et trouver un web service particulier
Page vertes : description technique des différents web services offerts par l'entreprise	permet de comprendre comment une application peut se connecter et interagir avec le web service trouvé.

4.4.2 Rôles d'UDDI

UDDI fournit trois services de bases :

- **Publish :** ce service gère comment le fournisseur de service web s'enregistre lui-même ainsi que ses services (en utilisant UDDI).
- **Find :** ce service gère comment un client peut localiser le service web désiré (cela peut passer par des invocations de services web pour une utilisation automatique par un programme ou par une consultation d'annuaire en utilisant des mots clés).
- **Bind :** ce service gère également comment un client peut se connecter et utiliser le service web une fois celui-ci localisé [69].

4.5 Avantages et inconvénients des standards XML/SOAP/WSDL/UDDI

Le tableau suivant représente les différents avantages et inconvénients des standards cités au dessus :

Tableau II.2 les avantages et les inconvénients de XML/SOAP/WSDL/UDDI

Standard	Avantages	Inconvénients
XML	- **un format standardisé** ouvert ne nécessitant aucune licence, intégralement basé texte et qui peut être associé à n'importe quel jeu de caractères. - **Fichier texte** : C'est un fichier texte, donc il sera toujours lisible dans des décennies. On garantit ainsi une meilleure pérennité de l'information - **Strict** : peut être validé par des règles strictes, contenues par des DTD ou des Schémas, décrivant sa structure et la hiérarchisation de ses données. - **Structuré et hiérarchique**: Le fichier contient des <BALISES> qui peuvent contenir d'autres balises et ainsi de suite (hiérarchie). L'ordre d'apparition des balises est conservé. - On peut ajouter des commentaires afin de garantir une meilleure pérennité de l'information	-**Verbeux** : Les données stockées au format texte sont en général plus volumineuses que celles stockées au format binaire. - impossible d'utiliser du XML pur pour créer les pages d'un site sans l'associer à d'autres langages tel que HTML, CSS…
SOAP	- Utiliser SOAP via HTTP facilite la communication et évite les problèmes de proxys et pare-feu par rapport à des technologies plus anciennes - Assez ouvert pour s'adapter à différents protocoles de transport. - Indépendant de la plate-forme. - Indépendant du langage programmation. - Extensible	De part le nombre d'informations qu'impose le format XML, SOAP peut être considérablement plus lent que les technologies middleware concurrentes tels que CORBA.
WSDL	Décrire les interfaces d'entrée et sortie de façon abstraite et indépendante de tout - langage de programmation - plate forme	Aucune description n'est faite concernant le comportement même du service
UDDI	**registres privés** : de nombreuses organisations utilisent les spécifications de UDDI afin d'implémenter leur propre registre de services Web.	- il n'existe pas de plate-forme d'édition. - les API de UDDI sont insuffisantes pour développer efficacement des méthodes de publication et de recherche.

		-le modèle de recherche de UDDI est pauvre (recherche portant sur l'identifiant, le nom du service, ou sur les éléments du document WSDL).

5. Types de composition de services Web

La plupart des travaux portant sur la composition de services Web reconnaissent deux types de composition : l'orchestration et la chorégraphie de services. Cependant, selon les travaux, les définitions des types de composition diffèrent.

Pour Peltz [59] et Benatallah et al [58], l'orchestration et la chorégraphie sont des moyens de concevoir la composition, tandis que dans Barros et al [57], l'orchestration et la chorégraphie sont des points de vue de la composition de services.

Afin de choisir l'un ou l'autre de ces types de composition (orchestration ou chorégraphie), le concepteur de systèmes doit prendre en compte différents paramètres.

5.1 Orchestration

Pour type de composition, il existe plusieurs définitions parmi les quelles nous citons :

Barros et al définissent : « l'orchestration comme un ensemble de processus exécutés dans un ordre prédéfini afin de répondre à un but ». Ce type de composition permet de centraliser l'invocation des services Web composants. Chaque service est décrit en termes d'actions internes. Les contrats entre deux services sont constitués selon le processus à exécuter.

À l'instar de Barros et al [57], Benatallah et al [58] définissent l'orchestration comme un processus exécutable. Benatallah et al ajoutent que l'orchestration est un ensemble d'actions à réaliser par l'intermédiaire de services Web.

Un moteur d'exécution, un service Web jouant le rôle de chef d'orchestre, gère l'enchaînement des services Web par une logique de contrôle. Pour concevoir une orchestration de services Web, il faut décrire les interactions entre le moteur d'exécution et les services Web. Ces interactions correspondent aux appels, effectués par le moteur, d'action(s) proposée(s) par les services Web composants.

D'après Peltz [59], l'orchestration de services Web consiste en « la programmation d'un moteur qui appelle un ensemble de services Web selon un processus prédéfini ».

Ce moteur définit le processus dans son ensemble et appelle les services Web (tant internes qu'externes à l'organisation) selon l'ordre des tâches d'exécution. La [Figure II.8] illustre l'exécution du moteur permise par l'enchaînement de l'exécution de deux autres services Web. Cet enchaînement est possible via un opérateur d'ordonnancement (représenté par le losange dans la figure). L'exécution de la composition repose sur l'appel du Service Web 1, puis sur l'appel du Service Web 2, réalisés tous deux par le Service Web Moteur.

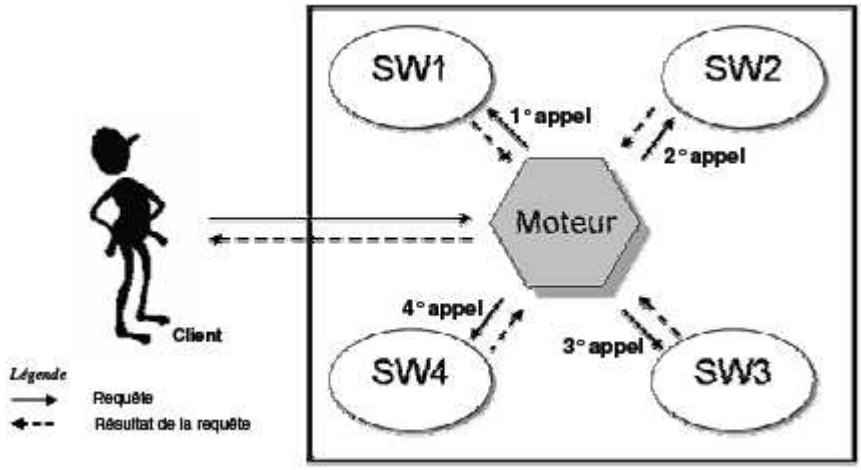


Figure II.7 Vue générale de l'orchestration [48]

En d'autres termes, l'orchestration de services Web exige de définir l'enchaînement des services web selon un canevas prédéfini, et de les exécuter selon un script d'orchestration. Ces derniers (le canevas et le script) décrivent les interactions entre services Web en identifiant les messages, et en spécifiant la logique et les séquences d'invocation. Le module exécutant le script d'orchestration de services Web est appelé un moteur d'orchestration. Ce moteur d'orchestration est une entité logicielle qui joue le rôle d'intermédiaire entre les services en les appelants suivant le script d'orchestration [48].

5.2 Chorégraphie

Ce type de composition de son tour a plusieurs définitions parmi les quelles nous citons :

D'après Barros et al [57] la chorégraphie permet de décrire la composition comme un moyen d'atteindre un but commun en utilisant un ensemble de services Web. La collaboration entre chaque service Web de la collection (faisant partie de la composition) est décrite par des flots de contrôle. Le fait que la chorégraphie mette en œuvre un ensemble de services Web afin d'accomplir un but commun apparaît aussi dans les travaux de Benatallah et al [58]. Pour concevoir une chorégraphie, les

interactions entre les différents services doivent être décrites. La logique de contrôle est supervisée par chacun des services intervenant dans la composition. L'exécution du processus est alors distribuée.

D'après Peltz [59], la description de chaque service Web intervenant dans la chorégraphie inclut la description de sa participation dans le processus. De ce fait, ces services peuvent collaborer à l'aide de messages échangés afin de savoir si tel ou tel service peut aider dans l'exécution de la requête. Chaque service Web peut communiquer avec un autre service Web par l'intermédiaire d'échange de messages [48].

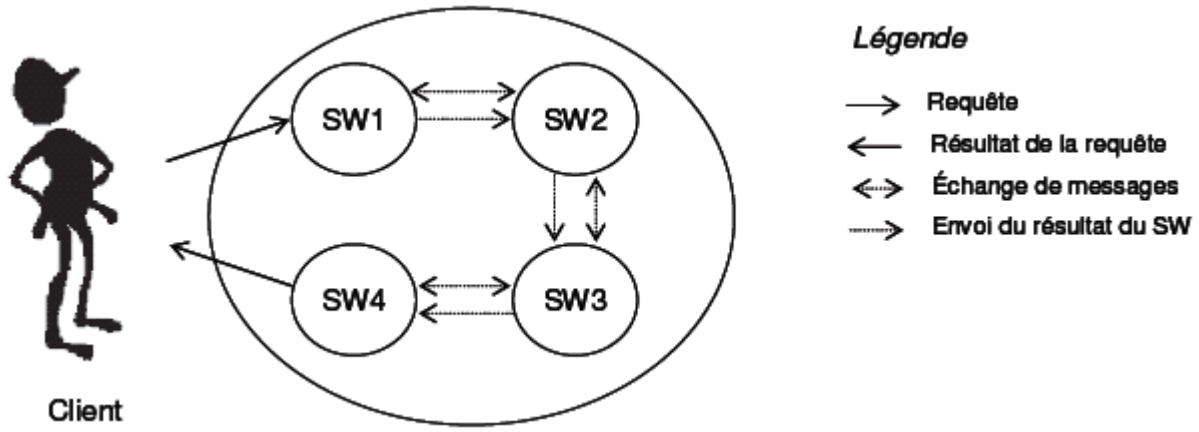


Figure II.8 vue générale de l'exécution d'une composition de service web de type chorégraphie [48]

6. Langage de composition de service web : BPEL4WS

BPEL4WS est le premier langage de composition de services Web adopté par la communauté des services Web. Ceci est principalement dû au fait que ce langage possède une grande expressivité dans la définition du processus exécutable [60]. Le but de ce langage est de remplacer les langages WSFL et XLANG de Microsoft pour l'orchestration des services Web et la gestion des processus d'affaires (Bussiness Process : processus

La première version du langage d'exécution des processus d'affaires (BPEL : Business Process Execution Language) a été crée par IBM, Microsoft et BEA en juillet 2002. La standardisation de langage BPEL4WS est effectuée en Mai 2003, par l'organisation de standardisation OASIS, et cette fois-ci avec l'association du Seibel et SAP [15],[16].

Le Langage BPEL est de format XML comme tous les autres langages et les spécifications reliées par les services Web. Et aussi, BPEL influx sur d'autres standards des services Web telles que WSDL et SOAP pour la description d'interface et le protocole de communication. BPEL décrit les interfaces des processus par des documents WSDL, ce qui permet à des clients d'un processus de contrôler et appeler un processus de BPEL juste comme n'importe quel autre service Web.

Le processus métier supporté par BPEL4WS peut spécifier la composition d'un ensemble de services Web, définir les données partagées entre ces services, spécifier les partenaires impliqués et le rôle qu'ils jouent dans le processus, et le gestionnaire commun de compensation de l'ensemble de services Web ...etc.

L'inconvénient principal de BPEL4WS est que la

- définition du processus est rigide. Si une activité du processus échoue, le processus dans son intégralité échoue. Aucun retour en arrière et aucune alternative au processus ne sont possibles.
- lors de la description du processus exécutable, la définition des flots de données ne permet pas de connecter des services dont les entrées/sorties ne correspondent pas exactement.
- BPEL4WS ne prévoit pas de mécanisme de transformation de données.

Les motivations derrière l'orchestration des services Web avec BPEL sont :

- ✓ l'augmentation de la productivité.
- ✓ La réduction des dépenses.
- ✓ L'amélioration des niveaux de service par l'automatisation de processus

en utilisant des technologies standardisées [12], [48].

7. Avantages et Inconvénients des services web

Malgré toutes les ambitions apportées par les services Web. Ces derniers ont aussi leurs avantages et inconvénients. Parmi ces avantages nous citons [19]:

1. les services pouvant interagir indépendamment de leur environnement technologique d'origine.
2. Ils offrent un procédé de "couplage lâche", expression souvent utilisée pour décrire un procédé d'intégration peu intrusif pour les applications impliquées.
3. Les services Web consistent en un semble assez simple de propositions.
4. Ils considèrent le Web comme environnement de développement.

5. Les services Web basée sur les standards de W3C.
6. Ils sont très adaptés aux problèmes des communications entre applications Web.

Malgré ces avantages, les services Web se heurtent aux inconvénients suivants :

1. IL reste d'autres problèmes qui ne sont pas réglés par les trois outils (SOAP, WSDL et UDDI), comme la sémantique de dialogue entre les applications.
2. Les services Web s'endurent des problèmes de performances.
3. La sémantique de requête / réponse utilisée par les services web est inefficace.

8.Vers le Web Sémantique

Depuis plusieurs années, une nouvelle idée du Web a vue le jour : celle d'un Web Sémantique. Ce dernier est une extension du Web actuel donnant un sens au contenu, proposé initialement par le W3C, est d'abord une nouvelle infrastructure permet d'aider plus efficacement différents types d'utilisateurs dans leur accès aux ressources sur le Web (sources d'information et services). Il s'appuie entre autres sur des ontologies et des langages, pour attribuer et représenter le sens de ses ressources, et ce afin de permettre à des programmes et à des agents d'y accéder à l'aide de langages développés par le W3C.

Peu à peu, le World Wide Web Consortium (W3C) se dote de nouveaux langages et d'outils plus performants : XML, RDF, n'en sont que des exemples. Tous ont pour objectif commun de participer à une formalisation des savoirs, en permettant de mieux exploiter, combiner et raisonner sur les contenus des ressources.

On présentera donc, dans une première partie, la sémantique, les principes de fondement d'ontologie ainsi que ses outils de génération. La deuxième partie, qui constitue l'annotation sémantique et ses différents modèles. Enfin, nous clôturons ce chapitre par une conclusion.

8.1 Les objectifs du web sémantique

- Générer des données sémantiques a partir de la saisie d'information par les utilisateurs.
- Agréger des données sémantiques à fin d'être publiées ou traitées.
- Publier des données sémantiques avec une mise en forme personnalisée ou spécialisée.
- Echanger automatiquement des données en fonction de leurs relations sémantiques.

8.2 Architecture du Web sémantique

L'architecture du Web sémantique [Figure II.9] repose sur une hiérarchie des langages d'assertion et de description d'ontologies ainsi que sur un ensemble de services pour l'accès aux ressources au moyen de leurs références sémantiques, pour gérer l'évolution et le versionnage des ontologies, pour l'utilisation des moteurs d'inférences capables d'effectuer des raisonnements complexes ainsi que des services pour la vérification de la validité sémantique de ces raisonnements [94].

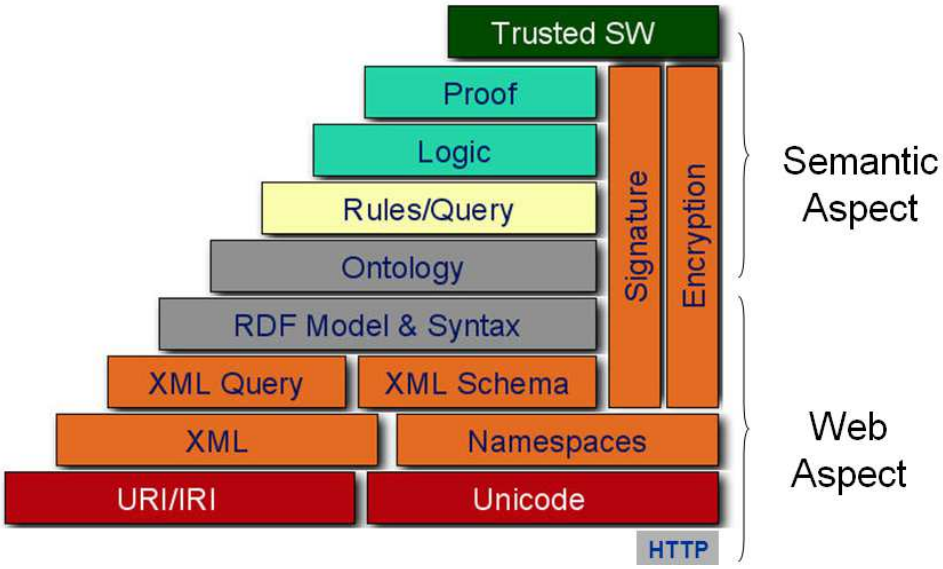


Figure II.9 Architecture du Web sémantique adapté de Berners-Lee [92] et d'Oberle et al [93]

Le W3C (World Wide Web Consortium) propose une hiérarchie des langages pour le Web sémantique dont seulement les couches inférieures sont stabilisées à l'heure actuelle (Berners-Lee, 2000). Cette architecture repose sur :

- URI, (*Uniform Resource Identifier*) qui permet l'attribution d'un identifiant unique à des ressources. XML (e*Extensible Markup Language*) est le langage de base qui procure une syntaxe aux documents structurés, mais il ne dispose d'aucune sémantique permettant de décrire la signification de ces documents.
- Le XML peut être vu comme la couche de transport syntaxique du Web sémantique, tous les autres langages étant exprimables et échangeables dans la syntaxe XML.
- Le langage RDF (Resource Description Framework) est un langage formel ayant une sémantique simplifiée permettant d'exprimer des relations entre les ressources du Web en utilisant des triplets de la forme (sujet-prédicat-objet).

9. Notion de sémantique :

9.1 Définition

L'origine du terme sémantique vient du grec semantikos "qui signifie". De nos jours, la sémantique constitue une branche de la linguistique qui s'occupe de l'étude des sens des termes. La sémantique est généralement utilisée pour compléter la syntaxe qui constitue un élément nécessaire pour pouvoir parler de sémantique. La syntaxe est à l'origine une branche de la linguistique qui étudie la façon dont les mots (morphèmes libres) se combinent pour former des syntagmes (nominaux ou verbaux) pouvant mener à des propositions (phrases), lesquelles peuvent se combiner à leur tour pour former des agrégats plus complexes, - les énoncés (textes).

Dans le domaine des systèmes d'information, la sémantique réfère plus précisément au sens des différents éléments d'un Système d'Information qui peuvent être des données, des fonctions, voire des processus. La difficulté liée à la sémantique des éléments d'un système d'information tient au fait qu'elle peut différer d'un système à un autre, en fonction du contexte d'utilisation. Ces hétérogénéités de type sémantique sont à l' origine des conflits entre applications se qui contraint leurs intégration [14].

9.2 Conflits et hétérogénéités sémantiques

Les conflits sémantiques surviennent lorsque, par exemple, l'information change de sens en fonction du contexte. Dans le cadre de conflits sémantiques, et plus précisément de conflits sémantiques de données, Goh [25] identifie trois catégories principales d'hétérogénéités sémantiques:

- Des conflits de confusion : où la donnée apparaît avoir le même sens alors qu'il n'en est rien.
- Des conflits de mesure : quand différents systèmes mesurent la même valeur.
- Des conflits de nomination : quand les noms de schémas diffèrent significativement du fait de la présence d'homonymes ou de synonymes.

Ces conflits sémantiques ne concernent pas seulement les données d'une application, mais peuvent aussi concerner la logique métier des applications et qui est aussi appelée hétérogénéité d'invocation selon Bussler [26].

Kavouras [27] précise que l'hétérogénéité sémantique est plus complexe que les autres types d'hétérogénéité : syntaxique et technique. Il propose plusieurs causes qui peuvent générer ces hétérogénéités sémantiques et qu'il résume dans deux points, à savoir

- Différence de couverture (niveau de détail) .
- Différence de classification (conceptualisation).

De plus, en considérant que les entités de la réalité sont caractérisées par un terme Ti et une définition Di, il propose une typologie d'hétérogénéités sémantiques découlant du croisement de situations portant d'une part sur les situations existantes entre les termes (T1, T2) et d'autre part sur les situations existantes entre les définitions (D1, D2) associées aux entités décrites (figure III.1).

La [figure4.1] illustre les notions suivantes :

- équivalence : deux entités ont le même terme et la même définition.
- Disjonction : deux entités ont des termes différents et des définitions différentes.
- Synonymie : cas où deux entités sont représentées par deux termes différents et une même définition.
- Homonymie : la situation dans laquelle deux entités sont représentées par le même terme et deux définitions différentes.
- Chevauchement : correspond à des situations dans lesquelles les définitions des deux entités se recouvrent.
- Complétude : cas où deux entités sont représentées par le même terme et la définition de l'une est incluse dans l'autre.
- Spécialisation : cas où deux entités sont représentées par deux termes différents et la définition de l'une est incluse dans l'autre.

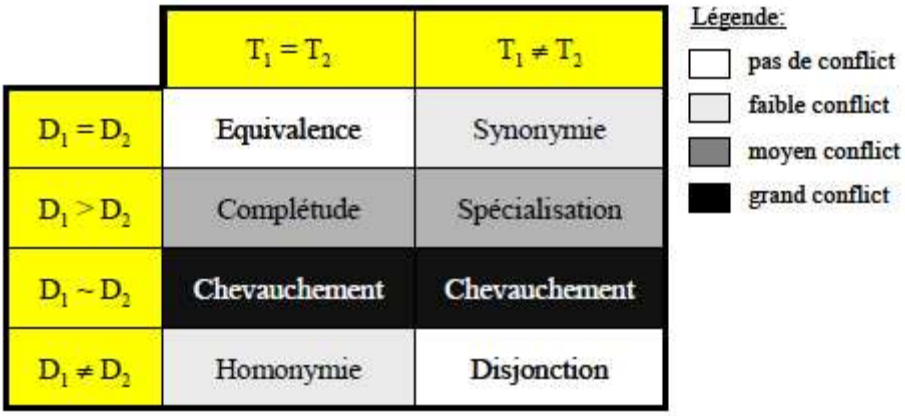


Figure II.10 Typologie des conflits sémantiques, kavoura [27]

9.3 Classification de la sémantique

Uschold et Gruninger [28] proposent une classification de la sémantique qui permet de distinguer les différents types de sémantique qui peuvent exister dans le contexte du web comme suit :

- **La sémantique implicite** : qui existe seulement dans le mental des gens.
- **La sémantique semi-informelle (explicite et informelle)** : qui est une sémantique explicite mais qui est souvent représentée de façon informelle en utilisant en général des langages naturels tels que le français ou l'anglais.
- **La sémantique semi-formelle** : qui désigne une sémantique explicite et relativement formelle qui est destinée principalement aux humains en utilisant des formalismes, le plus souvent graphique tels que les modèles sémantique, ou les diagrammes UML.
- **La sémantique formelle** : qui est une sémantique qui se base sur des formalismes mathématiques rigoureux (tels que la logique de description, la logique de premier ordre, etc .) qui lui permettent d'être traitée de façon automatique.

9.4 Représentation de la sémantique

Il est nécessaire de représenter la sémantique a l'aide des techniques de représentation de la connaissance. La représentation des connaissances étudie comment transformer l'expression du sens en une représentation formelle manipulable par une machine. Pour modéliser la sémantique nous pouvons utiliser l'une des deux approches à savoir :

- L'approche procédurale qui utilise des procédures ou des règles pour représenter la sémantique.

- L'approche déclarative qui se base sur la modélisation des faits.

Cette dernière est la meilleur du fait qu'elle présente des avantages tels que standardisation, capture, réutilisation, inférence et flexibilité.

L'un des outils les plus utilisés dans le cadre de représentation sémantique est la notion d'ontologie que nous détaillons dans ce qui suit.

10. Notion d'ontologie

10.1 Définition

Les ontologies sont apparues au début des années 90 à la suite des démarches d'acquisition des connaissances pour les systèmes à base des connaissances (SBC). Ces démarches proposaient de dégager les objets du domaine, leur signification et leur contenu, des objets du raisonnement décrivant les règles heuristiques d'utilisation des objets du domaine, le but étant de faciliter la construction des SBC en permettant la réutilisation des composants génériques (Van Heijst, Schreiber, et Wielinga, 1997).il existe plusieurs définitions de la notion d'ontologie nous retenons quelques unes[29]

Selon Pokarev et al [33] L'ontologie est :

« Une composante de la mémoire d'entreprise qui capture une connaissance potentiellement intéressante en elle-même pour l'entreprise ».

Selon Swartout [34] **:**

« une ontologie est un ensemble de termes structurés de façon hiérarchique, conçu a fin de décrire un domaine et qui peut servir de charpente à une base de connaissances »

En reprenant les définitions de Borst [30], de Gruber [31] et celle de Guarino et Giaretta [32], Rogozan [29] considère l'ontologie comme étant la spécification, plus ou moins formelle, rendant compte partiellement, mais explicitement d'une conceptualisation partagée, c'est-à-dire acceptée à l'intérieur d'une communauté. Cette conceptualisation est une théorie logique permettant la représentation d'un domaine de discours en terme de:

- concepts organisés taxonomiquement à l'aide des relations de subsumption spécifiant des liens de généralisation qui permettent l'héritage entre les concepts.
- relations représentant des types d'interactions entre les concepts.
- axiomes explicitant des énoncés conceptuels toujours vrais dans le contexte de l'ontologie et utilisés pour contrôler la signification des concepts ou des relations.
- instances, utilisées pour représenter des entités concrètes du domaine.

10.2 Les composants d'ontologie

Partant des définitions précédentes, les connaissances intégrées dans les ontologies peuvent être formalisées en mettant en jeu cinq types de composants [35]: les classes, les relations, les fonctions, les axiomes et les instances.

- **Les classes** : sont habituellement organisées en taxonomies. Elles réfèrent à des concepts, utilisés dans le sens large. Ces concepts peuvent être abstraits ou concrets, élémentaires (électron) ou composés (atome), réels ou fictifs.

- **Les relations(R) :** représentent un type d'interaction entre les notions d'un domaine (Ci). Elles sont formellement définies comme tout sous-ensemble d'un produit de n ensembles, c'est-à-dire Re CI XC2 X... XCn• Par exemple, les relations binaires sont du type *«sous-classe-de», «connecté-à»*, etc.

- **Les fonctions**: sont des cas particuliers de relations dans lesquelles le nième élément de la relation est défini à partir des n-1 premiers. Formellement, les fonctions (F) sont définies ainsi: F : CI x C2 X ... X Cn-I -7 Cn•

- **Les axiomes** : sont utilisés pour affirmer des phrases qui sont toujours vraies.

- **Les instances :** sont utilisées pour représenter des éléments selon un principe similaire à la relation classe/objet en UML. Exemple: *Paris* est une instance de Ville.

10.3 Utilité de l'ontologie

Les ontologies sont devenues des éléments fondamentaux dans toute une gamme d'applications faisant appel à des connaissances. La structure d'une ontologie permet de représenter les connaissances d'un domaine sous un format informatique en vue de les rendre utilisables pour différentes applications [32]. Une ontologie peut être utilisée soit :

- pour partager des connaissances entre agents humains et / ou artificiels.
- pour faciliter la recherche d'information au sein d'un même système ou de systèmes hétérogènes.

Elle est aussi primordiale pour la médiation des systèmes d'information ayant des modèles et des sémantiques hétérogènes. C'est donc une clé de toute première importance dans l'étude de l'agilité des partenaires et dans le développement de l'activité réseau [19].

10.4 Structuration de l'ontologie

La construction de l'ontologie fait partie d'un processus, qui permet d'évaluer et de faire évoluer cette ontologie lorsque de nouvelles connaissances sont détectées dans le système ou suite à une modification des propriétés d'un de ses éléments [Figure 4.2]. Cela signifie que l'ontologie doit être disponible et refléter l'existant à tout moment Pokarev et al [33].

Une ontologie peut s'exprimer sous la forme d'un ensemble de schémas XML ou en utilisant OWL. OWL (Ontology Web Language) est un méta-langage XML dédié à la description des ontologies standardisé par le W3C (World Wide Web Consortium) basé sur RDF (Resource Description Framework) et RDFS (RDF Schema). OWL permet de décrire des relations et contraintes complexes sur des ontologies (exclusion, dépendance, etc.) et est composé de trois sous-langages de complexité croissante : OWL-Lite, OWL-DL et OWL-Full. RDF et OWL sont au coeur des évolutions futures du « semantic Web », qui ajoute aux informations disponibles sur le Web une composante sémantique, ce qui permet des traitements plus intelligents de l'information et donc une intégration facilitée [19].

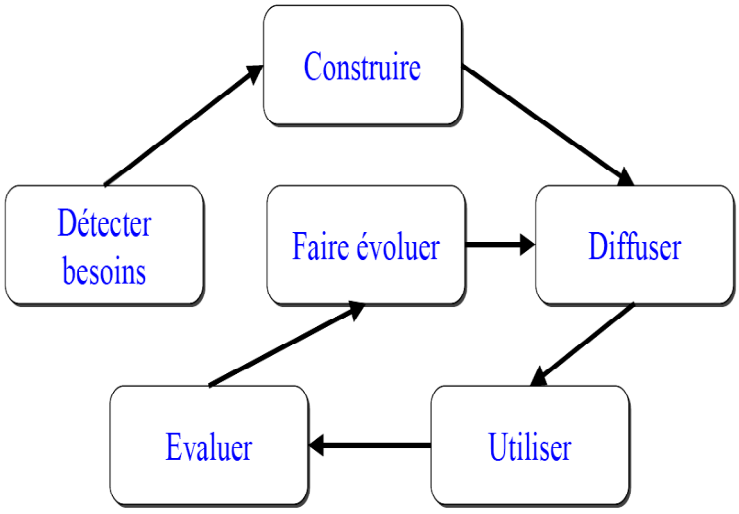


Figure II.11: Le cycle de vie de la construction de l'ontologie entreprise Pokarev et al [33].

10.5 Exemples d'ontologies standardisées

Un grand nombre d'initiatives ont été développées afin de spécifier des ontologies en utilisant des schémas XML. Elles sont classées généralement suivant deux axes :

- **Vertical** : s'il s'agit d'initiatives dédiées à un métier (domaine) particulier : MathML (Mathematic Markup Language) pour le domaine des mathématiques, GML (Geography Markup Language) pour le domaine de la géographie, etc.
- **Horizontal** : s'il s'agit d'initiatives pouvant être déclinées pour des métiers (des domaines) différents : UBL (Universal Business Language) est un standard de l'OASIS visant à normaliser en XML les documents commerciaux courants [19].

10.6 Catégorisation des ontologies

Une ontologie est le résultat d'une modélisation. La modélisation porte sur la caractérisation de primitives pour la représentation formelle des connaissances. Ces primitives ne sont pas des données du domaine qu'il suffirait de déterminer, mais des constructions théoriques pour les besoins de la modélisation. Les ontologies peuvent être classifiées selon plusieurs dimensions. Parmi celles ci, nous en examinerons quatre : 1) Objet de conceptualisation ; 2) Niveau de détail; 3) Niveau de complétude; 4) Niveau de formalisme de représentation. Le tableau suivant représente une classification des ontologies synthétisée de plusieurs travaux [19], [77].

Tableau II.3 Classification des ontologies

Niveaux	Classification des ontologies
l'objet de conceptualisation	- **Ontologie de représentation des connaissances**: ce type d'ontologies regroupe les concepts impliqués dans la formalisation des connaissances. - **Ontologie supérieure ou de Haut niveau :** Cette ontologie est une ontologie générale. Son sujet est l'étude des catégories des choses qui existent dans le monde, soit les concepts de haute abstraction tels que: les entités, les événements, les états, les processus, ... **-Ontologie Générique** : Cette ontologie aussi appelée, **méta-ontologies**, véhicule des connaissances génériques moins abstraites que celles véhiculées par l'ontologie de haut niveau, mais assez générales néanmoins pour être réutilisées à travers différents domaines. - **Ontologie du Domaine** Cette ontologie régit un ensemble de vocabulaires et de concepts qui décrit un domaine d'application. Elle permet de créer des modèles d'objets du monde cible.

	- **Ontologie de Tâches :** Ce type d'ontologies est utilisé pour conceptualiser des tâches spécifiques dans les systèmes, telles que les tâches de diagnostic, de planification, de conception,... Elle régit un ensemble de vocabulaires et de concepts qui décrit une structure de résolution des problèmes inhérente aux tâches et indépendante du domaine. **-Ontologie d'Application :** Cette ontologie est la plus spécifique. Les concepts dans l'ontologie d'application correspondent souvent aux rôles joués par les entités du domaine tout en exécutant une certaine activité
Détail de l'ontologie	- **Granularité fine** : correspondant à des ontologies très détaillées, possédant ainsi un vocabulaire plus riche capable d'assurer une description détaillée des concepts pertinents d'un domaine ou d'une tâche. - **Granularité large** : correspondant à un vocabulaire moins détaillé comme par exemple dans les scénarios d'utilisation spécifiques où les utilisateurs sont déjà préalablement d'accord à propos d'une conceptualisation sous-jacente .
Le niveau de complétude	**– Niveau Sémantique** : Tous les concepts doivent respecter les quatre principes différentiels : Communauté avec l'ancêtre; Différence par rapport à l'ancêtre; Communauté avec les concepts frères; Différence par rapport aux concepts frères. ça est pour assurer que chaque concept aura un sens univoque et non contextuel associé. **– Niveau Référentiel** : Outre les caractéristiques énoncées au niveau précédent, les concepts référentiels (ou formels) se caractérisent par un terme/libellé dont la sémantique est définie par une extension d'objets. – **Niveau Opérationnel** : Outre les caractéristiques énoncées au niveau précédent, les concepts du niveau opérationnel ou computationnel sont caractérisés par les opérations qu'il est possible de leur appliquer pour générer des inférences.
Le niveau du formalisme	**-Informelles :** ontologies opérationnelles dans un langage naturel; **-Semi informelles:** utilisation d'un langage naturel structuré et limité **-Semi formelles:** langage artificiel défini formellement; **-Formelles** : utilisation d'un langage artificiel contenant une sémantique formelle, ainsi que des théorèmes et des preuves des propriétés telles la robustesse et l'exhaustivité.

10.7 Fondements de l'ingénierie ontologique

Le processus de construction d'ontologies, appelé ingénierie ontologique, peut être décrit selon les principes qui le gouvernent, et les méthodologies et les outils qui le soutiennent.

10.7.1 Principes

Il existe un ensemble de critères et de principes qui ont fait leurs preuves dans le développement des ontologies et qui peuvent être résumés comme suit [77] :

- **Clarté et Objectivité**: L'ontologie doit fournir la signification des termes définis en fournissant des définitions objectives ainsi qu'une documentation en langage naturel.
- **Complétude**: Une définition exprimée par des conditions nécessaires et suffisantes est préférée à une définition partielle (définie seulement par une condition nécessaire et suffisante).
- **Cohérence**: Une ontologie cohérente doit permettre des inférences conformes à ces définitions.
- **Extensibilité monotonique maximale**: De nouveaux termes généraux et spécialisés devraient être inclus dans l'ontologie d'une façon qui n'exige pas la révision des définitions existantes.
- **Engagements ontologiques minimaux**: Ce principe invite à faire aussi peu de réclamations que possible au sujet du monde représenté. L'ontologie devrait spécifier le moins possible la signification de ses termes, donnant aux parties qui s'engagent dans cette ontologie la liberté de spécialiser et d'instancier l'ontologie comme elles le désirent.
- **Principe de distinction ontologique** : les classes dans une ontologie devraient être disjointes. Le critère utilisé pour isoler le noyau de propriétés considérées comme invariables pour une instance d'une classe est appelé le critère d'Identité.
- **Modularité**: Ce principe vise à minimiser les couplages entre les modules.
- **Diversification des hiérarchies**: Ce principe est adopté pour augmenter la puissance fournie par les mécanismes d'héritage multiple. Si suffisamment de connaissances sont représentées dans l'ontologie et que suffisamment de différentes classifications de critères sont utilisées, il est plus facile d'ajouter de nouveaux concepts (puisqu'ils peuvent être facilement spécifiés à partir des concepts et des classifications de critères pré-existants) et de les faire hériter de propriétés de différents points de vue. Certains chercheurs comme Mizoguchi *et al.*, sont opposés à l'idée d'héritage multiple en ingénierie ontologique.
- **Distance sémantique minimale :** Il s'agit de la distance minimale entre les concepts enfants de mêmes parents. Les concepts similaires sont groupés et représentés comme des sous -classes d'une classe, et devraient être définis en utilisant les mêmes primitives, considérant que les concepts qui sont moins similaires sont représentés plus loin dans la hiérarchie.

- **Normaliser les noms** : Ce principe indique qu'il est préférable de normaliser les noms aussi autant que possible. Cet ensemble de critères et de processus est généralement accepté pour guider le processus d'ingénierie ontologique.

10.8 Langages d'ontologie

Les ontologies jouent un rôle très important dans le Web Sémantique en permettant la spécification formelle des vocabulaires pour la description du contenu du Web. Ainsi, les informations du Web peuvent être accessibles et compréhensibles à la fois par les humains et par les machines. Plusieurs langages ont été proposés pour représenter les ontologies avec une expressivité plus ou moins élevée et une complexité de raisonnement plus ou moins grande. Nous nous intéressons aux trois langages de représentation d'ontologies recommandés par le W3C (World Wide Web Consortium).il s'agit d'OWL (Ontology Web Language), RDF (Resource Description Framework) et SWRL (Semantic Web Rule language)

a. RDF (Resource Description Framework)

Il s'agit d'un langage conçu initialement par le W3C, permet de décrire la sémantique des données accessible sur le Web, de manière compréhensible par les machines.

- Il est un modèle, associé a une syntaxe, dont le but est de permettre `a une communauté d'utilisateurs de partager les mêmes métadonnées pour des ressources partagées.
- Permet l'échange des métadonnées et leur traitement par des agents humains ou logiciels.
- Un des gros avantages de RDF est son extensibilité.

Le modèle de données RDF est constitué de trois types de composantes :

- Sujet : il représente la ressource à décrire. Il est nécessairement identifié par une URI.
- Prédicat ou Propriété : il s'agit d'une propriété utilisée pour caractériser et décrire une ressource. Un prédicat est nécessairement identifié par une URI.
- Objet : représente une donnée ou une autre ressource (identifiée par une URI).

En utilisant le langage RDF, les informations des ressources sont décrites par un ensemble de déclarations RDF sous forme de triplets : (Sujet, Prédicat, Objet).

```
<?xml version = “1.0” ?>
<rdf:RDF
Xmlns:rdf = http://www.w3.org/1999/02/22rdfsyntaxns#>
<rdf:Description
About = http://www.ifi.edu.vn/index.html>
<Creator> Institut de la francophonie pour l'Informatique </Creator>
</rdf:Description>
</rdf:RDF>
```

➢ Inconvénients

- Permet d’exprimer des assertions, mais pas de définir un vocabulaire de termes et de relations.
- RDFS permet de définir des classes (rdfs:Class), des sous-classes (rdfs:subClassOf), des sous propriétés (rdfs:subPropertyOf).
- RDFS apparait néanmoins insuffisant pour définir des ontologies.
- En RDFS il n’y a pas de distinction entre les classes et les instances et on ne dispose pas de la possibilité d’indiquer des contraintes sur un domaine, pas plus que sur les cardinalités ou bien encore de préciser qu’une propriété est transitive, inverse ou symétrique.

L’ensemble de ces manques fait que RDF/RDFS a un pouvoir expressif limité et insuffisant pour répondre aux exigences du Web sémantique. C'est pour cela qu'il est souhaitable d'utiliser d'autres langages plus expressifs pour la représentation d'ontologies. Parmi les langages proposés, on trouve notamment OWL (Ontology Web Language) qui peut être considéré comme une extension de RDFs pour définir des ontologies complètes, relations entre les classes (hiérarchies, transitivité..) et typage de propriétés plus riches (cardinalités...) [86], [88] et [89].

b. OWL (Ontology Web Langage)

Le langage OWL (Ontology Web Language) est un langage de représentation d'ontologies avec un pouvoir expressif plus important et une complexité plus grande pour l'inférence. L'origine du langage OWL est le langage DAML+OIL.

- OWL est un langage de définition d’ontologies destiné, en particulier, à décrire les ressources du Web.

- OWL permet de formaliser un domaine en définissant des classes et leurs propriétés. Comme OWL est doté d'une sémantique formelle en théorie des modèles. La sémantique formelle indique comment déduire les conséquences logiques d'une ontologie.

Concrètement, la sémantique formelle d'un langage comme OWL peut être vue comme un ensemble de règles génériques de raisonnement (par exemple transitivité de la relation de subsumption). Il est ainsi possible de mener des raisonnements sur les classes et les individus : raisonnements terminologiques et assertionnels.

Plus un outil est complet, plus il est, en général, complexe. C'est cet écueil qu'a voulu éviter le groupe de travail WebOnt du W3C en dotant OWL de trois sous-langages offrant des capacités d'expression croissantes et, naturellement, destinés à des communautés différentes d'utilisateurs :

- **OWL LITE**: permet d'établir une hiérarchie de concepts simples, contraintes simples.
- **OWL DL** (DL pour description logic): comprend toutes les structures de OWL, possède une expressivité plus importante, avec complétude de calcul.
- **OWL FULL** expressivité maximale, liberté syntaxique sans garantie de calcul, une classe peut aussi correspondre à l'instance d'une autre classe.

➢ **Inconvénient**

OWL fournit un grand nombre de constructeurs permettant d'exprimer de façon très fine les propriétés des classes définies. La rançon de cette expressivité est l'indécidabilité du langage obtenu en considérant l'ensemble de ces constructeurs. C'est pour cela un des buts d'OWL est d'améliorer les performances des moteurs de recherche du Web qui ne reposent que sur le principe des mots clés [85], [86] et [89].

c. SWRL (Semantic Web Rule Language)

Le langage SWRL a été soumis comme candidat à la recommandation par le W3C comme langage de règles pour le Web sémantique. SWRL est un langage qui est fondé sur la combinaison du langage OWL-DL avec le sous langage Datalog RuleML (Rule Markup Language). SWRL peut être vu comme une logique de description étendue par une syntaxe abstraite pour exprimer des règles de Horn21 en logique du premier ordre sans symboles de fonctions. Par exemple, on peut déclarer des règles telles que [85], [89] :

filleDe(?*x;* ?*z*) ^ *freres*(?*z;* ?*w*) ^ *filleDe*(?*y;* ?*w*) *=) cousines*(?*x;* ?*y*)

Avec *FilleDe*(?*x;* ?*z*), *Freres*(?*z;* ?*w*) et *FilleDe*(?*y;* ?*w*) sont les prémisses de la règle et *Cousines*(?*x;* ?*y*) est la conclusion de la règle.

Lorsque les prémisses sont satisfaites, la conclusion de la règle est inférée. Ce type de relations ne peut pas être exprimé dans le langage OWL. Cette règle est exprimée en syntaxe SWRL comme suit :

Implies (Antecedent (filleDe (I_v a r i a b l e (x) I_v a r i a b l e (z))
freres (I_v a r i a b l e (z) I_v a r i a b l e (w))
filleDe (I_v a r i a b l e (y) I_v a r i a b l e (w)))
Consequent (cousines (I_v a r i a b l e (x) I_v a r i a b l e (y))))

Où, le constructeur ***Antecedent*** indique que ses arguments sont les prémisses de la règle, le constructeur ***Consequent*** indique que son argument est la conclusion de la règle et le constructeur. ***I_variable*** indique que son argument est une variable.

10.9 Outils pour éditer les ontologies

Il existe un nombre important d'outils de manipulation d'ontologies, par exemple :

a) Protégé : Conçu par le SMI à Stanford, « Protégé » est un outil complet dont le système de plugins permet l'échange avec de nombreux formats, l'inclusion de moteurs de raisonnement, et en général une évolutivité très importante. Il permet notamment l'échange avec de nombreuses ontologies via XML (un des formats de Protégé étant déjà du RDF), de plus, un plugin permet la visualisation des ontologies via Graphviz.

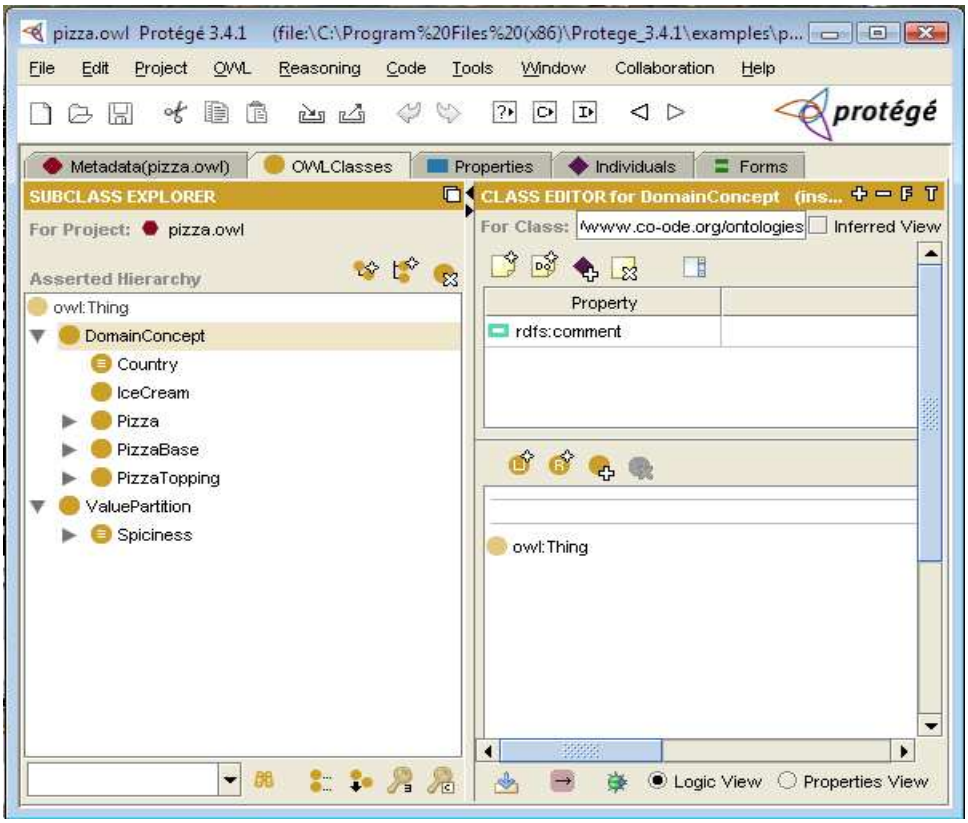


Figure II.12 Interface de l'outil 'Protégé'

b) OntoEdit : développé au Knowledge Management Group de l'université de Karlsruhe, est supporté par une architecture client-serveur appelée OntoServer, qui intègre un répertoire d'ontologies, un moteur de recherche et d'inférence, et plusieurs traducteurs.

11. Annotation sémantique des services Web

Une annotation sémantique dans un document est les informations supplémentaires qui identifient ou définissent un concept dans un modèle sémantique pour décrire la partie de ce document.

11.1 Le standard SAWSDL

SAWSDL est une extension simple de WSDL en utilisant des éléments d'extensibilité. Le World Wide Web Consortium propose avec les Semantic Annotations for Web Services Description Language (SAWSDL) un moyen d'annoter les descriptions WSDL 2.0 tout en supportant WSDL 1.1. SAWSDL est un ensemble d'attributs d'extension permettant de d´écrire la sémantique des éléments contenus dans les documents WSDL. L'objectif de SAWSDL est de définir comment une annotation doit être réalisée, tout en laissant le choix du langage utilisé pour la description sémantique. SAWSDL fournit les mécanismes permettant d'attacher des concepts décrits dans des ontologies aux annotations des descriptions WSDL. Cette proposition étend WSDL-S, elle peut être considérée comme une continuité de ce langage. Elle vise à apporter la valeur ajoutée de la sémantique non seulement lors de l'invocation des services Web, mais aussi durant la phase de découverte. Trois attributs d'extensibilité sont définis par défaut [36] :

- l'attribut <modelReference > permet l'association entre un composant WSDL

et un concept d'une ontologie.

- Les attributs, < liftingSchemaMapping > et < lowering-SchemaMapping>, sont ajoutés aux définitions de types pour spécifier les correspondances entre les éléments du schéma des données et l'information sémantique de l'ontologie.

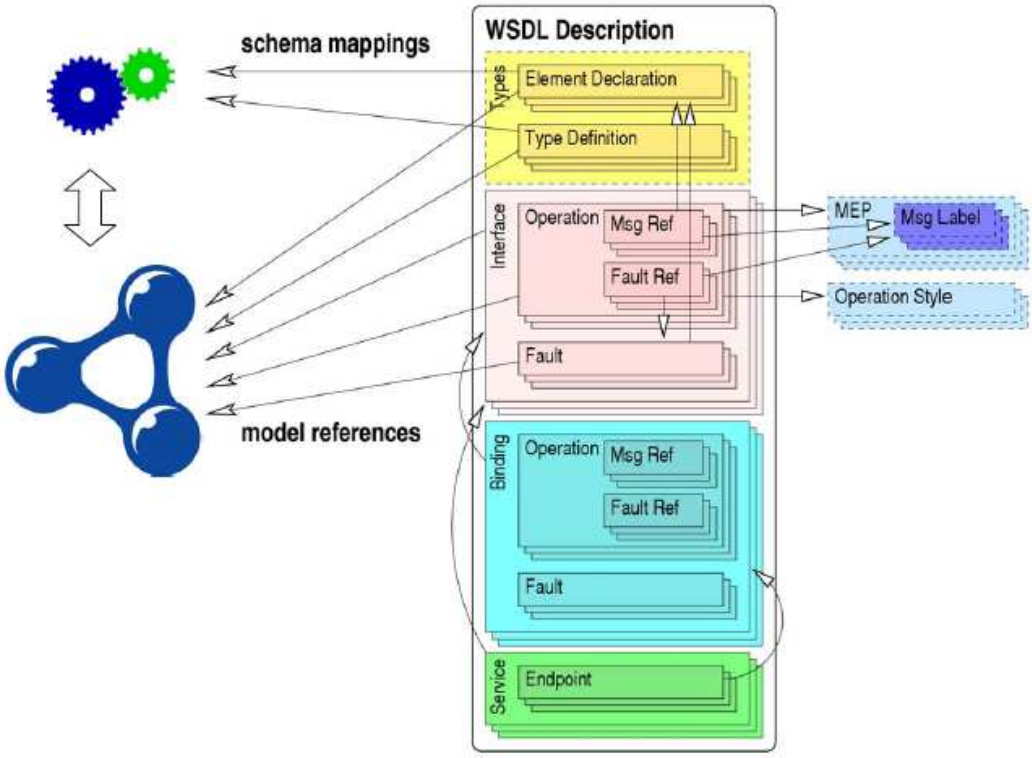


Figure II.13 : le standard SAWSDL [91]

11.2 OWL-S

OWL-S est un langage de description de service Web. L'organisation créatrice de l'ontologie, s'appelait initialement "DAML-S coalition" car elle était exprimée dans le langage DAML+OIL (Darpa Agent Markup Language + Ontology Inference Layer) ; elle a été renommée "OWL-S coalition" [93]. OWL-S est une approche dépendante au langage de description d'ontologies OWL.

La structuration de l'ontologie d'OWL-S est motivée par la nécessité de fournir trois types d'information essentiels pour un service, à savoir :

- Que fait le service : cette information est donnée dans le Service Profile par des propriétés : serviceName, textDescription (brève description de ce que le service offre et requiert), ContactInformation (responsables du service).
- Comment utilise-t-on le service : cette information est fournie dans le Service Model. Trois types de processus existent : Les processus atomiques (AtomicProcess), simples (SimpleProcess) et composites (CompositeProcess).

➢ Un AtomicProcess représente le niveau le plus fin pour un processus et correspond à une action que le service peut effectuer en une seule interaction avec le client.

- Les CompositeProcess sont décomposables en d'autres processus (composés ou non) ;
- Un SimpleProcess est utilisé pour fournir une vue d'un processus atomique ou une représentation simplifiée d'un processus composite ;

- Comment accède-t-on au service : cette information est donnée dans le Service Grounding. Celui-ci indique comment accéder concrètement au service et fournit les détails concernant les protocoles, les formats de messages et les adresses physiques [95].

11.3 WSMO

WSMO est une ontologie de services basée sur WSMF (Web Service Modelling Framework) qui spécifie les éléments principaux pour décrire les services web sémantiques tels que ontologies, objectifs, services Web et médiateurs. Les ontologies définissent la terminologie, utilisée par les autres éléments, en termes de concepts, relations, fonctions, instances et axiomes. Les buts indiquent ce que l'utilisateur attend du service. La description du service Web définit les fonctionnalités offertes par le service. Les médiateurs lient les différents éléments afin de permettre l'interopérabilité entre les composants hétérogènes. Le langage WSML est utilisé pour décrire formellement tous les éléments de WSMO [95].

11.4 Comparaison

- OWL-S et WSMO, sont des approches qui dépendent des langages de description d'ontologies spécifique, OWL et WSML, se limitent à un ensemble de concepts bien définie. Alors que SAWSDL permet d'utiliser tous types d'ontologies (OWL, WSML, UML, etc.).
- SAWSDL ne nécessite pas beaucoup d'efforts pour les développeurs habitués à WSDL. Les principaux objectifs des approches de description sémantique de service sont l'automatisation et l'amélioration de la découverte, la composition et l'invocation services. Par contre SAWSDL ne permet ni la composition ni la découverte de service basée sur propriétés non fonctionnelles. Ce défaut est principalement du de la limite d'expressivité du méta modèle de WSDL.

12. Conclusion

Dans ce chapitre, nous avons effectué une étude sur les services Web ou on a constaté qu'ils ne sont pas la seule solution pour mettre en place une architecture SOA.Quelques quinze ans après sa création, le Web semble enfin comprendre son objectif initial : le partage rapide de savoirs précis. Pas uniquement leur présentation anarchique mais, plutôt, leur mise à disposition dans un format non ambigu, compréhensible par tous et extensible.

Créés dans un objectif de partage des connaissances en réseau, les derniers langages du Web sémantique que sont RDF et OWL sont en vérité les premières pierres d'une nouvelle forme de Web. Facilitant l'appropriation des savoirs en les libérant de la couche présentationnelle qui les enfermait jusqu'alors, le Web sémantique contribue, en ce sens, à rejoindre les objectifs initiaux de Tim Berners Lee. D'une manière plus générale, la richesse logicielle qui entoure les créations du W3C ne trompe pas : le Web de demain sera sémantique.

Dans le chapitre suivant, nous exposons notre approche de migration des systèmes patrimoniaux vers les services Web en adoptant l'approche boîte grise, en détaillant toutes les étapes proposées et en expliquant les techniques utilisées dans chaque phases de l'approche.

Approche proposée

1. Introduction

Les systèmes patrimoniaux sont précieux pour les organisations. Ces dernières, pour qu'elles soient a la hauteur de la satisfaction de leurs clients, il est indispensable, que leurs patrimonial accompagne l'évolution technologique. Plusieurs travaux ont été proposés, permettant la transformation des systèmes patrimoniaux à des services Web. Nous citons l'approche boite noir, boite blanche et boite grise sur laquelle se base notre présent travail. Selon notre point de vue, tous les travaux existants ont abordé l'aspect syntaxique mais présentent un inconvénient majeur celui de l'hétérogénéité sémantique des services web résultants.

Pour pallier à cette insuffisance, nous enrichissant la combinaison des deux approches celle de S. Alhamari et celle de J. Zhi, en adoptant l'approche boîte grise. Cet enrichissement offre l'annotation sémantique des services web par le standard SAWSDL en se basant sur l'ontologie de domaine et l'ontologie contextuelle. L'annotation a pour objectif d'éviter l'hétérogénéité sémantique. Cette hétérogénéité qui est du, par exemple, au changement de sens des informations en fonction de leurs contextes.

Une vue globale de notre proposition sera exposée dans la section qui suit. Ensuite, nous détaillerons chaque phase de cette approche à part, en présentant le rôle de chacune dans le processus d'émigration. Nous terminerons ce chapitre par une conclusion.

2. Travaux connexes

Il existe pas mal des travaux dans le domaine d'identification des services à partir d'un Système patrimonial dont ils se concentrent sur la technique de clustering pour extraire le code réutilisable. Parmi les quels :

Celui de S. Alhamari [78] et all qui proposent une approche pour identifier les services clés dans un code hérité avec une granularité optimale. Cette approche définie les services potentiels a base des deux standards de modélisation : UML et BPMN. Elle se compose de trois étapes : Analyse et réingénierie, identification des

services en appliquant l'algorithme de clustering au niveau des processus atomiques pour définir les services de granularité fine et en fin l'évaluation des services.

Z.Zhang et all [76,75] proposent une approche d'identification du code hérité réutilisables pour soutenir la réingénierie de logiciel orientée service (SOSR). Cette identification est atteinte par le mapping des ontologies FO et SCO via une nouvelle méthode combinant l'analyse formelle des concepts (CAF) et l'Analyse des concepts relationnels (RCA).Cette approche comporte cinq étapes : Evaluation du système patrimonial, Analyse du domaine, Identification des Services candidats, Extraction des composants hérités et Intégration des Services.

J. Zhi li et all [74] proposent une approche comportant les étapes suivantes : évaluation et analyse du système, décomposition du système patrimonial, Grid service component packaging qui englobe deux sous étapes componentisation du code hérité et son encapsulation en fin l'optimisation de la granularité des services.

Le World Wide Web Consortium [36] propose avec les Semantic Annotations for Web Services Description Language (SAWSDL) un moyen d'annoter les descriptions WSDL 2.0 tout en supportant WSDL 1.1. SAWSDL est un ensemble d'attributs d'extension permettant de décrire la sémantique des éléments contenus dans les documents WSDL .Ces attributs sont : modelReference, liftingSchemaMapping et lowering-SchemaMapping.

Les approches décrites ci-dessus nous ont permis d'aborder notre proposition en les prenant comme modèle.

3. Approche proposée

Notre approche est inspirée de l'approche de S. Alhamari et celle de J. Zhi et all qui recouvre toutes les étapes nécessaires à l'extraction des services Web à partir des systèmes patrimoniaux [Figure III.1]. Pour tirer avantage de ces deux approches, nous avons procédé à leur combinaison en intégrant une annotation semi automatique aux services web, méritant d'être publiés, par le standard SAWSDL en se basant sur l'ontologie de domaine et l'ontologie contextuelle afin d'éviter le problème d'hétérogénéité sémantique.

Nous commençons par évaluer le système patrimonial à fin de pouvoir décider s'il est possible de faire la migration. Cette décision est basée sur l'existence des fonctionnalités méritant d'être réutilisées comme des services Web. Si l'étude de faisabilité est possible, nous entamons l'étape d'analyse et réingénierie pour déterminer les besoins des services vis-à-vis leurs exigences et faire le reverse ingeniering en modélisant le système par le standard UML. Apres, viendra l'étape d'identification des services ou nous distinguons les services qui doivent être extraits du système patrimonial et les autres qui doivent être implémentés à nouveau.

Sur la base des résultats obtenus à l'étape d'identification des services nous entamons l'étape de détection du code des fonctions méritant d'être publiées en tant que services Web en appliquant la technique de clustering.

Parallèlement à cette dernière, nous proposons de générer l'ontologie de concept de domaine (OCD) et contextuelle (OC) exprimées en OWL en se basant sur le modèle métier UML.

En suite, vient l'étape d'adaptation, a son achèvement nous proposons l'étape d'annotation des descriptions des services par le standard SAWSDL en se basant sur les ontologies résultantes pour ajouter de la sémantique aux services web et éviter l'hétérogénéité sémantique due au changement de sens des informations en fonction de leurs contextes.

Enfin, nous entamons la publication des services Web annotés.

Dans ce qui suit nous illustrons les étapes de notre proposition par la figure ci-dessous [figure III.1] :

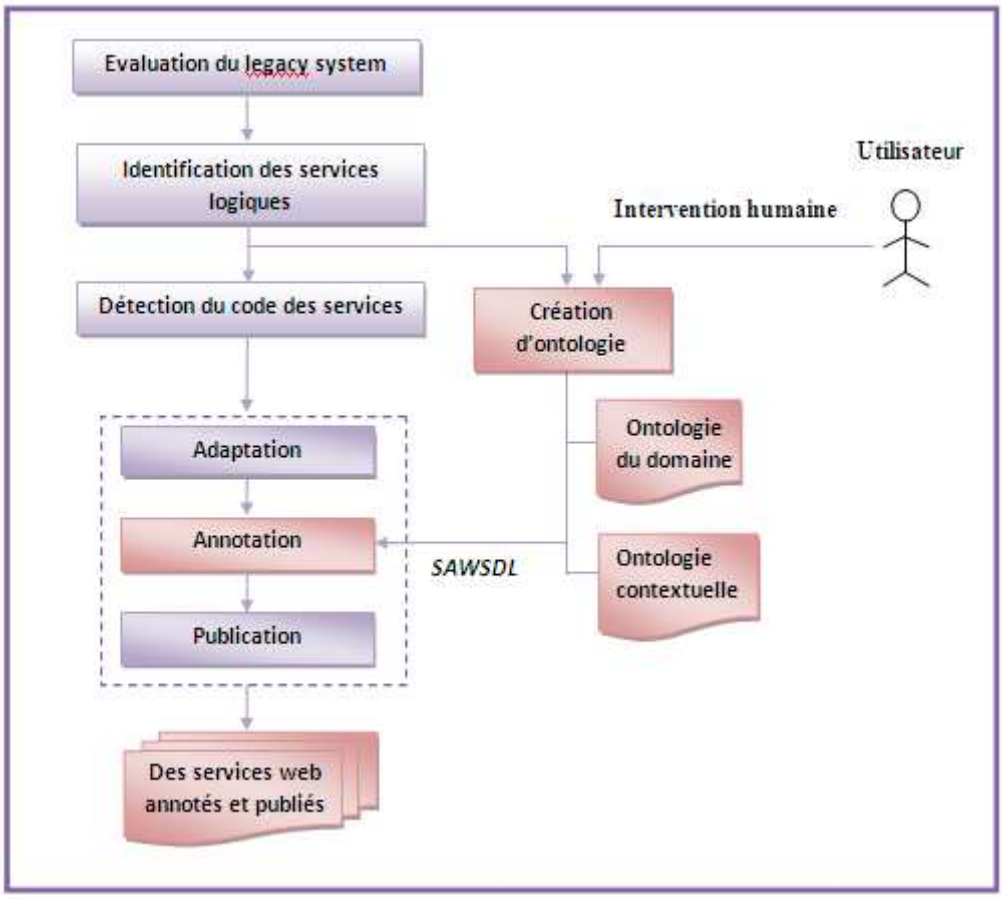


Figure III.1 Annotation des services web extraits d'un système patrimonial par le standard SAWSDL

4. Evaluation du système patrimonial

Cette évaluation est importante et de notre point de vue est indispensable pour vérifier la faisabilité de la migration du système patrimonial. Cette étape extraite de plusieurs travaux Z.Zhang [76], [45], Sneed [56], permet de savoir si c'est possible de migrer ce système patrimonial vers une architecture orientée service (SOA) et si cette migration est rentable.

D'une autre manière, cette phase nous permet, de vérifier si les fonctionnalités de ce système sont utiles pour être exposées en tant que services Web indépendants, de tirer le bénéfice de maintenir quelque fonctionnalités potentielles au lieu de maintenir le système entier et de garantir que ces fonctionnalités fonctionnent correctement sur internet. En général cette phase est adoptée au démarrage du

processus, qui est notre cas, pour gagner de l'effort dans le cas où les résultats de vérification sont décourageants.

5. Identification des services logiques

Pour construire un fondement suffisant pour l'identification des services, nous commençons par l'analyse et la réingénierie qui vont être appliquées pour déterminer les besoins des services d'après les exigences bisness. Cette phase comporte deux étapes qui sont les suivantes:

a. l'analyse du domaine : Ce processus d'analyse de domaine est important pour la conception des services et la réutilisation du système existant. Il Permet de comprendre le système et de connaitre les besoins métier.la compréhension du système est indispensable parce que le code hérité en lui-même ne suffit pas. L'analyse du domaine a pour but de recueillir les besoins, identifier les caractéristiques générales de l'information correspondantes au domaine de l'application et de distinguer à l'intérieur du code source, les règles d'affaires sur lesquelles le système est basé.

On peut comprendre le système patrimonial selon les transactions qui y sont effectuées.

- **Exemple des règles d'affaires**

Soit un système informatique d'assurance médicale pour les employés d'une entreprise. Des règles d'affaires pourraient être :

- Si l'employé fait une demande d'indemnisation.
  Si l'employé est assuré et l'employé a toujours payé ses cotisations.
  Alors la demande d'indemnisation est valide.
  Ajouter la demande à la liste pour appréciation.
  Changer le statut de la demande à "en attente d'approbation".

Les résultats de l'analyse de domaine spécifient des fonctions d'affaires précieuses et réutilisables, qui peuvent être exposées comme des services logiciels.

Par conséquent, les services logiques sont identifiés par l'analyse de domaine. Si le système requiert de nouvelles exigences, des nouveaux services peuvent apparaître comme candidats dans le processus de migration [76], [78].

Pour découvrir les fonctions principales et mieux comprendre le système nous proposons, comme c'est le cas de plusieurs travaux Z. zhang [76,75], S. Alahmari et all [78], J. Zhi li et all [74], d'introduire les documents disponibles, les questionnaires et les interviews avec les utilisateurs dont ils sont les seuls qui connaissent bien leurs systèmes. A partir de leur expérience dans l'utilisation de ce système et des autres différents systèmes, ils se forment une idée très claire sur les fonctionnalités qui doivent être réutilisées.

b. Reverse ingineering : c'est de générer le model du domaine UML via la technique reverse ingineering afin de décrire la structure fondamentale du système. En se basant sur le modèle de domaine d'application, des nouveaux services apparus comme des candidats dans le processus d'émigration.

6. Détection et extraction du code

Les systèmes logiciels évoluent avec le temps, à la suite de changements apportés aux exigences. Les systèmes qui en résultent ont tendance à avoir une structure riche, complexe et fortement couplée. En raison de maintenance à long terme et d'évolution, la documentation peut ne pas être en mesure de refléter la structure actuelle des systèmes existants.

Dans de nombreux cas, le partitionnement efficace est nécessaire pour décomposer les systèmes existants avec une forte cohésion et a couplage faible. Les techniques de clustering sont appliquées pour extraire des segments de code réutilisables, indépendants, autonomes, à grains grossiers, et faiblement couplés.

L'analyse par la méthode Clustering regroupe un grand support d'entités dans un ensemble de données appelé clusters en fonction de leur relation et de leur similitude. Ceci est aussi appliqué pour la compréhension du programme, tels que remodularisation et la récupération des composants où elle est particulièrement utile pour rénover les systèmes patrimoniaux. Les méthodes de Clustering peuvent être utilisées pour identifier des objets dans les systèmes existants basés-procédures.

Dans de nombreux travaux Z.Zhang et al [74, 80, 73] ont choisi des algorithmes dont la structure réponde a des contraintes qu'une décomposition efficace doit obéir.

La méthode de clustering analyse le code hérité et exprime les résultats en un dendrogramme, qui présente une vue hiérarchique du système patrimonial. Cette vue hiérarchique se compose de différents niveaux d'abstraction a partir du code source jusqu'aux sous-systèmes. Cette méthode de bottom-up clustering est faisable et efficace pour la compréhension du système existant.

- Les entités potentielles pourraient inclure un sous-système, répertoire, fichier, classe, fonction, structure de données ou variable.
- Un coefficient de ressemblance pour un couple donné d'entités indique le degré de similitude ou de dissemblance entre ces deux entités; en fonction de la manière dont les données sont représentées. Ce coefficient pourrait être qualitatif ou quantitatif. Dans les deux cas les données devraient être choisies en fonction des applications et des attributs.

Après l'application de l'algorithme de clustering, les résultats devraient être évalués et expliqués correctement. Cette méthode a une simplicité conceptuelle et mathématique.

6.1 Analyse de clustering

L'analyse de clustering peut être divisée en trois étapes sachant que l'auteur a défini :

- Les fonctions, les procédures sont considérées comme des entités pour être regrouper. Les attributs sont utilisés pour mesurer la similarité entre les entités.
- Il est Possible que les identificateurs des attributs puissent être le nom d'une fonction, d'une procédure ou d'une propriété.

Afin de récupérer un service particulier, des identificateurs concernant le service cible indiquent l'attribut de poids fort. La définition des attributs est bien la clé d'une restructuration plus fonctionnelle d'un système patrimonial. Les étapes qui constituent l'analyse de clustering sont [74]:

Etape 1 : obtenir les donnés de la matrice en définissant les entités (fonction, procedure).

Etape 2 : calcul du coefficient de ressemblance (similarité) entre les entités définies dans l'étape précédente. Pour calculer la similarité, Nous proposons d'utiliser le coefficient de Jaccard [81], qui se calcule selon la formule:

$$SIM(x, y) = \frac{card\ (Sx \cap Sy)}{card\ (Sx \cup Sy)}$$

Tel que :

X, Y : dénotent les entités (fonction ou procédure).

S : dénote les attributs de X ou de Y.

Etape 3 : Exécution de L'algorithme de Clustering

L'algorithme de clustering est une séquence d'opérations qui regroupe les entités similaire dans des clusters d'une façon incrémentale. Cette séquence commence par chaque entité trouvée dans un cluster (groupe) séparé. A chaque étape, les deux clusters les plus proches l'un de l'autre sont fusionnés et le nombre de clusters diminue de un.une fois les deux clusters sont fusionnés, Le coefficient de ressemblance est mis a jour entre le nouveau cluster et le reste des clusters pour reflété leurs rapprochement a ce nouveau cluster.cet algorithme est utilisé pour l'extraction du code des fonctionnalités d'un Systèmes patrimonial dont ses étapes sont les suivantes [74], [80] :

1. E_i (1 <= i <= n) est une entité dans un système basé procedure. F_i représente l'ensemble des attributs d'une entité E_i. $W(F_i)$ dénote le poids total de F_i, $L(k)$ est le niveau de $k^{ème}$ Clustering. A cluster avec le numéro de séquence m est dénoté par (m) et la similarité entre les clusters (a) et (b) est dénoté $Sim[(a),(b)]$.
2. $L(0) = 0$, numéro de séquence $m = 0$ et la matrice du similarité est $D_{n,n} = \{Sim\ [i\ ,\ j]\}$.
3. Chercher la paire (a,b) du clusters les plus proche (similaire) dans l'ensemble des clusters où : $Sim\ [(a),(b)] = MAX\ Sim[(i),(j)]$
4. $m=m+1$, fusionner les clusters (a) et (b) dans un seul cluster pour former le clustering m suivant. Mettre le niveau de ce clustering à $L(m)=Sim[(a),(b)]$.

5. Modifier la matrice de similarité, en supprimant la ligne et la colonne correspondante aux clusters (a) et (b) et ajouter une ligne et une colonne correspondant à nouveau cluster. La similarité entre le nouveau cluster, dénoté (a,b) et le cluster ancien (k) est défini par :
   Sim[(k),(a,b)] = MAX {Sim[(k), (a)], Sim [(k), (b)]};

6. Si tous les clusters sont dans un seul cluster, arrêter. Sinon, aller à l'étape3 [74], [79].

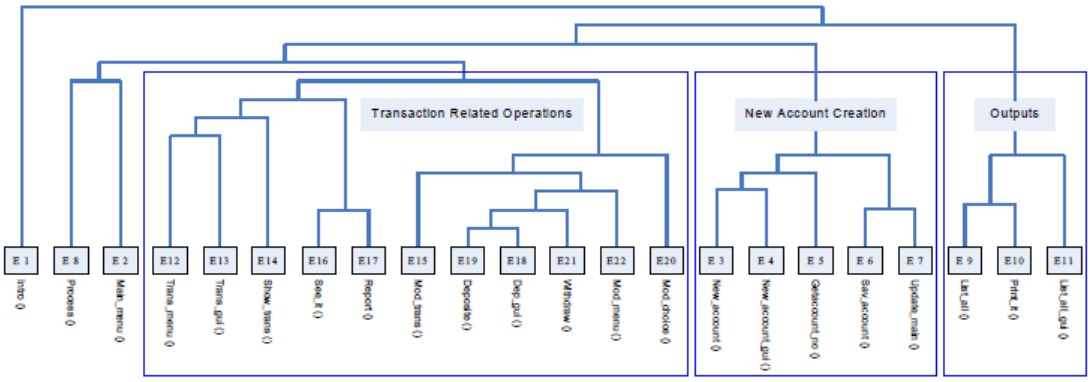


Figure III.2 Exemple de dendrogramme [74]

La figure [IV.2] ci-dessus représente le résultat d'analyse de clustering d'un système patrimonial. Cette méthode de classification, ainsi que la supervision de l'homme, fournit une analyse puissante du système existant, et le représente dans des modules de couplage faible et à grain grossier. A la fin de cette analyse nous bénéficions l'extraction du code des fonctionnalités pertinentes.

7. Création des ontologies

Les ontologies sur les quelles nous nous basons sont l'ontologie de domaine et l'ontologie contextuelle .la première est en raison d'avoir une définition commune d'une information dans un domaine et la deuxième est pour englober le changement de sens d'une information en fonction de son contexte.

7.1 Ontologie de domaine

Nous utilisons cette ontologie pour constituer un accord concernant les noms des concepts sémantiques utilisés, bien comprendre le rapport entre les concepts liés au système et englober les paramètres des services web. Nous résumons les étapes a suivre pour construire une ontologie de domaine dans ces points suivant :

1. Déterminez la couverture et domaine d'ontologie, cela devrait aider pour créer une vision claire du cadre d'ontologie et de son usage.
2. Identifiez la terminologie de l'ontologie.
3. Créez des classes et leur hiérarchie
4. Définissez des propriétés de l'objet. Ces propriétés décrivent la structure interne de concepts par clarifier leurs propriétés extrinsèques (ex., nom, durée et usage), intrinsèque propriétés (ex., poids)
5. Définissez les propriétés des données telles que le type de la valeur de son emplacement.
6. Implémentez l'ontologie. Cette étape utilise un environnement du développement de l'ontologie où nous utilisons un outil très connu et aisé 'protégé 2000'.

La [figure III.3] si dessous illustre un exemple d'ontologie de domaine de l'apprentissage pédagogique :

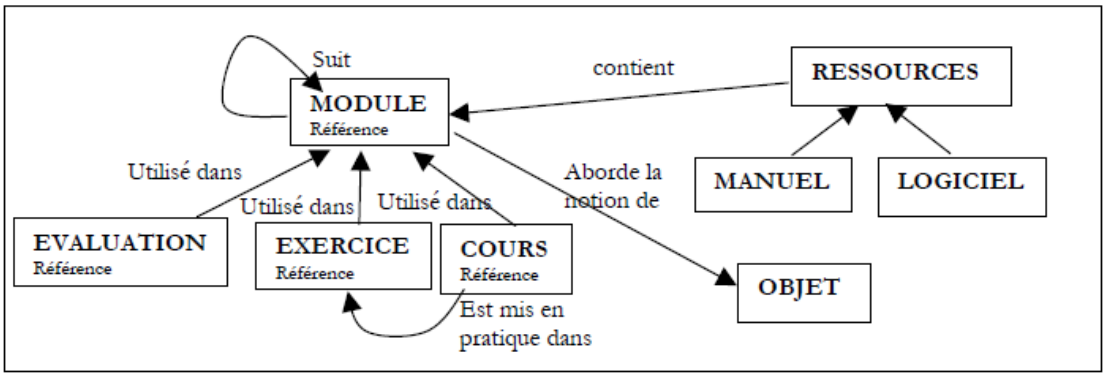


Figure III.3 exemple d'ontologie de domaine

Nous proposons dans cette étape, de faire du reverse ingénierie au code patrimonial vers un modèle métier UML afin de générer une ontologie de domaine. Le choix d'UML pour décrire des ontologies pourrait paraître comme une alternative

beaucoup plus simple. En observant rapidement les langages de description d'ontologies et UML on peut constater l'existence d'un certain nombre de notions communes : classes, relations, propriétés, héritage… Néanmoins, il existe plusieurs différences significatives entre ces concepts, La plus importante porte sur la notion de propriété, dans UML un attribut a une portée liée à la classe, à la différence des ontologies dans lesquelles une propriété représente un concept de premier niveau pouvant exister indépendamment d'une classe.

7.2 Ontologie contextuelle

Nous utilisons Cette ontologie parce qu'elle permet de décrire le contexte de chaque concept de l'ontologie de domaine. Ceci nous permet de décrire d'une façon claire et aisée les hétérogénéités sémantiques.

Pour arriver à de meilleurs résultats à l'issue de cette phase, beaucoup de travaux [12], [39] ont proposé d'inclure l'expérience des utilisateurs tels qu'ils sont les seuls qui connaissent bien leurs systèmes. A partir de leur expérience dans l'utilisation de ce système patrimonial et d'autres différents systèmes, ils ont formé une idée très claire sur les fonctionnalités qui doivent être réutilisées.

8. Evaluation des services web résultants

Comme le service est une composante à grain grossié avec fonction de métier indépendante, une granularité optimale est la clé d'un service bien conçu. Afin de parvenir à une granularité optimale, la rationalisation et la consolidation des services doit être considérés lors de l'identification du service et son adaptation.

La rationalisation des services implique une analyse vigilante de tous les composants et les applications assurant la fonction métier déterminée. Grâce à la rationalisation des services, des fonctionnalités d'entreprise prises en charge par les composants les moins fréquemment utilisés peuvent être mis en œuvre au sein de ceux qui sont plus fréquemment consultés.

La consolidation de service implique une redéfinition de toutes les instances de service dans une version consolidée qui prend en charge l'ensemble de toutes les interfaces exposées par les instances individuelles. La consolidation des services est un moyen efficace pour rationaliser les services multiples qui prennent en charge la même fonction métier.

9. Adaptation et Publication

La phase d'adaptation et de publication est présente dans les différentes approches puisqu'elle est nécessaire pour qualifier le code extrait pour être un service Web publique. Elle est composée de plusieurs étapes comme est le cas de plusieurs travaux :

9.1 Raffinement du code extrait

Bien que le code existant soit indépendant et faiblement couplé, il peut encore avoir des relations couplées avec d'autres entités. Le raffinement du code extrait est faite pour éliminer les répétitions, le code inutile et préciser les détails de l'interface du service afin de garantir que le futur service Web sera indépendant et de granularité fine et faiblement couplé avec les autres services.

9.2Le développement des nouvelles fonctionnalités

Les nouveaux besoins de bisness, dévoilés dans l'étape d'identification de service nous indique la nécessité de construire des nouveaux composants. Ces nouveaux composants sont conçus en fonction de la différence entre le modèle de domaine obtenu à partir de l'analyse de domaine et l'architecture récupéré de l'ancien code. Ils seront intégrés dans le service cible pour étendre la logique métier [80].

9.3 Description des services en WSDL

Une interface de service est la limite abstraite qu'un service expose. Chaque service expose une interface qui définit les types de messages et les modes de leurs échanges qui sont impliqués dans l'interaction avec le service, ainsi que les conditions sous-entendus par ces messages. Pour cela, il faut spécifier une interface bien précise pour chaque service extrait. Ensuite, nous passons à une définition très rigoureuse des fonctionnalités des services Web en WSDL. Un service doit avoir une description suffisante pour qu'il puisse êtres utilisé par d'autres parties. Dans I' idéale, une description de service fournira des informations suffisantes pour qu'un agent automatique puisse non seulement utiliser le service, mais également décider si le service est approprié [80], [12].

9.4 Mécanisme d'intégration de transport

Le mécanisme de transport est nécessaire pour un service en vue de sa nature orienté message. Pour mettre en œuvre un service Web, un processeur SOAP sera intégré. Il prend en charge le transfert des messages sur le Web à travers le firewall (pare-feu).

9.5 Intégration des Services

Au cours du processus d'intégration des services, les composants existants et les nouveaux construits sont interconnectés par des connecteurs afin de construire le service cible. Ces connecteurs sont principalement du code collé et des wrappers pour l'emballage de service.

10. Annotation sémantique des descriptions des services web

Nous proposons d'annoter les descriptions des services avec une ontologie. Nous utilisons dans notre cas une ontologie contextuelle et une ontologie de domaine exprimée en owl, générée à partir d'un modèle métier UML reflétant le système patrimonial en question.

Une annotation sémantique dans un document présente les informations supplémentaires qui identifient ou définissent un concept dans un modèle sémantique pour décrire la partie de ce document [37]. « L'annotation sémantique est un cas particulier d'annotation parce qu'elle fait référence à une ontologie. » d'après N. Zouggar et all [41]. Dans le cadre du Web Sémantique, une *annotation sémantique* réfère le plus souvent à l'annotation descriptive qui s'intéresse à la structure logique du contenu d'un document. Le terme « sémantique » est lui-même ambigu mais il indique une volonté de faire émerger le sens d'un contenu d'une manière formelle selon les préceptes de la logique [97]. L'objectif des annotations sémantiques, comme le souligne Corcho [98], est d'exprimer la « sémantique » du contenu d'une ressource afin d'améliorer sa compréhension, sa recherche et donc sa réutilisation par les utilisateurs finaux

Dans notre cas, l'annotation est faite par SAWSDL (Semantic Annotation for Web Description langage 'WSDL'), un des récents standards du Word Wide Web consortiumW3C. SAWSDL est une extension simple de WSDL et utilise des

éléments d'extensibilité [42], [40]. L'objectif de SAWSDL est d'ajouter de la sémantique à la description WSDL des services et des schémas XML [44].

Dans SAWSDL, les annotations sémantiques sont des attributs XML ajoutés au WSDL ou associés au document Schéma XML, à l'élément XML qu'ils décrivent. Il ya deux types de base d'annotations [37] :

1. les identificateurs explicites des concepts 'referencemodel'

2. les identificateurs du mapping de WSDL aux concepts et vice-versa (mapping du schema).

10.1 Mécanisme d'Annotation

Dans cette partie, on va introduire les deux annotations sémantiques de base fournies par SAWSDL. Après on va expliquer comment ils peuvent être appliqués au WSDL et aux documents Schéma XML. Le Word Wide Web consortiumW3C définit comment ajouter des annotations sémantiques aux différents composants d'un document WSDL2.0 (et supporte aussi WSDL1.1) tels que les messages d'entrée et de sortie, les interfaces et les opérations. Le SAWSDL définit aussi un mécanisme d'annotation pour spécifier les mapping syntaxique des types de schémas XML de et vers un modèle sémantique. Cette spécification se concentre sémantiquement sur l'annotation de la définition abstraite des services, mais pas sur celle de l'implémentation, pour permettre la découverte dynamique, la composition, l'invocation des services et sur tout éviter le problème d'hétérogénéité sémantique [42]. Pour accomplir l'annotation sémantique SAWSDL propose deux sortes d'annotations sémantiques : une première pour identifier le concept sémantique représentée par l'attribut modelReference et une seconde pour faire le lien entre le concept et le document WSDL représentée par les attributs liftingSchemaMapping et loweringSchemaMapping.

a. SAWSDL 'Model Reference'

L'attribut modelReference spécifie l'association entre un WSDL ou un composant Schéma XML et un concept dans un modèle sémantique. SAWSDL définit cet attribut pour annoter les différentes parties du document de description

wsdl :interface, wsdl :operation, wsdl :fault, xs :element, xs :complexType, xs :simpleType et xs :attribute [37].

Cependant, la recommandation de SAWSDL considère que l'annotation des éléments suivants apporte une signification particulière [47]:

- **L'élément interface**. Cet élément décrit l'ensemble des méthodes disponibles d'un service Web. L'annotation de cet élément apporte une définition sémantique de l'ensemble des méthodes proposées.
- **L'élément opération**. L'élément opération décrit une méthode particulière. Cette annotation permet d'expliciter les effets des méthodes proposées par le service.
- **L'élément fault**. Cet élément représente un message d'erreur lors de l'appel d'une méthode de l'interface. L'annotation de ce type d'élément donne un sens au message d'erreur.
- **L'élément type**. Cet élément représente les structures de données des paramètres dans des messages échangés. Si cet élément requiert des spécifications exprimées en *XML Schema* pour définir des types particuliers (simple ou complexe), l'annotation sémantique de SAWSDL peut être aussi utilisée dans cet élément afin d'annoter sémantiquement ces structures de données. Un attribut modelReference a pour valeur zéro, une ou plusieurs URI localisant des ontologies qui apportent de la sémantique aux éléments annotés. Ce type d'annotation est utilisé lors des processus de recherche et de sélection. Si le concept sémantique est décrit en format XML, le code peut être directement intégré dans le fichier WSDL.

L'extrait de schéma XML suivant définit l'attribut modelReference.

```
…
<xs:attribute name="modelReference" type="listOfAnyURI" />
<xs:simpleType name="listOfAnyURI">
   <xs:list itemType="xs:anyURI"/>
</xs:simpleType>
…
```

b. SAWSDL' Schema Mapping'

Deux autres attributs, nommés liftingSchemaMapping et loweringSchemaMapping, étendent le fichier WSDL afin de spécifier l'association entre la donnée sémantique (le concept) et l'élément WSDL à annoter et réciproquement. Ces mises en correspondance sont faites à l'aide de schémas XML (dont la localisation est donnée par la valeur de l'attribut). L'attribut liftingSchemaMapping associe aux éléments XML un modèle sémantique. Inversement, l'attribut loweringSchemaMapping associe au modèle sémantique une structure XML. Ces mises en correspondance sont utiles, lors de l'invocation du s ervice, lorsque la structure de données ne correspond pas de manière intuitive à l'organisation du modèle sémantique. L'utilisation de ces attributs (liftingSchemaMapping et loweringSchemaMapping) n'est pas illustrée ici du fait que dans notre travail nous ne nous préoccupons pas de la cohérence structurelle des données. Nous considérons que l'étape de sélection, préalable à l'étape d'invocation, vérifie la cohérence de la structure des données échangées [47].

```
…

<xs:attribute name="liftingSchemaMapping" type="listOfAnyURI" />

<xs:attribute name="loweringSchemaMapping" type="listOfAnyURI" />
<xs:simpleType name="listOfAnyURI">
   <xs:list itemType="xs:anyURI"/>
</xs:simpleType>
…
```

11.Publication

Une organisation a besoin de publier les Web Services qu'elle possède, pour d'autres organisations à fin qu'elles puissent les découvrir. UDDI est utilisé pour publier les Web Services sur un dépôt central UDDI. La publication, dans notre cas, concerne les ontologies et les services web annotés. Nous entamons la publication des ontologies dans un référentiel sous forme de base de données (mySQL) suivie de la publication des descriptions des services web ou nous utilisons le référentiel UDDI3.0.

12.Discussion

Dans cette section nous discutons ce que notre approche a ramené de nouveaux par a port aux deux approches celle de S. Alhamari et de J. Zhi.

Notre approche commence par une évaluation du système qui est essentiellement extraite à partir l'approche de J. Zhi li et all [74]. Cette phase étudie la faisabilité de la migration. Nous avons proposé cette phase au démarrage du processus, qui est le cas de plusieurs travaux, pour éviter la perte d'effort si le résultat de cette phase est négatif.

La deuxième phase de notre approche nous fournit le repère théorique des futurs services Web, sur lequel nous nous basons pour détecter les services Web potentiels dans notre système patrimonial. Apres nous proposons de générer le model du domaine UML via la technique reverse ingineering a fin de décrire la structure fondamentale du système. Et nous proposons d'évaluer la granularité des services résultants.

A la phase d'extraction du code nous proposons d'utilisé une méthode de clusternig extraite a partir de l'approche de J. Zhi [74] .cette phase a pour but d'extraire le code réutilisable indépendamment de tout langage de programmation.

La quatrième phase est le fondement de notre contribution ou nous proposons de générer l'ontologie de domaine et l'ontologie contextuelle .ces dernières nous aident a bien préciser une définition commune des informations dans un domaine et mettre a jours leurs contextes.

La phase d'adaptation et de publication est présente dans différentes approches. Dans notre cas, nous introduisons notre contribution sous forme d'une étape d'annotation juste après l'adaptation des services web suivie d'une publication sémantique. Cette annotation est réalisée à l'aide du standard SAWSDL en se basant sur les ontologies générées au paravent. Nous avons choisi ce standard en vue de son indépendance de tout langage de représentation d'ontologie.

Comme toute approche, la notre offre des avantages et soufre de quelques inconvénients :

➢ Avantages

- Elimination de l'hétérogénéité sémantique avant la publication nous offre une interopérabilité efficace une fois les services web sont découvert et interrogés par d'autres entreprise.
- L'utilisation de l'algorithme de clustering, appliqué sur un fond mathématique, nous offre des résultats exacts.
- L'utilisation de SAWSDL nous donne la liberté de choix du langage de modélisation d'ontologie et permet de modifier des descriptions WSDL existantes sans trop alourdir le fichier de description.

➢ Inconvénients

- Dans une description WSDL, Trouver les concepts appropriés pour les faire correspondre à des concepts ontologiques, peut être une tâche très fastidieuse, en Vu le nombre de services extraits d'un système patrimonial sont susceptibles d'être énorme.
- L'analyse de clustering est utilisée pour regrouper un grand nombre d'entités ; elle peut extraire les clusters existants et capturer leurs modules a partir du système patrimonial. Mais, cette technique ne peut pas dévoiler les détailles et les relations de chaque module.

13.Conclusion

Nous avons présenté dans ce chapitre notre approche pour l'annotation des services web extraits des systèmes patrimoniaux. Cette approche est inspirée des travaux de S. Alhamari et J. Zhi avec une contribution d'étendre le WSDL en ajoutant les attributs décrits par le standard SAWSDL. Nous procédons a cette annotation sur la base des ontologies générées a l'aide de l'expériences des utilisateurs du système.

Pour la validation de cette approche, nous allons l'appliquer, dans le chapitre suivant, sur un système patrimonial, implémenté en Langage C standard avec des outils simple, où le but est d'annoter un service web présentant une fonctionnalité apparaît utile et réutilisable.

Etude de cas

1. Introduction

Dans le chapitre précédent nous avons exposé notre approche pour l'annotation des services Web méritants d'être publier, extraits à partir des legacy system. Pour cela nous avons proposé l'utilisation du SAWSDL un des récents standards du Word Wide Web consortiumW3C en se basant sur les ontologies.

Dans ce chapitre nous essayons d'appliquer notre approche pour l'annotation des services Web existants dans une application patrimoniale développée en langage C standard. Cette application est implémentée initialement pour une raffinerie, et nous avons remarqué qu'elle contient des fonctionnalités très intéressantes et valables d'être réutilisées autant que services Web.

Ce chapitre commence par l'introduction de notre application patrimoniale, et les raisons argumentants notre choix. Ensuite, nous passons à la description détaillée de la projection des phases de notre approche sur le système à traiter. Nous démarrons ce processus par la phase de spécification des services logiques ou nous proposons de faire du reverse ingénierie au code patrimonial vers un modèle métier UML pour nous aider à générer l'ontologie du domaine.

Après, nous entamons la phase d'extraction du code réutilisable en utilisant l'algorithme de clustering. Nous passons ensuite à la phase d'adaptation où nous transformons le code extrait en services Web concrets. Ces services web vont être annotés par le standard SAWSDL.

Enfin, Nous terminons ce chapitre par une conclusion.

2. Système GSR (Gestion de Stock d'une Raffinerie)

Notre application patrimoniale que nous avons choisie pour illustrer la validité de cette approche est la gestion de stock d'une raffinerie produisant quatre produits finis (essence, kérosène, mazout et résidu) à partir de deux types de pétrole brut (A et B). Ce système reçoit la saisie d'un un numéro de compte client et une liste d'articles à commandés. A chaque commande le client précise la quantité et le code barre du produit (UPC). Le programme rendra le statut de la commande (ordre), qui peut être rejet ou, accepter. Cette

application a été développée en langage C standard. Nous avons choisi cette application puisqu'elle contient des fonctionnalités très intéressantes.

```
#include <stdio.h>
#include <stdlib.h>
#include<stdlib.h>
#define MaxIdentifiant 6 //le nbr maximum de caracteres de l'identifiant
#define MaxAnne 1990
#define MiniAnne 1935
#define AnneCourante 2006
#define qut // est la quantité commandée
#define qutstk // est la quantité stockée
char NomFichier[]="client.txt"; //le fichier ou sera stoke les donnees
typedef struct{
char nom[20];
char prenom[20];
char codepostal;
int date;
int numerocompte ;
char id[MaxIdentifiant];
} client;
typedef struct cellule{
client valeur;
client*suivant;
}cellule;
typedef cellule *liste;
liste InsererTete(liste l,client clt)
{ cellule *nc;
nc=(cellule*)malloc(sizeof(struct cellule));
nc->valeur=clt;
nc->suivant=l;
l=nc;
return(l);
} typedef struct{
char UPC[25];
float punit;
float qutstk;
float qut;
} produit;
 typedef struct cellule1{
produit valeur1;
produit*suivant;
}cellule1;
typedef cellule1 *liste1;
liste1 InsererTete1(liste1 l1,produit pr)
{
cellule1 *np;
np=(cellule1*)malloc(sizeof(struct cellule1));
np->valeur1=pr;
np->suivant=l1;
l1=np;
return(l1);
//************************commande************
void commandeprod ()
{
int p;
client pp ;
produit pr;
float prix;
float punit;
printf(" identifiant ):/t");
scanf("%s",&pp.id[MaxIdentifiant]);
if (ValideIdentfiant(pp.id)!=1)
printf("l'identifiant est erroné");
else
printf("code barre du produit :\t");
scanf("%s",&pr.UPC);
//printf("quantité du produit a commandé:\t");
//scanf("%s",&pr.qut);
If ((qut <= qutstk)&&(validnumcmpt(pp.numerocompte)))
```

```
printf ("la commande est acceptée");
prix == qut* punit;
printf ("le prix est =",prix);
else
printf ("la commande est rejetée");}
}
}
}
//************Supprimer Position*****************
void Supprimeposition(liste l,liste p)
{ liste q,r;
q=l;
while((q->suivant!=NULL)&&(q!=p))
{ r=q;
q=q->suivant;
} r->suivant=q->suivant;
free(q);
return l;
}
//**********Saisie  des clients*********************
Saisieclient()
{
int i;
client pp;
system("cls");
printf("Ajout client\n\n\n");
printf("Saisir les informations suivantes sur le client:\n");
printf("\t\t\tattention:ne faite pas des espaces dans la saisie\n");
do
{  printf("Nom:\t");
scanf("%s",&pp.nom);
if(ValideNom(pp.nom)==-1)
printf("\tNOM INVALIDE\a\n");
}while(ValideNom(pp.nom)==-1);
do
{ printf("Prenom:\t");
scanf("%s",&pp.prenom);
if (ValideNom(pp.prenom)==-1)
printf ("\tPRENOM INVALIDE\a\n");
} while (ValideNom(pp.prenom)==-1);
do
{printf("Sexe (M:male/F:female):\t");
scanf("%s",&pp.sexe);
if ((pp.sexe!='m')&&(pp.sexe!='f')&&(pp.sexe!='M')&&(pp.sexe!='F'))
printf("\tCHOIX INVALIDE\a\n");
}while((pp.sexe!='m')&&(pp.sexe!='f')&&(pp.sexe!='M')&&(pp.sexe!='F'));
system("cls");
do { pp.date=SaisieDate();
printf("\nDate invalide\n");
}while(ValideDate(pp.date)==-1);
system("cls");
printf("Identifiant:\t");
do { Scanf ("%s",&pp.id);
if (ValideIdentfiant(pp.id)!=1)
printf("\t\aidentifiant non valide\nVeuillez entrer un identifiant de la forme P0000:\t");
}while (ValideIdentfiant(pp.id)!=1);
system("cls");
return pp; }
//*********Creer Liste**********************
liste Insererclient(liste l,client e)
{
cellule  *nc,*pc;
nc=(cellule*)malloc(sizeof(struct cellule));
pc=l;
nc->valeur=e;
nc->suivant=l;
```

```
l=nc;
return(l);
}
//************Affiche Liste*****************
void AfficherListe(liste l)
{ liste pc;
pc=l;
if (pc==NULL){system("cls");
printf("\nFichier vide\n");
printf("frapper une touche pour retourner aux
MENU");
getch();}
while(pc!=NULL)
{ int i;
system("cls");
printf("Affichage de tous les clients\n\n\n");
printf("°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
printf("%s\t",pc->valeur.nom);
printf("%s\t(nom prenom)\n",pc-
>valeur.prenom);
printf("%c\t(Sexe)\n",pc->valeur.sexe);
printf("%d\\%d\\%d\t(date de naissance)\n",pc-
>valeur.date.j,pc->valeur.date.m,pc->valeur.date.a);
printf("%s\t(l'identifiant)\n",pc->valeur.id);
printf("°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°\n");
if(pc->suivant!=NULL)
{ printf("\nfrapper une touche pour voir la
personne suivant\n\n");
getch();
system("cls");
} if(pc->suivant==NULL)
{ printf("frapper une touche pour retourner aux
MENU");
getch();
system("cls");
} pc=pc->suivant; }
}
```

```
//*********Supprimer client***************
liste Supprimerclient(liste l)
{ liste p,pre;
char ide[6], nom[25];
p=l;
system("cls");
printf("\nsupprimer client\n");
do
{
printf("nom de client a supprimer:\t");
scanf("%s",&nom);
if(ValideNom(nom)==-1)
printf("\tNOM INVALIDE\a\n");
}while(ValideNom(nom)==-1);
printf("identifiant :\t");
do { scanf("%s",&ide);
if(ValideIdentfiant(ide)!=1)
printf("\t\aIdentifiant non valide\nveuillez
entrer un identifiannt de la forme P0000:\t");
}while (ValideIdentfiant(ide)!=1);
while ((p!=NULL)&&((strcmp(p-
>valeur.id,ide)!=0)||strcmp(p->valeur.nom,nom)!=0))
{ clt =p;
p = p->suivant;
}; If (p==NULL)
{ printf("\npersonne introuve\n");
return l;
}If (p==l)
{ l=l->suivant;
free(p);
printf("\nFiche supprime\n\n");
return l;
} clt->suivant=p->suivant;
free(p);
printf("\nFiche supprime\n\n");
return l;}
```

```
//********* Tri Par Nom ******************
            liste TriNom(liste l)
            { client aux;
            liste p,q;
            p=l;
            if (l==NULL)
            return l;
            system("cls");
            while(p!=NULL)
            { q=p->suivant;
            while(q!=NULL)
            { if(strcmp(p->valeur.nom,q->valeur.nom)>0)
            { aux=q->valeur;
            q->valeur=p->valeur;
            p->valeur=aux;
            } q=q->suivant;
            } p=p->suivant;
            }return l;
            }
//**************Lire fichier****************
            liste LireFichier(liste l)
            { cellule *pc;
            FILE *fichier;
            int i;
            fichier=fopen(NomFichier,"r");
            if(!fichier)
            {system("color 1f");
printf("le fichier:'personne.txt' n'existe pas ajoute client
pour creer le fichier\n");
printf("\nfrapper une touche pour voir le MENU\n");
            getch();
            return(l);
            } while(!feof(fichier))
            { pc=(cellule *)malloc(sizeof(cellule));
            while(true)
            { fscanf(fichier,"%d\n",&pc->valeur.nom);
            fscanf(fichier,"%s\n",&pc->valeur.nom);
            fscanf(fichier,"%s\n",&pc->valeur.prenom);
            fscanf(fichier,"%s\n",&pc->valeur.sexe);
            fscanf(fichier,"%d\n",&pc->valeur.date.j);
            fscanf(fichier,"%d\n",&pc->valeur.date.m);
            fscanf(fichier,"%d\n",&pc->valeur.date.a);
            fscanf(fichier,"%s\n",&pc->valeur.id);
            fscanf(fichier,"%d\n",&pc-
>valeur.numcompte);
            } pc->suivant=l;
            l=pc;
            }; fclose(fichier);
            return(l); }
//*****PROGRAMME PRINCIPALE**************
            void main()
            { liste l=CreerListeVide();
            char choix,c;
            int i;
char chaine[]="AUTEURS : M^ed  IA2C1 avril 2002";
            client p;
            l=LireFichier(l);
            l=TriNom(l);
            do
            { system("color 1f");
            system("cls");
            printf("\t\t\tGESTION de stock\n");
            printf("\n\t\t\t\tMENU\n");
            printf("\t\t ^^^^^^^^^^^^^^^^^^^^^^^^^^\n");
            printf("Ajouter un client     ---> 1\n");
            printf("Consulter tous les fiches---> 2\n");
            printf("Afficher client       ---> 3\n");
            printf("Ajouter produit         ---> 4\n");
            printf("Supprimer client      ---> 5\n");
```

```
printf("Tri par identifiant      ---> 6\n");
printf("Mise ajour  ---> 9\n");
printf("Supprimer tout           ---> S\n");
printf("A propos                 ---> h\n\n");
printf("Quitter                  ---> Q\n\n");
printf("Votre Choix:\t");
choix =getch();
switch(choix)
{ case '1':p=SaisieClient();
l=InsererClient(l,p);
l=TriNom(l);
EcrireFichier(l);
printf("frapper une touche pour retourner aux MENU");
break;
case '2':AfficherListe(l);
break;
case '3':system("cls");
l=AfficherClient(l);
printf("frapper une touche pour retourner aux MENU");
getch();
break;
EcrireFichier(l);
printf("frapper une touche pour retourner aux MENU");
getch();
break;
case '5':l=SupprimerClient(l);
printf("frapper une touche pour retourner aux MENU");
getch();
EcrireFichier(l);
break;
case '6':system("cls");
printf("\nTRI PAR IDENTIFIAN\n");
l=TriIdentifiant(l);
printf("\n\nLISTE TRIE PAR IDENTIFIAN\n\n");
printf("tapez une touche pour voir la liste trie\n");
getch();
EcrireFichier(l);
AfficherListe(l);
break;
case 'h':     system("cls");
printf("A Propos\n\n");
for(i=0;i<60;i++)
{ printf("%c\a",chaine[i]);
} printf("\n");
//printf("\t\t\tAUTEURS\nmadileila \n\t\tIA2C1\nfevrier 2011\n\n\n\n");
printf("frapper une touche pour retourner aux MENU");
getch();
break;
case 'S':system("cls");
system("color c0");
printf("\n ETES VOUS SUR DE SUPPRIMER TOUTE LA LISTE(o\\n)?\t");
c=getch();
if (c=='o')
{ l=CreerListeVide();
EcrireFichier(l);
printf("\nSupprission effectue\n");
printf("\nfrapper une touche pour retourner aux MENU\n");
getch();}
system("cls");
break;
case 's':system("cls");
system("color c0");
printf("\n ETES VOUS SUR DE SUPPRIMER TOUTE LA LISTE(o\\n)?\t");
c=getch();
if (c=='o')
{ l=CreerListeVide();
EcrireFichier(l);
```

```
printf("\nSupprission effectue\n");

printf("\nfrapper une touche pour retourner aux MENU\n");

getch(); }
```

```
system("cls");

break;}

} while ((choix!='q') && (choix!='Q'));

printf("\n\nFermeture\n\n");}
```

3. Spécification des services web

Cette étape commence par une étude du Legacy système, l'application d'achat de produit englobe beaucoup de fonctionnalités qui peuvent être réutilisé comme des services Web. Parmi ces fonctionnalités nous citons par exemples :

- Les petits services Web : ce type des services Web peut être définis à partir des fonctions qui composent le système, comme :

  1. calcul du prix de vente unitaire d'un produit
  2. facture de payement
  3. Etablissement de contrat

  …etc

- Les services Web qui ont des tâches bien spécifiques comme :
  1. Etablissement du devis (descriptif du produit plus cout)
  2. Etablissement d'ordre de la commande qui a comme résultat rejeter ou accepter.
  3. ...etc.

4. Détection du code des services web

L'approche de clustering que nous avons adopté est une approche ascendante (*bottom-up*) consisterait à partir de petits groupes, et à les améliorer en y insérant des entités. Il est par exemple envisageable de commencer avec des groupes contenant chacun une des entités (fonction, procedure) de système étudié, et de réunir des groupes entre eux par étapes successives, si leur réunion améliore la « qualité » du groupe. Si une entité reste à l'écart ça veut dire qu'elle ne forme pas un bon groupe avec les autres entités. D'une autre manière le code représentant cette entité est a ignoré et ne fait pas partie du code réutilisable.

Pour réaliser cette opération de regroupement, on fait fréquemment appel à la notion de similarité entre les entités. En effet, cette notion de similarité prend tout son sens en clustering car il s'agit d'évaluer à quel point deux éléments sont similaires (ou dissimilaires) pour les regrouper ou les séparer. Le choix de la mesure de similarité permettant de comparer les entités entre eux et va induire la façon de les regrouper.

Pour faciliter et donner une bonne représentation des graphes nous procédons à la notation suivante

- Nous dénotons les fonctions par : E_i telque i $\in \{1..6\}$.
- Nous dénotons les attributs des fonctions par $\{a, b, c, d, e\}$.

Tableau IV.1 Notation des différentes fonctions existantes dans le code source

Entité (fonction)	**notation**
Commandeproduit	E_1
Saisieclient	E_2
Supprimerposition	E_3
Insererclient	E_4
Affricheliste	E_5
supprimerclient	E_6
Lirefichier	E_7

Tableau IV.2 Notation des différents attributs liés aux différentes fonctions existantes

Attribut	**Notation**
Produit (pr)	a
Client (pp)	b
Liste1(L)	c
prix	d
Quantité (qut)	e
fichier	f

4.1 Propriétés à respecter dans le regroupement

Voici quelques propriétés intuitives qui devront être respectées par une mesure de similarité. Pour deux entités *X* et Y

- La similarité entre *X* et Y est en fonction de ce qu'ils ont en commun. Plus ils ont d'attributs en commun, plus leur similarité sera élevée. (P1)
- La similarité entre *X* et *Y* est en fonction de leurs différences. Plus ils ont de différences, plus leur similarité sera faible. (P2)
- La valeur de similarité maximale est obtenue lorsque *X* et *B* sont identiques, quel que soit leur nombre d'éléments communs. (P3)

Pour calculer la similarité, On utilise dans notre cas le coefficient de Jaccard [81], qui se calcule selon la formule:

$$SIM(x, y) = \frac{\boldsymbol{card}\ (\boldsymbol{Sx} \cap \boldsymbol{Sy})}{\boldsymbol{card}\ (\boldsymbol{Sx} \cup \boldsymbol{Sy})}$$

Tel que :

X, Y : dénotent les entités (fonction ou procédure).

S : dénote les attributs de X ou de Y.

Avant de commencer l'algorithme de clustering nous présentons les attributs liés à chaque fonction dans le tableau suivant :

Tableau IV.3 les attributs liés a chaque entité

Entité (Fonction)	Attributs
E_1	a,b,d,e
E_2	b
E_3	c
E_4	b,c
E_5	c
E_6	b,c
E_7	b,c,f

4.2 Application de l'algorithme

Etape 1

Nous formons des clusters tels que chaque cluster comporte une seule entité (fonction ou procédure) séparément se qui donne :

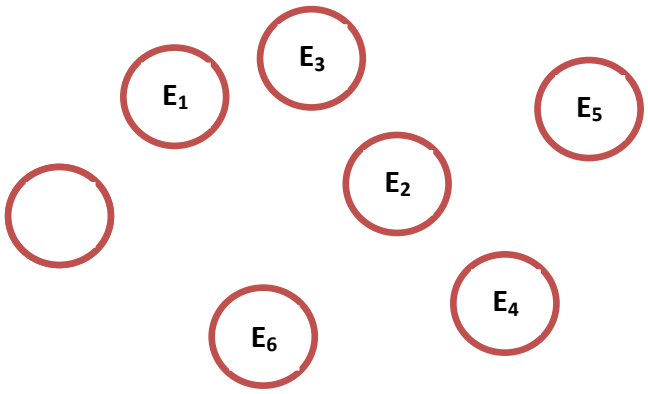


Figure IV.1 Représentation des entités chacune dans un cluster séparé

Etape2

Nous choisissons deux clusters les plus proches et nous étudions leur similarité. Par exemple nous commençons par l'entité **E_2** (saisieclient) et **E_4** (insererclient) ou nous interceptons la présence d'une relation entre eux qui se manifeste par l'attribut (b) en commun comme est représenté dans la figure si dessous.

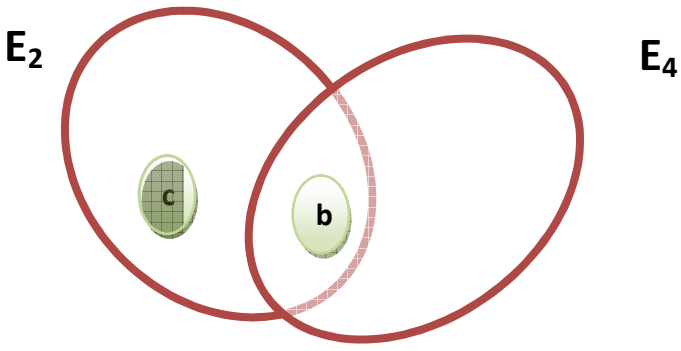


Figure IV.2 : la relation existante entre les entités E_2 et E_4

Etape 3

Dans cette étape nous entamons le calcul de la similarité entre $\mathbf{E_2}$ et $\mathbf{E_4}$. Sur la base de résultat de calcul, nous décidons de procéder à la fusion ou non de ces deux entités.

SIM (E_2, E_4) = ($card$ S_{E2} ∩ Card S_{E4}) / ($card$ S_{E2} ∪ Card S_{E4})

SIM (E_2, E_4) = 1 /2

Le résultat de calcul de la similarité entre les entités $\mathbf{E_2}$ et $\mathbf{E_4}$ indique qu'ils sont similaire, d'où la naissance d'un nouveau cluster $\mathbf{E_{24}}$ en procédant à la fusion de ces deux clusters $\mathbf{E_2}$ et $\mathbf{E_4}$.

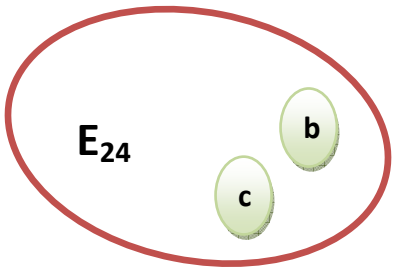


Figure IV.3 la fusion des deux clusters E_2 et E_4

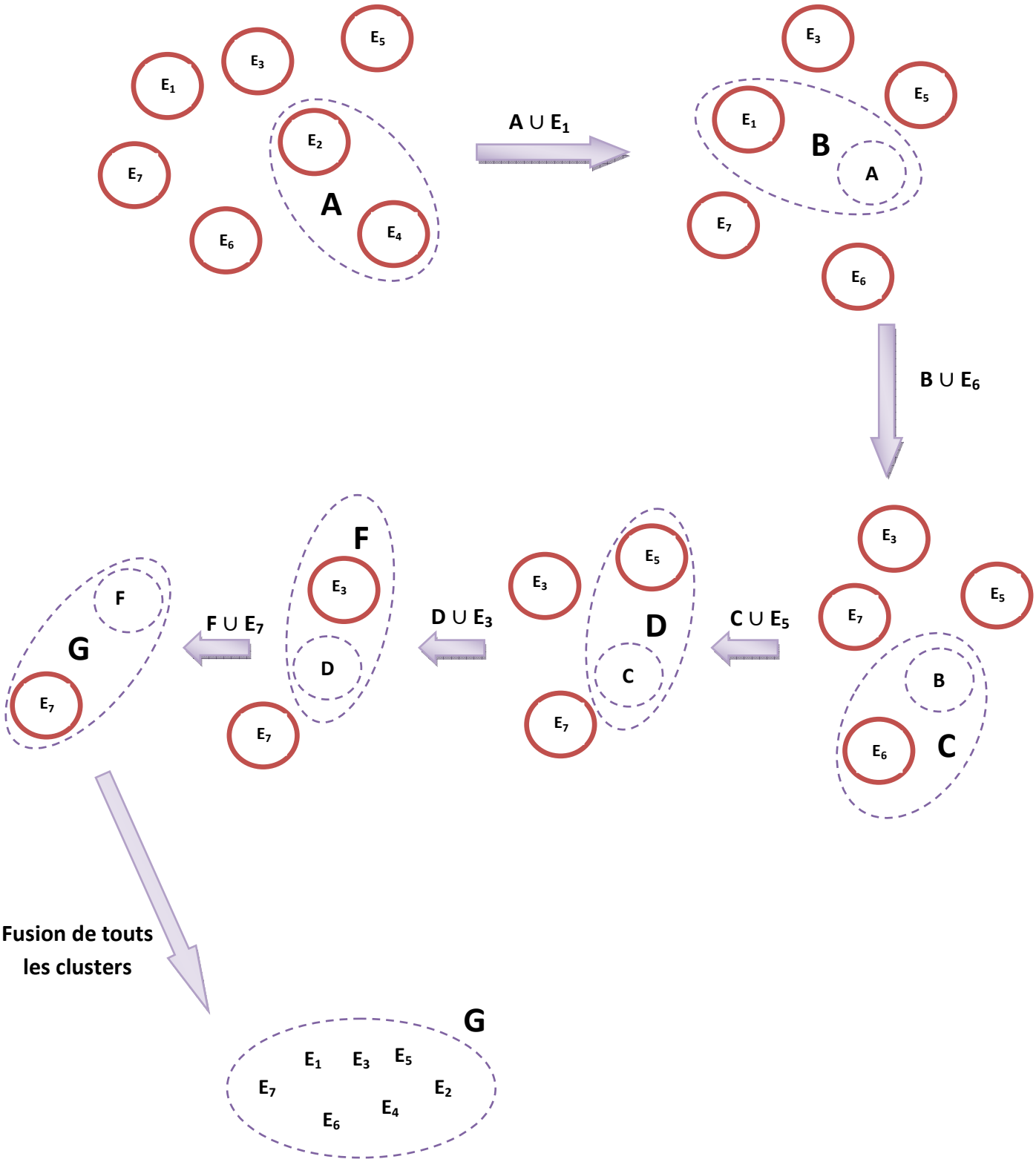
Etape4

Maintenant nous cherchons le cluster le plus proche au nouveau cluster $\mathbf{E_{24}}$ et nous calculons leur similarité sur la quelle se base la décision de la fusion ou la non fusion. Si la décision est pour la fusion, un nouveau cluster englobant $\mathbf{E_{24}}$ **et** $\mathbf{E_i}$ est créé. Ce processus se reproduit jusqu'à ce que toutes les entités soient dans un seul cluster. Le tableau suivant illustre le calcul de similarité entre les différents clusters :

Tableau IV.4 Calcul de similarité entre les différents groupes

Clusters candidats	$S_{E2} \cap S_{E4}$	$S_{E2} \cup S_{E4}$	Similarité	Fusion	Cluster résultant
E_2 et E_4	1	2	0.5	oui	E_{24} =A
A et E_1	1	4	0.25	oui	E_{124}=B
B et E_6	1	5	0.2	oui	E_{1246}=C
C et E_5	1	5	0.2	oui	E_{12456}=D
D et E_3	1	5	0.2	oui	E_{123456}=F
F et E_7	2	6	0.33	oui	$E_{1234567}$=G

➢ **Interprétation des résultats du tableau**



Résultats :

D'après cette illustration nous remarquons :

- il n'y a pas un cluster ignoré alors il n'ya pas du code à éliminer.
- il ya une relation et un degré de similarité entre les différents clusters alors le code de toute les fonctions étudiées est réutilisable.

5. Création d'ontologie

Dans cette étape, nous proposons la création d'ontologie ou nous utilisons un outil dite protegé 4.1 dont son environnement est présenté par la figure ci-dessous :

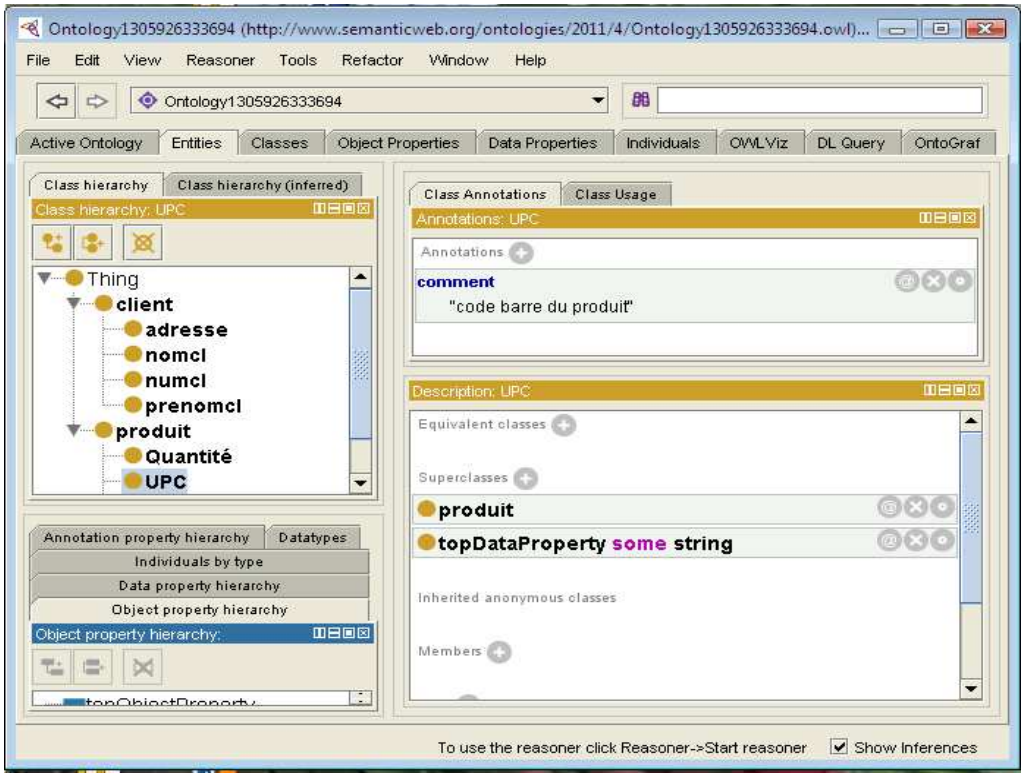


Figure IV.4 : Editeur protégé 4.1

Dans ce qui suit nous présentons les ontologies locales de données de la GSR. Ces ontologies véhiculent la sémantique locale.

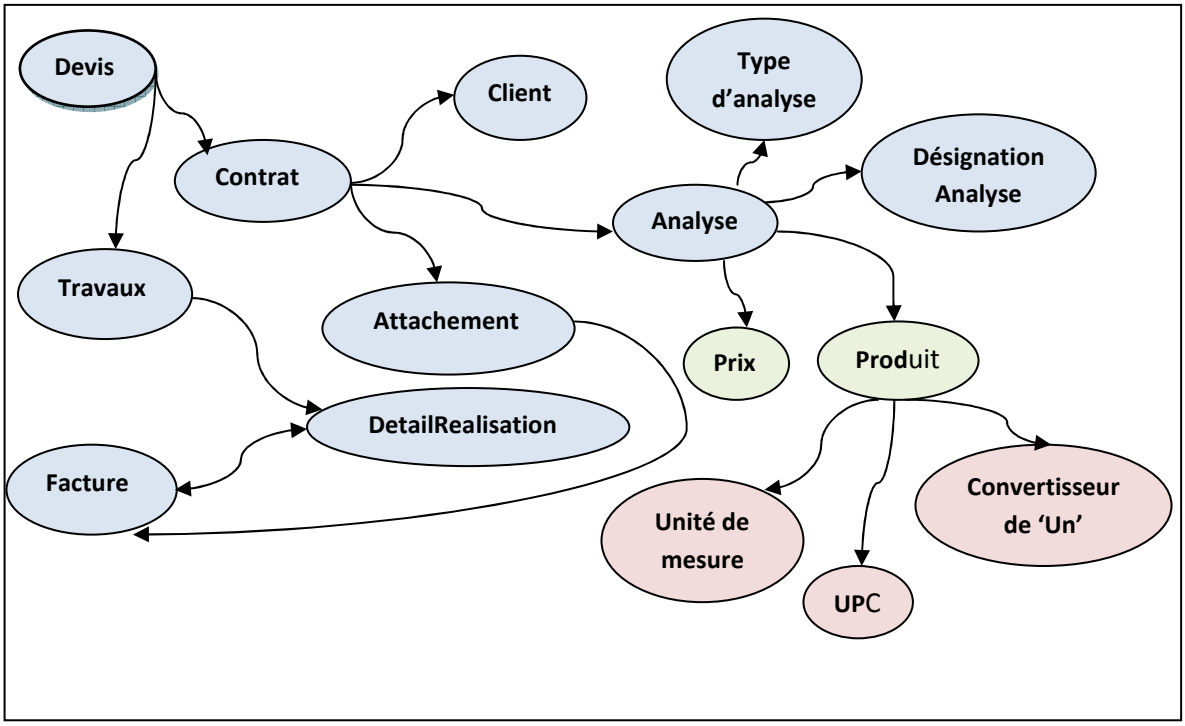


Figure IV.5 Extrait de l'ontologie locale de données de la GSR

Nous prenons, dans la figure ci-dessous, comme exemple le concept 'produit' pour illustrer l'ontologie contextuelle. Cette ontologie est mise a jour grâce a la sémantique locale de l'ontologie de domaine adhérée par le fournisseur.

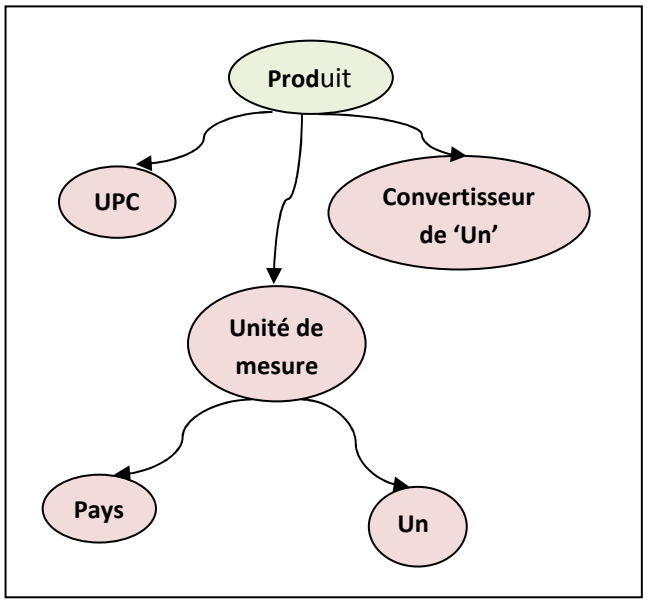


Figure IV.6 Ontologie contextuelle de données associées au concept 'produit'

6. Wrapping et Publication

Après la détection du code des futurs services Web, nous entamons la phase la phase de wrapping et publication qui est composée d'un ensemble des sous étapes, pour avoir à la fin un service web concret qui peut être publié ou même être réutilisé dans un autre système orienté service. On peut commencer, comme le cas de beaucoup de travaux, par le raffinement de code qui englobe l'élimination des redondances, minimisation de l'interactivité avec l'extérieur…Mais pour notre cas le code ne réclame pas d'intervention parce qu'il est dans son état raffiné. Dans ce cas on passe directement à l'étape de la spécification d'interface de chaque service.

6.1 Description de service en WSDL

Dans cette étape, nous choisissons une des fonctionnalités qu'on juge pertinente nommée commande et nous spécifions son interface.

```
<wsdl:description
 targetNamespace="http://www.w3.org/2002/ws/spec/wsdl/order#"
 xmlns="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
 xmlns:wsdl="http://www.w3.org/ns/wsdl"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 <wsdl:types>
  <xs:schema targetNamespace="http://www.w3.org/2002/ws/spec/wsdl/order#"
   elementFormDefault="qualified">
   <xs:element name="OrderRequest"
    <xs:complexType>
     <xs:sequence>
      <xs:element name="numclient" type="xs:integer" />
      <xs:element name="codepostal" type="xs :integer"/>

      <xs:element name="orderItem" type="item"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:complexType name="item">
    <xs:all>
     <xs:element name="UPC" type="xs:string" />
    </xs:all>
    <xs:attribute name="quantité" type="xs:integer" />
    <xs :attribute name="Un" type="xs:string" />
   <xs:attribute name="prix" type="xs:float" />
   </xs:complexType>
   <xs:element name="OrderResponse" type="confirmation" />
   <xs:simpleType name="confirmation"
    <xs:restriction base="xs:string">
     <xs:enumeration value="acceptée" />
     <xs:enumeration value="Rejetée" />
    </xs:restriction>
```

```
36       </xs:simpleType>
37     </xs:schema>
38   </wsdl:types>
39   <wsdl:interface name="Order"
40       <wsdl:input element=" OrderRequest " />
41       <wsdl:output element="OrderResponse" />
42     </wsdl:operation>
43   </wsdl:interface>
44 </wsdl:description>
```

6.2 Annotation sémantique des descriptions des services web

Une fois les services extraits de notre système patrimonial sont publiés, ils peuvent être interrogés par d'autres entreprises. En général, Les applications de ces dernières sont développées sur des plateformes matérielles et logicielles différentes et dans des contextes différents. Les informations partagées entre différents domaines d'application proviennent des différences d'interprétation par conséquent plusieurs types de conflits sémantiques apparaissent :

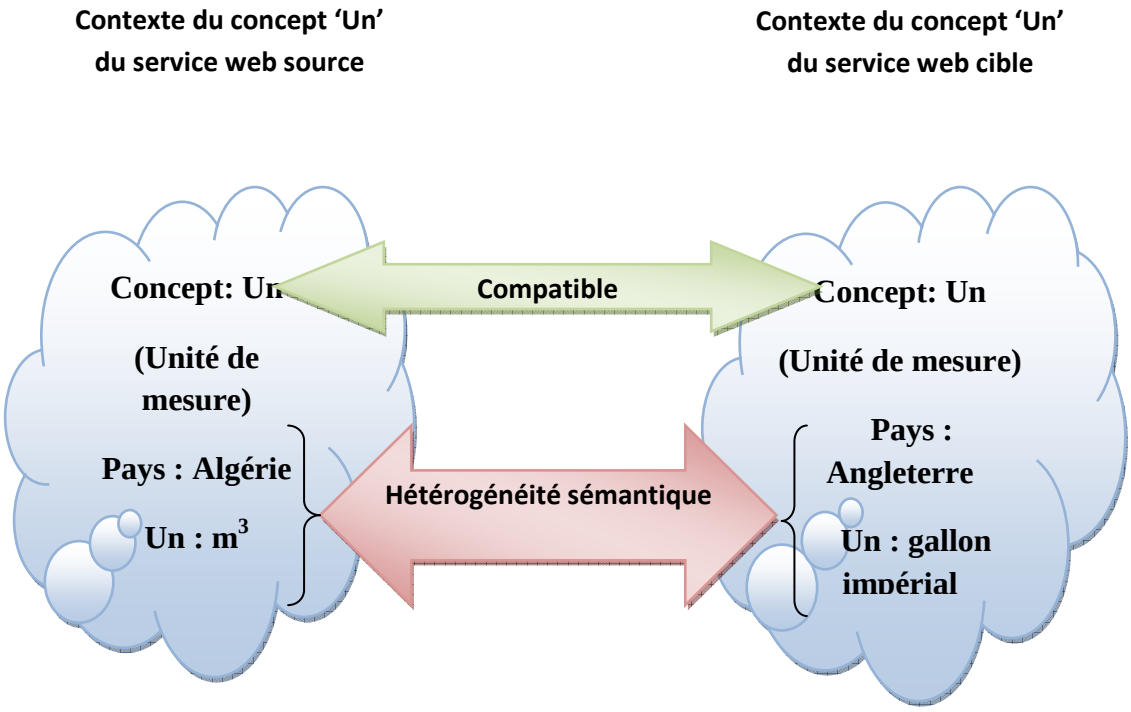
- Conflits de noms : problèmes taxonomiques et linguistiques
- Conflits de valeurs : problèmes d'unités et d'échelle
- Conflits cognitifs : problèmes de signification.

Alors il est nécessaire d'annoter ces services extraits pour empêcher ces conflits de se produire. Dans ce qui suit, nous présentons un exemples d'hétérogénéité de mesure (problèmes d'unités et d'échelle).

- **Exemple**

Le résultat du service commande de notre système sert comme entré pour le service facture d'un système distant (Angleterre). Ce dernier calcul le montant en se basant sur la quantité commandée par notre système. Mais pour achever ces calculs, se service se heurte a la non-conformité des données des deux systèmes précisément sur l'unité de mesure. Le schéma suivant illustre le concept 'Un' des deux systèmes chacun dans son contexte.

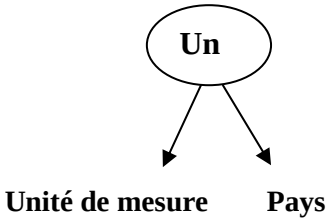
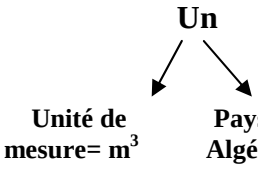
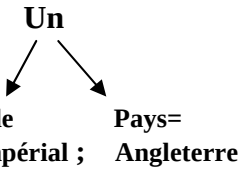
7. Exemple sur l'hétérogénéité sémantique



Dans cet exemple, nous remarquons que l'unité de mesure utilisée dans le service étranger d'un pays Anglophone est le gallon impérial, tandis que celle de notre service (francophone) est le m^3, ce qui génère un conflit de valeur. Cette hétérogénéité sémantique au niveau d'unité nécessite une annotation de la description de notre service en se basant sur les ontologies générées dans la section précédente (IV.5).

Dans ce qui suit nous présentons les parties des ontologies concernant le concept (Un) de notre service étudié dans l'exemple ci dessus.

Tableau IV.5 Exemple des ontologies liées au concept 'unité de mesure'

	Service source	Service cible
Ontologie de domaine		
Ontologie contextuelle du concept unité de mesure		
Ontologie locale de données		

A la base de cet extrait d'ontologies correspondant au concept 'Un', nous pouvons faire des relations d'équivalence entre les concepts , précisément pour l'unité de mesure (m^3 et le gallon impérial)

Par la suite, nous pouvons annoter notre service. Cette annotation se manifeste par l'ajout des attributs du standard SAWSDL à la description du service comme est illustré dans la [Figure IV.7].

```
…….
……
</xs:element>
<xs:complexType name="item">
<xs:all>
 <xs:element name="UPC" type="xs:string" />
 </xs:all>
<xs:attribute name="quantité" type="xs:integer" />
<xs :attribute name="Un" type="xs:string" />

 sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/
 purchaseorder#Un">
 sawsdl:loweringSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/
 Mapping /RDFOnt2Un.xml">

<xs:attribute name="prix" type="xs:float" />
</xs:complexType>
<xs:element name="OrderResponse" type="confirmation" />
<xs:simpleType name="confirmation"
<xs:restriction base="xs:string">
 ……..
 ….
 …
```

Figure IV.7 Annotation du service ‘commande’

A la fin de ce chapitre nous pouvons dire que notre but est atteint celui de l’ajout de la sémantique aux services web extraits du système patrimonial GSR.

8. Conclusion

A l’aide de cette approche, nous pouvons conserver la logique d’affaires contenue dans les systèmes patrimoniaux, en faisant migrer toutes leurs fonctionnalités importantes vers le paradigme des services Web. Nous ne contentons pas de l’aspect syntaxique de ces derniers mais nous transitons vers l’aspect sémantique. Ceux-ci sont prêts à la publication et à l’intégration avec d'autres services sans souci du problème d’hétérogénéité sémantique. On peut les réutiliser éventuellement dans l’implémentation de nouveaux systèmes orientés services.

A la fin de ce chapitre, nous pouvons dire que notre objectif de l’annotation ou d’une autre manière l’ajout de la sémantique aux services Web extraits d’un système patrimonial est atteint.

Conclusion Générale

Conclusion générale

Nous rappelons que notre travail de recherche s'inscrit dans le cadre de l'intégration des applications des entreprises (EAI), dont l'objectif principal est de faire communiquer et coopérer une panoplie d'applications d'entreprise hétérogènes, dont la difficulté majeure relève du fait que ces applications ont été généralement conçues de façon indépendante, voire incompatible. Récemment, les Services Web ont émergé et permettent d'offrir une myriade de standards pour l'intégration. Nous avons, en particulier, retenu le manque de prise en charge de l'aspect sémantique comme étant un des éléments critiques qui caractérisent les techniques syntaxiques actuelles. Pour remédier aux limites des techniques syntaxiques, et dans le but également de favoriser la flexibilité des systèmes d'information précisément patrimoniaux, nous avons proposée une approche d'après la quelle on a introduit l'aspect sémantique.

Il est nécessaire d'intégrer les systèmes patrimoniaux vue a leurs retard technologique, parce qu'ils sont développés par des outils et des techniques traditionnelles. Leur intégration dans un système EAI basé sur les services web, leur donne une autre chance d'être aux premiers rangs de la concurrence avec les nouveaux systèmes, ou même permet la réutilisation de leurs fonctionnalités intéressantes pour implémenter des systèmes avancés.

Dans cette étude nous avons présenté le paradigme des architectures orientées services, les services web pour dresser les techniques de migration des systèmes patrimoniaux. Nous avons mentionné les limites de ces paradigmes qui concerne l'aspect sémantique.

Dans ce mémoire, nous avons présenté une approche de migration de type boîte grise inspirée de l'approche de S. Alhamari et celle de J. Zhi et all qui recouvre toutes les étapes nécessaires à l'extraction des services Web à partir des systèmes patrimoniaux. Pour tirer avantage de ces deux approches, nous avons procédé à leur combinaison en intégrant une annotation semi automatique aux services web, méritants d'être publier, par le standard SAWSDL en se basant sur l' ontologie de domaine et l'ontologie contextuelle a fin d'éviter le problème d'hétérogénéité sémantique. Nous avons également proposé l'intégration des expériences des

utilisateurs des systèmes patrimoniaux dans les phases de l'analyse, de détection des services Web logiques et la création des ontologies .

Enfin, nous avons expliqué les différents concepts fournis, en appliquant cette approche sur une ancienne application, développée avec le paradigme procédural en utilisant le langage C standard. Cette application (GSR) est relative à la gestion de stock d'une raffinerie. Les résultats auxquels nous avons abouti sont très importants, ce qui nous permet de dire que nos objectifs sont atteints.

Perspective

Concernant les perspectives de notre travail, nous proposons d'ajouter une étape d'évaluation de granularité des services résultants. Aussi l'implémentation D'un outil pour achever la phase d'extraction automatique du code implémentant les fonctionnalités pertinentes en utilisant l'algorithme de clustering .

Références Bibliographiques

[1] : Seralia - Moteur d'intégration - Livre blanc sur l'EAI - Etude Integration Brokers: Market, Vendors and Trends 2001

[2] : livre blanc, Nouvelles technologies pour l'intégration : les ESB , EBM Websourcing, Janvier 2006 .

[3] : Schmidt J.,2000 « Enabling next-generation entreprises» et all journal, pages 74-80

[4] : Stonebraker M., 1999« Integrating islands of information » et all journal,

page 1-5

[5] : A.Semoud et ALaymy , Université Hassan II Mohamadia mémoire d'ingéniera « Gestion de l'informatique » .

[6] : Intégration d'applications d'entreprise - Définition - Encyclopédie scientifique en ligne.htm.

[7] : Wikipedia / Intégration_d'applications_d'entreprise.htm.

[8] : Axel Kamalak , les apport de la methode MDM dans la performance du SI des entreprises.

[9] : Lionel Pradelier, « Comment l'organisation de l'entreprise influence-t-elle la structure du système d'information ? Jalons théoriques et méthodologiques », 2008.

[10] : http://atilf.atilf.fr/academie9.htm.

[11] : Meinadier J.P. Ingénierie et intégration des systèmes, Hermès, (1998 , p28) .

[12] : M.N.MEADI, Réutilisation des fonctionnalités d'un système patrimonial dans une architecture orientée services Web, juin 2007.

[13] : Cédric Mora, SOA : Définition, Utilisation dans le monde de la banque et méthodologie de test.

[14] : Said Izza, Intégration des systèmes d'information industriels, une approche flexible basée sur les services sémantiques ,2006.

[15] : http://fr.wikipedia.org/wiki/Service_Web.

[16] : http://www.w 3.org

[17] : Myyad Jaber, architecture de système d'information distribué pour la gestion de la chaine logistique : une approche orienté service.

[18] : Damien Cortès, Sid Ali Guebli, web service et l'impact sur le e-bisness, décembre 2003.

[19] : Jihed Touzi , Aide à la conception de Système d'Information Collaboratif

support de l'interopérabilité des entreprises,09/11/2007.

[20] :Harry M. Sneed, Katalin Erdos, Extracting Business Rules from Source Code, Proc. Of 4th IWPC- 1996, IEEE Computer Society, Berlin, March 1996, p.240

[21]: H. Huang, W. T. Tsai, S. Bhattacharya, X. P. Chen, Y. Wang, J. Sun, Business Rule Extraction from Legacy Code, 1996 IEEE.

[22]: Frederick V. Ramsey, James J. Alpigini, A Simple Mathematically Based Framework for Rule Extraction Using Wide Spectrum Language, Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'02).

[23]: X. Wang, J. Sun, X. Yang, Z. He, S. Maddineni,Business Rules Extraction from Legacy Systems, Proceedings of the IEEE Conference on Software Maintenance and Reengineering (CSMR'04).

[24]: X. Liu, Z. Chen, H. Yang , H. Zedan, William C. Chu, A Design Framework for System Re-engineering, IEEE 1997.

[25] : Goh C. H., « Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources ». Phd Thesis, MIT, 1997.

[26] : Bussler C., « Semantic Web Services : Reflections on Web Service Mediation and Composition ». Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03), 2003.

[27] : Kavouras M ; »A unified ontological framwork for semantic integration ». International Workshop on Next Generation Geospatial Information, Cambridge, UK, 2003.
[28] : M.Uschold, and M.Gruninger ;creating semantically integrated communities on the world wide web.In semantique web workshop, Honolulu ,Hawaii-invited tallk , 2002

[29] : D. C. Rogozan ;Gestion de l'évolution d'une ontologie : méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions de l'ontologie .

[30] : Borst, W. N. Construction of Engineering Ontologies for Knowledge Sharing and Reuse. University of Tweenty, Enschede, Centre for Telematica and Information Technology, 1997
[31] : Gruber, T. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Paper presented at the International Workshop on Formal Ontology, Padova, Italy, 1995
[32] : Guarino, N., et Giaretta, P. (). Ontologies and Knowledge Bases : Towards a Terminological Clarification. In Mars N. J. I. (Ed.), Towards Very Large Knowledge Bases : Knowledge Building and Knowledge Sharing (pp. 25-32),1995.
[33] : S. Pokraev, D. Quartel, M. W.A. Steel, M. Reichert, Semantic Service Modeling : Enabling System Interoperability, Enterprise Interoperability : NewChallenges and Approaches- Springer Verlag –ISBN-10 : 1846287138, 2006

[34] :B. Swartout, R. patil , K. Knight and T. Russ ; Use a Large – Scale Ontologies. Spring symposium series on ontological Engineering. Stanford university, CA, pp138 -148 ,1997.

[35] : yassine Gargouri ; Contribution à la maintenance des ontologies à partir d'analyses textuelles : extraction de termes et de relations entre termes, Avril 2009 .

[36] : Michaèl Mrissa , Médiation Sémantique Orientée Contexte pour la Composition de Services Web.
[37] : J.Farrell, IBMHolger Lausen et D.Innsbruck ;Semantic Annotations for WSDL and XML Schema ,W3C Recommendation 28 August 2007.

[38] : SAWSDL : Semantic Annotations for WSDL SAWSDL Resource Page

[39] : http://www.w3.org/TR/xmlschema-1

[40] : SAWSDL: Semantic Annotations for WSDL, LSDIS and the University of Georgia, 2005.

[41] : N .Zouggar, B .Vallespir, D .Chen ; Enrichissement de la modélisation d'entreprise par les ontologies .LAPS/GRAI Université Bordeaux1, ENSEIRB, UMR CNRS 5131 6e Conférence Francophone de Modélisation et Simulation -MOSIM , 2006 -RABAT- Mroc « Modélisation , Optimisation et Simulation des systèmes : Défis et Opportunités ».

[42] : Joell Farrell, IBM ; Holger Lausen, Deri, Semantic annotation for WSDL.W3C Working draft 28 Septembre 2006 . Insbruk.http //www.w3.org/TR/2006/WD-sawsdl-20060928/

[43] : OMG. Ontology Definition Metamodel OMG Adopted Specification, November 2007.

[44] : Thompson.H.S., Beech .D, Maloney. M, Mendelsohn. N. XML Schema

Part 1, Structures Second Edition. W3C Recommendation [en ligne], 2004.

[45] : Zhuopeng Zhang, Jianzhi Li and Hongji Yang, A Grid Oriented Approach to Reusing Legacy Code in ICENI Framework, IEEE 2005.

[46] : Ward, M.P, K.H. Bennett, Formal Methods for Legacy Systems, Journal of Software Maintenance: Research and Practice, Vol 7, no 3, May-June 1995, pp 203-219

[47] : Céline Lopez-Velassco, Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation, 18 novembre 2008.

[48] : Pierre Beaudoin et associés, Étude exploratoire sur les systèmes patrimoniaux des entreprises et leurs exigences de compétences, édition TECHNO Compétences.

[49] : M. P. Ward, H. Zedan and T. Hardcastle, Legacy Assembler Reengineering and Migration, ICSM2004, The 20th IEEE International Conference on Software Maintenance.

[50] : K. H. Bennett and J. Xu, Software Services and Software Maintenance,

Proceedings of the IEEE Seventh European Conference On Software Maintenance

And Reengineering (CSMR’03),

[51] : Santiago Comella-Dorda, Kurt Wallnau, Robert C. Seacord, John Robert, A Survey of Black-Box Modernization Approaches for Information Systems, IEEE 2000

[52] : Robert C. Seacord, Daniel Plakosh, Grace A. Lewis Addison Wesley – Modernizing Legacy Systems; Software Technologies, Engineering Processes & Business Practices, edition : Addison Wesley 2003, ISBN: 0-321-11884-7.

[53] : Integrating Legacy Applications as Web Services, A HostBridge Technology. White Paper, 2003.

[54] : Zhuopeng Zhang, Hongji Yang, Incubating Services in Legacy Systems for

Architectural Migration, Proceedings of IEEE 11th Asia-Pacific Software Engineering Conference (APSEC’04).

[55] : Ying Zou and Kostas Kontogiannis, Towards a Web-centric Legacy System

Migration Framework, Proceedings of the 3rd International Workshop on Net-

Centric Computing (NCC): Migrating to the Web, International Conference on

Software Engineering (ICSE’01), Toronto, Canada, 2001

[56] : Harry M. Sneed, Integrating Legacy Software into a Service Oriented Architecture, Proceedings of IEEE Conference on Software Maintenance and

Reengineering (CSMR'06)

[57] : Barros A., Dumas, M., Oaks, P. Standards for Web Service Choreography and Orchestration: Status and Perspectives. In: Procs of the 3rd International Conference the Business Process Management (BPM 2005), 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management, Nancy, France, Sept. 2005, pp.1-15.

[58] : Benatallah B., Dijkman R., Dumas M., Maamar Z. Service Composition: Concepts, Techniques, Tools and Trends. In: Stojanovic Z., Dahanayake A., Eds.

Service- Oriented Software Engineering: Challenges and Practices. Idea Group Inc (IGI, pp.48-66), 2005.

[59] : Peltz, C. Web Services Orchestration and Choreography. IEEE Computer, vol.36, n°10, pp.46-52,2003.

[60] : Eric Darras, Plans de migration de systèmes Patrimoniaux vers des ERP, septembre 2004.

[61] : Cyril Faucher, Frédéric Bertrand, Jean-Yves Lafaye , Génération d'ontologie à partir d'un modèle métier UML annoté, Laboratoire L3i – Université de La Rochelle.

[62] : Hervé Verjus, Conception et Construction de Fédérations de Progiciels, l'universite de savoie.

[63] : Singh M., and Huhns M., "Service-Oriented Computing,
Semantics, processes, agents". Wiley, 2004.

[64] : Hasselbring W., "Information Systeme Integration".
Communications of the ACM, Vol. 43 (No. 6), pp. 33-38, 2000.

[65] : http://www.stream-consulting.fr

[66] : www.seralia.com

[67] : http://archimede.bibl.ulaval.ca/archimede/fichiers/21998/ch03.html

[68] : Seacord Robert, Santiago Comella-Dorda, Kurt Wallnau and John Robert, A

Survey of Legacy System Modernization Approaches, 2000 ;

[69] : Sylvain Rampacek, Sémantique, interactions et langages de description des services web complexes ,10 novembre 2006

[70] : http://fr.wikipedia.org/wiki/SOAP

[71] : Lionel Tricon, Initiation au couple gagnant WSDL/SOAP.

[72] : Meredith A. Barnes and Charmain Cilliers ,Comparing Legacy System Modernization Approaches for a Service Oriented Architecture .

[73] : Zhuo Zhang, Dongdai Zhou , Hongji Yang ,Shaochun Zhong , A Service Composition Approach Based on Sequence Mining for Migrating E-learning Legacy System to SOA

[74] : Jian.Z, Zhuo.P.Z, Bing.Q, Hong.J.Y, A component mining approch to incubate grid services in object oriented legacy systems, 2006.

[75] : Feng Chen, Zhuopeng Zhang, Jianzhi Li, Jian Kang and Hongji Yang, Service Identification via Ontology Mapping 2009.
[76] : Zhuopeng Zhang, An ontology-based reengineering methodology for service orientation, 2009.
[77] : Valéry Psyché, Olavo Mendes, Jacqueline Bourdeau, Contribution of Ontological Engineering to Distance Learning Environments.
[78] : Saad Alahmari, Ed Zaluska, Optimal Granularity for Service-Oriented Systems.
[79] : Jianzhi Li, Zhuopeng Zhang and Hongji Yang,A Grid Oriented Approach to Reusing Legacy Code in ICENI Framework .

[80] : Zhuopeng Zhang, Ruimin Liu and Hongji Yang, Service Identification and Packaging in Service Oriented Reengineering, 2005.

[81] : Etude d'un algorithme de clustering, Epreuve commune de tipe 2009 - partie d.
[82] : Germain Forestier, Connaissances et clustering collaboratif d'objets complexes Multisources , 29 Septembre 2010.
[83] : Conception et exploitation d'une base de métadonnées de traitements informatiques, représentation opérationnelle des connaissances d'expert
Application au domaine géographique, Yann Abd-el-Kader.
[84] : Vers une architecture d'intégration sémantique des composants métier
H. Elasri, L. Kzaz et A. Sekkaki
[85] : Fatiha Saais, Intégration sémantique de données guidées par une Ontologie, 2007.
[86] : Ontologies et l'éditeur Protégé, Application à la formalisation des concepts de description d'IMGT-ONTOLOGY
[87] : http://www.revuei3.org/hors_serie/annee2004/revue_i3_hs2004_01_01.pdf.
[88] : http://www.emse.fr/~beaune/websem/WS_Protege-2000.pdf.
[89] : Xavier Lacot, Introduction à OWL, un langage XML d'ontologies Web.
[90] : Philippe Laublet, ChantalReynaud, Jean Charlet, Sur quelques aspects du Web sémantique
[91] : Jacek Kopecký, SAWSDL Status and relation to WSMO
[92] : Berners-Lee,Semantic Web - XML 2000 Conference, 2000, from http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html

[93] : Oberle et al, An extensible ontology software environment. In S.Staab et R. Studer (Eds.), Handbook on Ontologies (pp. 299-320),2004.
[94] : Gestion de l'évolution d'une ontologie : méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions de l'ontologie
[95] : Yassin Chabeb, Samir Tata et Djamel Belaid, OTES : Une ontologie pour l'annotation de servicesWeb sémantiques.

[96] : Phan Quang Trung Tien, Ontologies et Web Services, 2005.

[97] : F. Amardeilh, Web Sémantique et Informatique Linguistique: propositions méthodologiques et réalisation d'une plateforme logicielle, Thèse de doctorat, Discipline : Informatique, Université Paris X – Nanterre, Mai 2007.
[98] : O. Corcho, Ontology based document annotation: trends and open research problems, in International Journal of Metadata, Semantics and Ontologies, 1(1), Inderscience, 2006, pp. 47-57.