République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider
BISKRA

Faculté des Sciences et des sciences de l'ingénieur
Département de l'informatique

Mémoire

En vue d'obtention du diplôme de Magister en informatique
Option : Systèmes d'Information Avancés et Intelligence Artificielle

Conception des services Web dans le processus d'intégration des applications d'entreprises (EAI)

Présenté par :
Meadi Mohamed Nadjib

Devant le jury :

Mr. Djeddi Nourredine	Président	Professeur, Université de Biskra.
Mr. Boufaida Mahmoud	Rapporteur	Professeur, Université de Constantine.
Mr. Zarour Nacereddine	Examinateur	Maître de conférences, Université de Constantine.
Mr. Kazar Okba	Examinateur	Maître de conférences, Université de Biskra.

Soutenue le 23/06/2006

سبحانك لا علم لنا
إلا ما علمتنا إنك
أنت العليم الحكيم

Remerciements

« El hamdou l'elleh » qui m'a donné la force et le pouvoir d'accomplir ce modeste travail.

À cette occasion je remercie :

Mon encadreur **Mr. Boufaida Mahmoud** pour ses orientations et ses conseils bénéfiques. Les membres de jury : **Mr. Djeddi Nourreddine, Mr. Zarour Nacereddine et Mr. Kazar Okba**. Mes remerciements vont également à **Mr. Hussin Drid**, qui m'a aidé au fond sans hésitation. De même, je remercie nos enseignants et l'équipe du département de l'informatique du l'université de BISKRA, ainsi que, mes amis et mes collègues qui m'ont aidé de près ou de loin.

Et à tout ceux que je n'avais pas cité, je dit encore :

Merci de mon cœur.

Dédicaces

A mes parents,

A mes grands-mères,

A mon frère et mes sœurs,

A mon cher amis Karim Ouafi,

A mes amis Amar NADJI, Bakr DRID,

A tous les personnes qui m'aiment.

Table des Matières

Liste des Figures

Liste des Tableaux

Liste des listings

Introduction générale

Contexte du travail

L’informatique dans l’entreprise a énormément évolué surtout avec la révolution technologique galopante actuelle, et avec le temps ces entreprises se fondent sur une myriade d'applications et de systèmes d'information, tels que les systèmes CRM (Customer Relationship Management), les systèmes de gestion de comptabilité et des finances, les systèmes ERP (Enterprise Resource Planning), les serveurs Web, les systèmes patrimoniaux, et bien d’autres [1].

L’apparition du Web en tant qu’espace libre pour la publication et le développement permet aux entreprises d’être basées sur le Web pour gérer et étendre leurs affaires. Mais pour aboutir à ces objectifs, il faut que ces entreprises unifient l’interface d’interaction de leurs systèmes informatique avec l’extérieur pour permettre l’accès aux informations qu‘ils contiennent. Ces informations sont nécessaires pour garantir le déroulement normal de leurs activités. Pour cela, elles doivent être accessibles à partir de plusieurs points d'accès, 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 le processus d'intégrer les différents systèmes d'informations, anciens (legacy) et avancés, d'une même entreprise à travers un réseau reliant les machines d'une entreprise ou des entreprises, où l'intégration exige un minimum de (ou aucun) changement aux applications existantes. Une autre définition propose que l'EAI permette de faire communiquer et collaborer les applications d'une même entreprise. Donc, l'EAI doit compléter les 4 fonctions suivantes [2]:

1-interfacer: extraire et injecter des données dans l'application.
2-Transformer: convertir les formats des données.
3-Router: transporter les données vers les destinataires.
4-Gérer: suivre l'état du processus métier (Business Process).

Il existe plusieurs solutions d’intégration des applications d’une entreprise (EAI), c’est à dire les méthodes d'intégration classique. Nous citons l’intégration des données, l’intégration au niveau des interfaces utilisateurs, l’intégration au niveau des applications, et l’intégration au niveau des méthodes. Mais chacune de ces méthodes possède ses propres inconvénients qui influent par la suite sur le rendement du système informatique intégré de l'entreprise.

Malheureusement, l'EAI traditionnel a présenté pas mal de problèmes, tels que les coûts du logiciel d'intégration et sa complexité, ceci exige un effort énorme et le grand problème qui réside 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 [1] qui permettent à l'entreprise de réduire les coûts investis pour les solutions EAI traditionnels, puisque ces services sont basés sur des normes industrielles et n'exigent pas des connaissances spécialisées sur les applications et les produits.

Nous pouvons définir les services Web comme des composants logiciels encapsulant des fonctionnalités métiers de l'entreprise et sont accessibles via des standards du Web (XML, SOAP, WSDL, UDDI.,...etc.) [2]. Les services Web servent alors d'interface d'accès à l'application, et dialogue avec elle au moyen de middleware (CORBA, RMI, DCOM,...). L'implémentation des solutions EAI basées sur les services Web permet à l'entreprise de réduire les coûts investis pour les solutions traditionnelles, puisque les solutions services Web sont basées sur des normes industrielles et n'exigent pas des connaissances spécialisées sur les applications et les produits.

Comme les entreprises sont basées sur des applications patrimoniales (Legacy systems), il devient primordial d'intégrer leurs systèmes d'information dans une seule enveloppe technologique. Ces systèmes sont importants d'un point de vue économique pour leurs propriétaires. En effet, le coût à investir pour la maintenance de ces systèmes est très élevé. Dans la plupart des cas, il y a abandon des systèmes patrimoniaux et reconstruction de nouvelles applications basées sur des technologies plus récentes. Grâce à la technologie services Web, cette solution est passée à une autre meilleure : 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 des quelques fonctionnalités intéressantes d'un système patrimonial et leur réutilisation comme des services Web se font d'une manière aisée.

Objectif du travail

Dans notre mémoire nous nous intéressons à l'approche d'extraction des services Web à partir des systèmes patrimoniaux de type « boîte grise ». Plusieurs travaux de ce type ont été proposés, qui sont assez semblables dans leurs méthodologies avec quelques différences dans les détails de chaque approche. Selon notre point de vue, ces travaux présentent quelques inconvénients majeurs. Le premier est lié à l'intervention décisive de la supervision humaine dans les phases de ces approches. Cette intervention est souvent coûteuse et sensible aux erreurs, surtout lors de la détection du code source qui implémente le service Web. Le deuxième est le manque de standardisation dans ces approches, car chaque langage de programmation nécessite d'effectuer des modifications approfondies pour être adaptables au langage désiré, nous citons par exemple le module de détection du code source qui implémente le service Web, le module de raffinement du code extrait et le module d'adaptation qui permet de qualifier le code source extrait à être un service Web concret...etc. Le dernier inconvénient consiste est lié l'omission de l'intégration des avis des utilisateurs des systèmes patrimoniaux dans le processus de migration.

Pour remédier à ces inconvénients, nous proposons une approche générique en adoptant l'approche « boîte grise ». Notre approche offre un processus basé sur le langage WSL (Wide Spectrum Language) comme langage intermédiaire pour l'extraction des fonctionnalités existantes dans les systèmes patrimoniaux des différents langages de programmation, et leur réutilisation en tant que services Web où l'intervention de la

supervision humaine est très limitée et permet aussi d'intégrer les efforts des utilisateurs dans les phases de processus d'émigration et surtout dans la phase de l'analyse et la définition des services Web logiques.

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 trois chapitres. Dans le premier chapitre, une étude approfondie sur le domaine d'intégration des applications des entreprises (EAI) est présentée, tout en exposant quelques définitions proposées ensuite nous présenterons les différentes approches d'intégration, point à point et middleware, et les différent niveaux d'intégration. Nous terminerons ce chapitre par une architecture type d'un EAI. Dans le deuxième chapitre, nous nous intéresserons par les services Web et les différents standards utilisés pour les déployés, et nous terminerons ce chapitre avec les avantages apportés par les services Web au monde industriel. Enfin, le dernier chapitre de la première partie entamera 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.

Le quatrième chapitre de ce mémoire, et avant d'introduire notre approche pour l'extraction des services Web à partir des systèmes patrimoniaux, nous présenterons en général les deux nouveaux concepts utilisés dans notre approche qui sont la règle métier et le langage WSL.

Le chapitre cinq, sera consacré à la validation de notre approche, où nous appliquerons cette approche sur une application patrimoniale qui a été développée en langage C pour la résolution des systèmes linéaires, dont nous avons introduit une fonctionnalité pour la réutiliser comme un service Web.

Chapitre 1 :

EAI:
Un nouveau paradigme

1 Introduction

Pour une raison ou une autre, les systèmes d'information et les applications dans une même entreprise sont très variés. Parmi ces raisons nous trouvons que les entreprises achètent ou développent des systèmes d'information dédiés chaque fois que les besoins de leurs activités augmentent ou l'exigent. Elles déploient des applications sur plusieurs plateformes et différentes architectures, et elles personnalisent ses systèmes pour être conformes avec leurs activités.

Avec l'évolution rapide de l'informatique dans l'entreprise, les conditions de l'échange de l'information et de la collaboration d'affaires entre différentes entreprises augmentent également, et les voies pour résoudre le problème des îlots des informations montent plus haut que jamais. L'historique des systèmes d'information dans les entreprises compte plusieurs tentatives pour l'intégration des applications et les sources des données, 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 des affaires d'une entreprise, et pour construire une base d'informations complète et exhaustive. Ces travaux ont donné naissance à un nouveau domaine de recherche, appelé EAI (Enterprise Application Integration), dont l'objectif est de définir tous les outils et les méthodes nécessaires pour accomplir l'intégration des applications dans une entreprise avec un temps très réduit.

Dernièrement, l'eBusiness s'envisage comme un nouveau modèle permettant de contrôler les chaînes d'achat et d'approvisionnement, les relations avec les clients et fournir des applications et des services basés sur le Web. Pour cela, les règles et les buts qui gèrent l'EAI ont été changées et passées à une autre meilleure qui est le renforcement des systèmes EAI classiques par des outils avancées pour qu'ils soient capables de supporter l'exposition des systèmes traditionnels des entreprises à travers le monde entier à l'aide du Web, dont le but est de chercher des nouveaux marchés, en adoptant le model eBusiness.

L'objectif de ce chapitre est d'explorer le domaine de l'intégration des applications dans l'entreprise (EAI). Nous commençons cette étude par l'introduction des différents systèmes d'information qui peuvent exister dans une entreprise. Après la présentation de quelques définitions de l'EAI tirées de littératures, nous allons révéler les différents problèmes qui arrivent devant le processus d'intégration. Ensuite, nous entamons les modèles d'intégrations logiques et en manifestant, à la fin de cette section, les avantages et les inconvénients de chacun, et nous les suivrons par les niveaux d'intégrations, qui sont l'intégration des données, le niveau d'API (Application Programming Interface), le processus

métiers et l'intégration de l'interface utilisateurs. La section suivante sera consacrée aux approches proposées pour accomplir la tâche de l'intégration qui sont, l'approche client serveur, utilisation des adaptateurs synchrones et asynchrones, utilisation d'un serveur d'application, et enfin la dernière proposition pour l'EAI qui est Architecture de Connecteur de J2EE (JCA). Nous exposons aussi une architecture de trois couches pour l'EAI. Cet étude est conclu par les avantages apportés aux entreprises grâce à l'EAI et une conclusion.

2 Systèmes d'Information des Entreprises (SIE)

Dans une même entreprise, il existe souvent des applications diverses qualifiées comme des Systèmes d'Information des Entreprises (SIEs), nous citons [10]:

- Les applications spéciales (custom applications) : sont des applications spéciales développées par les entreprises pour gérer ses affaires, en utilisant différents langages de programmation, comme par exemple : C/C++, COBOL,…etc.
- Les applications ERP (Enterprise Resource Planning): ces applications contiennent plusieurs fonctions utiles pour le bon déroulement des activités de l'entreprise, comme : la gestion des stocks, le contrôle de la production, la gestion des ressources humaines. Parmi les applications ERP nous trouvons les applications logistiques.
- Les programmes des transactions qui sont exécutés sur les systèmes de traitement des transactions.
- Les bases de données dites patrimoniales (Legacy Data Bases) : Le rôle de ce type de bases de données est la gestion des données critiques pour les activités des entreprises.

3 Notion d'Intégration des Applications d'Entreprises (EAI)

Dans la littérature, il existe plusieurs définitions de l'EAI et nous fournissons celles qui sont pertinentes et ont un effet sur notre travail :

« L'intégration des applications des entreprises (EAI : Enterprise Application Integration) est le processus de coordonner les opérations des différentes applications à travers l'entreprise. Ce terme est aussi appliqué à l'ensemble des applications commerciales, qui sont conçues pour faciliter le processus d'intégration » [1].

Une autre définition propose que l'EAI peut être défini selon deux perspectives [4] :

« De la perspective de Business, l'EAI est l'avantage concurrentiel qu'une entreprise obtient quand toutes les applications sont intégrées dans un système d'information unifié, capable de partager l'information et de supporter le déroulement des opérations de business.

L'information doit souvent être recueillie de plusieurs domaines et être intégrée dans un processus métiers ».

« De la perspective technique, l'EAI se rapporte au processus d'intégration des applications et des données différentes, pour permettre le partage des données et l'intégration des processus métiers entre les applications existantes sans devoir modifier ces derniers. L'EAI doit être exécuté en utilisant les méthodes et les activités qui lui permettent d'être pertinent en termes de coûts et de temps ».

4 Difficultés d'intégration des applications

L'intégration des systèmes d'information et des applications dans les entreprises est très difficile à cause de plusieurs problèmes de plusieurs niveaux [10]:

a. Niveau du support technologique

Les systèmes d'information des entreprises sont différents dans le niveau d'avancement technologique. Nous prenons par exemple la différence dans le support fourni pour la gestion des transactions et la sécurité. Plusieurs systèmes d'information sont des primitives et n'offrent pas des supports pour l'accès transactionnel, et il existe d'autres plus avancés qui permettent un accès transactionnel aux ressources.

b. Les restrictions technologiques et administratives

Plusieurs SIEs imposent des restrictions technologiques et administratives spécifiques aux utilisateurs. Par exemple l'utilisation des systèmes patrimoniaux (Legacy Systems) 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 de système doivent l'adapter avec les nouveaux systèmes et il faut les intégrer avec les différentes applications de l'entreprise.

c. L'aptitude de l'intégration avec les autres systèmes

Les SIEs diffèrent dans leurs modèles de programmation et API. Ce qui rend l'intégration de ces SIEs très difficile, parce que chacun de ces systèmes a été développé dans son propre environnement, et qui est peut être non conforme avec les autres systèmes, et en plus ils n'ont pas les mêmes objectifs.

d. 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, si nous avons par exemple un système d'information et que son client API est défini dans une bibliothèque C, et que cette bibliothèque a défini des applications clients utilisées pour gérer les transactions et effectuer l'accès transactionnel au système d'information, Alors il faut que le développeur étudie cette bibliothèque aussi pour pouvoir utiliser le client API de ce système.

5 Modèles d'intégration logique

Dans cette section, nous allons présenter les deux modèles existants pour intégrer logiquement les applications au sein d'une entreprise, l'intégration point à point (peer to peer) et l'intégration en utilisant un middleware:

5.1 Intégration point à point

La notion de point à point exprime l'idée d'intra-applications [5]. Dans ce modèle d'intégration, les applications intégrées sont connectées directement les unes aux autres [4], en engendrant un graphe fortement connecté (Full Connected) [Figure 1.1].

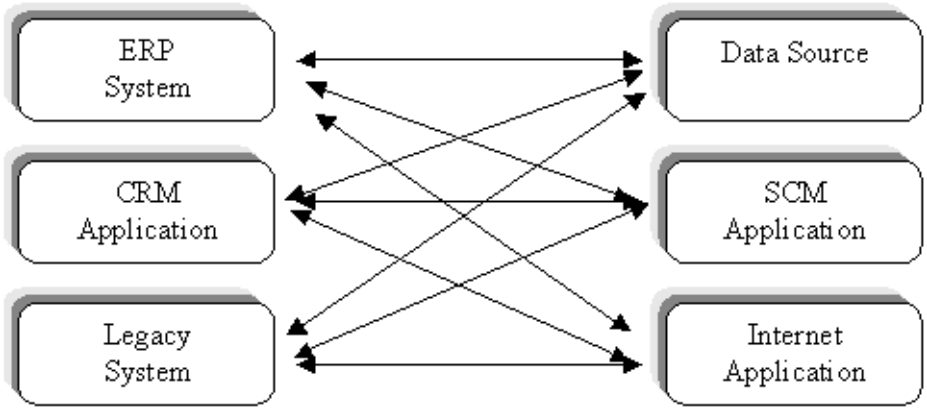


Figure 1.1. Intégration point à point [3].

5.2 Intégration basée sur le Middleware

L'objectif de cette approche est d'introduire un point intermédiaire entre les applications à intégrer [4]. Les Middlewares sont des logiciels intermédiaires et ont pour fonction la gestion des transactions, le partage des connexions, la mise à jour des serveurs et des bases de données. Pour que la communication devienne inter-applicative, la mise en place

d'un middleware rend le système d'information de plus en plus complexe, car chaque application reliée demande une intégration de middleware. En revanche, bien que très lourd en infrastructure, il améliore la communication des applications principales des entreprises, à savoir les E.R.P. (Entreprise Ressource Planning), la G.E.D. (Gestion Electronique de Données), la logistique ou SCM (Supply Chain Management), l'eBusiness, le CRM (Costumer Relationship Management) et le workflow (gestion des flux). Les middlewares permettent la connectivité entre toutes ses applications [Figure 1.2] [5].

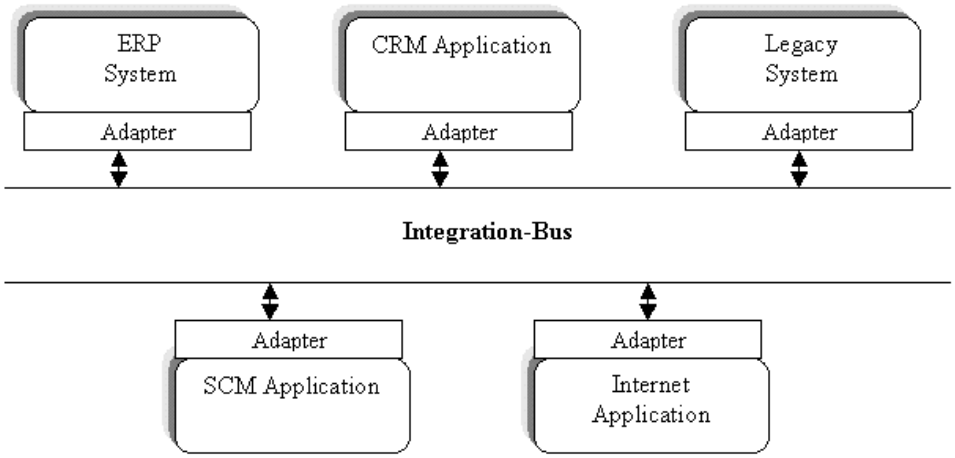


Figure 1.2. Intégration basée sur le Middleware [3]

5.3 Avantages et inconvénients

Les avantages et les inconvénients de ces deux modèles sont résumés dans le tableau suivant [6]:

Modèle	Avantages	Inconvénients
Point à point	• Simple à Comprendre. • Rapide à implanter surtout dans le cas où le nombre des applications à intégrer est petit.	• Manque du système d'intégration. • Sensible à la modification (modification d'une application peut causer une rupture de tout le système intégré). • Nombre des points d'intégration très élevé • Ajout d'une nouvelle application très difficile.

Middleware	• ajout ou changement d'une application n'a pas d'influence sur les autres. • Nombre de point d'intégration est égal au nombre des applications à intégrer. • Supporte un nombre élevé d'applications. • Ne nécessite pas beaucoup de maintenance. • Réalise des opérations très compliquées.	• Mise en œuvre difficile.

Tableau 1.1. Avantages et inconvénients des deux modèles de l'EAI

5.4 Comparaison

A partir des sections précédentes, une comparaison a été effectuée entre le modèle d'intégration point à point et l'autre de middleware.

Les critères	**Point à point**	**Middleware**
Simplicité	Simple	difficile
Mise en œuvre	Simple	difficile
l'implantation	Rapide	Lent
Sensibilité à modification	Sensible	Non
Nombre de points d'intégration	Elevé	Egale au nombre d'applications
Extensibilité	Très difficile	Extensible

Tableau 1.2. Comparaison entre le modèle point à point et Middleware.

6 Différentes approches de l'EAI

Il existe plusieurs approches permettant la construction des systèmes EAI, parmi celles-ci nous citons les approches suivantes :

6.1 Utilisation de l'approche Client/Serveur

Cette approche est de type deux tiers (Client, serveur). Elle était plus utilisée avant l'arrivée du Web. Dans cette approche l'accès aux données et les fonctions d'un système

d'information (SI) est effectué via une interface API fournie par un adaptateur [Figure 1.3]. La communication entre l'adaptateur et le SI est gérée par un protocole spécifique à ce SI [10].

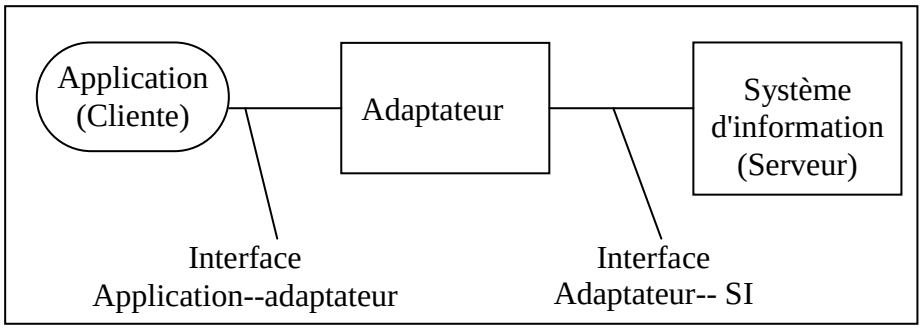


Figure 1.3. Approche Client/Serveur [10]

6.2 Utilisation des adaptateurs synchrones

Un adaptateur synchrone permet la transmission synchrone bidirectionnelle entre une application et un SI. Ce type d'adaptateur permet à un SI d'invoquer en mode synchrone une application. L'adaptateur synchrone basé sur le modèle REQUÊTE-RÉPONSE dans la communication entre le SI et l'application.

Alors l'adaptateur synchrone expose à l'application des APIs de type RPC (Remote Procedure Call). Ces RPCs créent des comptes récepteurs dans le SI. Quand une application veut agir avec le SI pour créer des comptes récepteurs, elle fait des appels à ces RPCs sur le SI. L'application qui lance l'appel attend jusqu'à ce que la fonction termine et renvoie la réponse à l'appelant. La réponse contient les résultats de l'exécution de la fonction sur le SI. Ce type d'interaction est considéré synchrone parce que l'exécution de l'application appelante attend synchroniquement pendant le temps d'exécution de la fonction sur le SI [10].

6.3 Utilisation des adaptateurs asynchrones

Dans la communication asynchrone, une application cliente envoie un appel au SI pour créer un nouveau compte récepteur, ensuite elle continue ses traitements. Dans l'autre côté, et après la réception du RPC, le SI le traite et renvoie les résultats de l'exécution au client dans un message de réponse.

Enfin, il est essentiel de rappeler que l'application cliente n'est pas suspendue quand son RPC est exécuté dans le SI, mais elle continue ses propres activités, et elle reçoit en temps différé les résultats de l'exécution de son appel.

Pour établir ce type de communication, il faut définir les mécanismes de messagerie qui garantit l'échange fiable des messages. Voici les trois mécanismes qui sont utilisés pour accomplir ce genre de communication [10] :

6.3.1 Utilisation d'une file des messages

Ce mécanisme est appelé messagerie de type Point à Point. Dans cette approche, nous trouvons que la file des messages est indépendante de l'émetteur et le récepteur et elle agit comme un buffer entre les applications communicantes. L'application émettrice envoie un message à cette file, ensuite l'application réceptrice reçoit le message à partir de la même file [10], comme il est illustré dans [Figure 1.4].

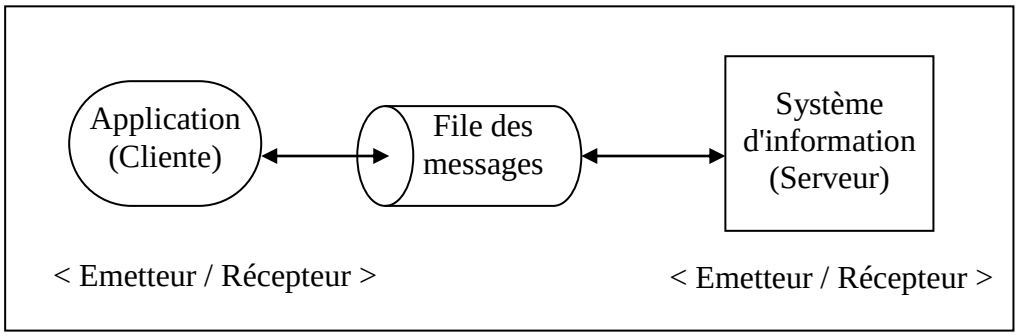


Figure 1.4. Utilisation d'une file de messages [10]

6.3.2 Le mécanisme Publier- Abonner (Publish - Subscribe)

Cette approche fonctionne d'une manière différente de la précédente. Il existe deux types différents d'applications : une application qui publie un message et des autres qui s'inscrivent pour recevoir le message publié. Cette approche assure la livraison normale des messages aux inscrits [10].

6.3.3 Utilisation de « message broker »

Dans ce cas, les systèmes d'information et les applications peuvent être en même temps des producteurs et des consommateurs, le « message broker » (courtier des messages) a le rôle de garantir l'intégration de ses systèmes et la livraison fiable des messages échangés. Le « message broker » fourni des services supplémentaires à l'EAI, nous citons [10] :

- Le routage des messages.
- La gestion des transactions.
- La livraison fiable des messages.
- L'assurance de la priorité et l'enchaînement des messages.
- La transformation des messages...etc.

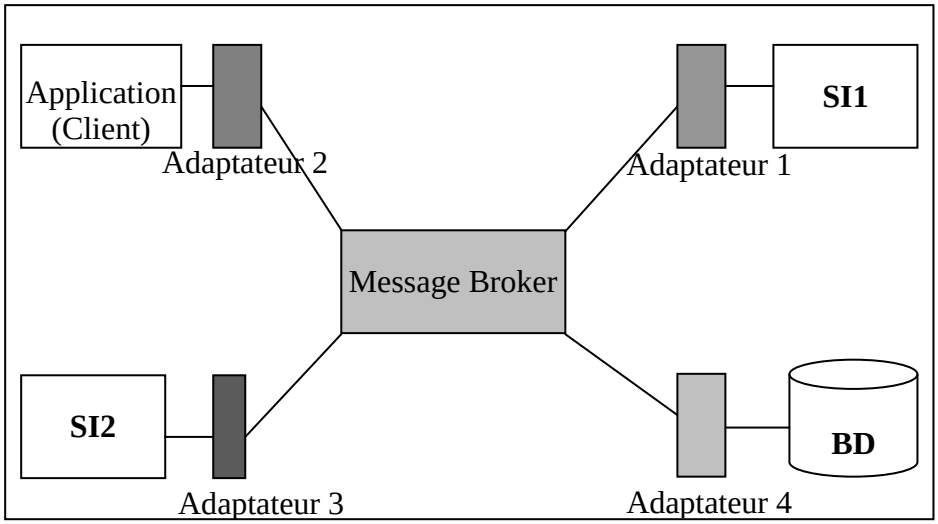


Figure 1.5. Utilisation du « Message Broker » [10].

Les étapes de l'échange basées sur l'approche du message broker sont [10] :

1. Réception du message provenant de l'adaptateur d'un système d'informations donné.
2. Interrogation du référentiel pour connaître la règle de transformation à utiliser.
3. Application de la règle de transformation.
4. Transfert des messages traités vers les applications destinataires.
5. Formatage des données par les adaptateurs des applications réceptrices.

6.4 Utilisation d'un serveur d'applications

Le serveur d'application est un pas très intéressant dans le domaine d'intégration des applications de l'entreprise. Cette approche fournit une plateforme pour le développement, le déploiement et la gestion des applications basées Web. Le serveur d'applications est très conforme à l'architecture multi-tiers, et exactement, il présente le tiers intermédiaire. Le serveur d'applications supporte les applications du modèle de développement basé composant, puisqu'il leur offre un support d'exécution et de déploiement [10].

Typiquement, un serveur d'applications fournit un ensemble de services d'exécution aux composants déployés. Ces services d'exécution sont cachés dans des composants d'application par un modèle de programmation simplifié. Les services fournis incluent [10] :

- Le support de transaction.
- Le mécanisme de sécurité.
- L'accès aux bases des données.
- La messagerie asynchrone.
- La communication distribuée.

- Les protocoles de Web.
- Support pour XML...etc.

Un exemple d'une architecture d'un serveur d'applications de plateforme J2EE pour l'EAI est présenté dans [Figure 1.6].

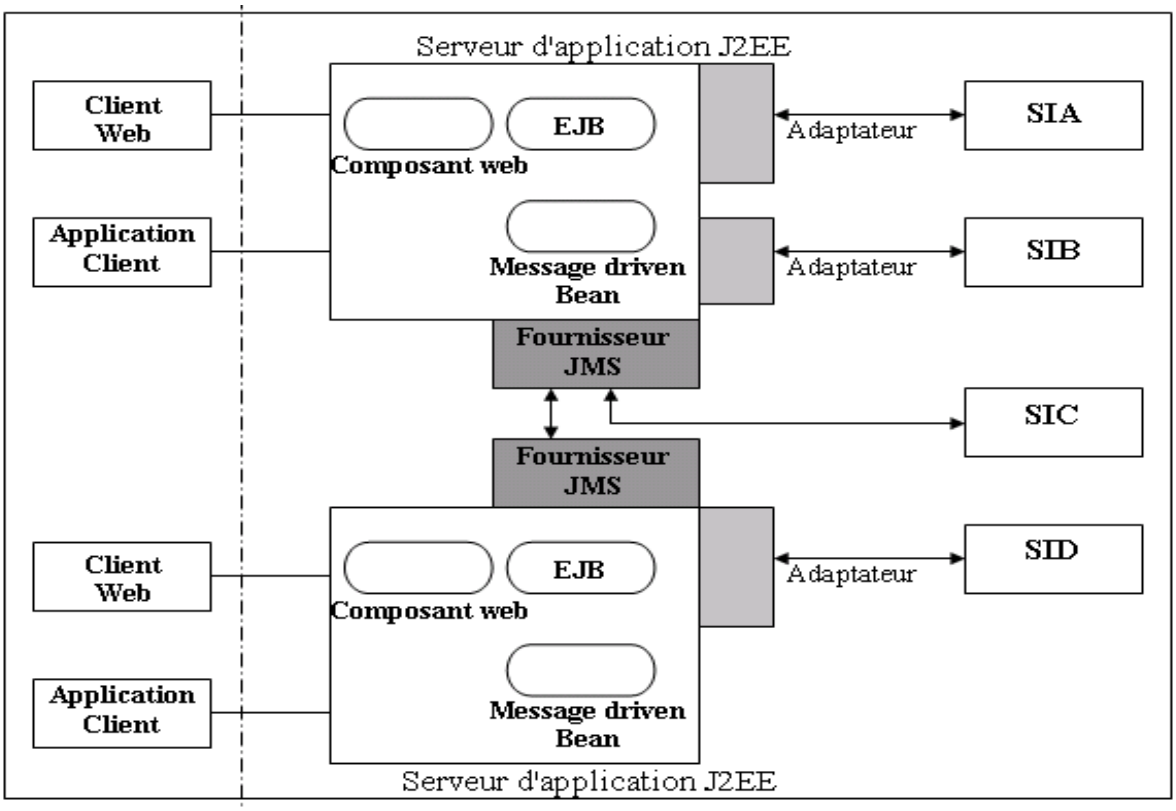


Figure 1.6. Un exemple de serveur d'applications [10].

7 Architecture de Connecteur de J2EE et EAI

Le groupe Java a défini une architecture de connecteur d'application [Figure 1.9], radicalement différente à l'intégration traditionnelle en plaçant les composants d'intégration dans le serveur d'applications de J2EE, en profitant des avantages de la maintenance et de la gestion de l'architecture centralisée. L'architecture de connecteurs proposée nommée JCA (J2EE Connector Architecture) définit un standard pour connecter la plateforme J2EE à des SIEs. L'architecture comporte des mécanismes d'invocations sécurisés et transactionnels [8].

Cette architecture nécessite une contribution du vendeur de serveur d'applications J2EE et du vendeur des SIEs. Le premier doit étendre son système pour supporter l'architecture du connecteur, et le second doit fournir un adaptateur de ressources standard pour son SIE. L'adaptateur permet le branchement au serveur d'applications et indirectement à l'application d'intégration. Un adaptateur de ressources pour chaque type de SIE est nécessaire pour la connexion au serveur d'applications.

Pour faciliter la connexion des serveurs d'applications et des SIEs, l'architecture spécifie un ensemble de contrats, c'est à dire des interfaces systèmes qui doivent être implémentées dans le côté serveur ou côté SIE. L'adaptateur implémente les contrats de côté SIE. Donc, Il s'agit d'un pilote de SIE pour les mécanismes de transactions, sécurité et pool de connexion.

Le contrat de gestion de connexions permet au serveur d'applications de gérer un pool de connexions avec des SIEs. Ceci permet d'ouvrir en avance multiples connexions et de les assigner aux applications dans le cas de la nécessité. Le contrat de gestion des transactions permet au serveur d'applications de gérer des transactions englobant plusieurs SIEs. Le contrat de sécurité permet un accès sécurisé à un SIE. Il propose un environnement sécurisé qui permet de limiter les menaces aux SIEs et de protéger leurs informations.

L'architecture de connecteur spécifie également une interface commune de client appelée CCI (Commun Client Interface) pour accéder à un SIE. Cette interface permet aux composants applicatifs de conduire des interactions avec des SIE hétérogène via une API unique et standard. Cette interface avec des contrats associés permet de réaliser des connecteurs transactionnels pour les outils d'intégration, tels les portails d'applications ou les EAI traditionnel.

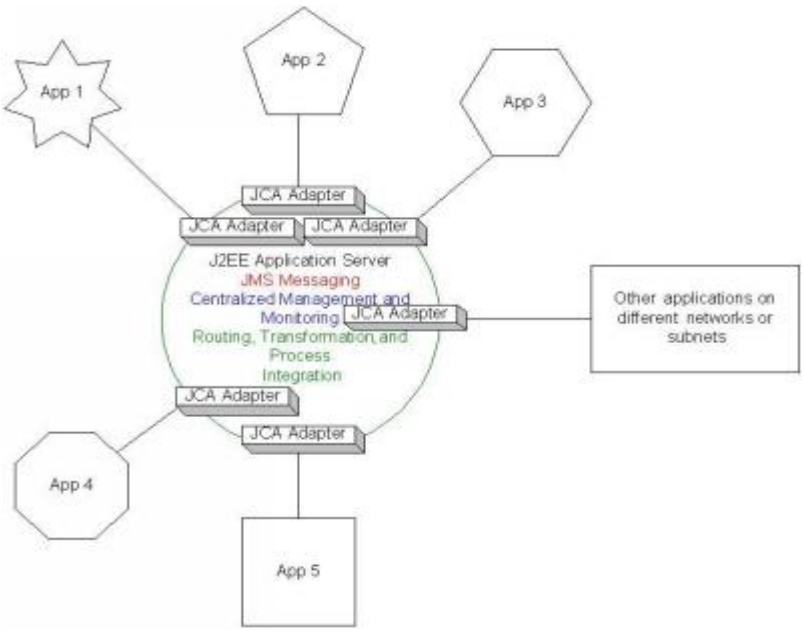


Figure 1.7. Architecture de Connecteur de J2EE (JCA) [4]

8 Différentes niveaux d'intégration

L'intégration des applications ou des systèmes d'information est effectuée selon 04 niveaux différents : l'intégration au niveau des données, l'intégration au niveau des interfaces

des applications (les APIs), l'intégration au niveau des processus et enfin l'intégration au niveau des interfaces utilisateurs. Le choix entre ces niveaux est conditionné par l'existant, c'est à dire les caractéristiques et les types des applications à intégrer. Selon ce qu'ils possèdent, les développeurs décident la méthode la plus adéquate à l'intégration [Figure 1.8].

8.1 Intégration des données

C'est le processus, les techniques et les technologies, de transfert de données entre les bases de données. Ceci peut être décrit en tant qu'extraction de l'information à partir d'une base de données, avec un traitement éventuel, la transformation et l'injection dans une autre base de données. Ce type d'intégration est utilisé dans le cas où les applications (SIs) à intégrer n'ont pas une interface des clients et des APIs [7].

Ce type d'intégration fournit deux primitives PUSH, PULL. En utilisant la primitive PUSH, une application peut ajouter des données dans une autre base de données, d'une autre application, par une requête SQL simple. Mais la primitive PULL utilise les déclencheurs (triggers) qui détectent un changement des données et stocker les nouvelles données dans les tables de l'interfaces, et ensuite les adaptateurs (wrappers) récupèrent les données pertinentes (après le changement) à partir des tables des interfaces des applications [4].

La technologie qui permet le transfert et la transformation des données n'est pas chère par rapport aux autres niveaux de l'EAI [7]. On peut conclure que ce type d'intégration relie les données au niveau de la pertinence des données.

8.2 Intégration des interfaces des applications (API)

Les interfaces de programmation d'application (API : Application Programming Interface) sont des appels qui peuvent être utilisés pour se connecter à une application, pour invoquer des services, et pour rechercher des données. L'intégration au niveau d'API permet le partage de la logique d'affaires, des entreprise, commune exposée en utilisant les interfaces de programmation prédéfinies. La majorité des vendeurs des applications publie des APIs pour faciliter l'intégration avec leurs produits.

Pour permettre l'intégration au niveau d'API, les deux côtés de l'échange de l'information doivent parler le même langage ou protocole. Beaucoup d'efforts ont été dépensés dans le développement des frameworks standards (ou protocole / langage) pour garantir l'interopérabilité d'application et de composant, on cite par exemple COM de Microsoft, CORBA, le framework java de Sun [8].

8.3 Intégration du Processus Métier (des méthodes)

Cette méthode s'intéresse au partage de la logique d'affaires qui peut exister au sein de l'entreprise. L'intégration des méthodes est utilisée dans le cas où chaque application intégrée fourni des APIs et des méthodes fonctionnelles similaires [4].

Les mécanismes de partage des données par les applications sont nombreux, on cite par exemple : les objets distribués, les serveurs d'application, les moniteurs de gestion des transactions, les frameworks. La méthode la plus simple, consiste à réunir les opérations communes sur des applications multiples dans une application simple qui affronte les applications intégrées.

Il y a deux approches de base : la création d'un ensemble partagé de serveurs d'applications qui existent sur un serveur physique partagé, ou le partage direct des méthodes existantes déjà à l'intérieur des applications en utilisant par exemple la technologie des objets distribués.

L'intégration des méthodes se fait par les composants (CORBA, Enterprise JAVA Beans (EJB), Distributed Component Object Model (DCOM),…etc.), et nécessite que les applications à intégrer supportent les RPCs (Remote Procedure Call) ou la technologie des composants distribués [4].

8.4 Intégration de l'interface utilisateur

L'intégration de l'interface utilisateur est un plus primitif, mais néanmoins une approche nécessaire. Ce niveau d'intégration relie la logique d'intégration dans l'interface des utilisateurs [4]. Dans ce modèle d'intégration, les concepteurs et les développeurs peuvent empaqueter des applications en utilisant leurs interfaces utilisateur comme point commun d'intégration (la technique de screen scraping).

Par exemple, les applications de mainframe ne fournissent pas l'accès au niveau des bases de données ou au niveau de processus d'affaires (business process), mais on peut y accéder à partir des interfaces utilisateurs des applications.

Cette méthode est basée sur deux primitives : scripting ou proxing. Dans le cas de scripting, le code d'intégration est fusionné dans l'interface utilisateur. Par exemple l'inscription dans un site quelconque (boite email), dés que le client clique sur le bouton « submit » (enregistrer) toutes les informations seront stockées directement dans la base de

données de ce site qui est situé dans un autre serveur. Le cas de proxing est utilisé pour échanger les données avec les patrimoines applicatifs (Legacy systems) [4,7].

La plupart des développeurs considèrent que l'ajustement des interfaces des utilisateurs pour être des points d'intégration est une approche instable et ancienne.

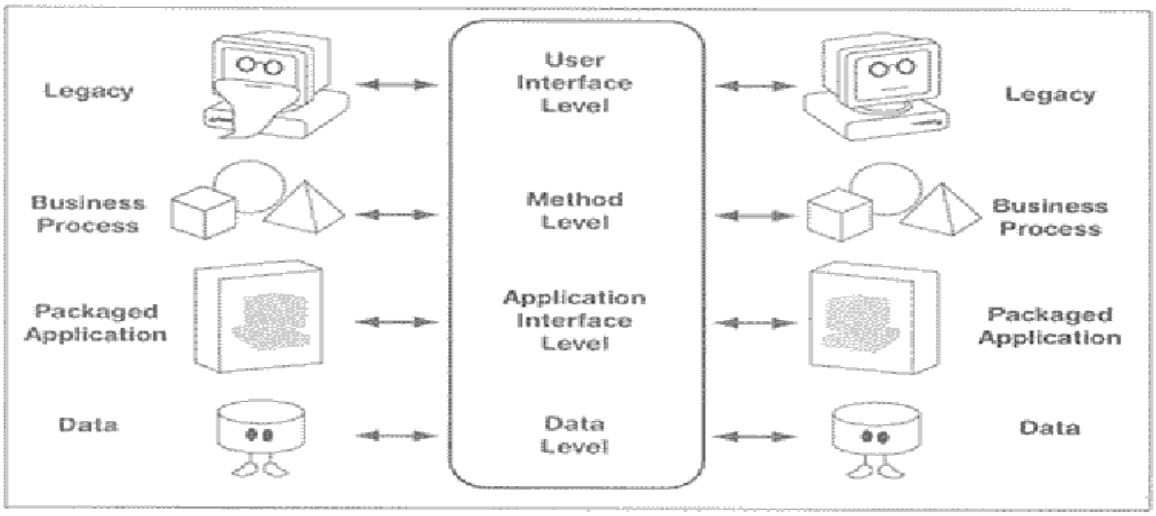


Figure 1.8. Les différents niveaux d'intégration [7].

De ce qui précède, nous pouvons distinguer quelques avantages et inconvénients de chaque niveau d'intégration [6,8] :

Niveau d'intégration	Avantages	Inconvénients
Intégration des données	• L'implémentation est simple. • Expérience étendue implique un usage fréquent. • Les changements aux applications existantes seront minimaux.	• La logique d'affaires n'est pas intégrée. d • Elle est difficile si les données sont stockées dans des entrepôts des différents vendeurs.
Interfaces des applications (API)	• Elle est transparente à l'application cliente. • L'intégrité des données des applications à intégrer est préservée. • La logique d'affaires est intégrée	• Les changements aux applications existantes sont étendus. • La transmission synchrone est requise. • Les applications seront fortement couplées. • L'intégration point à point

		est utilisée.
Processus Métiers	• L'approche de middleware est utilisée • le partage de la logique d'affaires est permis.	• Cette méthode est très compliquée. • Les applications seront fortement couplées
Interfaces utilisateurs	• L'implémentation est simple. • Les changements aux applications existantes seront minimaux. • la logique d'affaires est invoquée.	• L'intégration Point à point est utilisée • L'addition des nouvelles fonctionnalités est difficile • Les applications seront fortement couplées. • La communication est synchrone.

Tableau 3.1 Avantages et inconvénients des différents niveaux de l'EAI

9 Architecture type d'EAI

Cette architecture typique propose que l'EAI doit être constitué de trois couches successives : la couche d'intégration des processus, la couche de transformation et de routages des messages et enfin la couche de transport des messages. La communication entre ce EAI et les différentes applications intégrées est effectuée par des connecteurs spécialisés, comme il est illustré dans [Figure 1.9] :

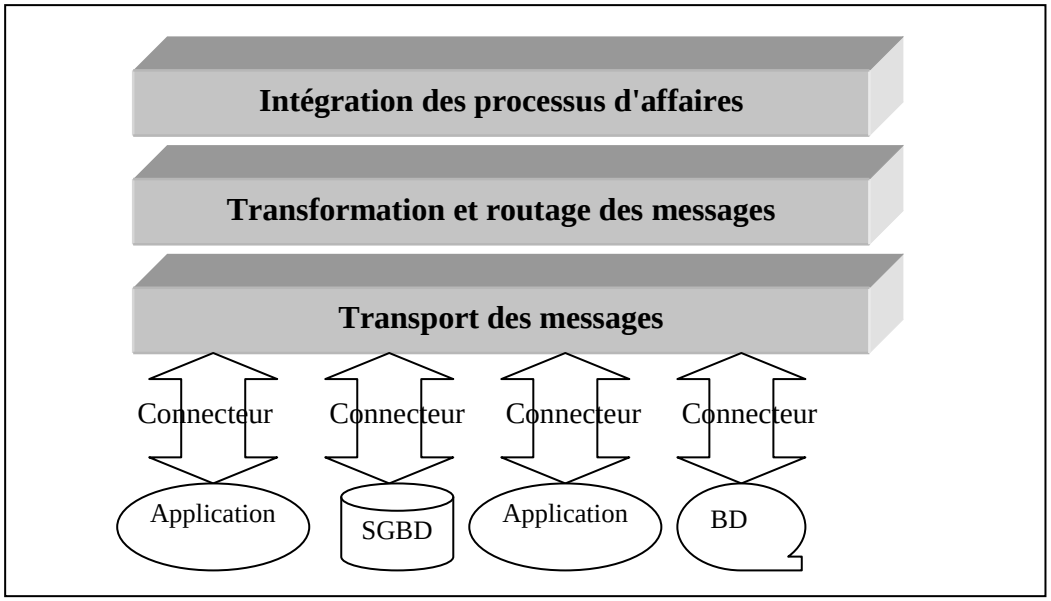


Figure 1. 9. Architecture typique d'un EAI

9.1 La couche du transport des messages

Cette couche assure le transport des messages depuis l'EAI aux applications et vice versa, la capture depuis les applications productrices ou la restitution aux applications consommatrices [7]. L'EAI utilise typiquement MOM (Message Oriented Middleware) comme un moyen fondamental de transport des messages entre les diverses applications et la couche de transformation et de routage. Puisque MOM emploie des files d'attente de message pour communiquer entre les applications, il y a un faible couplage entre le Middleware et l'application [8]. Elle peut utiliser HTTP, SOAP, SMTP, IIOP et RMI comme protocole du transport [7]. Pour fournir une couche d'isolation entre les applications et le MOM, des adaptateurs d'applications sont utilisés pour communiquer avec les différentes applications et pour se connecter à MOM pour envoyer et recevoir des messages. Les adaptateurs masquent les complexités des applications d'entreprise et permettent à des applications individuelles d'être modifiées et/ou améliorées sans la révision du système intégrée [8].

9.2 La couche de transformation et routage des messages

Puisque la transformation est une partie majeure dans le processus d'intégration, le rôle de cette couche est la conversion les données, XML ou non-XML, d'un format à un autre [4]. Le moteur XSL qui accomplit les transformations des messages XML [7]. Cette approche réduit au minimum les changements aux applications intégrées et réduit la future maintenance en centralisant la logique de transformation des données. En outre, les moteurs d'intégration peuvent déterminer le routage des messages intelligemment basé sur des règles prédéfinis et envoyer les messages édités à toutes les cibles nécessaires [8].

9.3 La couche d'intégration de processus d'affaires

Une des caractéristiques principales d'EAI est l'intégration de processus d'affaires. Il permet l'interopération des processus d'affaires en automatisant le flux et la distribution des messages aux systèmes appropriés. L'intégration de processus d'affaires est désigné sous le nom d'un moteur Workflow [8].

Les systèmes de gestion des affaires traditionnelles exigent souvent une programmation étendue et les révisions des systèmes quand les processus d'affaires sont changés. Avec l'EAI, l'intégration de processus d'affaires sépare la gestion des affaires du traitement des transactions, ce qui permet l'indépendance des changements. L'EAI offre des outils graphiques pour la modélisation des processus d'affaires et des flux d'information

complexes. Ceci permet aux utilisateurs d'établir et contrôler leurs processus d'une manière directe et naturelle. Quand les processus d'affaires changent, des modèles peuvent être changés sans la reprogrammation [8]. L'isolement des processus d'affaires des systèmes permet à des entreprises d'expérimenter les processus d'affaires pour arriver à des résultats optimaux.

10 Avantages de l'EAI

L'EAI a un très grand effet sur les activités des entreprises dans plusieurs niveaux et influe directement sur leurs rendements. Parmi ces avantages nous trouvons [8] :

a. Réduction du coût de développement et de maintenance

L'architecture ouverte de l'intégration permet l'ajout économique et sans problèmes de nouveaux systèmes dans une entreprise. La simplicité de l'architecture diminue également la gestion du système et le coût de la maintenance. En plus, la séparation de la logique d'affaires de la capacité de traitement transactionnel permet l'adaptation rapide aux climats économiques sans utilisation d'une application principale de réorganisation.

b. Amélioration de la performance et de la fiabilité

Les solutions d'EAI utilisent principalement les mécanismes de transmission des messages asynchrones semblables à la transmission basée par email pour transporter l'information de la source vers des applications destinataires. Ceci permet à l'expéditeur de retourner immédiatement à son traitement sur la transmission de message. En outre, la livraison de messages est garantie. Ce modèle de transmission augmente considérablement la performance et la fiabilité du système intégré.

c. Utilisation d'un bus d'information centralisé

Aux industries telles que les télécommunications et les services financiers qui utilisent souvent les environnements transactionnels répartis pour des différents produits, l'EAI peut être utilisé comme le circuit principal de l'information pour unifier les sources des données, les applications et les processus isolés, ce qui permet aux entreprises de poursuivre son évolution normalement. C'est l'une des raisons essentielles pour lesquelles ces deux industries ont été les premières à déployer des solutions d'EAI.

d. Extension du cycle de vie d'une application patrimoniale

Cet avantage est le but de notre mémoire où nous cherchons des méthodes et des techniques permettant d'étendre le cycle de vie d'un système patrimonial parce que beaucoup

d'entreprises font face au problème de manipulation de leurs applications patrimoniales qui sont considérées comme le noyau de leurs business. Ces systèmes sont âgés peut-être de plus de 30 ans et utilisent un groupe de 50 personnes ou plus pour la maintenance.

Afin de pouvoir intégrer les applications qu'elles souhaitent implémenter, les entreprises doivent d'abord auditer leurs systèmes patrimoniaux (Legacy systems) et les faire évoluer en fonction de leurs besoins [3]. Pour cela, nous considérons que l'intégration des applications patrimoniales des entreprises dans l'EAI est une nouvelle étape dans le cycle de vie de ces derniers (pour plus des détailles voir le chapitre 3).

e. Délai d'arrivée au marché très réduit

La flexibilité apportée par l'EAI permet à des entreprises de personnaliser facilement les règles de métiers existantes et d'étendre les fonctionnalités des applications. Ceci permet à des entreprises d'expérimenter plus aisément les nouveaux concepts, pour s'ajuster plus efficacement en se basant sur la réaction du client, et pour présenter de nouveaux produits dans un temps plus court.

11 Conclusion

Les solutions d'EAI existantes ne sont pas la réponse finale aux défis d'intégration mais elles offrent une valeur immédiate aux entreprises recherchant l'intégration principale pour supporter l'eBusiness et des projets avantagés. Elles offrent aussi la capacité des transactions distribuées, le processus complexe et l'automatisation de Workflow... etc.

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,en remplaçant les protocoles de communication par les normes de services Web.

Le chapitre suivant sera destiné à l’illustration du domaine des services Web et à signaler leurs impacts sur les activités des entreprises et le monde industriel en entier.

Chapitre 2 :

Caractéristiques des services Web

1 Introduction

Un des problèmes se 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 intégré, rapide 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, EJB du SUN...etc. Le problème toujours est resté posé dans le manque de la standardisation.

Dernièrement, un nouveau paradigme est apparu comme la solution le plus adéquate aux 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 sont arrivés pour permettre les échanges des informations entre différents systèmes d'information à travers le Web et en utilisant des standards tels que HTTP, SOAP, WSDL, UDDI,...etc.

Un service Web est considéré comme une interface placée entre le code d'application et l'utilisateur de ce code [16]. Il agit en tant que couche d'abstraction, séparant la plateforme et les détails de langage de programmation de la façon dont le code d'application est appelé réellement. Cette couche normalisée signifie que n'importe quelle langage supportant la technologie des services Web peut accéder à la fonctionnalité de l'application.

Ce deuxième chapitre est consacré pour étudier le domaine des services Web et leurs impacts sur le monde industriel. Nous allons d'abord présenter la notion d'Architecture Orientée Service (SOA), qui est considérée comme la metaclasse des services Web. Ensuite nous exposons quelques définitions tirées de la littérature. Dans la section qui suivra, 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 est composée de six couches. Nous discutons également sur les effets des services Web sur le monde industriel et sur les systèmes d'information dans la section suivante. Ensuite, nous allons présenter la notion du l'orchestration ou la chorographie des services Web. Enfin, nous donnerons un exemple des systèmes de gestion des processus métiers qui est BPEL4WS (Business Processus Execution Language For Web Services), tout en montrant les avantages et les inconvénients des services Web.

2 Architecture Orientée Service (SOA)

La programmation orientée service est la prochaine étape qui suit la programmation orientée objet (POO) et le développement basé sur les composants (D.B.C) [11]. L'architecture orientée service est considérée comme le modèle architectural de la prochaine génération qui permet d'étendre la flexibilité et la portabilité de l'infrastructure des systèmes informatiques existants dans l'entreprise [12].

L'approche de l'architecture orientée service (SOA) est fortement associée aux services Web, bien qu'elle les précède de plusieurs années. C'est une approche permettant de créer des applications réparties. Aujourd'hui, elle est proposée comme étant la meilleure solution pour l'intégration des applications multiples, le problème du l'intégration des applications de l'entreprise (EAI) [13] (voir chapitre 1). Cette approche aux systèmes et aux applications métiers d'entreprise considère des ressources de logiciel comme services disponibles et découvertes sur un réseau. De tels services fournissent les fonctionnalités aux affaires de l'entreprise tout en cachant les détails fondamentaux de l'implémentation [12].

Une architecture orientée service fait participer typiquement trois participants principaux [Figure 2.1] : fournisseur de services, Broker (courtier) de services, et demandeur de services [16].

- Le Fournisseur de services est pour la création de services et l'édition de sa description dans un entrepôt appelée UDDI (Universal Description, Discovery, and Integration)
- Le Broker (courtier) du service met à jour l'UDDI et il agit en tant que pages blanches et jaunes de services pour les services Web.
- Le Demandeur du service trouve un service dans l'UDDI auquel il se relie.

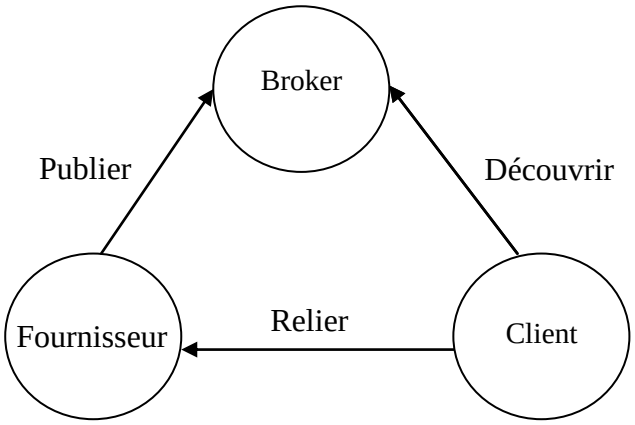


Figure 2.1. Architecture Orientée Service [16].

Gartner a prévu que l'Architecture Orientée Service (SOA) sera la méthode la plus pratique dans le génie logiciel avec la dominance absolue [11].

3 Notion des services Web

Les services Web sont basés sur le concept de l'architecture orientée service (SOA). Comme il a été expliqué dans la section précédente, SOA est la dernière évolution de l'informatique répartie, qui permet des composants d'un logiciel, y compris des fonctions d'application, des objets, et des processus de différents systèmes à être exposés comme services.

Selon les travaux de Gartner[11] « les services Web sont des composants logiciels faiblement couplés, fournis à travers les technologies standard de l'Internet ».

Les services Web, sont des applications métiers auto-explicatives et modulaires. Ils exposent la logique métiers comme services par l'intermédiaire de l'Internet en utilisant seulement des interfaces programmables et des protocoles d'Internet afin de fournir des moyens pour trouver, s'inscrire et appeler ces services [15].

Selon [16], les services Web sont des applications modulaires, auto-contenant, auto-décrivant, pouvant être éditées, publiées, et appelées à travers le Web. Les services Web sont des fonctions métiers rendues universellement disponibles sur l'Internet. Ils permettent l'interopérabilité des applications dans un environnement faiblement couplé, découvrant et se reliant dynamiquement aux services sans aucun agreement précédent ayant été établi entre eux.

Le groupe W3C [14] 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 et est décrit dans un format exploitable par machine, spécifiquement WSDL (Web Service Description Language). Les autres systèmes agissent avec le service Web en quelque sorte exigée par sa description en utilisant des messages 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 Composants d'une architecture service Web

Dans cette section, nous définissons les rôles individuels pour chaque acteur dans une architecture basée sur les services Web, qui sont: le fournisseur de services, le demandeur de services et le registre des services [Figure 2.2].

a. Fournisseur de services (Serveur)

Le fournisseur de services implémente les services Web et les rend disponibles sur l'Internet [19].

b. Demandeur de services (Client)

C'est n'importe quel consommateur du service Web. Le demandeur utilise un service Web existant en ouvrant une connexion de réseau et en envoyant une demande de XML [19].

c. Registre des services

C'est un répertoire de services logiquement centralisé. Le registre fournit un endroit central où les développeurs peuvent éditer des nouveaux services ou trouver celles existantes. Il sert donc de chambre de réparation centralisée des services [19].

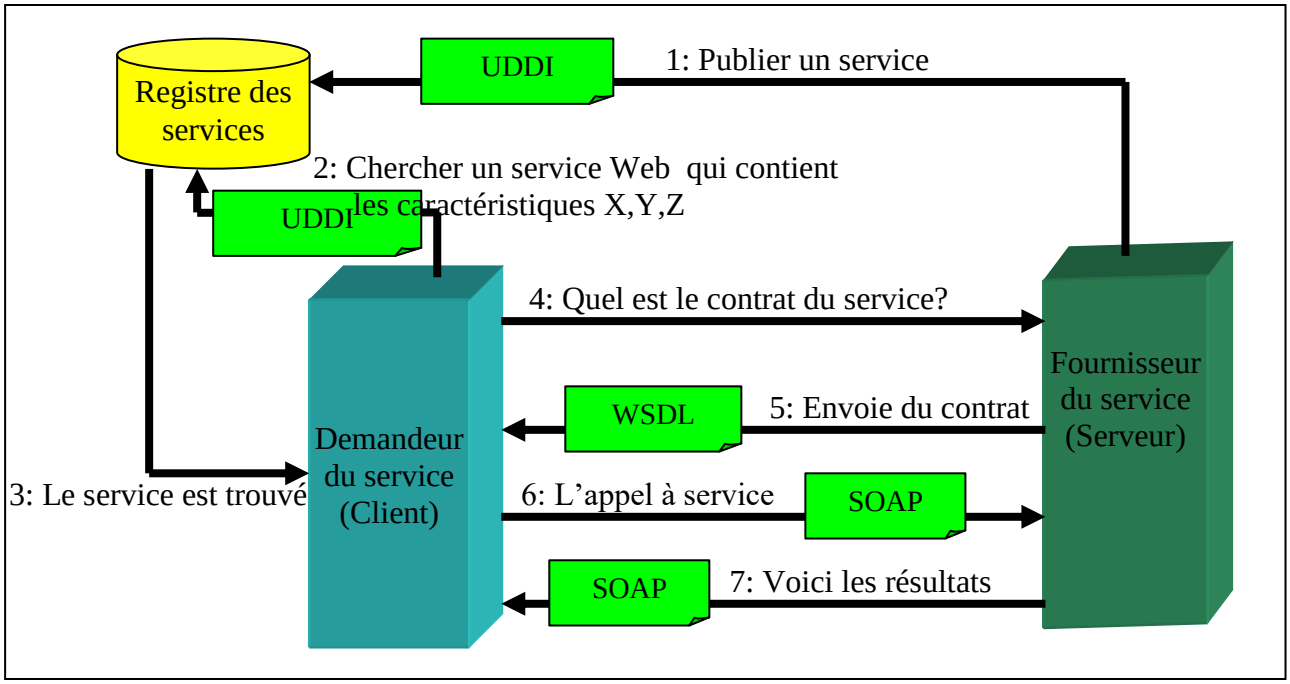


Figure 2.2. Les rôles dans les services Web [17]

5 Standards du Web

Selon les définitions que nous avons vu dans les sections précédentes, nous trouvons que les services Web sont fortement reliés au moins à quatre standards principaux , qui sont : XML, SOAP, WSDL et UDDI :

5.1 XML

XML (eXtensible Markup Language) est une famille de technologies développées par le XML Working Group au sein du World Wide Web Consortium (W3C). XML est née de la tentative de mettre SGML sur le Web. La première spécification de XML est apparue en février 1998 et se concentre sur les données [20]. Il permet d'échanger des informations marquées par des balises décrivant la structure et la sémantique des contenus. Ces derniers peuvent ensuite être présentés d'une manière diversifiée à l'aide de feuilles de style. XML permet la gestion des documents et de contenus sous une forme structurée facilitant les échanges et la publication sur supports variés. XML présente l'avantage de simplicité et de pouvoir être facilement compris par un grand nombre des outils et d'utilisateurs.

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 des hyperliens 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 langages particuliers créés 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 le commerce électronique, XMI (XML Metadata Interchange) pour l'échange des modèles UML ou encore DSML (Directory Service Markup Laguage) pour les annuaires. Donc, selon la spécification du W3C, le langage XML décrit :

- une classe d'objets de données appelées documents XML,
- et, partiellement, le comportement des programmes qui les traitent.

5.2 SOAP

SOAP (Simple Object Access Protocol) est un protocole standard destiné aux services Web. Il est Lancé par IBM et Microsoft, dont le but est de définir un protocole qui permet la structuration des messages échangés par les applications via l'Internet [17].

SOAP est un protocole indépendant de toute plateforme et tout langage de programmation. Il réalise le codage universel des appels et des réponses de procédures en

XML. Ce codage peut être vu comme un simple codage des appels de type RPC (Remote procedure Call). Couplé à WSDL (Services Web Description Language), SOAP fournit à l'application cliente les outils nécessaires pour invoquer des services distants en lui donnant l'illusion qu'ils sont locaux [7].

5.2.1 Structure d'un message SOAP

Selon [7], un message SOAP est défini comme un document XML de racine <Envelope> contenant un élément <Heading> et un élément corps <Body>, permettant d'invoquer une opération distante ou de transmettre la réponse de cette opération. Le message SOAP inclut l'entête du protocole support (HTTP, SMTP,...). Les messages SOAP peuvent être échangés sur HTTP permettant le franchissement du pare-feu (FireWall) et des contrôles associés. La structure du message SOAP est présentée dans [Figure 2.2] :

a. Enveloppe du message SOAP

L'enveloppe du message SOAP est la racine du document XML contenant le message SOAP et précisant diverses opérations telles que les espaces de noms et le style de codage, comme il est illustré dans l'exemple suivant :

```
< ENV:Envelope xmlns:ENV="http://schemas.xmlsoap.org/soap/envelope/"
  ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
…
</ENV:Envelope>
```

b. Entête SOAP

Les entêtes définissent un mécanisme très élégant et simple pour étendre les messages SOAP d'une manière modulaire et optionnelle. L'entête permet de passer au partenaire des informations de contrôle comme par exemple l'authentification et l'autorisation, la gestion de transaction, la gestion du paiement, le suivi et la vérification...etc. Des espaces de noms peuvent être déclarées à ce niveau [7,18]. Soit l'exemple suivant qui présente une demande d'engager la transaction N°7 :

```
< ENV:Enveloppe xmlns:ENV="http://schemas.xmlsoap.org/soap/envelope/"
  ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
<ENV:Header>
<t:Transaction xmlns:t ="http://www.pg2005.dz/2007/04/"  ENV:mustUnderstand="1">
<t:trid>7</trid>
<t:action> Commit</action>
</t:Transaction>
</ENV:Header>
…
</ENV:Envelope>
```

c. Corps SOAP

L'élément <Body> entoure immédiatement les informations noyau du message SOAP. C'est lui qui contient les appels des procédures ou les rapports des erreurs. Il peut comporter de 0 à N éléments fils appelés blocs. Ceux-ci peuvent être qualifiés par un ou plusieurs espaces de noms et par un style de codage [7,18]. Les méthodes invoquées par le corps de message SOAP sont décrites en WSDL (Web service Description Language). L'exemple suivant présente le corps d'un message d'erreur en SOAP

```
<env:Body>
<env:Fault>
<Fault>env:Client</Fault>
<FaultString>Le paramètre est incorrect </FaultString>
<detail = "212" />
</env:Fault>
</env:Body>
```

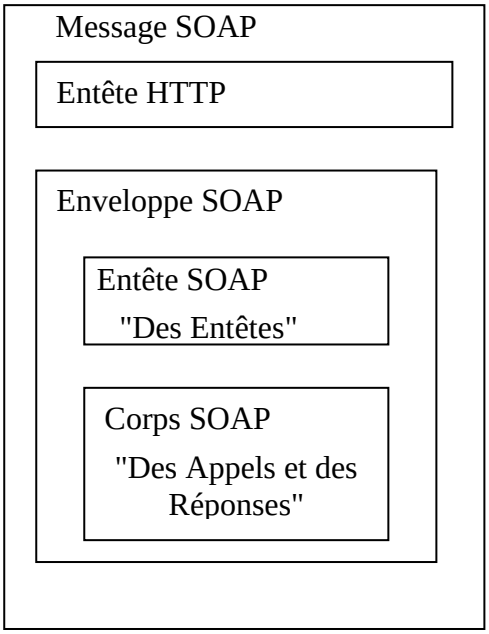


Figure 2.3. Format d'un message SOAP [17]

5.2.2 Types de message SOAP

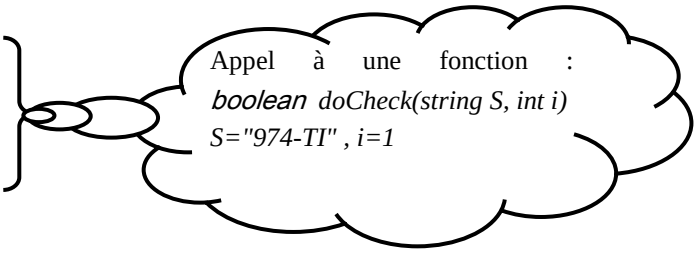
Le protocole SOAP a définit trois types de messages utilisés dans l'échange entre le fournisseur et le client du service Web :

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 fonction

booléenne qui reçoit deux paramètres, le premier est une chaîne de caractères et le deuxième est un entier :

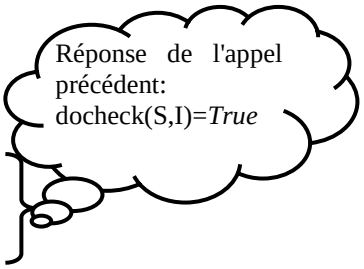
```
POST /bws/inventory/InventoryCheck.jws HTTP/1.0
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 426
SOAPAction: ""
<?xml version="1.0" encoding="UTF-8"?>
< ENV:Envelope ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<ENV:Body>
<doCheck>
<arg0 xsi:type="xsd:string">947-TI</arg0>
<arg1 xsi:type="xsd:int">1</arg1>
</doCheck>
</ENV:Body>
</ENV:Envelope>
```



b. Réponse

Le message de réponse est optionnel, émis par le fournisseur de services. Il est créé dans le cas où il y a des informations de retour qui doivent être renvoyées au demandeur du service. Un exemple présentant un message SOAP répondant à l'appel de l'exemple précédent, est le suivant :

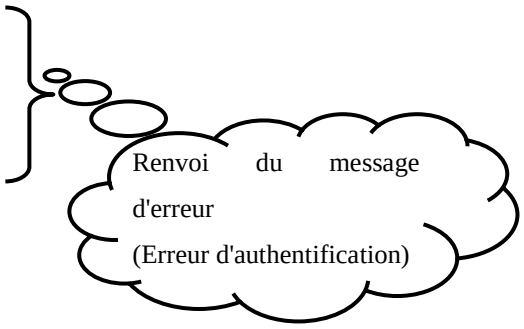
```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 426
<?xml version="1.0" encoding="UTF-8"?>
<ENV:Envelope
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<ENV:Body>
<doCheckResponse>
<doCheckResult xsi:type="xsd:boolean">true</doCheckResult>
</doCheckResponse>
</ENV:Body>
</ENV:Envelope>
```



c. Erreur

Ce message est aussi optionnel, émis par le fournisseur de services, il est utilisé dans le cas d'échec d'exécution du service dans le serveur. L'exemple suivant présente un message d'erreur qui est généré suite à une erreur d'authentification :

```
HTTP/1.0 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
<ENV:Envelope xmlns:ENV="http://schemas.xmlsoap.org/soap/envelope/"
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<ENV:Body>
<ENV:Fault>
<faultcode>Client.AuthenticationFailure</faultcode>
<faultstring>Failed to authenticate client</faultstring>
<faultactor>urn:X-SkatesTown:PartnerGateway</faultactor>
</ENV:Fault>
</ENV:Body>
</ENV:Envelope>
```



5.3 WSDL

WSDL (Web Service Description Language) est un dialecte XML dédié à la description de tous les éléments nécessaires pour interagir avec un service Web. Il est supporté principalement par Ariba, IBM et Microsoft. Il a été proposé en 2002 par W3C pour la standardisation [17]. WSDL peut être considéré comme un complément de SOAP car il facilite l'interopérabilité des services Web. Il joue le rôle de descripteur des services Web tout comme IDL (Interface Description Language) de CORBA [7].

Grâce à WSDL, les applications utilisant SOAP sont capables de générer les souches logicielles réalisant les échanges avec les services Web. Pour le client, un compilateur WSDL doit être capable de générer le code d'appel du service, alors que pour le serveur il génère le code réalisant l'interface entre le protocole et le service.

WSDL introduit les notions de Type, Message, Port, Liaison et Services pour décrire les services [7,18]:

a. Type

Il définit l'ensemble des types de paramètres, en utilisant un système de typage du schéma XML. L'exemple suivant représente la définition de l'enregistrement *availabilityType* de 03 champs :

(sku :string, price :double, quantityAvailable : entier) :

La déclaration de cet enregistrement selon le format WSDL sera comme suit:

```
<types>
<xsd:schema targetNamespace="http://www.skatestown.com/ns/availability"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:complexType name="availabilityType">
```

```
<xsd:sequence>
<xsd:element name="sku" type="xsd:string"/>
<xsd:element name="price" type="xsd:double"/>
<xsd:element name="quantityAvailable" type="xsd:integer"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
</types>
```

b. Message

C'est un groupe de données codées en XML décrit les noms et les types d'un ensemble de champs à transmettre, permettant d'invoquer une opération attachée à un port depuis un client ou de retourner une réponse au client. Il peut être comparé aux paramètres d'un appel de procédure. La section <message> décrit l'appel de la procédure « *PriceCheckRequest* » et l'autre décrit la réponse pour la même procédure « *PriceCheckResponse* »:

```
<message name="PriceCheckRequest">
<part name="sku" type="xsd:string"/>
</message>
<message name="PriceCheckResponse">
<part name="result" type="avail:availabilityType"/>
</message>
```

c. Liaison (Binding)

Elle permet la spécification du type d'accès, du protocole de transport utilisé et du codage des paramètres dans les messages pour chaque type de port, comme il est illustré dans l'exemple suivant, qui décrit comment définir la liaison WSDL—SOAP pour un service «*checkPrice* » :

```
<binding name="PriceCheckSOAPBinding" type="pc:PriceCheckPortType">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="checkPrice">
<soap:operation soapAction=""/>
<input>
<soap:body use="encoded" namespace= "http://www.skatestown.com/services/PriceCheck"
encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/"/>
</input>
<output>
<soap:body use="encoded" namespace= "http://www.skatestown.com/services/PriceCheck"
encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/"/>
</output>
</operation>
</binding>
```

d. Le type de port (portType)

C'est une définition abstraite de l'interface d'un service Web, où chaque élément définit une signature abstraite d'une opération. Soit l'exemple suivant qui définit le type du port «*PriceCheckPortType*», pour le service « *CheckPrice* » :

```
<portType name="PriceCheckPortType">
 <operation name="CheckPrice">
<input message="pc:PriceCheckRequest"/>
<output message="pc:PriceCheckResponse"/>
</operation>
</portType>
```

e. Port

Il est considéré comme un point de terminaison permettant l'identification d'un groupe d'opérations et l'accès à un service Web via un protocole spécifique. Le port est identifié de manière unique par la combinaison d'une adresse Internet et d'une liaison.

f. Service

Il est utilisé pour la spécification de la collection des ports et les URLs permettant l'accès au service Web, comme il est illustré dans l'exemple suivant :

```
<service name="PriceCheckService">
     <port name="PriceCheck" binding="PriceCheckSOAPBinding">
  <soap:address location= "http://www.skatestown.com/axis/services/PriceCheck "/>
     </port>
</service>
```

Alors, on peut dire que le *Type de port* (<PortType>) avec les détails contenu dans *le message* (<Message>) et *le type* (<Type>) décrit *qu'est ce que* le service Web. L'élément *Liaison* (<Binding>) décrit l'aspect *comment*, et *le port* et *service* décrivent l'aspect *où* du service Web [18].

L'exemple suivant présente un canevas d'un document WSDL utilisé pour décrire le service Web « PriceCheck » qui reçoit l'identificateur d'un article donné et retourne un enregistrement de trois champs : la désignation d'un article, son prix et sa quantité en stock :

```
<?xml version="1.0"?>
<definitions name="PriceCheck"
targetNamespace="http://www.skatestown.com/services/PriceCheck"
xmlns:pc="http://www.skatestown.com/services/PriceCheck"
xmlns:avail="http://www.skatestown.com/ns/availability"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
```

```
<Types>
<!--  Définition des types de données échangées--!>
...
</Types>
<message>
<!-- Déclaration des messages (entrées et sorties) --!>
...
</message>

<portType>
<!-- Déclaration des opérations (par association des messages) --!>
...
</portType>
<Binding>
<!-- Définition de la liaison WSDL-SOAP (noms d'actions et codage)--!>
</Binding>
<service name="PriceCheckService">
...
<Port>
<!--Déclaration des ports (entrées groupes d'opérations et protocoles d'accès)--!>
</Port>
...
</service>
</définitions>
```

5.4 UDDI

UDDI (Universal Discovery, Description and Intégration) est une spécification en langage XML d'un catalogue des services offerts par les entreprises sur leurs sites Web, lancée par Ariba, IBM et Microsoft. L'annuaire UDDI permettre à un logiciel de reconnaître automatiquement les services dont il a besoin et de s'interfacer avec eux. Le catalogue comprendra les adresses et les contacts des entreprises, une classification sectorielle et une description des services proposés. Toutes les spécifications techniques sont fournies sur le site officiel du UDDI [17].

L'annuaire UDDI propose un cadre technique indépendant des plateformes et ouvert pour permettre aux entreprises d'enregistrer et découvrir les services Web [7].

Architecture du UDDI

Puisque l'annuaire UDDI est un registre accessible en SOAP comme des services [7]. Il pose des conditions de base à considérer, qui sont [18] :

- La définition d'un ensemble de structures de données pour que les méta-services soient stockés dans le registre.

- La définition des opérations usuelles, création, lecture, modification et suppression pour le stockage, la suppression et la recherche des services dans l'UDDI.

L'UDDI inclut un schéma XML qui décrit les quatre types essentiels d'information :

a. Les pages blanches (Business Entity)

Ceux-ci incluent des informations générales au sujet d'une entreprise spécifique. Nous citons par exemple l'identifiant de business qui est unique pour chaque entité, nom de l'entreprise, la description des activités, les informations du contact, l'adresse et le numéro du téléphone...etc [9,19]. Les pages blanches contiennent les éléments suivants [18] :

- *BusinessKey* : Identifiant de l'entité.
- *AuthorizedName* : Nom de l'individu qui a publié l'entité.
- *Operator* : Nom de l'opérateur qui gère la copie maître.
- *Name* : Nom externe de l'entité.
- *Description* : Description des activité de l'entité.
- *CategoryBag* : Liste des catégories auxquelles appartient l'entité.
- *BusinessServices :* Description des services offerts.

b. Les pages jaunes (Business Services)

Les pages jaunes incluent des données de classification générale pour l'entreprise ou le service offert. Cette classification est basée sur les taxonomies standard. Par exemple, ces données peuvent inclure l'industrie, le produit, ou les codes géographiques [19]. Les pages jaunes contiennent les éléments suivants [18] :

- *ServiceKey* : Identifiant du service.
- *BusinessKey* : Identifiant de l'entité à laquelle appartient le service.
- *Name* : Nom du service.
- *Description* : Description du service.
- *CategoryBag* : Liste des catégories auxquelles appartient le service.
- *BindingTemplates* : Structure contenant les éléments techniques.

c. Les pages vertes (Binding Templates)

Ces pages contiennent des informations techniques du service Web. Généralement, elles incluent un pointeur vers une spécification externe et une adresse d'invocation du service Web. L'UDDI ne décrit pas seulement les services Web basés SOAP, mais il est utilisé pour la description de n'importe quel service [19]. Les pages vertes contiennent les éléments suivants [18] :

- *BindingsKey* : Clé unique pour une spécification technique.
- *ServiceKey* : Clé du service auquel appartient la description technique.
- *Description* : Description du point d'entrée du service technique.
- *AccessPoint* : URL, email ou numéro de téléphone permettant l'accès au service.
- *TModelinstances* : Références vers les descriptions techniques spécifiques du service offert.

d. tModel (Technology Model)

tModel représente le modèle technique. Cet élément permet aux groupes industriels, à des organisations de normalisation et à des différentes entreprises de spécifier les définitions abstraites de leurs services, pour que les autres peuvent les utiliser, les combiner, et créer une signature pour un service.

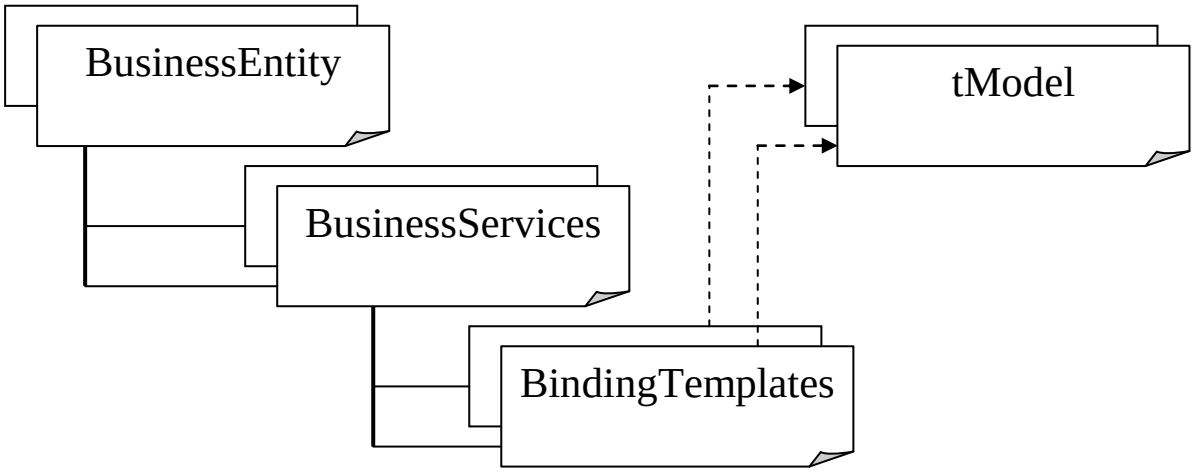


Figure 2.4. Entités composant l'UDDI [18]

6 Pile du protocole des services Web

Chaque vendeur, organisation de normalisation ou firme de recherche possède sa propre définition du service Web. Cette variété de définitions rend l'architecture de la pile du protocole du service Web varier d'une organisation à une autre. Le nombre et la complexité des couches de la pile dépendent de l'organisation. Mais tous les modèles contiennent

généralement les couches de gestion des processus métiers (Business Process Management), en plus des couches de messagerie et de description des services.

Couche	Exemple
Négociation	Les accords entre les partenaires commerciaux
Orchestration	IBM WSFL, MS XLANG
Découverte et Publication	UDDI, ebXML registeries, WSDL/WSCL
Echange des messages	SOAP/XML Protocol
Protocoles de transport	HTTP, HTTPS, FTP, SMTP
Issues de Business	La Gestion, Qualité de service, La sécurité, Les standards ouverts

Tableau 2.1. Architecture générique du comité ***WebServices.Org***

Parmi ces propositions, nous avons choisi celle du comité ***WebServices.Org*** [Tableau 2.1], qui est simple et générique. Les couches qui composent cette architecture sont les suivantes [21] :

a. Négociation

Le processus de la logique de business est déclenché à partir de la couche Négociation des services, qui permet à deux partenaires commerciaux de négocier et convenir sur les protocoles employés pour combiner des services Web. Cette couche est appelée aussi couche de définition des processus.

b. Orchestration

Cette couche est dédiée au traitement du Workflow. Nous prenons par exemple le langage IBM WSFL (Web Service Flow Language) et MSXLANG qui est un langage de format XML pour décrire le Workflow et le générer. WSFL indique comment un service Web est interfacé avec d'autres. Avec WSFL, Nous pouvons déterminer si les services Web devraient être traités comme activité dans un déroulement des opérations (Workflow) ou comme série d'activités.

Tandis que WSFL complète WSDL et basé sur les transitions, XLANG est une extension de WSDL et basé sur des blocs structurés. WSFL supporte deux types de modèles : flux et global. Le modèle du flux décrit les processus métiers (Business Process) qui est achevé par une collection de services Web. Le modèle global décrit comment les services

Web agissent l'un sur l'autre. XLANG, d'autre part, permet l'orchestration des services Web dans des processus métiers et des services Web composés. WSFL est fort pour la présentation du modèle tandis que XLANG marche bien avec l'interaction longue des services Web.

Parmi les logiciels qui supportent WSFL, nous trouvons celui de « IBM MQ Workflow série », maintenant est connu sous le nom de gestionnaire de processus de « WebSphere ». Il automatise les flux des processus métiers, optimise l'intégration des applications d'entreprise (EAI) avec le Workflow de personnes, permet l'extensibilité, se conforme aux associations des Workflow et aux capacités des multi-plateformes. MS XLANG est un langage implémenté dans BizTalk.

c. Découverte et Publication

La première version d'UDDI est devenue totalement opérationnelle en mai 2001, permettant à des entreprises d'enregistrer et de découvrir des services Web sur l'Internet. Son intention initiale était de créer des catalogues électroniques dans lesquels des entreprises et les services pourraient être listés.

En plus des sites d'UDDI publics, les entreprises peuvent également déployer des registres privés sur leurs Intranet pour contrôler des services Web internes en utilisant la spécification d'UDDI. L'accès à l'information interne de service Web peut également être étendu à un réseau privé des associés.

WSCL (Web Service Conversational Language) aide les développeurs dans l'utilisation du schéma XML pour mieux décrire la structure des données dans un format commun. Ce protocole peut être utilisé pour spécifier l'interface du service Web et décrire les interactions entre les services.

d. Echange des Messages

Dans la couche de messagerie dans la pile, le protocole SOAP agit en tant qu'enveloppe pour des messages basés XML, l'encodage des messages, la garantie du routage, la livraison et la sécurité des messages.

e. Protocoles de transport

La couche de transport utilise : HTTP (Hypertext Transfer Protocol), HTTP sécurisé (HTTPS), HTTP fiable (HTTP Reliable), FTP (File Transfer Protocol) ou SMTP (Standard Mail Transfer Protocol). Puis, chaque service Web fait un tour dans l'Internet pour fournir à

un demandeur les services ou présenter un rapport d'état à un fournisseur ou à un courtier de services (Service Broker).

f. Issues de Business

Dans cette couche, une liste des points très important pour la bonne marche des services Web est définie, qui sont par exemple : Management, Qualité des services, sécurité, les standards ouverts..

7 Intérêts des services Web

D'après les spécialistes du SOA, les services Web ont un effet très important sur plusieurs domaines, nous trouvons par exemple : l'intégration des applications (EAI et B2B), la réutilisation des services et la flexibilité du business [20]:

a. Intégration

Dans le passé, les données et la logique d'une application ne peuvent pas être utilisées dans une autre application. Cette problématique a donné naissance au domaine de l'EAI. Dans les solutions traditionnelles de l'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 entreprises d'améliorer leur investissement dans leurs applications critiques, plutôt que de réécrire l'application ou les faire migrer vers une nouvelle technologie. Cependant, l'EAI traditionnel a présenté un ensemble de problèmes, liés aux coûts de logiciel d'intégration, à la complexité, à l'effort et à l'inflexibilité des solutions. Ces problèmes obligent les entreprises à trouver d'autres solutions.

Parmi ces solution nous trouvons l'intégration des applications basée sur les services Web, qui attaque les trois problèmes cités ci-dessus avec un nouveau modèle qui est à prix réduit, plus facile à apprendre et à déployer, et plus adaptable aux business changeants au besoin. Le résultat est l'intégration des applications plus rapidement et plus facilement, permettant à des entreprises de relier beaucoup plus des applications au sein de leur entreprise et ouvrant la porte pour des processus métiers plus efficaces au sein et à travers des entreprises.

b. Réutilisation des services

Une des caractéristiques les plus importantes des services Web est la capacité de réutiliser un service plusieurs fois plutôt que de créer un nouveau service pour la même fonction pour répondre aux exigences de l'application consommant.

c. Flexibilité du business

Historiquement, les applications d'affaires sont plutôt rigides. L'enrichissement de ces applications avec de nouvelles fonctionnalités pour répondre aux exigences changeantes de business est très coûteux, complexe, et consomme beaucoup de temps. Cependant, les services Web fournissent un pas solide permettant d'établir rapidement une solution d'intégration basé service Web à un coût raisonnable. Puisqu'il est plus facile de réutiliser des services par des clients multiples dans l'environnement du service Web, une flexibilité de business supplémentaire est achevée.

La plupart du temps, les changements effectués sur les fonctionnements intérieurs d'un système n'affectera pas le service Web. Par exemple, changer la structure de base de données du système de traitement de commande (système de gestion commercial) n'exigera aucun changement des applications qui agissent l'un sur l'autre avec ce système par des services Web.

8 Orchestration des services Web

La définition des standards pour les modèles de processus métiers, permettrait à des processus d'être modelés, déployés, exécutés et contrôlés par les logiciels des différents vendeurs. En absence de tels standards, un certain nombre de conséquences indésirables surgissent. Celles-ci incluent [22]:

- Les constructeurs vont offrir des solutions exclusives, contre le principe de standardisation des services Web
- Les entreprises collaborées peuvent choisir des moyens incompatibles pour définir les modèles des processus partagés, menant aux inefficacités et aux opérations sensibles aux erreurs.
- La réutilisation des processus et des configurations à travers des produits des différents constructeurs est très difficile.
- l'apparition des outils améliorés pour modéliser et exécuter des processus sera gênée.

8.1 Caractéristiques des systèmes de gestion des processus d'affaires

Les fonctionnalités essentielles qui doivent être considérées par un système de gestion des processus d'affaires, sont les suivants :

a. Modélisation des processus d'affaires

Dans un modèle de collaboration d'un processus, les processus sont décrits comme un ensemble de collaborations entre les différents participants, incluant les organisations, les applications, les employés et d'autres processus d'affaires [22].

b. Composition de services Web

La composition de services permet d'intégrer un enchaînement de services Web de façon à fournir un nouveau service. L'intégration peut être simplement une séquence, mais aussi une exécution parallèle ou conditionnelle, ou une alternative [9]. Généralement la composition et la décomposition récursive des processus sont bien exigées [22].

c. Codification et exécution de Workflow

Un Workflow décrit comment les participants dans un processus collaborent pour exécuter un processus du début jusqu'à la fin. Un Workflow décrit d'autre part comment les participants échangent les données et aussi les conditions d'enchaînement entre les activités. La majorité des standards de Workflow supportent la représentation des sous modèles, qui permet l'implémentation des activités dans un Workflow comme un autre Workflow. Nous distinguons généralement le flux d'information du flux de contrôle. Le flux du contrôle défini la séquence des différentes activités dans un processus. Le flux d'information décrit comment les informations seront échangées entre les activités. Un Workflow peut être représenté par un réseau de Petri à prédicats, les transitions correspondent aux activités, les messages aux jetons et les conditions aux prédicats [9,22].

d. Gestion de transactions et exceptions

Les transactions sont des blocs cruciaux pour n'importe quel processus et il faut que le standard de gestion des processus d'affaires fournisse un mécanisme de gestion des transactions. Les transactions longues, durant des heures ou même des semaines, doivent être supportées. En cas d'échec, il faut pouvoir compenser certaines activités déjà exécutées [9,22].

e. Archivage des messages

Il est nécessaire de garder un journal des messages échangés et des opérations exécutées. Ceci permet de garder les traces légales dans le cas des échanges B2B, et plus généralement de pouvoir tracer des activités et les réexécuter en cas de perte de message ou de panne.

f. Sécurisation des messages

Pour tous les processus interentreprises avec échanges critiques, il faut pouvoir authentifier les utilisateurs, crypter les données et signer les messages. Ces fonctionnalités sont donc à intégrer dans une architecture distribuée autour des services Web.

8.2 Exemple des systèmes de gestion des processus d'affaires : BPEL4WS

La première version du langage d'exécution des processus d'affaires (BPEL : Business Process Execution Language) a été créée 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 [24,25]. 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 (Business Process : processus métiers) dans les entreprises.

Le Langage BPEL est de format XML comme tous les autres langages et les spécifications reliées par les services Web. BPEL influx sur d'autres standards des services Web telles que WSDL et SOAP pour la description d'interface et de 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 d'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. [25].

Les motivations derrière l'orchestration des services Web avec BPEL sont [23] :

- l'augmentation de la productivité.

- La réduction des dépenses.
- L'amélioration des niveaux de services par l'automatisation des processus en utilisant des technologies standardisées.

Les éléments d'un processus BPEL

La définition d'un processus métier en langage BPEL est effectuée à l'aide des environnements graphique [23]:

La première étape est la définition des partenaires impliqués dans le service. Les partenaires qui sont les services impliqués dans l'exécution du service Web ou le processus, peuvent être des clients ou fournisseurs de services au processus. Ils sont définis par des liens des partenaires, des rôles et les types de port de WSDL (portType).

La deuxième étape consiste à la définition des variables utilisées par le processus, qui sont des instances des messages WSDL, des types simples de schéma XML ou des éléments de schéma XML. Les variables sont utilisées comme des containers d'entrée ou de sortie pour des invocations de services et pour la sauvegarde de l'état d'un processus.

L'étape suivante est pour la définition de gestionnaire des fautes, c'est à dire la définition des activités à exécuter dans le cas de terminaison anormale d'un processus. Cette définition suit la même syntaxe que la définition d'un processus normal.

Enfin, le processus définit les activités qui composent l'exécution d'un processus dans des circonstances normales. BPEL définit un certain nombre d'activités pour décrire l'interaction entre le processus et leurs partenaires.

Il faut noter que BPEL fournit des mécanismes pour contrôler le flux des activités. Les processus de BPEL peuvent être exécuter séquentiellement, en parallèle ou en réponse à un message reçu à partir d'un partenaire.

9 Avantages &Inconvénients

Malgré toutes les ambitions apportées par les services Web. Ces derniers ont aussi leurs avantages et inconvénients. Parmi ces avantages nous citons [17]:

- Les services Web consistent en un semble assez simple de propositions.
- Ils considèrent le Web comme environnement de développement.
- Les services Web sont basés sur les standards de W3C.
- Ils sont très adaptés aux problèmes de communication entre applications Web.

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

- IL reste d'autres problèmes qui ne sont pas réglés par les 3 outils (SOAP, WSDL et UDDI), comme la sémantique de dialogue entre les applications.
- Les services Web s'endurent des problèmes de performances [17].
- La sémantique « Requête / Réponse » utilisée par les services web est inefficace.

10 Conclusion

Dans ce chapitre, nous avons effectué une étude sur les services Web. Ces derniers sont considérés comme un nouveau paradigme de développement des applications distribuées et influent directement sur plusieurs domaines, et surtout sur les domaines de génie logiciels industriel, et par conséquence sur le monde industriel en entier.

Parmi les avantages apportés par les services Web nous trouvons l'aspect de la réutilisation qui permet d'exposer des parties des systèmes patrimoniaux, contenant une logique d'affaires intéressante, autant que des services Web, dont le but est de permettre aux autres firmes de les réutiliser dans le développement des nouveau systèmes basés sur l'architecture orienté services.

Dans le chapitre suivant, nous étudierons les systèmes patrimoniaux et les différents travaux essayant de faire migrer ces systèmes vers la technologie des services Web.

Chapitre 3 :

Les systèmes patrimoniaux

1 Introduction

Dans l'économie actuelle, il est rare qu'une entreprise réussisse si elle reste basée seulement sur son patrimonial applicatif sans accompagner l'évolution technologique. 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.

Ces besoins forcent les entreprises à penser au remplacement de leurs systèmes informatiques existants par d'autres plus avancés et être capables d'améliorer leurs activités et le rendement et de les diffuser dans le monde par l'intégration au Web. Néanmoins, il existe beaucoup d'obstacles devant ces ambitions, comme l'énorme investissement qui a été perdu pour le développement et la maintenance de ces systèmes. Le problème le plus important est que ces systèmes implémentent bien la logique d'affaires des entreprises et contiennent des fonctionnalités restées toujours utiles. Les propriétaires n'ont pas une confiance totale en les nouveaux systèmes.

Les responsables des entreprises, n'ont trouvé que deux solutions possibles 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 d'autres systèmes d'information plus avancés et plus confiants.

Ce chapitre a pour objectif d'étudier les systèmes patrimoniaux et leurs approches de la modernisation, et de la migration vers le SOA. Pour cela nous allons démarrer cette étude par une collection de définitions permettant l'explication du terme système patrimonial. En suite, nous exposons les différentes phases de cycle d'évolution d'un système patrimonial, qui sont la maintenance, la modernisation et enfin la dernière phase qui est le remplacement. Nous exposons dans la section qui suivra les besoins qui nous imposent de suivre la modernisation et nous entamons aussi les défis qui gênent les agents de la maintenance; après nous énonçons les trois approches de la modernisation. La modernisation de type « boîte noire » qui voit le système à moderniser comme une boite noire et le travail sera concentré sur les interfaces exposées par ces systèmes. L'autre approche est de type « boîte blanche » qui considère le système comme une boite blanche. Cette deuxième approche nécessite une étude approfondie pour comprendre le système à moderniser et ensuite de l'implémenter à nouveau

en se basant sur les nouvelles technologies. Enfin, la dernière approche qu'est de type « boîte grise » qui est un mélange des deux approches précédentes, parce qu'il implique une étude partielle du système. Enfin, nous arrivons à la section de la migration des systèmes patrimoniaux vers les services Web, où nous présentons les approches existantes, qui sont essentiellement inspiré des approches vus à dans la section précédente. Nous illustrons par des exemples les deux types d'approches de migrations, boîte noire et Boîte grise.

2 Définitions des systèmes patrimoniaux

Puisque l'origine de l'expression « systèmes patrimoniaux » est l'expression anglaise « Legacy systems », nous exposons d'abord une définition pour le terme « Legacy » extrait du dictionnaire anglais Oxford (Oxford English Dictionary) [34]:

« Legacy : Une somme d'argent, ou un quelque chose de spécifique, donné à l'autre par sa volonté; quelque chose remis vers le bas ou reçu d'un ancêtre ou d'un prédécesseur. »

Les systèmes patrimoniaux sont des pièces du logiciel hérité qui 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 des raisons spécifiques, tel que le coût énorme de remplacement et de reconception. .

Une autre définition propose que [31] :« Les systèmes patrimoniaux sont des systèmes d'information ou logiciels, développés à l'époque avec des moyens et des techniques traditionnels par rapport à ceux de nos jours et pour effectuer des tâches nécessaires aux activités des entreprises de ces périodes. »

3 Evolution des systèmes patrimoniaux

L'évolution du système d'information est un terme à sens large qui recouvre le continuum qui est commencé par le premier ajout d'un champ à la base de données jusqu'à la réimplémentation du système d'information en entier [28,29]. Le cycle d'évolution d'un système patrimonial est composé de trois phases à savoir : ***la maintenance, la modernisation et le remplacement.***

Le cycle de vie d'un système patrimonial est résumé par la [Figure 3.1], où la ligne pointillée représente les besoins croissants du business tandis que la ligne solide représente la fonctionnalité fournie par le système patrimonial. Les phases répétées de maintenance du système patrimonial supportent les besoins du business pendant une période donnée, mais quand ce système commence à être périmé, la maintenance ne peut plus satisfaire les besoins du business. Alors, un effort de modernisation plus grand, en temps et en fonctionnalités que

l'activité de maintenance, est très exigé. Et enfin, quand le système patrimonial ne peut évoluer, il doit être remplacé.

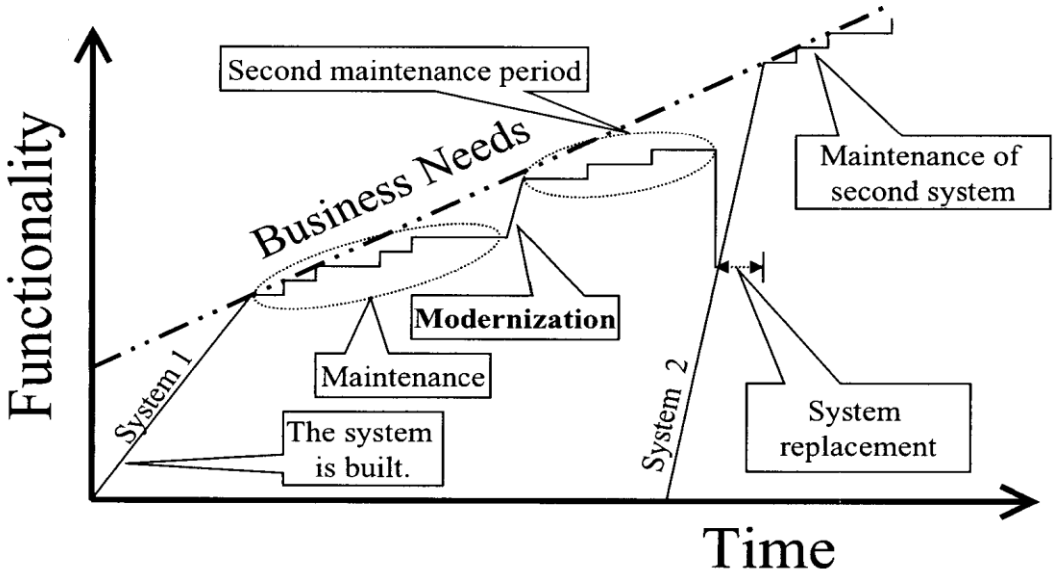


Figure 3.1. Evolution d'un système patrimonial [29]

3.1 Maintenance

La maintenance est un processus incrémental et itératif qui permet d'effectuer des petits changements sur le système patrimonial. Ces changements permettent souvent de régler des bugs ou de petites anomalies fonctionnelles qui n'exigent pas des changements structurels. La maintenance est exigée pour supporter l'évolution de n'importe quel système mais avec des limitations. Ces limitations incluent les suivantes [28]:

- L'avantage concurrentiel d'adopter des technologies récentes est sérieusement contraint. Par exemple des améliorations, tels que l'implémentation d'une architecture distribuée ou une interface utilisateur graphique, ne sont pas considérées comme des activités de maintenance.
- Les coûts des phases successives de maintenance des systèmes patrimoniaux peuvent augmenter avec le temps.
- La modification d'un système patrimonial pour l'adapter aux nouveaux besoins du business devient de plus en plus difficile, puisqu'il sera difficile de trouver des experts des technologies anciennes, et si on les trouve ils seront très coûteux.

3.2 Modernisation

La modernisation ou la migration implique des changements plus étendus que la maintenance, mais une partie significative du système existant est conservée. Ces changements incluent souvent la restructuration du système, l'amélioration des fonctionnalités ou la modification des attributs du logiciel [29]. La modernisation est utilisée quand un système exige des changements plus pénétrant que ceux possible pendant la maintenance.

3.2.1 Besoins de modernisation

Les raisons pour laquelle les entreprises se trouvent dans le besoin de moderniser leurs applications, sont résumées comme suit :

- Les systèmes patrimoniaux ne sont pas capables de répondre à des objectifs du business, qui sont dus à des facteurs tel que l'absence d'une fonctionnalité donnée, le manque du support ou leurs cycles de développement est long.
- Les systèmes patrimoniaux ne peuvent pas être intégrés avec d'autres systèmes internes même avec ceux du fournisseur ou du client.
- Le système patrimonial n'apte plus d'être maintenu...etc.

3.2.2 Bénéfices de la modernisation

La modernisation offre plusieurs bénéfices aux entreprises et à leurs services, parmi ces bénéfices nous citons :

- L'adaptation aux nouveaux besoins du business;
- L'amélioration des services pour les clients;
- Une intégration plus étroite avec les partenaires et les fournisseurs.
- La réduction du coût d'usage des systèmes d'information;
- L'amélioration de la qualité des données;
- L'amélioration du contrôle et de la gestion de sécurité.

3.2.3 Difficultés de la modernisation

Les efforts de modernisation de systèmes patrimoniaux ont échoué dans la plupart du temps à cause de plusieurs facteurs, nous citons les suivants :

a. La complexité des systèmes patrimoniaux

La complexité est considérée comme le plus grand limiteur du processus de modernisation des systèmes patrimoniaux; elle est produite à cause de la taille énorme, de l'incompréhensibilité des systèmes à moderniser et les phases successives de maintenance.

b. La technologie du logiciel et le processus d'ingénierie

Le génie logiciel et la technologie du logiciel sont séparés par un grand gouffre. Les technologistes considèrent que les ingénieurs du logiciel n'ont aucune relation avec le processus de modernisation. De l'autre côté, les ingénieurs du logiciel, ne reconnaissent pas que même les processus les plus avancés ne garantissent pas le succès si le produit n'est pas concurrentiel. Pour réussir à construire un système d'information pour les entreprise, il est nécessaire de comprendre la théorie derrière le génie logiciel, qui est « bits et bytes » de la technologie du logiciel, et les besoins des leurs affaires. La communication entre ces deux groupes brisés est dispersée et inutile, et les succès sont souvent le résulta des initiatives individuels.

c. Les risques de modernisation

Malheureusement, beaucoup d'entreprises sont incapables ou ne veulent pas contrôler le risque correctement. Ceci peut également provenir de l'insuffisance de compréhension de gestion des risques et des techniques de réduction du risque, tel que la planification d'urgence. Les risques de modernisation ne sont pas toujours majeurs, il est essentiel d'accepter un certain risque si nous devons accomplir une tâche significative. Cependant, beaucoup d'avantages peuvent être obtenus en identifiant simplement des risques, tôt dans le projet, en maintenant et en surveillant ces risques, et en se posant la question, « comment cette activité aide à atténuer le risque que j'ai détecté dans cet effort de modernisation ? »

d. Les composants logiciels utilisés dans la modernisation

Des efforts de développement du logiciel sont souvent confrontés avec le problème du quand adopter et du quand ignorer les nouvelles versions des composants logiciels, qui doivent être utilisés dans le développement. Généralement, les composants sont adoptés quand ils fixent un problème existant, moins souvent quand ils fournissent de nouvelles possibilités et si ces nouvelles possibilités résolvent un problème existant déjà. Si le système fonctionne correctement dans la phase du test, Il sera peu susceptible d'accepter l'introduction de nouvelles versions des produits des logiciels commerciaux, parce que le risque de présenter des instabilités dans le système est trop grand.

3.3 Remplacement

La dernière phase du cycle d'évolution d'un système patrimonial est le remplacement. Ce dernier nécessite la reconstruction des nouveaux systèmes à partir de zéro par des moyens et des techniques avancées. Le remplacement sera une nécessité lorsque le système patrimonial ne peut plus suivre les besoins du business de l'entreprise et quand aussi la modernisation n'est pas possible ou même n'est pas rentable.

Il existe deux propositions concernant la phase du remplacement, la première propose de remplacer le système patrimonial dans un seul coup en utilisant l'approche « grand coup ». L'autre approche propose de faire un remplacement incrémental. Celui-ci est utile si le système patrimonial a un degré de cohésion et de modularité. Pour préparer le système patrimonial à la phase de remplacement incrémentale, une phase de reingénierie préalable est effectuée.

Avant que les entreprises ne choisissent la technique à utiliser pour remplacer leurs systèmes patrimoniaux, il faut d'abord évaluer les risques suivants :

- Le personnel spécialisé des systèmes d'information qui accomplissent les tâches de maintenance peut ne pas être familiarisé avec les nouvelles technologies.
- Le remplacement exige un test étendu du nouveau système, puisque les systèmes patrimoniaux sont habituellement bien examinés, accordés et encapsulent une logique d'affaires considérable.
- Il n'y a aucune garantie que le nouveau système sera aussi robuste ou fonctionnel que le précédent, ce qui peut causer une période de dégradation des fonctionnalités du système.

4 Différentes Approches de modernisation

La modernisation du système peut être distinguée par le niveau de la compréhension de système exigé pour accomplir la modernisation. Il existe trois approches proposées pour la modernisation, qui sont la modernisation de type boîte noire, la modernisation de type boîte blanche et la modernisation de type boîte grise. Le choix entre ces approches est dépend des critères décisifs, tels que :

- La nature des systèmes informatiques existants.
- Les objectifs attendu de la modernisation.
- Les capacités des groupes de travail ...etc.

4.1 Modernisation de type boîte noire

Dans ce type de modernisation le système patrimonial est vu comme une ***boîte noire*** (en anglais : Black Box). Elle vise à analyser seulement les entrées et les sorties d'un système patrimonial dans un contexte de fonctionnement pour comprendre les interfaces du système. La modernisation boîte noire est souvent basée sur l'adaptation (en anglais : wrapping) [26].

L'adaptation consiste à entourer le système patrimonial par une couche logicielle qui cache sa complexité et exporte une interface moderne. L'adaptateur (en anglais : Wrapper) est employé pour retirer les désaccords entre l'interface exportée par le logiciel et les interfaces exigées par le système patrimonial pour accomplir l'intégration. Malheureusement, cette solution n'est pas toujours pratique, et exige souvent une compréhension des modules internes du logiciel en utilisant des techniques de modernisation « boîte noire» [29]..

4.2 Modernisation de type boîte blanche

La modernisation boîte blanche (en anglais : White Box) est commencée par un processus d'ingénierie inverse (en anglais : reverse engineering) pour comprendre le fonctionnement interne du système et identifier les composants du système et de leurs relations, et produire une représentation du système à un niveau plus élevé d'abstraction.

La compréhension du système inclus la modélisation du domaine, l'extraction des informations à partir du code en utilisant des mécanismes d'extraction et ensuite la création des abstractions qui va aider à la compréhension de la structure interne du système. L'analyse et la compréhension de l'ancien code sont des tâches très difficiles puisque avec le temps, le système patrimonial s'effondre sous sa complexité. Bien que quelques avancements aient été faites dans la compréhension du programme, elle reste toujours une tâche risquée et intensive.

Après que le code soit analysé et compris, la modernisation boîte blanche inclut souvent la restructuration du système ou du code. La restructuration peut être défini comme « La transformation d'une forme de représentation à une autre au même niveau d'abstraction, tout en préservant le comportement externe du système patrimonial (concernant les fonctionnalités et le sémantique) ».

Cette transformation est typiquement employée pour augmenter la qualité du système comme la maintenance ou l'exécution. La technique « Program Slicing » est une technique particulièrement populaire pour la restructuration du logiciel [26,29].

4.3 Modernisation de type boîte grise

Le troisième type d'approches de modernisation des systèmes patrimoniaux est une combinaison des deux approches précédentes, l'adaptation et la compréhension du système entier. Cette approche est appelée l'approche « boîte grise » (en anglais : Grey Box). Elle est basée sur l'analyse des besoins et la reingénierie du système. Il faut noter que cette approche apparaît comme une approche rapide parce qu'elle s'intéresse à des parties du système qui disposent de 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 du support de la technologie de la programmation. Avec le paradigme des services Web ainsi que SOA rendent possible la réutilisation de certaines fonctionnalités comme des services distribués et indépendants. L'ensemble des services produits contient la plupart de la logique d'affaires du système patrimonial [32,35,38].

5 Migration des systèmes patrimoniaux vers les services Web

Dans cette section, nous proposons quelques travaux proposés pour l'émigration des systèmes patrimoniaux vers une architecture orientée services (SOA). Ces travaux sont classés selon les types approches de migration, celle de type boîte noire et l'autre de type boîte grise, comme suit :

5.1 Migration boîte noire

Cette approche propose d'intégrer les systèmes patrimoniaux via des adaptateurs (en anglais : 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.

Selon [26], Il y a deux modèles de base pour l'intégration des systèmes patrimoniaux comme des services Web et qui sont désignés sous : modèle d'interface et modèle de passerelle (en anglais : Gateway).

- Les adaptateurs s'exécutent sur un ordinateur central (Mainframe) et peuvent utiliser des interfaces qui permettent l'intégration parfaite avec l'application cible.
- Les Passerelles (Gateways) s'exécutent indépendamment de l'ordinateur central (Mainframe), mais dans des serveurs intermédiaires d'une architecture de type trois

tiers, et emploient souvent des méthodes traditionnelles, telles que l'approche « screen scraping ».

Donc, le choix entre ces approches est déterminé selon les réponses aux questions suivantes :

- Où les services web existent-il ?
- Comment fonctionnent-ils ?
- Quels sont les types d'applications à intégrer ?

5.1.1 Méthode basée sur l'interface

Les adaptateurs permettent de transformer les systèmes patrimoniaux en services Web, et n'exigent pas l'utilisation de matériel additionnel [26], et même aucune modification ne sera effectuée sur le système à émigrer.

Les étapes suivantes décrivent comment découvrir et utiliser un système patrimonial comme un services Web, en utilisant la méthode de l'interface [**Figure2**] :

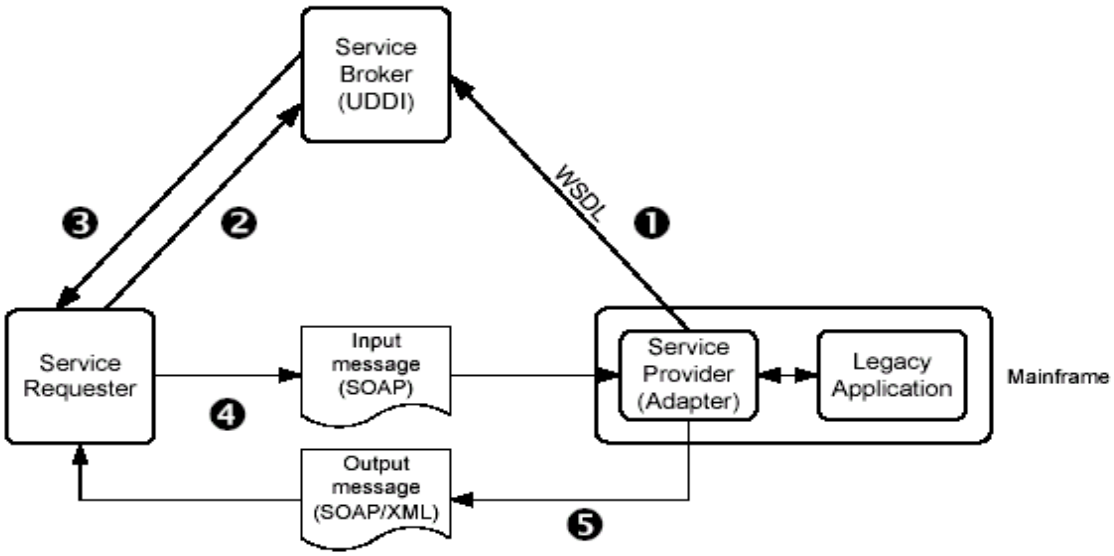


Figure 3.2. Méthode basée sur l'interface [26]

1. Le fournisseur crée des spécifications WSDL pour publier son service Web sur le Broker du service (Le courtier).
2. Le client lance une requête de recherche pour un service Web vers le Broker.
3. Le Broker selon la requête reçu et le nom et la catégorie du service Web demandé, choisit un fournisseur et renvois ses coordonnés vers le client.
4. Le client utilise les informations reçues à partir du Broker pour créer un message SOAP et l'émettre vers l'adaptateur sur le mainframe.

5. Après l'exécution de la requête reçue par le système patrimonial, le fournisseur du service rassemble les résultats dans un message SOAP/XML et le renvoie au client.

5.1.2 Méthode basée sur le Gateway

Les passerelles (ou Gateways) fonctionnent sur le tiers intermédiaire physique ou logique, de l'architecture trois tiers. L'emplacement où les Gateway fonctionnent est très important puisqu'il y a quelques options pour accéder au serveur à partir du tiers intermédiaire, ce qui permet aux passerelles d'exiger une certaine forme de la méthode ***screen scraping***. L'intégration entre le Gateway et une application spécifique est fortement couplée.

Les étapes suivantes résument le processus qui utilise un système patrimonial comme service Web en utilisant l'approche des passerelles [**Figure 3**] :

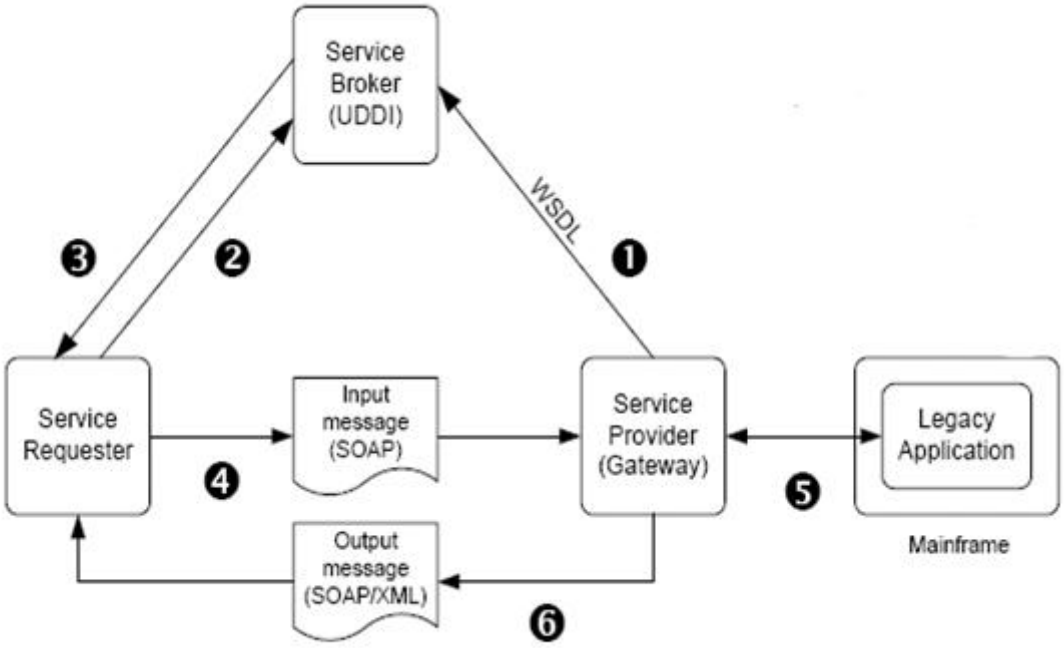


Figure 3.3. Méthode basée sur le Gateway [26].

1. Le fournisseur crée des spécifications WSDL pour publier son service Web sur le Broker (Le courtier).
2. Le client lance une requête de recherche d'un service Web vers le Broker.
3. Le Broker selon la requête, le nom et la catégorie du service Web, choisit un fournisseur et renvois ses informations (les coordonnés du fournisseur) vers le client.
4. Le client utilise les informations reçues à partir du Broker pour formuler un message SOAP et l'émettre vers l'adaptateur.

5. Le fournisseur commence une session d'émulation et à appliquer une série de transactions sur le système patrimonial pour rassembler les données demandées.
6. Le fournisseur convertit ces données en une forme appropriée et renvoie la réponse au client.

5.2 Migration boîte grise

Cette approche, dite « ***boîte grise*** », est le sujet de notre travail dont le but est d'intégrer les parties du système patrimonial possédant une logique métier importante. Cette approche est économique et plus pratique par rapport à d'autres approches, parce que l'extraction de ***quelques*** fonctionnalités intéressantes du système patrimonial et leur réutilisation comme des services Web se font d'une manière aisée.

Pour illustrer ce type de migration nous avons choisi les trois approches suivantes, qui sont apparu réalisables:

5.2.1 Approche basée CORBA

Y. Zou et K. [35] Kontogiannis présente un Framework de trois étapes pour la migration d'un système patrimonial vers le Web, comme suit :

a. Identification des parties réutilisables

L'objectif de cette étape est l'extraction des parties valables du système pour être réutilisées en tant que services Web, ces parties sont appelées des composants. Pour identifier ces composants, cette approche propose de transformer le code procédural d'un composant patrimonial *(Legacy component)* choisi en une architecture orientée objet où les interfaces du composant sont bien définies et les opérations relatives qui fournissent une fonctionnalité spécifique sont entièrement encapsulées en classes.

Le processus de transformation d'un système procédural à une architecture orientée objet est commencé par le choix des classes d'objets possibles. Ensuite, il utilise les techniques du Clustering et les techniques de détection de l'architecture pour découvrir la décomposition initiale des gros systèmes.

b. L'intégration des nouveaux composants dans un environnement centré réseau

Après que les composants soient extraits, leurs interfaces seront extraites aussi et représentées en XML. Dans cette phase, un outil permettant la génération des adaptateurs des objets et la description CORBA IDL (CORBA Interface Description Language) à partir de la spécification XML des interfaces, est utilisé. Cette phase est subdivisée en deux étapes, comme suit :

- **La représentation des services identifiés**

Pour créer un nouveau service, il faut d'abord déterminer son interface. Cette dernière doit être représentée indépendamment d'un langage de programmation, pour cela elle est spécifiée par le langage OMG IDL (Object Management Group Interface Description Language) qui est adéquat à CORBA. Dans cette approche, les informations de l'interface des composants sont codées en XML.

- **La Génération Automatique du « OMG IDL » et d'adaptateur CORBA**

Le processus d'adaptation est accompli aux termes de trois étapes principales :

La première étape se concentre sur la génération automatique de l'interface de CORBA IDL pour les composants identifiés. Le générateur d'IDL lit le document XML contenant les informations de l'interface et convertit les descriptions en IDL, en utilisant les règles de transformation des spécifications d'OMG IDL. Il crée aussi un nouveau fichier contenant l'interface décrite en IDL.

La deuxième étape utilise le compilateur CORBA IDL pour traduire les spécifications d'IDL en langage spécifique (C++ par exemple). Les classes de subrogé de client et les classes squelettiques du côté serveur sont produites automatiquement à partir des spécifications IDL correspondantes. Les classes de subrogé client sont des délégation (Proxy) qui permettent à une requête d'être traitée par l'intermédiaire d'un appel local normal de méthode. Les classes squelettiques du côté serveur permettent d'envoyer une demande reçue par le serveur à l'objet approprié.

La troisième étape se concentre sur la génération automatique de Wrappers, c à d des adaptateurs, des objets pour être vus comme des objets CORBA.

c. Adaptation des objets CORBA par SOAP

Dans ce contexte, le protocole de SOAP est adapté en tant que messager uniforme entre les objets CORBA et le reste de services Web.

Le framework proposé est illustré dans la [Figure 3.4]. Le client envoie sa requête, conforme au SOAP, par HTTP vers un serveur Web. Celui-ci réoriente la requête au moteur de Servlet, qui est responsable du traitement des requêtes de HTTP/POST et de HTTP/GET. Le moteur d'exécution de SOAP est susceptible de manipuler l'interprétation et l'expédition des messages SOAP. Au noyau, un traducteur de SOAP/CORBA IDL est implémenté comme une collection des JavaBeans pour traiter les messages SOAP entrantes, qui représentent des invocations aux objets CORBA. Intérieurement, ce traducteur établit le lien entre le SOAP/RPC et CORBA IDL et enfin, génère le subrogé pour expédier les requêtes aux services principaux. En attendant, le WSDL du serveur CORBA est automatiquement produit, à partir des spécifications d'interface de composant, et enregistré dans l'UDDI.

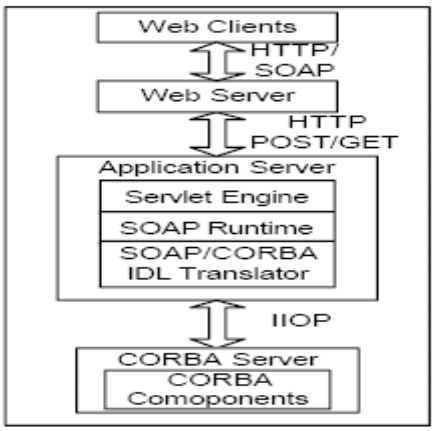


Figure 3.4. Approche basée CORBA [35]

5.2.2 Approche basée sur les techniques de Clustering

Z.Zhang et al ont proposé un ensemble de méthodes permettant l'extraction des services Web à partir des systèmes existants, mais ces méthodes sont légèrement différentes [32,36,37]. Pour cela, nous avons choisi une parmi celles [32], qui est composée des étapes suivantes :

a. Identification des services

Le processus d'extraction démarre par l'analyse du domaine de l'application du système, pour identifier et documenter les besoins sur l'ensemble des systèmes d'un même domaine. Les auteurs de cette approche ont proposé de subdiviser cette étape en deux étapes successives qui sont :

- Identification des sous domaines : qui a pour but de déterminer les limites de sous domaines, et aussi la décomposition du domaine du problème entier.

- Analyse du sous domaine choisi : qui a pour but d'analyser le domaine d'application.

Les résultats de l'analyse du domaine d'application sont résumés par un modèle récapitulatif, ce modèle peut être représenté en UML. En se basant sur ce modèle, quelques fonctions précieuses nécessitent d'être transformées en services Web. Mais, aussi de nouveaux services, qui n'existent pas dans le système patrimonial, sont envisagés. Ces services seront par la suite implémentés et intégrés dans le nouveau système orienté service.

b. Compréhension du système patrimonial

Pour comprendre le système, cette approche propose de faire une évaluation de l'état actuel du système, en spécifiant à quelle phase il est, dans son cycle de vie. Pour cela ils ont proposé d'utiliser l'arbre de décision. Un système qui convient à la migration vers une architecture orientée services (SOA), s'il satisfait les critères suivants :

- Certaines fonctionnalités réutilisables et fiables contiennent un logique d'affaires valable sont disponibles.
- Certains composants réutilisables sont assez maintenables comparés à la maintenance du système entier.
- Les fonctionnalités sont utiles pour être exposées en tant que services Web indépendants.

c. Analyse en utilisant l'algorithme de Clustering

La méthode de Clustering (regroupement) permet de regrouper un ensemble d'entités dans une base des données selon leurs relations et la similarité entre eux, le but de cette technique est d'extraire une structure existante des groupes (Clusters). Avant de lancer le processus de Clustering, il faut d'abord définir la similarité entre les entités et l'algorithme de Clustering qui doit être utilisé. Les résultats de cet algorithme, qui sont présentés sous forme d'un dendrogramme **[Figure 5]**, doivent être évaluées et expliquées correctement. Le dendrogramme présente une vue hiérarchique du système patrimonial.

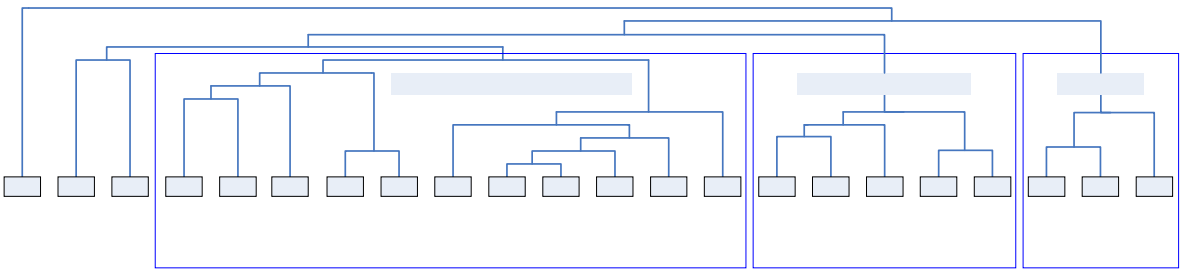


Figure 3.5. Exemple d'un dendrogramme [37]

Dans cette phase, les auteurs de l'approche utilisent une version améliorée de la technique de Clustering pour transformer le code procédural en model orienté objet, si le système patrimonial à traiter est de type procédural, et aussi pour la phase de compréhension du code du système patrimonial.

Les fonctions, les procédures et les classes des programmes orientés objet sont définis comme des entités à être groupées. Pour calculer la similarité entre les entités, cette approche a défini les caractéristiques selon l'occurrence des identificateurs, où les identificateurs peuvent être le nom de fonction, de procédure, une propriété d'un objet ou le nom d'une méthode d'un objet...etc.

L'algorithme de Clustering qui a été utilisé pour détecter l'architecture initiale d'un système patrimonial, est le suivant [32]:

1. E_i (1 <= i <= n) est une entité dans un système orienté objet. F_i représente l'ensemble des caractéristiques d'une entité E_i. $W(F_i)$ dénote le poids total de F_i, $L(k)$ est le niveau de $k^{ème}$ Clustering . Un 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 de similarité est $D_{n.n} = \{Sim [i , j]\}$.
3. Chercher la paire (a,b) du clusters les plus proches (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 au 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 regroupés dans un seul cluster, arrêter. Sinon, aller à l'étape3.

d. Wrapping et Intégration

Cette étape est nécessaire pour faire passer le service logique à un service fonctionnel. Cette étape inclut le raffinement du code, l'intégration des composants complémentaires et l'implémentation du « glue code ».

Alors pour le but de créer des services Web indépendant et faiblement coupler, il est nécessaire de raffiner le code extrait. Le raffinement élimine les interfaces d'interaction et

spécifie les détails d'interface pour la communication avec le service. Dans cette phase, le code inutile est détecté et supprimé.

Ensuite, les nouveaux services, qui sont détectés dans la phase d'identification des services, seront implémentés et intégrés dans le futur système orienté services. En outre, le « glue Code » est développé. Ce code intercepte les appels, entrants et sortants, et les remplace par une interaction appropriée. La réduction de la complexité du service est réalisée par des métriques du génie logiciel et des techniques d'analyse de dépendance.

Pour la mise en place des services Web, Il faut d'abord définir ses interfaces. Une définition des fonctionnalités des services est fournie en WSDL. Un Processeur SOAP est intégré. Enfin, l'enregistrement du service Web dans l'UDDI.

5.2.3 Approche basée sur les règles métier

Dans [38], 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 métier. La règle métier est définie comme un algorithme qui est utilisé pour calculer un résultat donné. Les étapes de cette approche sont les suivantes :

a. La récupération du code de service

La problématique essentielle posée dans cette première étape est comment détecter le code qui implémente la règle métier. L'auteur utilise l'approche citée dans [39] pour détecter la règle métier.

Selon cette approche, pour détecter le code qui implémente une règle métier, il faut d'abord déterminer le résultat produit par cette règle. Ceci est effectué par l'identification des variables qui sont retournées par les fonctions implémentant la règle métier et aussi par l'identification des fonctions. Le problème qui se pose ici est qu'un bloc de codes (fonction, subroutine, paragraphe,…etc.) peut traiter plusieurs règles métiers. Pour résoudre ce problème, cette approche propose de faire une analyse du flux de données 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 contenant 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. Ceci est effectué par la représentation de chaque règle métier par un diagramme des flux des données. Cette documentation est utile pour la décision si une règle existante, implémenté

dans un système patrimonial mérite d'être réutiliser comme un service public dans une architecture orientée service. Cette décision nécessite la compréhension totale du cette règle métier et aussi sa valeur économique. Les règles ayant une implémentation acceptable et ont des valeurs économiques élevées sont les candidats principaux pour la réutilisation.

b. Adaptation du Code

Le but de cette phase est de fournir une interface WSDL du service extrait du système patrimonial. Pour accomplir cette tâche, 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 génère aussi pour chaque service deux modules supplémentaires, un pour analyser le message entrant et extraire les données, les valeurs extraites sont alors assignées aux arguments correspondants dans le nouveau service, et l'autre pour créer le message de retour contenant les résultats produits par le service.

c. Liaison du service Web au Code

Dans cette étape, un nouveau composant appelé « Proxy » (procuration) dans la même adresse que la définition du processus, a été créé. Ce Proxy vérifie les paramètres et génère l'interface WSDL qui est expédiée par le message 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 le lui envoie. Après l'exécution du service, les résultats sont transformés par l'adaptateur en un document XML, qui va retourner vers le « Scheduler » pour le retransmettre au client.

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

6 Conclusion

Il très clair que les systèmes patrimoniaux contiennent une logique d'affaires très riche et confiante, ce qui oblige les propriétaires de les garder et de chercher des nouvelles solutions permettant la réutilisation de cette logique et en même temps de suivre les technologies les plus récentes. Parmi ces solutions nous trouvons les travaux concernant la migration des systèmes patrimoniaux vers les services Web.

Comme nous avons vu dans ce chapitre, il existe une concentration énorme sur les approches: l'approche boîte noire, l'approche boîte blanche et la dernière boîte grise, permettant la réutilisation des systèmes patrimoniaux comme des services Web. La différence entre ces approches réside dans le degré de compréhension du système. A partir de ce qui précède, il apparaît que l'approche boîte grise est la plus économique et la plus efficace puisqu'elle ne s'intéresse que par les parties du système patrimonial contenant une logique d'affaires importante pour les réutiliser dans l'architecture orientée service.

Le chapitre suivant, sera consacré à l'exposition de 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.

Chapitre 4 :

Approche proposée

1 Introduction

Comme nous avons signalé dans le chapitre précédent, plusieurs travaux ont été proposés, permettant la transformation des systèmes patrimoniaux en services Web, surtout celui de l'approche « boite grise». Selon notre point de vue, ces travaux présentent deux inconvénients majeurs. Le premier est lié à l'intervention décisive de la supervision humaine, dans les phases de ces approches surtout lors de détection du code source qui implémente un service Web. Cette intervention est souvent coûteuse et sensible aux erreurs. Le deuxième est le manque de standardisation de ces approches, car chaque langage de programmation nécessite d'effectuer des modifications sur ces approches pour être adaptables au langage désiré.

Pour pallier ces désavantages, nous proposons une framework générique en adoptant l'approche boîte grise. Ce framework offre un processus d'extraction de services Web à partir des systèmes patrimoniaux de différents langages de programmation et où l'intervention de l'expert humain est limitée, où nous considérons que chaque règle métier implémente un service Web, et nous utilisons le langage WSL (Wide Spectrum Language) pour unifier les langages des programmation et étendre la gamme des applications à émigrer.

Avant d'exposer en détails les différentes phases qui composent notre approche, nous avons choisi de commencer ce chapitre par l'introduction des différents concepts clés de notre framework qui sont la règle métier, qui dans notre contexte implémente un service Web, et le langage WSL qui nous avons utilisé pour unifier les langages de programmation des systèmes à transformer. Une vue globale de notre framework sera proposé dans la section qui suit. Ensuite, nous détaillons chaque phase du ce framework à part, en présentant le rôle de chacune dans le processus d'émigration. Une discussion est proposée ensuite, pour prouver notre choix en comparant notre framework avec les deux autres que nous avons vues dans le chapitre précédent. Ensuite, nous manifestons aussi les différents avantages et inconvénients liés à notre travail. Nous terminons ce chapitre par une conclusion.

2 Notions générales

Avant d'entamer l'explication de notre solution du problème d'extraction des services Web à partir des systèmes patrimoniaux, nous exposons d'abord les notions des termes clés utilisés dans cette approche qui sont, la règle métier et le langage WSL.

2.1 Règle métier

Le concept de « règle métier » (Business Rule) a été introduit pour la première fois vers la fin de l'année 1980, à l'occasion du projet AD-Cycle de IBM [42]. A partir du cette date plusieurs définitions de ce concept ont été proposées, et nous avons choisi quelques unes :

« Les règles métiers sont des règles opérationnelles, les organisations les suivent pour réaliser de différentes activités » [43,44].

« Une règle métier est un rapport qui définit ou contraint quelques aspects des affaires » [15].

« Les programmes utilisent les règles pour dicter ou contraindre des décisions ou des actions spécifiques. Ces règles ont été typiquement testées, révisées, et sont mises à jour. Elles représentent des business précieuses et substantielles ou des avantages intellectuels » [45].

La règle métier au niveau sémantique est définie en tant que quelque chose qui relie entre des données élémentaires et les instructions des programmes. Une règle métier est une fonction avec un ensemble d'arguments et d'un ou plusieurs résultats comme exprimé en équation : X = f (YI,YZ,Y3) [42].

Enfin, il existe plusieurs types de règle métier, nous citons comme exemple :

- Une règle métier décisionnelle (Booléenne) a pour résultat final la réponse OUI ou NON; c'est une règle qui calcule la valeur d'une variable et affiche une réponse Oui ou NON selon la valeur de cette variable, elle est illustrée dans l'exemple qui suit :

```
TEST: = INPUT1 * 2 + INPUT2 * 3 + 6;
IF (TEST >= 20) PRINTF ("RÉPONSE: OUI");
ELSE  PRINTF ("REPONSE: NON") ;
```

- Un autre type des règles est celles de calcul, c'est à dire une règle qui reçoit des données et après un ensemble d'opérations mathématiques, elle retourne des résultats numériques pour être affichés sur l'écran ou même pour être réutilisés par d'autres règles. L'exemple suivant présente une règle métier qui calcule le PrixTTC d'un article donnée :

```
PRIXTTC=PRIXHT +TVA*PRIXHT
```

2.2 Extraction des règles métiers

Ces dernières années ont vu une concentration croissante sur le développement des techniques permettant l'extraction des règles métier (Business Rule) qui était inhumées dans des systèmes patrimoniaux.

Il existe pas mal de travaux dans le domaine d'extraction des règles métiers à partir d'un système patrimonial. Nous prenons par exemple [44], qui propose une approche pour résoudre le problème d'extraction des règles métiers, en combinant la classification des variables utilisées dans le système patrimonial, la technique de "Program Slicing", et l'abstraction hiérarchique avec d'autres techniques de maintenance.

Une approche de Frederik. V et al [45] propose la transformation des applications de différents langages de programmation en WSL et l'utilisation des règles mathématiques formelles pour la récupération des règles métiers à partir des applications transformées.

Xinyu. W et al proposent dans [43] un framework pour l'extraction des règles métiers à partir des systèmes d'information volumineux. Ce framework se compose de cinq étapes : la technique de Program Slicing, identification des variables de domaine, l'analyse de données, présentation des règles métiers et la validation d'affaires.

Enfin, H.Sneed el al proposent une approche pour extraire le code source qui implémente la règle métier [42]. Ils considèrent que les résultats produits par les règles métiers sont la clé de détection de ces règles. Ils ont proposé d'identifier d'abord les variables retournées par les fonctions qui les implémentent. L'étape suivante consiste à faire une analyse du flux des données en démarrant par les instructions qui génèrent les résultats finaux et en revenant vers toutes les instructions qui interviennent dans la production des résultats finaux.

2.3 WSL (Wide Spectrum Language)

WSL (Wide Spectrum Laguage) langage de gamme étendu, est un langage qui contient des spécifications abstraites de haut niveau et des concepts de programmation de bas niveau dans un seul langage. Il a été développé en parallèle avec le développement des théories de transformation des programmes [50]. Il contient les primitifs et les instructions, les définitions de transformation, et les règles d'abstraction [30]. En travaillant dans un langage formel simple, nous pouvons montrer qu'un programme est implémenté correctement, ou ses spécifications capturent nettement son comportement, au moyen de transformations formelles

dans le langage. Nous ne devons pas développer des transformations entre langages de programmation et de spécification [51].

La plupart des concepts dans WSL, comme par exemple l'instruction *if*, la boucle *While*, les procédures et les fonctions, sont communes entre plusieurs langages de programmation. Il y a aussi quelques concepts qui sont reliés au niveau de spécification abstraite dans le langage.

Les expressions et les conditions sont représentées en WSL sous la forme de logique de premier ordre. Alors, la logique de premier ordre *infini* est utilisée pour calculer les disjonctions et des conjonctions infiniment. Ceci signifie que les opérateurs de quantification existentielle et universelle (∃,∀) sont inclus dans les instructions du WSL pour définir les ensembles infinis et les opérations similaires [42].

2.3.1 Les Composants du langage WSL

Selon [47] WSL peut être composée de 04 secteurs [Figure 4.1], dont l'objectif est d'étendre son utilisation dans le processus de reingénierie des systèmes patrimoniaux. Ces secteurs sont : ITL (Interval Temporal Logic) Logique de l'intervalle Temporelle, TGCL (Timed Guarded Command Language) Langage de Commande à Synchronisation Gardée, Ob-TAM (Object Oriented Temporal Agent Model) Model d'agent temporel orienté objet et Object-Action Model (model Objet–Action). La sémantique des 03 derniers secteurs sera basée sur ITL, les règles de transformation et les règles d'abstraction seront prouvées dans le cadre de ITL.

a. TGCL (Timed Guarded Command Language)

Si nous voulons faire la reinginierie d'un système patrimonial procédural, la première étape à faire sera la traduction du code source en TGCL en utilisant un traducteur universel. TGCL est la couche la plus abstraite dans le WSL. C'est une extension du langage de commande gardé de Dijkstra avec les caractéristiques de temps réel, tel que le temps, la simultanéité et les interruptions. Le rôle principal de TGCL est la « standardisation » puisque des systèmes patrimoniaux peuvent être programmés dans divers langages de programmation, tels que C, Pascal, Modula, etc. Avec TGCL nous pouvons obtenir une image concise des systèmes patrimoniaux. Cette image permet ensuite d'extraire des objets à partir du code TGCL. Si le système patrimonial est déjà écrit dans un langage orienté objet, alors la traduction ignore l'étape de TGCL [47].

b. ObTAM (Object Oriented Temporal Agent Model)

Le deuxième secteur du WSL s'appelle (ObTAM) le modèle temporel d'agent orienté objet. C'est une extension d'un autre modèle temporel d'agent proposé par H. Zedan avec les caractéristiques orientées objet. Le système qui était traduit en TGCL sera transformé sous forme d'ObTAM en appliquant les règles d'extraction d'objet. Quelques Systèmes patrimoniaux qui sont écrits en langage orienté objet tel que C++ peuvent être traduits directement en ObTAM. La raison pour laquelle le modèle ObTAM est conçu est que la technologie orientée objet a plusieurs avantages dans l'analyse et la conception de logiciel, telle que la réutilisabilité, la maintenance, la compréhensibilité, etc. Une autre raison d'adopter ObTAM est qu'il sera utilisé comme base d'abstraction de formes plus élevées de WSL, c'est à dire, le modèle objet-action et les formes de spécification ITL [47].

c. Le modèle Objet -- Action

Le troisième secteur du WSL est le modèle objet-action (Model Objet-Action). Selon ce modèle, le système à reimplémenter (reingénierie) est vu comme une collection d'objets qui ont leurs propres services appelés les actions [47].

d. Les spécifications ITL (Interval Temporal Logic)

Le dernier secteur du WSL est appelé les spécifications d'ITL. A la différence de la plupart des logiques temporelles, ITL peut manipuler la composition séquentielle et parallèle et offre des techniques, puissantes et extensibles, de spécifications et de preuves pour le raisonnement. Les contraintes de synchronisation sont exprimables et les constructions de programmation les plus impératives peuvent être regardées comme des formules dans une version légèrement modifiée d'ITL [47,48].

La description d'un système cible sous la forme de spécifications d'ITL est l'une des spécifications purement basées sur le formalisme ITL. Le modèle Objet-Action et les spécifications d'ITL sont assez abstraits pour que les programmeurs créent une nouvelle conception et spécification du système cible indépendamment du code source. Ainsi à partir de ces deux secteurs, quelques améliorations seront appliquées pour rendre le système patrimonial convenable aux nouvelles conditions. Après ces améliorations, le processus d'ingénierie du système peut être effectuée.

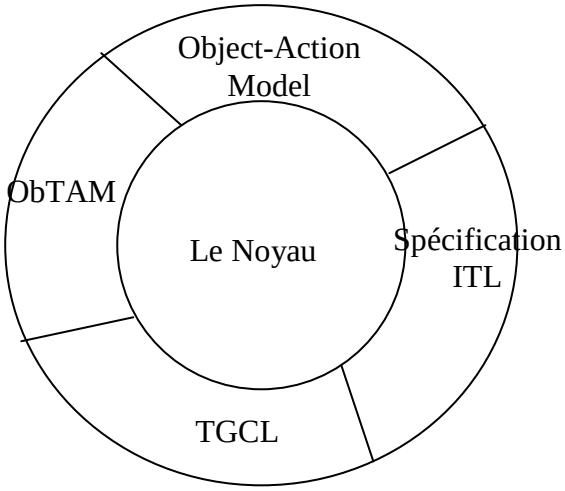


Figure 4.1. Architecture de 4 secteurs pour WSL [48].

2.3.2 Les avantages du WSL

Après ce qui précède, nous extrayons quelques avantages liés à l'utilisation du langage WSL dans le processus de reingénierie des systèmes patrimoniaux :

- Les différents programmes, écrits dans différents langages de programmation et avec différents styles peuvent être spécifiés de manière consistante en WSL.
- Permettre l'implémentation correcte d'une spécification par un programme de haut niveau.
- Les spécifications écrites en WSL capturent correctement le comportement d'un programme.
- Permettre à des parties de programme, à être comprises, d'être exprimées par différents niveaux d'abstraction.
- La syntaxe et les sémantiques de WSL sont strictement définies [30].
- L'architecture d'un système écrit en langage WSL peut être extraite facilement comme dans [40].

3 Approche proposée

L'approche que nous proposons recouvre toutes les étapes nécessaires à l'extraction des services Web à partir des systèmes patrimoniaux [Figure 4.2]. Pour tirer avantage des deux approches, Sneed [38] et Z.Zhang [32,36,37], et pallier leurs insuffisances nous avons procédé à leur combinaison en intégrant la transformation du code source original en WSL pour l'extraction directe du code qui implémente la règle métier.

L'approche commence par une étude de faisabilité, après laquelle nous déciderons si le système contient des fonctionnalités qui méritent d'être réutilisées comme des services Web. Dans le cas favorable, nous entamons la phase de l'analyse pour définir les services Web logiques. De ces derniers, nous distinguons les services qui doivent être extraits du système patrimonial et les autres qui n'existent pas dans le système patrimonial et qui doivent être implémentés à nouveau. En parallèle, une transformation du code source du système patrimonial en WSL aura lieu. Ensuite, en se basant sur les résultats des deux dernières étapes, le processus de détection du code source qui implémente une règle métier donnée est déclenché. Après la détection du code de toutes les règles métier existantes et méritant d'être publiées en tant que services Web, vient l'étape d'adaptation et de publication qui produit à la fin un ensemble de services Web. Les services Web restant seront implémentés à nouveau.

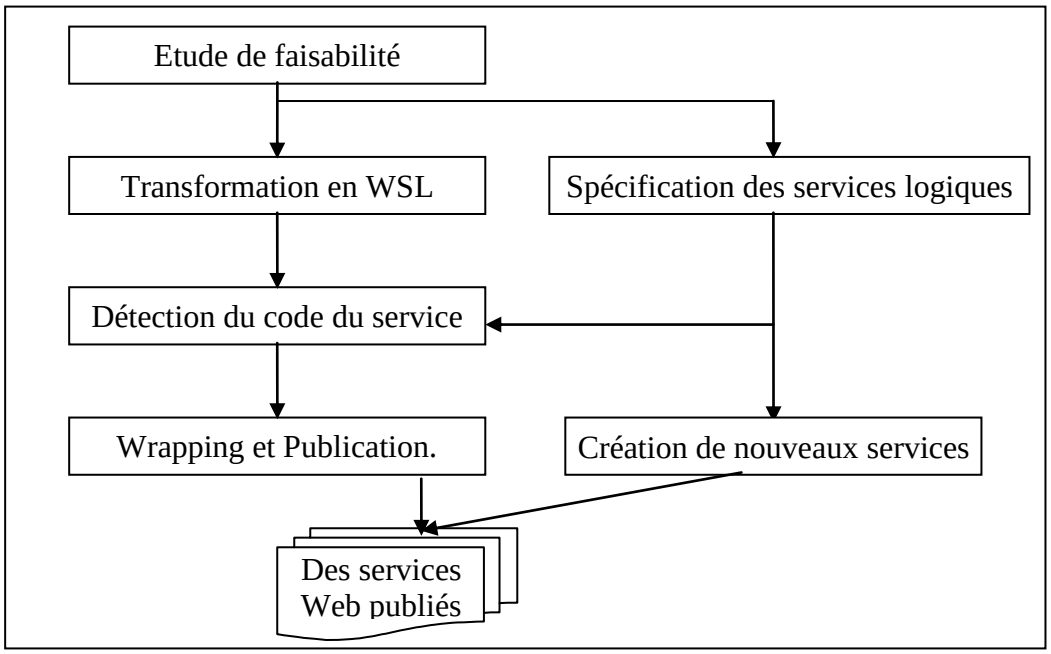


Figure 4.2. Processus d'extraction des services Web à partir d'un système patrimonial

4 Différentes étapes de l'approche

Dans cette section nous présentons en détail les différentes phases définissant notre framework [Figure 4.2] :

4.1 Etude de faisabilité

L'évaluation du système patrimonial est une phase essentielle pour une réutilisation éventuelle, puisque le système patrimonial a été employé et normalement maintenu pendant longtemps. Cette évaluation indique l'état actuel d'un système et indique quelle phase c'est

dans son cycle de vie [37]. Dans cette étape, les spécialistes du génie logiciel proposent d'utiliser quelques techniques qui nous aident à l'analyse et la décision, comme l'arbre de décision [36.37] et l'analyse des options pour la technique de reingénierie (OAR : The Options Analysis for Reengineering) [33].

En résumé, cette étude permet de répondre à la question : Est-il possible de faire migrer ce système patrimonial vers une architecture orientée service (SOA) ? Cette migration, est-elle rentable ?

La réponse à cette question est déterminée d'après les réponses aux questions suivantes [36] :

1. Le système a-t-il réellement besoin d'être migré vers un environnement distribué?
2. Le système contient-t-il des fonctionnalités réutilisables et fiables, englobant une logique d'affaires intéressante?
3. Ces fonctionnalités sont-elles utiles pour être exposées en tant que services Web indépendants?
4. La maintenance de ces fonctionnalités est-elle assez bénéfique par rapport à la maintenance du système entier ?
5. Ces fonctionnalités pourront-t-elles fonctionner correctement sur l'Internet où la fiabilité et la vitesse ne peuvent pas être garanties?

Une fois que ces conditions soient satisfaites, nous passons à l'étape suivante.

4.2 Définition des services logiques

Cette phase peut être subdivisée en deux étapes. La première permet de faire une étude du domaine d'application du système patrimonial qui est sujet de migration. L'autre phase consiste à faire une étude sur le système patrimonial lui-même.

Le but de l'analyse du domaine d'application est de définir et de documenter les besoins sur un ensemble de systèmes dans le même domaine d'application. Comme proposée dans [32,36], cette étape peut être accomplie en suivant deux processus, l'identification du sous domaine et l'analyse du sous domaine sélectionné.

Le processus d'analyse du domaine s'intéresse par le domaine du problème. L'identification des sous domaines permet de déterminer les frontières des sous domaines et décompose le domaine entier du problème, et permet aussi de modéliser les parties particulières du domaine. Les résultats de l'analyse du domaine sont résumés par un modèle

du domaine. Le modèle du domaine est représenté par des diagrammes UML [32]. En se basant sur le modèle du domaine d'application, des nouveaux services apparaissent comme candidats dans le processus de migrer. Ces services logiques se situent entre les services logiques et idéaux qui supportent exactement les fonctionnalités existantes dans le sous domaine et l'ensemble de composants de logiciel existants [32]. Donc, ces nouveaux services doivent être implémenté à part pour être intégré dans les nouveaux systèmes [36].

Pour arriver à de meilleurs résultats à l'issue de cette phase, nous proposons aussi d'inclure l'expérience des utilisateurs de ces systèmes patrimoniaux dans l'activité de l'analyse, pour les raisons suivantes :

- Les utilisateurs sont les seuls qui connaissent bien leurs systèmes. A partir de leurs expériences dans l'utilisation de ce système et d'autres systèmes différents, ils ont formé une idée très claire sur les fonctionnalités qui doivent être réutilisées. Ils peuvent même proposer facilement de nouvelles fonctionnalités à être ajoutées dans les nouveaux systèmes.
- Ils seront aussi les futurs utilisateurs des systèmes orientés services. Un principe fondamental du génie logiciel exige d'intégrer l'utilisateur dans toutes les étapes du cycle de vie d'un logiciel.

En conclusion, l'intégration des expériences des utilisateurs des systèmes patrimoniaux peut avoir une grande influence dans l'amélioration des résultats de cette étape et du projet en entier.

4.3 Transformation en langage WSL

Le but de cette phase consiste à étendre la gamme des applications qui peuvent être migrées par l'extension de l'ensemble des langages de programmation à traiter, car l'étape de détection du code du futur service Web dépend directement du code source et par conséquence du langage de programmation (C, COBOL, PASCAL,...etc.). Pour cela, nous proposons d'unifier le langage de programmation. En réalité, cette phase est composée de deux étapes successives, la traduction puis le raffinement [Figure 4.3] :

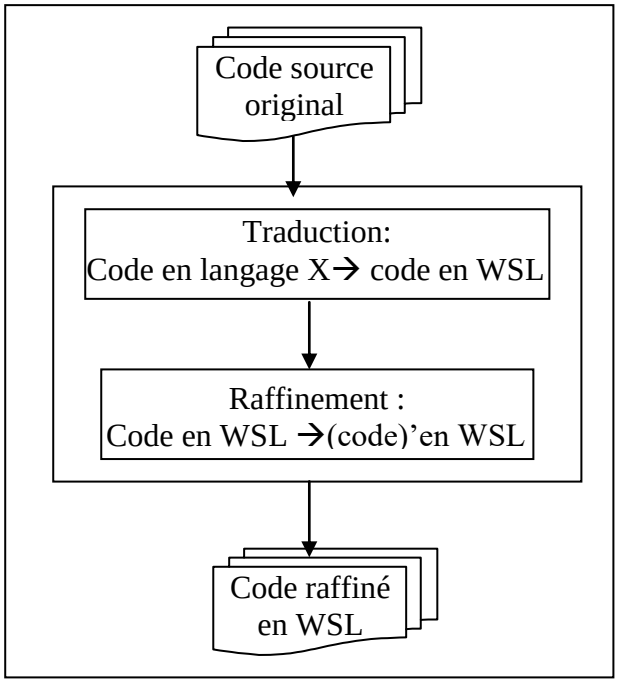


Figure 4.3. Phase de transformation

a. La traduction (Code Source Original→WSL)

La traduction du programme est une opération permettant de transformer le code source d'un programme vers une autre forme ayant le même comportement externe. Pour arriver à un programme en WSL équivalent au programme d'origine écrit dans un langage de programmation donné, il faut utiliser des règles de transformation bien précises et prouvées mathématiquement [45].

Ward et al [49,46,45] ont présenté plusieurs travaux concernant la transformation formelle des programmes des différents langages de programmation (Assembleur, COBOL, C,…) vers WSL. Ces travaux ont permis de définir toutes les bases théoriques et les sémantiques formelles de cette transformation. L'idée générale vise à représenter le programme comme une fonction qui transforme l'état initial d'un système donné avant l'exécution d'un programme à un état final de ce système après la fin de l'exécution du programme [45]. Alors, pour que cette transformation soit validée il faut que les deux programmes, le programme original et son équivalent en WSL arrivent au même état final s'ils ont démarré à partir d'un même état initial.

Donc, si l'utilisateur de cette approche désire de faire migrer un système patrimonial écrit dans un langage de programmation donné, il suffit seulement d'intégrer le transformateur (le traducteur) de ce langage à WSL. Nous fournissons un exemple qui

représente un programme écrit en langage C **(a)**, si nous appliquons le traducteur ***C→WSL***, nous pouvons arriver à une version équivalente en WSL **(b)** :

(a)

```
(1)  #include <stdio.h>
(2)  #include <stdlib.h>
(3)  #include <string.h>
(4)  #define CRITERION 20
(5)  int main(void)
(6)  {
(7)  char reply_yes[10] = "Yes";
(8)  char reply_no[10] = "No";
(9)  char reply[10];
(10) char buffer[80];
(11) double input1, input2, test ;
(12) printf("Enter first input: ");
(13) input1 = atof(gets(buffer));
(14) printf("Enter second input: ");
(15) input2 = atof(gets(buffer));
(16) test = input1 + input2 ;
(17) test = test + 2 ;
(18) test = test * 2 ;
(19) test = test + input2 ;
(20) test = test + 2 ;
(21) if(test >= CRITERION)
(22) strcpy(reply, reply_yes) ;
(23) else
(24) strcpy(reply, reply_no) ;
(25) printf("The answer: %s \n", reply);
(26) return(0);
(27) }
```

(b)

```
(1)  VAR < input1 := 0.0, input2 := 0.0,
         test := 0.0 >:
(2)  PRINFLUSH("Enter first input: ");
(3)  input1 :=
     String_To_Num(@Read_Line(
          Standard_Input_Port));
(4)  PRINFLUSH("Enter second input: ");
(5)  input2:= String_To_Num(@Read_Line(
          Standard_Input_Port));
(6)  test = input1 + input2 ;
(7)  test = test + 2 ;
(8)  test = test * 2 ;
(9)  test = test + input2 ;
(10) test = test + 2 ;
(11) IF test >= 20
(12) THEN PRINT("Answer: Yes")
(13) ELSE PRINT("Answer: No") FI;
(14) ENDVAR
```

b. Le Raffinement (Code en WSL →(code)'en WSL)

Après la traduction du programme source, l'étape suivante consiste à raffiner le code WSL résultant. Le but général de cette étape est de faciliter les étapes qui suivront, surtout la phase de l'extraction du code de la règle métier et aussi les autres futures phases de maintenances éventuelles. Il est nécessaire d'utiliser des règles de transformations prouvées mathématiquement à l'intérieur du langage WSL. A la fin de cette phase, quelques parties du code WSL peuvent être supprimées (code inutile) ou transférées et nous pouvons même fusionner quelques instructions [34,45]. Ceci permet d'obtenir un code WSL bien structuré, réduit, simple et compréhensible. Il est utile de noter que le raffinement n'arrive pas toujours à réduire la taille du programme à raffiner.

Pour clarifier cette idée nous présentons l'exemple suivant où la partie **(a)** présente un programme en WSL résultant de l'étape précédente. Après l'application de la phase de raffinement, nous pouvons arriver à une version raffinée du programme **(a)**, comme il est présenté dans **(b)** :

(a)

```
1   VAR < input1 := 0.0, input2 := 0.0,
            test := 0.0 >:
2   PRINFLUSH("Enter first input: ");
3   input1 := @String_To_Num(@Read_Line(
              Standard_Input_Port));
4   PRINFLUSH("Enter second input: ");
5   input2 := @String_To_Num(@Read_Line(
              Standard_Input_Port));
6   test = input1 + input2 ;
7   test = test + 2 ;
8   test = test * 2 ;
9   test = test + input2 ;
10  test = test + 2 ;
11  IF test >= 20
12  THEN PRINT("Answer: Yes")
13  ELSE PRINT("Answer: No") FI;
14  ENDVAR
```

(b)

```
1   VAR < input1 := 0.0, input2 := 0.0,
            test := 0.0 >:
2   PRINFLUSH("Enter first input: ");
3   input1 := @String_To_Num(@Read_Line(
              Standard_Input_Port));
4   PRINFLUSH("Enter second input: ");
5   input2 := @String_To_Num(@Read_Line(
              Standard_Input_Port));
6   test := input1 * 2 + input2 * 3 + 6 ;
7   IF test >= 20
8   THEN PRINT("Answer: Yes")
9    ELSE PRINT("Answer: No") FI;
10   ENDVAR
```

A la fin du cette phase, nous pouvons dire que le rôle du code source en langage d'origine est terminé. Les étapes restantes dans le processus d'émigration sont appliquées sur le nouveau code en WSL.

4.4 Détection du code des services Web

L'objectif de cette étape est d'extraire la partie du code qui implémente la règle métier désirée avec toutes les variables nécessaires. Plusieurs approches ont été proposées dans ce domaine. Nous avons choisi l'approche appelée « Code Stripping » présentée par Sneed [41]. Celle-ci est simple et réalisable dans le domaine des services Web [38,41].

Avant de déclencher le processus de détection, il faut d'abord spécifier la sortie finale retournée par chaque règle, puisqu'il sera possible de détecter où et comment ce résultat est calculé et déterminer quels sont les arguments utilisés.

Les étapes suivies, pour l'extraction du code de la règle métier, sont résumées comme suit [39] :

1. La première tâche est la collection de toutes les instructions d'affectation qui modifient directement le contenu de la variable du résultat final. Il est nécessaire de retourner l'instruction avec le numéro de la ligne qui la contient, comme il est indiqué dans l'exemple suivant :

   101 NOUVEAU-PRIX := 0;
   120 NOUVEAU-PRIX := ANCIEN-PRIX;
   122 NOUVEAU-PRIX :=ANCIEN-PRIX+ (ANCIEN-PRIX * TAUX-INFLATION) ;

2. La plupart des instructions sont incluses dans des structures de contrôle, c'est à dire des conditionnels simples (if), des alternatives (if….else), des boucles (while, for,…) et des instructions de sélection (Switch). Donc, il est nécessaire de collecter avec les affectations, qui modifient notre résultat final, les structures de contrôle qui les contiennent. L'exemple suivant présente une partie du code WSL, et contient quelques structures de contrôle :

   100 **FOR EACH** ARTICLE **DO**
   101 NOUVEAU-PRIX := 0
   119 **IF** TAUX -INFLATION < 0,02 **THEN**
   120 NOUVEAU-PRIX := ANCIEN-PRIX;
   122 **ELSE** NEW-PRICE = OLD-PRICE + (OLD-PRICE X INFLATION- RATE) **FI**
   125 **IF** ARTICLE-TYPE = TAXED-ARTICLE **THEN**
   130 NEW-PRICE = OLD-PRICE + (NEW-PRICE X VAT)**FI OD**

3. Répéter les étapes 1-3 pour toutes les variables qui influent directement ou indirectement sur le contenu du résultat final de la règle métier.

4. Après l'identification du code de la règle métier, il faut l'extraire et l'intégrer avec toutes les structures de données nécessaires dans un module séparé ayant une interface. Ceci produit une fonction avec une interface d'appel [38], où les variables d'entrées originales seront les paramètres d'entrée et les arguments de sortie seront les paramètres de sortie du futur service Web.

Nous avons résumé ces étapes dans l'algorithme suivant :

Algorithme Extraction de Code
Début
P : Code source WSL à traiter
Ligne : est un enregistrement de deux champs (Num, Instruction)
LLigne : est une liste des Lignes
LV : Liste des variables (Les structures des données utilisées par une règle métier)
StrctCtrl : est un enregistrement de 4 champs (Numligne, structure, numdébut, numfin) ;
LStrctCtrl : est une liste des StrctCtrl
V : Variable de sortie de la règle métier sélectionnée
Insérer (LV, V) ; /* la variable de sortie de la règle métier est insérée dans LV */

```
Collecter_les_structures_de_contôles(LStrctCtrl) ;
Tantque (LV !=Nil) faire
  Ligne= La première ligne de P ;
 Répétez
       Si LV est modifié par Ligne Alors
          début
            Insérer (LLigne, Ligne) ;
            Insérer (LV, toutes les variables V utilisées par cette instruction) ;
          finsi ;
    Ligne=Ligne->Suiv ;
  Jusqu'à la fin du programme ;
  Tantque (LStrctCtrl !=Nil) faire
     SI (LV→num > LStrctCtrl→numdébut et LV→num < LStrctCtrl→numfin) alors
          Insérer (LLigne, LStrctCtrl→Numligne) ;
     LStrctCtrl= LStrctCtrl →Suiv ;
  finTantque
  LV=LV→Suiv;
finTantque
FIN.
```

4.5 Adaptation et Publication

Cette phase est nécessaire pour qualifier les parties du code extraites dans la phase précédente, pour être des services Web fonctionnels. Elle est composée de plusieurs étapes comme décrit dans [32,36] :

a. Raffinement du code.

C'est la première étape dans le processus d'adaptation du code extrait, où nous proposons d'appliquer une phase de raffinement du code extrait, pour éliminer les répétitions et le code inutile afin de garantir que le futur service Web est indépendant, de granularité fine et faiblement couplé avec les autres services Web.

b. Définition de l'interface et description des services

Le but de cette phase est la définition des fonctionnalités des services Web en WSDL. Chaque service Web expose une interface qui définit les types de messages et les modes de leurs échanges. Pour cela, il faut d'abord spécifier une interface bien précise pour chaque service extrait. Ensuite, nous passons à une définition très rigoureuse des fonctionnalités des services Web en WSDL.

c. Intégration du mécanisme de transport

Une fois l'interface du service Web est définie, nous passons à l'intégration de mécanisme de transport qui sera le processeur SOAP qui garantit facilement l'échange des messages sur le Web à travers le firewall (pare-feu)**.**

d. Implémentation du « Code Interface »

Le « Code Interface » est une partie du code ajouté au-dessus du service Web, en tant qu'interface. Il joue le rôle de la réception des messages d'appels, l'extraction des paramètres d'entrées d'un service Web et l'empaquetage des résultats de l'exécution du service dans des messages et les envoyer vers les clients.

e. Création de nouveaux services

Dans cette étape, le développement des services, qui sont apparus dans la phase de détection des services logiques et qui ne peuvent pas être extraits du système patrimonial, est effectué. Ces nouveaux services seront intégrés par la suite dans les futurs systèmes orientés services pour étendre la logique d'affaires (Business Logic) de ces derniers.

f. Enregistrement du service

Enfin, les services Web résultants doivent être enregistrés dans l'UDDI, pour leur utilisation éventuelle par d'autres clients ou d'autres systèmes orienté services.

5 Discussion

Beaucoup de lecteurs peuvent se poser la question suivante : quelles sont les nouveautés de cette approche ? La réponse à la question est détaillée dans cette section où nous présentons une discussion mettant l'accent sur les points communs et les différences de notre approche des deux autres, celle de Sneed [38] et l'autre de Z.Zhang et al [32,36,37]. Nous exposons à la fin les avantages et les inconvénients.

Notre approche est commencée par une étude de faisabilité qui a été essentiellement extraite à partir de la phase de la compréhension de système patrimonial de Z.Zhang et al. Cette dernière est très vaste et nous nous sommes limités à la partie qui s'intéresse à l'étude de faisabilité. Nous avons proposé cette phase au démarrage du processus pour éviter les problèmes qui peuvent arriver si l'un des critères cités dans la section 4.1 n'est pas vérifié, à l'inverse de l'approche de Sneed qui reporte cette vérification après détection du code et applique quelques critères pour décider si ce code mérite d'être un service Web ou Non. De

cette manière, nous gagnons beaucoup d'efforts dans le cas où les résultats de vérification sont négatifs.

La deuxième phase de notre approche est aussi indispensable, puisqu'elle nous fournit le repère théorique des futurs services Web sur lequel nous nous basons pour détecter les services Web potentiels dans notre système patrimonial. Selon Sneed [38], cette phase est emboîtée dans la phase de détection du code. Au démarrage du processus, seules les variables de sortie des règles métier contenues dans le système patrimonial sont spécifiées.

Les deux approches principales [38,32,36,37] sont reliées fortement au langage de programmation du code source, et surtout dans la détection du code candidat à être un service Web, donc il est très clair que si on changera le langage de programmation il faudra effectuer de grandes modifications, au moins dans le module de détection du code. Pour cela, nous avons opté pour la transformation du code source original en langage WSL, pour arriver à un framework générique et extensible. Le choix du langage WSL est justifié par les avantages suivants [45]:

1. Les différents programmes, écrits dans différents langages de programmation et avec différents styles peuvent être spécifiés de manière consistante en WSL.

2. L'accomplissement des analyses d'un programme WSL permet d'analyser le code source ou même l'extraction de la règle métier indépendamment du langage de programmation initial.

3. Les règles métier dérivées de différents codes source peuvent être exprimées simplement par une forme commune en WSL.

En ce qui concerne la détection ***automatique*** du code du service Web, nous avons adopté la proposition de Sneed [38] qui considère que chaque service Web implémente une règle métier donnée. Il est donc nécessaire de détecter et de rassembler le code implémentant une règle métier et le transformer en un service Web, à l'inverse de Z.Zhang et al qui font recours à un expert humain dans cette phase. Alors que, il est très évident que l'intervention des humains dans ce type de processus est coûteuse et sensible aux erreurs.

Aussi, le raffinement, qui est proposé dans les deux autres approches, est d'un côté difficile et parfois même impossible dans la plupart des langages de programmation, le COBOL et l'assembleur par exemple et de l'autre côté il faut définir des règles de raffinement pour chaque langage de programmation utilisé. En utilisant le langage WSL, comme langage intermédiaire, nous atténuons ces problèmes.

La phase d'adaptation et de publication est présentée dans les trois approches puisqu'elle est nécessaire pour qualifier le code extrait pour être un service Web public, mais il faut signaler que selon notre approche une seule module d'adaptation, permettant la génération des documents WSDL décrivant les services Web résultants, quel que soit le langage de programmation original au contraire d'autres approches qui nécessitent de considérer chaque langage de programmation à part : c'est l'une des avantages de notre approche.

Au terme de cette discussion, nous dégageons quelques avantages et inconvénients de l'approche proposée. Parmi les avantages, nous citons :

- La possibilité d'extensibilité à plusieurs langages de programmation.
- La garantie de la standardisation en unifiant les langages de programmation.
- Les limites d'interventions des humains dans les phases du processus d'extraction.
- La facilité d'exploration des phases de maintenances éventuelles des services Web grâce à l'utilisation du langage WSL et aux phases de raffinement.

Néanmoins, cette approche nécessite une création et une intégration d'un traducteur du code pour chaque langage de programmation.

6 Conclusion

Dans ce chapitre, nous avons exposé notre approche pour l'extraction des services Web à partir des systèmes patrimoniaux. Cette approche est inspirée essentiellement des travaux de H.Sneed et Z. Zhang et al avec des contributions qui touchent la phase de l'analyse, en proposant l'intégration des expériences des utilisateurs des systèmes patrimoniaux dans la phase d'extraction des services Web logiques. D'autres changements sont suggérés en ce qui concerne la transformation du code source original en un langage formel qui est WSL. Nous attendons de cette étape l'unification du langage de programmation du code source qui sera traité par les étapes suivantes, pour arriver à une sorte de généricité absente dans les autres approches.

Pour la validation de cette approche, nous montrons, dans le chapitre suivant, son application sur un système patrimonial, implémenté en Langage C standard avec des outils simples, où le but est d'extraire une fonctionnalité qui apparaît utile et la réutiliser comme un service Web.

Chapitre 5 :

Etude de Cas

1 Introduction

Dans le chapitre précédent nous avons exposé notre approche pour l'extraction des services Web à partir des systèmes patrimoniaux où nous proposons d'ajouter quelques étapes pour l'amélioration du processus d'extraction des services Web et d'étendre la gamme des langages de programmation utilisés. Parmi ces améliorations, nous avons proposé l'intégration de l'expérience des utilisateurs dans la phase de détection des services Web logiques, l'utilisation du WSL comme langage universel pour unifier le langage de programmation et la considération de la règle métier comme le noyau d'un service Web .

Dans ce chapitre nous essayons d'appliquer notre approche pour l'extraction des services Web existantes dans une application patrimonial en langage C. Cette application est implémentée initialement pour résoudre les programmes linéaires de différents types, et nous avons remarqué qu'elle contient des fonctionnalités très intéressantes et valables pour être réutilisées comme des services Web.

Nous commencerons ce chapitre par l'introduction de notre application qui va être traitée, et les raisons qui nous ont amené à ce 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 ensuite nous passons à la phase de transformation duc code du langage C en langage WSL, puisque notre exemple est implémenté en langage C. Nous présentons d'abord la sémantique et la syntaxe des instructions WSL selon le langage C. Avant la transformation, nous proposons d'effectuer une étape de prétraitement, en essayant de supprimer les différentes instructions de branchement qui permettent de rendre le code déterministe et compréhensible. Après la détection et l'extraction du code implémentant une règle métier, nous passons ensuite à la phase d'adaptation où nous transformerons le code extrait en services Web concret, bien raffiné, faiblement couplé avec les autres services et de granularité fine. Enfin, Nous terminerons ce chapitre par une conclusion.

2 Système de résolution des programmes linéaires

L'application que nous avons choisi pour illustrer la validité de cette approche est un programme pour résoudre les trois types de programmes linéaires, primal, dual et mixte. Ce système reçoit un programme linéaire et retourne les solutions ou un message d'erreur. L'exemple qui suit présente les trois types de programmes linéaires qui peuvent être résolues par notre application à migrer :

(1)	(2)	(3)
3A+5B>=0	3A+5B<=0	3A+5B<=0
6A-B>=0	6A-B<=0	6A-B=0
Min(z)=14A+5B	Max(z)=14A+5B	Min(z)=14A+5B
Programme Dual	**Programme primal**	**Programme mixte**

Cette application a été développée depuis 06 ans en langage C et sous le système d'exploitation UNIX, en suivant le paradigme procédural, elle est composée de 17 fonctions. Nous avons choisi cette application puisqu'elle contient des fonctionnalités très intéressantes et elle est apparue comme un candidat idéal pour la migration puisque elle est implémentée en utilisant des moyens simples.

3 Extraction des services Web

Pour arriver à des services extraits à partir de cette application, il faut suivre les étapes suivantes :

3.1 Spécification des services Web

Cette phase commence par une étude qui a été effectuée sur notre application, l'application de résolution des programmes linéaires, nous avons découvert qu'il y a beaucoup de fonctionnalités qui peuvent être réutilisées comme services Web. Parmi ces fonctionnalités nous citons par exemple :

- Les services Web simples; ce type de services Web peut être défini à partir des fonctions qui composent le système, comme :
  i. permutation (float ta1[N],float ta2[N]): permutation de deux tableaux flottants.
  ii. transpose (float a[N][N], int *nbrx,int *nbry): transposer une matrice
  iii. permutchar(char *a, char *b)…etc. permutation de deux chaînes de caractères.
  iv. ....etc.
- Les services Web qui ont des tâches bien spécifiques comme :
  i. float convertnbr(char s[N],int m) : convertit une chaîne ce caractères numériques en une nombre flottant.
  ii. Traitement (char c, float t[N], char str[N], int r, float *b,char *x) : lire une contrainte et produire tous ses composant.
  iii. pivotement(float mat[N][N],float disp[N], float cout[N], int nbrx, int nbry, int s, int r, float *Z) : permet de réaliser la phase de pivotement du processus de résolution d'un programme linéaire.
  iv. ...etc.

3.2 Transformation en WSL

Après une spécification abstraite des services Web potentiels qui peuvent être extraits de cette application ,nous passons ensuite à l'étape suivante qui permet d'arriver à un code source WSL produit, bien raffiné et équivalent à l'application d'origine, comme l'indiquent les sous étapes suivantes :

3.2.1 Prétraitement

Avant de démarrer le processus de transformation du système du langage C à WSL, nous suggérons d'abord de faire des modifications permettant de supprimer les branchements existants dans l'application originale, surtout l'instruction « GOTO », et les deux autres instructions « BREAK » et « CONTINUE », pour arriver à un programme quasi déterministe. La procédure suivie pour l'élimination du « GOTO » est proposée dans [30] :

```
……
point1: Bloc d'instructions1
Si (Condition1) alors
      Bloc d'instructions2
      goto point2
finSi
Bloc d'instructions3
Si (Condition2) alors
       Bloc d'instructions4
       goto point1
finSi
point2: Bloc d'instructions5
……
```

(a) : programme contient goto

```
……
Répétez
Bloc d'instructions1
Si (Condition1) alors
      Bloc d'instructions2
Sinon
       Bloc d'instructions3
       Si (Condition2) alors
              Bloc d'instructions4
       finSi
finSi
jusqu'à (Condition1)
Bloc d'instructions5
```

(b): après la suppression du goto

L'exemple suivant présente une partie du code **(a)** et après l'application du processus de transformation sur notre programme nous arrivons à la partie du code **(b)** :

(a)

```
……
et: j=0;
fec[j++]=c;
while(j<=6)
    fec[j++]=getchar();
faux=comparmax(fec,mx,0);
if(faux==0)
{
  faux=comparmax(fec,mn,0);
  if(faux==0)
  {
    printf("il y a une erreur dans la fec\n");
    c=getchar();
```

(b)

```
……
do{
  j=0;
fec[j++]=c;
while(j<=6)
    fec[j++]=getchar();
faux=comparmax(fec,mx,0);
if(faux==0)
{
  faux=comparmax(fec,mn,0);
  if (faux==0)
   {
      printf("il y a une erreur dans la fec\n");
```

```
  goto et;
  }
 faux*=(-1);
}
……
```

```
      c=getchar();
   }
 else
          faux*=(-1);
 }
}while(faux==0)
……
```

Après la phase de prétraitement nous passons ensuite à la phase de transformation.

3.2.2 La transformation (Langage C→WSL)

Le but de cette phase est de transformer le code source original de l'application en une version équivalente, mais cette fois-ci en langage WSL. Puisque notre cas à étudier est en langage C, nous allons d'abord présenter la sémantique et la syntaxe des instructions WSL et selon le langage C [30.27] :

L'opération	**Langage C**	**Langage WSL**
L'affectation	x=e	x :=e
Le conditionnel	if condition Bloc d'instructions1 else Bloc d'instructions2	IF cond1 THEN instruction1 ELSIF cond2 THEN instruction2 ... ELSE instruction n FI
Le choix	Switch(x){ Case1 :instruction1 ;break ; Case2 :instruction1 ;break ; … Case n :instruction n ;break ; }	IF cond1 -> stat1 cond2 -> stat2 … condn -> statn FI
La boucle while	WHILE condition Bloc d'instructions	WHILE condition DO ... OD
La boucle for	FOR (i=0 ;i< N; i++) Bloc d'instructions	FOR i:=1 TO N STEP 1 DO ... DO
Nom du procedure	void nom_du_proc (parameters)	PROC nom_du_proc(parameters) ≡
Format du procedure	void nom_du_proc (parameters) { Déclaration des variables locales ; Partie d'actions }	PROC nom_du_proc (parameters) ≡ VAR (); ACTIONS main-procedure: Main-procedure ≡ …… sub-procedure1 ≡ …… sub-procedure2 ≡ …… ENDACTIONS
Appel du procedure	Main() {	….. CALL nom_du_proc; ….

	nom_du_proc (parameters) ; .... }	
Des relations usuels	a < b, a > b, a = b, a !=b, a <= b, a >= b	a < b, a > b, a = b, a <> b, a <= b, a >= b
……	…….	…….

Tableau 5.1 Sémantique du WSL selon langage C.

Puisque le programme est assez long, nous allons exposer seulement la partie du code qui nous intéresse dans les étapes restantes.

(a)

```
float convertnbr(char s[N],int m){
 int p,signe;
 float nbr,k;
 k=1;
 nbr=0;
 p=0;
 signe=1;
 if((s[0]=='+')||(s[0]=='-')) {
   if(s[0]=='+')
      signe=1;
   else
     signe=-1;
   p++; }
 if(p==m)
    nbr=1;
 else {
   while((p<m)&&(s[p]!='.'))
       nbr=nbr*10+(s[p++]-'0');
    if(s[p++]=='.')
      while(p<m)    {
          k*=0.1;
          nbr+=((s[p++]-'0')*k);  } }
 nbr*=signe;
 return nbr;}
/************************************/
.....
/************************************/
traitement(char c, float t[N], char str[N], int r,
float *b,char *x){
char tab[N];
int i,j,z,vrai;
z=j=0;
while(c!='\n')  {
     i=0;
    vrai=0;
    if((c=='+')||(c=='-'))
     { tab[i++]=c;
       c=getchar();     }
```

(b)

```
1.  PROC traitement (c, str, r,VAR t,b,x,v) =
2.  Var (tab:= " " ;i=1 ;z :=1) ;
3.  ACTIONS a-trait :
4.  a-trait ≡
5.  WHILE (c<>'\n') DO
6.  i:=1;
7.  v:=0;
8.  IF(c=='+'OU c=='-') THEN
9.  tab[i]:=c ;
10.i:=i+1;
11.c :=@Read_Char(port);FI
12.WHILE(c>='0'and c<='9' OUc='.') DO
13. tab[i]:=c
14.;i:=i+1;
15.c := @Read_Char(port) OD
16.IF(c>='A' AND c<='Z') THEN
17.z=call search(c,str,r);
18.IF(z<0) THEN
19.PRINT("ERREUR")
20.v=0 ;
21.ELSE
22.Call convertnbr(tab,I,t[z]);
23.c := @Read_Char(port);
24.IF(c='+' OU c='-' OU c='>' OU c='<' OU
   c='=' OU c='\n') THEN
25.v:=1 FI
26.i:=1 FI
27.IF (c='>' OU c='<' OU c='=') THEN
   x :=c FI
28.IF(v=1) THEN
29.IF(c='>' OU c='<') THEN
  c := @Read_Char(port) FI
30.IF (c='=') THEN
31. v=0 ;
32.c := @Read_Char(port);
33.IF ((c='+' )OU (c='-'))THEN
34.tab[i]:=c;
35.i:=i+1;
```

```
   while(((c>='0')&&(c<='9'))||(c=='.'))    {
     tab[i++]=c;
     c=getchar();    }
   if((c>='A')&&(c<='Z'))   {
     z=search(c,str,r);
     if(z<0)  {
      printf("érreur");
      vrai=0;   }
     else   {
 t[z]=convertnbr(tab,i);
 if(((c=getchar())=='+')||(c=='-
')||(c=='>')||(c=='<')||(c=='=')||(c=='\n'))
                    vrai=1;
        i=0;  }  }
  if((c=='>')||(c=='<')||(c=='='))
    *x=c;
   if(vrai==1)    {
     if((c=='>')||(c=='<'))
 c=getchar();
     if(c=='=')     {
         vrai=0;
     if(((c=getchar())=='+')||(c=='-'))    {
          tab[i++]=c;
    c=getchar();  }
  while(((c>='0')&&(c<='9'))||(c=='.')){
                    tab[i++]=c;
         c=getchar();    }
         if((c=='\n')&&(i>0))      {
           *b=convertnbr(tab,i);
    vrai=1;  } }}
   else  {
              printf("ERRRRRRREUR\n");
              while(c!='\n')
       c=getchar();    }  }
 return vrai;}
/************************************/
.....
/************************************/
Main()
{
Char c,horb[N],fec[7], bas[N], horb2[N],
sig[N];
 int nbrx, nbry, i, j, k, faux, s ,boole ,r,
cp1,cp2;
 float b,cont[N],mat[N][N],disp[N],cout[N],
coub[N],z,Z;
 .....
printf("ENTREZ LES CONTRAINTES\n");
 while((c=getchar())!='m')    {
faux=traitement(c,cont,horb,nbrx,&b,
```

```
36.c := @Read_Char(port) FI
37.WHILE ((c>=’0’)and (c<=’9’))OU(c=’.’))
   DO
38.tab[i]:=c;
39.i:=i+1;
40.c := @Read_Char(port) OD
41.IF (c='\n' AND i>0) THEN
42.Call convertnbr(tab,i,t[z]);
43.v:=1 FI
44.FI FI
45.ELSE PRINT ("ERRRREUR ") FI OD
46. WHILE PROC convertnbr (s,m,VAR nbr) =
47.Var (p:=1;signe:=1;nbr:=0.00;k:=1.00)
48.ACTIONS a-main:
49.a-main ≡
50.IF(s[1]=’+’ OU s[1]=’-’) THEN
51.IF(s[1]=’+’) THEN
52.signe=1 ;
53.ELSE
54.signe:=-1 FI
55.p:=p+1; FI
56.IF p=m THEN
57. nbr:=1
58.ELSE
59.WHILE (p<=m) AND (S[p]<>’.’) DO
   nbr:=nbr*10+(s[p]-‘0’);
60.p:=p+1 OD
61.IF s[p]=’.’ THEN
62. p:=p+1;
63.WHILE (p<=m) DO
64.k:=k*0.1;
65.nbr:=nbr+(s[p]-‘0’)*k;
66.p:=p+1 OD FI
67.nbr:=nbr*signe
.....
ENDACTION
END
.....
BEGIN
100. VAR< horb :=""; fec:="";bas:="";horb2:=""
   sig:="";c:=<>; nbrx :=0;nbry :=0,i :=0 ;j :=0;
   k :=0;faux:=0 ;s=0 ;boole :=0 ;r :=0 ;cp1 :=0 ;
   cp2 :=0 ;b :=0.00 ; cont := ARRAY(10, 0.00);
   disp :=ARRAY(10, 0.00);cout := ARRAY(10
   0.00); coub:= ARRAY(10, 0.00); Z:=0.00;
   z:=0.00;mat:= ARRAY((10,10), 0.00); >
….
110. PRINT(“ENTREZ LES CONTRAINTES”)
111. c := @Read_Char(port) ;
112. WHILE(c !=’m’) DO
```

&sig[nbry]); if((sig[nbry]!='<')&&(sig[nbry]!='>')&&(sig[nbry]!='=')&&(faux==1)) { printf("IL FAUT LE SIGNE\n"); faux=0; } else i=verification(&sig[nbry],b,&cp1,&cp2); if(faux!=0) { for(j=0;j<nbrx;j++) { mat[nbry][j]=i*cont[j]; cont[j]=0; } disp[nbry]=i*b; bas[nbry++]='y';} } ....}	113. c := @Read_Char(port) ; 114. CALL traitement(c,cont,horb,nbrx,b,sig[nbry],faux) 115. IF(sig[nbry]!='<'AND sig[nbry]!='>' AND sig[nbry]!='='AND faux==1) THEN 116. PRINT ("IL FAUT LE SIGNE\n"); 117. faux=0; 118. ELSE CALL verification(b, sig[nbry],i , cp1, cp2) FI 119. IF(faux!=0) THEN 120. FOR j := 1 TO nbrx STEP 1 DO 121. mat[nbry][j]:=i*cont[j]; 122. cont[j]:=0; OD 123. disp[nbry]:=i*b; 124. bas[nbry]:='y'; 125. nbry:= nbry+1 FI OD 126. END

Listing 1. Traduction du code C à WSL

3.2.3 Raffinement

Cette phase est très importante dans notre approche, puisqu'elle permet de supprimer les instructions inutiles selon le point de vue du WSL, et même de fusionner des instructions quand il est possible pour compacter le corps du programme. Une autre opération peut être aussi effectuée dans cette phase, à savoir l'adaptation des instructions résultantes au format WSL, comme il est illustré dans l'exemple suivant :

La séquence de conditionnel IF dans **(1)** sera représentée selon WSL comme dans **(2).**

(1)	**(2)**
….. 27. IF (c='>' OU c='<' OU c='=') THEN 28. x :=c FI 29. IF(v=1) THEN 30. IF(c='>' OU c='<') THEN 31. c:= @Read_Char(port) 32. …..	……. 27.IF 28.(c='>' OU c='<' OU c='=') →x :=c 29. (v=1) → IF(c='>' OU c='<') THEN 30.c:= @Read_Char(port) FI 31.….

3.3 Détection du code des services Web

Parmi les parties du système qui sont apparues intéressantes, nous avons sélectionné la fonctionnalité qui permet de lire une chaîne de caractères représentant une contrainte et retourne la liste des coefficients, des variables, le signe et la valeur de disponibilité, comme il est illustré dans l'exemple suivant :

L'entrée est : 3A+B>=12.5

Les sorties seront comme suit :

- Les variables sont : A et B.
- Les coefficients sont 3.00 et 1.00.
- Le signe est le symbole '>' (duale).
- La disponibilité est : 12.5.

Donc, le processus de détection code du service Web sera démarré par la détection de la variable de sortie de la règle métier qui implémente le processus d'extraction.

Avant de démarrer le processus de détection du code, il faut d'abord spécifier quels sont les sorties de notre règle métier et ensuite déterminer le partie du code qui est responsable de générer ces sorties et ensuite, nous suivrons les traces comme nous l'avons expliqué dans le chapitre précédent. Le résultat de cette phase sera comme suit :

```
PROC traitement (c, str, r,VAR t,  b,x,v) ==
1.  Var (tab:= " " ;i=1 ;z :=1) ;
2.  ACTIONS a-main2 :
3.  a-main2 ≡
4.  WHILE (c<>'\n') DO
5.  i:=1;
6.  v:=0;
7.  IF(c=='+'OU c=='-') THEN
8.  tab[i]:=tab[i]+c;
9.  c: =getchar() FI
10. WHILE((c>='0'AND c<='9') OU( c='.')) DO
11. tab[i]:=tab[i]+c;
12. c: =getchar() OD
13. IF(c>='A' AND c<='Z') THEN
14. z=call search(c,str,r);
15. IF(z<0) THEN
16. v=0 ;
17. ELSE
18. Call convertnbr(tab,i,t[z]);
19. c:=getchar();
20. IF(c='+' OU c='-' OU c='>' OU c='<' OU c='=' OU c='\n') THEN
21. v:=1 FI
22. i:=1 FI
23. IF (c='>' OU c='<' OU c='=') THEN
24. x :=c FI
25. IF(v=1) THEN
26. IF(c='>' OU c='<')  THEN
27. c:=getchar() FI
28. IF (c='=') THEN
29. v=0 ;
```

```
30. c:=getchar() ;
31. IF (c=’+’ OU c=’-‘)THEN
32. tab[i]:=c;
33. i:=i+1;
34. c:=getchar() FI
35. WHILE ((c>=’0’and c<=’9’)OU(c=’.’)) DO tab[i]:=tab[i]+c;
36. i:=i+1;
37. c: =getchar() OD
38. IF (c='\n' AND i>0) THEN
39. Call convertnbr(tab,i,b);
40. v:=1 FI FI  FI  FI OD
41. ENDACTION
END
```

Listing 2. Code WSL brute qui implémente la règle métier

Après la détection du code de la règle métier, nous pouvons dire qu'un grand pas dans le processus d’extraction de ce service Web a été effectué, et il ne nous plus reste que le raffinement et l’adaptation du code (le wrapping)

4 Adaptation et Publication

Après la détection du code du futur service Web, la phase suivante sera la phase d’adaptation et de publication. Cette dernière est composée d’un ensemble de sous étapes, qui nous mènerons à la fin à un service web concret qui pourra être publié ou même réutilisé dans d'autres systèmes orientés services.

4.1 Raffinement du code.

Dans cette étape nous proposons de faire quelques passages de raffinement, comme suit :

a. Minimisation de l’interactivité avec l’extérieur

Nous proposons la modification du code de la règle métier dont le but est de minimiser au maximum l’interactivité entre le programme et l’utilisateur, puisque le client du service Web résultant envoie toutes les entrées en bloc dans un message, selon bien sûr la description WSDL de ce service Web, et « le code interface » ensuite extraira ces entrées et les transférera vers le service. Les résultats de l’exécution de ce service seront renvoyés vers le client en bloc c'est-à-dire l’interactivité est différée. A l’issue de cette phase, nous obtenons les résultats suivants :

1.

Insertion d'une chaîne pour contenir la liste des variables

Après la minimisation du l'interactivité entre le code et l'extérieur, nous trouvons aussi que le code reste fortement dépendant avec l'extérieur, comme il est illustré dans la partie de code suivante :

.....

14.IF(s[k]>='A' AND s[k]<='Z') THEN

15.z=call search(c,str,r);

16.IF(z<0) THEN

17.v=0 ;.

…..

La procédure « *search(c,str,r)* » retourne l'indice d'une variable, trouvée dans la liste des variables déjà lue par d'autres services.

Dans ce nouveau contexte du service Web, cette étape n'est pas nécessaire. Pour cela, nous proposons d'éliminer cette partie de code et de la remplacer par une variable représentant une chaîne de caractères (**Z[N]**) pour contenir les variables détectées par notre service, comme suit :

PROC traitement (s VAR T,Z,z,b,x,v) ==

Var (k:=1; tab:= " " ; i:=1 ; z :=1) ;

ACTIONS a-main2 :

a-main2 ≡

WHILE (s[k]<>'\n') DO

i:=1;

v:=0;

IF (s[k]=='+'OU s[k]=='-') THEN

```
1. tab[i]:=tab[i]+ s[k];
2. k:=k+1 FI
3. WHILE((s[k]>='0'AND s[k]<='9') OU(s[k]='.')) DO
4. tab[i]:=tab[i]+ s[k];
5. i:=i+1
6. k+1 OD
7. Call convertnbr(tab,i,t[z]);
8. IF(s[k]>='A' AND s[k]<='Z') THEN
9. Z[z]=s[k];
10. z:=z+1;
```

```
11. k :=k+1 ;
12. IF(s[k]='+' OU s[k]='-' OU s[k]='>' OU s[k]='<' OU s[k]='=' OU s[k]='\n') THEN
13. v:=1 FI
14. i:=1;
15. IF(s[k]='>' OU s[k]='<' OU s[k]='=') THEN
16. x := s[k] FI
17. IF(v=1) THEN
18. IF(s[k]='>' OU s[k]='<')  THEN
19. k:=k+1 FI
20. IF (s[k]='=') THEN
21. v=0 ;
22. k:=k+1;
23. IF ((s[k]='+' )OU (s[k]='-'))THEN
24. tab[i]:= s[k];
25. i:=i+1;
26. k:=k+1 FI
27. WHILE ((s[k]>='0'and s[k]<='9')OU(s[k]='.')) DO
28. tab[i]:= s[k];
29. i:=i+1;
30. k:=k+1 OD
31. Call convertnbr(tab,i,b);
32. IF (s[k]='\n' AND i>0) THEN
33. v:=1 FI FI  FI  OD
34. ENDACTION
35. END
```

Listing 4. Code WSL résultant de la deuxième étape de raffinement

b. Elimination de la redondance

Nous essayons ici d'éliminer la redondance du code que nous avons remarqué dans la phase précédente. La partie du code redondant est :

```
IF (s[k]=='+'OU s[k]=='-') THEN
tab[i]:=tab[i]+ s[k];
k:=k+1 FI
WHILE((s[k]>='0'AND s[k]<='9') OU(s[k]='.')) DO
tab:=tab+ s[k];
k+1 OD
Call convertnbr(tab,i,V);
```

Ainsi, nous pouvons remplacer cette partie de code par une fonction appelée *LectureNombre* qui contient le même code et réalise la même tâche pour simplifier et minimiser le code résultant, comme suit :

```
PROC LectureNombre (s VAR k,V) ==
Var (tab:= " " ; i:=1 ) ;
ACTIONS act :
act ≡
IF (s[k]=='+'OU s[k]=='-') THEN
```

```
tab[i]:=tab[i]+ s[k];
i:=i+1
k:=k+1 FI
WHILE((s[k]>=’0’AND s[k]<=’9’) OU(s[k]=’.’)) DO
Tab[i]:=tab[i]+ s[k];
i:=i+1
k+1 OD
Call convertnbr(tab,i,V);
ENDACTIONS
```

Alors la nouvelle version de notre code sera comme suit :

```
1.  PROC traitement (s VAR T,Z,z,b,x,v) ==
2.  Var (k:=1; z :=1) ;
3.  ACTIONS a-main2 :
4.  a-main2 ≡
5.  WHILE (s[k]<>’\n’) DO
6.  v:=0;
7.  CALL LectureNombre (s k,t[z]) ;
8.  IF(s[k]>=’A’ AND s[k]<=’Z’) THEN
9.  Z[z]=s[k];
10. z:=z+1;
11. k :=k+1 ;
12. IF(s[k]=’+’ OU s[k]=’-‘ OU s[k]=’>’ OU s[k]=’<’ OU s[k]=’=’ OU s[k]=’\n’) THEN
13. v:=1 FI
14. IF(s[k]=’>’ OU s[k]=’<’ OU s[k]=’=’) THEN
15. x := s[k] FI
16. IF(v=1) THEN
17. IF(s[k]=’>’ OU s[k]=’<’)  THEN
18. k:=k+1 FI
19. IF (s[k]=’=’) THEN
20. v=0 ;
21. k:=k+1;
22. CALL LectureNombre (s k,b) ;
23. IF (s[k]='\n' AND i>0) THEN
24. v:=1 FI FI  FI  OD
25. ENDACTION
26. PROC LectureNombre (s VAR L,V) ==
27. Var (tab:= “ ” ; i:=1 ) ;
28. ACTIONS act :
29. act ≡
30. IF (s[L]==’+’OU s[L]==’-‘) THEN
31. tab[i]:=tab[i]+ s[L];
32. i:=i+1
33. L:=L+1 FI
34. WHILE((s[L]>=’0’AND s[L]<=’9’) OU(s[L]=’.’)) DO
35. Tab[i]:=tab[i]+ s[k];
36. i:=i+1
37. L+1 OD
38. Call convertnbr(tab,i,V);
39. ENDACTIONS
```

```
40. PROC convertnbr (s,m,VAR nbr) =
41. Var (p:=1;signe:=1;nbr:=0.00;k:=1.00)
42. ACTIONS cvtnbr:
43. cvtnbr ≡
44. IF(s[1]='+' OU s[1]='-') THEN
45. IF(s[1]='+') THEN
46. signe=1 ;
47. ELSE
48. signe:=-1 FI
49. p:=p+1; FI
50. IF p=m THEN
51.  nbr:=1
52. ELSE
53. WHILE (p<=m) AND (S[p]<>'.') DO nbr:=nbr*10+(s[p]-'0');
54. p:=p+1 OD
55. IF s[p]='.' THEN
56.  p:=p+1;
57. WHILE (p<=m) DO
58. k:=k*0.1;
59. nbr:=nbr+(s[p]-'0')*k;
60. p:=p+1 OD FI
61. nbr:=nbr*signe
62. END ENDACTIONS
```

Listing 5. Code WSL raffiné de la règle métier

Donc le code source résultat du futur service Web est bien raffiné et bien clair, ce qui nous permet de le qualifier en tant que service Web concret, et nous passons ensuite à la définition du l'interface en WSDL.

4.2 Développement de l'interface et la description des services.

Dans cette section, nous allons créer une description complète de notre service selon le format WSDL, pour que les différents utilisateurs puissent l'utiliser ou le réutiliser dans le développement des systèmes orientés services.

Alors, notre service est composé de trois fonctions : Une principale est deux secondaires. La fonction principale reçoit une chaîne de caractères comme entrée et retourne six sorties, qui sont un booléen qui décide si la contrainte est bien formée ou non, un tableau contenant les coefficients, un tableau contenant la liste des variables et un entier présentant la taille des ces deux tableaux, le signe de la contrainte (> ou < ou =) et enfin le valeur de disponibilité.

Comme nous avons vu dans le deuxième chapitre, le document WSDL décrivant notre service Web sera comme suit :

```
<?xml version="1.0"?>
<definitions name="lectureContrainte"
targetNamespace="http://www.pgprojet.com/ lectureContrainte .wsdl"
xmlns:tcns=" http://www.pgprojet.com/ lectureContrainte"
xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:xsdl= "http://www.pgprojet.com/ lectureContrainte .xsd"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
<schema targetNamespace=" http://www.pgprojet.com/ lectureContrainte .xsd "
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<complexType name="traitementType">
<sequence>
                        <element name="boole" type="xsd:boolean/>
                        <element name="Coefficentvariable" maxOccurs=”unbounded” >
                            <complexType >
       <sequence>
            <element name="Coefficient" type="xsd:float"/>
    <element name="variable" type="xsd:char"/>
        </sequence>
   </complexType>
                         </element>
                         <element name="signe" type="xsd:char"/>
          <element name="disponibilité" type="xsd:float"/>
</sequence>
</complexType>
</schema>
</types>
<message name=" traitementContrainte Request">
<part name="contrainte" type="xsd:string"/>
</message>
<message name=" traitementContrainte Response">
<part name="result" type="xsdl:traitementType "/>
</message>
<portType name=" traitementContraintePortType">
<operation name=" traitementContrainte ">
<input message=" tcns: traitementContrainte Request"/>
<output message=" tcns: traitementContrainte Response"/>
</operation>
</portType>
<binding name=" traitementContrainte SOAPBinding" type=" tcns:traitementContrainte PortType">
<soap:binding style="message" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name=" traitementContrainte ">
<soap:operation soapAction=""/>
<input>
    <soap:body use="encoded" namespace=       http://www.pgprojet.com/ lectureContrainte .xsd
                        encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/"/>
</input>
<output>
<soap:body use="encoded" namespace= " http://www.pgprojet.com/ lectureContrainte .xsd "
encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/"/>
</output>
</operation>
</binding>
<service name=" traitementContrainte Service">
```

```
<port name=" traitementContraintePort " binding=" tcns :traitementContrainte SOAPBinding">
    <soap:address location= " http://www.pgprojet.com "/>
 </port>
</service>
 </definitions>
```

Listing 6. Document WSL décrivant notre service Web résultant

4.3 Intégration de mécanisme de transport (SOAP)

Après ce qui précède nous trouvons qu'il faut trois types de messages SOAP, message d'appel, message de réponse et message d'erreur.

a. Message Appel

Ce message est envoyé par un client du service, il contient une chaîne de caractères, représentant une contrainte, comme paramètre :

```
POST  http://www.pgprojet.com HTTP/1.0
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: xxxx
SOAPAction: ""
<?xml version="1.0" encoding="UTF-8"?>
< ENV: Envelope xmlns:ENV="http://schemas.xmlsoap.org/soap/envelope/">
< ENV:Body>
        < traitementContrainte Response >
              < contrainte >3A+5B>=12.5< /contrainte >
          </ traitementContrainte Response >
</ ENV:Body>
</ENV:Envelope>
```

Listing 7. Message SOAP permettant l'invocation du notre service Web

b. Message Réponse

Ce deuxième message est envoyé par le serveur qui exécute le service Web. Il contient les résultats de l'exécution du service Web en cas de succès. Ces arguments sont : la liste coefficients-variables, valeur disponibilité, le signe de contrainte :

```
Content-Type: text/xml; charset=utf-8
Content-Length: xxxx
<?xml version="1.0" encoding="UTF-8"?>
< ENV: Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
< ENV:Body>
        < traitementContrainte Response >
              < boole>1</ boole>
                 < coefficentvariable>
 <coefficient>3.00</coefficient>
 < variable>A< /variable>
                 </ coefficentvariable>
```

```
                    < coefficentvariable>
  <coefficient>5.00 </coefficient>
  < variable>B < /variable>
                    </ coefficentvariable>
                 < signe> “>” </ signe >
          < disponibilité>12.5</ disponibilité >
          </ traitementContrainteResponse >
</ ENV:Body>
</ENV:Envelope>
```

Listing 8. Message SOAP de réponse de notre service Web

c. Message d’erreur

Ce dernier message est aussi envoyé par le serveur qui exécute le service Web, mais cette fois-ci dans le cas d’échec de l’exécution, c’est à dire dans le cas où une erreur est envisagée dans la contrainte reçue.

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: xxx
<?xml version="1.0" encoding="UTF-8"?>
< ENV: Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
< ENV:Body>
< ENV:Fault>
<faultcode>env:Client</faultcode>
<faultstring> Erreur dans la chaine émis</faultstring>
</ ENV:Fault>
</ ENV:Body>
</ENV:Envelope>
```

Listing 9. Message SOAP d’erreur de notre service Web

4.4 Implémentation du « Code Interface »

Comme nous avons indiqué dans le chapitre précédent, le « Code Interface » joue le rôle d’une interface qui reçoit les arguments d’un service Web et renvoie les résultats de l’exécution aux clients. Donc, l’algorithme qui peut être utiliser pour implémenter le « Code Interface » sera comme suit :

Algorithme Code_Interface

DEBUT
S[M],V[N]: chaîne de caractère ;
Coeff [N],disp :réel ;
Boole,nbv :entier ;
sig : caractère
Extraire_des_entrée (message SOAP Appel(S)) ;
Executer (traitement(S,Coeff,V,nbv,disp,sig ,boole)) ;

SI(boole==1) *alors* créer(message_SOAP_Response(Coeff,V,nbv,disp,sig)) ;
Sinon créer(message_SOAP_erreur) ;
FIN

5 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. Ceux-ci sont prêts à la publication et à l'intégration avec d'autres services pour la réutilisation éventuelle dans l'implémentation de nouveaux systèmes orientés services.

A la fin de ce chapitre, nous pouvons dire que notre objectif de l'extraction des services Web est atteint.

Conclusion générale

Conclusion et perspectives

Notre travail entre dans le domaine de l'intégration des applications de l'entreprise (EAI) dont le but est l'intégration d'une myriade d'applications hétérogènes existantes et nouvelles, qui sont vitales pour le bon déroulement des affaires et des activités, et les regrouper dans un système unique et cohérent. Pour atteindre ce but, nous avons utilisé une technologie récente qui est les services Web. Ces derniers ont prouvé leurs efficacités dans plusieurs domaines, et surtout dans le domaine de l'intégration.

Parmi les applications considérablement existantes dans l'entreprise, nous trouvons celles qui sont technologiquement en retard, puisqu'elles sont développées par des outils et des techniques traditionnelles. Ces applications, connues sous le nom de systèmes patrimoniaux (Legacy systems), sont vitales pour les affaires de l'entreprise. Donc, il est très utile d'intégrer ces systèmes dans un système EAI basé sur les services Web. Ainsi, les entreprises pourront prolonger au maximum le cycle de vie de ces systèmes, ou même réutiliser leurs fonctionnalités intéressantes pour l'implémentation des autres systèmes avancés.

Dans ce mémoire, nous avons présenté une approche de migration de type boîte grise permettant l'extraction des fonctionnalités intéressantes contenues dans des systèmes patrimoniaux et les exposer comme des services Web, dont l'objectif est que cette approche soit générique et l'intervention des experts humains dans ses étapes est minimisé au maximum. Cette approche est inspirée essentiellement des travaux de Sneed [38] et Z.Zhang et al [32,36,37]. Nous avons proposé l'intégration des expériences des utilisateurs des systèmes patrimoniaux dans les phases de l'analyse et de détection des services Web logiques. Nous avons également proposé la transformation du code source original vers un langage WSL pour unifier le langage de programmation du code source qui sera traité par les phases suivantes, afin d'obtenir un caractère de généricité, alors que cette dernière est absente dans les travaux existants.

Enfin, nous avons expliqué les différents concepts fournis, en appliquant cette approche sur une ancienne application, développée avec le paradigme procédural depuis 06 ans en utilisant le langage C standard sous Unix. Cette application est relative à la résolution de différents types de programmes linéaires. Les résultats auxquels nous avons abouti sont très importants, ce qui nous permet de dire que nos objectifs sont atteints.

Concernant les perspectives de notre travail, nous proposons d'abord l'implémentation des différents outils nécessaires pour achever les phases de notre approche. Nous citons par exemple les traducteurs de différents langages vers le langage WSL, le module permettant l'extraction automatique du code implémentant la règle métier à partir du code source du système en WSL. L'autre outil qui doit être implémenté est le module d'adaptation qui permet de lire le code source de la règle métier et génère à la fin le document WSDL décrivant cette règle pour ensuite le publier dans l'UDDI.

L'autre perspective, consiste en la substitution du WSL par le langage Python pour servir comme langage intermédiaire dans notre approche. Python est considéré comme la dernière évolution des langages de programmation; il est simple et très riche par les nouvelles techniques relatives au domaine des services Web, ce qui facilite l'implémentation des services Web avancés. Il peut être utilisé également pour le développement des solutions de l'EAI. Plusieurs travaux ont déjà confirmé que le langage Python peut être utilisé comme WSL avec succès.

Glossaire

A2A (Application to Application) : Mot tendance mode, style B to B ou B to C, désignant l'intégration des applications entre elles. On rencontre aussi A2A dans le même sens. D'une autre façon, il caractérise une communication se déroulant entre deux applications.

API (Application Programming Interface) : Une interface de code source fournie par un ordinateur ou une bibliothèque de programme en vue de supporter des requêtes pour des services qui peuvent l'employer par des programmes informatiques.

ASP (Application Server Page) : Technologie de pages web dynamiques de Microsoft. Ces pages HTML contiennent un ou plusieurs scripts qui sont exécutés dynamiquement sur le serveur Web, avant que la page soit envoyée au navigateur.

B2B (Business To Business) : Est le nom donné à l'ensemble d'architectures techniques et logicielles informatiques permettant de mettre en relation des entreprises, dans un cadre de relations clients/fournisseurs. Ceci se fait notamment au travers de l'intranet de l'entreprise est connecté aux intranets des fournisseurs, des sous-traitants, des clients, des distributeurs et des partenaires. L'objectif du B2B est la collaboration entre entreprises, en branchant en direct l'entreprise sur son environnement économique, en traçant les produits et en supervisant les opérations sur la totalité de la chaîne et en accédant à des communautés globales pour acheter ou vendre.

B2C (Business to Consumer): Ce nom donné à l'ensemble d'architectures techniques et logicielles informatiques permettant de mettre en relation des entreprises avec leurs clients, et des entreprises aux particuliers. Il fait référence en général aux affaires commerciales dirigées d'une entreprise vers une cible de particuliers via un site marchand.

BPML (Business Process Modeling Language): Est un langage basé sur XML permettant de modéliser des processus métier. BPML est une publication de BPMI (Business Process Management Initiative)**.**

CORBA (Common Object Request Broker Architecture) : Est un standard maintenu par l'Object Management Group. est une architecture logicielle, pour le développement de composants, qui sont assemblés afin de construire des applications complètes, peuvent être écrits dans des langages de programmation différent, être exécutés dans des processus séparés, voire être déployés sur des machines distinctes.

CRM (Customer Relationship Management) : Constitue un nouveau paradigme dans le domaine du marketing. Elle a pour but de créer et entretenir une relation mutuellement bénéfique entre une entreprise et ses clients. Dans ce mode de relations commerciales,

l'entreprise s'attache la fidélité du client en lui offrant une qualité de service qu'il ne trouverait pas ailleurs.

CXML (commerce XML) : Ensemble de DTD XML destinées à faciliter le commerce en normalisant les données concernant les produits. Cela devrait permettre de faciliter le commerce électronique, en aidant la mise en place de catalogues standardisés ou encore l'échange de données**.**

DCOM (Distributed Component Object Model) : Extension de COM pouvant supporter les objets répartis sur un réseau. DCOM ne supporte par contre pas de mécanisme d'héritage, ce qui fait qu'il n'a que très peu d'intérêt comme modèle objet.

DOM (Document Object Model) : Une recommandation du W3C qui décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents. Le document peut ensuite être traité et les résultats de ces traitements peuvent être réincorporés dans le document tel qu'il sera présenté.

DSML(Directory Service Markup language) : Un dialecte XML pour représenter les services d'annuaire en XML. La nouvelle version modélise les entités d'un annuaire LDAP en XML et les fonctions d'accès comme des opérations de type SOAP.

ebusiness : Ensemble des procédés basés sur les échanges de données via Internet/Intranet et le Web pour conduire les processus de commerce du monde des affaires.

ebXML : Groupe de standardisation des technologies de l'ebusiness patronnée par Oasis et EDIFACT, visant notamment à redéfinir l'EDI en XML.

EDI (Electronic Data Interchange) : Echange de données électronique organisé selon des messages à plusieurs niveaux, avec un entête de trois caractères et des codages longueur-champs, standardisé dans les années 1980.

EJB (Enterprise JavaBeans) : Une architecture de composants logiciels côté serveur pour la plateforme de développement J2EE. Elle propose un cadre pour créer des composants distribués écrit en Java hébergés au sein d'un serveur d'application.

ERP (Enterprise Resource Planning) : Est un logiciel permet de gérer l'ensemble des processus d'une entreprise, en intégrant l'ensemble des fonctions de cette dernière comme la gestion des ressources humaines, la gestion comptable et financière, l'aide à la décision, mais aussi la vente, la distribution, l'approvisionnement, le commerce électronique.

HTTP (Hypertext Transfer Protocol) : Est un protocole de communication client-serveur développé pour le World Wide Web. Le protocole HTTP peut fonctionner sur n'importe quelle connexion fiable, dans les faits on utilise le protocole TCP comme couche de transport. HTTP utilise alors le port 80.

IDL (Interface description language) : Un langage voué à la définition de l'interface de composants logiciels, laquelle permet de faire communiquer les modules implémentés dans des langages différents. IDL est défini par l'OMG et utilisé notamment dans le cadre d'applications CORBA.

Java Servlet : Une application Java qui permet de générer dynamiquement des données au sein d'un serveur HTTP. Ces données sont le plus généralement présentées au format HTML, mais elles peuvent également l'être au format XML ou tout autre format destiné aux navigateurs Web. Ce programme Java s'exécute dynamiquement sur le serveur Web et permet l'extension des fonctions de ce dernier, typiquement : accès à des bases de données, transactions d'e-commerce, etc.

JSP (Java Server Page) : Consiste en une page HTML incluant du code Java qui s'exécutera soit sur le serveur Web, soit sur le serveur d'application. Le langage HTML décrit la manière dont s'affiche la page, le code Java servant à effectuer un traitement, par exemple récupérer les informations nécessaires pour effectuer une requête dans une base de données.

OMG (Object Management Group) : Est une association américaine à but non lucratif créée en 1989 dont l'objectif est de standardiser et promouvoir le modèle objet sous toutes ses formes. L'OMG est notamment à la base des standards UML, , CORBA et IDL ...etc.

PHP : Technologie libre utilisée pour créer des pages dynamiques. Reprenant des éléments de syntaxe des langages C, Java et Perl, le code PHP est inclus dans les pages HTML pour être exécuté sur le serveur. On se sert habituellement de PHP pour extraire du contenu d'une base de données et l'afficher sur une page Web.

RDF (Resource Description Framework) : Un modèle et description de syntaxe, spécifiés par le W3C, en vue de l'utilisation de métadonnées sur le web. Le tout repose sur XML.

RMI (Remote Method Invocation) : Une interface de programmation (API) pour le langage Java qui permet d'appeler des objets distants. L'utilisation de cette API nécessite l'emploi d'un registre RMI sur la machine distante hébergeant ces objets que l'on désire appeler au niveau

duquel ils ont été enregistrés. Cette interface de programmation est très souvent utilisée en parallèle avec l'API d'annuaire JNDI ou encore avec la spécification de composants distribués transactionnels EJB du langage Java.

RPC (Remote Procedure Call) : Est un protocole permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'application. Ce protocole est utilisé dans le modèle client-serveur et permet de gérer les différents messages entre ces entités.

SAX (Simple API for XML) : Est une interface de programmation pour de nombreux langages permettant de lire et de traiter des documents XML. Il traite les documents élément par élément au fur et à mesure qu'ils sont rencontrés. Pour chaque élément (balise, commentaire, texte), la fonction de rappel correspondante est appelée.

SMTP (Simple Mail Transfer Protocol) : Protocole simple de transfert de courrier , est un protocole de communication utilisé pour transférer le courrier électronique vers les serveurs de messagerie électronique.

UML (Unified Modeling Language) : Un standard défini par l'OMG. C'est un langage graphique de modélisation des données et des traitements. Il est une formalisation très aboutie et non-propriétaire de la modélisation objet utilisée en génie logiciel. Il est l'accomplissement de la fusion des précédents langages de modélisation objet Booch, OMT, OOSE. Principalement issu des travaux de Grady Booch, James Rumbaugh et Ivar Jacobson.

workflow : Diagramme de flux proche d'un réseau de pétri permettant de décrire des enchaînements d'événements et d'actions séquentiels ou parallèles, et par extension, logiciel gérant les enchaînements et la synchronisations.

WSFL (Web service Flow Language) : Un langage basé sur XML proposé par IBM pour composer des services Web. WSFL peut être perçu comme un langage XML pour décrire des flux d'information et de contrôle, en langage Workflow.

XLang : Un langage a été proposé par Microsoft, pour composer des services Web dans un contexte transactionnel sur le Web. XLang est utilisé pour décrire les flux d'échange et les processus d'affaires dans l'EAI BizTalk.

XLink : Une spécification du W3C (appelé parfois XLL). Cette technologie permet de créer des liens entre fichiers XML ou portions de fichiers XML. Contrairement aux liens entre fichiers HTML, XLink permet de créer des liens liant plus de deux fichiers.

XMI (XML Metadata Interchange) : Un standard créé par l'OMG. C’est un standard pour l'échange d'informations de métadonnées UML basé sur XML.

XQuery (XML Query) : Un langage de requête permettant donc d'extraire des informations d'un document XML. XQuery est une spécification du W3C. Sémantiquement proche de SQL, et il utilise la syntaxe XPath pour adresser des parties spécifiques d'un document XML.

Référenes bibliographiques

[1] Atul Saini, **Dynamic Web Services**, ed : Fiorano Software, USA, 2003.

[2] **EAI, Services Web**, http://www.improve-institute.com.

[3] Manoj Seth, **Web Services A Fit for EAI**.

http://www.developer.com/tech/article.php/10923_1489501

[4] M. B. Juric, S. J. Basha, R. Leander, R. Nagappan, **Professional J2EE EAI** , ed: Wrox Press Ltd, ISBN 1-861005-44-X

[5] **LE LIVRE BLANC D'EAI**, MEDIADEV.

[6] Abraham Kang, **Enterprise application integration using J2EE**, 2002, http://www.javaworld.com/javaworld/jw-08-2002/jw-0809-eai-p1,p2p3.html

[7] David S. Linthicum, **Enterprise Application Integration**, Ed: Addison Wesley, 1999 ISBN: 0-201-61583-5.

[8] P. Shi, S.Gandhi, **Enterprise Application Integration**, viewpoint Volume 2,Number 3, 2001

[9] Georges Gardarin, **XML Des bases de données aux services Web**, DUNOD, 2002.

[10] R. Sharma, B. Stearns, T.Ng, **J2EE Connector Architecture and Enterprise Application Integration** , Publisher: Addison Wesley, First Edition December 01, 2001, ISBN: 0-201-77580-8

[11] He Guo, Chunyan Guo, Feng Chen and Hongji Yang, **Wrapping Client-Server Application to Services Web for Internet Computing**, Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05)

[12] Sumanta Deb, **Web Services: Towards Realization of the Service Oriented Enterprise**, SETLabs, Infosys February 2004

[13] Ronan Bradley, **Web Services and Application Integration: Not so simple after all**, PolarLake CEO

[14] D. Booth, H.Haas, F. McCabe, E0Newcomer, M. Champion, C. Ferris, D. Orchard, **Web Services Architecture**,W3C Working Group Note 11 February 2004

[15] R. Nagappan, R. Skoczylas,R. P. Sriganesh , Developing Java Web Services: Architecting and Developing Secure Services Web with Java , ed: WILEY, 2003.

[16] Christophe Coenraets, **Web Services: Building the Next Generation of E-Business Applications,** Macromedia JRun, October, 2nd 2001

[17] Delacrétaz Jean-Marc, **Web Services**, www.hec.unil.ch/sbenlagh/GD/Cours/Exposes2004/ WebServices.ppt

[18] S. Graham, S. Simeonov, T. Boubez, D. Davis, G. Daniels, Y. Nakamura, R. Neyama, **Building Services Web with Java - Making Sense of XML, SOAP, WSDL and UDDI** ed: Sams Publishing, 2001, ISBN : 0-672-32181-5

[19] Ethan Cerami, **Services Web Essentials** Ed: O'Reilly Anthology, February 2002, ISBN: 0-596-00224-6

[20] **Web services executive guide,** V3.0 Infravio. Inc, July, 2004.

[21] Judith M. Myerson, **Web Service Architectures,** ed: Tect, www.webservicesarchitect.com

[22] David O'Riordan, **Business Process Standards For Web Services** ed: Tect, www.webservicesarchitect.com

[23] Christoph Schittko, **Web Service Orchestration with BPEL**

[24] Boris Lublinsky, An **Introduction to Business Process Execution Language (BPEL),**

[25] Hone Guo, Xing Lin, **Research and Design on Inter-enterprise Application Integration Model**, the 9th Interational Conference on Computer Supported Cooperative Work in Design-2002, pp194-199

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

[27] John Bergey, Liam O'Brien, Dennis Smith, **Application of Options Analysis for Reengineering (OAR) in a Lead System Integrator (LSI)** Environment, March 2003

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

[29] 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.

[30] Jianjun Pu, Zhuopeng Zhang, Yang Xu and Hongji Yang, **Reusing Legacy COBOL Code with UML Collaboration Diagrams via a Wide Spectrum Language**, Proceedings of IEEE International Conference on Information Reuse and Integration (IRI'05), IEEE Systems, Man, and Cybernetics Society, 2005, pp. 78-83.

[31] 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),

[32] Zhuopeng Zhang, Hongji Yang, **Incubating Services in Legacy Systems for Architectural Migration**, Proceedings of IEEE 11th Asia-Pacific Software Engineering Conference (APSEC'04).

[33] John Bergey, Liam O'Brien, Dennis Smith, **Application of Options Analysis for Reengineering (OAR) in a Lead System Integrator (LSI) Environment**, March 2003.

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

[35] 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.

[36] Zhuopeng Zhang, Ruimin Liu and Hongji Yang, **Service Identification and Packaging in Service Oriented Reengineering**, Proceedings of IEEE 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'05).

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

[38] Harry M. Sneed, **Integrating Legacy Software into a Service Oriented Architecture**, Proceedings of IEEE Conference on Software Maintenance and Reengineering (CSMR'06).

[39] 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

[40] Richard Millham, *An Investigation:* **Reengineering Sequential Procedure-Driven Software into Object-Oriented Event-Driven Software through UML Diagrams,** Proceedings of the 26 the Annual International Computer Software and Applications Conference (COMPSAC'02).

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

[42] 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

[43] 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).

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

[45] 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).

[46] M. P. Ward, ***Reengineering from Assembler to C via FermaT Transformation,*** **Science of Computer Programming**, Special Issue on Program Transformation, Vol 52/1-3, pp 213-255, March 31, 2004.

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

[48] Monika Solanki, Antonio Cau, Hussein Zedan, **ASDL: A Wide Spectrum Language For Designing Web Services,** International World Wide Web Conference Committee (IW3C2), May 23–26, 2006, Edinburgh, Scotland.

[49] M. Ward, **Proving Program Refinements and Transformations**, Oxford University, DPhil Thesis, 1989.

[50] Ward, M.P, **Specifications from Source Code-Alchemists' Dream or Practical Reality?** 4th Reengineering Forum, September 19-21, 1994, Victoria, Canada

[51] H.A. Priestley, M.P Ward, **A Multipurpose Backtracking Algorithm** , Journal of Symbolic Computation, 18, pp 1-40, 1994.

ملخص:

في التكنولوجيات القديمة، المعطيات و طرق التسيير الموجودة في البرامج لا يمكن إعادة استعمالها في برامج أخرى، هذه الإشكالية مكنت من ظهور ميدان للبحث يهدف لدمج تطبيقات المؤسسة (EAI : Enterprise Applications Integration)، بحيث يهتم بوضع القواعد و إنشاء جميع الأدوات اللازمة لضمان الاندماج بين تطبيقات المؤسسة. لكن الحلول المقترحة لم تقض على كل المشاكل المطروحة في هذا الشأن مما جعل البحث عن حلول أخرى حاجة ملحة، ومن بين هذه الحلول نجد: خدمات الويب (Web services). نماذج الاندماج التي تعتمد على خدمات الويب مكنت من تصحيح جميع المشاكل التي عجزت عن حلها النماذج الكلاسيكية مع اقتراح نماذج جديدة بسيطة وبأسعار منخفضة وتتأقلم بسرعة مع الأنشطة المتغيرة للمؤسسات .

إن أغلبية نشاطات المؤسسات تعتمد على أنظمة معلومات قديمة والتي تعتبر موروثة من الماضي (Legacy systems)، وهي نوعا ما متخلفة تكنولوجيا عن بقية البرامج الأخرى ولكنها تحوز على قيمة اقتصادية مهمة و تعتبر كذلك حاسمة بالنسبة لأصحابها. لذلك أصبح من الضروري دمج هذا النوع من البرامج مع البرامج الأخرى وذلك باستعمال التكنولوجيات المتقدمة مثل: خدمات الويب. في هذه المذكرة اقتراحنا خطوات عملية تمكن من استخلاص خدمات الويب انطلاقا من الأنظمة الموروثة، لكي يتم إعادة استعمالها في أنظمة إدماج تطبيقات المؤسسة (EAI). وهذا الاقتراح يعتمد على قاعدة النشاط ولغة البرمجة ذات الطيف الواسع (WSL). و قد قمنا بتطبيق هذه الخطوات على برنامج يقوم بحل الأنظمة الخطية.

كلمات مفتاحيه:

خدمات الويب، دمج تطبيقات المؤسسة، الأنظمة الموروثة، قاعدة النشاط،لغة البرمجة ذات الطيف الواسع (WSL)

Résumé

Dans les technologies anciennes, les données et la logique d'affaires des applications ne peuvent pas être réutilisées dans une autre application, cette problématique donnait naissance à un domaine de recherche s'intéresse par l'intégration des applications des entreprises, dont le but est la mise en place des bases et la création tous les outils nécessaire à l'intégration des applications dans l'entreprise. Mais, ces solutions ne peuvent pas éliminer tous les problèmes posés dans ce stade. Ceci rend la recherche des autres solutions est très exigent. Parmi ces solutions nous trouvons, celles basées sur les services Web. Ces solutions ont permis de corriger les problèmes posés par l'EAI classique avec un nouveau modèle simple à prix réduit et plus adaptable aux affaires mutantes

La plupart des affaires des entreprises sont basés sur des systèmes d'information anciens, connues par les « systèmes patrimoniaux». Ces derniers sont en retard technologiques par rapport les autres systèmes mais elles possèdent une valeur économique très importante et sont cruciaux à leurs propriétaires. Donc, il est devenu nécessaire d'intégrer ces systèmes avec d'autres applications en utilisant de nouvelles technologies telles que les services Web.

Dans ce mémoire, nous proposons une approche pour l'extraction des services Web à partir des systèmes patrimoniaux, afin d'être réutilisés dans le processus de l'EAI. Cette approche est basée sur la règle Métier et le langage à gamme étendue (WSL). Nous avons l'appliqué sur un exemple relatif à la résolution des programmes linéaires.

Mots Clés

Services Web, Intégration des Applications des Entreprises (EAI), systèmes patrimoniaux, règle Métier, langage à gamme étendue (WSL).