الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université 08 Mai 1945-Guelma

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

جامعة 08 ماي 45 قالمة
مصلحة الإعارة الخارجية
كلية الرياضيات، الإعلام الآلي وعلوم المادة

Mémoire de fin d'étude de Master

Filière : Informatique

Spécialité : Ingénierie des Medias

Thème :

Fouille des processus métiers : Approche déclarative

Sous la direction de :

Mr. Khebizi Ali

Présenté par :

N'guessan B Gérard,

Med Fadel Ould Ebaby

Juin 2012

Remerciements

C'est avec un grand plaisir que nous réservons cette page en signe de gratitude et de profonde reconnaissance d'abord à Dieu le tout Puissant en lui rendant toute la grâce et aussi tous ceux qui ont bien voulu apporter l'assistance nécessaire au bon déroulement de ce travail.

Nous voulons remercier notre encadreur Monsieur Khebizi Ali qui n'a pas épargné le moindre effort dans l'encadrement de ce mémoire afin de nous permettre de défier les entraves rencontrées, de travailler avec volonté et qui était toujours disponible pour nous orienter à entreprendre les bonnes décisions. Nous espérons être à la hauteur de sa confiance. Qu'il trouve dans ce travail le fruit de son effort et l'expression de notre profonde gratitude.

Nous tenons à exprimer notre profond respect et nos vifs remerciements envers les membres de jury pour l'honneur qu'ils nous ont fait en acceptant de présider et d'examiner notre travail.

Un remerciement ensoleillé à Monsieur Mourad.Hadjeris, pour sa bienveillance, sa fraternité, sa gentillesse et son soutien constant qui nous a donné courage pour l'élaboration de ce mémoire. Nous apprécions ses grandes qualités morales et son extrême modestie. Qu'il trouve dans ce travail l'expression de notre profond respect et nôtres infinie reconnaissance.

Nous remercions aussi tous nos amis, ainsi que les membres de différentes communautés étrangère de Guelma pour l'ambiance chaleureuse qui règne entre nous et pour leur bonne humeur qui n'ont jamais hésité à nous offrir leur aide, leur coopération et l'élan qu'ils ont su insuffler à notre travail.

Il est indispensable de ne pas rater cette occasion pour exprimer nos reconnaissances à tous nos professeurs du département d'informatique de Guelma pour leurs efforts qui ont guidé nos pas et enrichi notre connaissances.

Je dédie ce modeste travail.

A mes parents adorables, ceux qui ont fait que je sois ici aujourd'hui par leur prière et leurs conseils chaleureux qui ne cessent de tourner encore dans ma tête.

A tous mes oncles et mes tantes qui m'ont élevé et éduqué... vous qui avez toujours été là pour moi et n'avez jamais cessé de croire en moi, aucun mot ni aucune langue ne pourrait exprimer ma profonde gratitude à votre égard.

A tous mes frères, mes sœurs, mes cousins et cousines adoré qui n'ont pas cessés de renouveler la confiance qu'ils ont en moi.

A tous mes professeurs qui m'ont formés et qui m'ont permis d'en arriver jusque là.

Et enfin A tous mes amis,
A tous ceux qui ont veillé à mon instruction
A tous ceux que j'aime et qui m'aiment
Avec l'expression de tous mes sentiments et mon respect.

Que Dieu accepte notre travail.

N' guessan .Behou. Gérard

Résume

Aujourd'hui la masse d'information disponible sur le Web est gigantesque et diversifiée. Cette diversité peut aller des simples messages, vidéo ou images à des bases de données professionnelles jusqu'au processus métier spécifiques à des secteurs particuliers.

Si les moteurs de recherche permettent de fournir des réponses à des requêtes bien formalisées et si les SGBD filtrent les données existantes pour n'afficher que celles vérifiant certains critères, cependant la recherche et le filtrage des processus métier restent un domaine peu ou mal exploré.

Dans ce mémoire, nous présentons une solution permettant de faciliter la fouille des processus métier à travers une approche déclarative. En effet, Nous utilisons les protocoles métier (Business Protocol) pour modéliser les échanges entre les différentes entités (fournisseurs, clients, intermédiaires). Puis, Nous utilisons des langages de requête (SPAROL, XQUERY) permettant la fouille des données.

Sommaire

Liste des Figures

Tableaux

Liste des abréviations et acronymes

AFD	Automate Fini Deterministe
API	Application Programming Interface
BPEL	Business process execution language
BPM	Business Process Management
BPMN	Business Process Modeling Notation
BPMN	Business Process Modeling Notation
CORBA	Object Request Broker Architecture.
DSS	Decision Support System
EAI	Enterprise Application Intégration
EDI	Echange de Données Informatisées
ERP	Eentreprise Ressource Planning
FTP	File Transfers Protocol
HTTP:	Hyper Text Transport Protocol.
IHM:	Interaction Homme Machine
MIME	Multipurpose Internet Mail Extensions
NASSL:	Network Accessibility Service Specification Language - IBM
OMG:	l'Object Management Group.
OSI	Open Systems Interconnection
OWL-S	Ontology Web Language for Services
P2P:	Pair-to-Pair
RdP:	Réseaux de Pétri
RPC:	Remote Procedure Call
S.I:	Système d'Information
SCL :	Service Conversation Langage - Microsoft
SGBD :	Système de Gestion de Base de Données
SGML	Standard Generalized Markup Language
SITI	Système d'Information et Traitement Informatique
SMTP:	Simple Mail Transport Protocol.
SOA:	Service Oriented Architecture
SOAP:	Simple Object Access Protocol,
UDDI:	Universal Description, Discovery and Integration
URI	Uniform Resource Identifier
WS-CDL:	Web Service Choreography Description Language
WSCI:	Web Service Choreography Interface
WSDL:	Web Service Description Language
WSFL:	Web Service Flow Language

W3C	World Wide Web Consortium
XML:	eXtensible Markup Language

Introduction Générale

Aujourd'hui, les Systèmes d'information (SI) sont omniprésents. Avec le développement et l'engouement des réseaux de communication, ces systèmes sont de plus en plus distribués, impliquant divers partenaires. Cependant, la diversité et l'hétérogénéité du matériel, des plateformes et des langages engendrent des incompatibilités entre ces systèmes. Les technologies d'intégration et, particulièrement, les services Web, en tant qu'ensemble de standards basés sur XML encouragent les différents partenaires à leur utilisation en vue de publier, invoquer et rechercher des applications disponibles sur le Web et répondant à leur spécification.

Par ailleurs, ces services Web permettent d'exprimer les activités associées à la logique métier d'une organisation : Ces les **processus métiers**. Ces derniers présentent un capital informationnel et un savoir faire de l'organisation. Différents modèles existent pour représenter les protocoles métiers. La question posée est de savoir comment rechercher et extraire ces processus métiers ou une partie spécifique les concernant ?

Bien que diverses méthodes aient été proposées pour répondre à cette question, elles restent limitées à leur aspect procédural et nécessitant un savoir faire informatique.

Pour répondre à ce déficit, nous proposons dans ce mémoire une **approche déclarative** permettant d'explorer les bases de données des protocoles métiers. Cela présente plusieurs avantages pour les gestionnaire de protocoles qui désirent rechercher certaines spécifications avec un niveau d'abstraction élevé (sans ce soucier ni du langage, ni du lieu de disponibilité des processus en question).

Dans ce mémoire, nous proposons une approche pour pallier à ce déficit.

Le mémoire est structuré comme suit :

Au premier chapitre nous palerons des systèmes d'information, tout en abordant leurs caractéristiques et leurs avantages.

Le deuxième chapitre est consacré à la technologie des services Web, ses standards techniques (*SOAP, WSDL, IDDI*), et son mode de fonctionnement.

Le troisième chapitre vise les différents modèles de représentation des processus métiers, et les concepts d'évolution de protocoles ainsi que les notions de changements.

Le quatrième chapitre est le centre de notre conception. Il traite les langages de requêtes et la conception de l'approche proposée.

Le cinquième chapitre est dédié a l'implémentation logicielle. On présentera les différentes interfaces ainsi que les fonctionnalités du système

. Enfin nous clôturerons par une conclusion générale.

Chapitre I : Systèmes d'Informations Distribués

Introduction

L'évolution des technologies informatiques au cours des dernières années a entraîné des modifications radicales dans la conception des applications. Les structures de dimension internationales ; telles que la défense, les banques ou les grandes entreprises voient leurs activités s'étendre planétairement et ne peuvent plus centraliser toute l'information sur un système unique. En outre, de plus en plus de transactions courantes s'effectuent via les réseaux. Tout ceci a favorisé la démocratisation des communications informatiques (à commencer par les messageries électroniques et la navigation sur le Web) puis, plus récemment, l'émergence des applications distribuées. [1]

À travers ce chapitre nous allons présenter les systèmes d'information au plan générique, puis prouver sa place au sein de la nouvelle conjoncture des systèmes d'information distribués.

I. Notions de Système d'informations

1. Définition

L'information se présente sous trois formes : les données, les connaissances et les messages. D'où un « système d'information » est l'ensemble organisé de ressources des moyens techniques et humains qui permet de stocker, de traiter, de transmettre, regrouper, de classifier, de diffuser de l'information sur un environnement donné [5]. On dira donc qu'un système d'information est « tout moyen dont le fonctionnement fait appel d'une façon ou d'une autre à l'électricité et qui est destiné à élaborer, traiter, stocker, acheminer, présenter ou détruire l'information » [2]. Il est basé sur un certain nombre de fonctionnements d'objets à savoir :

-un ensemble structuré de données (Base de Données).

-une batterie de logiciels (systèmes d'exploitation, applications ... etc.).

-ressources humaines et matériels.

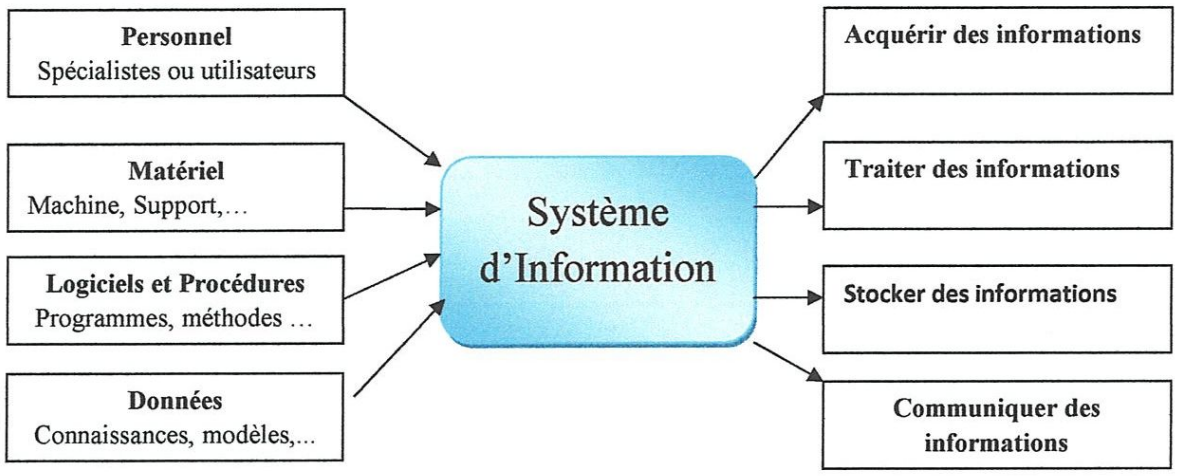


Figure 1.1 : *Structure d'un système d'information [2]*

Le système d'information a pour mission de créer des images ou des représentations de ce qui se passe autour et dans les systèmes opérant et de décision, puis de les ranger dans sa mémoire. Ce qui implique la collecte (ou la création), le stockage, la communication et le traitement d'informations.

2. Fonctionnalité d'un système d'information

le système d'information comporte (les processus de l'entreprise, les informations manipulées par ces processus et les fonctions qui traitent ces informations). Le système information comporte les composants techniques qui supportent le système d'information en permettant de l'automatiser et le distribuer tels que : [6]

a. La création ou la collecte des informations

L'information n'existe pas spontanément, il faut la créer et s'assurer qu'elle est de bonne qualité : c'est là une fonction vitale pour les organisations. Les informations doivent être saisies, c'est à dire mises en forme sur un support sans cela elles n'existent que pour celui qui la reçoit, et disparaissent immédiatement. Lors de la mise sur support, il convient de s'assurer que l'information représente bien la réalité (objet, activité, décision) qu'elle est censée représenter. [7]

b. Le stockage ou mémorisation

L'information est volatile, elle disparaît aussitôt que créée si sa conservation n'est pas organisée. La fonction de mémorisation consiste à organiser l'accès et la conservation de l'information.

c. Le traitement

La création de certaines informations, notamment les décisions, impliquent des opérations logiques et des calculs. Cette fonction est souvent mise en avant alors que les fonctions de création, de mémorisation et communication sont tout aussi vitales et la précèdent. [7]

d. La communication

L'idéal consiste à fournir de l'information à qui en a besoin quand il en a besoin. La communication est essentielle. Il est en effet exclu que chaque composante d'une organisation dispose en permanence de l'ensemble des informations créées dans l'organisation et susceptibles de l'intéresser. La communication concerne l'acheminement des informations. Elle est affaire de réseaux dits de communication. Elle ne doit pas être confondue avec l'accès aux informations à l'intérieur de la mémoire qui fait partie de la fonction de mémorisation. [7]

3. Rôle des S.I dans les organisations

Les SI occupent une place importante dans les investissements des entreprises et leur diffusion relève désormais d'un choix stratégique. A partir d'une certaine taille, les problématiques organisationnelles au sein des entreprises (découpage des périmètres des unités, relations hiérarchiques, structure de l'organisation) ne peuvent être dissociées d'une réflexion sur l'urbanisation du SI et l'élaboration d'un schéma directeur. Plusieurs travaux suggèrent que les SI auraient une dimension stratégique au sens d'être une ressource pouvant générer un avantage concurrentiel durable grâce à trois rôles qu'on retrouve dans les différents travaux : le contrôle et la normalisation, la coordination des différentes entités et le pilotage de l'organisation. [9]

Les systèmes d'information jouent un rôle opérationnel, tactique et stratégique pour toute organisation.

a. Les systèmes d'information comme dispositif de contrôle et de normalisation

Les SI s'insèrent dans la stratégie des firmes avec des objectifs récurrents de réduction des coûts et d'amélioration de la productivité. Les dimensions formelles de contrôle et d'évaluation sont devenues de plus en plus importantes et se traduisent par des exigences en matière de visibilité sur les processus et leur output. On retrouve dans les différentes typologies des SI une fonction de contrôle des processus au sens de l'agrégation des données dans une logique analytique. Les possibilités de formalisation offertes par les SI

sont particulièrement intéressantes pour matérialiser les dispositifs de contrôle au sein des organisations (états comptables, tableaux de bord, indicateurs de reporting et de performance).

On pourrait ainsi dire que les SI portent en eux un système de contrôle plus ou moins explicite. Cette fonction s'avère encore plus importante dans les organisations étendues qui doivent consolider les informations de plusieurs entités pour refléter un état le plus proche de la réalité dans une logique décisionnelle.

Au-delà du système de contrôle, la mise en place d'un SI est concomitante à la création d'une grammaire technique avec un processus de normalisation des langages et l'automatisation du traitement des données (EDI, ERP, workflow…). On voit alors se dessiner un processus informationnel qui permet de diminuer les coûts de coordination. Cette « normalisation électronique » a souvent été nécessaire dans l'intégration logistique de différents acteurs au sein d'une chaîne de valeur.

b. Les systèmes d'information comme dispositif de coordination des entités

Les organisations étendues doivent faire face à des problèmes de coordination plus complexes en raison d'un nombre important d'entités et d'une dimension spatiale et temporelle extensive et variable au fur et à mesure des décisions stratégiques (alliances, acquisitions, cessions, quasi-intégration…). La cohérence des décisions multiples repose alors sur des mécanismes de coordination où la communication d'information est primordiale, ce qui explique l'adoption rapide des SI dans les grandes organisations d'un point de vue historique. Cette logique a été élargie depuis plusieurs années aux relations externes de type « clients-fournisseurs » avec une imbrication de plus en plus forte des SI des différents acteurs dans la chaîne de valeur pour aboutir à l'apparition du concept de Système d'Information(S.I) interroganisationnels. Les travaux s'accordent ainsi à présenter les SI déployés dans les relations interentreprises et intra-entreprises comme des facilitateurs de la coordination entre les différents acteurs. Certains travaux ont souligné également le rôle des SI comme réseau de flux d'information pour créer et maintenir les contrats constitutifs de l'entreprise. Cette orientation souligne la notion d'engagement des acteurs via le SI.

La notion de modélisation de l'organisation est également présente dans le concept de SI car ce dernier doit permettre une structuration des données dans une logique organisationnelle définie selon la mise en adéquation de ressources et d'objectifs. Certains SI permettent de cartographier les entités de l'organisation et de les coordonner dans une

logique d'action organisée afin d'atteindre les objectifs fixés au préalable. L'alignement des différents sites ou groupes de travail est souvent une condition essentielle dans l'atteinte des objectifs qui nécessite une cognition distribuée avec des SI comme support. L'une des autres difficultés dans les organisations étendues est de savoir quelles sont les capacités organisationnelles des différentes entités, d'autant plus lorsqu'elles sont réparties d'un point de vue géographique et qu'elles possèdent un certain degré d'autonomie. Il est ainsi parfois difficile pour une organisation de connaître le fonctionnement réel de ses propres entités. En ce sens, les outils de gestion basés sur les SI permettent une investigation du fonctionnement organisationnel. [9]

c. Les systèmes d'information comme dispositif de pilotage

Les Système d'Information peuvent être analysés comme un outil de management qui explicite et réorganise les relations et activités de travail entre les différentes entités. Cela positionne le SI comme une interface entre le système de pilotage, qui conçoit, organise le système de représentations (les objectifs, les orientations, les projets de l'entreprise) et décide, et le système « opérant », opérationnel, qui conduit les actions quotidiennes. La figure1.2 résume le rôle du S.I en tant qu'outil de management et de pilotage des organisations avec des actions de commande, de régulation et de prise de décision entre le système décisionnel et le système opérationnel. On retrouve une approche classique des SI nommée *Décision Support System* (DSS), qui les positionne comme un processus de résolution de problèmes et de prises de décision pour les acteurs à partir du traitement des informations. On peut également déduire à partir de la figure que le fonctionnement des organisations repose sur des boucles de rétroaction entre la direction et les entités. Ces interactions supposent alors un système de communication dans un environnement où la réactivité constitue souvent un facteur clé de succès. Ces interactions s'appuient souvent sur le SI pour les échanges formels et visent à réguler les écarts constatés même si les relations informelles demeurent nécessaires pour enclencher des actions de régulations. [9]

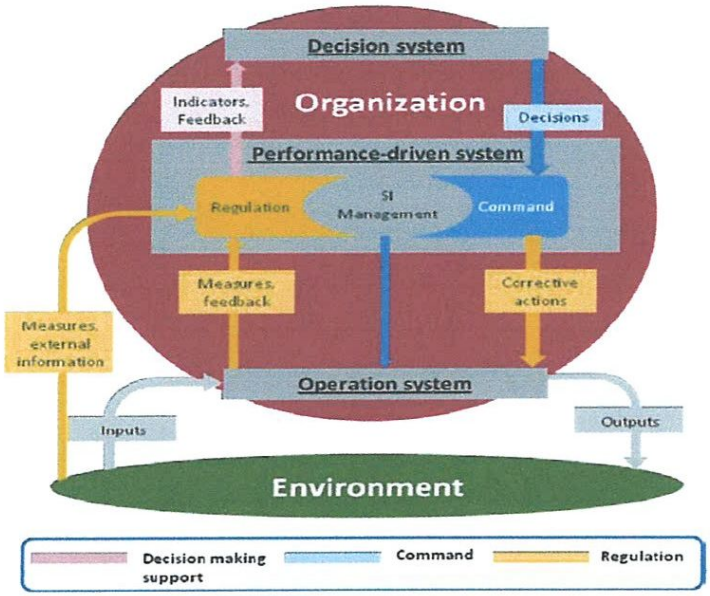


Figure 1.2: *Exemple de système de pilotage [9]*

d. Les différentes architectures des systèmes d'informations

Une application informatique peut être découpée en trois niveaux d'abstraction distincts :

- **La couche de présentation**, encore appelée interaction homme machine (IHM) permet l'interaction de l'application avec l'utilisateur. Cette couche gère les saisies au clavier, à la souris et la présentation des informations à l'écran. Dans la mesure du possible, elle doit être conviviale et ergonomique.
- **La logique applicative**, les traitements, décrivant les travaux à réaliser par l'application. Ils peuvent être découpés en deux familles :

-les traitements locaux, regroupant les contrôles effectués au niveau du dialogue avec l'IHM, visant essentiellement le contrôle et l'aide à la saisie,

-les traitements globaux, constituant l'application elle-même. Cette couche, appelée Business Logic ou couche métier, contient les règles internes qui régissent une entreprise donnée.

- **Les données**, ou plus exactement l'accès aux données, regroupant l'ensemble des mécanismes permettant la gestion des informations stockées par l'application. [11]

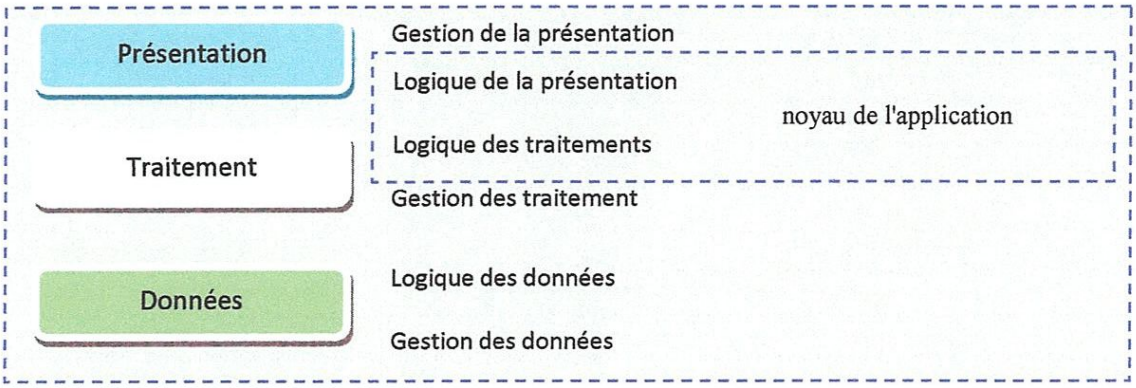


Figure 1.3 : *Les trois niveaux d'une application informatique*

Le noyau de l'application est composé de la logique de l'affichage et la logique des traitements. Le découpage et la répartition de ce noyau permettent de distinguer les architectures applicatives suivantes :

- l'architecture 1-tiers,
- l'architecture 2-tiers,
- l'architecture 3-tiers,
- Les architectures n-tiers.

i. L'architecture un tiers

Dans une application un tiers, les trois couches applicatives sont intimement liées et s'exécutent sur le même ordinateur. On ne parle pas ici d'architecture client-serveur, mais d'informatique centralisée.

Dans un contexte multiutilisateur, on peut rencontrer deux types d'architecture mettant en œuvre des applications un tiers :

- Des applications sur site central,
- Des applications réparties sur des machines indépendantes communiquant par partage de fichiers.
- Les solutions sur site central (mainframe)

Les applications sur site central furent les premières à proposer un accès multiutilisateurs. Dans ce contexte, les utilisateurs se connectent aux applications exécutées par le serveur central (le mainframe) à l'aide de terminaux passifs se comportant en esclaves. C'est le serveur central qui prend en charge l'intégralité des traitements, y compris l'affichage qui est simplement déporté sur des terminaux passifs.

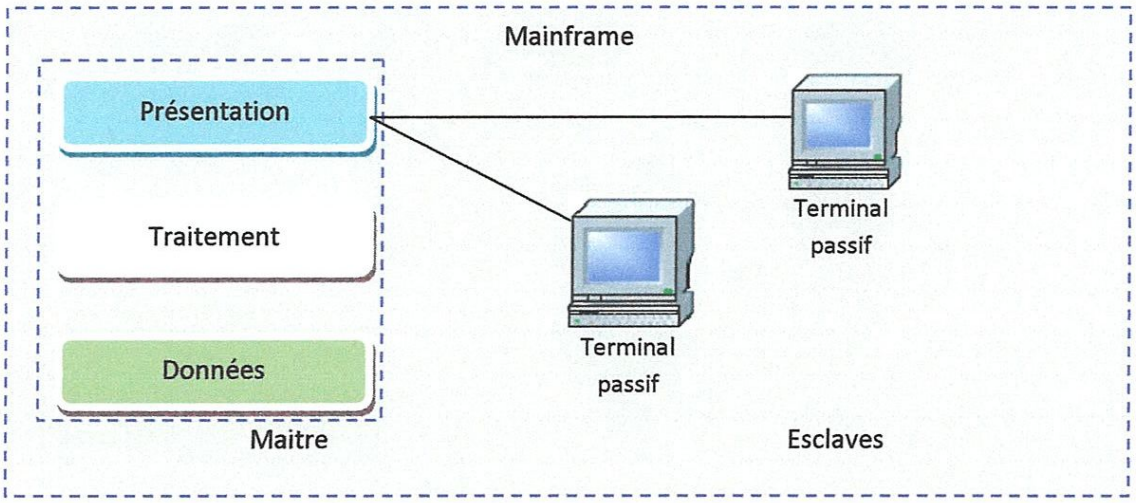


Figure 1.4 : *Architecture d'une application sur site central*

Ce type d'organisation brille par sa grande facilité d'administration et sa haute disponibilité.

- **Les applications un tiers déployées**

Avec l'arrivée dans l'entreprise des premiers PC en réseau, il est devenu possible de déployer une application un tiers sur plusieurs ordinateurs indépendants.

Dans ce contexte, plusieurs utilisateurs se partagent des fichiers de données stockés sur un serveur commun. Le moteur de base de données est exécuté indépendamment sur chaque poste client. La gestion des conflits d'accès aux données doit être prise en charge par chaque programme de façon indépendante, ce qui n'est pas toujours évident.

- **Limitations**

Les applications sur site central souffrent d'une interface utilisateur en mode caractères et la cohabitation d'applications micro exploitant des données communes n'est pas fiable.

ii. L'architecture 2-tiers

Dans une architecture deux tiers, encore appelée client-serveur de première génération ou client-serveur de données, le poste client se contente de déléguer la gestion des données à un service spécialisé. Le cas typique de cette architecture est l'application de gestion fonctionnant sous MS-Windows et exploitant un SGBD centralisé.

Le dialogue entre client et serveur se résume donc à l'envoi de requêtes et au retour des données correspondant aux requêtes. Ce dialogue nécessite l'instauration d'une communication entre client et serveur.

- Le client, c'est le programme qui provoque le dialogue,

- Le serveur, c'est le programme qui se contente de répondre au client.

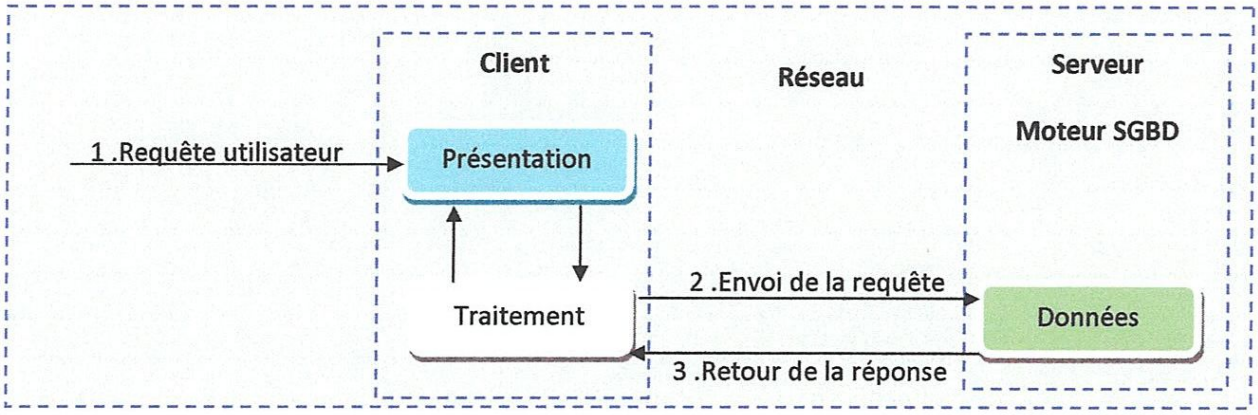


Figure 1.5 : *Accès aux données en mode deux tiers*

♦ **Limites du client-serveur deux tiers**

- Ce type d'architecture est grandement rigidifié par les coûts et la complexité de sa maintenance.
- Les applications se prêtent assez mal aux fortes montées en charge car il est difficile de modifier l'architecture initiale,
- La relation étroite qui existe entre le programme client et l'organisation de la partie serveur complique les évolutions de cette dernière

Malgré tout, l'architecture deux tiers présente de nombreux avantages qui lui permettent de présenter un bilan globalement positif :

- Elle permet l'utilisation d'une interface utilisateur riche,
- Elle a permis l'appropriation des applications par l'utilisateur,
- Elle a introduit la notion d'interopérabilité.

iii. L'architecture 3-tiers

L'architecture trois tiers, encore appelée client-serveur de deuxième génération ou client-serveur distribué, sépare l'application en trois niveaux de service distincts :

- premier niveau : l'affichage et les traitements locaux (contrôles de saisie, mise en forme de données...) sont pris en charge par le poste client,
- deuxième niveau : les traitements applicatifs globaux sont pris en charge par le service applicatif,
- troisième niveau : les services de base de données sont pris en charge par un SGBD.

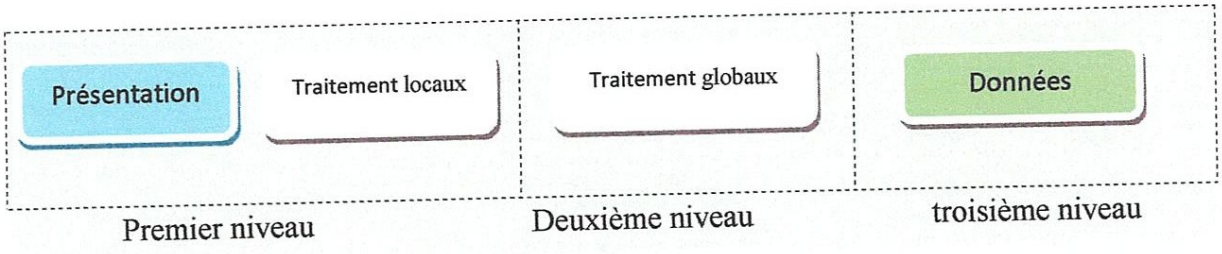


Figure 1.6 : *Le découpage d'une application en pavés fonctionnels indépendants*

Dans ce but, l'architecture trois tiers applique les principes suivants :

- Les données sont toujours gérées de façon centralisée,
- la présentation est toujours prise en charge par le poste client,
- la logique applicative est prise en charge par un serveur intermédiaire.

L'application de ces technologies dans une architecture trois tiers est complexe et demande des compétences très pointues, du fait du manque de standards. Malgré ses avantages techniques évidents, ce type d'application fut souvent jugé trop coûteux à mettre en place.

- **Limitations**
- le serveur HTTP constitue la pierre angulaire de l'architecture et se trouve souvent fortement sollicité et il est difficile de répartir la charge entre client et serveur. On se retrouve confronté aux épineux
- problèmes de dimensionnement serveur et de gestion de la montée en charge rappelant l'époque des mainframes.
- De plus, les solutions mises en œuvre sont relativement complexes à maintenir et la gestion des sessions est compliquée.

Le juste équilibrage de la charge entre client et serveur semble atteint avec la génération suivante : les architectures n-tiers (service Web).

iv. L'architecture n-tiers

L'architecture n-tiers a été pensée pour pallier aux limitations des architectures trois tiers et concevoir des applications puissantes et simples à maintenir. Ce type d'architecture permet de distribuer plus librement la logique applicative, ce qui facilite la répartition de la charge entre tous les niveaux.

Cette architectures offrent une plus grande souplesse d'implémentation et facilitent la réutilisation des développements. Ce type d'architecture supprime tous les inconvénients des architectures précédentes.

- Elle permet l'utilisation d'interfaces utilisateurs riches,
- Elle sépare nettement tous les niveaux de l'application,
- Elle offre de grandes capacités d'extension,
- Elle facilite la gestion des sessions.

En fait, l'architecture n-tiers qualifie la distribution d'application entre de multiples services et non la multiplication des niveaux de service. Avec les applications n-tiers, on dispose enfin d'une vision cohérente du système d'information. Tous les services sont représentés sous la forme d'objets interchangeables qu'il est possible d'implanter librement, en fonction des besoins. Le potentiel de ce type d'application est très important et permettrait enfin l'utilisation d'objets métiers réutilisables.

II. Problématique de l'intégration des S.I Distribués

L'intégration, actuellement terme très en vogue, est devenue pour beaucoup d'entreprises, une nécessité incontournable pour faire face aux exigences sans cesse évolutives du marché. Les contraintes qui pèsent de nos jours sur les entreprises sont souvent directement répercutées sur leurs systèmes d'information, à qui on demande d'être sans cesse plus agiles afin de pouvoir mieux soutenir la stratégie de l'entreprise. Pour faire face à ces nouvelles exigences, les entreprises ont très souvent recours au concept d'intégration, et parfois de façon plus spécifique au concept d'interopérabilité afin d'interconnecter leurs applications informatiques.

1. Problématique de l'interopérabilité

Les partenaires ont besoin d'échanger des données syntaxiquement et sémantiquement interopérables. Il faut donc disposer de formats et d'une ontologie en commun pour pouvoir appréhender ces informations aux plans syntaxiques et sémantiques. Selon le type de processus (conception, production, gestion globale…), et l'acteur en charge de la décision, différents types d'informations doivent être partagés. Pour cela, un format d'échange général et une politique d 'intégration des contraintes de sécurité doivent être définis. Une technique simple est réalisée par l'interconnexion de tous les éléments d'information à partager dans une structure lâchement couplée en se basant sur des objets métier communs. Ce couplage lâche (interconnexion via le partage d'information) donne une grande souplesse du système d'information global.

2. Techniques d'Intégration

Afin de construire un processus entre entreprises, chaque participant doit adapter non seulement ses processus internes (processus privés), mais également son comportement externe (processus publics). Une définition d'un processus public peut être l'exécution d'un échange formel de messages de sorte que les messages puissent être échangés avec d'autres entreprises dans un ordre prédéfini et avec des formats prédéfinis au travers de réseaux de communication. Les processus publics de deux entreprises doivent s'interfacer afin de permettre aux processus interentreprises de fonctionner. L'objectif est de montrer la possibilité de se servir des techniques existantes pour la mise en œuvre du système d'information interentreprises visé. Nous introduirons l'EDI (Echange de Données Informatisées) et le web EDI. L'EDI offre un moyen pour échanger les informations entre des entités de business séparées géographiquement. Ensuite nous présenterons les technologies d'EAI (Enterprise Application Intégration) et CORBA (Common Object Request Broker Architecture). En effet, CORBA peut servir comme un middleware dont l'objectif est de regrouper des applications réparties par l'intermédiaire d'un bus général de communication. Les détails des ces techniques sont traités dans les sections suivantes.

a. l'EDI

L'EDI peut être défini comme l'échange, d'ordinateur à ordinateur, de données concernant des transactions en utilisant des réseaux et des formats normalisés. Les informations issues du système informatique de l'émetteur transitent par l'intermédiaire de réseaux vers le système informatique du partenaire pour y être intégrées automatiquement. Ceci induit le traitement des points suivants:

- Echanger quoi ? il faut s'entendre sur ce qu'on échange et comment le modéliser.
- Echanger comment ? il faut transporter les informations via un média et des protocoles de communication donnés.
- Echanger pourquoi ? garantie sur les processus des deux côtés.

Un des défauts de l'EDI est son coût assez élevé, lié au besoin d'une infrastructure de réseau adaptée et à l'organisation de processus contraignant avec des interfaces « lourdes ». Ce défaut peut être évité avec la technologie du web EDI que nous présentons dans le paragraphe suivant.

b. Web EDI

Dans sa forme la plus simple, le Web EDI permet aux petites et moyennes entreprises de créer, gérer, recevoir et envoyer des documents électroniques en utilisant des technologies et outils du Web. Pour cela, un service dédié doit transformer de façon transparente les données de l'entreprise en format échangeable (respectant un standard EDI) et les transmettre aux partenaires commerciaux. Des formats simples sont utilisés pour permettre aux entreprises d'échanger avec leurs partenaires commerciaux. En utilisant une interface Web convenable, les transactions de type EDI peuvent être effectuées aussi facilement par le biais du courrier électronique. Le Web EDI permet également de recevoir et d'envoyer des documents métier sans recourir à un logiciel particulier. L'avantage du Web EDI est le fait qu'il est accessible partout dans le monde sans avoir besoin d'installer des logiciels métiers ou de mettre en place une infrastructure de communication particulière. Le Web EDI offre un moyen efficace pour échanger les données entre partenaires. En revanche, comme l'EDI classique, il n'a pas trouvé la reconnaissance méritée dans le domaine du business inter- organisationnel. Cela dû au fait qu'il est dédié à l'échange de données et non aux règles de décision, règles dont on a besoin pour gérer la chaîne de partenaires.

c. EAI

Les systèmes d'information d'entreprise sont composés de plusieurs "briques applicatives" souvent développées indépendamment et recourant à des technologies parfois incompatibles, ces outils ne sont que peu interopérables. Dans une logique d'intégration (ou tout au moins de couplage entre ces briques), il convient d'ajouter des systèmes intermédiaires qui facilitent la démarche d'interfaçage, voire d'intégration. C'est le rôle des outils d'EAI (Enterprise Application Intégration). L'EAI permet de résoudre le problème d'interopérabilité entre les briques applicatives constituant les applications propres à l'entreprise. Elle permet de spécifier les informations à échanger mais avec une localisation statique de ces informations. En revanche, et en se basant sur les facilités de CORBA, un middleware générique peut être mise en place permettant d'accéder aux informations quelle que soit leur localisation.

d. CORBA

CORBA (Common Object Request Broker Architecture) représente une solution d'intégration pour des applications développées indépendamment. Il s'agit d'un middleware orienté objet proposé par l'Object Management Group (OMG). CORBA

est un bus d'objets répartis qui offre un support d'exécution masquant les couches techniques d'un système réparti (système d'exploitation, processeur et réseau). CORBA prend en charge les communications entre les composants logiciels formant les applications réparties hétérogènes

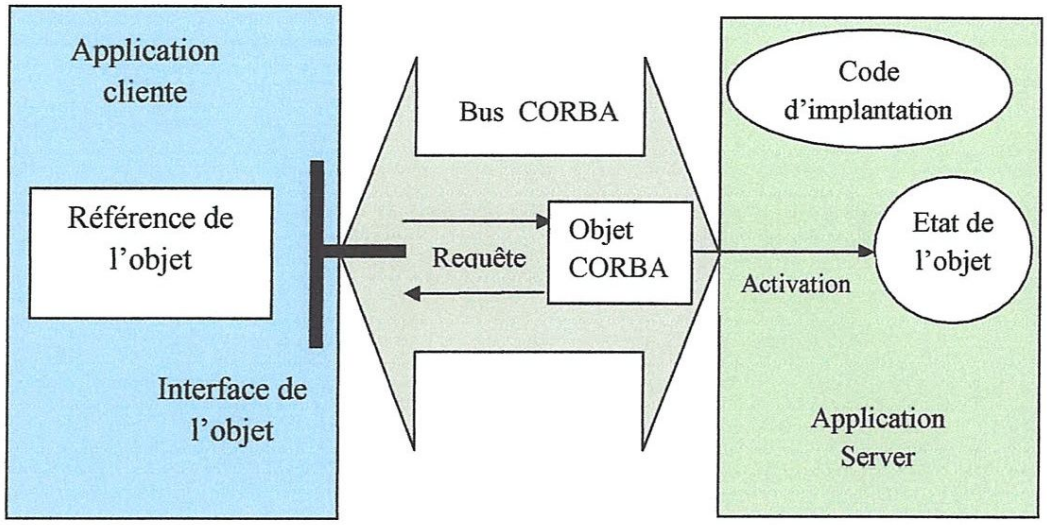


Figure 1.7 : *Notion client/serveur avec le bus de CORBA*

3. Limites des Middlewares conventionnels

On appelle middleware, littéralement "élément du milieu", l'ensemble des couches réseau et services logiciel qui permettent le dialogue entre les différents composants d'une application répartie. Ce dialogue se base sur un protocole applicatif commun, défini par l'API du middleware. Le Gartner Group définit le middleware comme une interface de communication universelle entre processus. Il représente véritablement la clef de voûte de toute application client-serveur.

L'objectif principal du middleware est d'unifier, pour les applications, l'accès et la manipulation de l'ensemble des services disponibles sur le réseau, afin de rendre l'utilisation de ces derniers presque transparente.

- Par contre les middlewares ne permettent pas, le plus souvent, l'accès à d'autres sources de données.
- Pour certaines applications devant accéder à des services hétérogènes, il est parfois nécessaire de combiner plusieurs middlewares. Dans ce cas, le poste client doit connaître et mettre en œuvre plusieurs IPC, on en vient à la notion de client lourd.
- le middleware entre client et serveur n'est pas standard.

Conclusion:

A travers ce chapitre nous avons présenté la notion de système d'information, ainsi que ses principales caractéristiques. Ceci nous a permis par la suite d'exposer le concept d'intégration et celui de l'interopérabilité en mettant en évidence la différence qui existe entre ces deux concepts. On a également exposé Les différentes architectures des systèmes d'informations en mettant en évidence les avantages et les limites de chaque architecture, puis les limites des Middlewares conventionnels. Bien vrai que les architectures ont un énorme avantage mais elles ne peuvent pas résoudre la totalité des problèmes à savoir l'architecture n-tiers propose effectivement un potentiel énorme, reste à voir comment il sera utilisé, à quel coût et, surtout, comment sera gérée la transition depuis les environnements actuels. d'où pour remédier à ces sérieux problèmes, a vu le jour une nouvelle technologie « les services Web » ce qui sera l'objet de notre prochain chapitre.

Chapitre II : Les Services Web

Introduction

Les services Web demeurent aujourd'hui un élément principal dans la mise en œuvre des applications distribuées. De telles applications sont souvent déployées à large échelle tels que les environnements de grille. Ils deviennent, de plus en plus, utilisés pour la mise en œuvre d'applications. Néanmoins, le caractère dynamique des ressources des environnements où sont déployés les services Web réduit les performances de l'exécution. En effet, à tout moment une ressource peut disparaître, apparaître ou subir une dégradation de performances, causant ainsi le ralentissement de l'exécution des applications ou même l'interruption de leur exécution. [3]

Avec l'évolution de la technologie Internet qui s'enrichit par de nouvelles prestations de services et de nouvelles applications modulaires, des organisations se chargent de décrire le contenu de ces prestations et regroupent leurs descriptifs dans des annuaires de "services Web". Chaque descriptif contient le nom du fournisseur et le nom du service, ainsi qu'une présentation du protocole d'accès au service proposé.

La définition exacte du terme "service Web" est encore fortement discutée. Les différentes définitions proposées s'accordent au moins sur l'idée qu'un service Web est un nouveau type de composant logiciel ayant la capacité de publier ses fonctions sur Internet sous forme de services, de rendre ces services facilement invocables et de les mettre à disposition par l'intermédiaire de protocoles Internet standardisés (HTTP, XML), de façon dynamique. Il faut alors distinguer les services Web de style RPC (Remote Procedure Call) et les services web de style Document.

Le style RPC utilise des appels de procédure à distance; autrement dit, le message SOAP contient une syntaxe d'appel de fonctions, avec laquelle il est également possible de réaliser une surcharge de méthodes (method overloading). Les types de données sont alors codés selon les spécifications de SOAP. Le style RPC est le style par défaut dans la plupart des implémentations (à l'exception de .NET).

Le style Document s'appuie sur l'échange de documents XML. Les types de données sont alors définis en totalité selon le schéma XML. Ce style de services Web est la variante par défaut des applications .NET.

Les modules de services Web reposent donc sur deux standards: XML (eXtensible Markup Language) et SOAP (Simple Object Access Protocol). Ces standards seront présentés dans la suite de ce chapitre.

I. Définition et Avantages des Services Web

1. Définition

Un service Web est une entité logicielle permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il vise à assurer l'interopérabilité, et ce à travers une présentation standardisée des services offertes et d'un protocole standard de communication permettant de structurer les messages échangés entre les composantes logicielles. Il offre aussi une spécification de publication et de localisation de services. La particularité des services Web réside dans le fait qu'elle utilise la technologie Internet comme infrastructure pour la communication entre les composants logiciels. Les architectures orientées service constituent un paradigme de conception et de réalisation applicable aux différents niveaux d'interaction d'un système communicant. Elles sont amenées à jouer un rôle de plus en plus important dans la conception des futurs systèmes de communication et leurs applications logicielles. Le concept du bus logiciel se généralise au niveau de couches hautes jusqu'à la couche application [8]. En d'autres termes les services Web sont des applications qui définissent un ensemble d'interfaces, s'appuyant sur XML, et qui peuvent interagir dynamiquement entre elles et avec d'autres applications grâce à l'échange de messages basés sur XML utilisant les protocoles de transport Internet disponibles

2. Les Avantages des Services Web

L'utilisation des services Web engendre plusieurs avantages dont on peut citer [12] :

– **L'interopérabilité :** C'est la capacité des services Web d'interagir avec d'autres composantes logicielles via des éléments XML et utilisant des protocoles de l'Internet.

– **La simplicité :** Les services Web réduisent la complexité des interactions entre les participants. Cela se fait en ne créant la fonctionnalité qu'une seule fois plutôt qu'en obligeant tous les fournisseurs à reproduire la même fonctionnalité à chacun des clients selon le protocole de communication supporté.

– **Une composante logicielle légèrement couplée** : L'architecture modulaire des services Web, combinée au faible couplage des interfaces associées [8], permet l'utilisation et la

réutilisation de services qui peuvent facilement être recombinés à différents autres applications.

– **L'hétérogénéité :** Les services Web permettent d'ignorer l'hétérogénéité entre les différentes applications. En effet, ils décrivent comment transmettre un message (standardisé) entre deux applications, sans imposer comment construire ce message.

– **Auto-descriptivité :** Les services Web ont la particularité d'être auto-descriptifs, c'est à dire capables de fournir des informations permettant de comprendre comment les manipuler. La capacité des services à se décrire par eux-mêmes permet d'envisager l'automatisation de l'intégration de services

3. Architecture et Fonctionnement des services Web

L'architecture des services Web est une architecture orientée composant (SOA). L'architecture SOA est un modèle qui définit un système par un ensemble de composants logiciels distribués qui fonctionnent de concert afin de réaliser une fonctionnalité globale préalablement établie. Les composants dans un système distribué n'opèrent pas dans un même environnement de traitement et sont obligés de communiquer par échanges de messages afin de solliciter des services dans le but d'accomplir le résultat souhaité.

a. Architecture

La définition de l'architecture des services Web consiste à mettre en évidence les concepts, les relations entre ces concepts ainsi qu'un ensemble de contraintes entre ces concepts. Les principaux concepts intervenant dans l'architecture des services Web sont :

– **le fournisseur du service :** désigne le serveur qui héberge les services déployés ;

– **le client du service :** représente l'application cliente qui invoque le service ;

– **le service :** désigne les fonctionnalités d'un élément logiciel qui implémente le service ;

– **la description du service :** c'est la spécification du service exprimée dans un langage de description interprétable par les machines, c'est-à-dire une description technique dans laquelle le service est vu en termes de messages, de types, de protocoles de communication et d'une adresse physique ;

– **les messages :** c'est la plus petite unité d'échange entre les clients et les services. La structure des messages qui permettent l'invocation d'un service doit être exprimée dans la description du service ;

– **la ressource :** désigne l'identifiant du service, c'est-à-dire son URI. Bien que le Web soit constitué de plates-formes totalement hétérogènes ou les intérêts des différents acteurs du marché s'entremêlent, ceci ne l'a pas empêché de se développer et d'être universel. Ce succès est du essentiellement à l'édition d'un ensemble de standards ouverts, dont les plus connus sont le protocole HTTP et le format MIME. Ensemble, ils offrent un mécanisme d'échange de données de toutes sortes quelle que soit la nature des plates-formes impliquées. Le développement des services Web a suivi la même approche en mettant en place une campagne de standardisation qui a touché les aspects les plus importants du développement d'un modèle d'intégration d'applications hétérogènes.

L'avantage de ce modèle est de présenter ces services comme des boîtes noires. En fait, les entrées-sorties d'un service sont gérées au sein de messages dont nous connaissons le format grâce à des interfaces clairement exposées mais sur les quelles l'implémentation interne du traitement n'influe pas au niveau de la structure. Ceci permet un haut niveau de modularité et d'interopérabilité.

L'avantage du modèle de message est qu'il permet de s'abstraire de l'architecture, du langage ou encore de la plate-forme qui va supporter le service : il suffit juste que le message respecte une structure donnée pour qu'il puisse être utilisé.

L'originalité de l'infrastructure des services Web consiste à mettre en place ces services exclusivement sur la base des protocoles les plus répandus sur Internet. Ces protocoles sont répartis selon quatre axes représentés dans la figure :

– **Couche de transport :** cette couche s'occupe de transporter les messages entre applications en utilisant les protocoles HTTP, FTP ou SMTP;

– **Message XML :** il s'agit de formaliser les messages à l'aide d'un vocabulaire XML commun (SOAP) ;

– **Description des services :** description de l'interface publique des services Web (WSDL) ;

– **Recherche de services :** centralisation des services et de leur description dans un référentiel commun (UDDI).

Ces standards sont exposés dans la figure 2.1

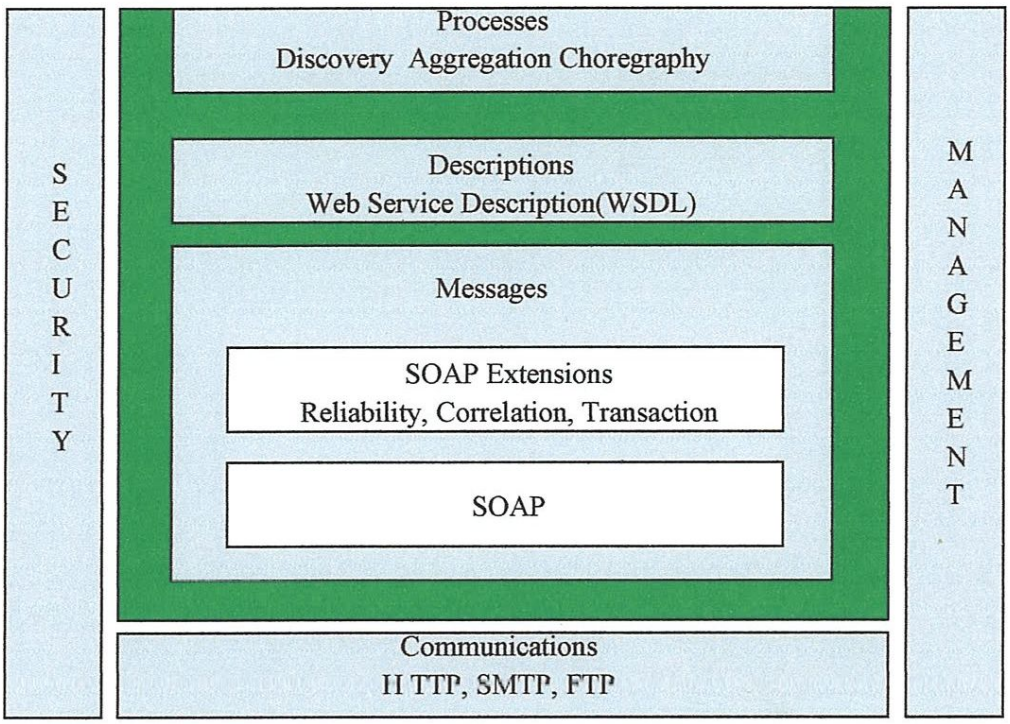


***Figure 2. 1** : Architecture des services Web*

b. Fonctionnement des Services Web :

Le cycle de vie d'un service Web se déroule de la façon suivante : une fois créé, le service est déployé sur le réseau (local ou Internet). Puis un utilisateur ayant des besoins spécifiques va rechercher un service correspondant à ses besoins à l'aide d'un annuaire spécialisé(UDDI). Enfin, une fois le service trouvé, l'utilisateur va invoquer le service : une communication va être mise en place entre l'utilisateur et le service Web. Ce cycle de vie, représenté dans la figure 2.2, fait appel à trois grandes technologies : SOAP, WSDL, UDDI

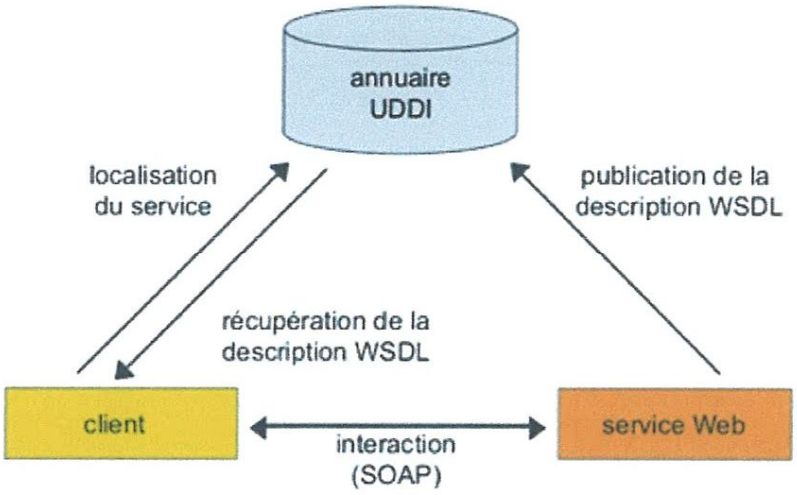


Figure 2. 2 : *Déploiement, découverte et invocation de services Web*

4. Les standards des services web

La description fonctionnelle des services Web montre l'utilisation de nombreux langages et protocoles durant le déploiement et l'invocation du service web. XML, SOAP, WSDL et UDDI sont les principaux langages et protocoles que cette section va décrire.

a. XML

XML (Extensible Markup Language, ou Langage Extensible de Balisage) est le langage destiné à succéder à HTML. Comme HTML (Hypertext Markup Language) c'est un langage de balisage (markup) : il représente de l'information encadrée par des balises. XML est un métalangage, ce qui veut dire que contrairement à HTML qui possède un ensemble de balises de présentation prédéfinies, il va permettre d'inventer de nouvelles balises d'isolement d'informations ou d'agrégats élémentaires que peut contenir une page Web.

Exemple

```
< ?xml version='1.0' encoding='ISO-8859-1' ?>
  <biblio subject='xml'>
    <livre isbn='9742212030811' lang='fr' subject='applications'>
      <auteur>
        <prenom>victor</prenom>
        <nom>hugo</nom>
      </auteur>
      <titre>les miserables</titre>
        <editeur>
          <nom>dupond</nom>
          <lieu>paris</lieu>
        </editeur>
      <datepub>1999</datepub> </livre>
    <livre isbn='3782242090420' lang='fr' subject='general'>
```

```
        <auteur>
            <prenom>frederic</prenom>
            <nom>beigbeider</nom>
        </auteur>
        <titre>windows on the world</titre>
            <editeur>
            <nom>fayard</nom>
            <lieu>paris</lieu>
        </editeur>
      <datepub>2004</datepub>
  </livre>
</biblio>
```

b. WSDL

Le WSDL (Web Services Description Language) est un langage basé sur XML, crée dans le but de fournir une description unifiée des services Web. Il se présente comme un standard actuel dans ce domaine, et il est, de plus, normalisé par le W3C. Il est issu d'une fusion des langages de descriptions NASSL (Network Accessibility Service Specification Language - IBM) et SCL (Service Conversation Langage - Microsoft). Son objectif principal est de séparer la description abstraite du service de son implémentation même. [12]

i. La structure d'un document WSDL

Un fichier WSDL contient une description de tout ce qui est nécessaire à l'appel d'un service Web SOAP et elle est faite sur deux niveaux, un niveau abstrait et un niveau concret.

Au niveau abstrait, la description du service web consiste à définir les éléments de l'interface du service Web tels que les types de données (**Data Types**), les messages (**Message**), les types de ports (**Port Type**). Ces parties décrivent des informations abstraites indépendantes au contexte de mise en œuvre. On y trouve les types de données envoyées et reçues, les opérations utilisables et le protocole qui sera utilisé.

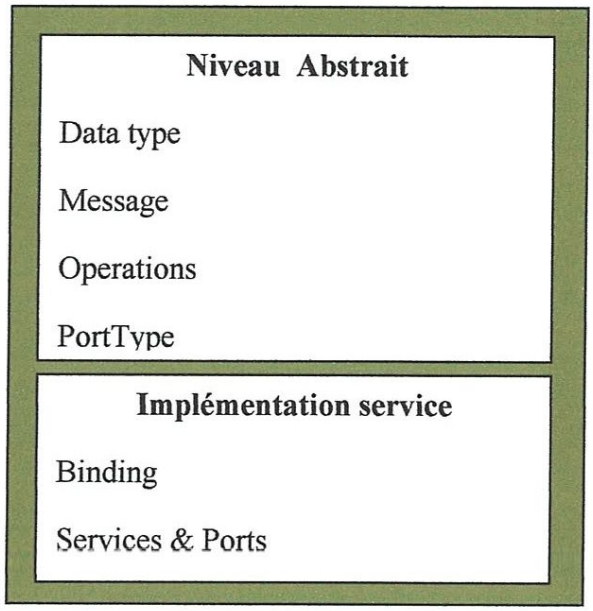


Figure 2. 3 : *Structure d'une description WSDL*

- **Data types :**

Est l'élément qui définit les types de données utilisées dans les messages échangés par le service Web.

➢ **Types (exemple)**

```
<types>
    <schema
    targetNamespace="http://example.com/stockquote.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
        <element name="TradePriceRequest">
            <complexType>
             <all>
             <element name="tickerSymbol" type="string"/>
            </all>
             </complexType>
         </element>
        <element name="TradePrice">
        </element>
    </schema>
</types>
```

- **Message :**

spécifie les types d'opérations supportées par le service Web, il permet d'incorporer une séquence de messages corrélés sans avoir à spécifier les caractéristiques du flux de données.

➢ **Messages (exemple)**

```
<message name="GetLastTradePriceInput">
    <part n a m e ="body" element="xsd1:TradePriceRequest"/>
</message>
<message name="GetLastTradePriceOutput">
    <part n a m e ="body" element="xsd1:TradePrice"/>
</message>
```

- **Port Type :**

est un groupement logique ou une collection d'opérations supportées par un ou plusieurs protocoles de transport, il est analogique à une définition d'un objet contenant un ensemble de méthodes

Au niveau concret, le service Web est défini grâce aux éléments **Bindings** et **Service & Port.** Ces deux derniers décrivent des informations liées à un usage contextuel du service Web. On y trouve l'adresse du fournisseur implémentant le service, et le service qui est représenté par les adresses des fournisseurs.

➢ **PortType (exemple)**

```
<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
        <input message="tns:GetLastTradePriceInput"/>
        <output message="tns:GetLastTradePriceOutput"/>
    </operation>
</portType>
```

- **Bindings :**

Décrit la façon dont un type de port est mis en œuvre pour un protocole particulier (HTTP par exemple), et un mode d'invocation (SOAP par exemple). Cette description est faite par un ensemble donné d'opérations abstraites. Pour un type de port, on peut avoir plusieurs liaisons, pour différencier le mode d'invocation ou de transport de différentes opérations.

➢ **Binding (exemple)**

```
<binding name="StockQuoteSoapBinding"
type="tns:StockQuotePortType">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="GetLastTradePrice">
<soap:operation
soapAction="http://example.com/GetLastTradePrice"/>
<input>

<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
</binding>
```

• **Port :**

Spécifie une adresse URL qui correspond à l'implémentation du service Web par un fournisseur et identifie unc ou plusieurs liaisons aux protocoles de transport pour un Port Type donné

• **Service :**

Spécifie l'adresse complète du service Web, et permet à un point d'accès d'une application distante de choisir à exposer de multiples catégories d'opérations pour divers types d'interactions.

➢ **Service (exemple)**

```
<service name="StockQuoteService">
<documentation>My first service</documentation>
<port name="StockQuotePort" binding="tns:StockQuoteBinding">
<soap:address location="http://example.com/stockquote"/>
</port>
</service>
```

Un WSDL décrit un Service Web avec ces principales balises:La structure principale d'un WSDL ressemble à cela (liste non exhaustive):

```
<definitions>
    <types>
definition des types.....
    </types>
    <message>
definition des messages ....
    </message>
    <portType>
definition des ports (opérations et messages).....
    </portType>
    <binding
Les protocoles de communication utilisés par le web-service
    </binding>
    <service>
Une adresse physique où accéder au service.
    </service>
</definitions>
```

Le document W3DL est indispensable au déploiement de services Web et décrit deux documents essentiels : un document pour l'interface du service Web et un autre pour son implémentation. La publication et la recherche d'un service Web au sein de l'annuaire UDDI se font via ces deux types de documents WSDL.

c. SOAP (Simple Object Access Protocol)

SOAP est un protocole de communication basé sur XML pour permettre aux applications de s'échanger des informations via HTTP. Il permet ainsi l'accès aux services Web et l'interopérabilité des applications à travers le Web. SOAP est un protocole simple et léger et qui repose entièrement sur des standards établis comme le HTTP et XML. Il est portable et donc indépendant de tous système d'exploitation et du type d'ordinateur. SOAP est une spécification non propriétaire.

i. Structure d'un message SOAP

SOAP définit un format pour l'envoi des messages. Les messages SOAP sont structuré en un document XML et comporte deux éléments obligatoires : Une enveloppe et un corps (une entête facultative).

Le protocole SOAP est une surcouche de la couche application du modèle OSI des réseaux. Le protocole applicatif le plus utilisé pour transmettre les messages SOAP est HTTP, mais il est également possible d'utiliser les protocoles SMTP ou FTP, la norme n'impose pas de choix.

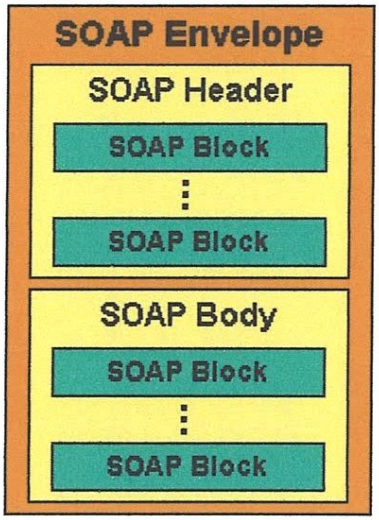


Figure 2. 4 : *Structure de base des messages SOAP*

ii. Principe de fonctionnement de SOAP

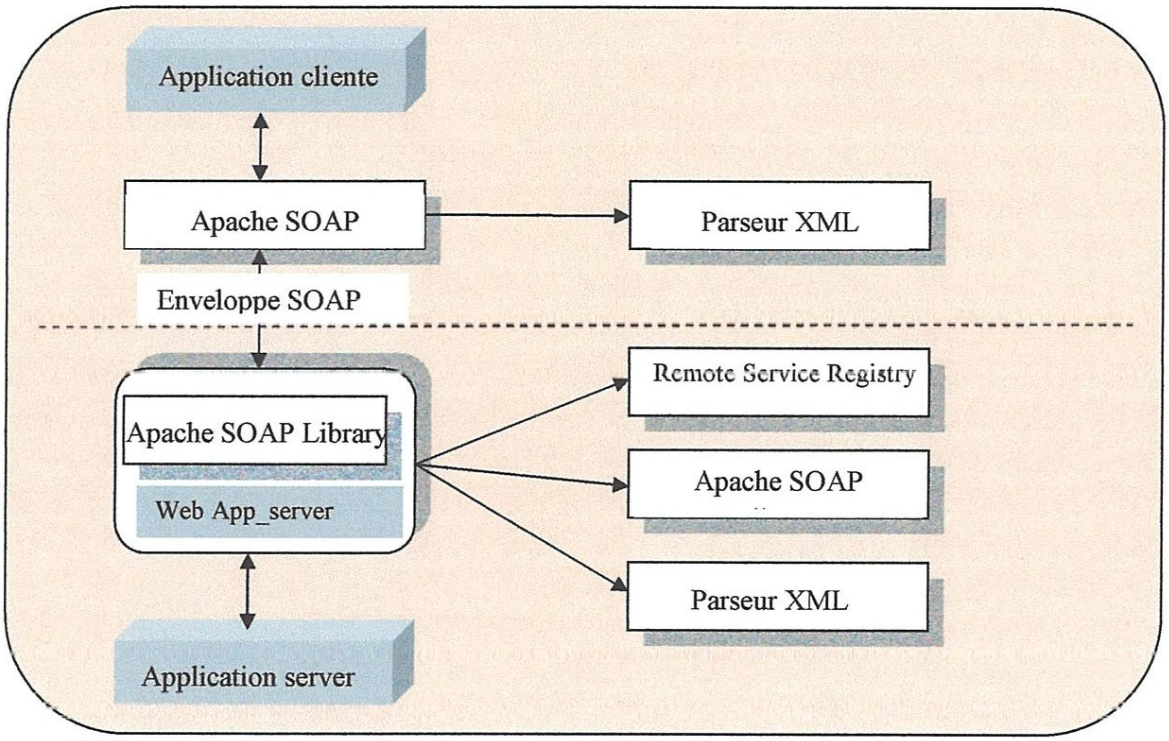


Figure 2. 5 : *Principe de fonctionnement de SOAP [12]*

❖ **Coté client**

Le client envoie des messages au serveur correspondant à des requêtes SOAP-XML enveloppés dans des requêtes HTTP. De même, les réponses du serveur sont des réponses HTTP qui renferment des réponses SOAP-XML. Dans le process client, l'appel de service est

converti en une requête SOAP qui est ensuite enveloppé dans une HTTP.

❖ **Coté serveur**

C'est légèrement plus complexe car il requiert un process listener correspondant au process serveur en attente de connexion cliente. Le listener est souvent implémenté au travers d'un servlet qui s'exécute qui a pour tache d'extraire le message XML-SOAP de la requête HTTP, de le désérialiser c'est à dire de séparer le nom de la méthode et les paramètres fournis puis invoquer la méthode du service en conséquence. Le résultat de la méthode est alors sérialisé, encodé HTTP et renvoyé au demandeur.

♦ **Exemple de requête SOAP/HTTP:**

```
Poste/ stockQuote http/1.1
Host: www.stockquoteserver.com
Content-type:text/xml;charset="utf-8"
Content-length:nnnn
SAOPAction:"URI"
<soapenv:Envelope
Xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/">
        <Soapenv:Body>
                <m: GETLastTradePrice xmlns:m="Some-URI">
                        <m: tickerSymbol>DIS</m: tickerSymbol>
                </m: GETLastTradePrice>
        </Soapenv:Body>
 </ soapenv:Envelope>
```

♦ **Exemple de réponse SOAP/HTTP**

```
HTTP/1.1 200  OK
 Content-Type: text/xml; charset="utf-8"
  Content-Length:  nnnn
 <soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
          <m:GetLastTradePriceResponse  xmlns:m="Some-URI">
              <m:price>34.5</m:price>
          </m:GetLastTradePriceResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

d. UDDI (Universal Description, Discovery and Integration)

UDDI est une norme d'annuaire de services Web appelée via le protocole SOAP. Pour publier un nouveau service Web, il faut générer un document appelé *Business Registry*, il

sera enregistré sur un UDDI *Business Registry Node*. Le *Business Registry* comprend 3 parties:

– **Pages blanches :** noms, adresses, contacts, identifiants des entreprises enregistrées

– **Pages jaunes :** informations permettant de classer les entreprises, notamment l'activité, la localisation, etc.

– **Pages vertes :** informations techniques sur les services proposés.

Le protocole d'utilisation de l'UDDI contient trois fonctions de base :

– ***publish*** pour enregistrer un nouveau service,

– ***find*** pour interroger l'annuaire,

– ***bind*** pour effectuer la connexion entre l'application cliente et le service.

Comme pour la certification, il est possible de constituer des annuaires UDDI privés, dont l'usage sera limité à l'intérieur de l'entreprise.

5. Techniques de Composition des services Web

Nous pouvons classer ces techniques en deux grandes familles : les techniques de composition dites « **statiques** », i.e., qui sont définies à l'aide de processus métiers (orchestration et chorégraphie) ; et les techniques de composition dites « **dynamiques** », lorsque la composition de services web tient compte des services disponibles, de leurs fonctionnalités et du but à atteindre que ce soit avant ou pendant l'exécution des services web

a. Les différentes approches de la composition statique

La composition des services Web peut se faire de deux manières : orchestration et chorégraphie

i. Orchestration

L'orchestration décrit comment les services Web peuvent interagir entre eux selon une perspective opérationnelle, avec des structures de contrôle, incluant l'ordre d'exécution des interactions (souvent qualifiée de "logique métier"). L'identité des services Web est alors connue. L'orchestration donne une vision concrète qui permet l'expression d'un processus exécutable.

Les services Web n'ont pas de connaissance (et n'ont pas besoin de l'avoir) d'être mêlées dans une composition et d'être partie d'un processus métier. Seulement le coordinateur de l'orchestration a besoin de cette connaissance.

Un coordinateur prend le control de tous les services Web impliqués et coordonne l'exécution des différentes opérations des services web qui participent dans le processus.

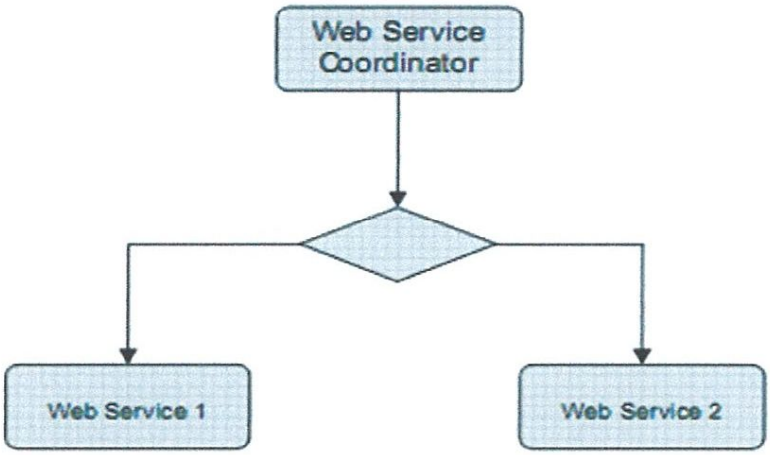


Figure 2. 6 : *Orchestration de services Web*

ii. Chorégraphie

La chorégraphie est, par nature, collaborative. Elle décrit les différents messages qui transitent entre les différents acteurs d'un processus (les services), l'identité de ceux-ci n'étant pas forcément encore connue. La chorégraphie permet la collaboration point-à-point entre plusieurs services Web et donne ainsi une vision abstraite des échanges au sein d'un processus. Elle ne permet en aucun cas une exécution, mais sert cependant de première spécification au processus concret (orchestration) à réaliser. Elle permet la modélisation d'un point de vue global afin de prendre en compte des situations de concurrences dans les environnements distribués et ainsi donner une vue plus flexible.

Contrairement à l'orchestration, la chorégraphie n'a pas de coordinateur central. Chaque service Web mêlé dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu. La collaboration dans la chorégraphie des services Web peut être représentée de la manière suivante

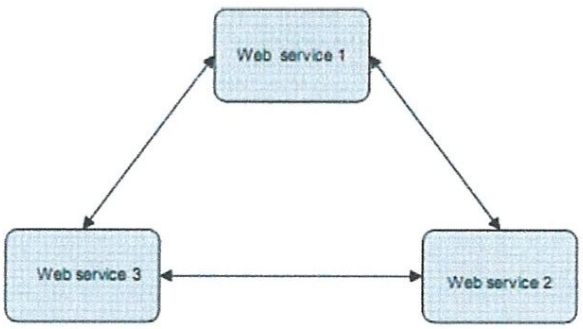


Figure 2. 7 : *Chorégraphie de services web*

Aussi il décrit les relations entre services composites mais contrairement à BPEL qui centralise le contrôle (orchestration) WS-Choreography s'intéresse à la description des messages inter changés entre les participants. Dans ce cas, le contrôle est distribué (chorégraphie).

b. **Les différentes approches de la composition dynamique**

On appelle composition dynamique l'agrégation de services Web permettant de résoudre un objectif précis soumis par un utilisateur en prenant en compte ses préférences. Cette composition peut se faire avant ou pendant l'exécution des services Web. Le service résultant de cette composition est appelé service composite. Il n'existe pas de standard : les recherches sont principalement académiques.

La composition de services Web est une tâche complexe. Cette complexité est due principalement aux raisons suivantes: le nombre de services disponibles sur le Web est potentiellement très important et en constante augmentation ; les services Web peuvent être crées et modifiés à la volée, le système de composition doit donc détecter et gérer ces changements ; les services Web sont crées par des organisations différentes qui n'ont pas forcément les mêmes modèles de concepts pour d'écrire ces services. En effet, deux services qui effectuent la même action peuvent être décrits d'une façon très différente.

Les approches proposées dans la littérature sont diverses. Aucune n'est reconnue comme une norme. La classification de ces approches nous permet de dégager trois grandes familles

- **L'approche industrielle** : propose des langages de formalisation basés sur XML comme BPEL4WS.
- **L'approche sémantique** : se base essentiellement sur l'ontologie OWL-S pour décrire les services composés.
- **L'approche formelle** : utilise des techniques de modélisation de validation formelle de processus (exemple : les réseaux de Pétri).

6. Les langages de composition des services web

Les services Web uniquement ne satisfait pas aux besoins des utilisateurs qui sont de plus en plus complexes. Pour fournir une solution à une tâche complexe, on peut combiner des services Web pour n'en former qu'un seul , on parle alors de composition de services Web. Ainsi il existe plusieurs langages qui permettent de décrire les services Web à savoir .

a. BPEL

Initialement connu sous le nom de *BPEL4WS*, renommé par la suite *WS-BPEL*, cette spécification est plus connu sous le nom de BPEL [4]. Il s'agit d'un langage d'exécution des processus métiers qui permet la composition d'un ensemble de services Web, et spécifie les règles de dialogue entre eux. Il définit un protocole d'interaction d'un seul service Web tout en spécifiant l'ordre d'invocation de ses opérations. Le fichier BPEL agit donc sur des éléments comme la transformation de données, l'envoi de messages ou l'appel d'une opération **[8].** BPEL est standardisé par de nombreux éditeurs de logiciel comme Adobe, BEA Systems, HP, IBM, Oracle, JBoss, Sun, Tibco, Webmethods, Microsoft.

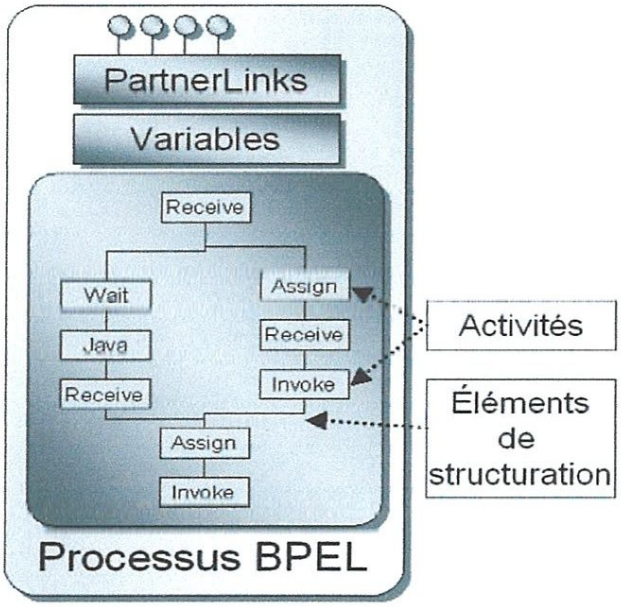


Figure 2. 8 : *Les éléments de définition des processus BPEL*

b. Ebxml : (Electronic bisness XML)

EbXML définit un ensemble de spécifications pour permettre des interactions B2B entre des organisations. La partie basique de l'infrastructure ebXML est le dépôt (ou repository). Il sauvegarde des informations d'entreprise importantes avec les produits et services qu'elles offrent. Le dernier permet l'utilisation de protocoles communs tels que SMTP, HTTP et FTP. A la couche de processus métier, ebXML définie un schéma de la spécification du processus métier disponible en UML ou XML. Elle fournie un ensemble de spécifications de processus métier qui sont partagées entre plusieurs industries en vue de construire des processus métier personnalisés.

c. Xlang

C'est un format proposé par Microsoft, en 2001, pour représenter en XML l'orchestration

d'activités qui constituent un processus métiers. Xlang est le format intermédiaire de stockage de l'environnement BizTalk server. Ces documents Xlang, ou encore appelé « schedules »: sont exécutés par le serveur BizTalk en phase de production. Xlang s'appuie sur WSDL en réutilisant un certain nombre de ses concepts

d. WSFL: Web Service Flow Language

Proposé par IBM, le WFSL est un langage de description de combinaison des Web Services de façon utile. Deux façons de combiner les services sont définies :

- **Des modèles de circuit (Flow Model)**, où un circuit représente une séquence à exécuter pour qu'un processus métiers puisse être réalisé ;
- **Des modèles globaux (Global Model),** où la perspective est celle des régularités dans l'interaction entre les Services Web dans le contexte de collections de services typiques. Ces interactions sont réglées par des liens entre des services, avec un lien pour chaque opération possible entre deux services dans une collection. Ce réglage peut être adapté soit à des interactions dans des hiérarchies, soit à des interactions collaboratives entre pairs.

e. Web Service Choreography Interface (WSCI)

Web Service Choregraphy Interface (WSCI), co-développé par BEA, Intalio, SAP et Sun. WSCI décrit comment les opérations des services Web (tel que ceux définissent par WSDL) peuvent être chorégraphiés dans le contexte d'échange de message dans lequel les services Web y participent.

f. Web Service Choreography Description Language (WS-CDL)

En 2004, la convergence de WSCI 24 et le WSCL 25 a donné lieu au Web Services Choreography Description Language (WS-CDL). Est un langage de description de la chorégraphie de services Web qui se base sur le langage XML. Ce dernier, décrit la collaboration des services Web à travers les entreprises participantes.

Conclusion

Ce chapitre a été consacré aux services Web, en décrivant leurs concepts de base, à savoir l'intégration et l'interaction. On a exposé l'architecture, quelques technologies et les standards permettant l'interaction entre partenaires.

Notre intérêt va s'orienter vers les processus métiers qui représentent la composante clé de la couche externe de l'échange inter-entreprise. Ainsi, le chapitre suivant va porter sur les processus métiers et plus précisément leur collaboration qui représente un défi dans la collaboration intra et interentreprises.

Chapitre III : Processus Métiers : Spécification et Modélisation

Introduction

Aujourd'hui plus que jamais, les organisations de toute taille se développent dans le monde entier: fournisseurs dans un pays, clients dans un autre. C'est la raison pour laquelle, il est devenu vital de faire collaborer et d'intégrer tous ces partenaires dans un environnement métier. **[14]** De l'informatisation à l'administration électronique, le besoin d'échange de données entre les applications hétérogènes des systèmes d'information est exponentiel. Le procédé technique existe, et il est fiable. Par contre les exigences des clients sont si multiples et les intervenants dans la chaîne (chaîne d'entreprises) si nombreux, que l'entreprise se trouve en situation d'interdépendance très forte avec ses partenaires. C'est la raison pour laquelle l'enjeu futur ne concerne plus la compétitivité au sein de l'entreprise mais dans les chaînes d'entreprises. **[15]** Le problème vient surtout de la difficulté des organisations à se mettre d'accord sur un langage commun. Nous avons l'habitude d'analyser les questions techniques également sous l'angle « management », « fonctionnel » et « métier ». Le changement auquel les entreprises font face depuis quelques années est lié à l'informatique et à l'automatisation qui en découle. Dorénavant, l'automatisation et l'informatisation lient l'entreprise aux contraintes et à l'inertie des systèmes informatiques, et c'est ce qui justifie le déploiement d'une nouvelle forme d'ingénierie dans les organisations, l'ingénierie des **processus métier**. **[2].** Ce chapitre va donc porter en premier lieu sur la définition des processus métiers et les différents langages les décrivant ensuite il décrira, les différentes solutions et propositions faites par des chercheurs et industriel pour permettre une collaboration métier efficace et ce en proposant des modèles universels de représentation de ces processus métiers.

I. Definition d'un Business Process (B.P)

Définition 1 : Un Business Process ou Processus Métier ou encore processus affaire est un ensemble de règles métiers (business), de définitions d'activités et d'événements déclencheurs qui fournissent un contexte d'échange d'information **[16]**

Définition 2 : Un Processus Métier est une unité persistante d'un travail pouvant avoir un nombre important de transactions. Il est déclenché par un événement métier tel que l'invocation, requête pour proposition ou une requête pour un transfert de fonds. Le processus est conduit par des règles métiers qui déclenchent les tâches et les sous-processus, avec chaque transition d'état exécutée dans une transaction et auditée pour des raisons métiers. Les tâches et les sous-processus sont assignés à des ressources qui sont des unités organisationnelles capables et autorisés à jouer des rôles spécifiques dans le processus. La

définition des règles, des tâches, des sous-processus, et des stratégies de ressources constitue une description de processus. L'exécution d'un processus métier consiste en l'invocation des services métiers existants, qui peuvent résider n'importe où.

Définition 3 : Un Processus Métier est un ensemble d'activités conçues pour produire des outputs spécifiques pour un marché ou un client particulier. Il peut être défini sur la base de trois dimensions :

Entités : les processus prennent place entre les entités organisationnelles. Il peut être inter-organisationnels, inter-fonctionnel ou encore entre personnes.

Objets : les processus sont le résultat d'une manipulation d'objet. Ces objets peuvent être physiques ou informationnels. Notifier au client (figure 3.1) est un exemple d'objet.

Activités : les processus peuvent impliquer deux types d'activités: les activités opérationnelles et les activités de gestion.

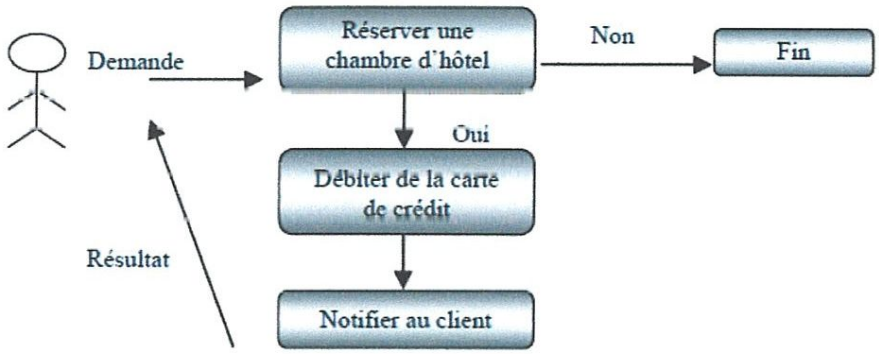


Figure 3. 1 : *Exemple d'un processus métier.*

1. Processus métier et Système d'Information(S.I)

Les processus métier permettent d'organiser et de répartir les tâches entre les différents acteurs d'une entreprise. Ceci contribue à la réalisation des objectifs stratégiques de celle-ci. Ce processus métier obéit également à des règles et procédures représentant le point de vue organisationnel de celui-ci. Un processus métier est mis en œuvre à l'aide du SI de l'entreprise. Ainsi, un processus métier peut être décomposé en un ou plusieurs processus SI **[17]**

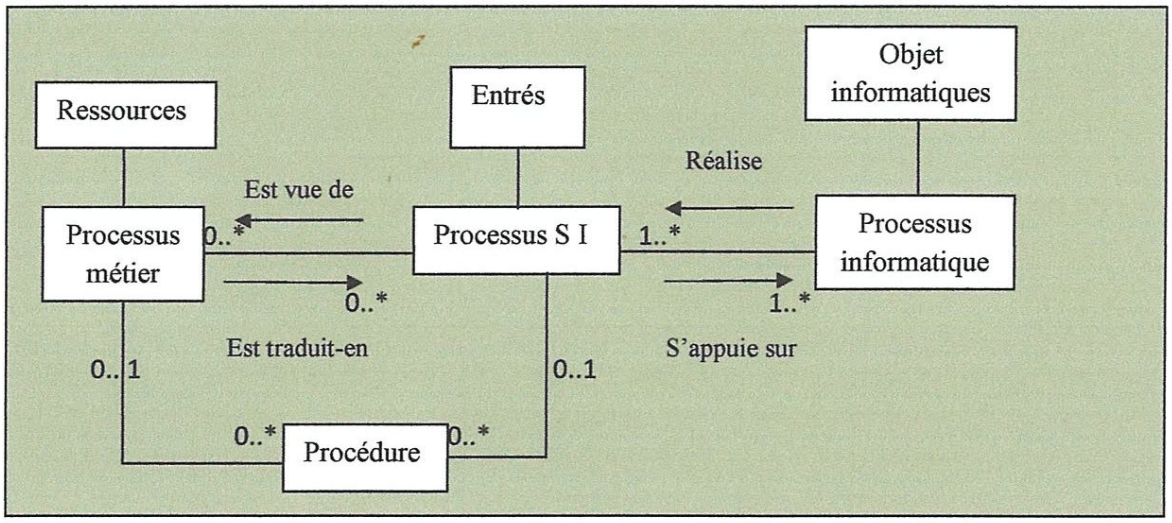


Figure 3. 2 : *Processus métier, processus SI, processus informatique*[?]

Ces processus SI représentant tout ou partie du processus métier participent aux mêmes objectifs mais se focalisent sur la mise à disposition et le traitement de l'information. Ces processus SI sont mis en œuvre par des processus informatiques, soit un ensemble d'activités logicielles, exécutées par des machines et dialoguant éventuellement avec des humains. L'articulation de ces différents concepts est représentée Figure ci dessus.

2. Le cycle de vie d'un Processus Métier

Selon « Tanguy Crusson » le cycle de vie d'un processus métier est présenté sur quatre étapes. Ces étapes sont illustrées dans la figure 3.3 ci-dessous.

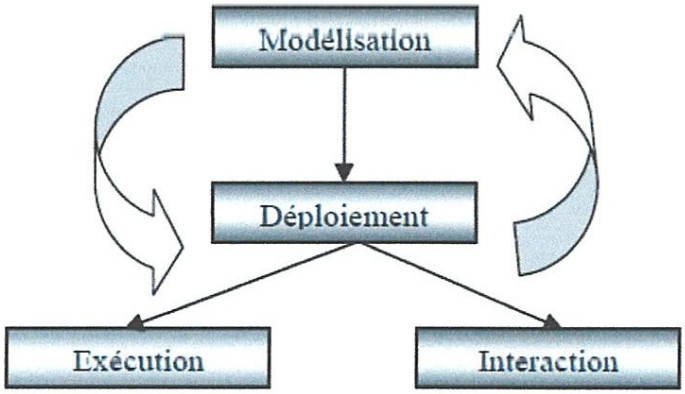


Figure 3. 3 : *Cycle de vie d'un Processus Métier.*[?]

a. Modélisation du processus métier

Un modèle de processus métier consiste en un ensemble de modèles d'activités et de contraintes d'exécution entre eux. Elle représente la phase où l'importation des définitions des processus métiers dans les outils techniques est possible, afin d'ajouter le lien avec les applications du SI. Les capacités de modélisation des outils devraient permettre de déployer directement les processus modélisés sans passer par une phase d'implémentation.

b. Déploiement du processus métier

Consiste à déployer le code réalisé durant la phase de modélisation

c. Exécution du processus métier

Consiste en l'exécution des applications existantes de l'entreprise.

d. Interaction du processus métier

Consiste en l'interaction des utilisateurs avec ce processus. Une fois le processus déployé, la phase d'interaction permet de l'optimiser. Cette phase permet d'identifier les modifications à effectuer. Pour les réaliser, on entre alors dans une autre phase de modélisation, et ce cycle se répète. Généralement, après trois itérations, il est plus simple de redévelopper l'application hébergeant le processus que d'en modifier l'implémentation

3. Implémentation de processus métier

Implémenter un processus métier, c'est utiliser les ressources logicielles et matérielles adéquates pour exécuter ses activités. Les activités sont implémentées comme des processus matériels et/ou électroniques. Les processus matériels incluent la production, la transformation, la mesure et l'assemblage d'objets physiques. Un exemple de processus matériel est l'assemblage de voitures. **[13]**

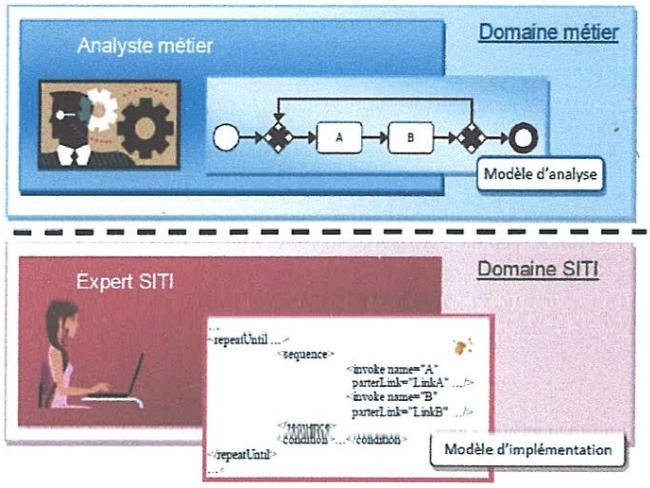


Figure 3. 4 : *Domaines, acteurs, modèles*

4. Les concepts de bases pour la description des Processus Métiers

La description des processus métiers reposent sur deux concepts à savoir :

XML et les services Web.

- **XML** (eXtensible Markup Language) : modèle de données semi-structuré. Il se base sur HTML et le SGML. Il représente un format d'échange universel.
- **Services Web**: sont décrits grâce à des fichiers WSDL (Web Service Definition Language) enregistrés dans un répertoire UDDI (Universal Description, Discovery, and Intégration) et communiquent grâce au protocole SOAP.
  - **L'orchestration et la chorégraphie des Services Web:**

Décrivent deux aspects de création de processus métiers à partir des services Web composites. L'orchestration et la chorégraphie sont utilisées dans les processus métiers et les systèmes de gestion de processus métiers

5. La collaboration des Processus Métiers

Une collaboration entre les processus métiers requiert un mécanisme qui permet aux entreprises participantes d'approuver la description du processus métier commun et des données à être échangées durant l'exécution du processus. La collaboration entre entreprises au niveau processus métier se définit en trois étapes :

1ère étape : Définition d'un processus métier collaboratif approuvé par tous les participants.

- Un processus collaboratif comporte une liste de rôle de processus qui indique la logique des participants.

- Un nœud de travail qui représente une tache associée à une activité, i-e une partie du travail qui contribue à l'accomplissement du processus, doit correspondre à un rôle dans le processus. Une activité a également un rôle appelé rôle de l'activité. Cette notion de rôle d'activité existe dans les spécifications des processus métiers.

2ième étape : Identification des processus métiers publics dans le processus collaboratif au niveau de chaque participant.

3ième étape : Mise en correspondance des processus publics avec les processus privés de chaque participant.

Différentes topologies de la collaboration métier ont été présentées:

- Collaboration Horizontale : une collaboration de nature pair-to-pair (P2P), qui est représentée par la formation d'une coopération dynamique de réseaux métiers dans lesquels les partenaires implémentent les fonctions centrales des affaires.

- *Collaboration Verticale* : se base sur le paradigme d'externalisation, des processus métiers, en délégant des activités métiers secondaires à des fournisseurs de services qui sont spécialistes.

6. Technologies de processus métier

Construire des processus métier inter-organisationnels requière une intégration à des niveaux différents, et plus spécialement aux niveaux communication et processus métier. La couche communication est concernée par l'échange de messages entre partenaires. L'objectif d'intégration de cette couche est de permettre des interactions transparentes entre partenaires. Celle-ci, étant une tâche difficile, peut être assurée en utilisant les API (Application Programming Interface) et les workflows. Ceci représente quelques technologies qui implémentent et gèrent les processus métier (technique ad hoc, Workflows, Services Web,...etc). 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 [18]

a. Techniques 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. Ces technologies se basent principalement sur certains middlewares invoqués de manière programmatique pour définir des passerelles entre les applications [19], 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. [18] (voir figure 3.5)

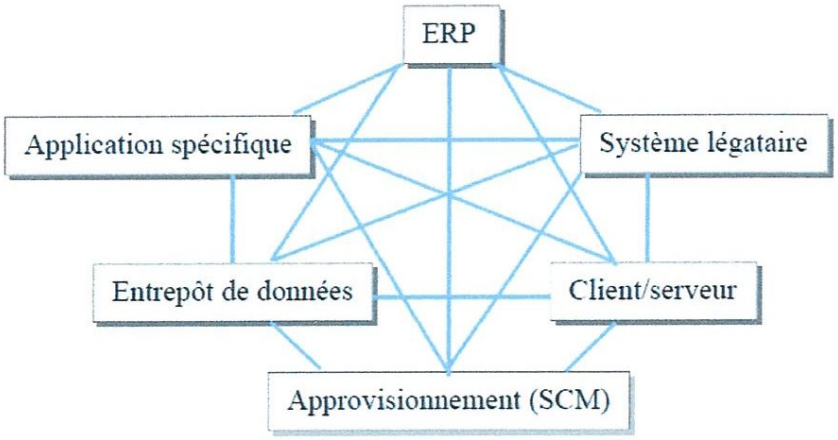


Figure 3. 5 : *Principe de technique ad hoc [8].*

L'analyse de cette technologie nous a permis de résumer les avantages et les inconvénients dans le tableau suivant :

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

Tableau 3. 1 : *Avantages et inconvénients de technologies d'intégration.*

b. Workflows

Un workflow est l'automatisation d'un processus métier, en tout ou en partie, durant lequel des documents, informations, ou tâches sont passés d'un participant à un autre pour action, d'après un ensemble de règles procédurales. Ayant la définition de workflow, il y a un grand nombre d'activités métier dans une organisation qui se classifient dans la catégorie de workflow. Elles incluent un ordre d'achat, une demande de crédit, une location de voitures, …etc. Un outil de gestion de workflow permet de modéliser et d'automatiser les flux d'informations dans l'entreprise. Par exemple, l'outil de workflow permet de préciser les circuits de cheminement de documents en identifiant les intervenants concernés, les actions à réaliser et les délais [20]. La Figure 3.6 montre en effet que le cycle de vie d'un processus selon BPM englobe les mêmes phases que celui de la gestion du workflow : [17]

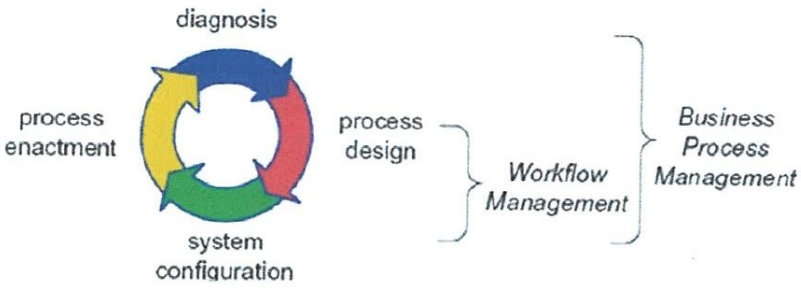


Figure 3. 6 : *Gestion Workflow et BPM [17].*

c. BPMN

Le BPMI issue d'un regroupement d'entreprises, a développé le standard Business Process Modeling Notation (BPMN) Ce dernier a pour objectif primaire de fournir une notation compréhensible par tous les utilisateurs métiers; des analystes métiers qui créent des drafts initiaux de processus, aux développeurs techniques responsables de l'implémentation de la technologie qui exécute ces processus, et finalement, aux gestionnaires et ceux qui surveillent ces processus. Il crée un lien standardisé entre la conception et l'implémentation des processus. [16]

7. Les protocoles métiers : Business Protocols

Un protocole métier constitue l'aspect formel et apparant d'un BP. Ils Constituent une partie importante de la description de services Web. D'abord, il assure la spécification de la conversation supportée par les services. En outre, il permet une interaction efficace entre les clients et les services. Enfin, il garanti la simplification, le développement, le déploiement, la gestion et l'exécution du service. Les « business protocols » définissent donc le comportement externe des services. Si deux services ont besoin d'interopérer, leurs protocoles doivent être compatibles, équivalents ou remplaçables Les protocoles ont pour objectifs la simplification, le développement et la gestion du cycle de vie de service. Pour cela, ils ont besoin de langages et de modèles formels, des opérateurs de gestion de protocoles.

a. Caractéristiques de « business protocols »

Les protocoles décrivent les comportements externes de services. Ils sont très importants pour les développeurs pour créer des clients pouvant interagir correctement avec les services. Les spécifications de protocoles peuvent simplifier considérablement la gestion de cycle de vie des services. Exemple, durant le développement des services Web, les protocoles des clients et des fournisseurs peuvent être analysés pour identifier quels sont les conversations

qui peuvent être établies entre deux services Le but est de diminuer le taux d'erreurs et de déterminer les modifications possibles pour améliorer la compatibilité des services. L'analyse et la gestion de protocoles permettent aussi la gestion automatique des exceptions, la vérification de conformité, la correspondance statique ou dynamique, tout en prenant l'avantage de la description des protocoles [24].

8. Techniques de modélisation des processus métier

En utilisant les langages de modélisation des processus métier, les concepteurs ne sont pas à l'abri de commettre des erreurs de spécification. Pour cela, les concepteurs ont souvent recours aux modèles formels. Ces modèles exigent de formaliser de processus en utilisant un modèle donné (par exemple réseau de Pétri, MCT, OWLS, Automates…etc.).

a. Les réseaux de pétri

Les Réseaux de Pétri (RdP) ont été introduits par Carl Adam Pétri, en 1962, afin de modéliser et d'analyser les comportements des systèmes d'une manière graphique en se basant sur un modèle mathématique. Les RdP sont largement utilisés pour la modélisation des processus métier. Comme l'illustre le RdP de la figure 3.7, un réseau de Pétri est un graphe orienté et biparti c.à.d. qu'il a deux types de nœud Place/Transition: Une place modélise une condition ou l'état d'une ressource du système. Un réseau de pétri est défini alors par le tuple SN (P, T, W, i, o, l) avec :

P: ensembles de places,
T: ensemble de transitions représentant les opérations du service
$W \subseteq (P*T) \cup (T*P)$: ensemble d'arcs
i: le point d'entrée du service, $i= \{x \in P \cup T(x,i) \in W\} = \emptyset$,
o: le point de sortie du service, $o= \{x \in P \cup T(o,x) \in W\} = \emptyset$,
$l: T \rightarrow A \in \{t\}$ avec:
A: l'ensemble des noms des opérations
$t \notin A$, t est une opération qui ne porte pas de nom

Les conditions nécessaires pour déclencher une action sont modélisées par les arcs reliant une place aux transitions. Les effets d'une action sur l'état du système sont modélisés par les arcs joignant les transitions aux places. Si une place contient un jeton (ou marque) cela signifie que la condition représentée par cette place est vérifiée (exemple : impression terminée) ou indique la disponibilité d'une ressource dans le cas ou on a plusieurs jetons dans la place (exemple : nombre de pièces en stock). On dit que la transition est tirable si les conditions requises (ou les ressources nécessaires) pour déclencher l'action (ou l'événement), représenté par cette transition, sont satisfaites

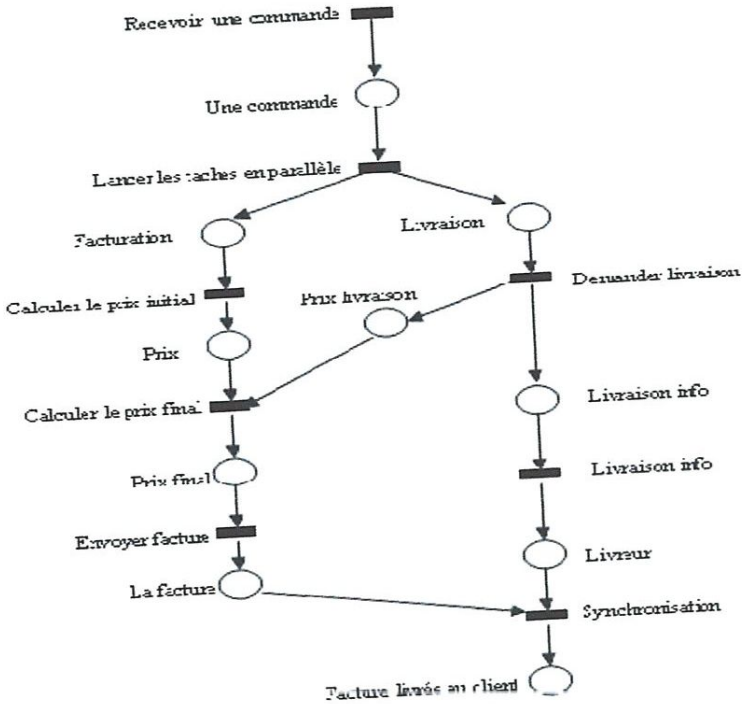


***Figure 3. 7** : Le processus de traitement de commande modélisé en RdP*

b. MCT : Model Conceptucl de traitement de Merise

Le MCT issus de Merise modélise les activités du domaine, activités conditionnées par les échanges avec l'environnement, sans prise en compte de l'organisation. Ainsi, chaque activité (nommée opération) regroupe un ensemble d'activités élémentaires réalisables au sein du domaine, sans autres informations extérieures (on n'a pas besoin de s'arrêter pour attendre des informations extérieures). De ce fait, une opération du MCT présente une vision macroscopique qui en fait l'intérêt dans l'analyse des processus métier, en particulier dans le Business Process Reengineering [21].

Le langage le plus utilisé est Unified Modeling Language (UML) qui permet de modéliser des processus métier grâce à un mécanisme d'extensibilité permettant de spécialiser les diagrammes généraux. [22]

c. Unified Modeling Language (UML)

UML est un langage pour spécifier, visualiser, construire, et documenter les artefacts des systèmes logiciels, ainsi que pour la modélisation d'entreprise et des systèmes non logiciels Comme son nom l'indique, ce langage est né de la fusion de plusieurs langages de modélisation, issus notamment de la méthode de Grady Booch particulièrement adaptée à la conception et à l'implémentation, de la méthode OOSE (Object Oriented Software Engineering) de Ivar Jacobson: qui permettait essentiellement l'expression des besoins, et de

la méthode OMT (Object Modelling Technique) de James Rumbaugh: pour l'analyse et applications orientées données[23]

UML propose des diagrammes spécialisés (dont les diagrammes d'activité, de séquence, de classes. . .) ayant chacun une fonction précise. Il n'existe pour le moment pas de diagramme UML spécialisé pour la modélisation des processus métiers. UML propose cependant un mécanisme d'extensibilité permettant de spécialiser chaque diagramme pour une utilisation particulière. Il est par exemple possible de spécialiser les diagrammes d'activité pour la modélisation des processus métiers [22].

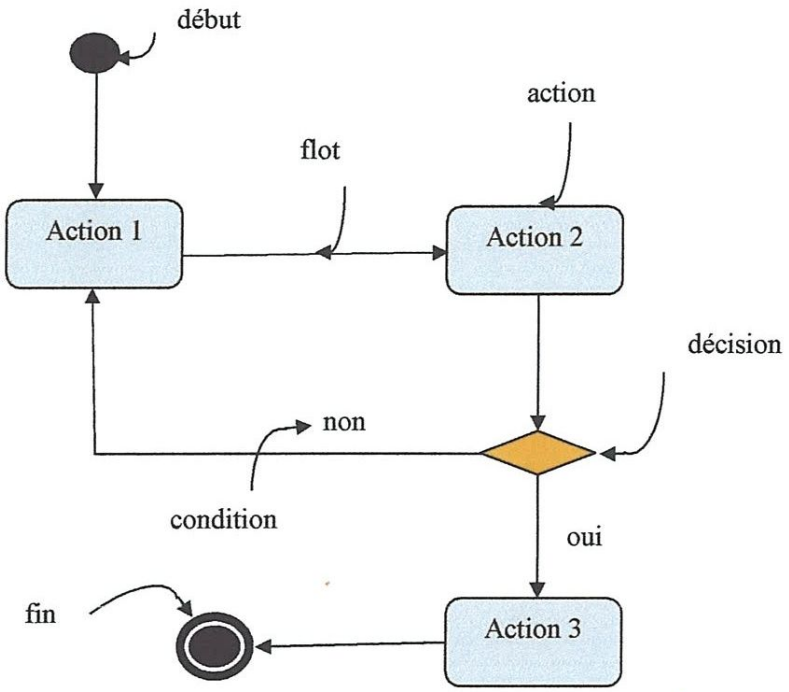


Figure 3. 8 : *diagramme d'activité d'un BP*

❖ **Remarque**

La panoplie de diagrammes de modélisation qu'offre UML permet de satisfaire les différents niveaux de modélisation des processus métiers, depuis la description générale jusqu'à la conception détaillée. Cependant le manque d'outils d'analyse rigoureux dans UML peut réduire sa capacité. Pour cela nous recourons aux réseaux de Pétri pour remédier à cet inconvénient. On peut constater que les réseaux de Pétri et UML possèdent des caractéristiques réciproques:

- UML est convivial alors que les réseaux de Pétri possèdent une rigueur formelle.

- UML peut décrire les systèmes de façon efficace pendant que les RdPs peuvent les analyser strictement.
- Le modèle UML peut être facilement implémenté tandis que le modèle des RdPs est plus adéquat pour la simulation.

d. Automates

Un automate est un modèle très connu dans le domaine de la spécification formelle de systèmes. Un automate est composé d'un ensemble d'états ou nœuds, un ensemble d'actions et un ensemble de transitions étiquetées entre les états. Les actions sont modélisées par des étiquettes et une transition étiquetée modélise ainsi l'exécution de l'action lors du franchissement de celle-ci.

- **Définition :**

Un automate est représenté formellement par un 5-uplet $(Q, \Sigma, \delta, q_0, F)$, où :

– Q est un ensemble d'états

– $\sum$ est un ensemble fini de symboles, appelé alphabet ou langage reconnu par l'automate (un automate peut reconnaître un langage formel)

– δ est la fonction de transition, telle que δ. $Q \times \sum \longrightarrow Q$

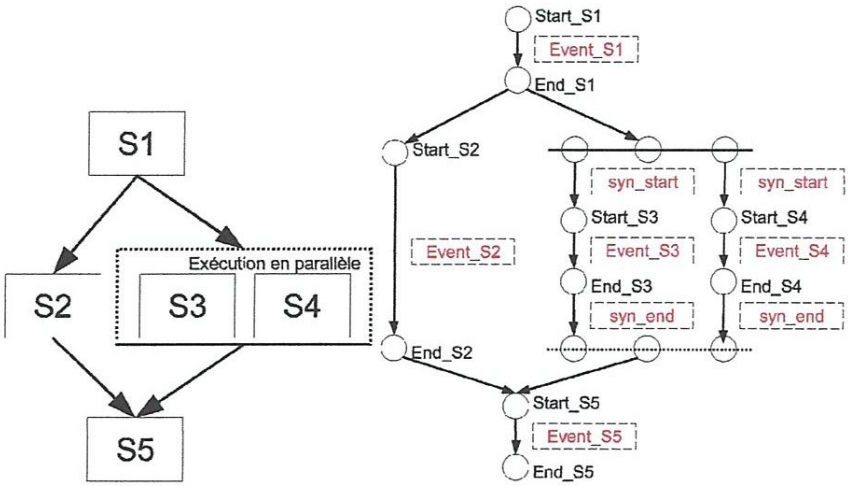
– q0 est l'état initial, c'est à dire l'état dans lequel est l'automate lorsqu'aucune entrée n'a encore été traitée, où $q0 \in Q$

F est un ensemble d'états de Q (c.à.d. $F \subseteq Q$) appelés états accepteurs.

Un automate prend en entrée un mot qui est un sous-ensemble de l'alphabet reconnu par l'automate. Lorsque le mot est entièrement lu, l'automate est stoppé et il est alors dans un état final. En fonction de l'état final, on dit que l'automate accepte ou refuse le mot en entrée. Plus précisément, si l'état final est un état accepteur alors on dit que le mot est accepté, sinon il est refusé.

- **Type d'automate**

De nombreux types d'automates existent tels que les automates à états finis, Entrée/Sortie ou temporisés. Les modèles à base d'automates peuvent être utilisés pour décrire, spécifier et vérifier la composition de services. Pour modéliser la composition de service à l'aide d'un automate, on assigne généralement un état à chaque invocation de service, un événement à chaque réponse d'un service et un deuxième état pour indiquer la fin de l'activité d'invocation. Un exemple d'une telle modélisation est fourni dans la figure 3.9 ci-dessous



(a) Scénario de composition (b) Modélisation avec les automates

Figure 3. 9 : *Exemple de modélisation de service composé avec les automates [12].*

i. Automates à états fini

Un automate à états finis est un modèle comportemental qui contient un nombre fini d'états. Un des états est appelé état initial et on distingue un autre sous-ensemble d'états finaux.

Un automate à états finis est un quintuplé A= (X, Q,q_0, II, F) tel que :

- X est un alphabet en entrée.
- Q est un ensemble fini d'états de A.
- $q_0 \in Q$ est l'état initial.
- $F \subseteq Q$ est l'ensemble des états finaux.
- II : $Q \times X \rightarrow Q$ est la fonction de transition de l'automate.

✓ Il existe deux types d'automates à états finis : « **Automate à états finis déterministe** et **Automate à états finis non déterministe** »

- **Automate à états finis déterministe (*AFD*)**

Dans ce type d'automate la fonction de transition ***II*** est définie par ***II*** : $Q \times X \rightarrow Q$ telle que ***II*** $(q, x) = q' \ \forall q \in Q$ et $x \in X$ *i.e.* l'image par ***II*** d'un couple (q, x) est un état unique de l'automate.

Remarque : cette définition veut dire qu'à partir d'un état on ne peut transiter vers qu'un seul état en lisant la même lettre.

Les protocoles d'un service Web seront représentés par des *AFD*.

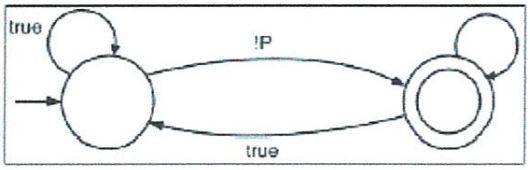


Figure 3. 10 : *Représentation graphique d'un automate d'états déterministe*

- **Automate à états finis non déterministe**

Dans ce type d'automate la fonction de transition Π est définie de Q×X dans l'ensemble de parties de Q.

- Π : Q×X→Q (Q) L'image par Π d'un couple (q, x) est un sous ensemble de Q.
- $(q, x) \longrightarrow \{q_i\}$

RQ : cette définition veut dire qu'on peut transiter d'un état vers un ensemble d'états en lisant la même lettre contrairement à l'automate déterministe.

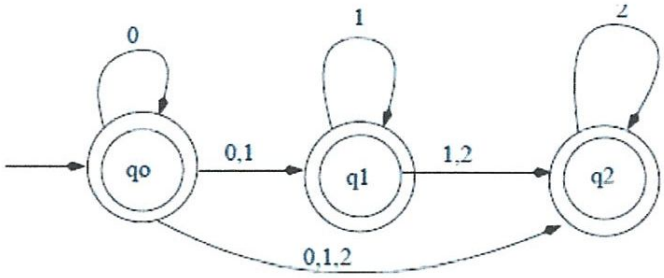


Figure 3. 11 : *Automate non-d'eterministe*

ii. Les automates temporisés

Ils étendent les automates d'états finis par l'ajout de contraintes temporelles afin de modéliser le comportement de systèmes temps-réel. Plus simplement, cela signifie que des horloges peuvent être définies et utilisées pour servir de garde au niveau des transitions. La valeur des horloges est incrémentée avec le temps et celle-ci peut-être réinitialisée. Ces valeurs sont utilisées pour la vérification de contraintes temporelles et pour s'assurer que les transitions ne sont franchies que si la contrainte associée est satisfaite. Ceci garantie que les transitions ont lieu au bon moment dans le processus. Les automates temporisés sont particulièrement pratiques et efficaces pour la vérification.

iii. Les automates à entrées/sorties (E/S)

Ce sont des automates où les actions peuvent être des entrées, des sorties où des comportements internes. Ils sont conçus pour permettre la modélisation de systèmes à événements discrets formés de composants concurrents. Les automates E/S ont initialement

été introduits pour modéliser les calculs distribués dans les réseaux asynchrones et pour la preuve d'algorithmes distribués. Ils sont désormais largement utilisés pour modéliser tous types de systèmes distribués et réactifs.

N.B l'utilisation d'automate basée sur les états est:

- ✓ facile à comprendre pour les utilisateurs
- ✓ approprié pour décrire le comportement réactif
- ✓ utile pour le contrôle d'exécution de services.

Par conséquent la modélisation avancée de processus métier basée sur les automates permet l'amélioration et l'extension pour la couverture des besoins d'échanges d'informations entre les processus métier complexes.

e. OWL-S (Ontology Web Language for Services)

Aussi appelé DAML-S vise à réaliser une vision de Web sémantique pour rendre les ressources Web disponibles aussi bien par contenu que par mots clés. OWL-S organise les services Web en ontologies. Une ontologie est définie par une spécification formelle et explicite d'une conception partagée. OWL-S divise les descriptions de service en profil de service, modèle et fondement. Le profil de service fournit une description haute niveau d'un service Web. Il exprime l'entrée requise du service et les sorties que le service va fournir au demandeur. Le modèle de service définit les opérations et leurs flots d'exécution dans le service Web. Le fondement de service fournit un mapping entre OWL-S et le standard WSDL et décrit comment le service est invoqué [24]. C'est un modèle pour la description déclarative de processus de services web. Leur transformation en systèmes d'états transitions permet de les soumettre à un moteur de planification (et ensuite pour les auteurs de comparer les performances entre une approche de composition sémantique vs. exécutable). La tâche de composition consiste alors à trouver un plan satisfaisant un but de composition dans le domaine des services disponibles. Les plans générés par l'algorithme de planification sont des automates pouvant être traduits en code exécutable.

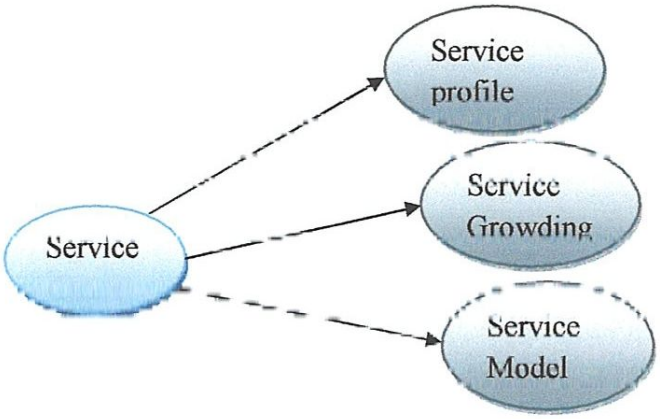


Figure 3. 12 : *Exemple d'OWL-S.*

f. Le processus BPEL abstrait

L'utilisation du langage BPEL est la définition de protocoles métiers entre plusieurs partenaires également désignés comme « processus abstraits ». Un processus abstrait ne décrit que les aspects comportementaux entre processus BPEL indépendamment de leur exécution plaçant l'ensemble dans une mode de fonctionnement plus proche de la chorégraphie que de l'orchestration de services Web. Pour rappel, un protocole définit un mode de communication en établissant des règles d'échange de messages entre plusieurs intervenants. La définition d'un protocole métier nécessite d'adresser plusieurs points en particulier :

- La description du comportement en ajoutant éventuellement l'utilisation de constructions logiques et temporelles de traitement des informations.
- La description des conditions d'exceptions, leurs conséquences et leurs activités de recouvrement.
- La description des interactions de longues durées et la coordination du succès ou de l'échec des composantes à différents niveaux de granularité.

Les processus abstraits utilisent l'ensemble des éléments définis du langage BPEL. Cependant, la description des données et des propriétés des messages reste abstraite en s'arrêtant uniquement au niveau public du processus BPEL. L'extension du langage BPEL sur la définition des processus BPEL abstrait concerne essentiellement la définition et l'affectation de valeur des variables dans les échanges de messages.

9. Evaluation des modèles de représentation des BP

Apres l'exposé des différents modèles de représentation des Bisness Process , nous avons dégagé un ensemble de critères permettant de dresser une comparaison et une analyse comparative sur les forces et les faiblesses de chacun (tableau 3.2)

	Contrôle d'exécution	Facilité de composition	Expressivité de model	Formalise mathématique	Notion d'Etat	Outils de vérification et de description
Approche de réseau de pétri	+	+	+	+	+	**Oui**
Approche par automate	++	++	++	++	++	**Oui**
MCT	+	+/-	+	-	+	**Non**
BPEL	+	+	+	-	-	**Oui**
OWL-S	+	+/-	+	+	+	**Oui**
Diagramme de Séquence	+	-	+	-	-	**Oui**

Tableau 3. 2 : *Comparaison entre les différentes modèles de représentation des Processus métiers*

10. Avantages et Inconvénients des processus métier :

L'étude de principe de chaque approche dans quelques travaux nous a permis de synthétiser dans le tableau 3.2 les avantages et les inconvénients des processus métier, ainsi que le domaine d'application typique.

	Avantages	Inconvénients	Applications candidates
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.

Tableau 3. 3 : *Avantages et inconvénients du processus métier.*

11. Problématique et Motivations

Si les BP sont exprimés avec des langages procéduraux (type BPEL [10], BPMN [11]), il est très difficile, voire impossible, de pouvoir extraire certaines caractéristiques exprimant un besoin particulier (une spécification fonctionnelle, un pattern de sécurité, un pattern d'exécution…) vue la syntaxe complexe de ces programmes. Par contre, avec un langage déclaratif (et même visuel, comme celui permettant la spécification des BP proposés par les fournisseurs de logiciels : Friendly GUI), la tâche serai plus facile pour les gestionnaires de protocoles, les utilisateurs et les développeurs voulant explorer les BP à la recherche de certains éléments (sous protocoles, patterns) représentant un intérêt particulier. En se sens nous proposerons de répondre à ce souci par une approche déclarative (type SQL) qui permettra d'interroger et de filtrer facilement une base de BP.

Les motivations de notre travail sont les suivantes :

i. Offrir un langage commun : Langage de requêtes générique

Nous visons à proposer un langage commun qui permettra aux programmeurs, développeurs et utilisateurs des BP, d'une part de spécifier leur BP à travers une interface graphique et visuelle, et d'autre part de formuler leurs requêtes dans le même langage sans avoir à recourir, ni à l'exploration du code source du BP généré, ni à se plonger dans les détails liés au langage de programmation. Pour atteindre cet objectif, il faut se situer à un niveau d'abstraction élevé afin de répondre à ce double objectif et accélérer l'apprentissage du langage de requêtes à proposer.

Cette motivation répond d'une manière conséquente aux insuffisances liées aux anciens systèmes, où les développeurs utilisent deux langages: Le premier de haut niveau : Graphique, visuel avec une interface conviviale permettant la spécification des BP. Le deuxième niveau, permettant d'exprimer les besoins spécifiques (patterns d'intérêt), est formulé dans le langage (BPEL) et exige donc une connaissance profonde de sa syntaxe et du fonctionnement du BP lui-même. L'écart entre ces deux langages est analogue à celui qui existe entre les langages évolués et les langages assembleur.

Il s'agit, donc, **d'unifier le niveau d'abstraction** en rehaussant celui relatif à la spécification des patterns. Cette motivation permettra d'éviter que les requêtes ne soient exprimées dans un langage prioritaire et ne seront pas, donc, portables.

ii. Facilité la rétro-conception des BP:

L'analyse de l'impact de l'évolution dynamique des protocoles, d'une manière graphique et visuelle via le langage de requêtes, offrira une convivialité et une supervision souple des BP. La valeur ajoutée consiste à décharger les gestionnaires de processus du suivi individuelle des instances en cours et de se focaliser, par la suite, sur l'amélioration du BP lui-même, en prenant en compte les critères de performance telles que : les chemins qui ne sont jamais parcourus dans l'ancienne version, les états dans la nouvelle version qui ne sont plus exécutés, les statistiques d'abandon des processus (ancien /nouveau) et à partir de quelle phase...etc. Ces éléments sont déterminants sur les plans économique et organisationnel. Armés d'une visibilité claire et doté de ces métriques opérationnels pour la prise de décision, les gestionnaires de processus, pourront opter pour une rétro-conception (Ré-ingéniering) de processus métier afin d'en améliorer les performances, sans se soucier des détails de bas niveau.

D'un point de vue purement pragmatique, la motivation principale de ce travail de recherche est la proposition d'un cadre conceptuel (Framework) et un outil logiciel pour la supervision des protocoles de services, et plus particulièrement, lors des opérations d'évolution de ces protocoles. Le cadre à proposer permettra de définir les nouvelles stratégies d'exécution (Nouveau Business Protocoles) sur la base des résultats des interrogations portant sur les exécutions en cours. En plus, il permettra d'évaluer les performances des services proposés via leur protocole et d'appréhender l'impact de leur évolution sur les conversations en cours. Il offrira, par ailleurs, au gestionnaire des services un tableau de bord et un outil d'aide à la décision (visualisation des ressources, paramètres de qualité de service, détection de goulots d'étranglement).

que RQL, TRIPLE, SQL, XPATH… SPARQL, un langage de requête proposé par W3C et dédié à RDF, est largement utilisé dans le domaine de recherche et d'extraction d'informations [25].

A. RDF (Resource Description Framework)

RDF est un langage qui permet de décrire des ressources. Une ressource peut être, par exemple, une page Web, un élément de fichier XML, un concept (utilisé pour décrire la ressource Web) que l'on veut lui-même décrire. RDF est un modèle de données, domaine-indépendant. Le vocabulaire d'un domaine donné doit être décrit en RDF schema (RDFS).

RDFS permet d'écrire le vocabulaire d'un domaine donné et les relations entre les objets de ce vocabulaire. Ce vocabulaire est utilisé pour annoter les ressources en RDF. RDF permet de définir des méta-données associées aux ressources du Web. En d'autre terme un document RDF est un ensemble d'instructions (statements).

Une instruction est un triplet **Ressource-Propriété-Valeur**

- Chaque **ressource** a un identifiant unique (URI : Universal Ressource Identifier).
- Une **propriété** est également une ressource qui a un rôle spécial. Elle permet de décrire une autre ressource.
- Une **valeur** peut être un littéral ou une ressource.

Un document RDF est un graphe orienté avec des noeuds étiquetés et des arcs étiquetés.

✓ **Par exemple :**

Gérard. Fadel est le propriétaire de la page Web http://www.inapg.fr/omip/gerard.htm

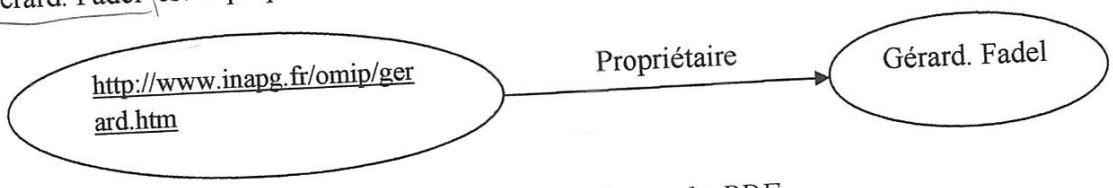


Figure 4. 1 : *Exemple de graphe RDF*

- La représentation choisie pour les traitements automatiques est exprimée en syntaxe XML.

Un document RDF est un arbre XML dont l'élément racine a pour nom rdf:RDF.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:monDom="http://www.inapg.fr/mon-rdf-ns#">
<rdf:Description rdf:about="http://www.inapg.fr/omip/david.htm">
<monDom:proprietaire>
Gerard.Fadel
</monDom:proprietaire>
</rdf:Description>
</rdf:RDF>
```

Figure 4. 2 : *Document RDF simple*

- La représentation en syntaxe XML d'un document RDF impose une sérialisation des instructions RDF. L'ordre des instructions n'a pas d'importance.

a. L'attribut rdf:resource

Cet attribut permet de référencer, dans une instruction RDF, une valeur de type ressource

```
<rdf:Description rdf:about="http://www.inapg.fr/omip/cours111.htm">
<monDom:nomCours> Java avancé </monDom:nomCours>
<monDom:estEnseignePar rdf:resource="1234"/>
</rdf:Description>
<rdf:Description rdf:about="1234">
<monDom:nom> Gérard. Fadel</monDom:nom>
<monDom:title> étudiant master </monDom:title>
</rdf:Description>
```

Figure 4. 3 : *Document RDF avec attribut*

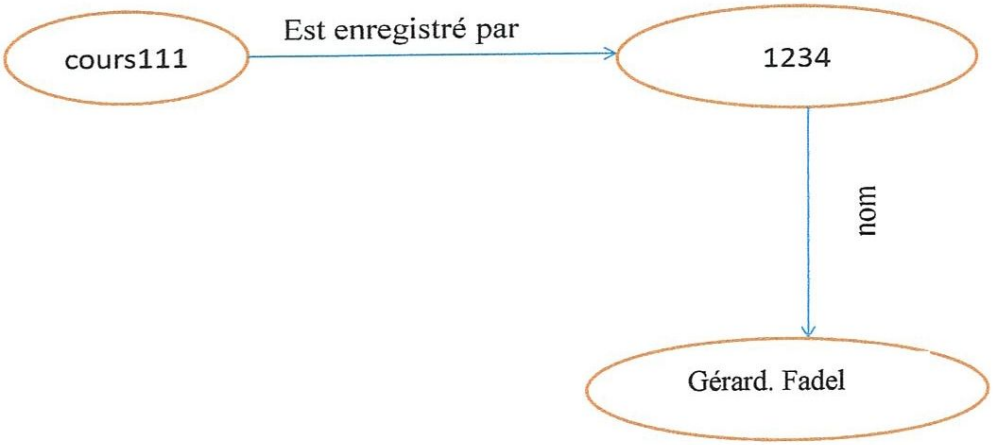


Figure 4. 4 : *Graphe RDF*

b. Descriptions imbriquées

Une description peut être définie à l'intérieur d'une autre description. Par exemple :

```
<rdf:Description rdf:about="http://www.inapg.fr/omip/cours111.htm">
<monDom:nomCours> Java avancé </monDom:nomCours>
<monDom:estEnseignePar>
<rdf:Description rdf:about="1234">
<monDom:nom> Gérard. Fadel </monDom:nom>
<monDom:title> étudiant master </monDom:title>
</rdf:Description>
</monDom:estEnseignePar>
</rdf:Description>
```

Figure 4. 5 : *Description de Ressource imbriquées*

La description imbriquée peut être référencée par une autre description. Sa portée est globale.

c. L'élément rdf:type

L'élément rdf:type permet de définir qu'une ressource est une instance d'une classe RDFS.

```
<rdf:Description rdf:about="http://www.inapg.fr/omip/cours111.htm">
<rdf:type rdf:resource="monDom;cours"/>
<monDom:nomCours> Java avancé </monDom:nomCours>
<monDom:estEnseignePar rdf:resource="1234"/>
</rdf:Description>
```

Figure 4. 6 : *Description RDF avec élément Type*

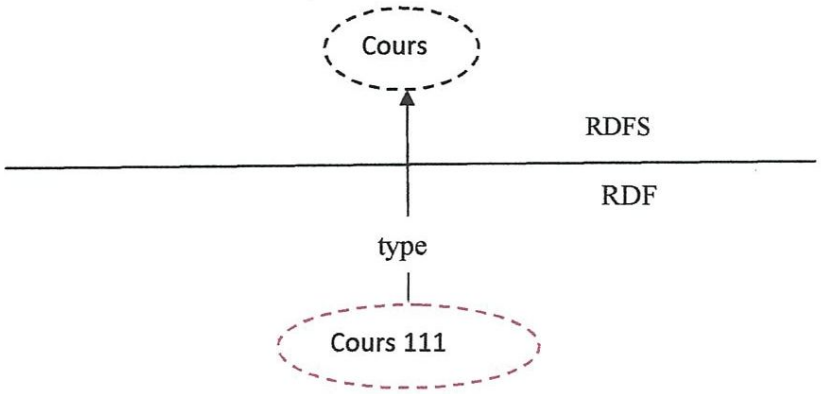


Figure 4. 7 : *Graphe RDF avec élément type*

i. Syntaxe abréviée

Un élément propriété sans fils peut être remplacé par un attribut XML de l'élément description dans lequel il est inclus.

```
<rdf:Description rdf:about="Cours111" monDom:nomCours="Java avancé">
<rdf:type rdf:resource="monDom;cours"/>
<monDom:estEnseignePar rdf:resource="1234"/>
</rdf:Description>
```

Dans un élément description incluant un élément de typage (rdf:type), on peut utiliser le nom du type à la place de **rdf:description.**

```
<monDom:cours rdf:about="Cours111" monDom:nomCours="Java avancé">
<monDom:estEnseignePar rdf:resource="1234"/>
</monDom:cours>
```

d. RDF schéma

RDF est un langage universel qui permet de décrire des ressources. Il est indépendant de toute application. Le vocabulaire employé pour décrire les ressources est défini en RDFS : il est composé d'un ensemble de classes et d'un ensemble de propriétés.

→ RDFS permet de définir des classes et une hiérarchie de spécialisation sur les classes. Une ressource RDF peut être une instance d'une classe RDFS (rdf:type).

→ RDFS permet également de définir des propriétés et une hiérarchie de spécialisation sur les propriétés.

→ RDFS permet de définir des restrictions sur la valeur d'une propriété (range) et sur le type de ressource décrit par la propriété.

e. Le modèle formel de RDF

Un ensemble de descriptions RDF est un graphe orienté étiqueté dans lequel les nœuds sont des ressources (ou des littéraux) et les arcs représentent les propriétés des ressources.
Un schéma RDF définit le vocabulaire des étiquettes des nœuds du graphe (appelées classes ou types de littéraux) et des arcs (appelés types de propriété). Les deux types d'étiquettes peuvent être organisés en taxonomie (hiérarchie de spécialisation). On note C, l'ensemble des noms de classes, P, l'ensemble des noms de propriétés et L, l'ensemble des types de littéraux.

Définition d'un schéma RDF : Un schéma RDF est un quintuplet $RS = (Vs, Es, \Psi, \lambda, H)$ ou

- VS et ES représentent respectivement l'ensemble des noms de nœuds et d'arcs ;
- H représente la taxonomie notée : $H = (C \cup P, \propto)$;
- λ est une fonction de typage définie sur : $\lambda : Vs \cup Es \rightarrow T$, T est l'ensemble des noms de types manipulés en RDF (type classe, type propriété, type littéral, type URI, type Bag, type Séquence, type Alternative, type méta-classe) ;

Ψ est une fonction de contrainte sur les propriétés :

$\Psi : Es \rightarrow Vs \times Vs, \forall e \in Es, domain(e) \in Vs \subset C$ et $range(e) \in Vs \subset C \cup L$

B. Le Langage de Requête SPARQL :

SPARQL est un langage de requêtes et un protocole pour l'accès RDF, conçu par le groupe de travail du W3C RDF Data Access depuis Avril 2006. Permettant d'interroger un ensemble de descriptions RDF à partir d'opérations de mise en correspondance de patterns de graphes, de conjonctions, de disjonctions de patterns et de patterns optionnels. SPARQL intègre également des balises (tags) spécifiques l'union et l'intersection, le filtrage, les opérateurs de comparaison des valeurs... permettant d'effectuer des requêtes plus efficaces et flexibles.

1. Caractéristiques

a. Structure du SPARQL

Il est adapté à la structure spécifique des graphes RDF, et s'appuie sur les triplets qui les constituent. En cela, il est différent du classique SQL (langage de requête qui est adapté aux bases de données de type relationnelles), mais s'en inspire clairement dans sa syntaxe et ses fonctionnalités. Il a aussi quelques traits de ressemblance mineurs avec Prolog.

SPARQL permet d'exprimer des requêtes interrogatives ou constructives :

La structure d'une requête SPARQL est très similaire à celle employée dans le langage SQL Une requête SELECT, de type interrogative, permet d'extraire du graphe RDF un sous-graphe correspondant à un ensemble de ressources vérifiant les conditions définies dans une clause WHERE:

SELECT? v1? v2 ... ? Vn
FROM <description.rdf>
WHERE {
(sujet$_1$ | v$_i$) (predicat$_1$ | v$_j$) (objet$_1$ | v$_k$).
...
(sujet$_x$ | v$_a$) (predicat$_y$ | v$_b$) (objet$_z$ | v$_c$).
}

De plus, une requête SPARQL peut avoir d'autres finalités que de fournir un ensemble de correspondances aux variables spécifiées dans le SELECT. En effet, dans le langage SPARQL, il est possible de demander si une requête dispose d'au moins une solution. Pour ce faire, le SELECT est remplacé par un ASK. Il est aussi possible de construire un nouveau graphe RDF via le terme CONSTRUCT.

b. Pattern de graphe de base :

Un pattern est composé d'une ou plusieurs description(s) RDF dans lesquelles on peut introduire une (ou plusieurs) variable(s) à la place de la ressource/propriété/valeur littérales à retourner.

✓ **Exemple :**

1°) Besoin

Un utilisateur veut chercher une ressource désignant le nom d'une personne se trouvant dans un fichier bien spécifique. Soit la liste de descriptions RDF interrogée est la suivante :

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
_:a foaf:name "Gerard .Fadel" .
_:a foaf:mbox <mailto:outlaw@example.com> .
_:b foaf:name "A. N. Other".
_:b foaf:mbox <mailto:other@example.com> .
```

```
PREFIX foaf: <http://xmlns.com/foaf.rdf/0.1/>
SELECT ?mbox
WHERE
{ ?x foaf:name "Gerard.Fadel".
?x foaf:mbox ?mbox }
```

Figure 4. 8 : *Exemple simple de requête RDF*

Ce qui se lit :

chercher dans le fichier "foaf.rdf" (ci-dessus) toutes les chaînes (ici désignées par "?mbox") telles qu'il existe dans le graphe correspondant deux triplets de la forme "?x foaf:name "Gerard Fadel"." et "?x foaf:mbox ?mbox ", avec la même valeur pour "?x".

Le résultat est :

mbox
<mailto:outlaw@example.com>

Remarque

Une requête SELECT, de type interrogative, permet d'extraire du graphe RDF un sous-graphe correspondant à un ensemble de ressources vérifiant les conditions définies dans une clause WHERE

c. Pattern de graphe optionnel :

Un pattern de graphe de base est satisfait si toutes les variables peuvent être liées. Certaines parties du pattern peuvent être rendues optionnelles.

✓ **Exemple :**

1°) Besoin

Un utilisateur veut chercher une ressource d'une personne se trouvant dans deux fichiers quelconques bien spécifiques. Soit la liste de descriptions RDF interrogée est la suivante :

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
_:a rdf:type foaf:Person .
_:a foaf:name " Gerard. Fadel " .
_:a foaf:mbox <mailto:alice@example.com> .
_:a foaf:mbox <mailto:alice@work.example> .
_:b rdf:type foaf:Person .
_:b foaf:name "Bob".
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT? name? mbox
WHERE {? x foaf:name ?name .
OPTIONAL {? x foaf:mbox ?mbox }
}
```

Figure 4. 9 : *Requête RDF avec paramètre optionnels*

Ce qui se lit :

chercher dans les fichier "foaf.rdf" (ci-dessus) tous les chemins (ici désignées par "?mbox"et namc) tcllcs qu'il cxiste dans le graphe correspondant deux triplets de la forme "?x foaf:name "Gerard. Fadel"," et OPTIONAL "? x foaf:mbox ?mbox ", avec la même valeur pour "? x".

Le résultat est la suivante :

name	mbox
" Gerard. Fadel "	<mailto:alice@example.com>
" Gerard. Fadel "	<mailto:alice@work.example>
"Bob"	

d. Union de pattern de graphe:

SPARQL permet aussi de combiner des patterns de graphe de telle manière qu'une des alternatives puisse être matchée.

✓ **Exemple :**

1°) Besoin

Un utilisateur veut chercher une ressource concernant un livre se trouvant dans deux fichiers quelconques bien spécifiques. Soit la liste de descriptions RDF interrogée est la suivante :

Soit la liste de descriptions RDF interrogée

```
@prefix dc10: <http://purl.org/dc/elements/1.0/>.
@prefix dc11: <http://purl.org/dc/elements/1.1/>.
_:a dc10:title "SPARQL Query Language Tutorial".
_:a dc10:creator "Alice" .
_:b dc11:title "SPARQL Protocol Tutorial".
_:b dc11:creator "Bob".
_:c dc10:title "SPARQL" .
_:c dc11:title "SPARQL (updated)".
```

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT? title
WHERE { { ?book dc10:title ?title } UNION { ?book dc11:title ?title } }
```

Ce qui se lit :

Renvoie la liste des titres de livre enregistrés soit avec les propriétés de la version 1.0 ou 1.1.

Le résultat est suivant:

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

e. Le filtrage

La clause *FILTER* introduit une expression booléenne que les solutions à la requête doivent satisfaire. Le langage dans lequel s'écrit cette expression est très riche. Il contient des opérateurs de comparaison et des fonctions d'accès.

✓ **Exemple**

1°) Besoin

Nous faisons l'hypothèse que deux personnes qui ont le même nom de famille se connaissent (foaf:knows). Mais nous voulons éviter les résultats banals "X foaf:knows X". Pour cela, nous pouvons demander que les deux ressources anonymes désignant les "vcard:N" soient différentes. Ainsi cette requête ci-dessous explique un peu cela

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX vcard:  <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT {?x foaf:knows ?y }
FROM <http://pagesperso-systeme.lip6.fr/Jean-
Francois.Perrot/inalco/XML/RDF/Exemples/vcardCat.rdf>

WHERE
{       ?x  vcard:N ?va .
        ?va vcard:Family ?f .
        ?y  vcard:N ?vb .
        ?vb vcard:Family ?f .
    FILTER (? va != ?vb )
```

Figure 4. 10 : *Requête RDF avec filtrage*

a pour résultat:

```
@prefixvcard. <http://www.w3.org/2001/vcard-rdf/3.0#>.
@prefixfoaf:  <http://xmlns.com/foaf/0.1/>.
                <http://somewhere/RebeccaSmith/>
                 foaf:knows<http://somewhere/JohnSmith/>.
                <http://somewhere/JohnSmith/>
                 foaf:knows  <http://somewhere/RebeccaSmith/>.
                 <http://somewhere/SarahJones/>
                 foaf:knows  <http://somewhere/MattJones/>.
```

<http://somewhere/MattJones/>

foaf:knows <http://somewhere/SarahJones/> .

II. Comparaison SQL / SPARQL

	SQL	SPARQL
Critère de comparaison	Base de données	Schémas et graphe
Elément de base	Enregistrement	Descripteur de Ressources
Critère d'accès	Identifiant unique	URI unique
Structure des éléments	Attributs des enregistrements	Liste des prédicats
Valeur du contenu	Valeurs des attributs ou des clés Etrangères	Valeurs qui sont des littéraux ou des liens vers d'autres ressources

Tableau 4. 1 : *comparaison des langages SQL et SPARQL*

III. Exploration des bases de données de protocoles avec le langage déclaratif :

Un gestionnaire de protocole a toujours besoin de chercher non pas un protocole en entrer mais certains éléments (structure) ou spécifications qui répondent à un besoin de gestion particulier. Par ailleurs, il peut interroger la base des traces pour voir le niveau d'avancement des instances actives dans leur exécution, aux fins de monitoring (supervision).

a. Notion de spécification

Une spécification relative a un processus métier est une description plus ou moins formelle d'un sous ensemble d'activités.

✓ **Exemples** dc préoccupations pouvant être exprimées par des Spécifications

i. Sur les schémas

Quels sont les protocoles qui contiennent c passant par **a** suivis de **b** ?

Quels sont les protocoles qui contiennent une opération **a** suivis de **b** ?

Quels sont les protocoles qui contiennent une opération **a** non suivis de **b** ?

Quels sont les protocoles qui contiennent une opération bouclée sur **c** et passé par **a** suivis de **b** ?

Quels sont les chemins qui n'ont jamais été parcouru ?

j. Sur les traces

Quels sont les instances qui ont déjà atteint **c** passant par **a** suivis de **b** ?

Quels sont les instances qui ont déjà exécutée **a** suivis de **b** ?

Quels sont les instances qui ont parcouru **a** sans passée par **b** ?

Quels sont les instances qui ont déjà bouclée sur **c** et **a** suivis de **b**

IV. Représentation formelle des spécifications relatives aux BP :

L'utilisation intensive de modèles permet un développement souple et itératif, grâce aux raffinements et enrichissements par transformations successives. Par ailleurs, les transformations permettent de passer d'un espace technique à un autre (par exemple d'automate vers les graphes ou vers du XML) [26]. De ce fait, les transformations permettent de choisir l'espace technique et le formalisme le plus approprié à chacune des activités.

Une instance de méta-modèle est un modèle d'automate observateur. Les modèles d'observateur sont au format XML. Un modèle d'automate observateur est donc représenté par un fichier XML. Il décrit l'ensemble des états et des transitions, qui décide le comportement du système. [27]

✓ **Exemple 1** (XML et automate)

Soit un modèle, on utilise <xmlns :automaton > </automaton> pour définir que c'est un automate. L'attribut "Etat_Destination" est utilisé pour déterminer l'état final et "Etat_Source" pour déterminer l'état initial. < Etat/> est pour définir un état. Il a l'attribut "Transition Etat_Source" pour déterminer les transitions entrantes et l'attribut Transitions "Etat_Destination" pour déterminer les transitions sorties. <transition/> est pour définir une transitions. L'attribut EtatX pour déterminer l'état de départ et l'attribut EtatY pour déterminer l'état d'arrivée. Le modèle ci-dessous est de l'automate observateur de 'An action from X leads to an action from Y'.

```
<?xml version="1.0" encoding="UTF-8"?>
< xmlns:Automate>
  <Etats>
    <Etat Type="Final" Position="(355,39)">Etat01</Etat>
    <Etat Type="Final" Position="(381,284)">Etat02</Etat>
    <Etat Type="Simple" Position="(222,156)">Etat03</Etat>
    <Etat Type="Simple" Position="(556,151)">Etat24</Etat>
  </Etats>
  <Transitions>
    <Transition Etat_Source="Etat03" Etat_Destination="Etat01"
Position="(288,98)">Message0</Transition>
    <Transition Etat_Source="Etat03" Etat_Destination="Etat02"
Position="(301,220)">Message1</Transition>
    <Transition Etat_Source="Etat03" Etat_Destination="Etat24"
Position="(389,154)">Message2</Transition>
  </Transitions>
</Automate>
```

Il y a trois états. Le troisième état est l'état final. Le premier état est état initial. Il y a trois transitions

✓ **Exemple 2 (graphe RDF)**

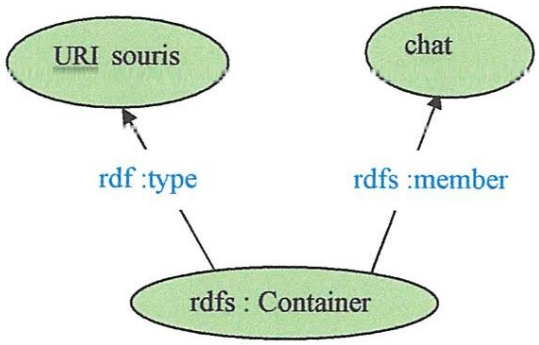


Figure 4. 11 : *Exemple de graphe RDF avec types de ressources identifiés*

Son code source est le suivant .

```
<?xml version="1.0"?>
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-  rdf-syntax-
ns#"xmlns:ex="http://ex.org/#">
        <ex:Chat rdf:about="http://ex.org/# Container ">
        <ex:Type rdf:resource="http://ex.org/#Souris"/>
</ex:Chat>
</rdf:RDF>
```

a. Exemple de requêtes d'extraction de spécification

Soit un graphe RDF ci-dessous donc le but est de trouver la personne avec un prénom et l'adresse de courrier électronique en exécutant les données RDF

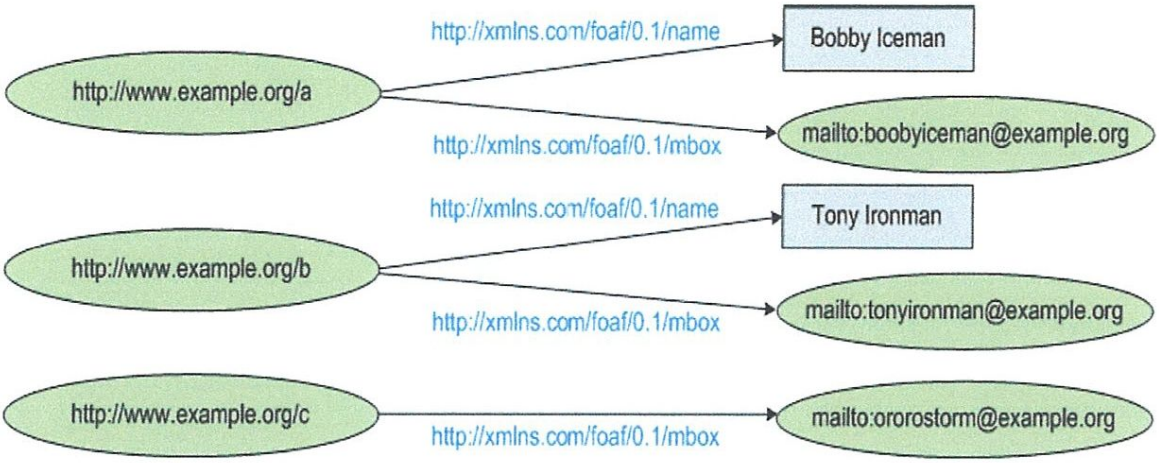


Figure 4. 12 : *Exemple de graphe RDF*

La question permettant d'interroger ce graphe et de répondre à la requête posée est la suivante :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
{ ?x foaf:name ?name .
?x foaf:mbox ?mbox
}
```

La clause du PRÉFIXE définit un nom pour l'emplacement FOAF. La clause ci-dessus spécifie ce que la question devrait rendre dans ce cas c'est le (nom variable et mbox). ou elle fournie le modèle graphique de base des données. Le résultat de cette question est montré dans Tableau 4.2 ci-après.

Name	Mbox
Bobby Iceman	<mailto : boobyiceman@example.org>
Tony Ironman	<mailto : tonyironman@example.org>

Tableau 4. 2 : *du résultat d'une requête SPARQL*

✓ **Exemple des traces**

Nous allons nous focaliser a la rechercher des étudiants moyen et les major dans une classe de fin d'année en vu d'effectuer l'attribution des sujets (PFE):

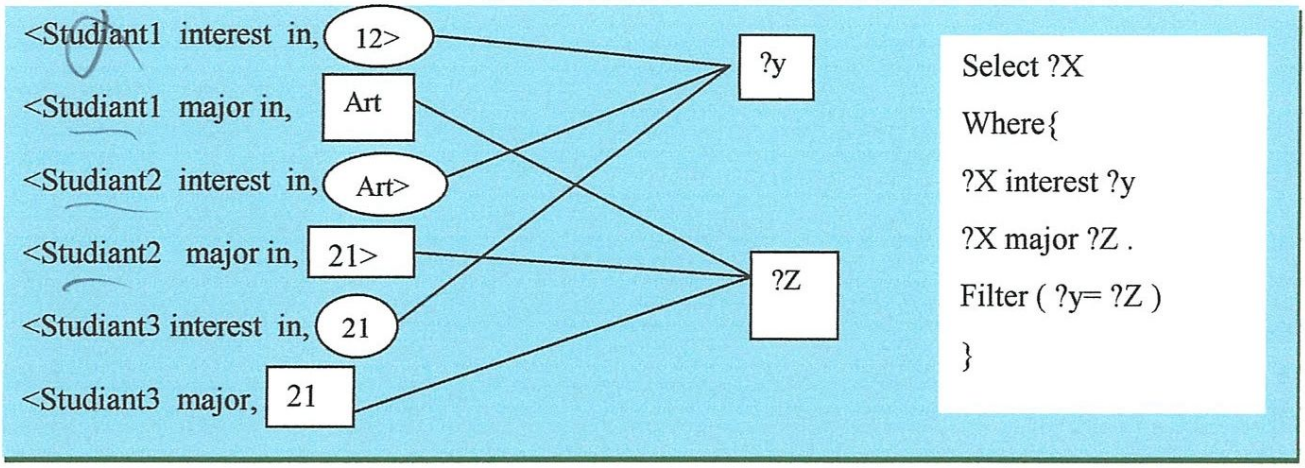


Figure 4. 13 : *Un exemple de données et de requête RDF sur les traces en SPARQL*

b. Utilisation de SPARQL pour l'interrogation des schémas

Toutes les préoccupations exposées précédemment trouvent leurs réponses en exploitant SPARQL. En effet en tant que langage de requête il permet de filtrer les structures (schémas) ou les traces (instances) qui répondent aux critères introduits dans la requête.

A titre d'illustration nous répondant aux deux premières questions pour chaque famille.

✓ **Exemple 1**

Elle affichera uniquement abc

```
SELECT * {
FILTER{
        a?  b?  c?
          }
ORDER BY? a?b ?c
}
```

✓ **Exemple 2:**

La requête suivante donnera les résultats de a à b

```
SELECT * {
FILTER{
        a?  b?
          }
}
```

2°) **base de schémas (RDF) :** les fichiers XML stockée dans la base de protocole subissent une transformation qui permet de transformer ces fichier en RDF.

3°) **traces :** elle est générée automatiquement a chaque événement exécutés depuis le début jusqu'à l'état actuel.

Des modules permettent la transformation d'une description d'un format a un autre XML vers RDF.

Trois modules principaux interagissent avec les BDD précitées, soit en consultation soit en mise à jour.

Le noyau du système est le moteur SPARQL qui permet d'exécuter les requêtes en interrogeant la BDD (RDF) et fourni les résultats.

Par contre XQUERY interroge la base des traces et fourni les résultats au format XML.

Description des flux

Numéros de Flux	Désignations de Flux
(1)	Transmission des protocoles au format XML dans la base
(2)	Transformation au format RDF
(3)	soumission d'une requête SPARQL en mode interrogatoire
(4)	Transmission de la réponse
(3')	soumission d'une requête XQUERY en mode interrogatoire
(4')	Affichage des résultats
(9)	Recherche d'un protocole en ligne 'internet '
(10)	Appropriation d'un résultat
(11)	Enregistrement des traces
(12)	Contacter le fournisseur i.e intéressé par un protocole
(13)	Vérification des protocoles en ligne

Tableau 4. 3 : *description des flux*

- **Exemple 1**

Supposons qu'un gestionnaire de protocole veut chercher un sous protocole qui permet de réaliser les opérations suivantes : sélection d'article suivi de commande puis validation alors il soumet sa requête ….etc.

- **Exemple 2**

Maintenant supposons qu'un fournisseur de service veut savoir quels sont les clients qui ont exécuté un chemin particulier de leur service Web qui est déjà publier .soit à chercher les traces contenant les 02 opérations : réservation puis payement. Dans ce cas : il va interroger la base de traces en envoyant une requête Xquery.

Conclusion

Nous avons présenté dans ce chapitre le langage de description du processus métier d'une manière déclarative. Nous avons vu que ce langage permet de couvrir tous les flux de contrôle de base et permet aussi de voir le processus selon trois plans d'abstraction : un plan métier pour décrire le processus à l'aide d'un nouveau langage appelé SPARQL, un plan de comportement pour gérer automatiquement le changement du processus et un plan opérationnel pour simuler et vérifier la bonne exécution du processus.

Le prochain chapitre sera destiné à l'implémentation de notre système et sa mise en œuvre.

Chapitre V : IMPLEMENTATION

Introduction

Dans ce chapitre, on va se consacrer à l'implémentation de l'approche proposée. Après avoir conçu l'architecture générale du système dans le chapitre précédent et avoir identifié les différents modules et leur interaction, nous allons aborder l'aspect pratique par la mise en œuvre opérationnelle dans les environnements logiciels adéquats.

Toutefois, nous avons développé les fonctionnalités qui nous ont paru les plus pertinentes pour l'illustration du filtrage des spécifications relatives aux processus métier.

Avant de présenter l'architecture, les fonctionnalités et les interfaces de notre système, nous citerons d'abord les outils utilisés pour son implémentation.

I. Présentation de l'environnement de travail

Pour le développement de notre système nous avons opté pour le langage JAVA. Les raisons qui ont motivé ce choix sont les suivantes.

1. Le choix d'Eclipse et JAVA

a. Eclipse

Eclipse est un environnement de développement gratuit. Il permet de développer, aussi bien, des programmes en java qu'en PHP. Il utilise le principe intéressant des plugins, ce qui permet d'ajouter des fonctionnalités selon les besoins.

Il possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

- Eclipse supporte plusieurs plate-forme d'exécution (Windows, linux ...) et il est extensible grâce au mécanisme de plug-ins.
- Supporte tout le cycle de vie du développement du fait qu'il intègre des dispositifs pour la modélisation, la programmation, le test et le déploiement des applications.

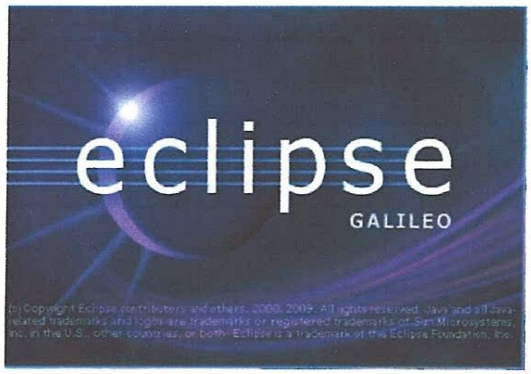


Figure 5. 1 : *interface Eclipse*

b. Java

Le langage java, né en 1995, est un langage orienté objets, il permet d'écrire de façon simple et claire des programmes portables sur la majorité des plates-formes. De plus, il bénéficie d'une très grande bibliothèque de classes avec lesquelles l'utilisateur pourra composer des interfaces graphiques, créer des applications multithreads, animer une page HTML par des applets ou encore communiquer en réseau. De plus Java assure une totale indépendance des applications vis-à-vis de l'environnement d'exécution : c'est à dire que toute machine supportant Java est en mesure d'exécuter un programme sans aucune adaptation (ni recompilation, ni paramétrage de variables d'environnement);

- Portabilité : Exécution du ByteCode.
- Java est un langage orienté objet, c'est à dire que nous n'allons pas manipuler des fonctions et des procédures mais des objets qui vont s'échanger des messages. Le principal avantage est que l'on peut réaliser une programmation modulaire : tous les objets peuvent être mis au point séparément ;
- Java nous permet un accès simplifié aux bases de données et bien d'autres fonctionnalités

2. Les Outils utilisés :

a. SPARQL:

SPARQL (prononcer « sparkle », en anglais « étincelle ») est un langage de requêtes et un protocole qui permet de rechercher, d'ajouter, de modifier ou de supprimer des données RDF disponible à travers Internet pour notre travail nous avons utilisé la version SPARQL 1.1. Son nom est un acronyme qui signifie « SPARQL Protocol and RDF Query Language ». Il permet de récupérer des informations depuis des serveurs SPARQL (ou endpoint SPARQL), aussi appelés triplestore. Parmi ces triplestores, on peut citer 4Store, Sesame, Jena et bien

d'autres. Il offre le même usage que SQL, mais en respectant les standards du W3C pour transmettre des données à travers le réseau. L'avantage est qu'une application pourra en théorie interroger à l'avenir n'importe quel serveur SPARQL sans se soucier du constructeur de la base de données. Aussi SPARQL permet de découvrir la structure d'une base de données.

b. Jena :

Jena est une API Java qui peut être utilisée pour créer et manipuler des graphes RDF comme nous avons utilisé ici la version jena-2.6.4-tests. Il possède des classes pour représenter des graphes, des ressources, des propriétés et des littérales. Les interfaces représentant des ressources, des propriétés et des littérales sont respectivement nommées Resource, Property et Literal. Dans Jena, un graphe est appelé un modèle et est représenté par l'interface Model. L'API de Java Jéna offre l'accès convenable et efficace à un serveur lors d'une application Java. Cet API fournit des méthodes pour créer, en mettant en question et gardant des données RDF, et pour diriger les triples entreposés.

c. XQUERY:

Le Xquery est un langage de requête destiné pour XML en cour de spécification par le W3C basé sur le système de type de XML Schéma .Il est compatible avec autres outils XML de W3C, notamment XPATH et XSLT. Il peut être formée d'expressions simples et composées combinées par des operateurs ou voir des mots clés.

II. Description de l'application de fouille de processus métiers

Après avoir justifié les choix des environnements et les outils qui nous ont aidés dans le développement de notre système, nous allons détailler les différents modules de notre système de fouille de processus métiers par une approche déclarative (D**BPM : Declarative approach for Business Processes Maining**). Nous exposerons les différentes étapes nécessaires à l'alimentation des bases de données par les protocoles métiers et par les traces d'exécution ainsi que des exemples de requêtes d'extraction de spécification ou de type de traces de traces qui répondent à certains besoins très pragmatiques pour le gestionnaire de protocoles. Nous illustrons notre exposé par des captures d'écran extraites du système réalisé.

1. Page d'accueil

Cette page de la figure 5.1 ci-dessous permet de fournit un guide sur l'ensemble des fonctionnalités offertes par le système.

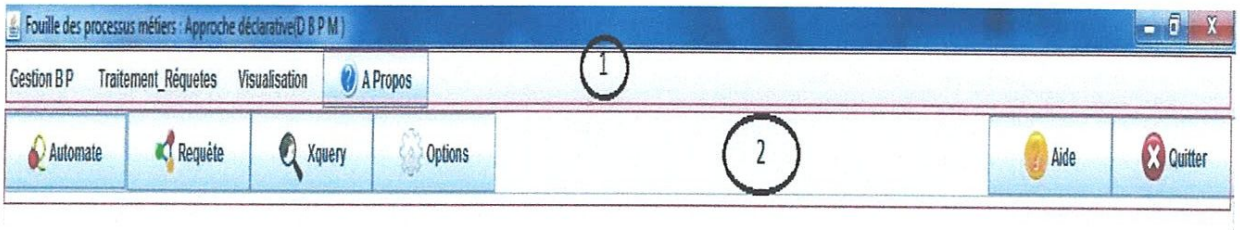


Figure 5. 2 : *Page d'accueil.*

1- Représente le Menu principale : Comporte déférente menu déroulants (gestion de BP, traitements de requête, etc.)

2- C'est la barre d'outils qui permet d'accélérer la navigation .

2. Gestion des BP

Cette option permet de faire la mise à jour de la base de données des protocoles, par l'ajout, la modification et la suppression des protocoles. Les protocoles sont gérés en tant que fichier XML et visualisés sous forme graphique (automates).

a. Création d'automates:

En vu de pouvoir créer un automate ou d'effectuer tout autre opérations la dessus ont procède par cliquer sur le bouton suivant 

ou depuis *Gestion des BP →automate*. Apres cette action on clique sur *fichier→ouvrir* puis l'automate affichera voir figure 5.3

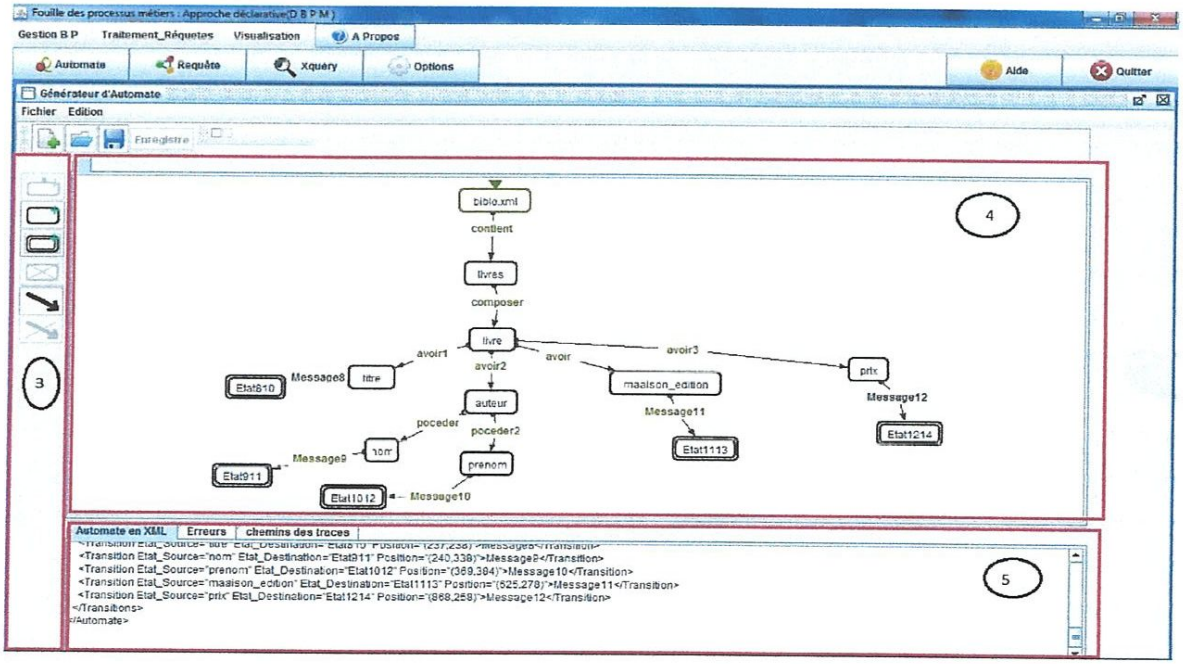


Figure 5. 3 *Interface de création d'automate.*

```
<Transition Etat_Source="biblo.xml" Etat_Destination="livres" Position="(426,53)">contient</Transition>
<Transition Etat_Source="livres" Etat_Destination="livre" Position="(433,144)">composer</Transition>
<Transition Etat_Source="livre" Etat_Destination="titre" Position="(370,208)">avoir1</Transition>
<Transition Etat_Source="livre" Etat_Destination="auteur" Position="(434,223)">avoir2</Transition>
<Transition Etat_Source="livre" Etat_Destination="maaison_edition" Position="(513,211)">avoir</Transition>
<Transition Etat_Source="livre" Etat_Destination="prix" Position="(644,202)">avoir3</Transition>
<Transition Etat_Source="auteur" Etat_Destination="nom" Position="(373,293)">poceder</Transition>
<Transition Etat_Source="auteur" Etat_Destination="prenom" Position="(433,303)">poceder2</Transition>
<Transition Etat_Source="titre" Etat_Destination="Etat810" Position="(237,238)">Message8</Transition>
<Transition Etat_Source="nom" Etat_Destination="Etat911" Position="(240,338)">Message9</Transition>
<Transition Etat_Source="prenom" Etat_Destination="Etat1012" Position="(369,384)">Message10</Transition>
<Transition Etat_Source="maaison_edition" Etat_Destination="Etat1113" Position="(625,278)">Message11</Transition>
<Transition Etat_Source="prix" Etat_Destination="Etat1214" Position="(868,258)">Message12</Transition>
</Transitions>
</Automate>
```

- **Les erreurs**

Liste d'erreus
*Liste d'etats non accissible :
*Liste d'etats qui n'accedont pas a un etat finale :

- **Chemins des traces**

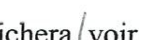
biblo.xml.contient.livres.composer.livre.avoir1.titre.Message8.Etat81
biblo.xml.contient.livres.composer.livre.avoir2.auteur.poceder.nom.Message9.Etat91
biblo.xml.contient.livres.composer.livre.avoir2.auteur.poceder2.prenom.Message10.Etat101
biblo.xml.contient.livres.composer.livre.avoir.maaison_edition.Message11.Etat111
biblo.xml.contient.livres.composer.livre.avoir3.prix.Message12.Etat121

7. Traitement des requêtes:

On peut réaliser deux types de requêtes : Sur les schémas et sur les traces d'exécution.

a. Requêtes sur les schémas :

Cette interface est accessible via le bouton suivant 

ou par *Menu-> traittement_requete-> requete* l'interface s'affichera sur laquelle on lance la requête ensuite on clique sur 

et le résultat s'affichera (voir la figure 5.4).

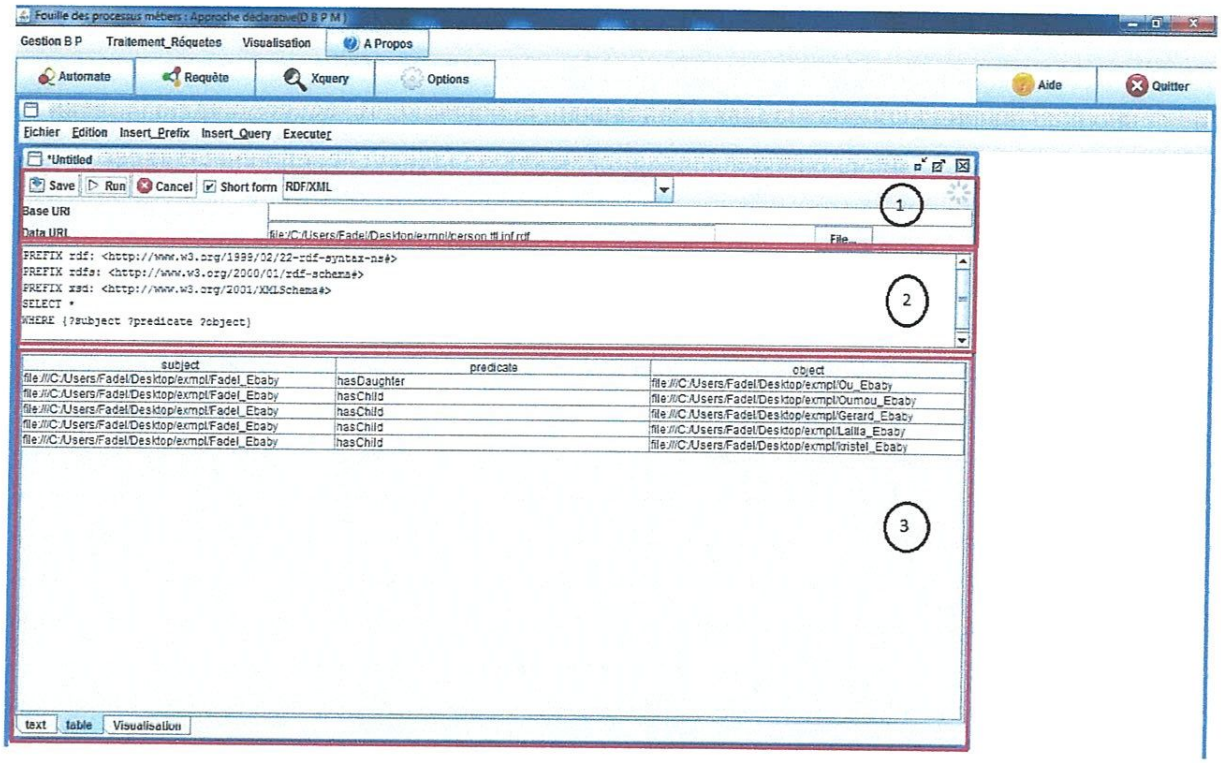


Figure 5. 6 : *Interface SPARQL et réponse*

1- Cette case favorise l'indication du chemin du protocole sur lequel la requête sera appliquée

2- C'est un panneau réservé permettant d'écrire les requêtes SPARQL

3- C'est un panneau réservé pour afficher les résultats des requêtes en plusieurs formats (Tableau, Text,..)

i. Exemple de requête en SPARQL

On cherche a afficher tout les attributs d'un ficher person.ttl.inf.rdf

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT *
WHERE {?subject ?predicate ?object}
```

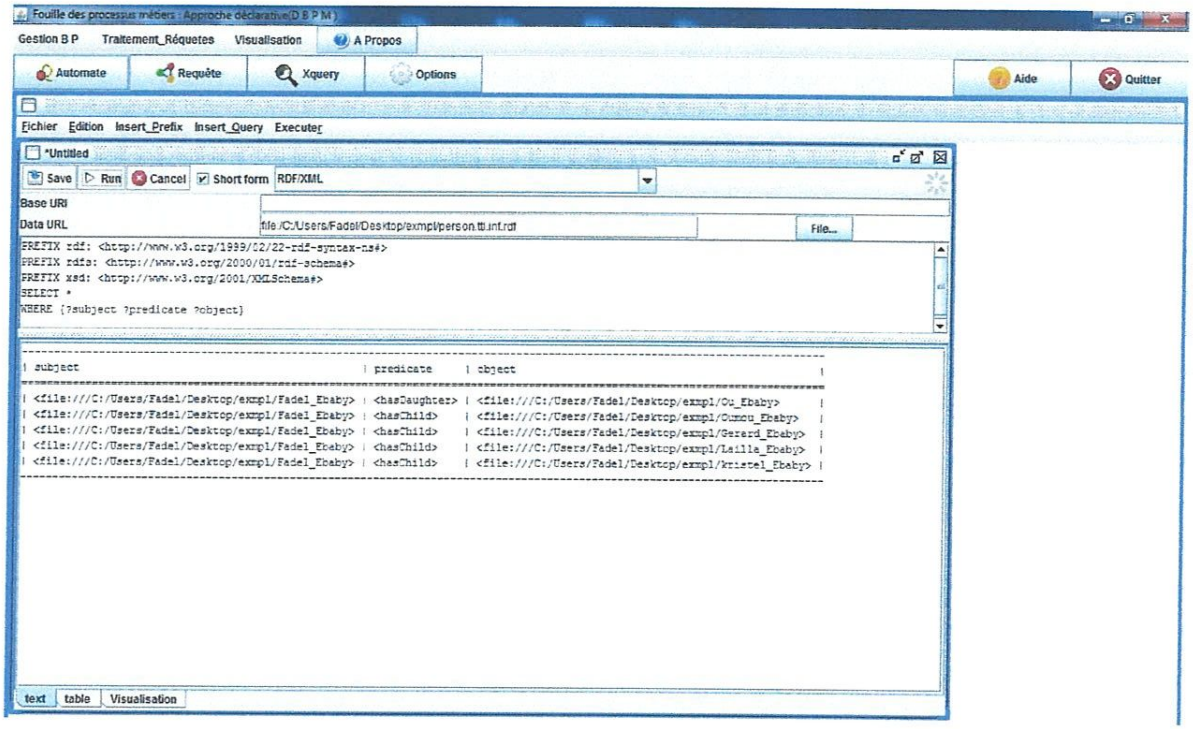


Figure 5. 7 : *exemple de requête SPARQL*

b. Les requêtes sur les traces :

Pour effectuer l'opération on commence tout d'abord par cliquer sur le bouton ci-après

ou par *Menu-> traittement_requete-> Xquery.* Puis apparaitra la figure 5.6 ci-dessous Qui va nous permettre de faire afficher les différentes traces c'est-à-dire (toutes les opérations effectuées depuis le début jusqu'à a l'heure actuelle)

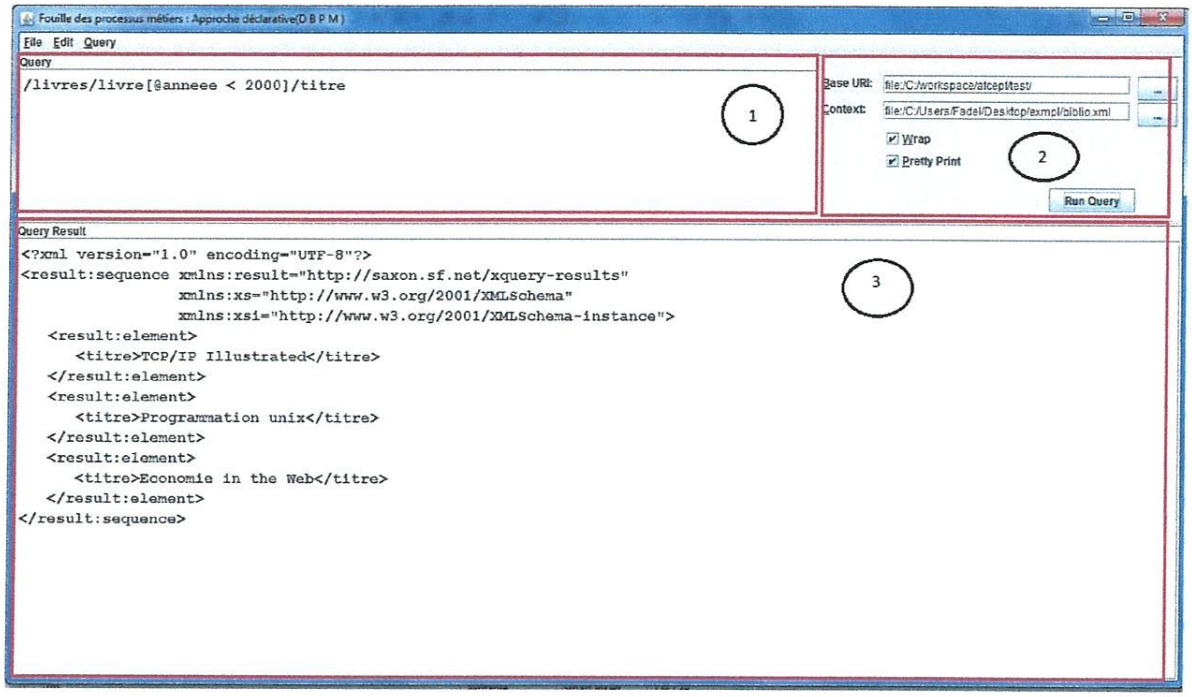


Figure 5. 8 : *Interface des traces*

1- C'est un panneau réservé permettant d'écrire les requêtes Xquery
2- Cette case favorise l'indication du chemin du protocole sur lequel la requête sera appliquée
3- C'est un panneau réservé pour afficher les résultats des requêtes

ii. Exemple des traces en Xquery

On cherche à afficher des traces des livres publier avant l'année 2000

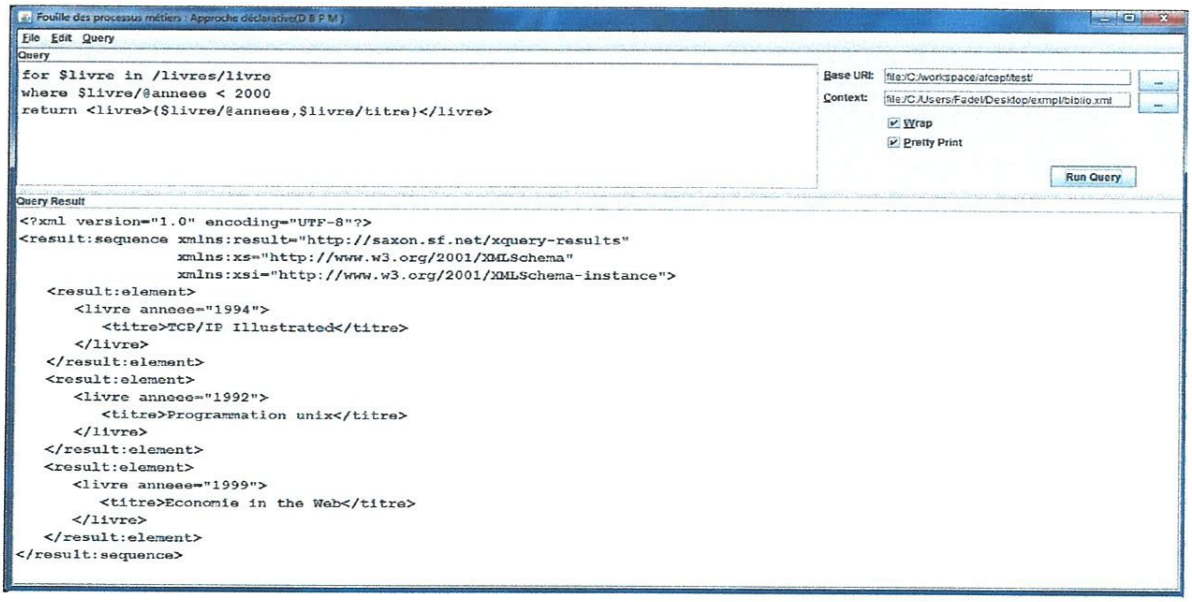


Figure 5. 9 : *exemple de trace*

Conclusion

Dans ce chapitre, nous avons mis en évidence les différents résultats lors de l'implémentation de notre processus en question tout en mettant en œuvre les outils nécessaires qui nous ont permis d'abouti a sa réalisation. Enfin des tests et des essais ont été effectués sur le système afin de vérifier sa fiabilité et sa stabilité.

Conclusion générale

Aux termes de ce travail nous pouvons dire que nous avons eu l'occasion et le cadre adéquat pour aborder une problématique assez intéressante qui est relative à la famille des processus métiers.

Nous pouvons dire que nous avons capitalisé des connaissances sur trois axes différents

1- **Sur le plan scientifique**

Nous avons étudié le domaine des services Web et nous avons appris leur fonctionnement et leur technologie.

Nous sommes actuellement conscients que les services Web sont des environnements dominant pour l'intégration des applications hétérogènes et distribuées

2- **Sur le plan technologique**

La mise en œuvre des connaissances acquises nous a permis d'aborder l'aspect pratique par l'implémentation de l'approche proposée. En ce sens, nous avions eu l'occasion de maitriser les environnements de programmation et les outils logiciels afférents a notre sujet (XQuery , SPARQL)

3- **Le plan méthodologique**

La conduite de ce projet de fin d'étude était une occasion pour nous permettant de mettre en œuvre les connaissances théoriques acquises le long dc notre formation cycle de formation (analyse, conception et implémentation) dans un cadre global de conduite de projet

En résumé, la réalisation de notre système nous a permis de manipuler de nouvelles technologies.

Comme Perspectives : nous projetons continuer sur cette lancée par la finalisation du système proposé.

Nous espérons que les futures promotions se pencheront sur ce travail pour l'exploiter et l'améliorer.

Bibliographies

[1] Objets distribués, Introduction aux systèmes d'objets distribués ISIMA *3– 1997/1998*

[2] Guide de la sécurité des systèmes d'information : CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE : Robert Longeon ; © *CNRS - 2e trimestre 1999 - ISBN 2-910986-21-7*

[3] Évaluation de Politiques d'Auto- Adaptabilité basées sur la Mobilité des Services Web Orchestrés : Afef JMAL épouse MAÂLEJ ; *30 Juillet 2009*

[4] Gestion des processus métier et travail collaboratif : Université de Technologie de troyes ; Laboratoire CNRS ISTIT – équipe Tech-CICO 12 rue Marie Curie – BP 2060 - 10010 TROYES Cedex

[5] -wikipedia : Système'information.htm

[6]Modéliser son système d'information –Sabine Bohnké : EYROLLES

[7] Organisations et systèmes d'information (SI)

[8] Conception, implantation et expérimentation d'une architecture en bus pour l'auto-préparation des applications distribuées à base de services web l'Université Toulouse III - Paul Sabatier et la Faculté des Sciences Économiques et de Gestion – Sfax : RIADH BEN HALIMA ; *Jeudi 14 Mai 2009*

[9] L'apport des SI aux outils de gestion dans les organisations étendues ? Le cas des roadmaps de management- Sébastien TRAN Ecole de Management de Normandie (EMN) 9 rue Claude Bloch 14000 Caen

[10] ÉCOLE DOCTORALE : École Doctorale Informatique Information et Société Par MAYYAD JABER Ingénieur en informatique

[11] vers une architecture n-tiers -Rémi LEBLOND

[12] Approche dirigée par les modèles pour la spécification, la vérification formelle et la mise en œuvre de services Web composés : Christophe Dumez ; *31 août 2010*

[13] Contribution à la Modélisation et à la Vérification de Processus Workflow : Zohra SBAÏ; *13 novembre 2010*

[14] Collaboration des Processus Métiers dans les Echanges inter-entreprises (B2B) basée sur le Web Service Resource Framework (WSRF) du Grid ; *FEVRIER 2008*

[15] Composants pour la Modélisation des Processus Métier en Productique, basés sur CIMOSA : ABDMOULEH Anis ; *15 septembre 2004*

[16] Business Internet Consortium (BIC) Whitepaper «High-Level Conceptual Model for B2B Integration»; *2002*

[17] Approche générique pour la modélisation et l'implémentation des processus *: vendredi 11 février 2011*

[18] Un processus d'Intégration d'Applications Intra & Inter- Entreprises : Jean-Stéphane ULMER; Melle. Soumia BENDEKKOUM ; *17 / 05 /2009*

[19] 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.*

[20] http://www.piloter.org/process-management/workflow.htm; *jeudi 6 février 2012*

[21] http://www.Tag - Processus métier.htm : *Mars 2012*

[22] contribution à l'intégration des processus métier : application à la mise en place d'un référentiel qualité multi-vues : Anis Ferchichi ; *1er Juillet 2008*

[23] Modélisation et Vérification des processus métiers dans les entreprises virtuelles : Une approche basée sur la transformation de graphes, Thèse de Doctorat en Sciences en Informatique : Raida ElMansouri ; Université Mentouri Constantine

[24] Composition de protocoles métier de services web pour le passage à l'échelle en utilisant les automates : kaouthar FAKHFAKH ; *22 juin 2006*

[25] gestion de l'évolution d'un web sémantique d'entreprise : Phuc Hiep LUONG ; *14 décembre 2007*

[26] Une Approche de transformation de la notation BPMN vers BPEL basée sur la transformation de graphe : Kholladi Mohamed Naoufel

[27] Génération d'automates observateurs dans le format du langage FIACRE : HANOI, PHAN Tran Hieu – Promotion 13 *JUILLET 2008*

[28] An Effective SPARQL Support over Relational Databases; Shanghai Jiao Tong University, Shanghai 200030, China {robert lu,yyu}@cs.sjtu.edu.cn.

[29] Interrogation à base d'annotation sémantique : Laboratoire EEDIS, UDL de Sidi Bel Abbes, ALGERIE

Bibliographies

[1] Objets distribués, Introduction aux systèmes d'objets distribués ISIMA *3–1997/1998*

[2] Guide de la sécurité des systèmes d'information : CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE : Robert Longeon ; © *CNRS - 2e trimestre 1999 - ISBN 2-910986-21-7*

[3] Évaluation de Politiques d'Auto- Adaptabilité basées sur la Mobilité des Services Web Orchestrés : Afef JMAL épouse MAÂLEJ ; *30 Juillet 2009*

[4] Gestion des processus métier et travail collaboratif : Université de Technologie de troyes ; Laboratoire CNRS ISTIT – équipe Tech-CICO 12 rue Marie Curie – BP 2060 - 10010 TROYES Cedex

[5] -wikipedia : Système'information.htm

[6]Modéliser son système d'information –Sabine Bohnké : EYROLLES

[7] Organisations et systèmes d'information (SI)

[8] Conception, implantation et expérimentation d'une architecture en bus pour l'auto-préparation des applications distribuées à base de services web l'Université Toulouse III - Paul Sabatier et la Faculté des Sciences Économiques et de Gestion – Sfax : RIADH BEN HALIMA ; *Jeudi 14 Mai 2009*

[9] L'apport des SI aux outils de gestion dans les organisations étendues ? Le cas des roadmaps de management- Sébastien TRAN Ecole de Management de Normandie (EMN) 9 rue Claude Bloch 14000 Caen

[10] ÉCOLE DOCTORALE : École Doctorale Informatique Information et Société Par MAYYAD JABER Ingénieur en informatique

[11] vers une architecture n-tiers -Rémi LEBLOND

[12] Approche dirigée par les modèles pour la spécification, la vérification formelle et la mise en œuvre de services Web composés : Christophe Dumez ; *31 août 2010*

[13] Contribution à la Modélisation et à la Vérification de Processus Workflow : Zohra SBAÏ; *13 novembre 2010*

[14] Collaboration des Processus Métiers dans les Echanges inter-entreprises (B2B) basée sur le Web Service Resource Framework (WSRF) du Grid ; *FEVRIER 2008*

[15] Composants pour la Modélisation des Processus Métier en Productique, basés sur CIMOSA : ABDMOULEH Anis ; *15 septembre 2004*

[16] Business Internet Consortium (BIC) Whitepaper «High-Level Conceptual Model for B2B Integration»; *2002*

[17] Approche générique pour la modélisation et l'implémentation des processus *: vendredi 11 février 2011*

[18] Un processus d'Intégration d'Applications Intra & Inter- Entreprises : Jean-Stéphane ULMER; Melle. Soumia BENDEKKOUM ; *17 / 05 /2009*

[19] 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.*

[20] http://www.piloter.org/process-management/workflow.htm; *jeudi 6 février 2012*

[21] http://www.Tag - Processus métier.htm : *Mars 2012*

[22] contribution à l'intégration des processus métier : application à la mise en place d'un référentiel qualité multi-vues : Anis Ferchichi ; *1er Juillet 2008*

[23] Modélisation et Vérification des processus métiers dans les entreprises virtuelles : Une approche basée sur la transformation de graphes, Thèse de Doctorat en Sciences en Informatique : Raida ElMansouri ; Université Mentouri Constantine

[24] Composition de protocoles métier de services web pour le passage à l'échelle en utilisant les automates : kaouthar FAKHFAKH ; *22 juin 2006*

[25] gestion de l'évolution d'un web sémantique d'entreprise : Phuc Hiep LUONG ; *14 décembre 2007*

[26] Une Approche de transformation de la notation BPMN vers BPEL basée sur la transformation de graphe : Kholladi Mohamed Naoufel

[27] Génération d'automates observateurs dans le format du langage FIACRE : HANOI, PHAN Tran Hieu – Promotion 13 *JUILLET 2008*

[28] An Effective SPARQL Support over Relational Databases; Shanghai Jiao Tong University, Shanghai 200030, China {robert lu,yyu}@cs.sjtu.edu.cn.

[29] Interrogation à base d'annotation sémantique : Laboratoire EEDIS, UDL de Sidi Bel Abbes, ALGERIE