République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mentouri – Constantine –
Faculté des sciences de l'ingénieur
Département Informatique

N° d'ordre : ……

Série : ……

Mémoire

Présenté en vue de l'obtention du diplôme de

Magister en Informatique

Option : Systèmes distribués

Un processus d'Intégration d'Applications Intra & Inter- Entreprises

Présenté par : *Melle. Soumia BENDEKKOUM.*

Dirigé par : *Pr. Mahmoud BOUFAÏDA.*

Soutenu le : 17 / 05 /2009

Devant le jury composé de:

Président :

Mohamed BENMOHAMED Professeur Université Mentouri - Constantine.

Rapporteur :

Mahmoud BOUFAÏDA Professeur Université Mentouri - Constantine.

Examinateurs :

Nadia ZEGHIB Maître de conférence Université de Constantine.

Nacereddine ZAROUR . Maître de conférence Université de Constantine.

Remerciements

« *El hamdou l'ellah* » 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. Benmohamed Mohamed, Mr. Zarour Nacereddine* et *Mme. Zeghib Nadia.* Mes remerciements vont également à *Mr. Cheikh Ahmed* l'ingénieur responsable de système d'information de l'entreprise *PMO,* pour m'avoir permis d'accéder aux informations indispensables à la réalisation de mon projet. Je remercie *Mme. Cheikh Fadila* ainsi que mon collègue *Laboudi Zakaria*, qui m'ont aidé sans hésitation. De même, je remercie tous mes collègues et mes amis (es) qui m'ont aidé de prés ou de loin.

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

Merci de mon cœur.

Dédicaces

À ceux qui font de moi que je suis,

Mes chers parents,

À mes frères et sœurs,

À mon grand père,

À mes cousins et cousines

À mon oncle et mes tentes,

À mes amis Tahani, Hadjer, Souheila, Hamza

Bahdja, Nadia, Merièm, Fayçal et Chaouki…

À Mr. et Mme Cheikh

Qui m'ont aidé énormément,

À tout mes collègues, Zaki, surtout

À toutes les personnes qui m'aiment,

Et que je n'avais pas cité,

Je dédie ce travail.

Soumeya

Université Mentouri de Constantine
Département Informatique

N° ordre : ……..

Prénom, Nom : Soumia BENDEKKOUM

Titre de la thèse : UN PROCESSUS D'INTEGRATION D'APPLICATIONS INTRA ET INTER-ENTREPRISES.

Mots clés :

Architecture Orientée Services (AOS), Intégration d'applications, Migration vers SOA, Services Web, Ingénierie inverse, Modèles d'Entreprise.

Option : Systèmes Distribués (SD)

Résumé :

Aujourd'hui, plusieurs entreprises se basent sur des systèmes d'information complexes, monolithiques, et inflexibles, parfois non conformes aux changements rapides de marché. L'Architecture Orientée Services (AOS) apparaît comme la meilleure approche permettant l'intégration flexible des applications autonomes, distribuées et hétérogènes au sein et au delà de l'entreprise. Cependant, l'analyse des différentes approches de migration des systèmes d'information d'entreprise vers un environnement SOA, montre que ces approches ne permettent pas de résoudre automatiquement la complexité des systèmes de services cibles engendrée par les systèmes patrimoniaux, et manquent de la standardisation dans leurs phases de migration. Nous proposons un processus générique de migration des Systèmes d'Information d'Entreprise vers une Architecture Orientée Services Web, basé sur les techniques de réingénierie des systèmes. Ce processus propose d'intégrer l'utilisation des modèles d'entreprise dans ses phases pour faciliter la compréhension de métier de l'entreprise, unifier la représentation de métier de l'entreprise à niveau d'abstraction plus élevé et pour réduire la complexité des systèmes orientés services cibles.

University Mentouri of Constantine
Department of Science Computing

Order N°: ……..

First Name, Last Name: **Soumia BENDEKKOUM**

Title of the thesis: A FRAMEWORK FOR INTEGRATING APPLICATIONS INTO AND ACROSS ORGANIZATIONS

Keywords:

Service Oriented Architecture (SOA), Application systems Integration, SOA migration, Web Services, Reverse Engineering, Enterprise Models

Options: Distributed Systems (DS)

Abstract :

Still today many enterprises rely on inflexible, complex monolithic systems inadequate for fast changing global markets. Service Oriented Architecture (SOA) represents the most efficient recent approach aiming at flexible integration of heterogeneous, autonomous and distributed application systems within and across organizations. But analyzing existing approaches to migrate Enterprise Information Systems (EISs) to Service Oriented Architecture environment, do not resolve automatically the complexity of the target service oriented system, most of them lack of standardisation during their phases. This thesis proposes a generic framework for migrating Enterprise Information Systems to Web Service Oriented Architecture (WSOA) based on Reverse Engineering techniques. This framework proposes to incorporate the use of Enterprise Models (EM) in migration process's phases to facilitate the understanding of business enterprise, to unify the representation of business enterprise in high abstraction level, and to reduce the complexity of the target service oriented system.

جـــامعة متنوري قسنطينـــــــة
قســـم إعلام آلي

رقم : .....

الإسـم، اللقب : سميـة بن دكـــوم

عنـوان المذكـرة : UN PROCESSUS D'INTEGRATION D'APPLICATIONS INTRA ET INTER-ENTREPRISES.

كلمات مفتاحية!:

بنية موجهة للخدمات، تكامل التطبيقات، تحويل إلى البنية الموجهة للخدمات، خدمات الويب، إعادة بناء الأنظمة، نماذج المؤسسات.

إختصاص!: الأنظمة الموزعة.

ملخـص:

لا تزال إلى يومنا هذا العديد من المؤسسات تعتمد على أنظمة معلوماتية معقدة، متمركزة وغير مرنة. وغالبا لا تتماشى مع التغير السريع للإعلام الآلي بالمؤسسات والتكنولوجيا بصفة عامة، في الأسواق.

ولذلك ظهرت البنية الموجهة للخدمات كأفضل طريقة لتحقيق تكامل و اتصال متجانس بين الأنظمة المختلفة والموزعة داخل وخارج المؤسسة. ورغم النجاح الذي تحقق بظهور هاته البنية، أظهرت دراستنا و تحليلنا للطرق المقترحة و نخص بالذكر الطرق التي تهدف إلى تحويل بنية الأنظمة المعلوماتية القديمة منها والمتطورة إلى البنية الموجهة للخدمات، أن هذه الطرق لا زالت تطرح عدة مشاكل، حيث أنها لا تتناول بطريقة آلية مشكل تعقيد وصعوبة بنية النظام الموجه للخدمات، الناتجة عن تعقيد الأنظمة المعلوماتية الموجودة بالمؤسسة. بالإضافة إلى ارتباط مراحل هذه الطرق بالشفرة المصدرية للتطبيقات.

وبالتالي، فإننا وأمام كل هذه التحديات، نقترح طريقة عامة تهدف إلى إعادة بناء الأنظمة المختلفة والموزعة، في إطار بنية نظام واحد موجه للخدمات. وتعتمد هذه الطريقة في مراحلها على استعمال نماذج المؤسسات التي تقترحها طرق إعادة بناء الأنظمة، وذلك لتسهيل فهم عمل تطبيقات المؤسسة و بالتالي التقليص بطريقة فعالة من تعقيد بنية النظام المتكامل الموجه لخدمات الويب.

Table des Matières

INTRODUCTION GENERALE

Contexte de travail…….. 1

Objectifs…….. 3

Organisation du document…….. 4

CHAPITRE I : EVOLUTION DE L'INTEGRATION D'APPLICATIONS (AI)

1. Introduction…… 6
2. Typologie d'applications d'entreprise……..... 7
3. Notion d'intégration d'applications (AI)……..... 9
4. Défis d'intégration d'applications……..... 10
5. Niveaux d'intégration…….. 11
   - 5.1 Intégration des données……... 12
   - 5.2 Intégration de l'interface utilisateurs……... 13
   - 5.3 Intégration des traitements…….. 13
   - 5.4 Intégration des processus métiers…… 13
   - 5.5 Avantages et inconvénients…… 14
6. Typologies d'intégration d'applications…….. 15
   - 6.1 Intégration intra-entreprise…….. 15
   - 6.2 Intégration inter-entreprises……. 16
   - 6.3 Business collaboration A2B…… 16
   - 6.4 Synthèse……... 17
7. Technologies d'intégration d'applications…….. 17
   - 7.1 Techniques basées sur les réseaux ad hoc……... 18
   - 7.2 Techniques de standardisation et d'unification……... 19
   - 7.3 Technique basée sur les intergiciels…… 20
   - 7.4 Techniques basées sur l'EAI……... 22
   - 7.5 Techniques basées sur le moteur BPM…… 23
   - 7.6 Techniques basées sur les ESBs…….. 25
   - 7.7 Avantages et Inconvénients……. 26
   - 7.8 Discussion…… 28
   - 7.9 Comparaison…… 29

8. Apport de l'intégration d'applications aux entreprises……... 29
9. Limitations de l'intégration d'applications……. 31
10. Synthèse du chapitre…… 32
11. Conclusion……... 34

CHAPITRE II : ARCHITECTURES ET METHODOLOGIES D'INTEGRATION

1. Introduction……. 35
2. Architectures d'intégration…….. 36
   2.1 Architecture hub-and-spoke…… 36
   2.2 Architecture en bus…….. 37
   2.3 Architecture network centric……... 38
   2.4 Synthèse……... 38
3. Solutions d'intégration d'entreprises…….…….. 39
   3.1 Remplacement « COTS Replacement »…….. 40
   3.2 Adaptation et intégration « Wrapping »…….. 40
   3.3 Migration « SOA Migration »……. 41
      3.3.1 L'Architecture Orientée Services (SOA)……. 41
      3.3.2 Les services web…….. 43
      3.3.3 La migration vers SOA…… 44
4. Différentes approches de migration…… 45
   4.1 Approche "Top Down"……..... 45
   4.2 Approche "Bottom Up"……... 46
   4.3 Approche "Outside In"…… 46
   4.4 Approche "Middle Out"……... 47
5. Migration des SIEs vers SOA……..... 47
   5.1 Migration Bottom Up…….. 48
      5.1.1 Approche de Sneed…….. 48
   5.2 Migration Meet In The Middle…… 50
      5.2.1 Approche de Smith et al…….. 51
      5.2.2 Approche de Lewis et al…….. 53
6. Synthèse……... 56
7. Conclusion……... 58

CHAPITRE III : UN PROCESSUS DE MIGRATION VERS L'ARCHITECTURE ORIENTEE SERVICES WEB

1. Introduction……. 59
2. Concepts clés …….. 60
   - 2.1 Services et granularité des services……. 60
     - 2.1.1 Décomposition des services……. 61
     - 2.1.2 Meilleur niveau de décomposition des services…….. 62
     - 2.1.3 Composition des services…… 63
   - 2.2 La Modélisation en Entreprise…… 63
     - 2.2.1 Modèles d'architecture…… 64
     - 2.2.2 Avantages des modèles d'entreprise…… 64
3. Processus proposé…… 65
4. Différentes activités de processus…….. 66
   - 4.1 Etude de contexte de migration……... 66
   - 4.2 Etude de faisabilité…….. 67
     - a. Modélisation…….. 68
     - b. Etude de faisabilité…… 70
   - 4.3 Identification des services……... 73
     - a. Spécification et raffinement……... 74
     - b. Projection des modèles et identification des services…… 74
   - 4.4 Adaptation et publication…….……..... 77
     - a. Définition de l'interface de description des services WSDL……. 78
     - b. Création de mécanisme de transport SOAP……... 78
     - c. Implémentation du code de l'interface…….. 78
     - d. Création de nouveaux services…….. 78
     - e. Enregistrement de service…….. 78
   - 4.5 Composition des services web…… 79
5. Discussion…….…….. 79
6. Conclusion……... 81

CHAPITRE IV : ETUDE DE CAS & QUELQUES ASPECTS D'IMPLEMENTATION

1. Introduction……. 82
2. Présentation de l'étude de cas……. 83
3. Objectif de l'étude de cas…… 85

4. Application du processus proposé……..... 85
4.1 Etude de faisabilité…….. 85
4.1.1 Construction des Modèles ……... 87
4.1.2 Identification des services éventuels…… 93
4.2 Identification des services…… 93
4.3 Adaptation et publication……. 97
4.3.1 Développement de l'interface et la description des services…… 97
4.3.2 Implémentation du code de l'interface……. 99
4.4 Composition des services web……. 99
5. Réalisation……... 99
5.1 Création des interfaces du code patrimoniale…….. 99
5.2 Création de code « Service_Interface »……... 101
5.3 Création de processus métier……... 102
6. Conclusion……... 104
CONCLUSION GENERALE…… 105
REFERENCES BIBLIOGRAPHIQUES……. 108

Liste des Figures

Figure I.1 : Dimensions de l'intégration d'application…… 8

Figure I.2 : D'une entreprise chaotique vers une entreprise organisée…… 10

Figure I.3 : Différents niveaux d'intégration d'applications…… 12

Figure I.4 : Typologies d'intégration…… 15

Figure I.5 : Différentes techniques d'intégration syntaxique……... 18

Figure I.6 : Principe de technique ad hoc…….. 18

Figure I.7 : Exemple d'un processus métier d'une entreprise représenté par un méta modèle UML……. 19

Figure I.8 : Principes des intergiciels …….. 20

Figure I.9 : Exemple d'un serveur d'application…….. 21

Figure I.10 : Principe des techniques d'EAI…….. 22

Figure I.11 : Architecture type de l'EAI…… 23

Figure I.12 : Principe des moteurs BPM…… 24

Figure I.13 : Architecture type d'un SGPM……... 24

Figure I.14 : Principe des techniques ESB……. 25

Figure I.15 : Architecture type d'un ESB……... 26

Figure II.1 : Typologie d'intégration Hub-and-Spoke……... 37

Figure II.2 : Typologie d'intégration en bus…….. 37

Figure II.3 : Typologie d'intégration multi-hub…… 38

Figure II.4 : Remplacement par COTS applications……. 40

Figure II.5 : Scénario de fonctionnement de l'architecture SOA…….. 42

Figure II.6 : Standards des services web…… 43

Figure II.7 : Différentes approches de migration……... 45

Figure II.8 : Récupération du code de service……... 49

Figure II.9 : Adaptation du code récupéré……. 50

Figure II.10 : Activité de l'approche MUSHUP…….. 51

Figure II.11 : Activités de processus SMART……. 53

Figure III.1 : Processus de migration des applications d'entreprise vers une architecture orientée services web…… 65

Figure III.2 : Phase de l'étude de faisabilité……... 67

Figure III.3 : Détection des services web éventuels……... 71

Figure III.4 : Phase d'identification des services web…….... 73
Figure III.5 : Définition des services web……... 77
Figure IV.1 : Processus métier de l'entreprise PMO……... 84
Figure IV.2 : Modélisation du code Basic en un arbre fonctionnel…… 91
Figure IV.3 : Modèle d'objectifs métiers de l'entreprise PMO…….. 92
Figure IV.4 : Modèle d'un processus métier élémentaire……... 94
Figure IV.5 : Création des interfaces COM en VB 6.0……... 100
Figure IV.6 : Implémentation des modules de classes qui correspondent au code patrimonial…….. 101
Figure IV.7 : Implémentation du code « Interface_Service »…….. 102
Figure IV.8 : Création d'un projet réalisant le processus de traitement des commandes de l'entreprise PMO (Moteur d'exécution)…… 102
Figure IV.9 : Interface utilisateur interagisse avec le facturier…… 103
Figure IV.10 : Appel des services web…….. 103

Liste des Tableaux

Tableau I.1 : Avantages et inconvénients…….. 14

Tableau I.2 : Avantages et inconvénients des technologies d'intégration……. 27

Tableau I.3 : Comparaison entre les différentes technologies d'intégration…….. 29

Tableau I.4 : Comparaison entre A2Ai et B2Bi……. 32

Tableau II.1 : Caractéristiques des différents modèles d'intégration…….. 39

Tableau II.2 : Bénéfices des architectures orientées services……. 42

Tableau II.3 : Différentes approches de migration vers SOA……. 57

Tableau IV.1 : Analyse des applications à migrer…… 85

Liste des listings

Listing III.1 : Modélisation du code source……..... 69

Listing IV.1 : Code des services web extraits du code de l'application …….. 96

Listing IV.2 : Document XML décrivant un service web détecté…… 97

Introduction générale

Introduction générale

Contexte de travail

Un système d'information d'une entreprise n'est pas constitué d'une seule application, il a énormément évolué surtout avec l'évolution des technologies logicielles (*objet, composant, services web*, …), l'évolution des technologies matérielles et aussi avec l'évolution des organisations (*fusion, acquisition, mondialisation*) **[6]**. Comme conséquences de tous ces facteurs et avec le temps, les systèmes d'information des entreprises deviennent de plus en plus complexes et hétérogènes. Ils se fondent sur une myriade d'applications et de systèmes d'information tels que les systèmes CRM[1], les systèmes de gestion de comptabilité et des finances, les systèmes ERP[2], les serveurs web, les systèmes patrimoniaux et bien d'autres.

Chaque application spécifique ou progiciel dans une entreprise répond à un besoin fonctionnel, mais nécessite d'une part, des informations gérées par d'autres applications et d'autre part, produit de l'information qui intéresse également d'autres applications. Pour cela, l'intégration des sources de données avec les applications dans un seul système est devenue une nécessité incontournable, c'est le but de l'EAI (***E**nterprise **A**pplication **I**ntegration*) **[33]**.

L'EAI signifie le processus d'intégrer les différents systèmes d'information, anciens ou avancés, d'une même entreprise à travers un réseau reliant les machines d'une entreprise, où il est effectué un minimum de (ou aucun) changement aux applications existantes. L'intégration d'applications d'entreprise permet de faire communiquer et collaborer les applications d'une même entreprise. Autrement dit, elle permet à une entreprise de gérer et d'étendre ses affaires inter-applicatives.

Plus récemment, l'e-Business s'envisage comme un nouveau modèle permettant de fournir des applications et des services d'une entreprise aux systèmes externes. Ainsi, les règles et les buts qui gèrent l'EAI ont été changés et sont passés à une autre meilleure, qui est le renforcement de l'EAI classique par des outils avancés pour qu'il soit capable de supporter l'exposition des applications d'une entreprise à travers le monde entier à l'aide du web.

Les anciens systèmes d'information contiennent souvent de vastes quantités de données qui ont une très grande importance d'un point de vue économique pour leurs propriétaires. La

[1] **CRM :** Consumer Relationship Management
[2] **ERP :** Enterprise Ressource Planning.

manipulation de ces données se fait par des applications dont le code dépend des règles de gestion de l'entreprise, ce qui signifie que les informations et les connaissances représentées dans le système existant peuvent constituer le noyau de l'organisation dont dépend toute son existence. En effet, le coût de maintenance de ces systèmes est très élevé et trop difficile. Les responsables des entreprises ont donc pensé à la reconstruction de nouvelles applications basées sur des technologies plus récentes. Grâce à la technologie des services web, cette solution est passée à la migration des systèmes d'information d'entreprises vers une nouvelle technologie de services web, tout en conservant le noyau de ce système. Dans le domaine de migration des systèmes quatre approches ont été définies dans différents travaux.

La première approche propose d'intégrer les systèmes d'information d'entreprises via des adaptateurs, pour que les applications puissent être invoquées comme des services web. Dans ce type de migration le système d'information de l'entreprise est vu comme un ensemble de blocs applicatifs autonomes et hétérogènes. Chaque composant applicatif est analysé et décomposé en services web réutilisables. Cette approche est intéressante, puisque elle permet la réutilisation des fonctionnalités intéressantes de l'ancien système. Cependant, elle présente un inconvénient majeur, qui est la non prise en charge des nouveaux besoins de l'entreprise **[17]**.

La deuxième approche appelée « *Top Down* ». Dans ce type de migration le système d'information de l'entreprise est vu comme un super bloc applicatif. Cette approche est descendante et globalisante dans la mesure où elle touche à la globalité de l'entreprise. Le point de départ de cette approche est la définition des processus métiers, qui représentent les objectifs fonctionnels de l'entreprise, pour descendre ensuite au travers les différentes couches du système afin de définir les services nécessaires à la réalisation de ces processus. Cette approche est très coûteuse, très longue (plusieurs années) et trop risquée, puisque elle cause la reconstruction de tout ou d'une partie de système d'information **[17].**

La troisième approche appelée « *Outside In*[3] », est une combinaison des deux approches précédentes. Cette approche vise à l'extraction des parties des systèmes existants qui ont une logique métier importante pour l'entreprise et les intégrer comme des services, tout en satisfaisant les nouveaux exigences et besoins métiers de l'entreprise. Cette approche est plus intéressante par rapport à d'autres approches, puisque l'extraction et la réutilisation des fonctionnalités importantes se font de manière aisée et guidée par les besoins métiers **[44].**

[3] **Outside In:** cette approche est appelée aussi Meet In the Middle.

La dernière approche, par opposition à l'approche *Outside In*, propose de commencer au milieu, c'est-à-dire là où le métier et les technologies d'information parlent le même langage. Elle nécessite une phase de compréhension de l'existant et des contraintes métiers, cela veut dire, la compréhension du métier de l'entreprise par les gens de métier et inversement**,** la compréhension des contraintes de la technologie de l'information (ou IT, pour Information Technology) par les gens de l'existant, et ensuite une deuxième phase d'implémentation des services et des processus métiers. Cette approche limite par contre le pilotage de la SOA[4] par les besoins métiers, puisque le point de départ est l'identification des services métiers nécessaires **[17] [44].**

Objectifs

Dans ce mémoire nous nous intéressons à l'approche de migration des Systèmes d'Information d'Entreprises vers l'Architecture Orientée Services de type « ***Meet In the Middle*** » **[44]**, qui permet de répondre aux nouveaux besoins d'intégration de l'entreprise tout en capitalisant sur l'existant. Plusieurs travaux de ce type ont été proposés, qui sont assez différents dans leurs méthodologies. Selon notre point de vue, ces travaux présentent quelques inconvénients majeurs. Le premier est lié à *la complexité de système basé sur les services* due à l'existence des services redondants et/ou des services sans ou avec une grande valeur ajoutée métier. Cette complexité est engendrée par la complexité des anciens systèmes, dont dépend toute leur existence, elle a une influence directe sur le rendement et les performances du système cible intégré. Le deuxième est lié au *manque de standardisation dans ces approches*, particulièrement dans la phase de détection du code source qui implémente le service web. Cette phase est liée intimement au langage de programmation du code source de système **[37]**, dans la mesure où il est nécessaire d'effectuer des modifications approfondies pour détecter, extraire et adapter du code comme des services web. Dans ce dernier point, *l'intervention de la supervision humaine* est nécessaire.

Pour pallier ces inconvénients, nous proposons un processus générique en adoptant l'approche « ***Meet In The Middle*** ». Notre processus offre une méthodologie stratégique basée sur les Modèles d'Entreprises (ou EM, pour Enterprise Models) **[56]** comme un langage intermédiaire de haut niveau pour la détection, l'extraction et l'intégration des fonctionnalités des anciens systèmes de différents langages de programmation, et leur réutilisation en tant que services web. L'utilisation des modèles d'entreprise est très utile pour localiser et extraire des

[4] **SOA: S**ervice **O**riented **A**rchitecture.

services ayant un niveau de granularité adéquat qui intéresse le métier de l'entreprise. Ceci permet de réduire automatiquement la complexité de système basé sur les services, et aussi de minimiser l'intervention de la supervision humaine notamment dans cette étape.

Organisation du document

Dans ce mémoire, nous avons adopté une organisation en trois grandes parties, la première est consacrée à l'état de l'art, la deuxième contient notre solution de la problématique posée, et nous terminerons par la troisième partie, où nous allons valider notre solution à travers une étude de cas.

Dans la première partie de ce mémoire, deux chapitres seront consacrés pour présenter l'état de l'art. Le premier chapitre présente une vue de littérature sur le domaine d'intégration des applications (*AI : Application Integration)* afin d'éclaircir la panoplie de concepts et de connaissances liées à ce vaste domaine, tout en exposant quelques définitions choisies. Ensuite, nous présentons les typologies d'applications d'entreprise, les approches d'intégration d'applications (intégration par données, intégration par présentation et intégration par traitement) et les différents niveaux d'intégration (A2A, B2B et B2C). Nous terminerons ce chapitre par la spécification et les caractéristiques de l'évolution des technologies d'intégration du réseaux Ad-hoc jusqu'aux ESB. Dans le deuxième chapitre, nous intéresserons aux modèles d'architectures d'intégration d'applications, puis nous exposons les différentes solutions d'intégration des applications d'entreprises proposées dans le domaine : le remplacement COTS[5], l'adaptation et l'intégration et enfin la migration vers SOA. Nous présentons les différentes approches de migration vers SOA, en focalisant notre étude sur les deux approches *Buttom up* et *Meet In The Middle*. Le deuxième chapitre se termine par une étude de quelques travaux à savoir : l'approche basée sur les MASHUPs[6], le processus SMART[7] et enfin l'approche SWA[8].

Le troisième chapitre de ce mémoire, sera consacré entièrement à la présentation de notre processus de migration des applications d'entreprises vers une architecture orientées services web (WSOA[9]). Nous exposons dans un premier temps les concepts clés utilisés dans ce processus qui sont : la granularité des services et les modèles d'entreprises. Ensuite nous

[5] **COTS: C**omponent **O**ff **T**he **S**helf.
[6] **MASHUP: M**igr**A**tion to **S**ervice **HU**rmonization **P**latform.
[7] **SMART: S**ervice-oriented **M**ogration **A**nd **R**euse **T**echnique.
[8] **SWA: S**alvaging **W**rapping **A**pproach.
[9] **WSOA: W**eb **S**ervice **O**riented **A**rchitecture.

exposons le processus global et le rôle des différentes activités constituant ce processus à part. Le troisième chapitre se termine par une discussion, qui présente les bénéfices marquant notre processus par apport aux autres approches vues dans le deuxième chapitre.

Afin de valider les idées présentées dans l'approche proposée, une étude de cas fera l'objet d'un quatrième chapitre. Dans cette étude de cas, des détails d'implémentation expliquant la solution adoptée pour réaliser la migration d'un système patrimonial vers un environnement orienté services. L'entreprise sur laquelle nous avons appliqué notre travail est **PMO** (**P**roduction **M**achines **O**utils). C'est un établissement public à caractère industriel et commercial qui s'est vu confier l'exercice du monopole de la production et la commercialisation des machines outils.

CHAPITRE I

Evolution de l'Intégration d'Applications (AI)

1. Introduction

Depuis sa naissance, l'entreprise a fait évoluer au grès de ses besoins et des exigences de ses utilisateurs, son système d'information par des développements spécifiques ou par l'intégration de progiciels répondant à des besoins fonctionnels (ERP, CRM, SCM, etc.). Le système d'information d'une même entreprise est donc en constante évolution ; il est constitué de plusieurs applications hétérogènes, qui ne sont pas toutes conformes aux standards actuels, et qui sont excessivement liées entre elles **[33]**. Aussi avec l'évolution de l'Internet, ces entreprises se regroupent, dont l'objectif principal est de s'ouvrir sur de nouveaux marchés, pour rester compétitives et de bénéficier d'un avantage concurrentiel.

Avec l'évolution rapide de l'informatique dans l'entreprise, les occasions d'arriver à une architecture chaotique (appelée architecture spaghetti) **[9]** sont de plus en plus nombreuses, que l'on parle de gestion des relations clients, d'optimisation de la chaîne logistique ou encore d'échange de données avec les fournisseurs et les collaborateurs d'affaires. Ceci provoque un éclatement de données posant des cloisons entre les différentes activités, et un éclatement des processus métiers limitant la qualité de service **[15]**, à l'heure où nous parlons du temps réel, d'une meilleure qualité de service, de système d'information agile et d'unification de l'offre de service.

Pour répondre à ces nouvelles exigences, l'entreprise se trouve alors en face de grand challenge, qu'il s'agit de faire parler entres elles de nombreuses applications distribuées, autonomes et hétérogènes. Ceci peut être dans le cadre intra-entreprise, où il s'agit principalement d'interconnecter diverses applications d'une même entreprise, ou dans le cadre inter-entreprises pour faire communiquer des applications de plusieurs partenaires. Au cœur de cette problématique se pose entre autres le problème de l'interopérabilité, qui est devenu aujourd'hui, crucial car les applications composant les systèmes d'information des entreprises ont besoin de plus en plus de travailler ensemble.

L'objectif de ce chapitre est d'explorer le domaine d'intégration des applications (ou **AI**[10]) afin d'éclaircir et synthétiser les connaissances liées à ce domaine, mais avant de ce faire il est utile d'introduire cette section par les différents types d'applications pouvant exister dans une même entreprise. Après la présentation de quelques définitions de concept d'AI, nous allons révéler les grands challenges d'un projet d'intégration d'applications d'entreprise. Ensuite, nous entamons les différents types d'intégration, et nous les suivrons

[10] AI : Application Integration.

par les niveaux de l'intégration applicative. A l'issue de cette section, nous présentons les différentes approches d'intégration qui sont l'intégration de données, d'API (Application Programming Interface), de processus métiers et l'intégration de l'interface utilisateurs. La section suivante sera consacrée à l'évolution des technologies d'intégration de réseau « ad hoc » jusqu'aux « ESB », dont laquelle nous allons exposer les différentes technologies et techniques utilisées dans les solutions d'intégration syntaxique, avec les avantages et les inconvénients de chacune. Cette étude se termine par une présentation de quelques bénéfices apportés aux entreprises grâce à l'intégration des applications.

2. Typologie d'applications d'entreprise

Les systèmes d'information et les applications dans une même entreprise sont très variés, nous distinguons dans la littérature plusieurs manières de classification des applications d'entreprise, selon le critère de classification choisi. Izza distingue les applications selon leur nature : progiciels ou applications spécifiques **[6],** ainsi que leur style architectural : application 1-tiers ; 2-tiers ou application 3-tiers. Il a aussi dans **[7]** classifié les applications selon le mode d'exécution des événements en applications batch, applications transactionnelles, applications client/serveur, applications web, applications fil de l'eau et progiciels.

Nous distinguons dans cette section, selon la nature des applications **[4] [6]** :

- **Les applications compaquetées** (*Packaged Application*) **:** ces applications contiennent un grand nombre de fonctions utiles pour le bon déroulement des activités de l'entreprise, comme : la gestion de stocks, de production, et des ressources humaines…etc. Exemple de ce type d'application : les progiciels intégrés ERP (***E**nterprise **R**essource **P**lanning*), CRM (***C**ustomer **R**elationship **M**anagement*) et les applications logistiques (*SCM : **S**upply **C**hain **M**anagement*).
- **Les bases de données patrimoniales** (L*egacy databases*) **:** ce type d'application permet la gestion et le contrôle des données critiques pour les processus d'affaires de l'entreprise.
- **Les applications spécifiques** (C*ustom applications*) **:** sont les applications spécialisées développées par l'entreprise 'sur mesure' pour satisfaire ses besoins d'affaires, en utilisant différents langages de programmation, comme : C/C++, COBOL, … etc. Les applications patrimoniales sont un exemple typique de ce genre d'applications.

- **Les programmes des transactions :** sont des programmes exécutés sur les systèmes de traitement des transactions[11].
- **Les applications Web :** sont une forme particulière d'applications transactionnelles exécutées sur des technologies Web **[6]**.

Dans le contexte de l'intégration, ces applications sont caractérisées par les propriétés suivantes (figure I.1) **[6] [18]** :

- *Autonomie* : dans la mesure où chaque application peut être conçue et exécutée indépendamment des autres.
- *Distribution* : cela signifie que les applications sont très souvent réparties physiquement sur le réseau de l'entreprise.
- *Dynamisme* : dans la mesure où les applications doivent être capables d'évoluer de façon dynamique, pour faire face aux changements d'ordre organisationnel, fonctionnel et technique que subit l'entreprise.
- *Hétérogénéité* : cela signifie que les composants autonomes peuvent être développés en utilisant des approches et des méthodologies différentes **[22]**, ce qui donne lieu à l'hétérogénéité des applications au niveau technique (les différences liées aux matériels et aux plateformes), au niveau syntaxique (les différences liées aux formats de données, aux signatures des fonctions d'une applications) et au niveau sémantique (les différences liées au sens associé aux données et aux fonctions d'une application).

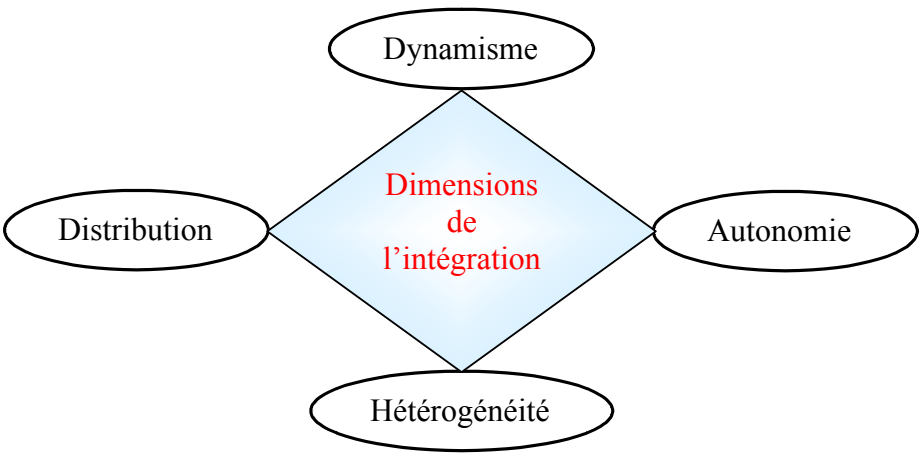


Figure I.1 : Dimensions de l'intégration d'application.

[11] Une transaction est un ensemble d'événements s'exécutant les unes après les autres.

Tous ces facteurs rendent la communication et l'interopérabilité entre les applications au sein et au delà de l'entreprise difficiles voire impossibles. L'intégration des applications est donc devenue une nécessité incontournable.

Il s'agit donc de faire tomber les barrières organisationnelles, fonctionnelles et techniques au sein de l'entreprise, afin que l'ensemble soit vu comme un tout cohérent **[13].**

3. Notion d'intégration d'applications (AI)

Il existe, dans la littérature, plusieurs définitions de l'**I**ntégration d'**A**pplications d'**E**ntreprise IAE (ou EAI[12]). Nous citons dans ce qui suit celles qui ont un effet et prennent un relief particulier dans notre travail.

L'intégration des applications d'entreprises :

Selon Linthicum, l'IAE est « ***Une approche*** dite ***stratégique*** qui permet de relier plusieurs systèmes d'informations les uns aux autres aussi bien au niveau service qu'au niveau informationnel, en permettant ainsi le partage à la fois des informations et des processus. » **[3].**

Manouvrier, de son côté propose une autre définition :

« L'intégration applicative regroupe l'ensemble des ***méthodes*** et ***outils*** organisant les échanges entre applications et les processus métiers en ***intra*** ou ***inter entreprises***. Devenue un outil ***stratégique*** pour les entreprises, elle permet une réelle réactivité du système d'information face aux évolutions métiers et par conséquent d'en optimiser fortement les coûts » **[31]**.

En ce qui nous concerne, nous considérons l'IAE comme étant à la fois la stratégie et le processus de l'entreprise permettant d'atteindre une solution d'intégration optimale des différentes applications hétérogènes, autonomes et distribuées, que ce soit dans le cadre intra-entreprise pour faire communiquer les diverses applications de son système d'information, ou dans le cadre inter-entreprises pour faire communiquer les applications de plusieurs partenaires. Ceci afin de rationaliser les échanges de données et aussi pour partager et réutiliser leurs fonctionnalités.

[12] EAI : **E**nterprise **A**pplication **I**ntegration

Il s'agit donc d'un moyen efficace pour réorganiser et structurer l'entreprise (voir figure I.2), augmenter la productivité et la réactivité des entreprises, et faciliter la communication et l'interopérabilité[13] des entreprises dans le monde.

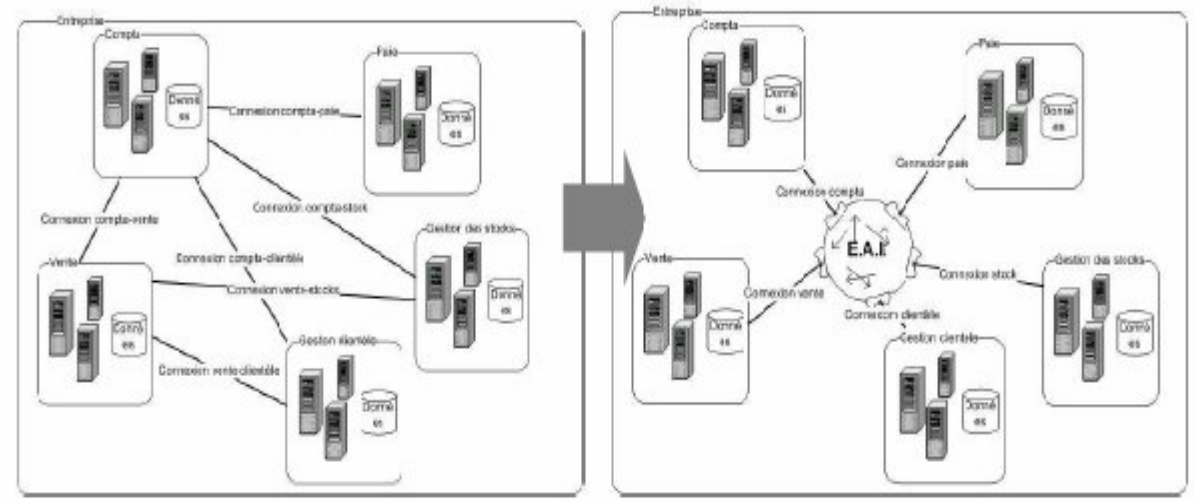


Figure I.2 : D'une entreprise chaotique vers une entreprise organisée [9].

4. Défis d'intégration des applications

L'intégration des systèmes d'information et des applications dans les entreprises est très difficile, à cause de plusieurs problèmes situés à plusieurs niveaux **[4]** :

1. 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 primitifs et n'offrent pas de supports pour l'accès transactionnel, et il existe d'autres plus avancés qui permettent un accès transactionnel aux ressources.

2. Restrictions technologiques et administratives :

Plusieurs systèmes d'information imposent des restrictions technologiques et administratives spécifiques aux utilisateurs. Par exemple l'utilisation des systèmes patrimoniaux est très compliquée (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

[13] Le problème de l'interopérabilité est au cœur de problème de l'intégration : les applications hétérogènes ont besoin de travailler ensemble et s'échanger des services entre elles.

doivent l'adapter avec les nouveaux systèmes, comme il faut les intégrer avec les différentes applications de l'entreprise.

3. Aptitude de l'intégration avec les autres systèmes :

Les systèmes d'information différent dans leurs modèles de programmation et APIs[14], ce qui rend leur intégration très difficile, parce que chacun d'eux a été développé dans son propre environnement, et qui ne peut pas être conforme avec les autres systèmes, vu qu'ils n'ont pas les mêmes objectifs.

4. Difficulté de compréhension des systèmes :

Avant que le développeur démarre l'intégration des applications, il faut d'abord qu'il comprenne tous les détails de bas niveau de programmation des systèmes d'information. Si nous avons, par exemple, un système d'information et que son API client est défini dans une bibliothèque C, et que cette dernière a défini des applications clients utilisées pour gérer les transactions et qu'elle a effectué l'accès transactionnel au système d'information, alors il faudra que le développeur étudie aussi cette bibliothèque, pour pouvoir utiliser le client API de ce système.

5. Niveaux d'intégration

L'intégration peut concerner un ou plusieurs niveaux de système d'information et des applications de l'entreprise. Linthicum définit deux niveaux d'intégration : l'intégration au niveau des données et l'intégration au niveau des traitements et processus métiers. Ce dernière type peut être à son tour représenté par le niveau d'interfaces d'applications, le niveau des méthodes et le niveau des interfaces utilisateurs **[3]**. De son côté Janarthanan et al **[14]** définit les trois niveaux d'intégration : données, interface utilisateur et méthode et considère le niveau API de **[3]** comme étant le niveau d'application. Kang et Seth montrent la nécessité de choix d'une méthode d'intégration qui peut être réalisée à travers quatre niveaux d'intégration d'applications : intégration par données, par interface utilisateurs, par applications et par méthodes **[2] [1]**.

En fait, toutes ces classifications d'intégration sont très similaires, et comme nous pouvons le remarquer, les approches d'intégration par données, intégration par interfaces utilisateurs et intégration par méthodes sont communes à toutes les taxonomies présentées ci-dessus. Ces approches sont le résultat de la structuration en couche des applications dans

[14] API : Application Programing Interface.

l'entreprise (couche données, présentation et traitement), et elles sont considérées comme des approches de base **[7].** L'intégration au niveau processus métier de la classification décrite par **[3]** et **[7]** est le résultat d'utilisation d'un moteur d'orchestration des processus métiers de l'entreprise (voir section 7.5).

Nous proposons dans ce qui suit, la taxonomie des approches d'intégration qui est définie par **[7]** et **[3],** comme le montre la figure I.3 :

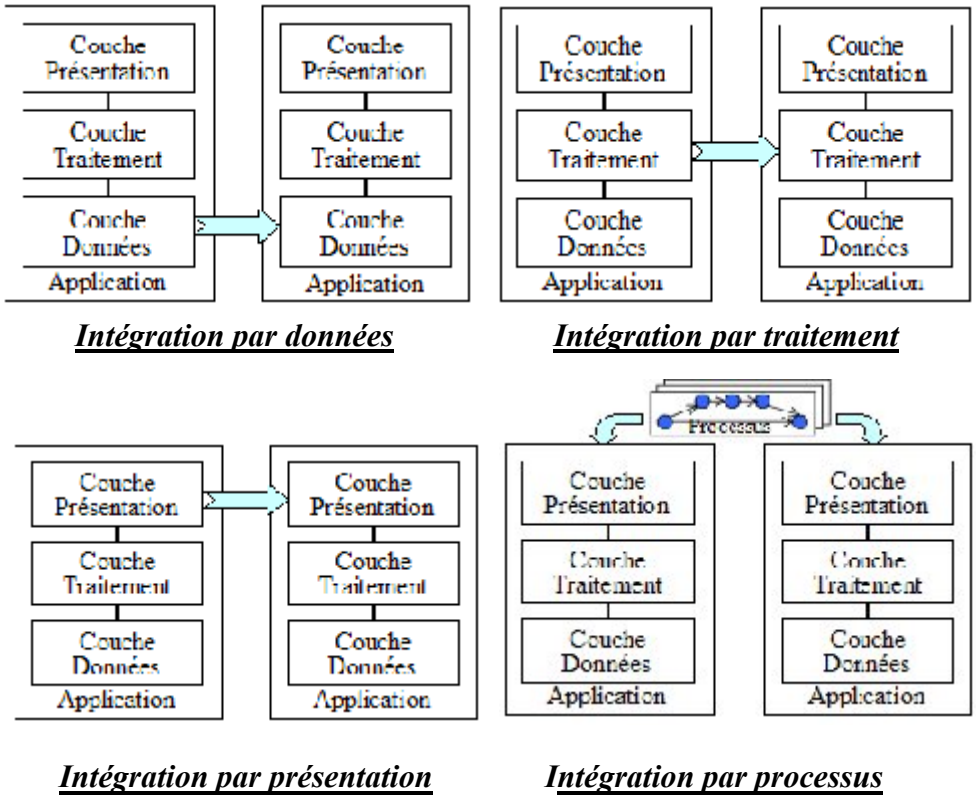


Figure I.3 : Différents niveaux d'intégration d'applications [7].

5.1 Intégration des données :

C'est la manière la plus répandue pour intégrer les applications de l'entreprise **[14] [7]**. L'intégration à ce niveau se réalise par la migration des données d'une base de données à une autre sans avoir besoin d'introduire des changements sur la logique des applications de l'entreprise. Ceci peut être décrit tant qu'extraction de l'information à partir d'une base de données, avec un traitement éventuel, la transformation et l'injection de cette information dans une autre base de données. Ce type d'intégration est utilisé dans le cas où les applications à intégrer n'ont pas une interface des utilisateurs et des APIs **[3]**.

Il existe plusieurs technologies permettant d'adresser l'intégration au niveau données qui sont **[3] [14]** : les middlewares d'accès aux bases de données (ODBC : *Open DataBase*

Connectivity, JDBC : *Java DataBase Connectivity*), les middlewares orientées messages (MOM, Message brokers), les outils de réplication de données, et les outils de fédération de données (*ETL : **E**xtract, **T**ansform and **L**oad*). (Voir section 7.3)

5.2 Intégration de l'interface utilisateur (présentation) :

Cette approche est aussi appelée screen-scrapping **[14]** qui permet de relier la logique d'intégration dans l'interface des utilisateurs **[2].** L'intégration à ce niveau se fait par le développement d'une interface commune (partagée) pour exposer les différentes applications isolées en intra et inter-entreprises **[7]**.

Un exemple de ce type d'intégration, un client qui veut s'inscrire dans un site (créer une boîte e-mail par exemple) peut accéder directement à la base de données du serveur distant et enregistrer toutes ses informations une fois qu'il clique sur le bouton « submit ». Il peut aussi extraire et échanger des données avec des applications distantes.

5.3 Intégration des traitements :

L'intégration à ce niveau permet à des applications de partager les fonctionnalités offertes par d'autres applications hétérogènes et indépendantes et cela par l'invocation des services qu'elles exposent **[2] [3]**. Le mécanisme le plus simple permettant de réaliser ce type d'intégration est l'utilisation des APIs (***A**pplication **P**rogramming **I**nterface*), ou encore plus récemment les web services (voir chapitre II) **[7]**.

5.4 Intégration du processus métier :

L'intégration des applications par processus est une approche très intéressante, elle permet le partage de la logique d'affaires, ainsi que l'intégration de nouvelles applications à travers les applications existantes. Cela peut être fait pour créer une nouvelle interface ou intégrer une nouvelle application. Cette approche est basée généralement sur la mise en place d'un moteur d'orchestration : BPI Engine ou encore un moteur de workflow : WFMS (***W**orkflow **M**anagement **S**ystem*), afin de relier et intégrer les applications dans un processus commun (voir section 7.5) **[7]**.

La réussite d'un projet d'intégration compte sur l'approche d'intégration choisie, cela ne veut pas dire que ces approches sont exclusives, elles peuvent être mises en œuvre simultanément comme dans le cas d'une entreprise où certaines applications sont intégrées par les données alors que d'autres par les traitements **[3]**.

5.5 Avantages et Inconvénients :

Après l'étude de principe de chaque approche dans quelques travaux **[1] [3] [2] [14] [7]**, nous avons voulu synthétiser dans le Tableau I.1 les avantages et les inconvénients de chacune d'elles, ainsi que leurs domaines d'application typique :

Niveau d'intégration	Avantages	Inconvénients	Applications candidates
Intégration des données	♦ Facile à comprendre. ♦ Facile et simple à réaliser. ♦ Moins coûteuse. ♦ Pas de changement de la logique d'application.	♦ Incohérence des données.[15] ♦ La logique d'affaires n'est pas intégrée.	♦ Applications spécifiques qui n'ont pas d'UI[16] et d'API.
Intégration des traitements	♦ Préserve l'intégrité des données. ♦ Permet l'intégration de la logique d'affaire. ♦ Permet la réutilisation et la distribution.	♦ Les applications sont fortement couplées. ♦ L'intégration est en mode point à point. ♦ La communication est en mode synchrone.	♦ Applications client/serveur. ♦ Les applications compaquetées
Intégration de processus métiers	♦ Préservation de la flexibilité. ♦ Médiation de la logique d'affaire permise. ♦ Approche de middleware utilisée.	♦ Manque de maturité des technologies utilisées. ♦ Complexité de réalisation. ♦ Couplage fort des applications.	♦ Applications fourni des APIs et/ou ont des méthodes fonctionnelles similaires.
Intégration des interfaces utilisateurs	♦ Moins risquée. ♦ Moins coûteuse. ♦ Disponibilité et stabilité des technologies utilisées. ♦ Facilité et simplicité de réalisation.	♦ Pas d'interaction entre données et applications. ♦ Ajout de nouvelles applications difficile. ♦ Communication synchrone bloquante.	♦ Applications patrimoniales. ♦ Applications client/serveur. ♦ Application web

Tableau I.1 : Avantages et inconvénients.

[15] Les données sont stockées dans des entrepôts de différents vendeurs.
[16] UI : User Interface, en français : Interface Utilisateur.

6. Typologies d'intégration d'applications

Le domaine de l'intégration d'applications traite les échanges entre les applications. Cependant, il ne se limite pas qu'à la communication à l'intérieur de l'entreprise mais aussi à l'extérieur, dont lequel l'entreprise essaye d'obtenir des avantages concurrentiels, il s'agit principalement des deux champs d'intégration **[6]** et **[7]** : l'intégration intra-entreprise, et l'intégration inter-entreprises. Linthicum **[19]** et Gomez **[39]** distinguent aussi deux formes d'intégration : intégration d'applications internes (EAI)[17] et l'intégration d'applications externes (*Business to Business integration*)[18] (voir figure I.4).

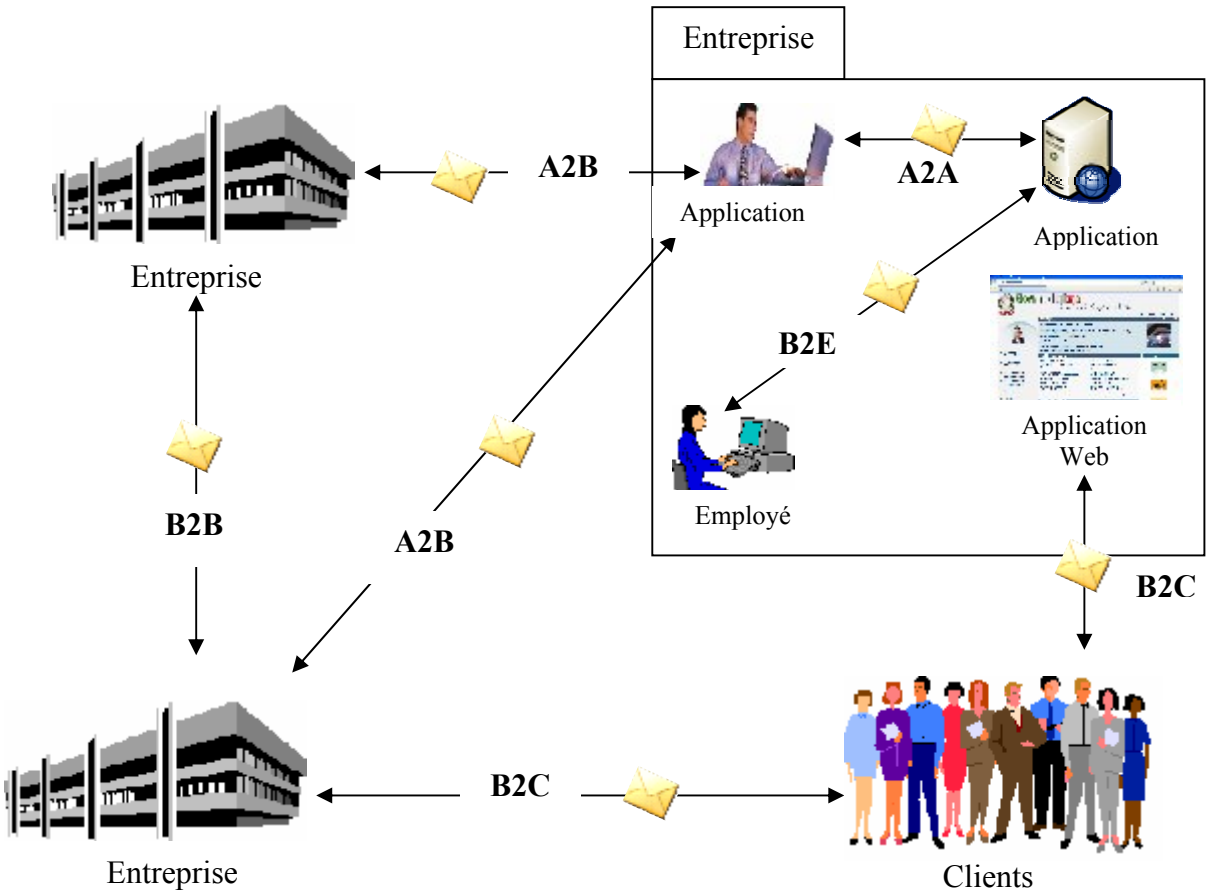


Figure I.4 : Typologies d'intégration.

6.1. Intégration intra-entreprise :

Concerne les scénarios d'intégration et d'échange d'informations entre les différents systèmes au sein d'une même entreprise **[6] [39],** elle est généralement notée par l'intégration A2A (*Application to Application integration*). Dans ce type d'interaction, nous distinguons

[17] Appelée aussi intégration horizontale.
[18] Appelée aussi intégration verticale.

aussi ***l'intégration B2E*** (***B**usiness to **E**mployee **I**ntegration*) pour désigner la relation entre une entreprise et ses employés, notamment via la mise à disposition de formulaires à leur attention pour la gestion de leur carrière, de leurs congés ou de leurs relations avec le comité d'entreprise.

6.2. Intégration inter-entreprise :

Désignée par l'intégration B2B (***B**usiness to **B**usiness integration*), implique l'intégration des applications appartenant à des entreprises différentes **[6]**, elle supporte la communication et la collaboration entre les clients externes, les partenaires, les fournisseurs, les sous traitants, et les collaborateurs d'affaires. Exemple de ce type d'intégration les chaînes logistiques[19] et l'approvisionnement électronique (e-Procurement[20]) [**19]**.

L'intégration des clients, appelée aussi ***l'intégration B2C*** (***B**usiness to **C**onsumer*), est aussi incluse dans ce niveau. Dans cette catégorie « Affaire à Client » nous trouvons toutes les formes de communication directe entre une entreprise et ses clients. L'idée fondamentale est de permettre aux clients d'accéder à des services et/ou à des informations depuis chez eux.

Les clients peuvent trouver toute l'information dont ils ont besoin sur Internet, poser des questions et recevoir des réponses, acheter des produits ou payer les factures, etc.

L'adoption de l'entreprise d'une stratégie e-Business signifie qu'elle doit fusionner avec d'autres entreprises et réorganiser sa structure interne en adaptant de nouvelles technologies et plateformes pour contrôler les issues d'achat, d'approvisionnement, et de fournisseurs, ainsi que pour gérer les rapports de client et lui fournir de services **[4].**

6.3. Business collaboration (A2B) :

La collaboration des affaires est le couplage entre A2A et B2B, qui désigne une relation entre une entreprise et les administrations publiques (administration fiscale, etc.) s'appuyant sur des mécanismes d'échange numériques (téléprocédures, formulaires électroniques, etc.).

[19] La chaîne logistique représente l'ensemble des maillons relatifs à la logistique d'approvisionnement : achat, approvisionnement, gestion des stocks, transport, manutention.
[20] Pour electronic procurement : l'entreprise achète de grandes quantités de fournitures de fabrication, et de matériels directement de différents fournisseurs en utilisant des moyens électroniques.

6.4. Synthèse :

D'après cette section, nous remarquons que le périmètre couvert par les échanges et la communication entre applications, influe sur le choix de l'approche utilisée pour l'intégration : comme, par exemple, dans le cas de l'intégration A2A l'approche par processus est souvent utilisée, ce qui n'est pas le cas pour l'intégration B2B dans laquelle l'approche par traitement est utilisée et cela du fait que l'entreprise dans le deuxième cas n'a aucune visibilité de processus principal. Pour l'intégration B2C l'approche d'intégration par l'interface utilisateur est l'approche la plus adéquate. L'approche d'intégration par les données est très simple et intuitive, elle est utilisée quelque soit le type d'interaction entre les intervenants de la chaîne de l'entreprise.

7. Technologies d'intégration d'applications

L'intégration des applications constitue un domaine prometteur en constante évolution **[7]**. Il existe principalement sur le marché plusieurs techniques d'intégration qui permettent d'adresser l'intégration syntaxique, et qui peuvent être mise en œuvre au sein des entreprises ou plus communément au sein des organisations.

Ces technologies sont aussi diverses que variées, nous pouvons dans cette partie distinguer six grandes classes, qui sont selon **[6]** : les techniques réseau ad-hoc de communication ; les techniques d'unification et de standardisation ; les techniques basées sur les intergiciels de communication (*Middleware*) ; les techniques basées sur l'EAI (***E**nterprise **A**pplication **I**ntegration*); les techniques basées sur le moteur BPM (***B**usiness **P**rocess **M**anagement*) ; et les techniques basées sur l'ESB (***E**nterprise **S**ervice **B**us*).

La figure I.5 est un méta modèle UML permettant de résumer les principales technologies d'intégration syntaxique. Le méta modèle est un diagramme de classe. Chaque classe représente une catégorie de technique.

Après avoir présenté le principe de chaque technologie, nous allons terminer cette section par une synthèse des avantages et des inconvénients de chacune des technologies étudiées, avec une comparaison entre elles.

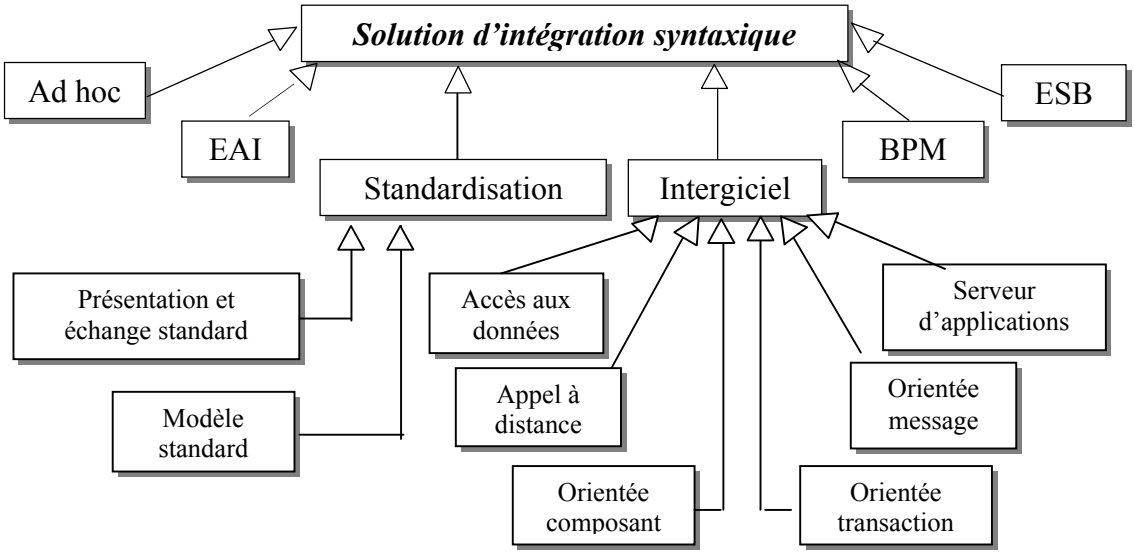


Figure I.5 : Différentes techniques d'intégration syntaxique [6].

7.1 Techniques basées sur les réseaux ad hoc :

Ces techniques sont utilisées, dont le besoin principal, est de faire communiquer les différentes applications qui envahissent l'entreprise pour : supprimer les saisies multiples, consolider les informations, et réduire les tâches administratives **[20]**.

Ces technologies se basent principalement sur certains *middlewares* invoqués de manière programmatique pour définir des passerelles entre les applications **[6]**, comme le *RPC (Remote Procedure Call)* qui consiste à mettre en œuvre des appels à distance entre les procédures des applications. Dans cette approche, les applications sont connectées directement les unes aux autres, et communiquent en face à face en n'utilisant pas de structures évoluées d'intégration intermédiaires, ce qui donne lieu à une architecture accidentelle difficile à maintenir. **[6]** (voir figure I.6)

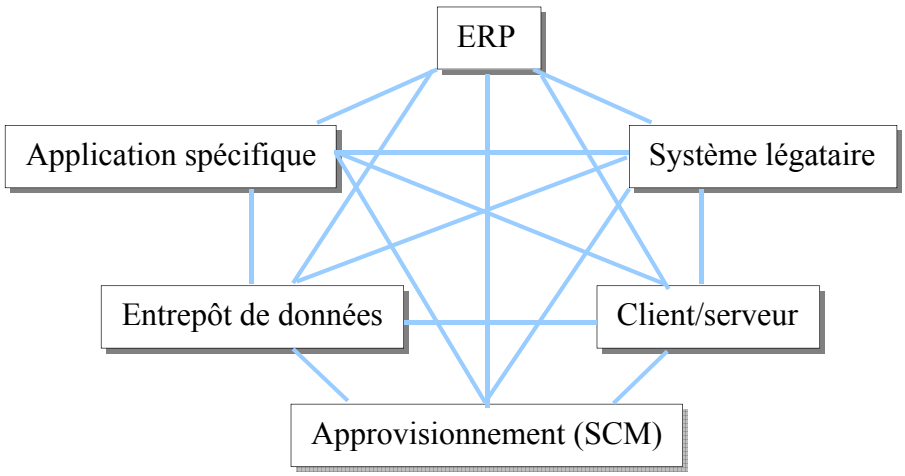


Figure I.6 : Principe de technique ad hoc [6].

7.2 Techniques d'unification et de standardisation :

Dans l'entreprise, on utilise une multitude de langages qui permettent de représenter les modèles de données[21], les modèles de processus[22], ou encore d'échange de données. Une des solutions très simple pour réaliser l'intégration entre applications, est d'unifier la manière de présenter et d'échanger les données en utilisant des standards de représentation **[6]** tels que : UML (***U**nified **M**odeling **L**anguage)* (voir figure I.7) *et* XSD (***X**ML **S**chema **D**efinition*) qui sont généralement utilisés pour décrire les schémas de données et des bases de données, EDI (***E**xchange **D**ata **I**nterface*), XML (*e**X**tencible **M**arkup Language*) et ebXML (***e-b**usiness **XML***) pour l'échange de données intra et inter-entreprises**.** Des standards des modèles de processus et des applications sont aussi utilisés, comme : UEML (***U**nified **E**nterprise **M**odelling **L**anguage*) qui est un langage de modélisation d'entreprise **[13]** et BPML (***B**usiness **P**rocess **M**odeling **L**anguage*) qui est un langage unifié de modélisation de processus métiers de l'entreprise **[6]**.

Les techniques de standardisation offrent, en effet, un moyen d'entente aux partenaires, qui présentent leurs entreprises selon des formalismes de modélisation incompatibles et qui ne couvrent pas les mêmes vues, comme : CIMOSA **[23]**, GRAI, etc.

Ces techniques permettent aussi de standardiser les outils, méthodes et stratégies pour faciliter l'intégration optimale des applications hétérogènes. Cependant elles n'ont pas réussi à s'imposer à grande échelle, car l'échange standardisé s'effectue sur un mode point à point **[15]**.

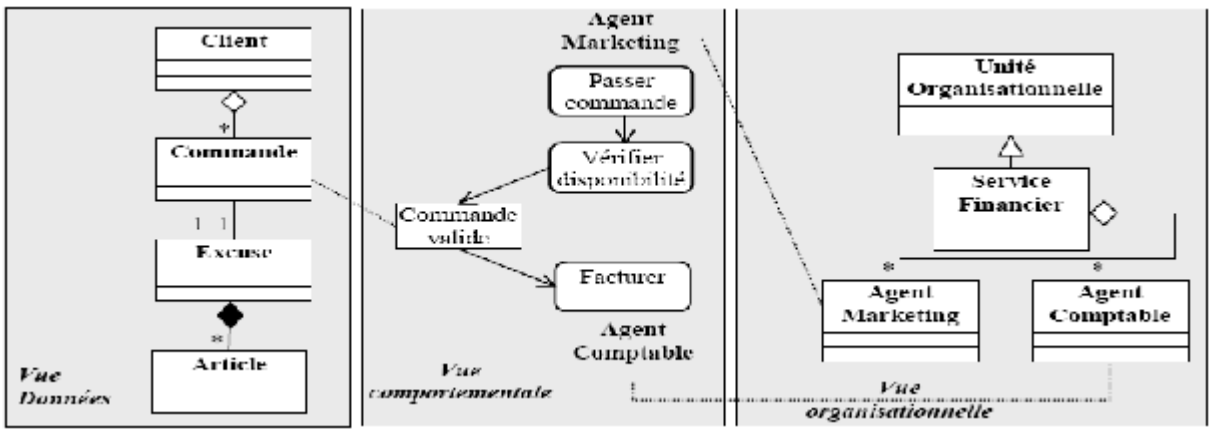


Figure I.7 : Exemple d'un processus métier d'une entreprise représenté par un méta modèle UML [13].

[21] Permettent de donner une représentation très souvent graphique de la réalité à travers principalement l'utilisation des entités, des attributs et des relations.

[22] Permettent de décrire les différents éléments d'un processus métier à travers principalement la notion d'activité, d'événement et de rôle.

La figure I.7 illustre comment UML est utilisé pour modéliser les processus métiers. Il permet de lier les différentes vues (données, comportementale et organisationnelle) entres elles dans un seul diagramme en intégrant les diagrammes UML entres eux (activité, classe, séquence, etc.) **[13].**

7.3 Techniques basées sur les intergiciels :

Les inconvénients des techniques de communication point à point et le besoin des entreprises à améliorer leurs réactivités sur le marché, ont poussé l'entreprise à utiliser des systèmes intermédiaires appelés : *intergiciels* **[20]**. Les intergiciels (en Anglais middleware) sont des logiciels qui jouent le rôle d'un intermédiaire dédié pour la gestion des communications inter applications. Ils se chargent de la communication, du routage des messages entre les applications, de la transformation des données, la gestion des transactions, et le partage des connexions **[3].**

La figure I.8 illustre le principe des intergiciels, qui permettent de masquer toute l'hétérogénéité technique liée aux plateformes (langage de programmation, réseau, système d'exploitation) **[6].**

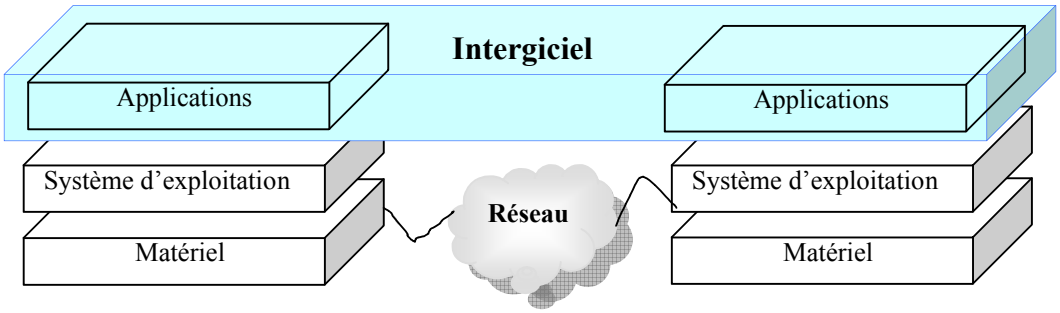


Figure I.8 : Principe des intergiciels [6].

La notion de middleware est aujourd'hui plus évoluée, elle inclue d'autres services de haut niveau, tels que la gestion des processus, la sécurité, le management des échanges (administration et exploitation), la qualité de services (performances, disponibilité, etc.) **[3]**. Il existe plusieurs types de middlewares, parmi lesquels nous citons **[3] :**

- *Les middlewares d'accès aux bases de données :* permettant d'accéder à des données de manière transparente, quelque soit le type de la source de données (DB2, Oracle, …), exemple : ODBC et JDBC.

- *Les middlewares d'appel de procédures à distance :* permettant d'accéder et d'exécuter des programmes (procédures) sur un site distant à travers le réseau de manière transparente, en utilisant une communication synchrone.

- *Les middlewares orientés messages :* permettant l'échange des messages en utilisant une communication asynchrone.

- *Les middlewares orientés composants :* ce type d'intergiciel offre une solution d'intégration distribuée, en exploitant les objets distribués[23] et le principe d'appel à des méthodes distantes. Un exemple de ce type : CORBA (***C**ommon **O**bject **R**equest **B**roker **A**rchitecture*)[24] et COM/DCOM (***C**omponent **O**bject **M**odel*)[25].

- *Les middlewares orientés transactions :* permettent principalement de gérer les accès aux bases de données de façon à garantir l'intégrité des transactions qui y sont effectuées**,** donc à assurer la consistance des données partagées.

- *Les serveurs d'applications :* ce type d'intergiciel fournit un ensemble de services d'exécution aux composants déployés (applications). Ces services d'exécution sont cachés dans des composants d'applications du modèle de programmation simplifié. Les services fournis incluent : le support de transaction, le mécanisme de sécurité, l'accès aux bases de données, la messagerie asynchrone et la communication distribuée **[4].**

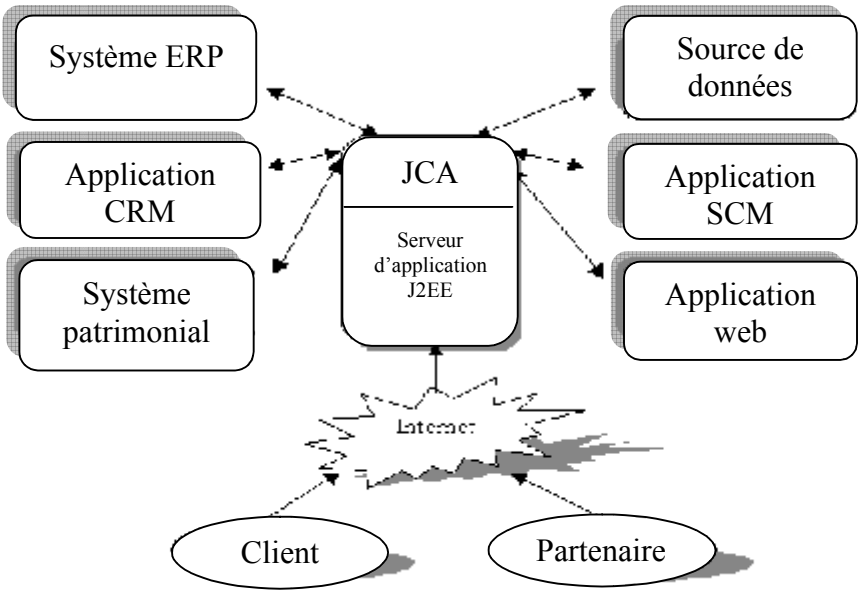


Figure I.9 : Exemple d'un serveur d'application [4].

[23] Les objets distribués sont des objets qui peuvent résider n'importe où sur un réseau et continuer à exister en tant qu'entités autonomes physiques tandis qu'accessibles à distance par d'autres objets.
[24] CORBA : une architecture proposée par l'OMG (Object Management Group).
[25] COM/DCOM : des modèles basés composants introduits par Microsoft en 1995.

Un exemple d'un serveur d'applications la plateforme ***Java 2 Edition Entreprise*** (J2EE)[26] (voir figure I.9), qui utilise les connecteurs JCA (***Java Connector Architecture***) pour se connecter aux applications distribuées **[4].**

7.4 Techniques basées sur l'EAI :

Ce sont des techniques où chaque application se connecte de manière indépendante et unique à un « système central », sans avoir de connaissances à priori de typologie de l'application ou du système destinataire (voir figure I.10) **[21]**. Ce système est à l'origine, un progiciel EAI (***Enterprise Application Integration***), qui joue le rôle d'un médiateur. Il est responsable de plusieurs fonctions : interfacer les applications généralement par l'utilisation des connecteurs propriétaires difficilement maintenables, transformer les messages à un format pivot pour le "mapper" ensuite au format propriétaire attendu par les applications, rediriger (Routing) le message vers une application destinataire en fonction de son contexte, et contrôler l'exécution de processus métiers d'un système d'information de l'entreprise **[17]**.

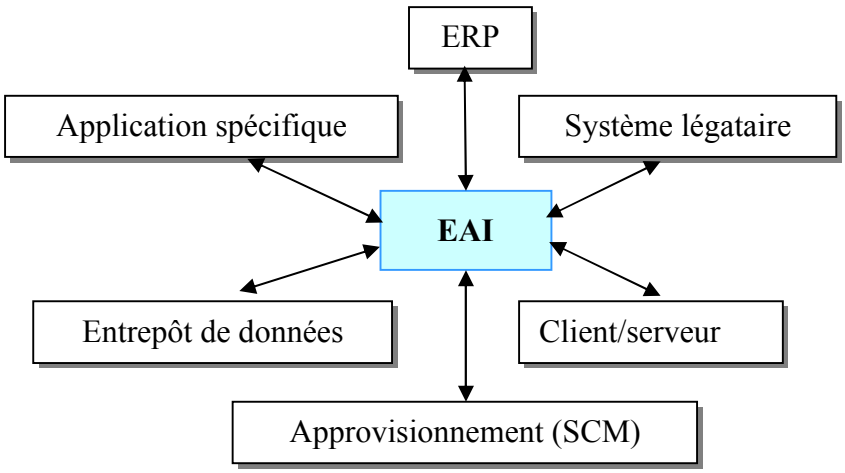


Figure I.10 : Principe des techniques d'EAI [6].

L'architecture type de l'EAI permettant d'assurer ces fonctionnalités, est représentée par l'architecture en couche suivante (voir figure I.11), où chaque couche est responsable d'un service d'intégration :

- *La couche connexion des connecteurs* **:** assure le transport des messages depuis l'EAI aux applications et vice versa, c'est-à-dire, l'extraction depuis les applications productrices ou la restitution aux applications consommatrices **[3]**. L'EAI utilise typiquement

[26] J2EE : un environnement de développement et de déploiement des applications rédigées en langage de programmation JAVA.

MOM (***M**essage **O**riented **M**iddleware*) comme un moyen fondamental de transport de messages entre les différentes applications et la couche de transformation et de routage, et le HTTP, SOAP, SMTP, IIOP et RMI comme protocole de transport **[3]**.

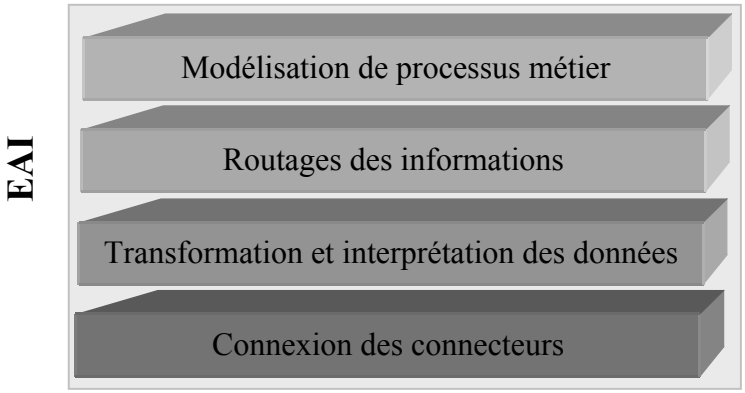


Figure I.11 : Architecture type de l'EAI. [13]

- *La couche transformation et interprétation des données* : cette couche fournit des services permettant la conversion des données d'un format à un autre, par exemple le moteur XSL qui effectue des transformations des messages XML **[13].**

- *La couche de routage des informations* : fournit des moteurs d'intégration pouvant déterminer le routage des messages intelligemment basé sur des règles prédéfinies et envoyer les messages à la cible nécessaire **[17].**

- *La couche modélisation de processus métier* : fournit les outils nécessaires à la modélisation des processus métiers sous une forme permettant leurs automatisations, en organisant les enchaînements des tâches entre les applications **[33]**.

7.5 Techniques basées sur le BPM :

Les outils basés sur la gestion de processus métiers (GPM[27]) ne sont pas nouveaux. Ils sont déjà utilisés et vus incorporés dans les outils EAI. Ceci ne veut pas dire que ces derniers ne peuvent pas être considérés comme des outils d'intégration à part entière **[7]**.

Les techniques de GPM sont des solutions basées sur la notion de processus métier (voir chapitre III). Elles ont été proposées pour répondre à la fois à un besoin fort des entreprises de bien formaliser, gérer et contrôler le processus métier de leurs systèmes d'information dans leur globalité, ainsi qu'un besoin d'intégration avec les divers composants et progiciels, qui

[27] GPM : la Gestion de Processus Métier traduction de Business Process Management (BPM).

sont utilisés tout au long du cycle de vie d'un produit, d'une relation avec un client ou d'une relation avec un fournisseur **[27]**.

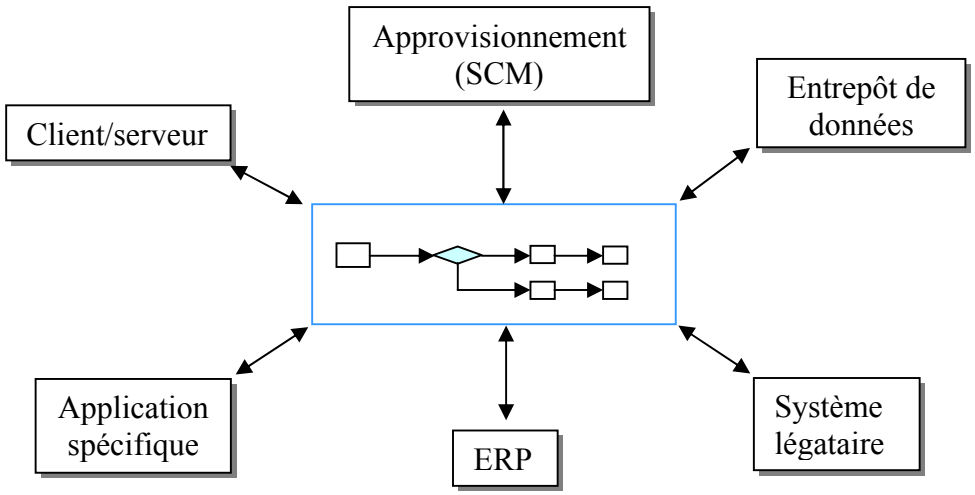


Figure I.12 : Principe des moteurs BPM [6].

La gestion de Processus Métier (PM[28]) nécessite la collaboration d'un ensemble d'outils composant la solution de BPM tel qu'il est présenté dans la figure I.13. Nous décrivons dans ce qui suit l'architecture type d'un Système de Gestion de Processus Métier (SGPM) : **[35]**

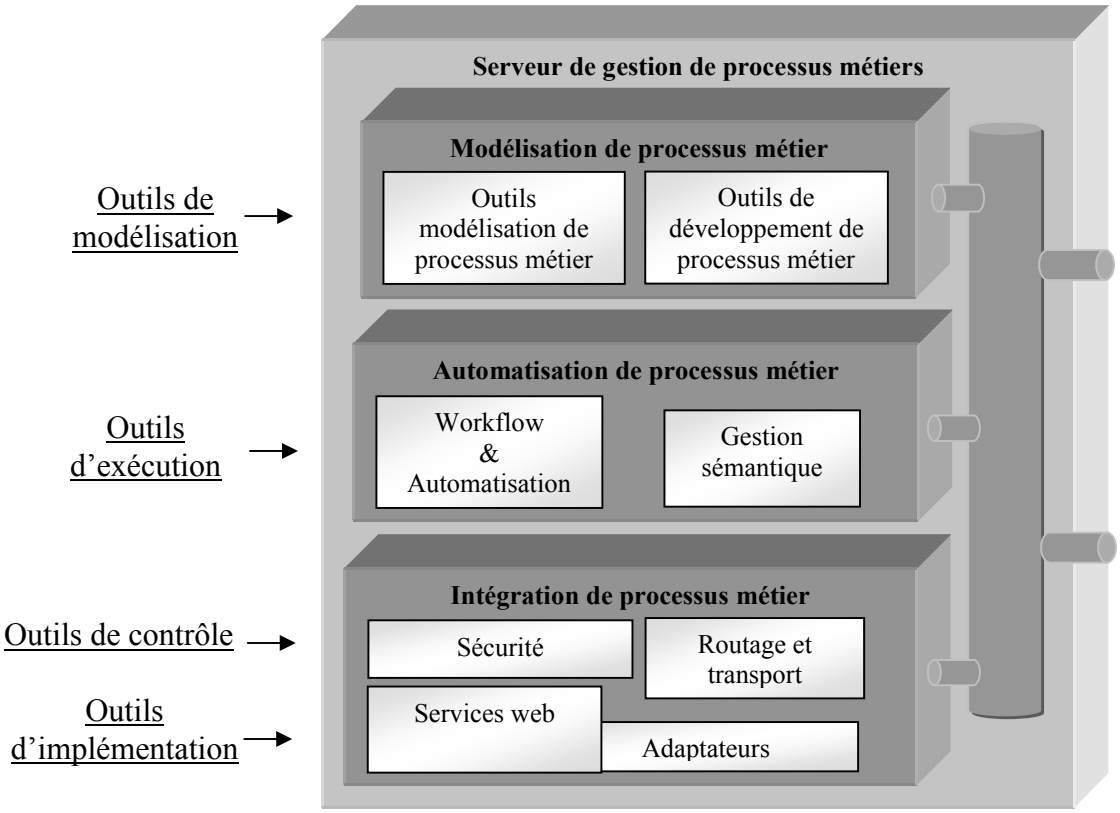


Figure I.13 : Architecture type d'un SGPM [35].

[28] PM : processus métier en anglais : Business Process (BP).

- Les outils de modélisation de processus, permettent de modéliser à l'aide d'une interface graphique les processus métiers de l'entreprise.

- Les outils d'automatisation (moteur d'exécution) chargés d'instancier le processus et de le gérer, valider et remonter ses activités, en éliminant le plus possible les étapes manuelles et les interruptions de processus.

- Des outils de pilotage et de contrôle basés sur des indicateurs précis et pertinents afin de disposer de tableaux de bord permettant de prendre rapidement les bonnes décisions. On parle ainsi de BAM (***B**usiness **A**ctivity **M**onitoring*) pour désigner la notion de contrôle du déroulement des processus de l'entreprise.

- Des outils d'aide à l'implémentation, c'est-à-dire des interfaces (API) et des connecteurs qui permettent d'intégrer la solution de BPM au système d'information.

7.6 Techniques basées sur l'ESB :

Les faiblesses des solutions basées sur les outils d'EAI[29] **[20]**, ainsi que l'émergence de nouvelles approches orientées services comme les services web et SOA (voir chapitre II), ont modifiés le paysage de l'intégration **[21].** De nouvelles technologies appelées l'ESB (*Enterprise Service Bus* : en français *Bus de services d'entreprise*) ont été défini en 2003 par le Gartner Group, dont l'objectif principal est de définir un nouveau type de middleware d'intégration d'applications **[5].** Comme le présente la figure (voir figure I.14) la technologie ESB est centrée sur la notion de bus, qui permet d'assurer une intégration distribuée des différents services métiers connectés (les fournisseurs et les consommateurs de services) **[13].**

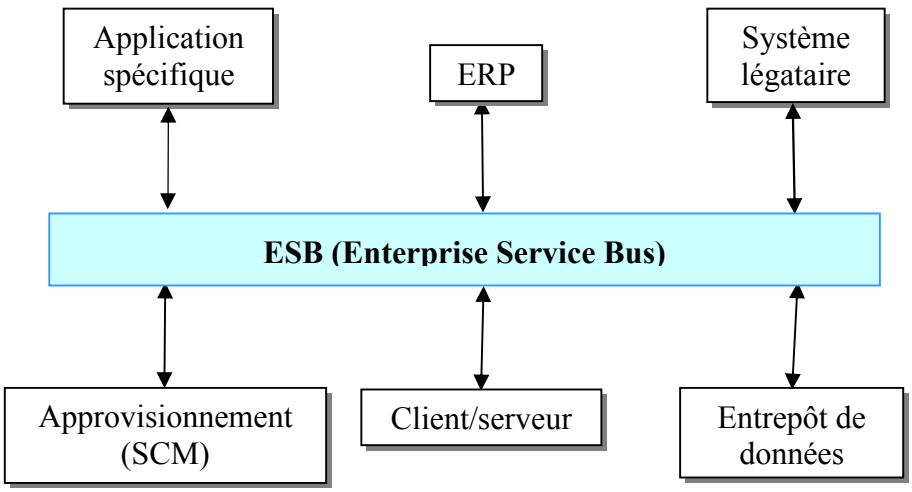


Figure I.14 : Principe des techniques ESB [6].

[29] Les faiblesses liées à leur caractère propriétaire et au problème de centralisation des traitements "single point of failure".

« L'ESB est une nouvelle architecture qui exploite les services web et les normes des services web dite « WS.* [30] », le middleware de la messagerie, le routage intelligent et transformation. Les ESBs agissent comme des composants légers, intégration omniprésente par laquelle les services logiciels et les composants d'application émergent » **[13]** (voir figure I.15).

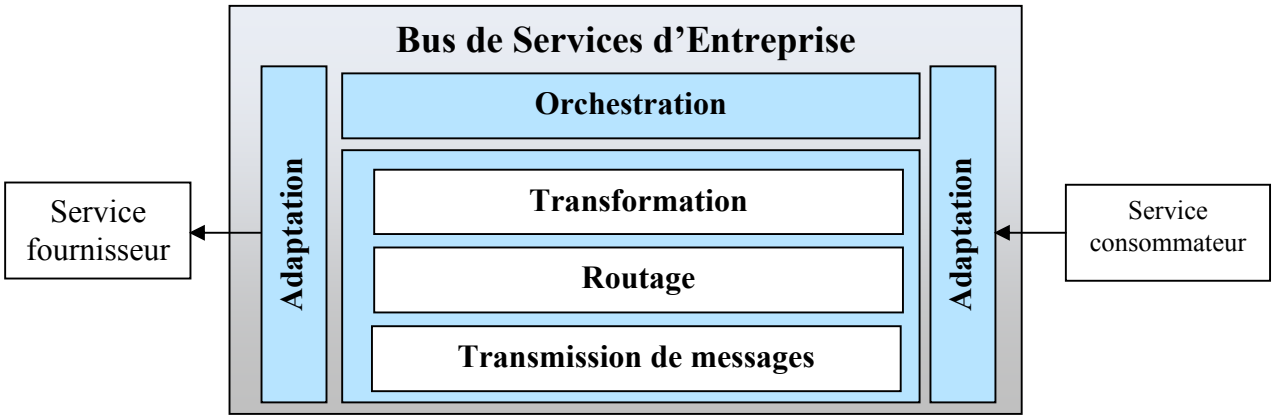


Figure I.15 : Architecture type d'un ESB [13].

La figure I.15 représente les principales fonctionnalités d'un ESB, et les composants permettant de les fournir : **[34]**

- *Transmission de messages :* une demande d'un service (message) est transmise en mode asynchrone au fournisseur de service.
- *Routage* : permet d'envoyer la demande de service au fournisseur de service nécessaire, en utilisant des règles prédéfinies.
- *Transformation* : permet de transformer le format de message à un autre format.
- *Adaptateur* : généralement dans l'ESB les applications utilisent le standard SOAP pour l'échange de messages (un format standard). Cependant, il existe des applications qui ne supportent pas le protocole SOAP, ceci nécessite l'utilisation des adaptateurs pour transformer le message.
- *Orchestration :* un moteur d'orchestration permet de gérer et contrôler les flux de contrôle d'un service à un autre, constituant les processus métiers de l'entreprise.

7.7 Avantages et inconvénients :

L'analyse de quelques travaux **[20] [21] [32]** et **[13],** nous a permet de résumer les avantages et les inconvénients de chacune de ces technologies dans le tableau suivant :

[30] Les normes des services web dite « WS.* » : un ensemble de normes associées aux web services et traitent par exemple la sécurité ou l'orchestration des services.

Technologie	Avantages	Inconvénients
Ad hoc	♦ Facile à mettre en œuvre. ♦ Développement sur mesure (interfaces personnalisées). ♦ Simple à comprendre.	♦ Redondance de données. ♦ Coût de développement très élevé. ♦ Non fiabilité de données. ♦ Difficile à maintenir et à faire évoluer. ♦ Complexité surtout dans le cas où le nombre d'application à intégrer est grand.
Standardisation	• Une grande capacité d'ouverture.	♦ Très coûteuse en termes d'investissement afin d'adopter les standards proposés. ♦ Communication en mode point à point
intergiciel (middleware)	♦ Disponibilité et fiabilité des données. (pas de redondance) ♦ Réduction de coût d'implémentation.	♦ Lourdeur et lenteur de mise en œuvre et difficulté de gestion. ♦ Portabilité des applications non assurée. ♦ Des solutions propriétaires
EAI	♦ Relie à moindre coût les applications internes et externes. ♦ Favorise l'intégration fonctionnelle plutôt que l'intégration de donnée. ♦ Favorise la réutilisation.	♦ Trop lent à mettre en œuvre. ♦ Moins flexible (caractère propriétaire de protocole, connecteurs, format de données…). ♦ Intégration centralisée (introduit un SPOF[31]).
BPM	♦ Une visibilité sur l'ensemble des processus métiers ♦ Facilité de l'optimisation de processus métier.	♦ Adaptation semi automatique des modèles de processus aux modèles d'exécution et difficile et peut entraîner des erreurs.

[31] SPOF (Single Point Of Failure) : un point unique par lequel passe toutes les traitements ce que paralyse les systèmes en cas de panne.

ESB	♦ Adaptation rapide aux différentes architectures. ♦ Grande capacités d'ouverture. ♦ Basée standards donc plus agile. ♦ Grande visibilité sur les processus métiers internes et externes.	♦ Une gestion centralisée donc ne préserve pas la scalabilité du système (c'est-à-dire la montée en charge).

Tableau I.2 : Avantages et inconvénients des technologies d'intégration.

7.8 Discussion :

L'évolution de l'informatique dans l'entreprise a permet l'utilisation des technologies plus récentes. Nous discutons dans cette section quelques points importants sur l'évolution des technologies d'intégration sur le marché.

Plusieurs travaux **[20], [6], [22]** et **[45]** considèrent les progiciels de gestion intégrés (PGI), dotés de nombreuses fonctionnalités qui supportent les besoins des utilisateurs et la diversité des applications d'entreprise, comme étant des techniques permettant d'offrir une intégration complète. Cependant ils n'ont aucun avenir car l'émergence de ces techniques (PGI) n'a fait qu'amplifier la problématique d'intégration des applications, du fait qu'elles n'offrent que des solutions centralisées (création des silos d'information donc le retour au problème de l'intégration des PGI), souvent partielles et limitées dans leurs périmètres de déploiement **[6] [17]**.

Les techniques basées sur les outils d'EAI offrent une solution purement technique, il s'agit en effet, d'assemblage d'une multitude d'intergiciels (pour le transport et la connectivité), des outils ETL (pour la transformation et le routage) et des moteurs de workflow ou des outils PBM (pour la gestion de processus métier de l'entreprise) **[8]**.

Ainsi les outils d'ESB n'inventent aucune technologie et se contentent principalement d'assembler des solutions existantes fiables et éprouvées **[12],** et d'utiliser des standards ce qui est réellement montrer dans la section 7.6 de ce chapitre.

7.9 Comparaison :

Après la présentation des principes de chacune des technologies, nous avons choisi de faire une comparaison entre les technologies d'intégration basée sur quelques critères que nous citons ci-joint **[7]** :

Les critères	Ad hoc	Standards	intergiciel	EAI	BPM	ESB
Simplicité	Simple	Simple	Difficile	Difficile	Difficile	Très difficile
Mise en œuvre	Simple	Simple	Difficile	Difficile	Difficile	Difficile
Coût Maintenance	Coûteuse	Très coûteuse	Coûteuse	Coûteuse	Moins coûteuse	Moins coûteuse
Modification	Difficile et coûteuse	Difficile /coûteuse	Difficile et coûteuse	Difficile	Très rapide et facile.	Très facile
Flexibilité	Non flexible (Câblage dure).	Flexible	Non flexible (Propriétaire)	Non flexible	Non flexible	Flexible
Extensibilité	Très difficile	Difficile	Extensible	Extensible	Extensible	Extensible
Architecture	Distribuée	-	Centralisée	Centralisée	Centralisée	Distribuée
Périmètres	A2A	A2A B2B	A2A	A2A	A2A	A2A B2B

Tableau I.3 : Comparaison entre les différentes technologies d'intégration.

8. Apports de l'intégration d'applications aux entreprises

Quelle que soit la raison qui conduit une entreprise à entreprendre un projet d'intégration d'applications, et quelque soient les difficultés et les problèmes qu'elle va rencontrer, l'entreprise cherche, avant tout, à réduire ses coûts, à produire de la valeur et à gagner en agilité et bien d'autres bénéfices.

a. Réduction des coûts de développement et de maintenance

Les solutions d'AI réorganisent l'entreprise, et fournissent une architecture ouverte et simple. La simplicité de l'architecture diminue également le coût de la gestion et de maintenance du système d'information **[33].** En plus de 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 de réorganisation.

b. Support efficace des opérations de fusions, et d'acquisitions

Les solutions d'AI sont généralement basées sur des standards. Ceci facilite l'amélioration de système d'information par l'addition et le développement de nouvelles applications supportant ces standards, il permet aussi à l'entreprise de se regrouper facilement et interagir avec d'autres entreprises **[36].**

c. Prolonger le cycle de vie des applications patrimoniales

Afin de pouvoir intégrer les applications qu'elles souhaitent implémenter, les entreprises doivent d'abord faire évoluer leurs systèmes patrimoniaux[32] en fonction de leurs besoins. L'intégration des applications patrimoniales des entreprises dans des solutions d'AI permet d'étendre le cycle de vie de ces applications **[36].**

d. Réduction du délai d'arrivée au marché

La flexibilité apportée par les solutions d'AI permet aux entreprises de réagir plus rapidement aux évolutions du marché et aux changements réglementaires, de personnaliser et adapter facilement leurs processus métiers internes et externes aux besoins des clients et aux comportements des concurrents, et aussi d'étendre les fonctionnalités des applications pour présenter de nouveaux produits dans un temps court **[33] [36].**

e. Amélioration de la performance et de la fiabilité

Les solutions d'AI utilisent principalement les mécanismes de transmission des messages asynchrones non bloquants pour transporter l'information de l'application productrice à l'application consommatrice. Par conséquent, la livraison des messages est garantie. Ce modèle de transmission augmente considérablement la performance et la fiabilité du système intégré **[14].**

[32] Actuellement, beaucoup d'entreprises font face au problème de manipulation de leurs applications patrimoniales qui sont considérées comme le noyau de leurs affaires. Ces systèmes sont âgés plus de 30 ans et nécessitent un groupe de 50 personnes ou plus pour la maintenance.

9. Limitations de l'intégration des applications

Le domaine d'intégration d'applications (AI) présente certaines limitations, à l'heure où nous nous trouvons entourés par une myriade d'approches et de technologies pertinentes et éprouvées. Nous allons résumer celles les plus importantes dans ce qui suit :

- Les solutions d'intégration d'applications (AI) sont généralement des solutions propriétaires , dans la mesure où le protocole utilisé pour l'échange et le transport des messages, les connecteurs d'accès aux applications et les formats de données utilisées, sont encore largement spécifiques et propriétaires et ce malgré la percée des services web qui ont des limitations particulièrement au niveau transactionnel et sécurité, ce qui nécessite dans certains cas l'utilisation d'autres technologies pour des raisons de performances et de qualité du service **[7] [17].**

- Les solutions actuelles d'intégration d'applications (AI) sont des infrastructures construites sans vision stratégique. Elles se basent essentiellement sur des technologies, et qui ne suffisent pas pour avoir une intégration optimale et efficace des applications. Ces solutions souffrent principalement du manque d'abstraction et de méthodologie stratégique permettant d'atteindre la solution souhaitée **[29].**

- Les solutions d'intégration proposées sont basées essentiellement sur des mécanismes pour l'interconnexion des applications au niveau technique et au niveau syntaxique (des mécanismes de mapping ou de transformations syntaxiques), mais elles ne traitent pas correctement, pour le moment, la problématique liée à l'intégration sémantique **[6] [28].** Cela reste un domaine ouvert, même si des travaux de recherche **[6] [39] [40]** se sont intéressés à ce problème, en proposant des approches orientées services sémantiques qui permettent de résoudre les différentes hétérogénéités qui peuvent exister au niveau des ontologies.

- Les solutions de l'intégration d'applications actuellement souffrent d'un problème sérieux, celui du manque d'adaptation, qui s'avère un principe fondamental et nécessaire notamment dans certains cas liés aux changements organisationnels, techniques et fonctionnels que subit l'entreprise **[28]**.

- Les solutions d'intégration d'applications (AI) actuelles ne répondent qu'à une partie des besoins d'intégration : l'urbanisation des systèmes d'informations. La dimension

métier n'est pas adressée dans ces solutions, qui souffrent du manque de services pour la modélisation et l'intégration de processus métiers, donc le besoin d'une meilleure manière à gérer et exécuter les processus métiers de leurs systèmes d'informations **[30] [17].**

Notre travail vise essentiellement à donner une vision stratégique aux projets d'intégration des applications d'entreprises, en proposant une méthodologie stratégique permettant d'exploiter des technologies standards 'les services web' (voir le chapitre II) de manière structurer et efficace pour atteindre une intégration étroite des différentes applications.

10. Synthèse du chapitre

Nous avons parlé de l'intégration des applications (A2A) au sein de l'entreprise et l'intégration entre entreprises (B2B, B2A, administration to administration). En effet, il n'y a pas, d'un point de vue technique, de différence de traitement au niveau de l'intégration B2B ou A2A, seul le contenu est différent. A l'issue de ce chapitre, nous avons choisi de faire une comparaison entre l'intégration A2A et B2B. (voir le Tableau I.4).

	A2A application intégration	**B2B application intégration**
Périmètres d'intégration	• Entre applications d'une même entreprise. • Entre l'entreprise et ses employés.	• Entre entreprises. • Entre applications d'entreprise et des clients. • Entre partenaires d'affaires
Approches d'intégration	• Intégration par données • Intégration par interface d'applications (API) • Intégration par méthodes • Intégration par processus métier	• Intégration par données. • Intégration par interface d'applications (API). • Intégration par méthodes. • Intégration par portail. • Intégration par processus métier.
Technologies	• CORBA, Intergiciels, EAI, BPM, serveur d'applications, …	• EAI, BPM, ESB.
Standards	• XML, CORBA, RMI, …	• XML, ebXML, services web, …

Caractéristiques	• Un environnement de communication et d'interopérabilité des applications internes de l'entreprise. • Fournit un support efficace de gestion de transactions dans un environnement distribué.	• Un environnement de communication et de collaboration des partenaires d'affaires. • Un support de protocole d'affaires (cXML[33], XOCP[34], ebXML, …). • Fournit des services de sécurité efficaces : authentification des identités des partenaires, associer des signatures numériques aux messages échangés, l'utilisation des protocoles qui supportent le cryptage des messages échangés.

Tableau I.4 : Comparaison entre A2Ai et B2Bi[35].

En ce qui nous concerne, notre travail s'inscrit dans le cadre d'intégration des applications internes de l'entreprise ainsi qu'à l'extérieure de l'entreprise, entre les partenaires et les clients particuliers. Ceci est réalisé par la structuration des ses fonctionnalités en architecture de services web (voir chapitre II), et l'intégration de leurs processus métiers.

[33] cXML : commerce eXtensible Murkup Language.
[34] XOCP : eXtensible Open Collaboration Protocole.
[35] A2Ai : Application to Application integration.
B2Bi: Business to Business integration.

11. Conclusion

L'entreprise de demain sera l'entreprise intégrée, dont la compréhension et la maîtrise de multiples domaines d'activités, de stratégies, d'architectures, d'unités organisationnelles, d'utilisateurs, de processus, de systèmes automatisés, de plateformes, de technologies, de données et d'informations, forme un tout consistant, cohérent et adaptatif.

L'Intégration des Applications d'Entreprise et l'interopérabilité des services intra et inter applications est la meilleure solution, pour mieux répondre aux attentes des entreprises, qui cherchent, avant tout, à gagner en souplesse et en réactivité. Ce chapitre a été consacré à la présentation des approches, des technologies, et des avantages d'intégration d'applications intra et inter-entreprises. Nous avons également présenté les problèmes qui arrivent devant le processus d'intégration.

La nature complexe du processus de développement de nouvelles applications, et d'intégrations des systèmes existants, implique l'utilisation de plusieurs standards, de diverses technologies et la participation de plusieurs acteurs. En outre, la manière de structurer et d'exploiter ces outils, est aussi un aspect important dans un projet d'intégration d'applications.

Le prochain chapitre, a pour objectif de présenter une étude approfondie des différents modèles d'architectures existants permettrant d'assurer les échanges A2A et B2B. Nous présentons, aussi, parmi la panoplie de solutions et des techniques présentées dans ce chapitre, la meilleure solution qui réalise une intégration étroite entre les différentes parties d'une compagnie. Et leurs différentes méthodologies de développement.

CHAPITRE II

Architectures & Méthodologies d'Intégration d'Applications

1. Introduction

Dans l'économie actuelle et particulièrement dans le monde industriel, l'entreprise ne peut pas être en mesure de répondre aux nouvelles exigences de ses clients si elle reste basée seulement sur son système patrimonial applicatif sans accompagner l'évolution technologique. Les entreprises sont alors obligées d'évoluer pour rester au premier rang du monde industriel. Elles doivent fréquemment fusionner avec d'autres entreprises (partenaire et\ou fournisseur), réorganisant leur structure interne et adoptant de nouvelles technologies et plateformes pendant qu'elles essayent de rester compétitives et d'obtenir des avantages concurrentiels.

Les besoins d'intégration et d'acquisition de nouvelles technologies et applications externes, forcent les entreprises à penser à la reconstruction de nouveaux systèmes par des moyens et des techniques avancées, pour être capable d'améliorer leurs activités pour les clients, et de réussir une intégration étroite avec les partenaires et les fournisseurs. Néanmoins, le remplacement des Systèmes d'Information des Entreprises (SIEs) provoque beaucoup de problèmes, comme l'énorme investissement perdu pour le développement de ces systèmes. Le problème le plus important est que les responsables des entreprises n'ont pas une confiance totale aux nouveaux systèmes, dans la mesure où le développement de nouveaux systèmes qui implémentent bien la logique d'affaires des anciens systèmes est une tâche très difficile voire impossible.

Les responsables des entreprises, n'ont trouvé que deux solutions possibles pour fusionner leurs applications hétérogènes, ou même pour acquérir des applications étrangères afin de les embarquer comme une étape dans leurs processus d'affaires. La première est la modernisation de ces systèmes d'information pour les adapter avec les nouvelles technologies, mais cette solution selon Ziemann **[41]** est très compliquée et très coûteuse à cause de l'absence d'une documentation détaillée de ces systèmes. L'autre solution propose de traiter les applications des systèmes à intégrer afin d'exposer leurs fonctionnalités aux applications internes et/ou au marché, comme des services web. Cette solution permet aussi, la réutilisation de ces fonctionnalités pour créer de nouvelles applications.

Ce chapitre a pour objectif d'étudier les différentes architectures d'intégration et leurs modèles de base. Ensuite, nous exposons les différentes solutions d'intégration des entreprises, qui sont le remplacement, l'adaptation et l'intégration et enfin la migration. Dans la section qui suivra, nous présentons la notion de l'architecture orientée services, qui est

considérée comme étant la meilleure solution pour l'intégration des applications multiples, et la notion de service web, qui est proposée comme étant la technologie la plus efficace pour la réalisation de SOA, nous exposons aussi dans cette section les exigences et les difficultés de migration des systèmes d'information d'entreprises vers un environnement SOA. Après nous discutons les quatre approches de la migration. L'approche *Top Down*, l'approche *Bottom Up*, l'approche *Outside In* et enfin l'approche *Middle Out.* Enfin nous arrivons à la section de migration des systèmes d'information d'entreprises vers les architectures orientées services web, où nous présentons des exemples des différents travaux existants.

2. Architectures d'intégration

Les architectures d'intégration constituent le socle de mise en œuvre des différentes technologies d'intégration, vues dans le chapitre précédent. En plus des modèles logiques d'intégration point à point[36] et celles basées sur les middlewares, Kang **[2]** distingue deux typologies d'architectures d'intégration, il s'agit de l'architecture ***Hub-and-Spoke*** et de l'architecture ***bus***. Ces dernières sont aussi considérées par Izza **[7]** et Arnaud **[33]** comme des architectures d'intégration de base permettant de structurer les solutions d'intégration. Ils considèrent aussi toutes les autres architectures comme des formes hybrides combinant les deux formes de base *hub-and-spoke* et *bus*, telle que l'architecture ***multi-hub*** (en anglais : *network centric*) **[8] [14]**.

1.2 Architecture Hub-and-Spoke :

C'est le modèle centralisé des solutions d'Intégration d'Applications **[33]**. Il correspond à un modèle étoilé qui consiste à déployer en un seul point focal : le hub[37] central, l'ensemble des services assurés par l'infrastructure **[8]**. La typologie Hub-and-Spoke utilise le modèle d'interaction M*essager*, qui permet d'encapsuler la logique d'interaction de l'application, en minimisant les dépendances de communication entre applications pour passer l'information et les traitements entres eux **[7]** (voir la figure II.1)**.**

Comme le montre la figure II.1, les applications à intégrer sont reliées via des connecteurs à un hub central d'EAI. Quand une application envoie un message, ce dernier est expédié à l'hub où il est traité afin que l'application cible le reçoive **[8]**.

[36] La topologie structurelle point à point n'a peu d'intérêt.
[37] Le hub est un élément focal de système d'information, il centralise les flux en assurant les transformations et les routages nécessaires.

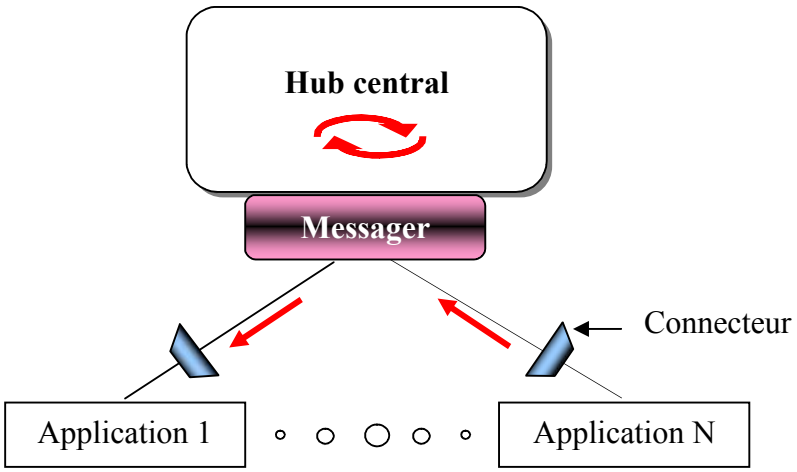


Figure II.1 : Typologie d'intégration Hub-and-Spoke [8].

2.2 Architecture en Bus :

C'est le modèle décentralisé des solutions d'Intégration d'Applications **[33]**. Il consiste à distribuer l'ensemble de services de l'infrastructure sur des nœuds, ce sont les points de connexion à une application **[8]**. L'architecture en bus utilise plusieurs modèles d'architecture pour faire communiquer et interopèrer ses applications d'une manière distribuée à titre d'exemple l'*automate de processus*, qui permet d'encapsuler la logique d'automatisation du processus et les applications **[7]** (voir la figure II.2).

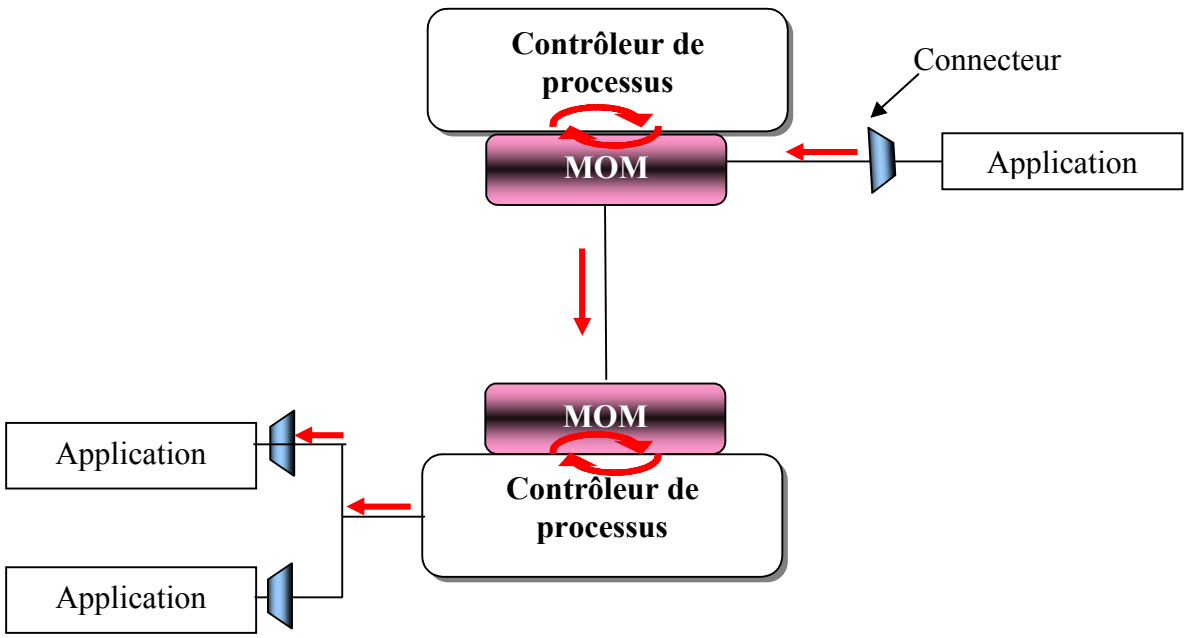


Figure II.2 : Typologie d'intégration en bus [8].

Dans la figure II.2, les applications à intégrer sont reliées via des connecteurs à des nœuds d'interaction. Quand une application émet un message, ce dernier est traité par le nœud correspondant de l'application, afin que les applications cibles le reçoivent. Dans cet exemple deux modèles d'architecture sont utilisés : le modèle *Contrôleur de processus* qui gère l'ordonnancement des activités de processus métier du système entre les application et le modèle *Messager* qui permet de gérer les interactions et le transfert des messages entre eux. Avec ce type d'architecture la charge est donc naturellement répartie sur l'ensemble des nœuds, qui vaut au nombre des applications à intégrer **[8].**

2.3 Architecture Network Centric :

Ce modèle permet de combiner les deux approches Hub-and-Spoke et l'approche en bus, en mettant ainsi en évidence la subdivision de système d'information en plusieurs îlots informationnels, découlant généralement d'une approche d'urbanisation menée en amont. L'intégration intra-îlots est réalisée selon la typologie *hub-and-spoke*, tandis que inter-îlots est réalisée généralement selon une typologie *bus* **[7]** (voir la figure II.3).

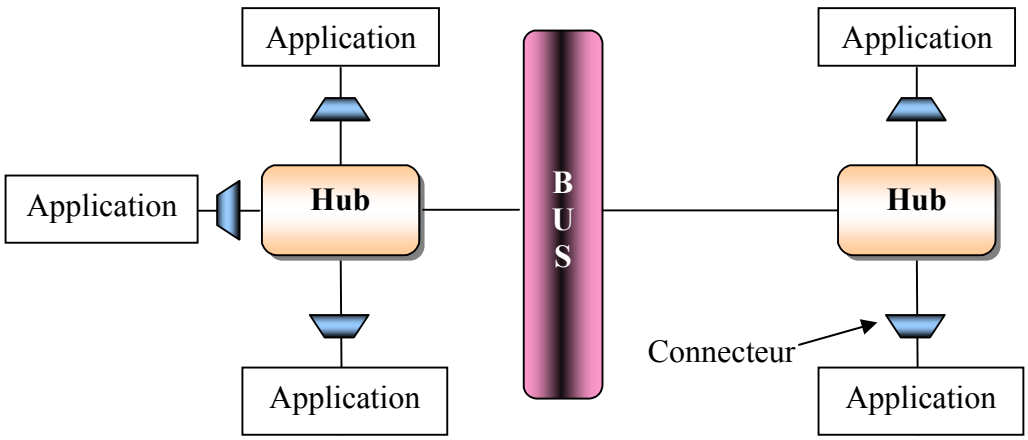


Figure II.3 : Typologie d'intégration multi-hub [7].

2.4 Synthèse :

La différence entre ces typologies n'est pas uniquement technique, et le choix d'une architecture meilleure qu'une autre est principalement due à des raisons tarifaires, dans la mesure où, dans le modèle en bus le client a la possibilité d'utiliser et de payer que pour l'ensemble des applications qu'il souhaite couvrir, par opposition à l'architecture hub-and-spoke, le client a la possibilité de n'utiliser et de payer que pour un seul hub, pour couvrir un large éventail d'applications même s'il n'en y pas besoin **[8]**.

Dans le tableau suivant nous avons voulu récapituler les points forts et faibles des différentes typologies d'architecture d'intégration, ainsi que les principaux modèles qui constituent la base de ces architectures, synthétisés de quelques travaux **[7] [8] [33]**:

Architectures d'intégration	Points faibles	Points forts	Modèles de base
Point à point	• Sa complexité augmente avec le nombre d'application	-	• Adaptateur[38]. • Façade[39].
Hub-and-spoke	• Le point centralisé constitue le maillon faible en cas de surcharge de traitement	• Administration de l'infrastructure est grandement facilitée. • Simple à implémenter. • Facilite la maintenance	• le modèle messager. • Le modèle médiateur.
Bus	• Infrastructure lourde.	• La charge est répartie sur plusieurs nœuds d'interaction. • Renforce la scalabilité	• Messager. • Médiateur[40]. • Contrôleur de processus

Tableau II.1 : Caractéristique des différents modèles d'intégration.

3. Solutions d'intégration des entreprises

Un des aspects les plus importants, que rencontrent les entreprises pour fusionner leurs applications, ou encore pour acquérir des applications étrangères est l'introduction de nouvelles technologies et de nouveaux environnements sans aucune considération de la façon de les obtenir **[51]**.

Les responsables des entreprises ont pensé à la reconstruction de nouveaux systèmes par des moyens et des techniques plus avancées. Cela, et par opposition aux années passées, représente un grand problème, en raison de la complexité excessive des systèmes d'information qui ont subit plusieurs reprises de maintenance pendant des années. Ceci rend la réimplémentation d'un nouveau système une tâche difficile voire impossible. L'objectif de

[38] L'adaptateur permet de convertir l'interface d'une application existante en une interface souhaitée par d'autres applications ayant besoins d'utiliser cette interface [7].
[39] Le modèle Façade permet de fournir une interface simplifiée de l'application [7].
[40] Le médiateur : permet d'encapsuler la logique d'interaction de l'application, en minimisant ainsi les dépendances entre les applications [7].

cette section est d'explorer les solutions proposées dans le domaine d'intégration des applications. Nous annonçons trois solutions différentes :

3.1 Remplacement : ou « COTS replacement »

Cette solution consiste à faire un remplacement incrémental de système. Comme le montre la figure II.4, les applications de système d'information sont remplacées par des applications dites COTS[41] « Commercial Off-The Shelf applications » plus avancées et qui supportent les nouveaux besoins métiers de l'entreprise **[51].**

Cette solution est utile si le système d'information de l'entreprise a un degré important de cohésion et de modularité.

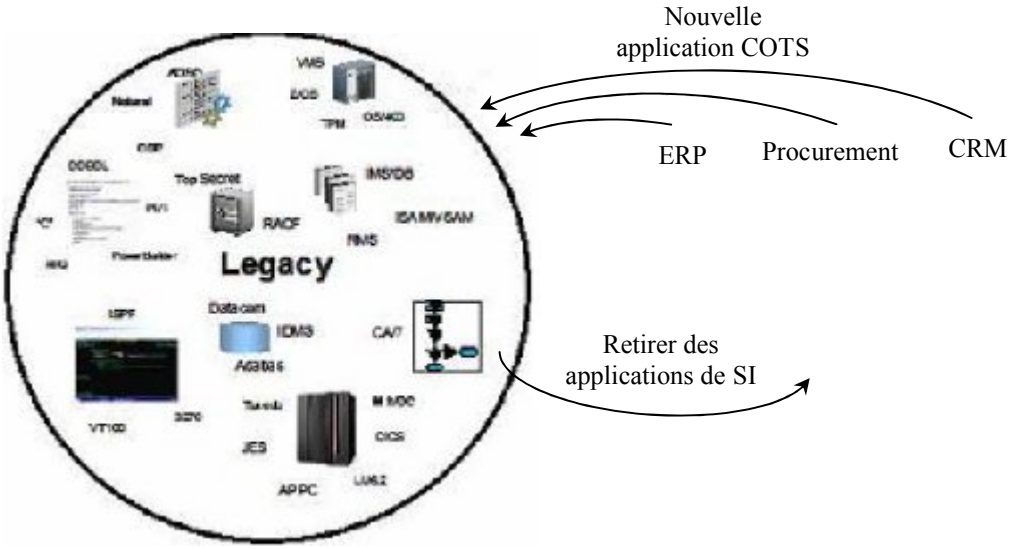


Figure II.4 : Remplacement par COTS applications [51].

Cette solution n'a peu d'intérêt, puisque elle n'a, en réalité, fait qu'amplifier et renforcer le problème d'intégration au sein de l'entreprise[42]. Et elle est très coûteuse, dans la mesure où, l'adaptation de système à ces nouvelles applications COTS, est très complexe et nécessite un énorme investissement.

3.2 Adaptation et intégration : ou « Wrapping »

La manière la plus simple pour assurer l'intégration des entreprises est de les adapter. Cette solution est basée sur le principe d'adaptation qui consiste à entourer le système d'information de l'entreprise par une couche logicielle qui cache sa complexité et exporte une interface moderne.

[41] Applications COTS : (Composants logiciels à l'étagère), des composants logiciels disponibles sur le marché.
[42] Cette solution engendre le problème d'intégration des silos d'applications COTS au sein de l'entreprise.

La couche logicielle est composée d'un ensemble d'adaptateurs (en anglais : Wrappers), qui sont utilisés principalement pour retirer le désaccord entre l'interface exportée par le logiciel et les interfaces exigées par le système. Malheureusement, cette solution n'est pas toujours pratique, parce qu'elle exige une compréhension approfondie des modules de système d'information **[51].**

3.3 Migration : ou « SOA Migration »

Avant d'entamer la présentation de cette solution, nous avons choisi, tout d'abord, d'exposer les notions clés de cette solution qui sont : l'architecture orientée services et les services web.

3.3.1 Architectures Orientées Services (AOS[43])

La notion des architectures orientées services ne date pas d'hier puisque la première utilisation de ce concept remonte vers les débuts des années 90. Il a néanmoins fallu attendre la fin de ces années pour que cette notion se popularise notamment grâce aux modèles d'Architectures de services web.

Selon Jiang et Willey, l'Architecture Orientée Service (AOS) est un style architectural de développement et d'intégration *dynamique* des applications d'entreprise. Elle permet notamment de structurer le système d'information en une collection de services, la brique de base de cette architecture. Ces derniers peuvent être librement (indépendamment du matériel) interopérés et réutilisés par d'autres systèmes d'information via une interface commune, qui est le bus de services **[49]**.

Les systèmes basés sur les architectures orientées services offrent une solution flexible au problème d'intégration des systèmes d'information, de protocoles, de sources de données, et de processus **[49]**.

La figure II.5 explique le scénario d'exploitation de SOA pour l'intégration de deux entités étrangères. Deux acteurs interviennent dans ce scénario : l'agent fournisseur qui publie les services et l'agent demandeur (ou consommateur) de services.

Les services sont décrits à travers une représentation structurée : XML (*e**X**tensible **M**arkup **L**anguage*), par l'agent fournisseur. Ensuite, ce dernier enregistre les descriptions de ses services dans un référentiel (ou annuaire). L'agent demandeur cherche dans l'annuaire des

[43] En anglais SOA pour Services Oriented Architecture.

services selon des critères spécifiques au mécanisme de recherche. L'annuaire retourne à l'agent demandeur les informations d'un service demandé. L'agent demandeur récupère la description de ce service et l'utilise pour échanger des messages et communiquer avec le service.

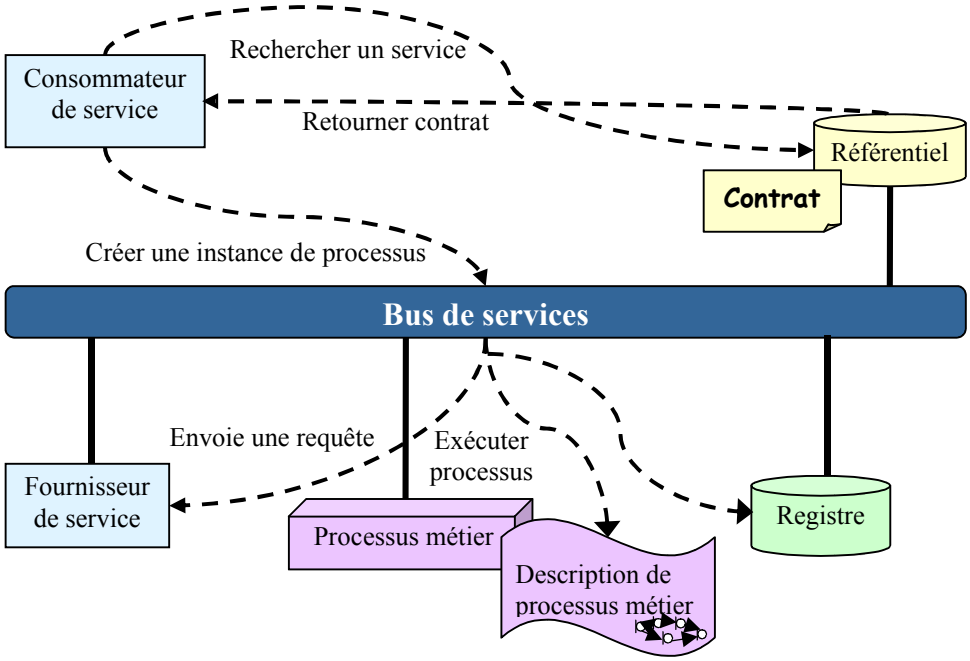


Figure II.5 : Scénario de fonctionnement de l'architecture SOA [40].

❖ **SOA et l'Intégration d'Applications :**

Par définition **[43]**, les systèmes basés sur l'architecture orientée services sont une réponse efficace aux problématiques d'intégration que rencontre l'entreprise en termes d'encapsulation, de réutilisation, de composition, d'interopérabilité, et de réduction de couplage entres les systèmes d'information d'entreprise. L'architecture de services peut être utilisée dans un cadre interne de l'entreprise, ainsi que dans un cadre externe, elle facilite surtout la communication entre entreprises.

L'apport de cette architecture dans le domaine d'intégration n'est pas uniquement technique, dans le tableau suivant nous récapitulons les bénéfices métiers et techniques de ce style d'architecture dans le domaine d'intégration d'applications :

Bénéfices métiers	Bénéfices techniques
• Améliore l'agilité et la flexibilité de métier aux changements des besoins clients. • Facilite la gestion de processus métier. • Réduire en temps le cycle de développement	• Réduire la complexité de la solution. • Garantir une intégration standardisée et le support des clients

des produits. • Améliore le retour en investissement et la productivité de système d'information.	hétérogènes. • Facilite la maintenance de la solution.

Tableau II.2 : Bénéfices des architectures orientées services [40].

3.3.2 Services Web (WSA[44])

Les services web sont des composants logiciels encapsulant des fonctionnalités métier de l'entreprise et accessibles via des protocoles standards du web. Une définition plus précise des services web est fournie par le magazine de développement Programmez **[43]**. Il définit un service web comme « *une manière standardisée d'intégration des applications basées sur le web en utilisant les standards ouverts XML, SOAP, WSDL, UDDI et les protocoles de transport de l'Internet. XML est utilisé pour représenter les données, SOAP pour transporter les données, WSDL pour décrire les services disponibles, et UDDI pour lister les fournisseurs de services et les services disponibles* » (voir figure II.6).

De nos jours, les services web fournissent les technologies les plus adaptées pour rendre possible la création de l'architecture orientée services **[16]**. Cependant, il ne faut pas confondre les services web avec SOA. Les services web fournissent le support pour la description et l'infrastructure de communication standard des services (voir figure II.6), alors que SOA décrit, comment un système composé de services peut fonctionner **[46]**.

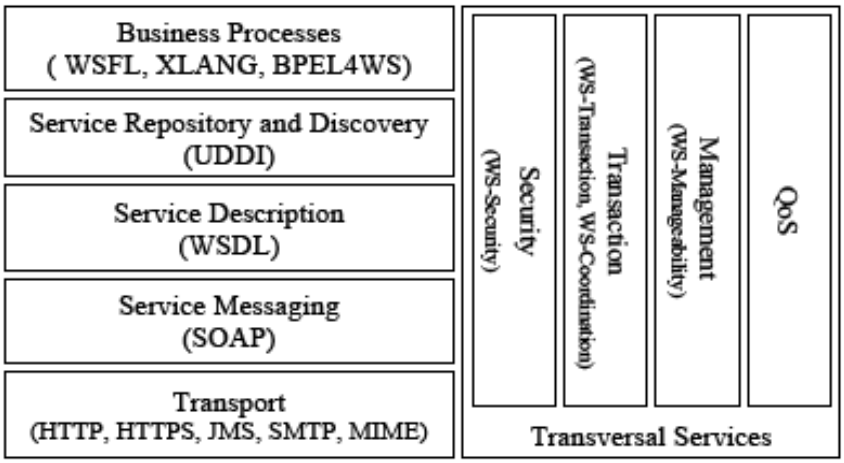


Figure II.6 : Standards des services web [46].

[44] WSA: Web Service Based Architecture.

3.3.3 La migration vers SOA :

La migration ou la modernisation implique des changements plus étendus qu'une simple maintenance, mais une partie significative de système est conservée. Ces changements incluent souvent la restructuration du système, l'amélioration des fonctionnalités ou la modification des attributs du logiciel **[51]**.

a. Bénéfices de la migration :

La migration offre plusieurs bénéfices aux entreprises et à leurs services, parmi ces bénéfices nous citons **[48]**:

- L'adaptation aux nouveaux besoins,
- L'amélioration des services 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,
- L'augmentation de la flexibilité et la réactivité des systèmes d'information,
- L'élimination de la dépendance forte à l'ensemble des anciennes compétences.

b. Difficultés de la migration :

Les efforts de la modernisation de systèmes d'information d'entreprises ont échoué dans la plupart du temps à cause de plusieurs facteurs, nous citons les suivants **[48]:**

- La complexité des systèmes patrimoniaux : la complexité est considérée comme le plus grand limiteur du processus de migration, elle est produite à cause de la taille énorme de système, de l'incompréhensibilité des systèmes à migrer et des phases successives de maintenance.
- Les risques de migration : les risques de migration ne sont pas majeurs, il est possible d'accepter un certain risque si nous devons accomplir une tâche de migration. Malheureusement, beaucoup d'entreprises sont incapables ou ne veulent pas contrôler le risque correctement. Ceci également prévenir de l'insuffisance de compréhension de gestion des risques et des techniques de réduction du risque.

4. Différentes approches de migration

La migration ou la modernisation du système peut être distinguée par la stratégie de développement exigée pour accomplir la migration. Il existe deux approches proposées pour la migration, qui sont l'approche "*Top Down*", et l'approche "*Bottom Up*" **[52]**. Ces dernières sont considérées comme des approches de base. D'autres approches baptisées "*Outside In*" (aussi appelée *Meet In The Middle*) et "*Middle Out*" sont des approches hybrides des deux approches de base, et qui sont proposées pour pallier les désavantages des approches *Top Down* et *Bottom Up* **[44]** (voir la figure II.7). Le choix entre ces approches dépend de certains critères décisifs, tel que :

- L'état des systèmes informatiques existants (nature, degré de complexité, capacités fonctionnelles et techniques).
- Les objectifs attendus de la migration.
- Les capacités des groupes de travail…etc.

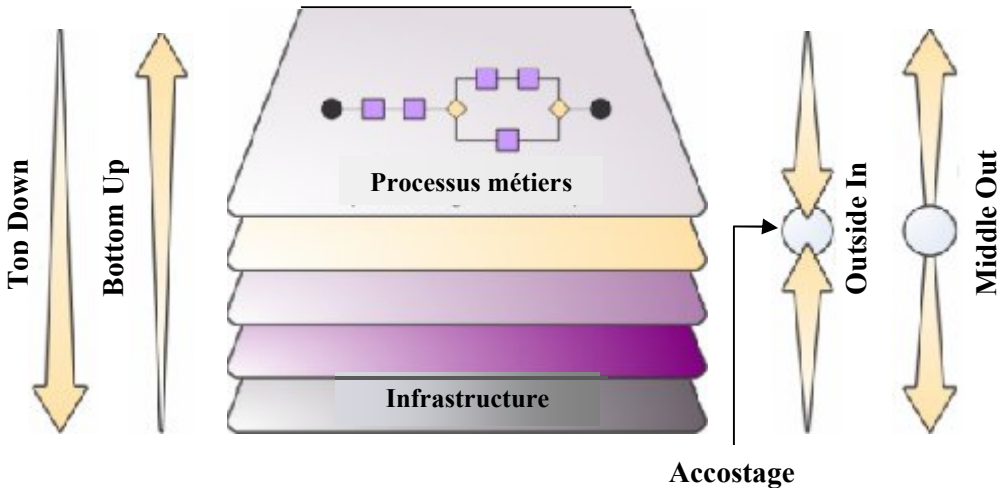


Figure II.7 : Différentes approches de migration [44].

4.1 Approche Top Down :

Dans ce type de projet d'intégration le système d'information de l'entreprise est vu comme un super bloc applicatif. Cette approche est descendante et globalisante dans la mesure où elle touche à la globalité de l'entreprise (voir la figure II.7). Elle vise à créer un socle commun d'intégration aligné sur le métier de l'entreprise.

Le point de départ de cette approche est la définition et la formalisation des processus métiers[45], qui représentent les objectifs fonctionnels de l'entreprise, pour descendre ensuite au travers des différentes couches du système afin de définir les services nécessaires à la réalisation de ces processus.

Cette approche peut être appliquée dans le cadre d'un système urbanisé ou pour démarrer un nouveau projet. Cependant, elle est très coûteuse, très longue (plusieurs années) et trop risquée, puisque elle cause la reconstruction de tout ou d'une partie de système d'information **[7] [39]**.

4.2 Approche Bottom Up :

A l'inverse de l'approche "*Top Down*", le système d'information de l'entreprise dans cette approche est vu comme un ensemble de blocs applicatifs autonomes et hétérogènes. L'approche "*Bottom Up*" est une approche ascendante (voir figure II.7), conduisant à des projets départementaux[46], où chaque projet démarre par une phase d'analyse de l'existant afin de déterminer les fonctions transverses existantes dans le système. Ensuite, ces fonctions sont mises en mode service, pour qu'ils soient exploitables au sein d'un autre service et/ou un processus métier.

Cette approche est intéressante, puisque elle contraint à réaliser une cartographie du système, ce qui facilite la publication et la réutilisation de ses fonctionnalités intéressantes. Cependant, cette approche présente un inconvénient majeur, est qu'elle ne prend pas en charge les nouveaux besoins de l'entreprise, dans la mesure où elle se limite à encapsuler les fonctions existantes au niveau du système dans des services, qui restent très fortement couplés à leur application d'origine **[17] [39]**.

4.3 Approche Outside In :

Cette approche est faite d'itérations successives de l'approche "*Top Down*" et de l'approche "*Bottom Up*", afin de définir les services métiers qui satisfont au mieux les exigences et les contraintes métiers d'une part, et informatiques d'autre part.

L'approche "*Outside In*" propose de démarrer en parallèle (figure II.7), l'approche *Top Down* "sans se préoccuper de l'existant", pour définir les besoins métiers en processus métiers et les services nécessaires à leur réalisation. Et l'approche *Bottom Up* "en ne considérant que

[45] La définition des processus métiers est le point de départ, puisque le but principal de migration vers un environnement SOA, est de répondre aux besoins fonctionnels de l'entreprise.
[46] Dans l'approche Bottom Up, le système d'information est appréhendé bloc applicatif par bloc applicatif.

l'existant", afin de cartographier l'existant applicatif dont dispose l'entreprise pour supporter les services métiers à forte valeur ajoutée métier.

Une fois ces deux chantiers en phase finale, commence l'étape de l'**accostage**. Son objectif est de réconcilier les résultats des deux approches afin de déterminer comment seront réalisés les processus métiers. Il faut, pour cela, comparer les besoins en services exprimés par l'approche *Top Down* avec ceux remontés de l'approche *Bottom Up.*

Cette approche réunit les bénéfices des approches *Top Down* et *Bottom Up*. Elle permet de piloter le projet d'intégration par les besoins métiers tout en facilitant la réutilisation de services et la capitalisation sur l'existant. Mais la réussite de cette approche dépend de la délicate étape de l'accostage qui nécessite de faire les compromis nécessaires pour réutiliser le maximum de code **[44]**.

4.4 Approche Middle Out :

Par opposition à l'approche *Outside In*, cette méthode propose de commencer au milieu « en anglais : In the middle », c'est-à-dire là où le métier et les technologies d'information (ou, IT pour ***I****nformation* ***T****echnology*) parlent le même langage (en tout cas presque). Elle s'attaque donc d'emblée à ce qui reste un des facteurs limiteurs à l'adoption des SOA : La compréhension du métier de l'entreprise par les maîtrises d'œuvre et inversement**,** la compréhension des contraintes IT par les maîtrises d'ouvrage.

Une fois les différentes parties sont d'accord sur un premier socle de services "métiers" nécessaires :

- Les maîtrises d'ouvrage engagent un chantier "*Middle Up*" pour spécifier les processus métiers.
- Les maîtrises d'œuvre engagent un chantier "*Middle Down*", pour spécifier le socle de services de plus bas niveau permettant la réalisation des services métiers.

Cette approche limite par contre le pilotage de la SOA par les besoins métiers, puisque le point de départ est l'identification des services métiers nécessaires et non la définition des processus réalisant le métier de l'entreprise **[26] [44]**.

5. Migration des SIEs vers SOA

Dans cette section, nous présentons quelques travaux proposés pour la migration des systèmes d'information d'entreprise (SIEs) vers une architecture orientée services (SOA). Ces travaux sont présentés selon les classes d'approches de migration discutées dans la

section précédente, celle de type *Bottom Up* et l'autre de type *Meet In The Middle*, comme suit :

5.1 Migration Bottom Up :

Cette approche propose d'intégrer les systèmes d'information d'entreprises via des adaptateurs, pour que les applications puissent être invoquées comme des services web. Dans ce type de migration le système est appréhendé par composant applicatif, dont chaque composant est analysé et décomposé en services web réutilisables. L'intégration des systèmes se réalise donc, par la composition et l'intégration des services web dans des processus métiers.

Pour bien illustrer ce type de migration, nous avons choisi l'approche suivante :

5.1.1 Approche de Sneed [45] :

Dans **[45]**, Sneed propose une approche, dite « **SWA** : ***S**alvaging & **W**rapping **A**pproach*), pour la création des services web à partir des systèmes d'information d'entreprises, basée sur la détection du code qui implémente la règle métier. La règle métier est 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. Selon cette approche, pour détecter le code qui implémente la règle métier, il faut d'abord déterminer le résultat produit par cette règle. Ceci est effectué par l'identification des résultats qui sont retournées par les fonctions implémentant la règle métier et aussi par l'identification des fonctions **[37]** (figure II.8). Le problème qui se pose ici est qu'un bloc de code 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, ensuit les blocs de code contenant ces instructions seront recopiés à partir du code original avec les variables utilisées en formant un nouveau module séparé.

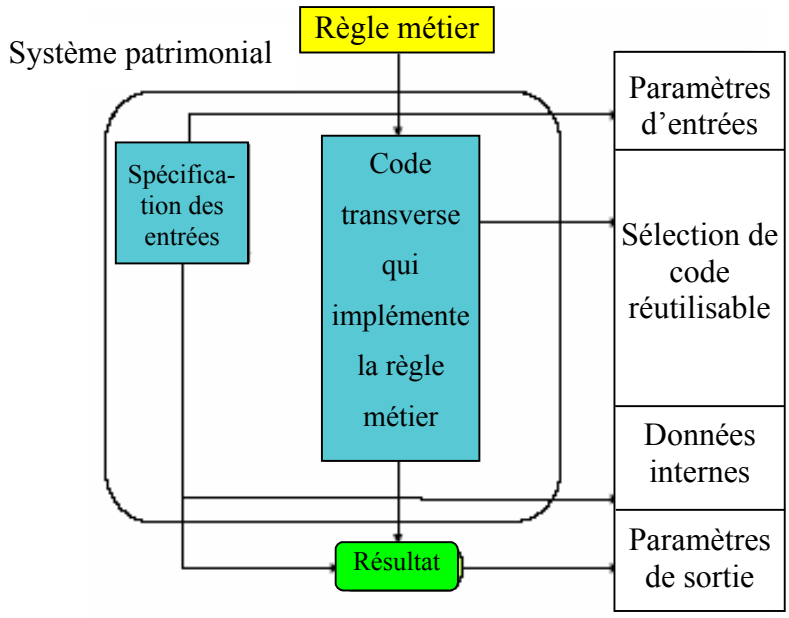


Figure II.8 : Récupération du code de service [37].

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 de flux des données. Cette documentation est utile pour la décision si une règle existante, implémentée dans un système peut être réutiliser comme un service public dans une architecture orientée service. Cette décision nécessite la compréhension totale de cette règle métier et de sa valeur économique. La règle métier ayant une implémentation acceptable et ont une valeur économique 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. Pour accomplir cette tâche Sneed utilise un outil logiciel appelé Software **[38]** (figure II.9). Cet outil a été développé essentiellement pour automatiser le processus de génération d'une interface WSDL pour un service écrit en (PL/1, COBOL ou C/C++), et génère aussi pour chaque service deux modules supplémentaires, un pour analyser le message entrant et extraire les données, et l'autre pour créer le message de retour contenant les résultats produits par le service.

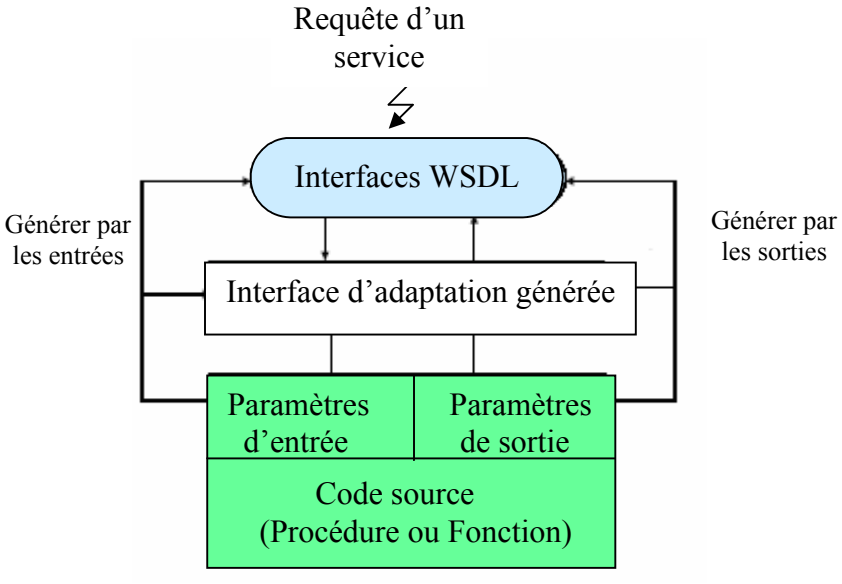


Figure II.9 : Adaptation du code récupéré [38].

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 applicatif. Sur le serveur d'application, il existe un planificateur « Scheduler », qui reçoit le message SOAP, détermine par quel service web doit être exécuté et l'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 transmettre au client.

5.2 Migration Meet In The Middle :

Cette approche, dite ***"Outside In"***, est le sujet de notre travail, dont le but est de définir et d'intégrer les parties intéressantes des blocs applicatifs, qui ont besoin de travailler ensemble et de s'échanger des services entre eux. Cette approche est plus intéressante par rapport à d'autres approches, puisque l'extraction et la réutilisation des fonctionnalités importantes c'est-à-dire les parties des systèmes interopérables se font de manière aisée et guidée par les besoins métiers.

Pour bien présenter cette approche de migration, nous avons choisi les deux approches suivantes :

5.2.1 Approche de Smith et al. [54] :

Dans **[54]**, Smith et al proposent d'intégrer les entreprises par la couche présentation, en plus de la couche donnée et application. Ils ont proposé, pour cela, l'utilisation de la technologie des applications web composites (ou **Mashup**).

Le Mashup désigne une application web qui combine différentes applications composites internes (CRM, ERP, applications spécifiques, …) **[54].**

La stratégie de migration basée sur les *Mashups* (**M**igtrAtion to **S**ervice **H**armonization **P**latform), comme le montre la figure II.10, est constituée de six phases, qui permettent d'adresser les issues métiers et architecturaux de migration vers un environnement SOA, comme suit :

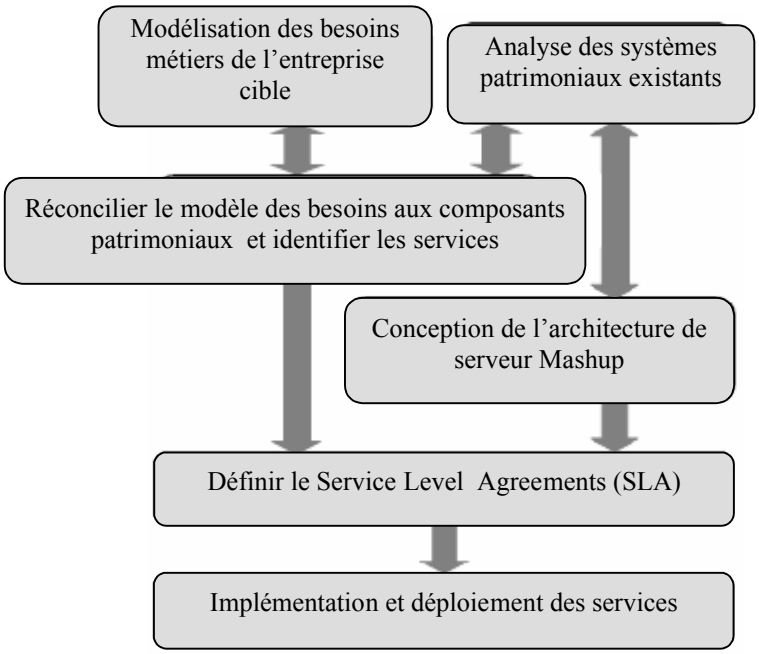


Figure II.10 : Activités de l'approche MUSHUP [54].

a. Modélisation des besoins :

L'objectif de cette phase est la spécification des besoins métiers de l'entreprise cible, afin de bien comprendre les fonctionnalités de futur système d'information intégré. Les auteurs de cette approche proposent de modéliser ces besoins en processus métiers, en utilisant le langage BPMN « ***B**usiness **P**rocess **M**anagement **N**otation* » **[42]**. Le BPMN est une notation non ambiguë de représentation des processus métiers, elle est indépendante de toute plateforme et de toute langage de programmation.

b. Analyse des systèmes patrimoniaux :

La deuxième phase est l'analyse de système patrimonial existant. Selon les auteurs l'analyse de système existant fournie les informations nécessaires à la conception d'un environnement spécifique du domaine, et le développement d'une architecture de référence pour la réutilisation des capacités et des technologies existantes. Les résultats de cette phase incluent :

- Des informations sur l'architecture de système d'information, et les composants logiciels qui le constituent ;
- Des détails d'infrastructures, et les caractéristiques des interfaces utilisateurs ;
- Des attributs de qualité de service ;
- Des informations définissant l'état courant de système, comme le niveau de maintenabilité, le niveau de complexité et de couplage de ces composants logiciels…etc.

c. Mapping et identification :

L'objectif de cette phase est le mappage des besoins et les composants logiciels du système et l'identification des services, dans la mesure où le nouveau système peut réutiliser les capacités existantes ou encore créer de nouveaux composants.

Smith et al dans cette étape, proposent d'utiliser un processus itératif. Nous avons résumé ce processus dans les étapes suivantes :

(1) . Si le besoin métier est bien satisfait par un des composants existants alors :
Adapter les composants logiciels toute en considérant les attributs de qualité de services.

(2) . Si le composant à adapter, ne retourne pas tout les résultats attendus alors :
Chercher une composition de services existant (services composite) qui répond au besoin ou,
Maintenir les services existants, en ajoutant d'autres fonctionnalités qui permettent au service de répondre à ce besoin.

(3) . Si le besoin ne peut pas être réalisé par les composants existants alors :
Créer un nouveau service qui implémente les métiers attendus.

d. Définition de Service Level Agreement (SLA):

Dans la troisième étape, la réconciliation (ou le mapping) et la définition d'un socle finale de services est finalisée. Le but de cette étape est la construction d'un référentiel qui contient les informations nécessaires décrivant le contexte et la qualité des services au sein de serveur. Le serveur Matshup gère les informations dans le référentiel pour exécuter les services publics.

e. Implémentation et déploiement :

Le but de cette phase est l'implémentation et le déploiement des services. Cette activité inclut :

- L'adaptation des composants logiciels existants ;
- L'amélioration et l'intégration des applications existantes ;
- Le développement de nouveaux services ;
- L'introduction et l'adaptation de nouveaux composants COTS.

5.2.2 Approche de Lewis et al. [53] :

G. Lewis et al. **[53]** ont présenté un processus, dit **SMART** « ***S**ervice-oriented **M**igration **A**nd **R**euse **T**echnique* ». Dans ce processus Lewis et al ont utilisé des méthodes d'analyse des options pour la technique de reingénierie (OAR: the **O**ption **A**nalysis for **R**eengineering) **[55],** pour l'analyse des capacités existantes des organisations pour la réutilisation dans un environnement SOA. L'approche SMART est composée de six étapes et un point de décision principal (voir la figure II.11), pour la migration des systèmes d'information d'entreprises vers un environnement SOA, comme suit :

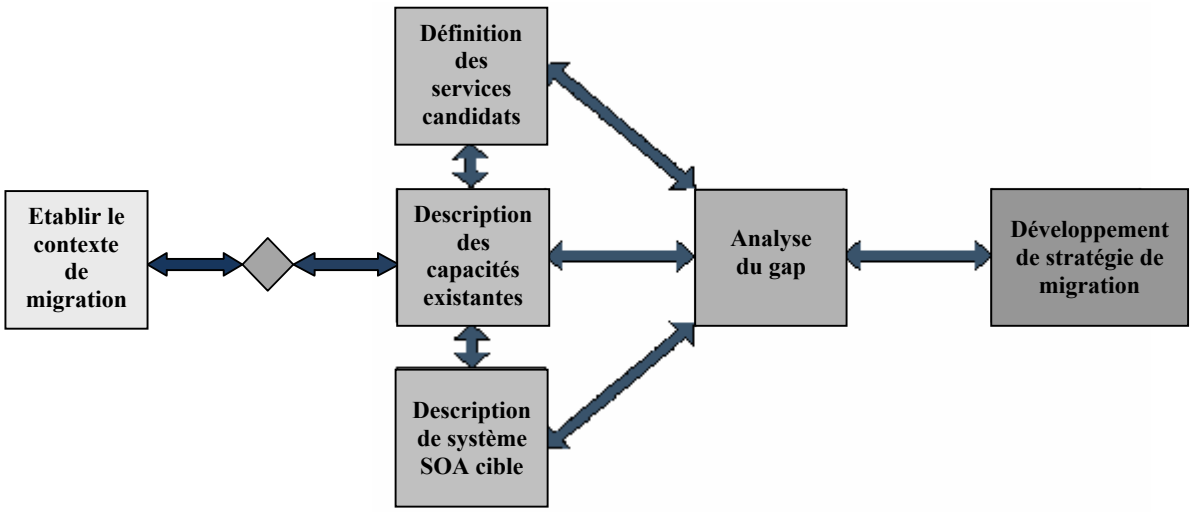


Figure II.11 : Activités de processus SMART [53].

a. Etablissement de contexte de migration :

Le processus de migration « SMART » démarre par l'étude et la compréhension des contextes métiers et techniques pour la migration, dont le but est d'identifier les objectifs et les attentes de migration et aussi la documentation de système d'information de l'entreprise. Cette phase utilise, pour cela, un guide, dit « **SMIG**[47] : *the **S**ervices **M**igration **I**nterview **G**uide* », afin de solliciter les informations sur les systèmes existants, les objectifs attendus, les utilisateurs de systèmes patrimoniaux et même les utilisateurs éventuels de futur système **[48]**. Lewis et al ont proposé de subdiviser cette phase en tâches, comme suit :

a1. Compréhension des objectifs est des attentes métiers et techniques pour lesquels le projet de migration vers un environnement SOA est démarré.

a2. Compréhension de système patrimonial. Ici toutes les informations concernant l'ancien système sont recueillies, comme les fonctionnalités principales, les technologies utilisées, l'âge, le langage de programmation…etc.

a3. Identification des fonctionnalités candidates à la migration. Les auteurs de processus ont utilisé pour la sélection du code réutilisable à la fois les résultats de l'approche *Bottom Up* basée sur l'existant, et l'approche *Top Down* guidée par les besoins métiers, pour cela ils ont proposé d' :

- Identifier les objectifs de migration ;
- Identifier les principaux processus métiers qui supportent ces objectifs ;
- Identifier les activités/tâches communes des ces processus ;
- Identifier les fonctionnalités de l'ancien système qui implémentent ces activités/tâches.

Les auteurs de cette approche, ont proposé de passer par un point de contrôle principal avant de démarrer le projet de migration. Ce point de contrôle permet de décider si le système d'information de l'entreprise, est un bon candidat pour la migration vers l'architecture de services. Si ce n'est pas le cas, l'arrêt de processus de migration à ce point est nécessaire, et permet le gain en temps, en coût et en ressources.

[47] SMIG : un guide qui dirige la collecte d'information dans les trois premières phases de SMART, il contient plus de 60 classes de questions sollicitant des informations sur : le contexte de migration, les composants de système et l'environnement SOA cible [48].

b. Définition des services candidats :

Une fois la migration de système est faisable, le processus continue par la phase de définition des services candidats. L'objectif de cette activité est l'identification et la spécification des services candidats. Il faut, pour cela, sélectionner de la liste initiale des services candidats établis dans la phase de l'étude de contexte de migration, ceux qui sont de bons candidats. Selon les auteurs de ce processus, un service est un bon candidat, si :

- Il a des entrées et des sorties claires.
- Il est utilisé pour calculer un résultat donné.
- Il a un degré d'autonomie et de réutilisation.

L'ensemble des services choisi dans cette phase sont, ensuite, bien spécifiés en définissant pour chaque service les entrées et les sorties, ainsi que les informations de qualité de service attendue (QOS[48]).

c. Description des capacités existantes :

L'objectif de cette phase de SMART est d'analyser les composants logiciels de système patrimonial existant pour collecter des informations sur les parties du code, qui implémentent la logique d'affaire des services éventuels définis dans la phase précédente. Cette phase utilise, pour cela, le guide « **SMIG** ».

d. Description de l'environnement SOA :

Cette activité indique des informations techniques et de performances attendues des services candidats de futur environnement SOA. Ces informations incluent principalement :

- Les composants logiciels qui constituent le futur système orienté services.
- L'influence des technologies spécifiques et des standards utilisées dans cet environnement.
- Les attributs de qualités de services (QOS) et de performance d'exécution de l'environnement SOA.

Toutes ses informations sont ensuite utilisées pour identifier les services nécessaires à leur réalisation. Ceci sera réalisé dans l'étape suivant d'analyse et de mappage des besoins et de l'existant.

[48] QOS : Quality Of Service.

e. Analyse de gap :

Le but de cette phase est la réconciliation des besoins identifiés dans la phase de description de l'environnement SOA avec les services identifiés dans la phase de description des capacités existantes, afin de définir les services composant l'environnement SOA.

f. Développement d'une stratégie de migration :

L'activité finale de SMART consiste à choisir la stratégie de migration pour réaliser le but attendu. Cette activité implique toutes le informations résultantes des étapes précédentes.

La stratégie de migration peut impliquer une étape simple telles que : la structuration et l'adaptation de l'application via des adaptateurs spécifiques ou standards, ou encore la modification de l'application pour créer d'autres services. Selon les auteurs de SMART, la stratégie de migration peut impliquer de multiples étapes dans le même projet. L'activité de développement d'une stratégie de migration peut inclure l'identification et la création de nouveaux services.

6. Synthèse

La migration des systèmes d'information d'entreprises vers SOA, par l'adaptation de ses fonctionnalités en utilisant les standards de services web, est une solution efficace. Mais elle est trop compliquée, cette complexité est liée principalement à certaines caractéristiques des systèmes patrimoniaux existants, et de système orienté services cible **[47]**. Le tableau suivant, tente de récapituler l'essentiel des caractéristiques de chaque classe des approches de migration vers SOA, où les symboles " √ " et " – " signifient respectivement la satisfaction ou la non satisfaction de la propriété considérée (voir Tableau II.3). Le symbole " ? " dans la sixième ligne signifie que les deux approches SOMA et SWA satisfassent partiellement la propriété. Elles proposent une démarche de décomposition des systèmes en services, mais cette démarche n'est pas pratique particulièrement dans le cas des grandes entreprises (voir l'approche de Sneed dans la section 5.1 de ce chapitre).

Approches de migration vers SOA	SMART [53]	MASHUP [54]	SOMA [52]	SWA[49] [45]
Fournit une démarche/stratégie de travail	√	√	-	√
Permet l'intégration des services/processus	√	√	√	√
Permet l'intégration de la couche présentation	-	√	-	-
Indépendante de la plateforme/outils/technologies	√	√	-	√
Permet la capitalisation de l'existant	√	√	√	√
Adopte une stratégie de composition/décomposition des services web.	-	-	?	?
Gère la complexité de l'architecture	-	-	-	-
Prend en charge l'aspect de la disponibilité de service/monté en charge/granularité de services	-	-	-	-
Qualité de services (QOS)	-	-	?	-

Tableau II.3 : Différentes approches de migration vers SOA.

Remarquons que les approches de migration existantes, notamment l'approche SMART et l'approche basée sur les MUSHUPs permettent l'intégration des systèmes par l'intégration de leurs services et processus métiers implémentant la logique d'affaire de système attendu, tout en capitalisant sur l'existant. Mais, ces approches ne permettent pas de résoudre automatiquement la complexité de l'architecture de services, engendrée par le patrimonial applicatif. Ceci, selon notre point de vue, revient à la stratégie adopter pour la décomposition de système en services, et aussi pour la recomposition des services web en processus métiers.

Pour tirer avantage des deux approches, SMART [53] [48] et Mashup [54], et pallier leurs insuffisances, nous avons procédé dans notre travail (voir chapitre III) à leurs combinaison en intégrant les étapes nécessaires à la décomposition et la composition des services web de manière efficace réduisant la complexité de système de services cible.

[49] SWA : Salvaging & Wrapping Approach.

7. Conclusion

La migration des systèmes d'information d'entreprise est considérée comme la meilleure solution permettant l'intégration des applications dans le périmètre de l'entreprise ainsi qu'à l'extérieur de son périmètre. Elle permet de garder et réutiliser les parties intéressantes qui supportent les besoins métiers de système intégré et en même temps d'accompagner l'évolution technologique. Dans ce chapitre, nous avons présenté plusieurs travaux permettant la migration des systèmes d'information d'entreprises vers une architecture orientée services.

Comme nous avons vu, il existe principalement quatre grandes classes d'approches de migration : l'approche *Top Down*, l'approche *Bottom Up*, l'approche *Outside In* et l'approche *Middle Out*. Ces approches se distinguent principalement par la stratégie d'évolution et de développement adopté dans un environnement SOA.

Le chapitre suivant, sera consacré à la présentation des différentes activités de notre processus de migration des systèmes d'information d'entreprises vers une architecture orientée services web en adoptant l'approche *Meet In The Middle*. Nous allons donc détailler le rôle de chaque activité à part, et les techniques utilisées dans chaque phase de ce processus.

CHAPITRE III

Un Processus de Migration vers une Architecture Orientée Services Web

1. Introduction

Comme nous avons signalé dans le chapitre précédent, plusieurs travaux ont été proposés, permettant la migration des systèmes d'information d'entreprises vers les architectures orientées services, particulièrement celui de l'approche « *Meet In The Middle* ».

Selon notre point de vue, ces travaux présentent deux inconvénients majeurs. Le premier est lié à la complexité du système basé sur les services : dans la mesure où ces approches ne permettent pas de résoudre automatiquement la complexité engendrée par le patrimonial applicatif de l'entreprise, elles se contentent à exposer les fonctionnalités de ce système sous forme de services web. Par conséquent les entreprises se trouvent toujours en face du problème de complexité, à titre d'exemple, la complexité liée à la gestion d'un grand nombre de services due à l'existence de services redondants et de services sans valeur ajoutée métier. Le deuxième est le manque de standardisation dans les phases de ces approches **[37], [45]**, particulièrement la phase de détection des blocs fonctionnels qui implémentent les services web. Dans la mesure où, les approche qui adoptent une stratégie de détection des services web, sont trop liées au code source du système.

La complexité du système orienté services et le manque de standardisation dans les phases de processus de migration sont donc les problèmes clés de notre travail.

Pour pallier ces désavantages, nous proposons un processus générique en adoptant l'approche ***Meet In The Middle*** présentée dans le chapitre précédent. Ce processus permet une intégration flexible des entreprises par la migration de leurs applications vers un seul système de services web cohérent et dont la complexité est réduite automatiquement. Nous y utilisons des Modèles d'Entreprise (ou **EM**[50]) comme un langage intermédiaire dans la phase d'analyse de l'existant, pour bien comprendre les applications et aussi, pour unifier la description et étendre la gamme des applications à migrer. Nous proposons aussi une procédure d'identification des services centrée sur les besoins, permettant de localiser et extraire les services de bon niveau de granularité.

Avant d'exposer en détail les différentes activités qui composent notre processus, nous avons choisi de commencer ce chapitre par l'introduction des différents concepts clés de notre processus qui sont la granularité de service et les modèles d'entreprise que nous avons utilisé pour faciliter la compréhension de l'existant et unifier la manière de le présenter, et donc de faciliter l'identification des services web. Une vue globale de notre processus sera proposée

[50] EM : Enterprise Modeling.

dans la section qui suit, puis nous détaillons chaque phase de ce processus à part, en présentant le rôle de chacune dans le processus de migration. Une discussion est proposée ensuite, pour motiver notre choix en comparant notre processus avec d'autres déjà vus dans le chapitre précédent. Ensuite, nous manifestons les bénéfices qu'apporte notre travail dans le cadre du problème posé. Enfin, nous terminons ce chapitre par une conclusion.

2. Concepts clés

Avant d'entamer le détail de notre solution du problème de migration des systèmes d'information d'entreprises vers l'architecture orientée services web en vue de leur intégration, nous exposons d'abord les notions des différents concepts clés utilisés dans cette approche qui sont, la granularité de service et les modèles d'entreprise.

2.1 Services et granularité de services :

Comme nous avons signalé dans le chapitre précédent, les approches de migration existantes ne permettent pas de résoudre automatiquement la complexité de système basé sur les services résultant. Ceci, selon notre point de vue, revient aux stratégies inefficaces adoptées pour la décomposition et la composition de système existant en des fonctions qui représentent par la suite les services web de ce système. Pour cette raison, nous tentons dans cette section, de discuter la notion de granularité des services particulièrement dans le domaine de migration, et les stratégies efficaces de décomposition des anciens systèmes en services de bon niveau de granularité.

Dans une architecture orientée services, la responsabilité d'un service doit être définie clairement. La granularité de service permet de mesurer le niveau de détail de l'entité fonctionnelle définissant un service. Une autre définition, consiste à présenter la notion de granularité de service comme étant le périmètre fonctionnel couvert par le service. On peut donc classifier les services selon leur degré de granularité à **[44]**:

- *Des services à grain fin*: ce sont des services unitaires encapsulant des fonctionnalités de calcul simple fondées sur des règles métier, des fonctions d'accès comme conversion de devise.
- *Des services à gros grain*: ce sont des services complexes, des fonctionnalités encapsulant un processus métier comme une demande de prêt bancaire.

Par définition **[44]**, dans une architecture orientée services la mauvaise granularité est qu'un service couvre trop ou trop peu de fonctionnalités, ceci peut conduire à :

- Des mauvaises performances ;
- Une faible capacité de réutilisation ;
- Une architecture complexe : existence des services redondants et des services sans valeurs ajoutée métier.

Les principales causes de ces travers sont :

- L'adoption d'une démarche de conception «full Top Down» ou «full Bottom Up».
- Les difficultés liés à la compréhension de système patrimonial existant.
- Une perspective trop étroite du métier de l'entreprise.

En effet la granularité des services influe de manière directe sur la simplicité et les performances de l'architecture du système migré. Par conséquent, l'adoption d'une démarche stratégique efficace dans un processus de migration pour la définition des services est nécessaire. Ce dernier point est l'objet de notre projet, où nous proposons d'utiliser une procédure de localisation et d'identification des services basée sur les modèles d'entreprise. A la suite de ce chapitre, nous expliquons l'application de cette procédure pour éviter les inconvénients de la mauvaise granularité.

2.1.1 Décomposition des services :

La décomposition des systèmes existants en services est une étape essentielle dans le projet de développement d'une SOA, ce que justifie l'existence d'un certains nombre de travaux qui se concentrent principalement sur la décomposition des systèmes d'information d'entreprises en unités fonctionnelles qui peuvent être représentées comme des services web.

Parmi les travaux proposés dans le domaine de décomposition des systèmes d'information d'entreprises, nous prenons par exemple celui de H. Sneed dans **[37]**, qui propose une approche de création des services web à partir des systèmes patrimoniaux basée sur la détection du code qui implémente la règle métier. Il utilise pour cela les résultats produits par les règles métiers, en identifiant d'abord les valeurs retournées par les fonctions qui les implémentent. Ensuite, les flux de données sont analysés, en commençant par les fonctionnalités qui génèrent les résultats finaux et en revenant vers les instructions qui interviennent dans la production de ces résultats finaux. L'approche de Sneed n'est pas pratique et complexe particulièrement dans le cas des grandes entreprises qui possèdent un

large système d'information puisque, elle est trop liée au code source de système. Aussi elle ne prend pas en charge les besoins métier de futur système, elle permet de décomposer le système en services (règle métier) qui n'ont pas de valeur ajoutée métiers au niveau l'entreprise.

Une approche de J. Ziemann et al **[41]** propose une solution basée sur les techniques de l'ingénierie inverse (ou Reverse Engineering), et les modèles d'entreprises (voir section 2.2 de ce chapitre), pour résoudre le problème de décomposition des systèmes patrimoniaux en fonctions (services web), en spécifiant et représentant les applications des systèmes patrimoniaux dans un modèle d'arbre des fonctions (en anglais : ***fonction tree model***). Ce modèle est une abstraction hiérarchique de système d'information, il représente une structuration hiérarchique de toutes les fonctions du système, où chaque nœud représente une fonction, et chaque sous nœud représente une sous fonction. L'étape suivante consiste à la sélection des fonctions réutilisables qui implémentent les besoins de système pour leur créer une interface WSDL et les publier comme étant des services web.

D'autres d'approches utilisent pour la décomposition des anciens systèmes en fonctions (services web), une solution classique permettant la combinaison des méthodes de conception "*bottom up*" et "*top down*", mais cette solution présente des difficultés notamment dans l'étape de réconciliation des résultats obtenus des deux méthodes. Cette étape dite « accostage » est très compliquée et très coûteuse.

2.1.2 Meilleur niveau de décomposition des services :

Le fait de trouver la bonne granularité de services n'est pas aussi simple que de suivre une procédure. Le fait de trouver le bon équilibre exige des connaissances, de l'expertise et l'expérimentation de différents systèmes ainsi leurs domaines d'application. Voici cependant quelques lignes directrices qui permettent d'éviter certains aspects qui mènent à une mauvaise granularité et donc à une architecture complexe difficile à gérer :

- Les services devraient être décomposés en ensemble de "plus petites unités réutilisables", chacune de ces unités devant rester compréhensible par les gens du métier. De plus, un service ne doit pas être une composition de services à grains plus fins qui ne sont pas disponibles en tant que services distincts. Nous devons laisser ouverte la possibilité de composer d'autres services.

- D'autre part, il est important que la responsabilité d'un service soit clairement définie. Or, si les services sont définis avec un grain trop fin, il sera très difficile de les assigner à un propriétaire même à un utilisateur.
- Enfin, un service doit rester aussi autonome que possible. Si quelqu'un veut utiliser un service, il ne doit pas être contraint à utiliser d'autres services, par contre cela n'empêche pas le service en question de faire appel à d'autres services.

2.1.3 Composition des services :

La composition de service est une activité qui permet d'intégrer un ensemble de services web de façon à fournir un nouveau service web à forte valeur ajoutée métier. L'intégration de services web dans un seul flux de contrôle peut être simplement une séquence, mais aussi une exécution parallèle ou conditionnelle.

2.2 La Modélisation en Entreprise :

Le concept de « la Modélisation en Entreprise » (ou EM, pour Enterprise Modelling) a été introduit pour la première fois au milieu de l'année 1970 dans le domaine du génie logiciel pour la conception et le développement des systèmes d'information **[22]**. Ce paradigme a été introduit une autre fois vers la fin de l'année 1990 dans le domaine de réingénierie[51] pour la restructuration et l'amélioration des systèmes d'entreprises existants **[25]**. A partir de cette date plusieurs définitions de ce concept ont été proposées, et nous avons choisi quelques unes :

« La Modélisation en Entreprise est l'art de capture et d'externalisation des connaissances définissant une entreprise pour ajouter de la valeur métier à l'entreprise, partager l'informations entre les différents partenaires et pour représenter toutes les entités de l'entreprise : les informations, les ressources, et la structure de l'organisation » **[24]**.

« Les modèles d'entreprise (ou Enterprise Models) est une méthode de réingénierie des systèmes d'information des entreprises (ou une partie de système), permettant de fournir une compréhension claire des informations gérées par le système et des processus des applications, en utilisant les différents modèles d'entreprise (Informationnels, fonctionnels, des ressources, …etc.). Cela inclut les modèle du domaine d'application de l'ensembles des systèmes constituant une entreprise » **[57]**.

[51] La Reingénierie des Systèmes d'Information maintient l'efficacité et la valeur métier de ses fonctionnalités, qui sont aussi bien maintenues (modifiées) pour améliorer ses performances, et les adapter aux environnements changeants [41].

« Les modèles d'entreprise est une représentation abstraite des différentes structures (c-à-d des entités et des relations entre eux) comme : les activités, les processus, les acteurs (ou utilisateurs), les informations, les ressources, le comportement, les objectifs d'une entreprise et aussi de ces partenaires » **[56]**.

Il existe plusieurs méthodes de modélisation en entreprise, utilisent plus de 300 modèles d'entreprise **[41]**, parmi lesquels nous citons :

2.2.1 Modèles d'architectures : (Enterprise Architectures)

La modélisation de métier de l'entreprise et de son environnement permettant de faciliter la création, la compréhension et l'amélioration de ses systèmes et ses processus métiers et plus particulièrement les relations entre eux. Ceci inclut la modélisation de toute les entités de l'entreprise : composants, ressources, processus métiers, objectifs, besoins, utilisateurs, fonctions …etc. ce type de modèles peuvent être aussi utilisé pour la modélisation des affaires externes de l'entreprise et les relations entre eux. Ce type de modèle est le plus communément utilisé pour la description fonctionnel de l'entreprise comme : ARIS et Zachman **[59].**

2.2.2 Avantages des modèles d'entreprises :

Les bénéfices qu'apporte l'utilisation des Modèles d'Entreprises dans le processus de réingénierie des systèmes d'information d'entreprises se résument aux points suivants **[56] [57]**:

- Elle permet une meilleure conception des entreprises ;
- Elle permet une présentation correcte d'une spécification par des modèles (ou notation) de haut niveau d'abstraction ;
- Les spécifications décrites par des modèles capturent correctement le comportement interne d'un programme, et donnent une vision globale et claire des fonctionnalités de l'entreprise et même d'un ensemble d'entreprises interdépendantes ;
- La notation sémantique des modèles est clairement définie et facilement comprise par le groupe des concepteurs et des développeurs distribués ;
- La logique d'affaires d'un système décrit par des modèles peut être comprise et maintenue facilement comme dans **[41]** ;
- L'utilisation des modèles fournit un niveau élevé d'intégration et d'interopérabilité à travers le développement des modèles et des relations entre les modèles ;

- L'utilisation des modèles facilite la maintenance et l'amélioration des performances de métier d'une entreprise.

3. Processus proposé

La solution que nous proposons offre un processus qui recouvre toutes les étapes nécessaires à la migration des applications des systèmes d'information d'entreprises vers une architecture de services web (voir Figure III.1). Dans notre processus, nous avons procédé à la combinaison des deux approches, G. Lewis **[48], [51]** et Smith **[54]**, en intégrant la spécification et la description des besoins et des capacités existantes du système en des modèles pour la détection directe et efficace des parties intéressantes qui implémentent le service web de bonne granularité.

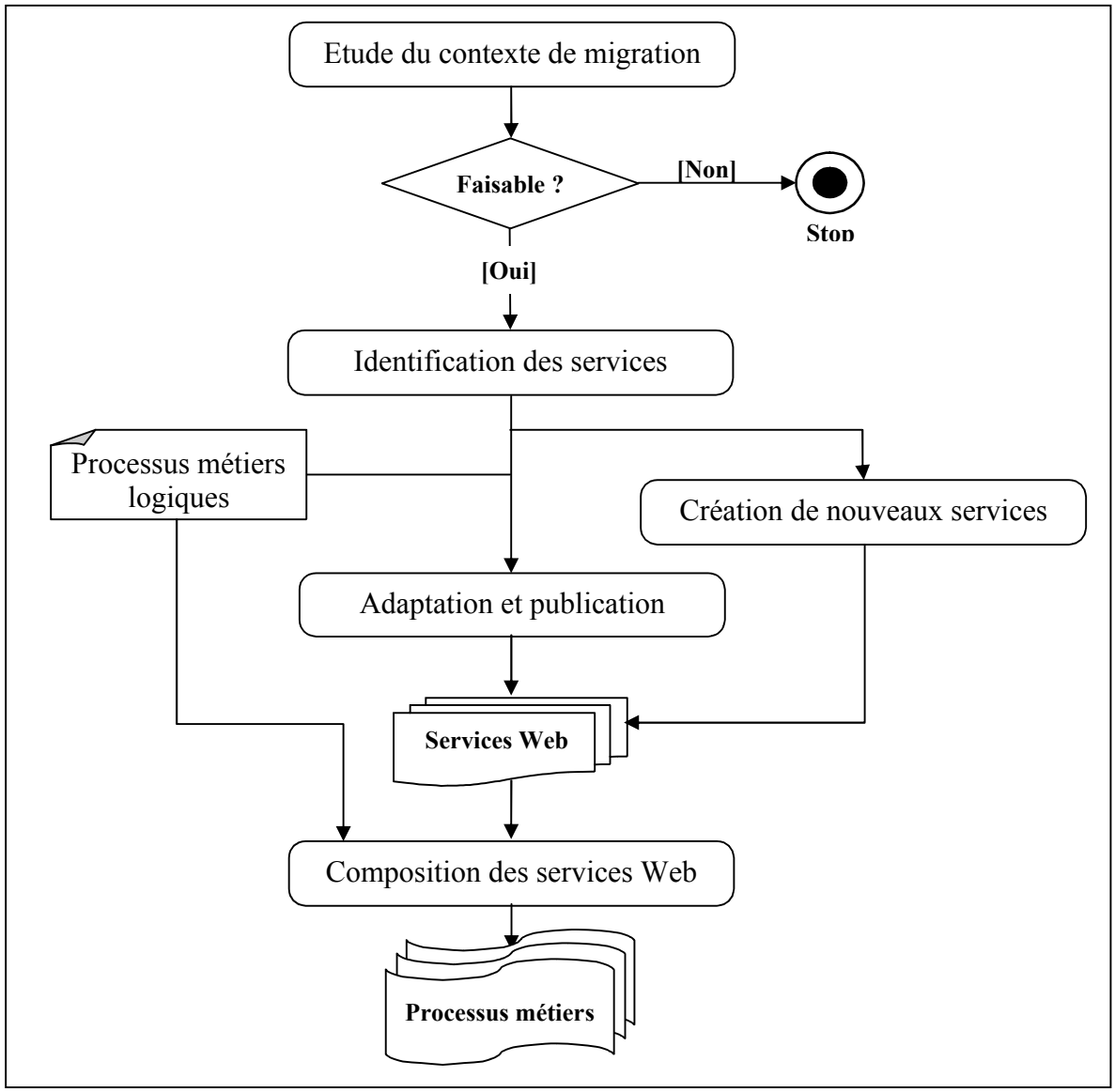


Figure III.1 : Processus de migration des applications d'entreprise vers une architecture orientée services web.

Le processus commence par une étude de contexte de migration, qui est considérée comme une étape préparatoire de la phase de l'étude de faisabilité, où nous déciderons s'il est possible ou encore rentable de continuer le projet de migration vers un environnement SOA.

Ces deux premières phases sont très importantes pour construire une base solide de migration, d'où la compréhension claire et approfondie de l'existant et des nouveaux besoins est très nécessaire. Dans le cas favorable, nous entamons la phase d'identification des services. De ces derniers nous distinguons les services qui existent dans le système et qui doivent être extrait du code de ce système et les autres qui n'existent pas dans le système et qui doivent être implémentés à nouveau Après la détection du code qui implémente les activités communes des processus de l'entreprise, vient l'étape d'adaptation et de publication qui produit à la fin un ensemble de services web. L'intégration des entreprises se réalise par l'intégration de leurs applications via des interfaces de services web. L'étape finale est alors l'intégration des services web dans un flux de contrôle central.

Dans notre processus les résultats d'une activité sont les entrées de l'activité suivante. Pour cette raison, nous suggérons comme dans **[53] [54],** que les activités de notre processus sont itératives, dans la mesure où le résultat d'une activité peut poser un problème qui nécessite de revoir l'activité précédente pour plus d'information.

4. Différentes activités de processus

Le processus de migration des systèmes d'entreprises vers une architecture orientée services web, que nous proposons, est constitué de cinq phases et un point de décision principal (lié à la faisabilité), comme il est présenté dans la figure III.1.

Dans cette section nous présentons en détail les différentes activités définissant notre processus (Figure III.1) :

4.1 Etude du contexte de migration :

L'étude du contexte de migration est une phase essentielle pour construire une base solide de migration. L'objectif de cette phase est la compréhension et l'analyse de contexte dans lequel démarre le projet de migration. Cette phase peut être subdivisée en deux étapes :

La première est l'évaluation de système à migrer : l'étude de l'existant est une étape essentielle pour une réutilisation éventuelle. Elle permet d'indiquer l'état actuel d'un système,

dans la mesure où il est utilisé pendant des années, et pratiquement subit plusieurs reprises de maintenance pour des raisons métiers et/ou techniques.

La deuxième est la compréhension et la définition des objectifs attendus de projet de migration, pour bien comprendre les besoins fonctionnels de nouveau système intégré et la manière dont ces besoins sont réalisés.

En réalité cette activité est une phase préparatoire de la phase de l'étude de faisabilité, dont des informations sur les fonctions principales de systèmes existants et des informations sur les besoins et les contraintes fonctionnelles de système attendu sont recueillies.

4.2 Etude de faisabilité :

Le but de cette phase est de vérifier s'il est réellement possible de faire migrer un système existant vers un environnement SOA distribué, en se basant sur l'étude établit dans la phase précédente. Cette étape nécessite une meilleure compréhension de systèmes existants, ainsi qu'une documentation approfondie des besoins de futur système intégré. Pour cela, nous proposons d'utiliser les modèles d'entreprises pour la présentation de système existant et des besoins attendus à un niveau d'abstraction plus élevé.

La représentation de système existant en des modèles est utile pour la décision si une fonction donnée implémentée dans un système, mérite d'être un service public dans une architecture orientée services ou non.

Cette phase est composée de deux tâches successives : la modélisation et l'étude de faisabilité :

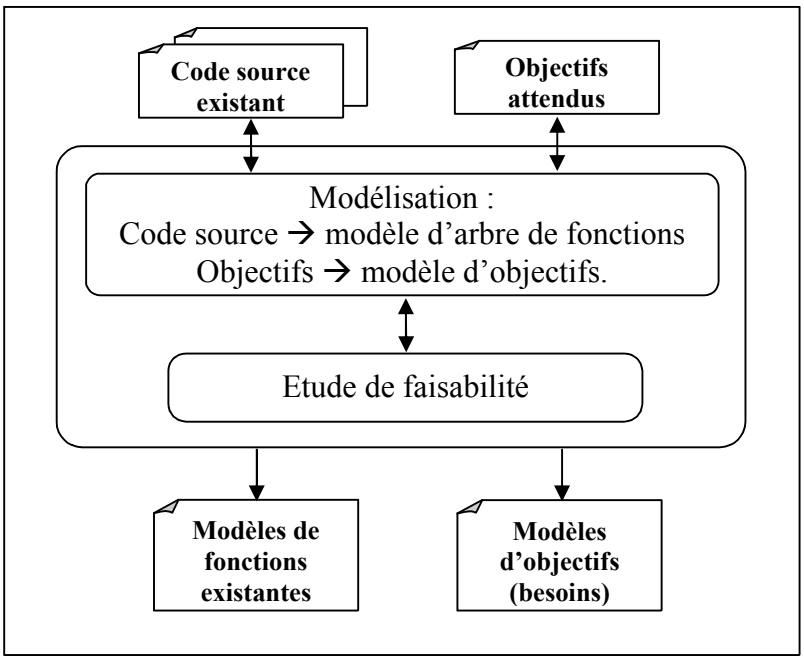


Figure III.2 : Phase de l'étude de faisabilité.

a. Modélisation :

Le but de cette phase consiste à représenter graphiquement les fonctionnalités de système existant. La description des besoins de système cible en des modèles est aussi réalisée dans cette étape. Nous proposons, pour cela, d'utiliser une structuration hiérarchisée de toutes les fonctionnalités de système à migrer et des objectifs attendus de futur système **[41]**, dont le but est d'identifier les composants de système et les relations entre eux et de créer une représentation claire des besoins attendus. L'utilisation des modèles pour la présentation des entreprises est très utile, puisqu'elle permet une description abstraite des applications de systèmes à migrer et donne une vision globale de ses processus métiers et la manière dont ils sont réalisés et, donc de faciliter leur compréhension, et d'unifier la stratégie de détection du code réutilisable pour la phase suivante.

Ziemann et al **[41] [58]** ont présenté plusieurs travaux concernant la modélisation des programmes patrimoniaux de différents langage de programmation basée sur le modèle d'arbre de fonctions (ou **Function Tree Model**). Ces travaux ont permis de définir une stratégie simple de détection du code réutilisable des systèmes patrimoniaux de différents langages de programmation. L'idée générale vise à représenter le programme dans un arbre de fonctions. Ce modèle présente une structuration hiérarchisée de toutes les fonctionnalités d'un système, dont chaque nœud représente une fonction et chaque sous nœud représente une sous fonction. Le code réutilisable représente les fonctions et/ou activités communes des applications de système.

En ce qui concerne notre travail, nous proposons d'utiliser une structuration hiérarchisée, le modèle d'arbre de fonctions, pour :

1. La description de toutes les fonctionnalités des systèmes existants, de façon à ce que:
   - La racine de cet arbre représente la fonction principale «main ( )» d'un système d'information ou d'une application d'une entreprise,
   - Chaque nœud de cet arbre représente une fonction,
   - Chaque sous nœud représente une sous fonction qui implémente la fonction principale,
   - Les feuilles de cet arbre représentent les instructions qui interviennent dans l'implémentation des fonctions représentées par les nœuds de niveau supérieur.

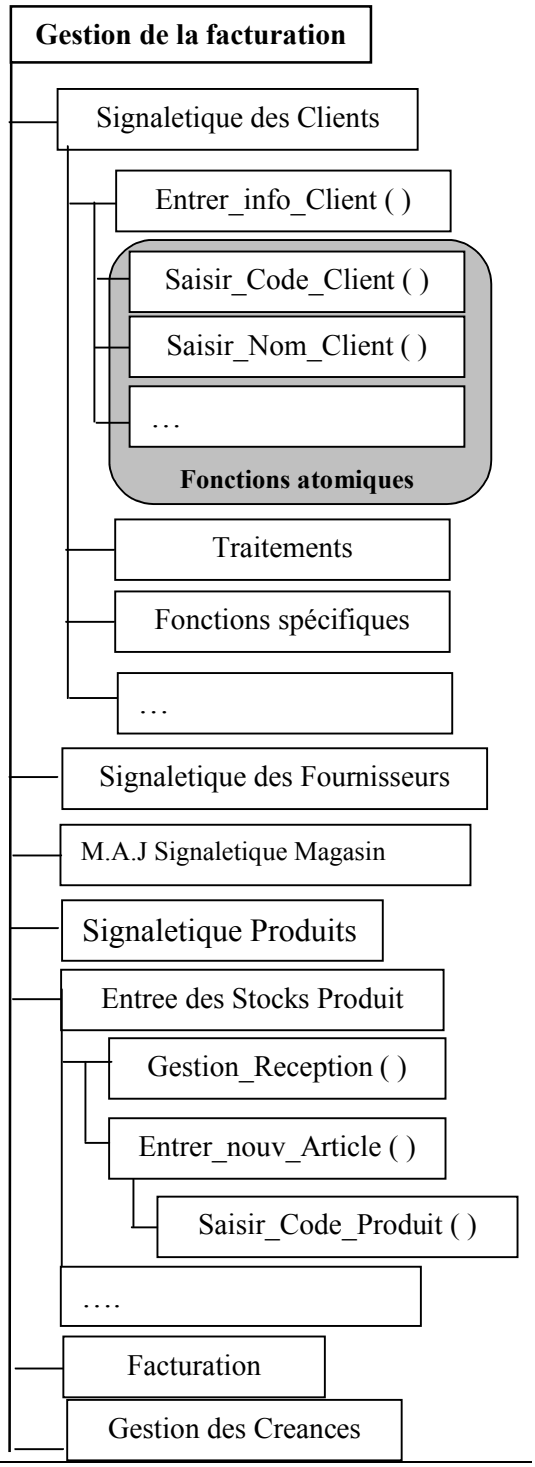
2. La description des besoins de système cible, en un modèle hiérarchisé similaire à celui de système existant, dont chaque nœud représente un objectif et chaque sous nœud représente un sous objectif permettant la réalisation de l'objectif principale.

Nous fournissons un exemple qui représente un programme écrit en langage Basic, le programme **(a)** est une application générée par un générateur du code Basic "TOP KEY". Si nous modélisons ce programme nous pouvons arriver à ce modèle **(b)** :

(a)

```
1       ' --- TOP KEY ---              SP00  #THR
2       CALL FBR3
99      REM $INCLUDE: 'MLSIV'
900     ' ----- nom du programme
905     CVNPROG="SP00"
999     REM $INCLUDE: 'THW02'
1500    ' ----- debut
1600    GOSUB 52150
1610    GOTO 47000
10000   ' ----- titre du menu
10005   CV9(1)="GESTION DE LA FACTURATION"
10050   RETURN
10500   ' ----- fin du menu
10505   GOSUB 52270
10510   CLOSE
10515   IF CRET="" THEN CALL FBR4:END
10520   CU1=CRET:GOSUB 59305:GOTO 53000
11000   ' ----- retour vers menu principal
11050   RETURN
11500   ' ----- mot de passe
11550   RETURN
12000   ' ----- code texte a inserer
12005   ITXT=5000
12050   RETURN
12500   ' ----- programmes
12505   CV6(1)="1,SP01,Signaletique des Clients"
12506   CV6(2)="2,SP02,Signaletique des Fournisseurs"
12507   CV6(3)="3,SP04,M.A.J Signaletique Magasin"
12508   CV6(4)="4,SP05,Signaletique Produits"
12509   CV6(5)="5,SP11,Entree des Stock Produit"
12510   CV6(6)="6,SP12,Facturation GROS"
12511   CV6(7)="7,SP17,Facturation Detail"
12512   CV6(8)="8,SP13,Facture A V O I R"
12513   CV6(9)="9,SP14,Facture Proformat"
12514   CV6(10)="10,SP15,Bon de Livraison"
12515   CV6(11)="11,SP16,Bon de Sortie"
12516   CV6(12)="12,SP30,Editions des Etats"
12517   CV6(13)="13,SP06,Gestion des Creances"
12518   CV6(14)="14,SP07,Situation des Creances"
12519   CV6(15)="15,SP40,S a u v e g a r d e"
12520   CV6(16)="16,SP88,Transfert Facture Gros"
12521   CV6(17)="17,SP90,Transfert Facture Detail"
12900   RETURN
24099   REM $INCLUDE: 'THW62'
39999   REM $INCLUDE: 'THW57'
40199   REM $INCLUDE: 'THW57S'
46999   REM $INCLUDE: 'THW61'
```

(b)



Listing III.1: Modélisation du code source.

Dans cet exemple, la fonction main de l'application «Gestion de facturation» représente la racine du modèle. L'application contient plusieurs fonctions comme: *Signaletique des Clients*, *Signalitique des Fournisseurs* ( ), *Facturation* ( ) …etc. qui représentent les nœuds du modèle. Les feuilles : *Saisir_Code_Client* ( ), *Saisir_Nom_Client* ( ), …etc. représentent les instructions qui implémentent la sous fonctions *Entrer_info_Client* ( ).

En effet, si deux entreprises ou plus, utilisateurs de ce processus, désirent migrer leurs systèmes d'information écrits dans des langages de programmation différents, il suffit seulement d'intégrer un modeleur (crée des modèles d'arbre de fonctions du code source) qui permet de fournir une vue globale des fonctionnalités des entreprises, en intégrant leurs modèles.

Nous fournissons dans la figure III.3 un exemple d'un modèle d'objectifs (Goals Model) d'une entreprise commerciale **(c).**

b. Etude de faisabilité :

Cette étape permet de répondre aux questions suivantes :

« *Est-il possible de migrer ce système vers une architecture orientée services web* ? ». La réponse à cette question va nous permettre de décider si le système contient des fonctions qui méritent d'être des services web ou non.

Et si cette migration est possible, « Est-elle rentable ? ». La réponse à cette question permet de vérifier si le système de services répond aux besoins métiers de l'entreprise ou non.

La réponse à ces deux questions est donc déterminée d'après la vérification de certaines conditions **[48] [53] :**

1. Le système contient-t-il des fonctionnalités réutilisables et fiables, englobant une logique d'affaire qui répond aux besoins de système orientée services ?
2. La maintenance des ces fonctionnalités est-elle assez bénéfique par apport à la maintenance de système entier ?
3. Ces fonctionnalités sont-elles utiles pour être exposées en tant que services web indépendants ?
4. Les objectifs de migration sont-ils clairs et partagés entre les utilisateurs de système d'information de l'entreprise, ou encore entre les collaborateurs d'affaires ?
5. Les services web candidats sont-ils capables de répondre aux besoins d'intégration de l'entreprise ?

nous proposons pour vérifier ces conditions d'analyser les modèles établis dans la phase précédente, en associant les nœuds de modèle d'objectifs **(c)** à la fonction de modèle d'arbre de fonctions permettant la réalisation de cet objectif **(b),** telle que présenté dans la figure III.3.

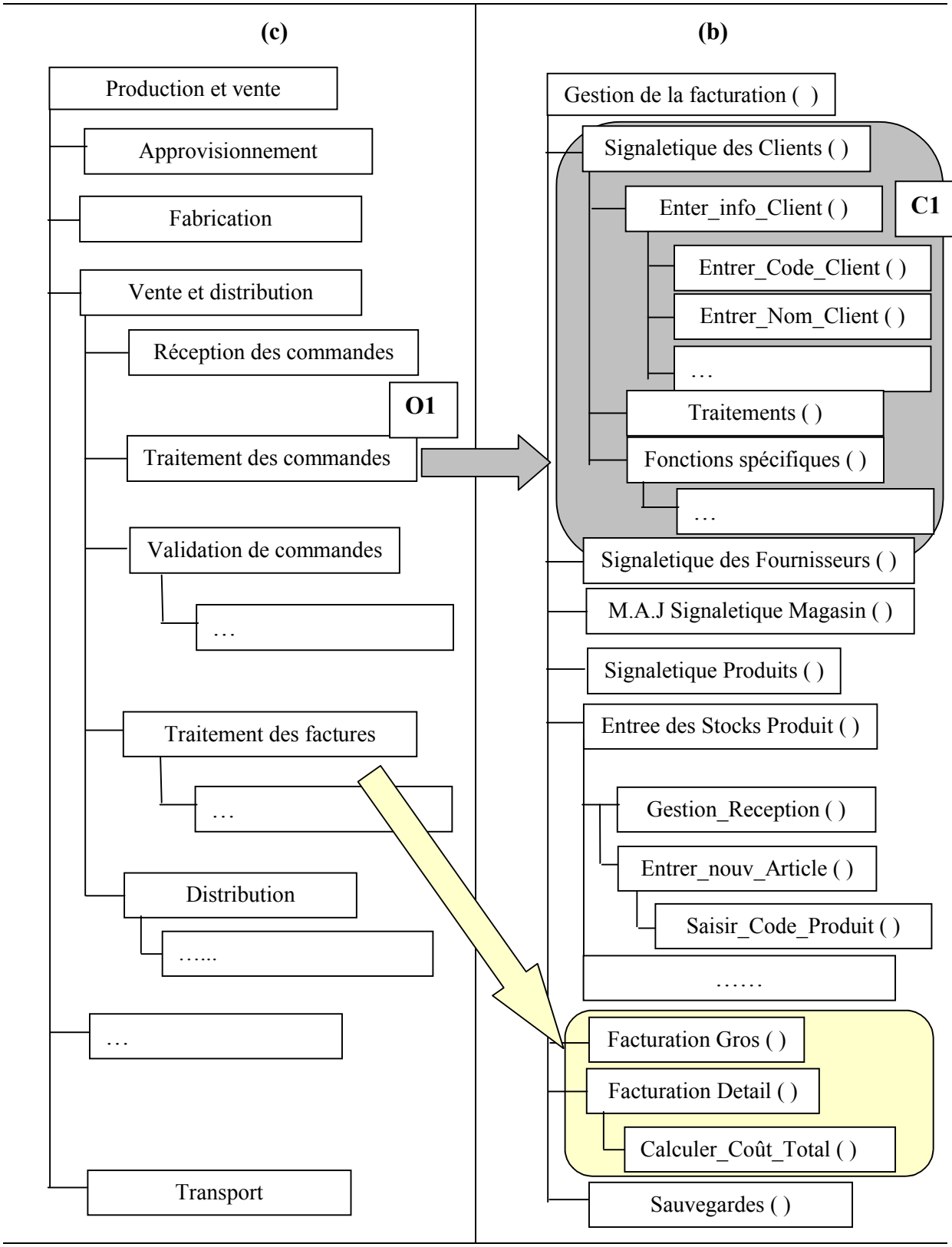


Figure III.3 : Détection des services web éventuels.

En effet dans cette définition d'une première liste de services web éventuels, seul les composants permettant d'offrir ces services sont détectés. On ne précise pas quelles fonctions ou tâches (actions) réalise effectivement une activité d'un objectif donné – comment un processus métier est réalisé.

Dans l'exemple présenté ci-dessus le composant **C1** réalise l'objectif **O1**, mais à ce niveau on ne sait pas précisément quelle fonction de composant **C1** permettant effectivement la réalisation de l'objectif **O1**.

A l'issue de cette phase d'analyse, nous obtenons deux modèles à savoir le modèle d'arbre des fonctions de système existant et le modèle d'objectifs. Nous obtenons aussi une première liste des services web éventuels.

Pour arriver à des meilleurs résultats à l'issue de ces deux phases, nous proposons aussi d'impliquer les utilisateurs des systèmes de l'entreprise et l'expérience des experts du domaine dans les activités de compréhension et d'analyse de l'existant et des besoins de futur système intégré, pour les raisons suivantes :

- Les utilisateurs des systèmes existants dans l'entreprise à intégrer 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 ajouter dans les nouveaux systèmes
- Ils seront 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.
- Les experts du domaine sont aussi indispensables, puisque ils permettent, à l'aide des utilisateurs, de documenter les besoins sur un ensemble de systèmes dans le même domaine d'application.

En conclusion, l'intégration des expériences des utilisateurs des systèmes existant peut avoir une grande influence dans l'amélioration des résultats de ces étapes et du projet entier.

Une fois que ces conditions soient satisfaites, nous passons à la phase suivante.

4.3 Identification des services :

La problématique essentielle posée dans cette étape est comment détecter les fonctions principales qui implémentent effectivement un objectif donné. Ce dernier point est le but de cette phase, qui consiste à détecter et extraire les bons candidats de services web[52]. Nous proposons dans cette étape une procédure de projection, qui permet de détecter et d'extraire des services web de bon degré de granularité à partir d'un composant logiciel. L'idée principale de cette procédure, consiste à appliquer l'approche de localisation et d'extraction des services dite « Code stripping » présentée par Sneed **[45]** sur les modèles fonctionnels des composants logiciels de la manière suivante :

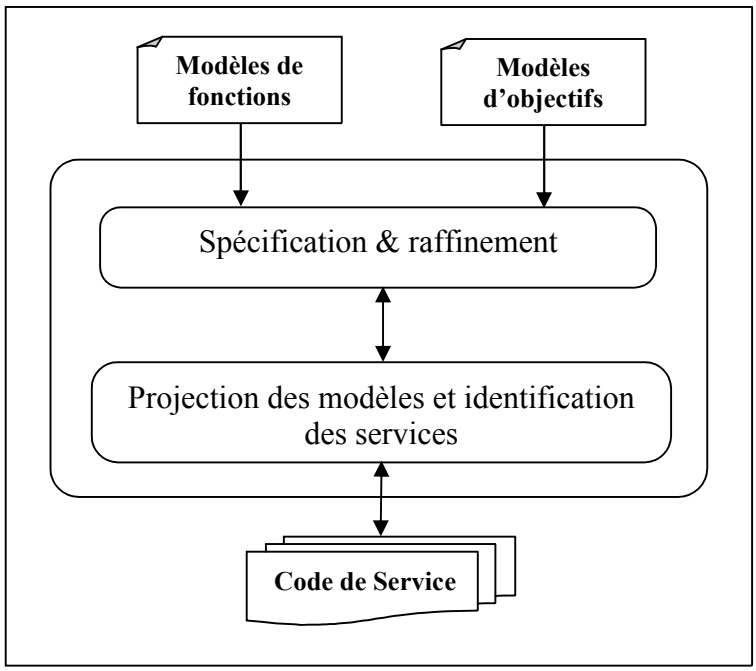


Figure III.4 : Phase d'identification des services web.

Cette phase consiste à deux étapes successives, sont : la spécification des composants. Ceci est effectué par la spécification des entrées ainsi que de la sortie finale de chaque fonction d'un composant, comme le fais Sneed dans **[37]**. Ensuite l'application de l'approche de projection sur ces modèles pour l'extraction du code minimal adéquat (voir la figure III.4)

[52] Selon notre point de vue un services web est un bon candidat, s'il a une valeur ajoutée métier équivalente à un objectif donné.

a. Spécification et raffinement :

Cette étape utilise les résultats de la phase précédente. Elle consiste principalement à raffiner les modèles des fonctions des composants logiciels de systèmes existant, dont le but général est de faciliter les étapes suivantes, surtout l'étape de projection et d'identification des services qui sont considérées de bons candidats. Dans cette phase, nous compléterons la spécification de toutes les fonctionnalités représentées dans les modèles d'arbre fonctionnel, en analysant les flux de données entre eux. Nous représentons pour chaque fonction les entrées et les sorties finales, de ce fait il sera possible de détecter où et comment un résultat d'un objectif donné est calculé et aussi de déterminer quels sont les arguments utilisés. Cette spécification est aussi utile pour faciliter la description du code de futurs services web dans un document WSDL dans l'étape suivante.

Le modèle d'objectifs qui représentent les processus métiers de système, est aussi raffiné dans cette étape, en définissant des processus métiers élémentaires (voir le modèle **(a)** dans la figure III.5) pour chaque objectif ou sous objectif donné.

b. Projection des modèles et identification des services :

Le but de cette étape est de déterminer les fonctions qui peuvent être exposé comme étant des services web. Nous avons proposé, pour cela, une procédure de projection des modèles. Cette procédure permet d'associer les nœuds du modèle fonctionnel d'un bloc applicatif aux nœuds du modèle d'un processus métier élémentaire (voir chapitre III.5).

Les étapes de l'approche de détection de services web basée sur la projection sont résumées comme suit :

1. La première tâche est la sélection d'une activité d'un processus métier élémentaire, cette dernière est associée à un composant qui contient les fonctions réalisant cette activité.

2. Commençons par les feuilles (instruction) de composant sélectionné, les variables du résultat final retourné par cette instruction sont donc comparées aux variables de résultat retourné par l'activité de processus sélectionnée.

3. Si l'instruction sélectionnée retourne peu de variables par apport à l'activité, il est nécessaire de sélectionner le nœud qui se trouve à un niveau hiérarchique supérieur.

4. Dans le cas contraire, c'est-à-dire les variables retournées sont équivalentes à celles de l'activité sélectionnée, la fonction est donc définie comme étant le futur service public qui implémente l'activité sélectionnée de processus élémentaire.

5. Répéter les étapes 1-4 pour tous les nœuds de modèle de processus élémentaire.

Nous avons résumé ces étapes dans l'algorithme suivant :

Algorithme d'extraction de code des services web

```
Début
  MF : Modèle de fonction de code de système à traiter.
  MO : Modèle d'Objectif élémentaire à traiter.
  LStructCtrl : est une liste des StructCtrl de code à traiter. /* structures d'un composant*/
  StructCtrl : est une structure d'un nœud.
  LAct : Liste d'activité d'un processus métier élémentaire à traiter.
  EV : Ensemble des variables (la liste des données utilisées par une structure de contrôle
       associée à une activité d'un processus métier).        /* arguments*/
  VS : Variable de sortie finale d'un objectif donnée.
  CodeServ : le code service détecté qui influe sur la liste de la variable traitée.

  Sélectionner_une_structure_de_contrôle (StructCtrl) ;

  Sélectionner_une_activité_de_processus_métier (ActPro) ;

  Tantque (LV != vide) faire :
    Instruction= feuille (StrucCtrl) ;          /*commençons par les feuilles de la structure de
                                                  contrôle séléctionneé*/

    Répétez
      Si (EV, SV) sont modifiées par Instruction Alors :
        début
          Insérer (CodeServ, Instruction) ;   /*Insérer l'instruction dans le code de services */
          Insérer (EV, toutes les variables VAR utilisées par cette instruction) ; /* définir tout
                                les arguments et les sorites finales de code de service CodeServ*/
        Finsi
      Instruction = Instruction→ suivant ;
    Jusqu'à la racine de StrucCtrl ;

    Tantque (LAct != Nil) faire :
        S'il existe une activité et un composant associé
        LAct= Lact→suivant ;
    finTantque

    LV= LV→suivant ;
  finTantque
FIN
```

A l'issue de cette phase, nous obtenons des parties de code (instruction ou fonction) qui satisfont les conditions suivantes (voir figure III.5) :

- Chaque bloc de code posséder des entrées sorties claires,
- Chaque bloc réalise une fonction concrète et produit un résultat donné.
- Chaque bloc de code possède un consommateur.
- Chaque bloc de code possède un valeur ajoutée métier et réalise un objectif donné.

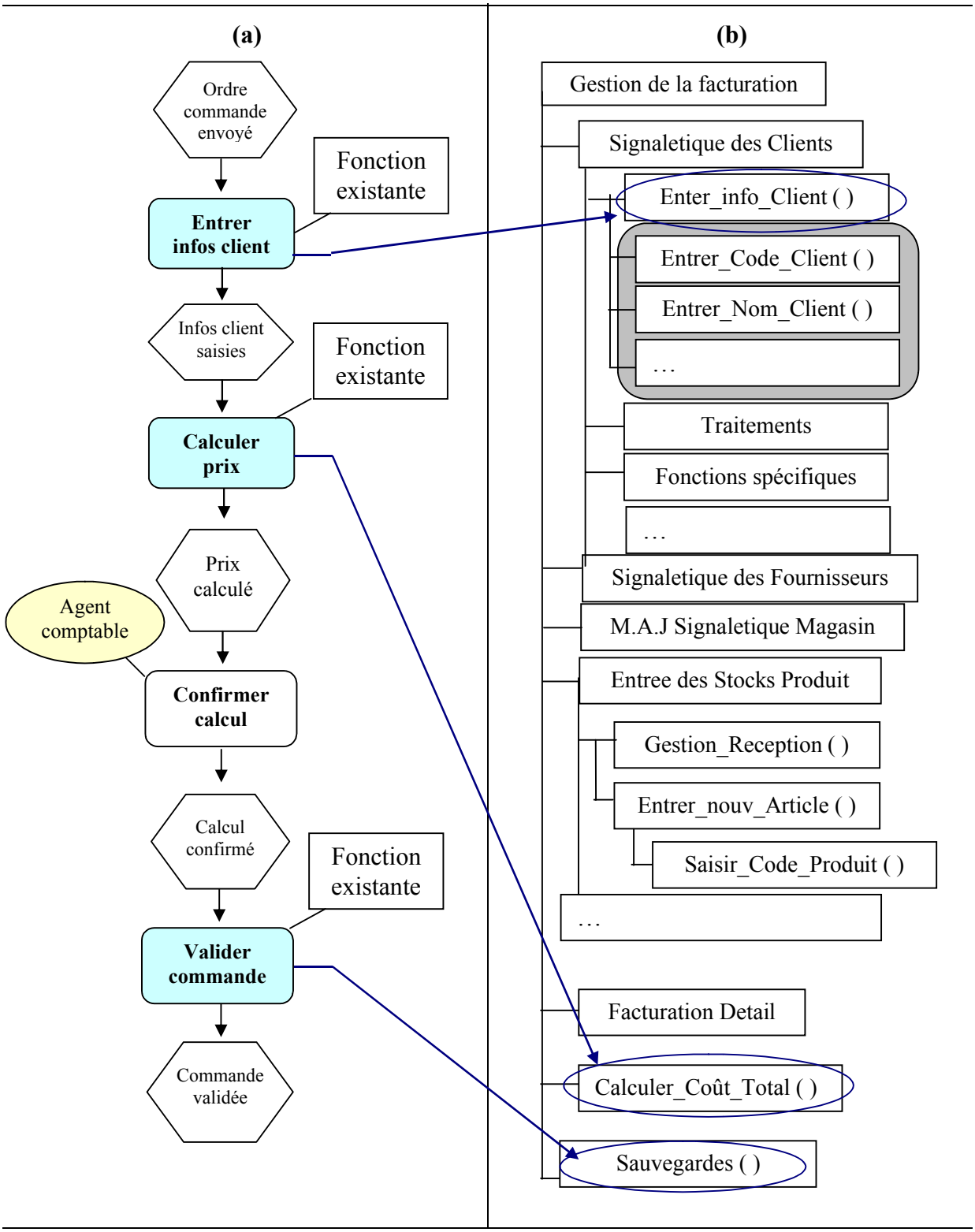


Figure III.5 : Définition de services web.

4.4 Adaptation et publication :

Cette phase est nécessaire pour qualifier les fonctions détectées dans la phase précédente, pour être des services web fonctionnels. Elle consiste à l'adaptation technique (technical

wrapping) des fonctions identifiée par des services web. Elle est composée de plusieurs étapes comme décrit dans **[45]** et **[41]**:

a. Définition de l'interface de description des services "WSDL" :

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

b. Création du mécanisme de transport "SOAP" :

Une fois l'interface du service web est définie, nous passons à la création et l'intégration du mécanisme d'accès et de transport des messages qui sera le processeur SOAP, qui garantit facilement l'échange des messages sur le web à travers le pare-feu.

c. Implémentation du code « Interface_Service » :

« Interface_Service » est une partie du code, ajoutée 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 sur service dans des messages et les envoyer vers les clients.

d. Création de nouveaux services web :

Dans cette étape, le développement des services qui sont apparus dans la phase d'identification des services et ne peuvent pas être extrait du système existant même par la composition des services web définis, est effectué. Ces nouveaux services seront intégrés par la suite dans les futurs systèmes orientés services pour améliorer la logique d'affaire de ces derniers.

e. Enregistrement du service :

Enfin, les services web résultants doivent être enregistrés dans l'UDDI, pour leur utilisation par d'autres clients ou d'autres systèmes orientés services, en les intégrant dans un processus métier.

4.5 Composition des services web :

A l'issue de la quatrième phase, le code extrait de système existant est interfacé et publié comme des services web prêtent à être réutilisé par d'autres services. De nouveaux services web peuvent être aussi apparaître dans cette phase. L'objectif de cette étape est l'intégration de ces services web dans des processus métiers en bon détail d'exécution. Dans notre approche, nous proposons d'utiliser les modèles de processus métiers élémentaires, dont chaque activité correspond à un service web

Nous avons choisi d'utiliser les notation BPEL (Business Process Execution Language) pour la création des fichier décrivant le détail d'exécution des processus métiers définissant les objectifs attendus se système d'information de l'entreprise. Dans notre processus, chaque activité de processus métier élémentaire représente un service web.

5. Discussion

Le but de cette section est de bien motiver les gains qui marquent notre solution, par apport aux autres approches présentées dans le chapitre précédent, en d'autre terme, nous donnons une réponse détaillée à la question : « *Quelles sont les nouveautés de ce processus* ? », en mettant l'accent sur les points communs et les différences de notre processus des deux autres, celle de G. Lewis et al **[48] [53]** et l'autre de Smith et al **[54]**.

Notre approche commence par une analyse des nouveaux besoins et des capacités existantes, qui a été essentiellement inspirée à partir de la phase d'établissement du contexte de migration de G. Lewis et al. Cette dernière est très longue par apport à la phase de notre processus où nous nous somme limités à la partie de compréhension de système existant et à la compréhension des nouveaux besoins de futur système intégré. Nous avons proposé cette phase au démarrage de processus de migration, puisque la décision à ce niveau de processus est très importante pour éviter les problèmes qui peuvent survenir si le système n'est pas vraiment un bon candidat pour la migration vers un environnement SOA ou si le projet de migration ne répond pas aux besoins pour lesquels ce projet est démarré. De ce fait nous gagnons beaucoup en temps, en coût et en efforts en cas défavorables.

A l'inverse de l'approche de Lewis qui réalise cette vérification en utilisant une méthode classique d'analyse et d'application de quelques critères pour décider si ce code mérite d'être un service web ou non. Nous avons proposé dans notre processus d'utiliser une méthode de

réingénierie basée sur les modèles d'entreprise pour la prise de décision et la détection des services web éventuels en cas favorable.

Par opposition au autres approche, la prise de décision dans notre processus est centrée sur les besoins métiers de l'entreprise c'est-à-dire les objectifs ciblés. Elle est aussi basée sur les capacités de système existant.

L'approche **[38]** est intimement liée au langage de programmation du code source, et surtout dans la détection du code candidat à être un service web. Dans notre approche, nous avons réussi à éviter ce problème par l'utilisation des modèles d'entreprise comme un langage intermédiaire, pour unifier les langages de programmation et étendre la gamme des applications à migrer.

En ce qui concerne la détection automatique de code de service web, nous avons proposer une procédure simple qui considère que chaque partie de code associée à une activité de processus métier de l'entreprise représente un service web, autrement dit, chaque service web implémente un objectif donnée. La détection du code est réalisée par la projection des modèles, ce code est ensuite rassemblé et transformer en services web. Ce qui n'est effectivement fait dans les approches discutées dans le chapitre précédent, dans ces dernières la détection des services web nécessite l'intervention des l'expert humain, cela nécessite beaucoup d'efforts et provoque des erreurs qui influent sur le système par la suite.

Pour finir cette discussion, nous présentons quelques avantages de processus proposé dans les points suivants :

- les modèles d'entreprise supportent l'analyse automatique et la vérification de certaines caractéristiques, ce que justifie la facilité de réaliser la phase de l'étude de faisabilité proposée. Cette dernière basée sur l'existant et guidée par les besoins grâce à l'utilisation des méthodes de modélisation en entreprises (Enterprise Modelling).
- Les limites d'intervention des humains dans les phases de processus, plus précisément dans la phase de détection des services web.
- Offre une intégration étroite et flexible dans la mesure où la réconciliation des modèles des besoins aux modèles de l'existant permet à l'organisation de s'adapter facilement aux changements rapides de ses besoins métiers.

- Etendre la gamme d'applications à migrer (progiciel, CRM, ERP, programmes spécifique écrits en langages différents, …etc.), dans la mesure où l'utilisation des modèles est indépendante des caractéristiques techniques (technologie utilisée, langage de programmation, …etc.)
- Facilite les phases de maintenances éventuelles des services web et des processus métiers toujours en terme de coût, de qualité et de délai, grâce à l'utilisation des modèles.

6. Conclusion

Dans ce chapitre, nous avons exposé notre processus pour l'intégration des applications d'entreprises en utilisant la technique de migration des applications vers une architecture orientée services web. Les phases de ce processus sont inspirées essentiellement des travaux de G. Lewis et Smith et al avec des contributions qui touchent principalement les phases d'analyse et de conception.

Nous avons proposé d'utiliser les modèles d'entreprise dans la phase d'analyse et d'étude de faisabilité, comme un langage intermédiaire pour étendre la gamme des applications à migrer. Nous avons aussi proposé d'impliquer l'expérience des utilisateurs et des experts du domaine dans ces phases. Ceci est très utile pour bien comprendre le métier de l'entreprise et pour documenter les besoins de système attendu.

D'autres changements sont suggérés dans la phase d'identification des parties de code patrimonial qui implémentent les services répondant aux besoins de l'entreprise. Nous avons proposé pour cela une procédure permettant de localiser et d'extraire le code qui implémente un objectif donné avec toutes les variables nécessaires. Cette procédure vise à intégrer la spécification des entrées et des sorties des fonctions de système patrimonial dans le modèle fonctionnel. Nous attendons de cette étape à minimiser la complexité de système orientée service, par l'application d'une procédure centrée sur les besoins en définissant des services de bonne granularité et à étendre la gamme d'applications à migrer en unifiant leur présentation par les modèles.

CHAPITRE IV

Etude de cas & quelques aspects d'Implémentation

1. Introduction

Après une représentation théorique des différentes étapes de notre processus basé sur les modèles d'entreprise pour la migration des Systèmes d'Information d'Entreprise vers une Architecture Orientée Services Web. Et afin de constater l'exploitation de ses principes et de ses standards, nous avons choisi de présenter une étude de cas afin de pouvoir constater certains aspects techniques de l'approche. Dans cette étude de cas, les aspects d'hétérogénéité de données et d'applications et aussi l'aspect d'ergonomie ont été abordés.

Pour valider ce processus, nous présentons un scénario d'une étude de cas consiste à faire migrer deux applications vers un environnement distribué de services web. Ces deux applications à savoir : la facturation et la gestion de stocks sont interdépendantes et hétérogènes, elles doivent exposer les fonctionnalités intéressantes comme des services web pour permettre leur interopérabilité en échangeant des données entre elles.

L'étude de cas consiste à un simple processus métier ayant recours à des applications appartiennent à des systèmes d'information hétérogènes.

Ce chapitre décrit l'application de notre processus. Il explique d'abord l'énoncé du scénario de l'étude de cas qui a été proposé. Puis, illustre l'application de l'approche proposée sur l'étude de cas.

Nous exposons dans cette étude de cas trois niveaux d'intégration : le premier niveau d'intégration représente la communication entre les clients et l'entreprise, le niveau d'intégration entre applications qui représente la communication entre les applications de l'entreprise à savoir : l'application de 'Facturation' et l'application de 'Gestion de stocks' et enfin, l'intégration inter-entreprises qui représente la communication entre l'entreprise et ses unités de fabrication.

Ce processus comprendra le modèle d'arbre fonctionnel pour la modélisation des systèmes existants et des besoins métiers attendus. Nous proposons le langage de spécification de processus métiers BPEL[53] **[42]** ainsi que le langage VB.NET[54] **[60]** pour l'implémentation de nouveaux services web et de moteur Workflow.

[53] BPEL4WS: Business Process Execution Language.
[54] VB.Net : implémentation sous la plateforme .Net en utilisant le langage Visuel Basic (VB).

Pour bien expliquer les étapes de notre processus, nous donnerons des modèles et/ou des morceaux de codes qui ont été réalisés dans chaque étape. Des interfaces des différentes applications sont aussi exposées.

2. Présentation de l'étude de cas

L'intégration des applications apparaît comme un réel besoin des entreprises, particulièrement les entreprises industrielles. Nous avons choisi pour illustrer la validité de notre processus, un exemple d'une entreprise commerciale de production de machines outils, appelée «PMO: Production Machines Outils».

PMO est une société dont l'activité principale est la commercialisation des machines outils. Cette activité consiste à la production et la vente en détail ou en gros des machines outils.

PMO contient six services : le service commercial, le services de gestions des factures, le service de gestion de stocks, le services de comptabilité et le service de paie.

PMO souhaite déployer une nouvelle application de gestion des factures, afin d'améliorer la présentation des factures et des bons de livraison de l'ancienne application.

PMO souhaite collaborer avec ses unités de fabrication distribuées géographiquement pour pouvoir régler les commandes clients, en cas de rupture de stocks.

Notre étude de cas porte sur un simple processus métier de l'entreprise, ayant recours aux sous systèmes de gestion de stocks et de gestion des factures. Le schéma suivant permet d'illustrer le processus de l'entreprise (figure IV.1):

> *Un client contacte l'entreprise (PMO) pour faire une commande de produits. La commande est, ensuite, saisie dans l'application de gestion des factures par le facturier à partir des informations fournies par le client. Lors de la prise de la commande le facturier doit disposer le coût et les dates probables de livraison. Ces informations doivent pouvoir envoyées directement par fax ou courrier électronique au client. Une fois confirmée, la commande est mise à disposition immédiate au service de gestion de stocks de l'entreprise (PMO), ces informations portent principalement sur la description des produits, les quantités demandées et les délais d'envoi. A la réception de la commande, l'agent de marketing étudie la quantité demandée de chaque produit, elle est comparée à la quantité disponible dans le*

magasin (quantité en stocks). En cas de rupture de stocks, l'entreprise passe la commande à une autre unité de fabrication. Une fois la commande est validée, le facturier génère les factures et un ordre d'envoi.

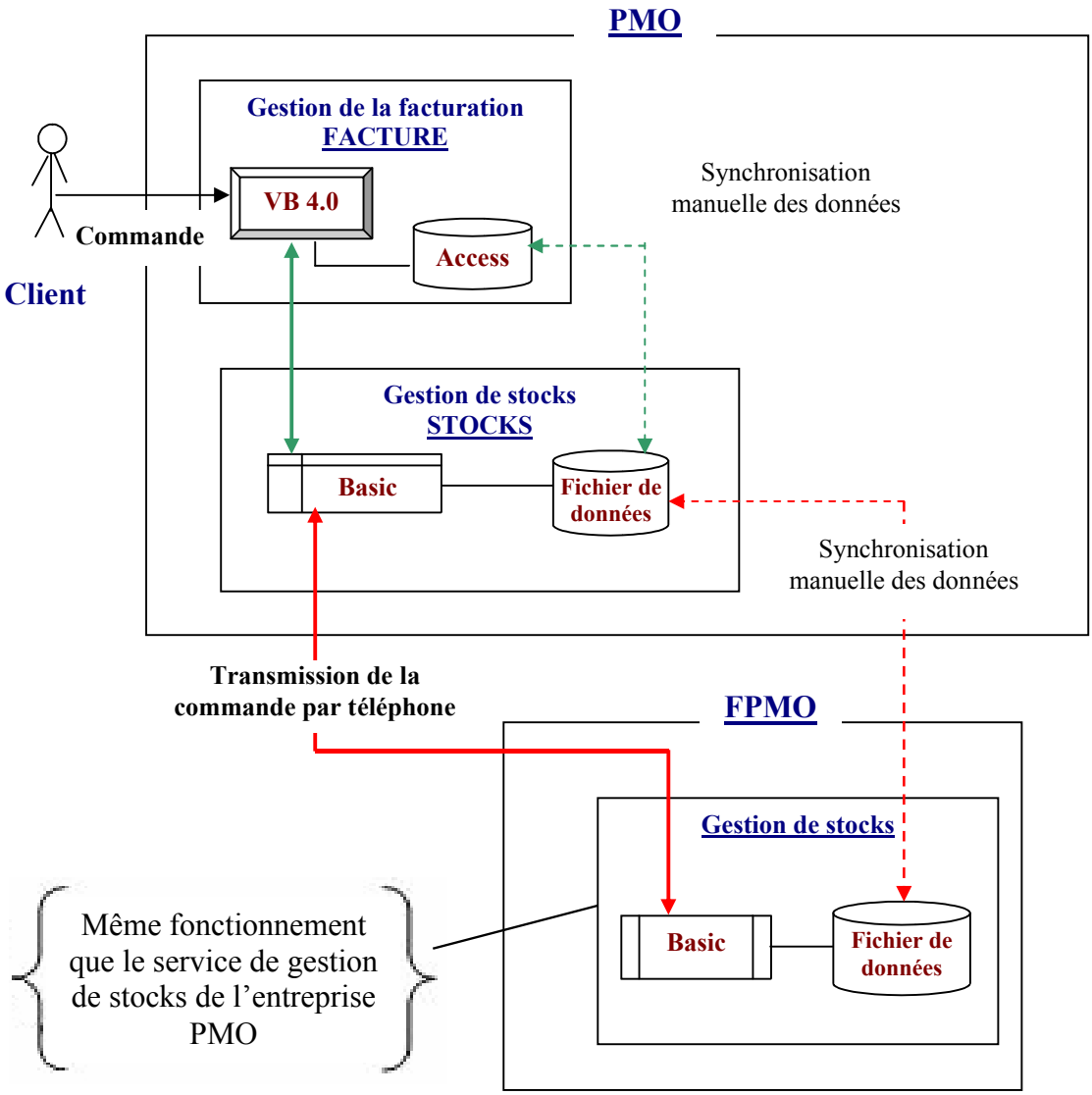


Figure IV.1 : Processus métier de l'entreprise PMO.

A la fin d'une journée, l'administrateur de système met à jour les bases de données de système (synchronisation manuelle des données), pour assurer la consistance des données.

La migration de système d'information de l'entreprise PMO permet l'extraction de ses fonctionnalités intéressantes et les réutilisées par d'autres systèmes. L'intégration des applications se réalise par l'intégration de leurs services web dans des processus métiers répondant aux besoins métiers de l'entreprise. Pour cela, un exemple d'un processus métier simple a été choisi (voir figure IV.4 dans la page 94).

Ce processus métier permet l'échange des différents messages entre les services web extraits des applications de système d'information de l'entreprise PMO.

3. Objectifs de l'étude de cas

L'énoncé de l'étude de cas présentée ci-dessus, permet de valider trois cas d'intégration majeurs. Le premier cas au sein de l'entreprise PMO pour valider la communication et l'interopérabilité entre l'application de gestion des factures et l'application de gestion de stocks en utilisant la technique de migration des fonctionnalités intéressantes communes vers une technologie efficace d'interopérabilité, qui est les services web.

Pour valider l'interopérabilité entre plusieurs entreprises en utilisant la technique de migration, nous avons proposé deux scénarios : celui où le client (Particulier et/ou Entreprise) contacte l'entreprise PMO et ses unités de fabrication pour effectuer une commande, dans le cas où la quantité demandée n'est pas disponible.

4. Application du processus proposé

Après avoir présenté l'énoncé de l'étude de cas et son objectif, nous allons l'appliquer le processus proposé dans le chapitre III.

4.1 Etude de faisabilité :

Cette phase commence par une étude qui a été effectué dans la phase précédente[55] sur nos applications, l'application de facturation et l'application de gestion de stocks, nous avons découvert qu'il y a beaucoup de fonctionnalités qui peuvent être réutilisées comme des services web. Cette étude nous a permet d'accueillir beaucoup d'informations techniques et métiers, que nous résumons dans le tableau suivant :

Application\ Informations recueillies	**FACTURE**	**STOCKS**
Taille de l'application	L'application de facturation contient 06 modules[56] (Reception, Facturation, Edition, Creance, …), cette application possède interface utilisateur pour l'interaction humaine.	Le système de gestion de stocks contient 35 programmes (Signaletique des Fournisseurs, Signaletique des Magasins, Signaletique des Produits,…), il possède interface utilisateur.

[55] La phase de l'étude de contexte est une étape préparatoire à la phase de l'étude de faisabilité.

Age de l'application	Application utilisée pendant 10 ans (depuis 1998)	Application utilisée depuis 1988
Historique (modification, maintenance…etc.)	Quelques modifications effectuer sur les modules de cette application concernent principalement l'utilisation des composants de VB 5.0 et VB 6.0. (migration de l'application vers VB 6.0)	Aucune modification n'est effectuée sur cette application
Utilisateurs	Service commercial (comptables, facturiers, …)	Service commercial. Service de stocks (agents de marketing, …etc.).
Langage de programmation	Visuel Basic (VB 6.0)	Basic
Technologies utilisées	/	Générateur d'application (langage Basic) TOPKEY. Compilateur Quick Basic.
Principales fonctionnalités	Il existe des services simples d'accès aux sources de données (access) : pour lire et écrire des données. Et des services qui ont des tâches bien définies comme : - Traitement : qui vérifie la quantité de produits en stocks par apport aux quantités demandées. - CalculPrix : qui permet de calculer le montant total d'une facture.	Il existe des services qui implémentent les règles de gestion de l'entreprise, comme : - Traitements : mettre à jour les quantités de produits en stocks après la validation d'une commande ou la réception d'un produit, …etc. Et d'autres services simples : - les services d'accès aux sources de données (lire des donnée d'un fichier de données et les affectées aux variables et écrire le contenu d'une variable dans un fichier).

Tableau IV.1 : Analyse des applications à migrer.

[56] Un module : contient des fonctions et des sous programmes utilisables selon les besoins d'autres modules dans un programme.

Une étude est effectuée aussi sur les besoins de futur système. Pour cette étude de cas seule la problématique de gestion des commandes client est traitée. Cette dernière représente l'objectif principal de notre projet, il consiste à la réalisation d'un simple processus métier de traitement d'une commande d'un client (voir la figure IV.4).

4.1.1 Construction des modèles :

L'application de notre étude de cas, contient des primitives de branchement permettant de faciliter l'importation de textes source existants, comme : GOTO, GOSUB et RETURN. Cependant, l'utilisation des ces primitives rend le code source difficile à lire, ce qui complique sa compréhension et son traitement dans les futurs phases. Pour cela, nous avons choisi de faire une phase de prétraitement permettant de rendre le code à traiter clair et déterministe.

a. Prétraitement :

Avant de commencer l'étape de modélisation de système, nous avons choisi d'amblé de faire des modifications permettant de supprimer les branchements existants dans le code de l'application, notamment l'instruction « GOTO[57] » et les deux autres « GOSUB et RETURN[58] ». En général, toute instruction "Goto" peut être remplacée par une boucle d'un bloc "IF" **[59]**. Un processus d'élimination du « Goto » est proposé dans **[58],** comme suit :

(a) : programme contient GOTO	**(b)** : après la suppression du GOTO
…. **A1 :** Bloc d'instructions1 Si (Condition 1) alors Bloc d'instructions2 GOTO **A2** finSi Bloc d'instructions3 Si (Condition2) alors Bloc d'instruction4 GOTO **A1** finSi **A2 :** Bloc d'instructions5 ….	….. **Répétez** Bloc d'instruction1 Si (Condition1) alors Bloc d'instuctions2 Sinon Bloc d'instructions3 Si (Condition2) alors Bloc d'instructions4 finSi finSi **Jusqu'à** (Condition1) Bloc d'instructions5 ….

[57] GOTO : instruction d'héritage du passé, elle permet de créer un branchement quelconque à l'intérieure d'un programme ou un sous programme.
[58] GOSUB : est une instruction de branchement dans une procédure vers une étiquette dont les instructions du code exécutées après le branchement, sont terminées par le mot clé RETURN.

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)	**(b)**

.....
12200 IF CXFAF<>"ZGC" THEN
GOTO 12202
12201 IF (IZ2=0) AND (INSTR("C ",CX8))>0
THEN
PXFAS=0
RETURN
ELSE PXFAS=IXFAZ
RETURN
12202 IV3=0:RETURN
.....
	…….
	Repeate
	IF (IZ2=0) AND (INSTR("C ",CX8))>0
	THEN
	PXFAS=0
	Else
	PXFAS=IXFAZ
	While (CXFAF<>"ZGC")
	IV3=0
	…….

Après la phase de prétraitement, nous passons ensuite à la phase de transformation de code en des modèles unifiés à un niveau d'abstraction plus élevé.

b. Modèle d'arbre de fonction de système existant :

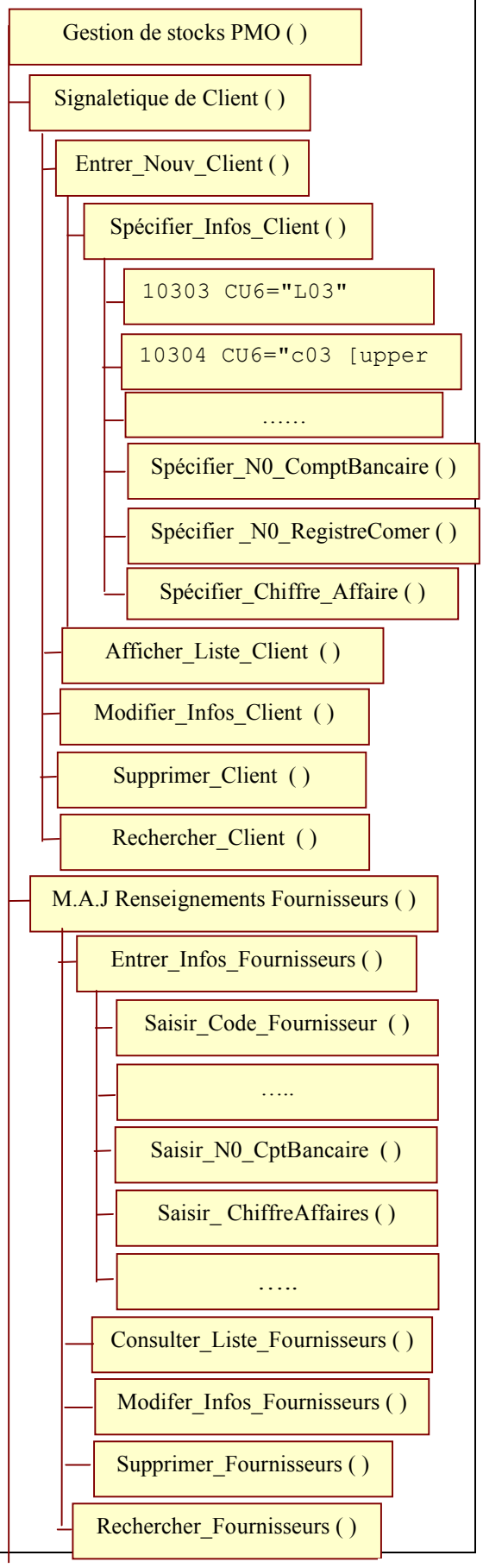
Comme nous l'avons présenté dans le chapitre 3, notre approche basée sur les modèles d'entreprise commence par une représentation haut niveau de l'existant. Le but de cette phase est de modéliser le code source de l'application en modèle hiérarchisé de fonctions, ce modèle, dit le modèle d'arbre de fonction de système existant (ou **F**unction **T**ree **M**odel of legacy systems), est la description abstraite et structurée de toutes les fonctions de l'application en un modèle fonctionnel. Ce dernier est indépendant de tout langage de programmation. Tel que nous l'avons expliqué dans le chapitre précédent, la transformation de code source en modèle fonctionnel se fait de la manière suivante **:**

- La racine de ce modèle représente la fonction main ( ) de l'application,
- Les nœuds représentent les fonctions principales de l'application,
- Les sous nœuds représentent les sous fonctions interviennent à la réalisation des fonctions principales, qui sont considérées à un niveau hiérarchique supérieur dans l'application,
- Les feuilles de modèle représentent les instructions participant à la réalisation des fonctions qui se trouvent à un niveau hiérarchique plus élevé dans le modèle.

Puisque les applications sont assez longues, nous allons exposer seulement la partie du code qui nous intéresse dans cette étape et même dans les phases restantes.

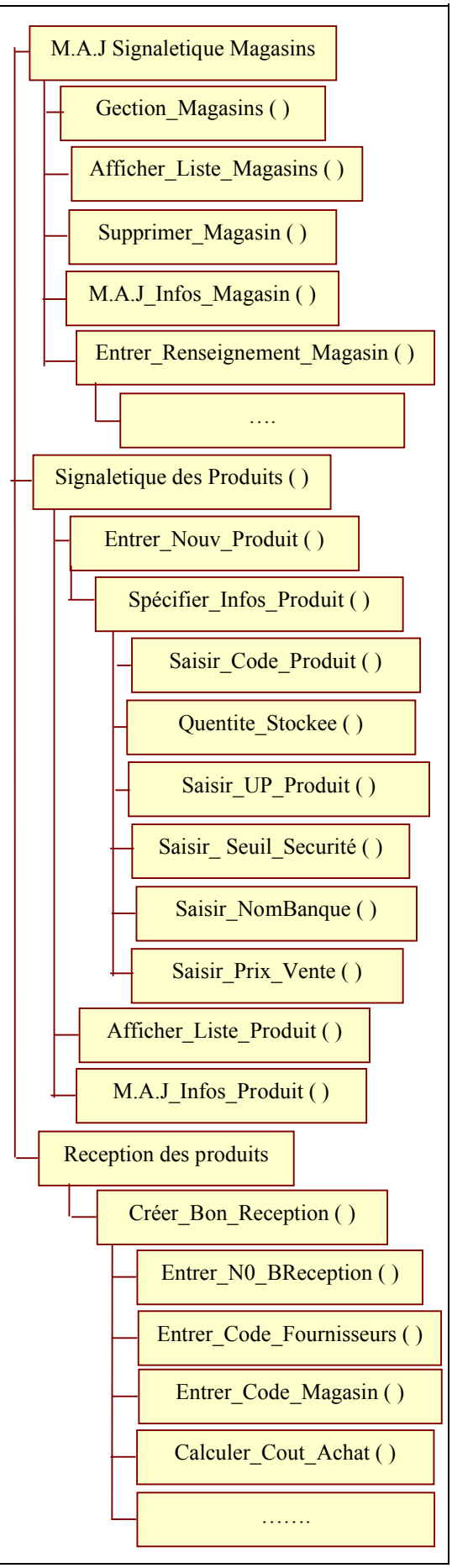
(a)

```
s1 ' --- TOP KEY ---      SP00      #THR
2 CALL FBR3
99 REM $INCLUDE: 'MLSIV'
900  ' ----- nom du programme
905 CVNPROG="SP00"
999 REM $INCLUDE: 'THW02'
1500  ' ----- debut
......
10000  ' ----- titre du menu
10005 CV9(1)="GESTION DE STOCKS"
10050 RETURN
10500  ' ----- fin du menu
.....
10510 CLOSE
10515 IF CRET="" THEN CALL FBR4:END
10520 CU1=CRET:
11000  ' ----- retour vers menu principal
11050 RETURN
11500  ' ----- mot de passe
11550 RETURN
12000  ' ----- code texte a inserer
12005 ITXT=5000
12050 RETURN
12500  ' ----- programmes
12505 CV6(1)="1,SP01,Signaletique des
Clients"
/*****************************************/
.....
/*****************************************/
10300 ' ----- traitements
10301 CU6="P ecp3701,X2701,DNP"
10302 ' CU6="c02 val(&02)"
10303 CU6="L03"
10304 CU6="c03 [upper]"
10305 CU6="l04"
10306 CU6="l05"
10307 CU6="l07"
10308 CU6="V14 > 0 , < 6"
10309 CU6="v15 O,N"
10497 ' ----- references externes
10498 CVXTRF=""
10499 RETURN
10500 ' ----- fin du programme
......
10511 IF IVXTRF=1 THEN END
10515 CU1="SP00"          ' retour vers SP00
10800 ' ----- ouverture du fichier
principal
10805 CU1="SF01":CX3=CU1
SF01  Structure / Client
10806 IX7=8:IX4=170
10840 ISWN=1:CSWF=CX3:GOSUB 60001
10842 IF ISWE>0 THEN 50140
10844 IF ISWE=0 THEN 10849
10846 ISWD=IX4:ISWK=IX7
10847 IF ISWE>0 THEN 50140
10849 IWFNM(ISWN)=ISWH:CVCLE=SPACE$(IX7)
/*****************************************/
```



```
.....
/*****************************************/
12000 ' ----- fonctions specifiques
12099 REM $INCLUDE: 'THW59'
12200 IF CXFAF<>"UPPER" THEN 12203
12201 IF IXFA=0 THEN CVY=CV4(IJ5) ELSE
CVY=CXFA(1)
12202 CXFAS=CVY
12203 IV3=0
/*****************************************/
.....
/*****************************************/
12506 CV6(2)="2,SP02,Signaletique des
Fournisseurs"
.......
12507 CV6(3)="3,SP04,M.A.J Signaletique
Magasin"
....
12508 CV6(4)="4,SP05,Signaletique Produits"
.....
12509 CV6(5)="5,SP11,Reception"
/*****************************************/
.....
/*****************************************/
10300 ' ----- traitements
10301 CU6="R05 2,8,0,15"
10302 CU6="R07 3,10,0,25"
10303 CU6="A08 3,10,0,25"
10304 CU6="A10 3,10,84,8"
10305 CU6="A14 3,10,84,8"
10306 CU6="A16 3,10,76,8"
10307 CU6="A17 3,10,92,8"
10308 CU6="A18 2,8,111,8"
10309 CU6="M14 3,10,84,8,200,14 ; [zgc]"
10310 CU6="M16 3,10,76,8,200,16"
10311 CU6="M17 3,10,92,8,200,17"
10312 CU6="M18 2,8,111,8,120,18"
10313 CU6="P ecp3711,fnv07/18,ngm,te"
10314 CU6="c03 [INC] ; [zgc]"
10315 CU6="l04"
10316 CU6="B07"
10317 CU6="l07"
10318 CU6="c13 &09*&10 ; [zgc]"
10319 CU6="c14 &10"
10320 CU6="m14 ; [zgc]"
10321 CU6="c15 &12/&09 ; [zgc]"
10322 CU6="c16 &16+&09 ; [zgc]"
10323 CU6="c16 &16-&09 ; [zgs]"
10324 CU6="m16"
10325 CU6="c17 &17+&13 ; [zgc]"
10326 CU6="c17 &17-&13 ; [zgs]"
10327 CU6="m17"
10328 CU6="c18 &18+&13 ; [zgc]"
10329 CU6="c18 &18-&13 ; [zgs]"
10330 CU6="m18"
10497 ' ----- references externes
10498 CVXTRF="&RX110.05=(EFFO)
&RX110.07=(ARTI)"

10500 ' ----- fin du programme
```



```
.......
10511 IF IVXTRF=1 THEN END
10515 CU1="SP00"'  retour vers SP00
/*************************************/
......
/*************************************/
12000 ' ----- fonctions specifiques
12099 REM $INCLUDE: 'THW59'
12200 IF CXFAF<>"INC" THEN 12203
12201 IF IXFA=0 THEN PXFA(1)=PV5(IJ5)
:CXFA(1)=CV4(IJ5)
12202 PXFAS=VAL(CXFA(1))+1
:CXFAS=MID$(STR$(PXFAS),2)
:II0=IWLG(IJ5):CXFAS=RIGHT$
(STRING$(II0,"0")+CXFAS,II0)
:PXFAS=PXFA(1)+1:RETURN     '$13
12203 IF CXFAF<>"ZGC" THEN 12205
12204 IF (IZ2=0) AND (INSTR("C ",CX8))>0
THEN RETURN ELSE PXFAS=IXFAZ:RETURN
12205 IF CXFAF<>"ZGS" THEN 12207
12206 IF (IZ2=0) AND (CX8="S") THEN RETURN
ELSE PXFAS=IXFAZ:RETURN          '
12207 IV3=0:RETURN
/*************************************/
......
/*************************************/
.....
IF ISWE>0 THEN 50140
60002:IWFNM(ISWN)=ISWH:IF ISWE>0 THEN 50140
10999 RETURN
12517 CV6(12)="12,SP07,Suivi des Creances"
.....
12518 CV6(13)="13,SP40,Sauvegarde"
.....
12900 RETURN
24099 REM $INCLUDE: 'THW62'
39999 REM $INCLUDE: 'THW57'
40199 REM $INCLUDE: 'THW57S'
46999 REM $INCLUDE: 'THW61'
49999 REM $INCLUDE: 'THW22'
......
```

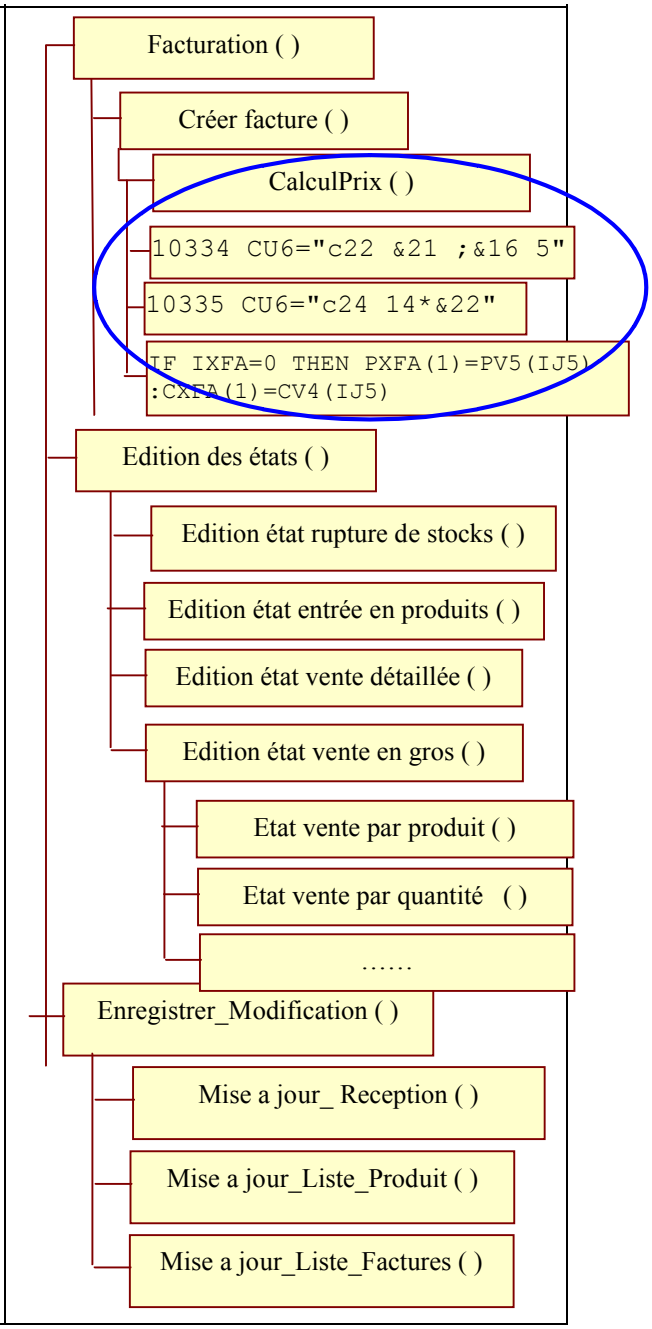


Figure IV.2 : Modélisation du code Basic en un arbre fonctionnel.

A l'issue de la phase de prétraitement, la plupart des instructions du code source de l'application sont incluses dans des structures de contrôle, c'est-à-dire des structures conditionnelles simples « if », des alternatives « if…else » et des boucles « for…next ». La modélisation de ce code, donne un modèle dont les feuilles sont des instructions simples d'affectation, seules ou avec les structures de contrôles qui les contiennent. Comme le montre l'exemple précédent, la partie encerclée (figure IV.2) représente les feuilles de modèle, contiennent une structure de contrôle "IF".

c. Modèle d'objectifs métiers de l'entreprise :

Dans notre approche basée sur les modèles d'entreprises, la phase de modélisation implique aussi la description et la formalisation des processus métiers de l'entreprise, tel que nous l'avons expliqué dans le chapitre 3. Ceci est utile pour une décision centrée sur les besoins (les contraintes et les exigences métiers) et non seulement sur l'existant. Parmi les différents modèles d'entreprise que nous avons choisi est un modèle hiérarchique similaire à celui de modèle fonctionnel de système existant.

Nous commencerons par présenter les principales étapes du processus métier de l'entreprise ainsi que leurs agencements. Le processus métier de l'entreprise est présenté dans le modèle suivant (figure IV.3):

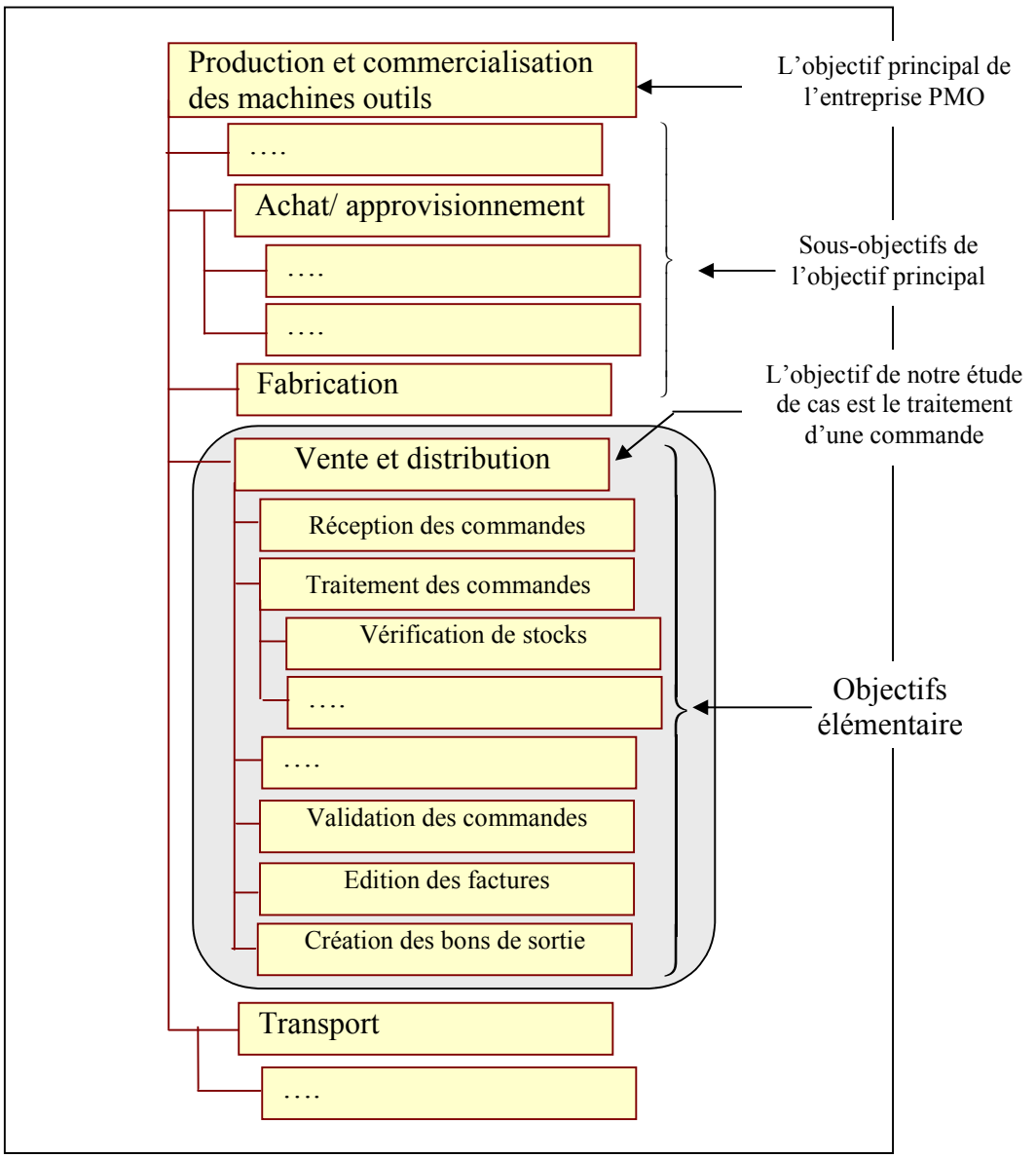


Figure IV.3 : Modèle d'objectifs métiers de l'entreprise PMO.

Dans ce modèle, l'objectif principal de l'entreprise est «la production et le commercialisation des machines outils». Ensuite, les sous objectifs permettant la réalisation de l'objectif principal à savoir : l'achat et l'approvisionnement de la matière nécessaire à la fabrication des produits, la vente et la gestion des commandes. Le processus de l'entreprise se termine par la livraison et le transport des commandes aux clients.

4.1.2 Définition des services éventuels :

Après la modélisation de l'existant et des objectifs de migration, nous avons trouvé que le système contient des fonctions qui méritent d'être publiés comme des services autonomes et réutilisables, nous citons par exemple :

- le composant *Signaletique Des Clients* qui contient des fonctions permettant la gestion des informations des clients lors de la réception d'une commande et le suivi de l'état détaillé des clients.
- le composant *Signaletique produits* qui permet la gestion des produits et le suivi de l'état de stocks dans les magasins.
- les composants *M.A.J Signalétique des Magasins* et *Réception des Produits* qui contient des fonctions permettant la gestion des activités d'achat et d'approvisionnement.

En effet dans cette définition d'une première liste de services web éventuels, seules les composants permettant d'offrir ces services sont détectés. On ne précise pas quelles fonctions ou tâches (actions) réalise effectivement une activité d'un objectif donné – comment un processus par exemple de traitement d'une commande est réalisé – quelle application est utilisée pour vérifier la quantité en stocks d'un produit demandé.

4.2 Identification des services :

A partir des modèles établis dans la phase précédente à savoir : le modèle d'arbre de fonctions de système existant et le modèle d'objectifs métiers, nous pouvons localiser et extraire les parties de code intéressantes ayant un niveau de détail adéquat, qui seront ensuite qualifier (adapter) en tant que services web.

Cette phase commence par une étape de raffinement et de spécification des modèles. Nous commençons par présenter les principales étapes du processus métier, ainsi que leur

agencement permettant la réalisation de l'objectif de notre étude de cas à savoir le traitement d'une commande d'un client. Notre processus comprend les étapes suivantes (figure IV.4):

1. *Réception d'une commande de la part de client : Spécification des informations de client (référence, raison sociale, adresse, numéro compte bancaire, chiffre d'affaire, …etc.). Spécification des informations de la commande.*
2. *Vérification de la quantité des produits de la commande par apport aux quantités en stocks.*
3. *Si la quantité demandée est disponible en stocks, le coût total de la commande est calculé.*
4. *Dans le cas contraire, c'est-à-dire, la quantité demandée n'est pas disponible en stocks, l'entreprise transmit la requête à une de ses unités de fabrication.*
5. *Une fois la commande est validée, une information notifiant l'évaluation de coût et des dates prévisionnelles d'envoi est envoyée au client.*

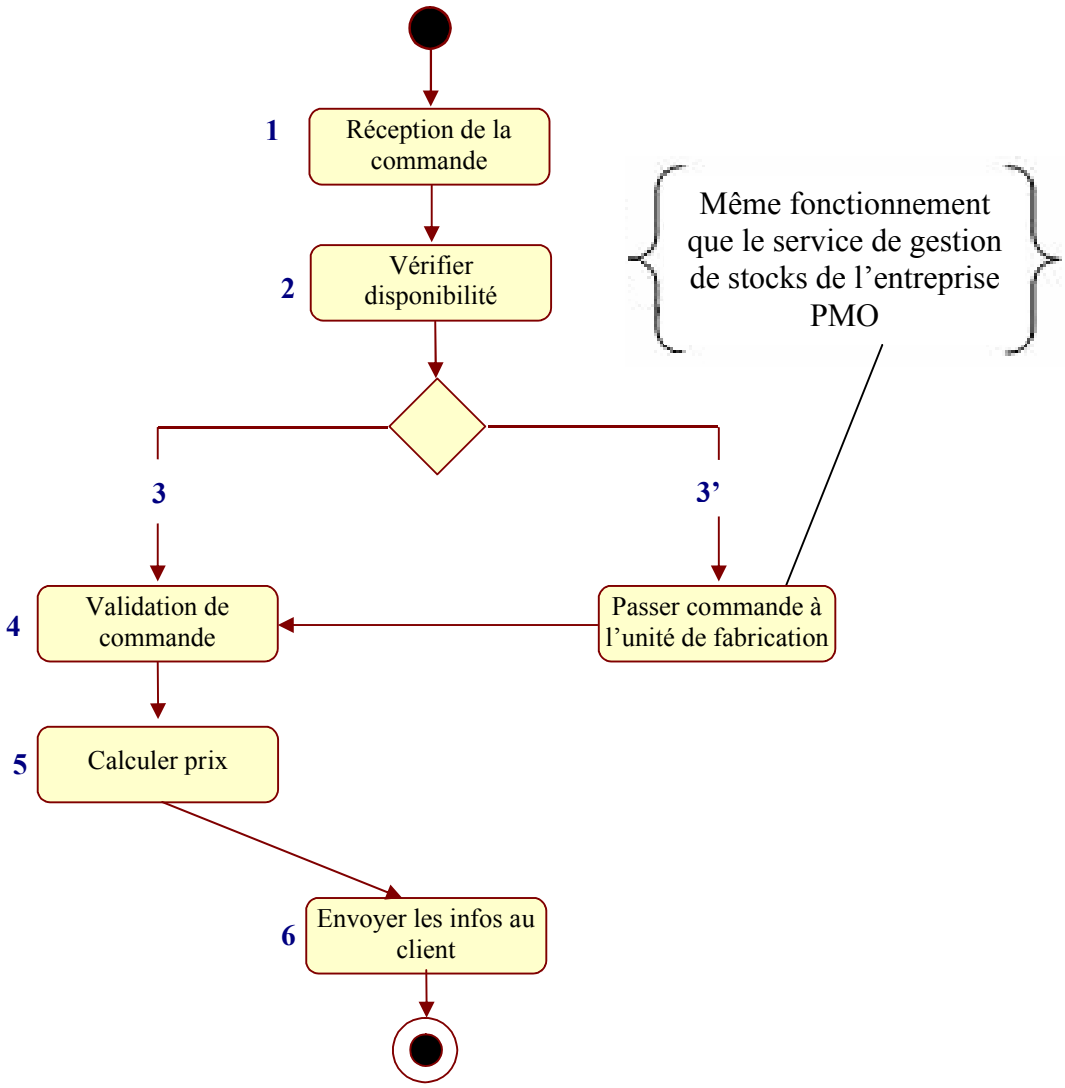
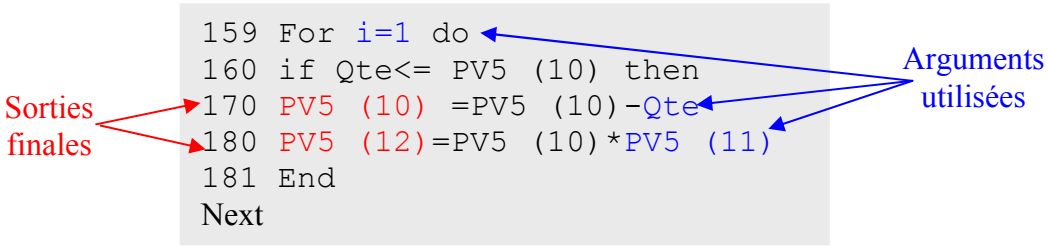


Figure IV.4 : Modèle d'un processus métier élémentaire.

Avant de démarrer le processus de projection des modèles, il faut d'abord compléter la spécification des composants logiciels associés aux activités de processus métiers (Figure IV.2). Nous spécifions les sorties finales de chaque nœud du modèle fonctionnel de composant considéré avec les arguments utilisés, comme il est illustré dans l'exemple suivant :



La projection des modèles consiste à déterminer la partie du code qui est effectivement responsable de réaliser les sorties d'une activité donnée, en suivrant les traces des résultats générés par chaque nœud du modèle comme nous l'avons expliqué dans le chapitre précédent. Le résultat de cette phase sera comme suit :

```
1 ' --- TOP KEY --- SP05  #THG  &THQM3
2 ' --- procedure de lecture à partir de fichier des articles
3 ' --- utilse des argument (CV6 (50), PV5(50), cart$, Qte
4 ' --- aig, message
10 DIM CV6 (50),PV5(50),cart$, Qte
25 LINE INPUT C1$: LINE INPUT C2$
30 aig=0
31 message
40 ' ----- enregistrements
46 FIELD 1,10 AS CV6(2),25 AS CV6(3),25 AS CV6(4),3 AS CV6(5)
47 FILED 1,13 AS PV5(6),3 AS CV6(7),2 AS cv6(8),13 AS PV5(9)
48 FILED 1,13 AS PV5(10),13 AS PV5(11),13 AS PV5(12),13 AS PV5(13)
49 FILED 1,5 AS PV5(14),13 AS PV5(15),5 AS PV5(16),13 AS PV5(17)
50 FILED 1,13 AS PV5(18),AS PV5(19),13 AS PV5(20),13 AS PV5(21)
51 '------ouverture de fichier
60 OPEN "I",1,"ARTICLE"  : I=0  :J=0
61 nbenrg=taillefich/250
70 for i=1 to nbenr
80 LINE INPUT #1,C1$ : PRINT C$(I)
90 cv6(2)=mid$(c1$,1,10)
99 if cv6(2)=cart$ then
100 J=I
101 '------ chargement des enregistrements
110 PV6(3)=mid$(c1$,10,25):PV6(4)=mid$(c1$,35,25):
111 pv5(6)=val(mid$(c1$,63,13)):CV6(7)=mid$(c1$,76,3)
112 PV5(8)=val(mid$(c1$,79,2)):PV5(9)=val(mid$(c1$,81,13))
113 PV5(10)=val(mid$(c1$,94,13)):PV5(11)=val(mid$(c1$,107,13))
114 PV5(12)=val(mid$,120,12)):PV5(13)=val(mid$(c1$,133,13))
115 PV5(14)=val(mid$(c1$,146,5)):PV5(15)=val(mid$(c1$,149,13))
116 PV5(15)=val(mid$(c1$,152,13)):PV5(16)=val(mid$(c1$,165,5))
117 PV5(17)=val(mid$(c1$,170,13)):PV5(18)=val(mid$(c1$,183,13))
118 PV5(19)=val(mid$(c1$,196,13)):PV5(20)=val(mid$(c1$,209,13))
119 PV5(21)=val(mid$(c1$,222,13)):PV6(5)=mid$(c1$,60,3)
```

```
120 I=nbenrg
121 aig=1
125 END       ' IF
130 I+I+1
140 next      ' for
141 close
142 ' --- procédure de vérification de quantité en stocks de
143 ' --- chaque article dans une commande
144 '------ traitements
150 if aig=1 then
160 if Qte<= PV5(10) then
170 PV5(10)=PV5(10)-Qte
180 PV5(12)=PV5(10)*PV5(11)
185 else
186 message=" LA QUANTITE DEMANDEE EST INSUFFISANTE"
187 end    'if Qte
190 else
195 message= "L'ARTICLE N'EXISTE PAS"
200 end      ' if aig
201 '------ ouverture de fichier
210 OPEN "O",1,"ARTICLE"
220 LINE INPUT #1, C1$ : PRINT C1$(J)
221 mid$(str(c1$,94,13))=PV5(10)
222 mid$(str(c1$,146,5))=PV5(14)
223 Close
224 ' --- procédure de synchronisation des fichiers de données
225 ' --- des articles, client et facture après la validation
226 ' ---  d'une commande client
227 '------ enregistrements
228 FILED 1,4 AS CV6(2),8 AS CV6(3),10 AS CV6(4)
229 FILED 1,25 AS CV6(5),25 AS CV6(6)
230 FILED 1,20 AS CV6(7),5 AS PV5(8),10 A CV6(9),25 AS CV6(10)
231 FILED 1,13 AS PV5(11)
232 FILED 1,13 AS PV6(12),2 AS PV6(13),1 AS PV5(14),8 AS PV5(15)
233 FILED 1,8 AS PV5(16)
234 FILED 1,8 AS PV5(17),8 AS PV5(18),8 AS PV5(19),8 AS PV5(20)
235 FILED 1,8 AS PV5(21)
236 FILED 1,1 AS PV5(22),13 AS PV5(23), 13 AS PV5(24)
300 '-------enregistrer
320 OPEN "O",2, "FACTURE" : I=0
322 for I=1 to Nombrerg
333 LINE INPUT #2,C1$ : PRINT C$(I))
334 if mid$(c2$,1,10)="" then
340 Close
```

Listing IV.1 : Code des services web extraits du code de l'application.

Après la détection des parties de code patrimoniales qui méritent d'être publiées comme des services web, nous pouvons dire qu'un grand pas dans le processus de migration vers les services web est effectué, et il ne nous plus reste que l'adaptation et l'intégration de code dans des processus métiers.

4.3 Adaptation et publication :

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

a. Développement de l'interface et la description des services :

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

L'exemple suivant est une description complète de service web verificationStocks, ce service contient deux fonctions. La première fonction reçoit un tableau contenant le code des articles d'une commande comme entrée, et retourne la sortie, qui est un tableau de valeurs décimales qui représentent les quantités en stocks de chaque article. La deuxième fonction reçoit comme entrées un tableau contenant la liste des articles, un tableau contenant les quantités en stocks et retourne la sortie, qui est un booléen qui décide si la quantité demandée est disponible ou non pour chaque article.

```
<?xml version="1.0" encoding="utf-8"?>
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://projetPMO.com/verifivatioStocks.wsdl"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
 targetNamespace="http://projetPMO/verificationStocks.xsd">
  <s:element name="lectureDonnees">
  <s:complexType />
   <s:element name="LectureDonneesRequest">
    <s:complexType>
     <s:sequence>
      <s:element minOccurs="CArticle" maxOccurs="1unbounded"
      </s:element>name="LectureDonneeResult" type="d:Decimal" />
<s:sequence>
      <s:element minOccurs="Quantite" maxOccurs="1unbounded"
</s:sequence>
       </s:complexType>
      </s:element> <s:element name="CompareQte">
        <s:complexType />
      </s:element> s:element name="CompareQteResponse">
        <s:complexType>
 <s:sequence>  <s:element minOccurs="0" maxOccurs="1"
name="CompareQteResult"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
  </wsdl:types>
```

```
<wsdl:message name="lectureDonneesSoapIn">
    <wsdl:part name="parameters" element="tns:lectureDonnees" />
  </wsdl:message>
  <wsdl:message name=" lectureDonnees SoapOut">
 <wsdl:part name="parameters" element="tns: lectureDonneesResponse" />
  </wsdl:message> <wsdl:message name="compareQteSoapIn">
    <wsdl:part name="parameters" element="tns: compareQte" />
  </wsdl:message>
  <wsdl:message name="compareQteSoapOut">
    <wsdl:part name="parameters" element="tns: compareQteResponse" />
  </wsdl:message>
  <wsdl:portType name="Service1Soap">
    <wsdl:operation name="lectureDonnees">
      <wsdl:input message="tns: lectureDonneesSoapIn" />
      <wsdl:output message="tns: lectureDonneesSoapOut" />
    </wsdl:operation>
    <wsdl:operation name=" compareQte ">
      <wsdl:input message="tns: compareQteSoapIn" />
      <wsdl:output message="tns: compareQteSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="Service1Soap" type="tns:Service1Soap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  wsdl:operation name="lectureDonnees"> <soap:operation
soapAction="http://localhost/lectureDonnees" style="document" />
      <wsdl:input> <soap:body use="literal" /> </wsdl:input>
      <wsdl:output> <soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="compareQte"
<soap:operation soapAction="http://localhost.com/ compareQte "
style="document" />
      <wsdl:input> <soap:body use="literal" /> </wsdl:input>
      <wsdl:output> <soap:body use="literal" /> </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="Service1Soap12" type="tns:Service1Soap">
   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="lectureDonnees">
<soap12:operation soapAction="http://localhost.com/lectureDonnees"
style="document" />
 <wsdl:input> <soap12:body use="literal" /> </wsdl:input>
<wsdl:output> <soap12:body use="literal" /> </wsdl:output>
<wsdl:operation name=" compareQte"> </wsdl:operation>
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>style="document" />
      <wsdl:input> <soap12:body use="literal" /> </wsdl:input>
      <wsdl:output> <soap12:body use="literal" /> </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:types>style="document" />
      <wsdl:input> <soap12:body use="literal" /> </wsdl:input>
      <wsdl:output> <soap12:body use="literal" /> </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
<wsdl:service name="Service2">
    <wsdl:port name=" compareQte Soap" binding="tns: compareQteSoap">
<soap:address location="http://localhost:1052/Service1.asmx" />
```

```
   </wsdl:port>
<wsdl:port name=" compareQte Soap12" binding="tns:compareQte Soap12">
<soap12:address location="http://localhost:1052/ compareQte.asmx"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listing IV.2: Document XML décrivant un service web détecté.

b. Implémentation du code de l'interface

Comme nous avons indiqué dans le chapitre précédent, le code « Interface_Service » joue le rôle d'une interface qui reçoit les arguments d'un service web et renvoie les résultats de l'exécution à l'application client.

4.4 Composition des services web :

La composition et l'intégration des services web résultants réalisant l'objectif de notre étude de cas à savoir le traitement d'une commande d'un client, dans un processus métier et leur contrôle selon le modèle présenté dans la phase précédente, implique la création d'un nouveau projet. Ce projet inclura toute les classes nécessaires pour l'exécution et le contrôle du processus métier.

5. Réalisation

La réalisation de notre application sur notre étude de cas concernant le traitement d'une commande d'un client, doit commencer par la création des interfaces permettant la communication du code extrait de l'application patrimoniale avec le service web.

5.1 Création des interfaces du code patrimoniale

Selon le type de l'application patrimoniale, Il existe trois méthodes principales permettant son interfaçage, sont **[61]** :

- *L'application patrimoniale expose une interface "COM*[59]*"* : Dans ce cas, l'application client fait appel à l'interface COM, pour interagir avec l'application patrimoniale. Cette méthode est généralement utilisée si l'application patrimoniale n'a pas d'interfaces utilisateur ou d'API.

[59] COM : Component Object Model

- *L'application patrimoniale est définie dans un "DLL[60]"* : Si toute la logique d'affaires de l'application patrimoniale existe dans les DLLs de l'application client, alors il sera possible de l'importer et d'interagir avec toutes ses fonctionnalités via leurs interfaces.

- *L'application patrimoniale est une application "exe" autonome*: Dans ce cas, il est très facile de transformer une application EXE autonome en une DLL, qui peut être publiée, ensuite aux autres applications. Cette méthode est généralement utilisée si l'application patrimoniale possède une interface API ou interface utilisateur.

En ce qui concerne notre étude de cas, le code (c'est-à-dire les services) extrait de l'application patrimoniale n'a pas d'API et d'interface utilisateur. Pour cela, nous avons choisi la méthode des interfaces COM pour exposer ce code aux autres applications de différentes plateformes. Pour ce faire nous utiliserons le système de programmation Visual Basic 6.0 **[59]**, qui permet de générer des programmes autonomes et aussi des composants ActiveX pour des programmes.

Pour créer notre interface COM, nous avons créé un nouveau projet de type DLL ActiveX, tel que présenté dans la figure IV.5.

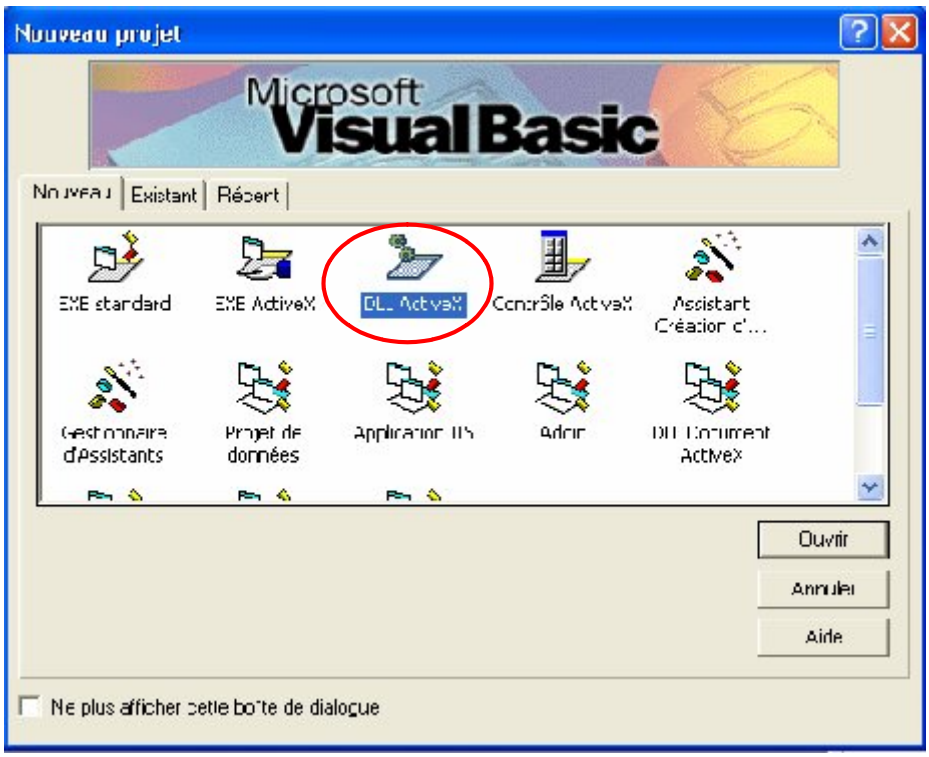


Figure IV.5 : Création des interfaces COM en VB 6.0.

[60] DLL : Dynamically Linked Libraries

L'application ActiveX "*Traitement Commande*" rend le code de services extraits de l'application patrimoniale sous forme de module de classes. Chaque classe implémente et expose les méthodes d'un service par VB 6.0 sous l'extension « .cls », tel que le présente la figure IV.6.

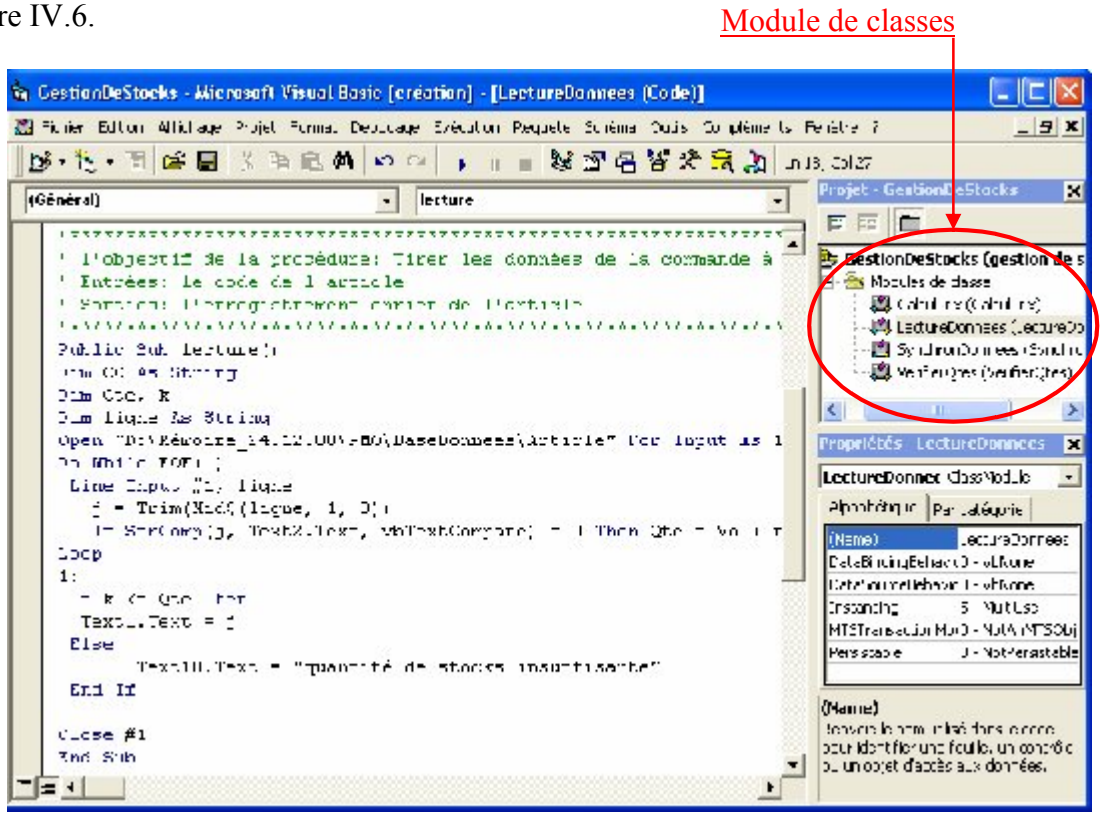


Figure IV.6 : Implémentation des modules de classes qui correspondent au code patrimonial (services).

Après la création des interfaces permettant la communication avec le code patrimonial, vient l'étape de publication et d'adaptation du code via les interfaces de services web.

5.2 Création du code « Interface_Service »

Nous avons créé un nouveau projet, qui contient les interfaces "Interface_Services" des services web (figure IV.7)

La figure IV.7 présente une de ces interfaces « Interface_Code » qui charge la requête xml "RequestXML" envoyée par l'application client au service web **(1)**. La partie encerclée désigne les instructions et le code qui crée la classe "ParceXml" chargée de chercher et extraire les arguments et le service en question. Par la suite, le service proposé est utilisé comme n'importe quelle autre méthode grâce à la méthode "Interop.Application_ patrimoniale.lectureDonnees".

Le résultat est ensuite envoyé sous forme d'un document xml "xmlOutput" **(2)**

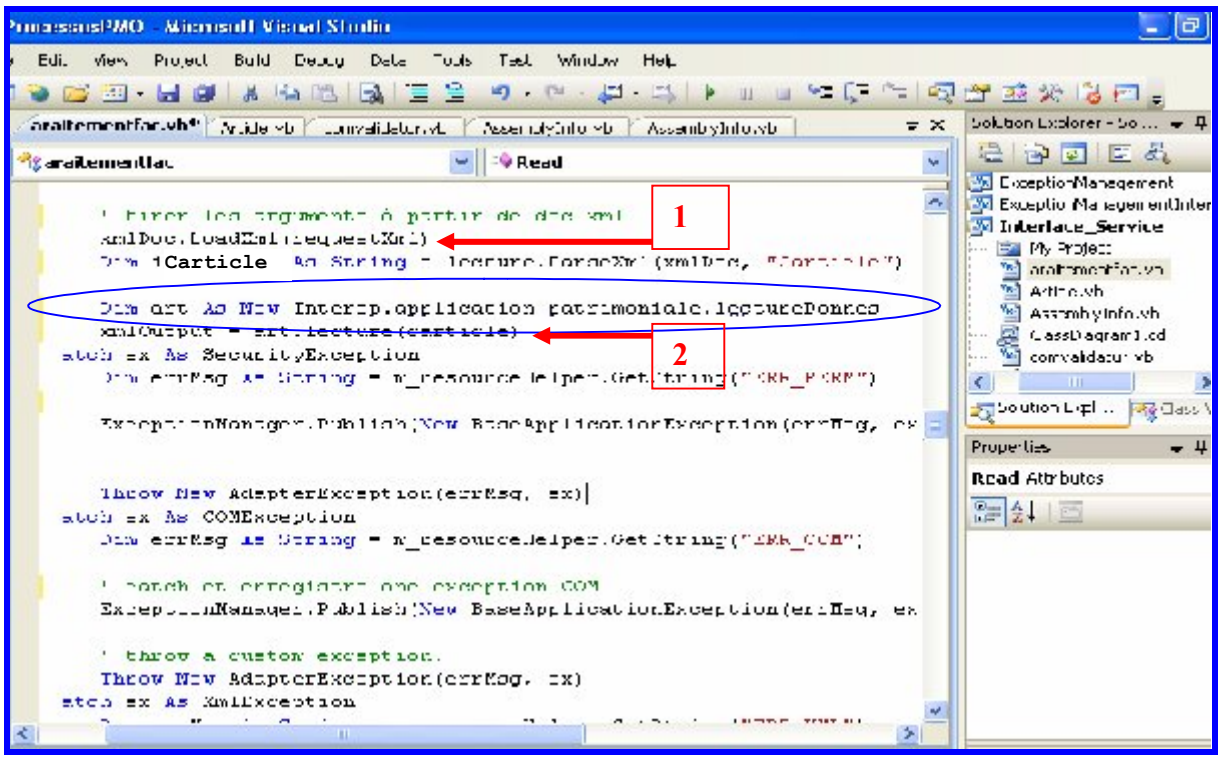


Figure IV.7 : Implémentation du code « Interface_Service »

5.3 Création de processus métier

La création de processus métier qui implémente notre objectif à savoir le traitement d'une commande client, implique la création d'un nouveau projet. Ce projet inclura toutes les classes nécessaires pour l'exécution et le contrôle du processus métier. Nous avons choisi le langage Visuel Basic sous la plate forme .Net, pour la création de ces classes.

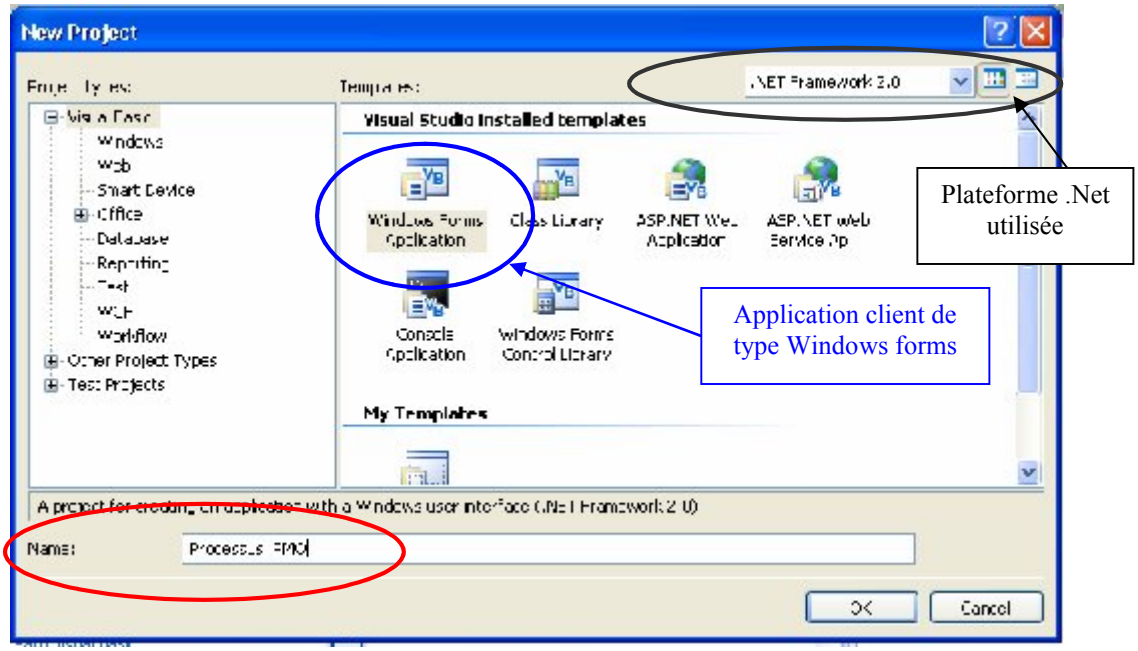


Figure IV.8 : Création d'un projet réalisant le processus de traitement de commandes de l'entreprise PMO (Moteur d'exécution).

Nous aurons besoin d'une interface qui communique avec l'utilisateur (facturier), pour saisir les informations nécessaires à l'évaluation de commande client, pour cela nous avons créé un nouveau projet (voir la figure IV.9).

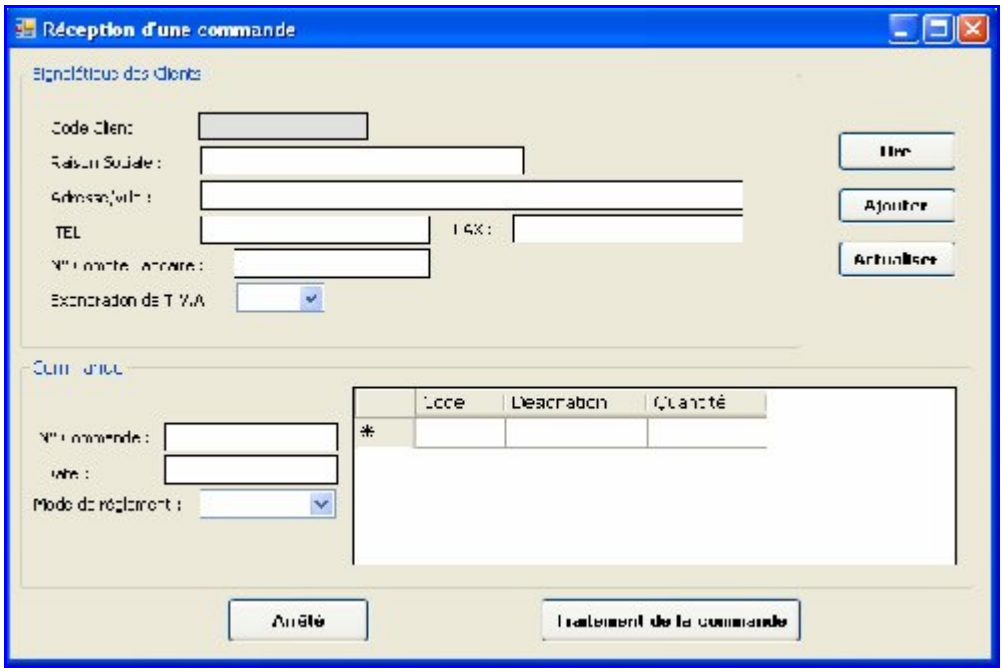


Figure IV.9 : Interface utilisateur interagisse avec le facturier.

La figure IV.10 présente une des classes de projet d'exécution de processus métier « DemArr_Processus », qui fait appel à nos services web.

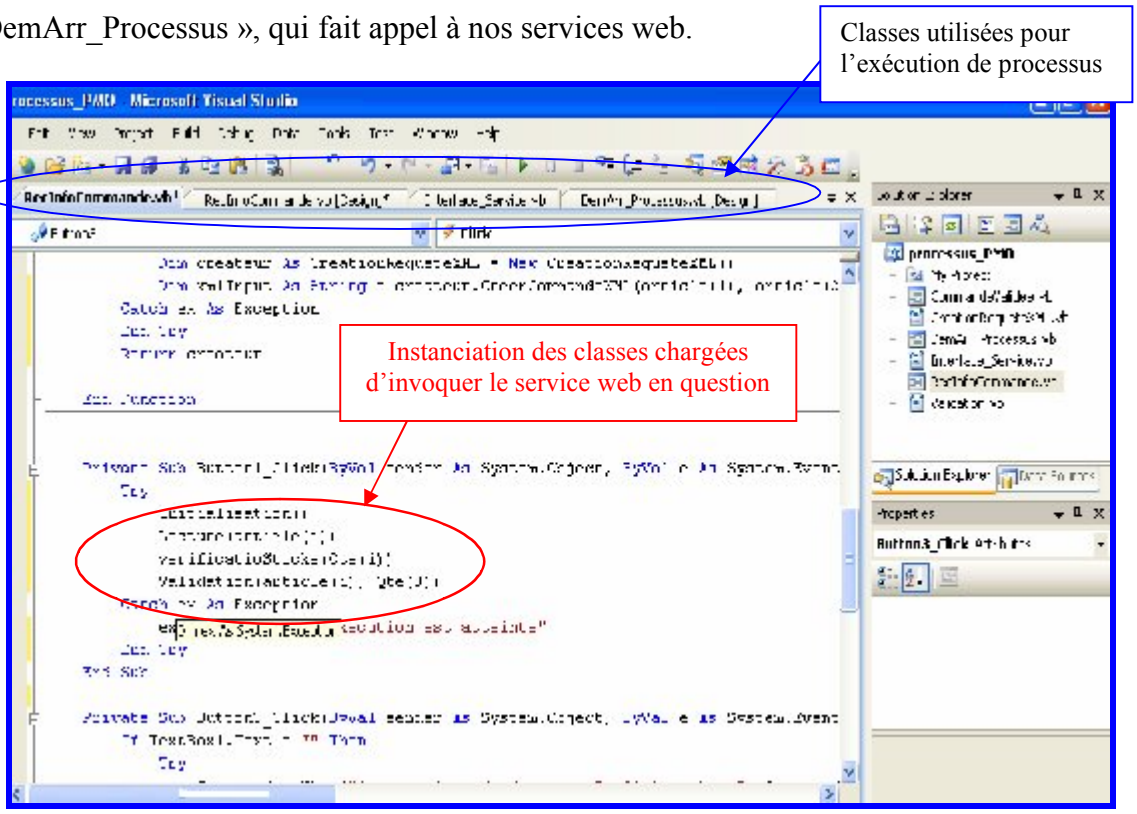


Figure IV.10 : Appel des services web.

Les onglets apparaissant dans cette même figure représentent toutes les classes indispensables déjà vues à l'exécution du processus

La partie encerclée désigne les instructions et le code qui crée la classe chargée de chercher le service web en question. En effet les Classes DemArr_Processus, Interface_Services, Validation, CreerRequeteXML, figurent dans le projet pour assurer l'exécution du processus.

6. Conclusion

Dans ce chapitre, nous avons proposé une étude de cas permettant l'application de notre processus proposé pour faire interopéré des applications hétérogènes d'une entreprise commerciale entre eux, et avec d'autres applications étrangères de ses unités de fabrication. En faisant migrer toutes leurs fonctionnalités interopérables vers le paradigme des services web[61]. Ces derniers sont prêts à la publication et l'intégration avec d'autre services pour une réutilisation éventuelle dans la réalisation de processus métiers d'autres systèmes orientée services.

L'application de notre processus sur une étude de cas a montré que notre processus, par opposition aux autres déjà vu (chapitre II), est facile à réaliser même dans le cas de grands systèmes patrimoniaux complexes. Ceci est grâce à l'utilisation des modèles d'arbre fonctionnels et d'objectifs, qui facilite la compréhension et l'analyse des systèmes existants.

Les résultats de validation de notre processus sur une application réelle nous montrent que nous avons réussit à atteindre nos objectifs, à savoir : la décomposition des systèmes existants en services web ayant une valeur ajoutée métier élevée dans l'ensemble des entreprises intégrées constituant l'e-Business. Et la recomposition de ces services web dans des processus métiers ayant un bon niveau de détail d'exécution, tout en minimisant l'intervention de la supervision humaine dans les phases du projet, et aussi minimisant la complexité engendrée par le système orienté services résultant.

[61] La technologie services web est une technologie efficace d'interopérabilité.

Conclusion générale

Conclusions et perspectives

L'e-Business représente le meilleur moyen pour que les entreprises commerciales et industrielles ou plus communément les grandes organisations économiques, subissent une modernisation et une réorganisation énorme et efficace. Il consiste à améliorer les relations avec le client et entre les partenaires d'affaires, et de réduire le coût et la complexité engendrée par l'entreprise.

L'étude de l'existant a montré les caractéristiques des solutions d'intégration proposées dans le domaine. La migration des applications vers des technologies plus avancées ou encore vers une technologie d'interopérabilité efficace 'Services Web', est considérée comme étant la meilleure solution qui permet une intégration flexible guidée par les besoins et basée sur l'existant, qui implémente les règles de gestion de l'entreprise dont dépend toute leur existence.

La grande ambition est de faire intégrer et interopérer un grand nombre d'entreprises, tout en réduisant la complexité des systèmes constituant l'e-Business.

Le travail présenté consiste à un objectif principal qui est la migration des systèmes d'information d'entreprises vers (ou la construction d') une architecture de services web. Il se focalise sur deux sous-problématiques complémentaires qui sont respectivement, la problématique de migration de noyau des entreprises tout en réduisant automatiquement la complexité de système cible qui influe directement sur leurs performances, et l'intervention décisive de la supervision humaine, particulièrement pour la définition des parties de code qui méritent d'être des services web dans le futur système intégré.

Le processus proposé dans ce travail recouvre toute les étapes nécessaires à la migration des systèmes d'information des entreprises vers une architecture orientées services web, dont l'objectif principal est l'intégration flexible des systèmes. Les étapes de ce processus sont inspirées principalement des travaux de G. Lewis et al **[48] [51]** et Smith et al **[54]**, avec l'amélioration constituant notre contribution qui consiste à ajouter une procédure stratégique de décomposition de système en services adéquats en un bon niveau de détail d'exécution, dans la phase de détection et d'identification des services web, et une procédure de composition des services en processus métiers. Ces procédures utilisent principalement une méthode de réingénierie basée sur la modélisation en entreprises. Notre idée vise à modéliser le métier de système à migrer en des modèles hiérarchiques, facilitant la compréhension de la

manière de son fonctionnement et réduisant la complexité pour pouvoir définir les services web intéressants. Ceci est réalisé par l'analyse de ces modèles, en suivrant les traces des résultats finaux de chaque fonction.

Outre, l'importance d'utiliser les méthodes de modélisation en entreprise est éprouvée par leur importance et efficacité dans les phases de modélisation des méthodes classiques de génie logiciel. Notre idée vise principalement à construire le modèle fonctionnel de l'ancien système et le modèle d'objectifs métiers d'intégration. Ensuite, grâce à l'utilisation de l'approche de *projection des modèles*, proposée nous pouvons détecter et décomposer automatiquement le système en des services adéquats « en bon niveau de granularité » et les adapter comme des services web. L'approche de projection des modèles est centrée principalement sur l'analyse de résultat finale et des arguments utilisés par chaque fonction, qui représente les nœuds de modèle fonctionnel.

Le quatrième chapitre a présenté certains détails techniques de notre processus, où il est présenté l'application des différentes activités de processus sur des applications de l'entreprise commerciale PMO (Production Machine Outils). D'après cette étude de cas nous pouvons dire que notre objectif de détection et d'intégration des services web en utilisant les techniques de réingénierie basées sur les modèles d'entreprises est atteint. Mais notre travail reste toujours guidé par de nombreuses issues qu'on gardera pour les futurs travaux :

- Nous proposons d'abord l'implémentation des différents modules nécessaires pour achever les phases de notre processus. Nous citons par exemple : le modeleur des différents langages vers le modèle d'arbre de fonction (FTM[62]), le module permettant la projection des modèles et la détection automatique du code qui implémente les services web à partir du code source du système, le module d'adaptation qui permet de générer le document WSDL à partir du code source, et enfin le module de création du document BPEL à partir des processus métiers de l'entreprise.

- Nous proposons l'utilisation des documents XML correspondants aux modèles fonctionnels de système existant et de modèles d'objectifs, afin de nous permettre d'étendre nos travaux futurs vers les web sémantiques. Cela nous permet de formaliser les concepts du modèle proposé par un langage standard formel tel que

[62] **FTM :** Function Tree Model

OWL[63] et construire une ontologie à partir des modèles fonctionnels, et donc se focaliser sur le problème complémentaire d'hétérogénéité sémantique.

- Une autre proposition très intéressante, est la réalisation de notre processus en utilisant des méthodes de modélisation formelles, qui utilisent des langages mathématiques, par exemple :

- Nous proposons d'un point de vue technique de déployer des registres (UDDI[64]) publiant les services web pour chaque unité de fabrication de l'entreprise PMO, ceci permet d'étendre les affaires entre ces différentes unités.

Pour terminer, nous souhaitons que cette contribution pourra être exploitée par l'entreprise **PMO** pour améliorer son système d'information, et plus spécialement de faire profiter leurs clients et leurs partenaires d'affaires des services de l'entreprise publiés sous web, en utilisant le registre des services web.

[63] **OWL :** Web Ontologie Langage.
[64] **UDDI:** Universal Discovery, Description and Integration.

Références bibliographiques

Références bibliographiques

[1] Manoj Seth, **Web services- A Fit for EAI**, p1, p2. http://www.developer.com/tech/article.php/10923_1489501.

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

[3] David S. Linthicum, **Enterprise Application Integration,** Ed: Addison Wesley, 2001 ISBN: 0-201-61583-5.

[4] 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.

[5] David A. Chappell, **Enterprise Service Bus,** Publisher: O'Reilly Media, First Edition June 2004. ISBN: 0-596-00675-6.

[6] Saïd Izza, **Intégration des systèmes d'information industriels : *une approche flexible basée sur les services sémantiques,*** thèse de doctorat de l'école supérieure nationale des Mines de Saint-Étienne, novembre 2006.

[7] S. Izza, L. Vincent, P. Burlat, H. Solignac et P. Lebrun, **Intégration d'applications : *Etat de l'art et perspectives*,** JIE 2'04, les 2 emes Journées d'Informatique pour l'Entreprise, université Saad Dahleb, Blida (Algérie) 2004.

[8] Thomas Jardin, **Solutions EAI : *Présentation et mise en œuvre*,** octobre 2003. http://www.sterlingcommerce.fr/Products/EAI.html.

[9] Fabien Cleuet, Vincent Goujon, Henry Peyret, **EAI et urbanisation : *comment réussir* ?** l'Association Française d'Adit et de conseil Informatique (AFAI), Mai 2001.

[10] François Rivard, Thomas Plantain, **EAI par la pratique,** Publisher Groupe Eyrolles, 2003. ISBN : 2-212-11199-1.

[11] Xebia : Business Integration Architects, **Comprendre et savoir utiliser un ESB dans une SOA,** livre blanc, 2007.

[12] Progress Sonic, **Architecture SOA (Service Oriented Architecture) distribuée: *l'intégration basée sur des standards, les avantages de l'ESB (Enterprise Service Bus)***, White Paper, 2007.

[13] Jihed Touzi, **Aide à la conception de système d'information collaboratif support de l'interopérabilité des entreprises,** thèse de doctorat de l'école des Mines d'Albi Carmaux, Novembre 2007.

[14] S. Janarthanan, A. Ekambaram, M.M. Kusum, N.K. Nair, **Integrating heterogeneous systems using Enterprise Application Integration**, research & analysis report, pp (1-20).

[15] Octo tecnologies, **Le livre blanc de l'EAI - Intégration des Applications d'Entreprise**, Octo Technologies, octobre 1999.

[16] eFORCE : **Comment tirer profit des web services dans le monde réel**, white paper, Mars 2003.

[17] Tanguy Crusson, **Business Process Management, de modélisation à l'exécution : Positionnement par apport aux Architectures Orientées Services**, INTALIO, 2003.

[18] Saïd Izza, Lucien Vincent, Patrick Brulat, **A Framework for Semantic Enterprise Integration,** proceedings de INTEROP-ESA'05 "Interoperability of Enterprise Software and Applications" pp (75-86), 2005.

[19] David S. Linthicum, **Next Generation Application Integration,** Ed: Addison Wesley, 2004. ISBN: 0-201-61583-5.

[20] Philippe Courty, **L'évolution des besoins et des solutions d'intégrations,** CJP 2005.

[21] EBM Websourcing : **Nouvelles technologies pour l'intégration : les ESB**, white paper, janvier 2006.

[22] Yuehui Chen, Ju Yang, Yong Zhang and Jiwen Dong, **Evolving Additive Tree Models for System Identification,** International Journal Of Computational Cognition, VOL. 3, NO. 2, JUNE 2005.

[23] Abdmouleh Anis, **composant pour la modélisation des processus métier en productique**, thèse de doctorat de l'Ecole Nationale d'Ingénieur de METZ, université de METZ, septembre 2004.

[24] Desmond DSouza, **Model-Driven Architecture and Integration Opportunities and Challenges,** 2001
http://www.ftp://ftp.omg.org/pub/docs/ab/01-03-02.pdf

[25] N. Boudjlida, K. Benali, H. Panetto, J. F. Petin, **Towards a Unified Language for Enterprise Modelling,** Proceedings de French I^3 (Information, Interaction, Intelligence) Workshop**,** Nancy, December 2002. France.

[26] Syntec Informatique, **Les Architectures Orientées services : le moteur de l'innovation,** livre blanc de Si (Syntec informatique), juin 2007.

[27] Marc Boulier, Luc Geoffray, Erwan Le Gouzouguec, Bertrand Masson, **Le livre blanc du BPM : l'orchestration de processus métier,** Vistali S.A, novembre 2002.

[28] Ian Gorton, Dave Thurman, Judi Thomson, **Next Generation Application Integration: Challenges and New Approaches,** Proceedings of the 27th Annual

International Computer Software and Applications Conference (COMPSAC'03), IEEE Computer Society, 2003.

[29] Octo tecnologies, **Architecture d'applications : la solution .NET (dotNET)**, Octo Technologies, janvier 2003.

[30] Phil Gilbert, **A Business Oriented Architecture: combining BPM and SOA for Competitive Advantage,** Mars 2007. http://www.bptrends.com/deliver_file.cfm?fileType=publication&fileName=TWO%2003-07-ART-ABusinessOrientedArchitecture-Gilbe.

[31] Bernard Manouvrier, Laurent Ménard, **Intégration applicative EAI, B2B, BPM et SOA.** Editeur : Hermès Lavoisier.

[32] Macdonald Ian, **Defining an Enterprise Application Integration (EAI) Architecture.** http://www.infosys.com/sap/enterprise-application-integration-services-and-solutions.pdf.

[33] Florent Arnaud, **EAI (Enterprise Application Integration) : principes, avantages, applications, exemples,** CNAM (Conservatoire National des Arts et Métiers), école d'enseignement de Lyon, 2004.

[34] Kiran Kanetkar, **A Roadmap to building an ESB,** Integration for Enterprise, Boston, May 2006.

[35] **E-Business BPM,** http://www.commentcamarche.net/entreprise/bpm.php3.

[36] William Tse, **Enterprise Application Integration,** UCL Computer Science. http://www.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-03-04/EAI.pdf

[37] Harry M. Sneed, Katalin Erdos, **Extracting Business Rules from Source code,** Proceeding of the 4th IWPC 1996, IEEE Computer Society, Berlin, p: 240-247.

[38] Harry M. Sneed, **Wrapping Legacy Software for Reuse in a SOA,** Proceeding of IEEE Conference on Software Maintenance and Reengineering (CSMR'05), 2005.

[39] P. Sarang, R. Loganathan, F. Jennings, M. Juric, **SOA Approach to Integration: XML, Web services, ESB, and BPEL in real-world SOA projects.** Packt Publishing 2007. ISBN: 13 978-1-904811-17-6

[40] Octo technologies, **Architectures Orientées Services (SOA) : Une politique de l'interopérabilité,** Octo Technologies livre blanc, Mars 2005.

[41] J. Ziemann, K. Leyking, T. Kahl, D. Werth, **Enterprise Model driven Migration from Legacy to SOA,** Software Reengineering and Services Workshop, 2006.

[42] BPMI, **Business Process Management Initiative,** http://www.bpmi.org/.

[43] Programmez (F. Tonic, M. Boulier, B. Paroissin, J. Clune, F. Bernnard, M. Gardette, etc.), **SOA : votre nouvelle architecte**, Le magazine du développement : Programmez!, N° : 78, Juin 2006.

[44] Xebia France, **Mise en œuvre d'une SOA : Les clés de succès**, Février 2008. http://www.archives-de-méthodes--agiles-par-J2EE--Agilité&SOA-le-blog-de-XebiaFrance.htm.

[45] Hurry M. Sneed, **Integrating legacy Software into a Service Oriented Architecture,** Proceedings of IEEE Conference on Software Maintenance and Reengineering (CSMR'06), IEEE 2006.

[46] V. N. Gudivada, J. Nandigam, **Enterprise Application Integration Using Extensible Web Services**, Proceedings of the IEEE International Conference on Web Services (ICWS'05), 2005 IEEE.

[47] Rainer Gimnich, **SOA Migration – Approaches and Experience,** Proceedings of the 10th IEEE European Conference on Software Maintenance and Reengineering (CSMR'08), 2008.

[48] G. Lewis, E. Morris, D. Smith, L. Wrage, L. O'Brien, **Service-Oriented Migration and Reuse Technique (SMART),** Proceedings of 13th IEEE International Workshop on Software Technology and Engineering Practice (WSTEP'05), September 2005.

[49] M. Jiang, A. Willey, **Service-Oriented Architecture for Deploying and Integrating Enterprise Applications**, Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005 IEEE.

[50] Oracle, **Oracle IT Modernization Series: Why Modernize?,** an Oracle White Paper, September 2008. http://www.oracle.com/technologies/modernization/docs/whymodernize.pdf.

[51] Oracle, **Oracle IT Modernization Series: The Types of Modernization,** an Oracle White Paper; September 2008. http://www.oracle.com/technologies/modernization/docs/typesofmodernization.pdf

[52] A. Arsanjani, **Service-Oriented Modeling and Architecture : How to identify, specify, and realize services for your SOA**, November 2004. http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/,

[53] G.A. Lewis, E.J. Morris, D.B. Smith, S. Simanta, **SMART: Analyzing the reuse potential of migrating legacy components to a service-oriented architecture**, Proceedings of the 10th IEEE European Conference on Software Maintenance and Reengineering (CSMR'08), September 2008.

[54] S. Cetin, N. Ilker Altintas, H. Oguztuzun, A.H. Dogru, O. Tufekci, S. Suloglu, **A Mashup-Based Strategy for Migration to Service-Oriented Computing,** Proceeding of IEEE International Conference on Pervasive Services (CPS'07), July 2007, Istanbul, Turkey.

[55] John Bergey, Liam O'Brien, Dennis Smith, **Application of Options Analysis for Reengineering (OAR) in a Lead System Integrator (LSI) Environment,** March 2003. TN: CMU/SEI-2003-TN-009.

[56] Mark S. Fox, Michael Gruninger, **Enterprise Modelling, Publication** of The American association for Artificial Intelligence, AI Magazine 1998. 0738-4602-1998.

[57] R. Martin Andersson, **Reverse Engineering of Legacy Systems: from value-based to object-based models,** thèse de doctorat de Ecole Polytechnique Fédérale de Lausanne (EPFL), 1996.

[58] Jianjun Pu, Zhupeng Zhang, Yang Xu and Hongji Yang, **Reusing COBOL Code With UML collaboration Diagrams via a Wide Spectrum Language,** Proceeding of IEEE International Conference on Information Reuse and Integration (IRI'05), IEEE Systems, Man and Cybernetic Society, 2005, pp. 78-83.

[59] Jean-Marc Moster, **Microsoft Visuel Basic 6,** grand livre 1999. ISBN:2-7429-1384-X.

[60] Lasserre Philippe, C**ours Visuel Basic Dot Net (VB.Net)**, Mars 2005. http://plasserre.developpez.com/vbintro.htm.