الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

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

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Ingénierie des Medias

Thème :

Extraction des patterns à partir de la spécification des protocoles de services

Encadré Par :

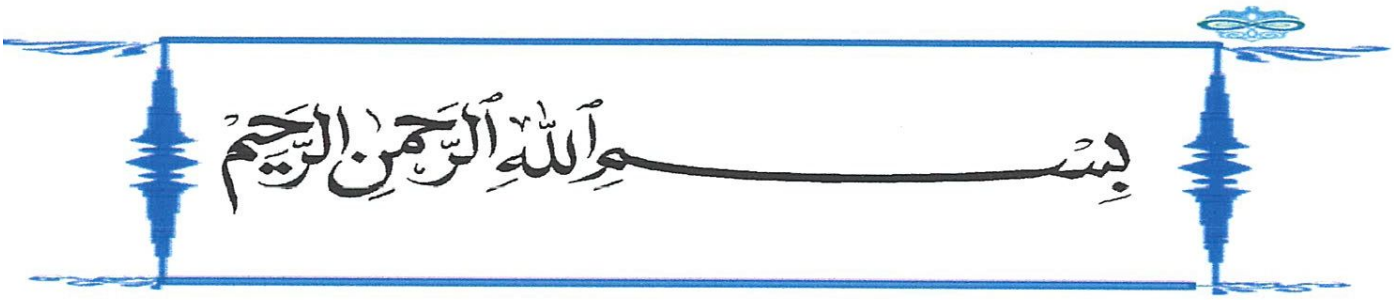
Mr. KHEBIZI Ali

Présenté par :

Kenouz Saliha

Azzouz Samah

Juin 2012


بسم الله الرحمن الرحيم

Remerciement

Nous remercions Dieu le tout puissant de nous avoir accordé cette chance, de nous prodiguer la force et la patience pour achever ce travail ;

Nos vifs sincères remerciements à la personne qu'on admire et on respecte le plus : notre encadreur MONSIEUR KHBIZI ALI qui nous à fait le grand honneur d'accepter de nous guider dans ce modeste travail par ces conseils et sa collaboration afin de mener bien à ce mémoire ;

Avec une très grande gratitude nous remercions nos parents pour leurs support et prières pour qu'on puisse arriver ici ;

Nous tenons à remercier tous les enseignants qui nous ont suivis durant notre formation.

Merci e à tous…

Dédier à notre petit ange **Ghofrane**


Résumé

Résumé

Les services Web offrent de fortes possibilités pour l'intégration des systèmes d'information hétérogène et distribués. Ils sont basés sur la standardisation des divers protocoles d'échanges.

La pile de protocoles les constituant permet de répondre aux différents besoins des utilisateurs et des applications (publication, découverte, invocation).

Dans ce mémoire nous allons nous intéressé à l'aspect évolution des protocoles des services Web et nous aborderons la spécification et l'extraction des attributs caractérisant l'évolution.

Nous avons proposé une analyse de l'impact de l'évolution sur les instances en cours d'exécution.

Par ailleurs, nous avons implémenté un outil logiciel pour concrétiser notre approche et ce par la création d'une base de données des protocoles et une autre pour les instances actives. Différents fonctionnalités répondant aux besoins des gestionnaires de protocoles ont été implémentées.

Mots Clés : Services Web, Processus métiers, Patterns, Migration des traces, Système d'information, Automate, Evolution de protocole.

Sommaire

Sommaire

Sommaire

Sommaire

PARTIE II : CENCEPTION ET IMPLEMENTATION

Chapitre. IV. Analyse et conception

Chapitre. V. Implémentation

Sommaire

Liste des figures

Liste de figures:

Liste des figures

Liste des tableaux

Liste des tableaux :

Liste des Abréviations et Acronymes

Acronymes	*Désignations*
AEF	Automates d'états finis
AFD	Automates d'états finis déterministe
API	Application Programming Interface
BDD	Base De Données
BP	Bisness Process
BPEL	Business Process Execution Language
BPEL-Abstract	Abstract Business Process Execution Language
CORBA	Commune Object Request Broker Architecture
CRM	Customer Relationship Management
DCOM	Distributed Component Object Model
DTD	Document Type Definition
EDI	Electronic Data Interchange
ERP	Enterprise Ressources Planning
HTTP	Hypertext Transfer Protocol Secure
IBM	International Business Machines
IDE	Integrated development environment
IDL	Interactive Data Language
IP	Internet Protocol
MAJ	Mise à jour
MCT	Le modèle conceptuel des traitements de MERISE
MOM	Message-Oriented Middleware
OWL_S	Ontology Web Language for Services
PDA	Personal digital assistant
QoS	Quality of service
RdP	Réseau de Petri
RPC	Remote Procedure Call
SCM	Supply Chain Management
SGBD	Système de gestion de base de données
SGML	Quality of service
SI	Système d'Information
SID	Systèmes d'information distribués
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SW	Web services
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier

Liste des Abréviations et Acronymes

URL	Uniform Resource Locator
VTS	Visual Timed Event Scenarios
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XML	Extensible MarckupLanguage

Introduction générale

Introduction générale

Les réseaux informatiques ont aujourd'hui tellement d'importance au point que la plupart de nos activités ne pourraient plus être envisagées sans la mise en place de ces réseaux. On assiste à leur déploiement à tous les niveaux de la société, dans les entreprises, au niveau national et international, y compris dans les domiciles des usagers.

Le réseau n'est pas une entité statique. Il subit des évolutions qui a mis en jeu bien des aspects technologiques. On peut même parler de mutation au vu des progrès fulgurants qu'il a enregistrés depuis ces débuts.

L'évolution de réseau dont nous venons de parler pose un problème majeur : la manière dont ceux-ci sont pratiquement gérée .A cette difficulté s'ajoute aussi le problème d'hétérogénéité des technologies sous-jacentes et celle d'offrir une vue unifiée du réseau à l'administrateur et, d'autre part, la distribution et la grande quantité d'informations à collecter et à traiter.

Les services Web constituent la solution à ces problèmes puisque ils sont fondus sur le principe de standardisation et offre la possibilité d'interopérabilité entre des environnements hétérogènes et distribués.

Pour l'élaboration de ce présent travail, nous avons procédé d'abord par une lecture approfondie des ouvrages de référence en la matière en vue de bien assimiler les concepts théoriques de base qui régissent la modélisation des protocole de services Web et la programmation en java.

Ce travail comporte 5 chapitres brièvement décrits comme suivent :

- Le chapitre 1 est une introduction aux concepts de base de système d'information distribuée ; Le chapitre 2 décrit les services Web. Il explique le fonctionnement, les différentes composantes d'une architecture d'un service Web et montre les échanges entre ces composants.
- Le chapitre 3 présente les processus métier, on détaillera leurs intérêts et la manière dont ils sont modélisés avec une comparaison entre ces différents types de modélisations.
- Le chapitre 4 montre les outils de conception de notre système et les étapes de cette conception. L'architecture du système conçu est y présentée.
- Le chapitre 5 traite l'implémentation de l'application en java. Il montre pas à pas l'exploitation de notre application d'extraction de patterns d'évolution et de gestion des instances en cours d'exécution.

Chapitre I :

Les systèmes d'information distribués

Chapitre I : Les systèmes d'information distribués

Introduction :

Les systèmes d'information sont censés aider les utilisateurs dans leurs activités : stocker et restaurer l'information, faire des calculs, permettre une communication efficace, ordonnancer et contrôler des tâches, etc.

La distribution et l'accès à l'information sont des facteurs fondamentaux de succès dans un système d'information qui est par nature distribué, évolutive et varié et pour cela on a eu besoin de nouvelles architectures présenté sous forme de systèmes distribués.

Dans ce premier chapitre on va parler des systèmes d'informations en général et leurs conceptions, leurs architectures et on va détailler l'intérêt et les objectifs d'un système distribué et les concepts associés aux middlewares pour gérer l'hétérogénéité dans les systèmes d'informations issus de partenaires différents.

1. Définition du Système d'Information :

Un système d'information est un ensemble d'éléments qui recueillent de l'information, la traitent, la stockent et la diffusent afin d'aider à la prise de décision, à la coordination, et au contrôle des activités d'une entreprise.

Le système d'information est utilisé par des acteurs (utilisateurs, administrateurs, managers) pour manipuler (consulter, modifier, communiquer) des données à l'aide de procédures.

Dans le système d'information, la production de l'information nécessaire à l'organisation se compose de trois activités [1]

- **L'entrée**: arrivée des données brutes (saisie d'un utilisateur, provenance d'un autre système). Par exemple, la commande d'un client.

- **Le traitement des données**: transformation des données brutes afin qu'elles puissent être compréhensibles par l'utilisateur. Cette transformation consiste à appliquer un ensemble de règles de gestion.

Exemple :

Stocke existant=quantité+entrée - sortis.

- **La sortie**: processus de diffusion de l'information traitée aux utilisateurs qui en ont besoin. Par exemple, un utilisateur souhaite consulter l'état des stocks. [1]

Le périmètre du terme « **système d'information** » peut être très différent d'une organisation a une autre et peut recouvrir selon les cas tout ou partie des éléments suivants :

- base de données de l'entreprise.
- progiciel de gestion intégré : Enterprise Ressources Planning (ERP).

- Outil de gestion de la relation client (Customer Relationship Management) : CRM.
- Outil de gestion e la chaîne logistique (SCM - Supply Chain Management).
- Applications métiers. (paie, comptabilité, stock, inventaires, etc.).
- Infrastructure réseau.
- Serveurs de données et systèmes de stockage,
- Serveurs d'application.
- Dispositifs de sécurité.

2. Conception d'un système d'information :

2.1. Les couches d'un système d'information :

La représentation en couche permet de faciliter la conception et le déploiement du système d'information.

Les systèmes d'informations sont conçus de trois couches : couche présentation, couche logique d'application et couche gestionnaire de ressources. **(Fig. 1 .1)**

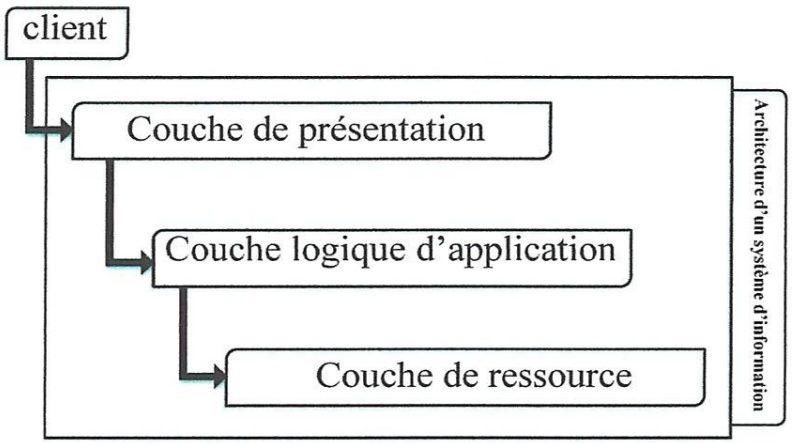


Figure 1.1 *: couche d'un système d'information.*

a- Couche présentation :

Elle peut être implémentée en tant qu'interface graphique d'utilisateur ou sous forme de module représentant les données sous forme syntaxique.

Elle sert a assuré la communication entre le client et le système par la soumission de requêtes et l'obtention des réponses.

b- Couche logique d'application :

C'est les processus métiers, exécuté sous forme de programmes qui implémentent les opérations demandés par le client dans la couche présentation.

c - Couche gestionnaire de ressources (Resource Management Layer) :

Elle traite et met en œuvre les différentes sources des données pour le système d'information sans prendre en compte la nature de ces sources qui peuvent êtres une base de données ou un autre système d'information.

2.2. Les stratégies de conception d'un système d'information :

a- Approche descendante (Top Down)

Son principe est de définir les fonctionnalités du système selon le point de vue du client. Dans cette stratégie la conception débute par la couche présentation, décent à la partie logique métier et se termine par la gestion des ressources. [2]

b- Approche ascendante (Bottom Up)

Elle est utilisée dans la manipulation de fonctionnalités des systèmes déjà existant pour les intégrés en un nouveau ensemble cohérent dans un contexte diffèrent.

Dans le tableau ci-dessous on expose les différences entre les étapes de processus de conception selon les deux stratégies présentées précédemment :

	Top Down	**Bottom Up**
Client	Définir les panneaux d'accès et la plateforme client	Définir les panneaux d'accès et la plateforme client
Couche de présentation	Définir des formats de présentation et les protocoles pour les clients choisis	Examiner les ressources existant et les fonctionnalités qu'elles offrent.
Couche logique d'application	Définir les fonctionnalités nécessaire pour fournir les contenues et les formats à la couche de présentation	Envelopper les ressources existantes et intégrer leurs fonctionnalités dans une interface cohérente.
Gestionnaire de ressource	Définir les sources et l'organisation nécessaire des données pour implémenter l'application logique	Adapter la sortie de la logique d'application de sorte qu'il peut être utilisé avec des canaux d'accès requis et le protocole du client.

Tableau 1.1 *: Comparaison des approches de conception des systèmes d'information.*

3. Architectures d'un système d'information :

On distingue différentes architectures des systèmes d'information, selon leur infrastructure matérielle et son organisation.

3.1. Architecture 1-Tiers : Mainframe

Ce type d'architecture est appelée *Mainframe*. Historiquement, les applications sur mainframe ont été les premières à proposer un accès multiutilisateurs. Dans ce contexte, les clients se connectent aux applications exécutées par le serveur central à l'aide des terminaux se comportant en esclaves.

La couche de présentation, logique d'application et gestionnaire de ressources sont construites comme une entité monolithique. **(fig.1.2)**

C'est le serveur central qui prend en charge l'intégralité des traitements, y compris l'affichage qui est simplement déporté sur des terminaux.

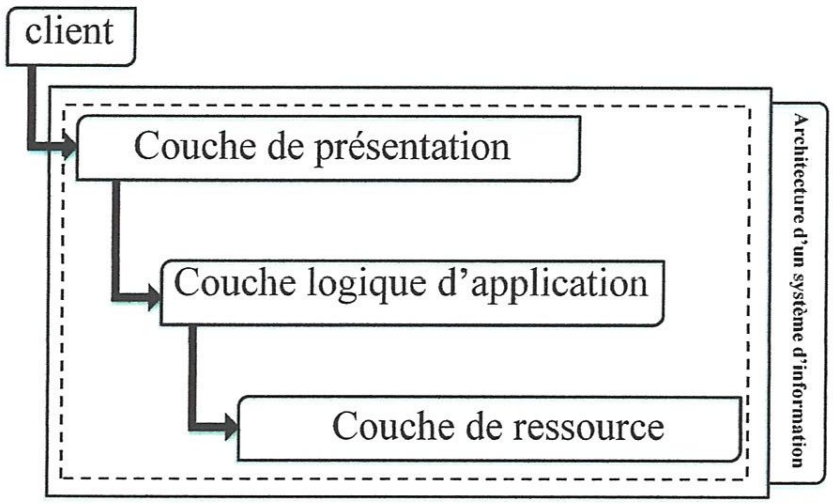


***Figure 1.2** : Architecture mainframe.*

Evaluation :

Avantage :

- Une facilité d'administration dûe au fait que toutes les couches applicatives sont centralisées uniquement sur un seul nœud (le serveur).
- L'installation des éléments matériels revient moins chère.
- Une maintenance centralisée : toutes les ressources logicielles et les données étant mobilisées uniquement sur la machine centrale, un cas de maintenance applicative ne porte que sur elle.

Inconvénients :

- Une fragilité du serveur : le serveur est le seul élément actif de cette architecture. il est donc le seul à gérer les différents services.
- Des postes non autonomes : les postes clients se comportent en esclave par rapport au serveur. ils ne disposent d'aucun pouvoir de décision.
- Le coût excessif des logiciels sur mainframe.

3.2. Architecture 2-tiers : Client / Serveur

De nombreuses applications fonctionnent selon un environnement client/serveur, cela signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en termes de capacités d'entrée-sortie, qui leur fournit des services.

Les services sont exploités par des programmes, appelés programmes clients, s'exécutant sur les machines clientes, capable de traiter des informations qu'il récupère auprès d'un serveur. **(fig.1.2)**

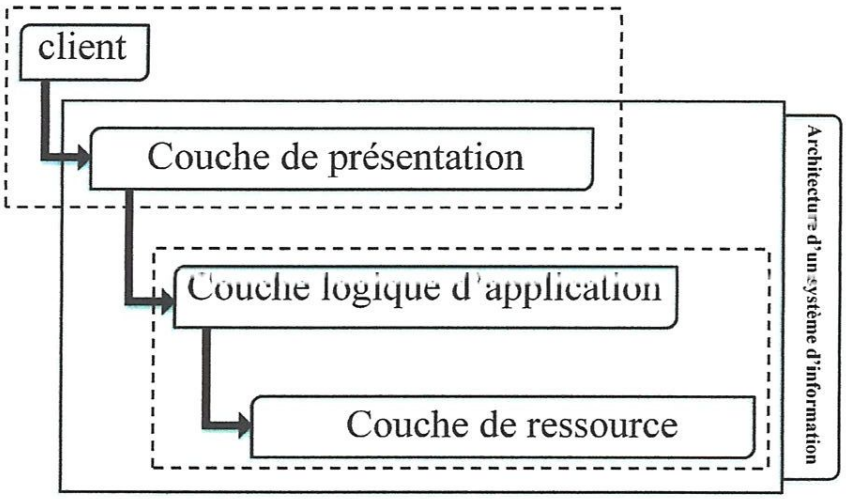


Figure 1.3 : *Architecture Client / Serveur.*

Evaluation :

Avantage :

- Une bonne sécurité : car le nombre de points d'entrée permettant l'accès aux données est moins important. aucune donnée n'est stockée sur le poste client.
- Un réseau évolutif : grâce à cette architecture, on peut supprimer ou rajouter des postes clients sans perturber le fonctionnement du réseau et sans modifications majeures.
- Offre une facilité d'intégration des ressources existantes et réduit le coût de déploiement des systèmes.

- Permet de répartir les traitements sur différents processeurs et minimiser le trafic sur les réseaux de communication.

Inconvénients :

- Un maillon faible : le serveur est le seul maillon faible de ces systèmes, étant donné que tout le réseau est architecturé autour de lui.
- Les risques sont croissants dus au fait que les postes clients du système peuvent être facilement infectés par des virus.
- Coût de déploiement et de maintenance.
- Problème d'administration réseau.

3.3. Architecture 3-tiers : Les middlewares

Dans un environnement client-serveur trois tiers, on a : le client « dit client léger», un serveur d'application (appelé aussi middleware) et un serveur de données tous distincts.

Cette architecture fonctionne selon trois niveaux :

· Premier niveau : l'affichage et les traitements locaux (contrôles de saisie, mise en forme des données,...) sont pris en charge sur le poste client.

· Deuxième niveau : les traitements applicatifs globaux sont pris en charge par le service applicatif du serveur.

· Troisième niveau : les services de base de données sont pris en charge par un SGBD (voir serveur de base de données). **(fig.1.4)**

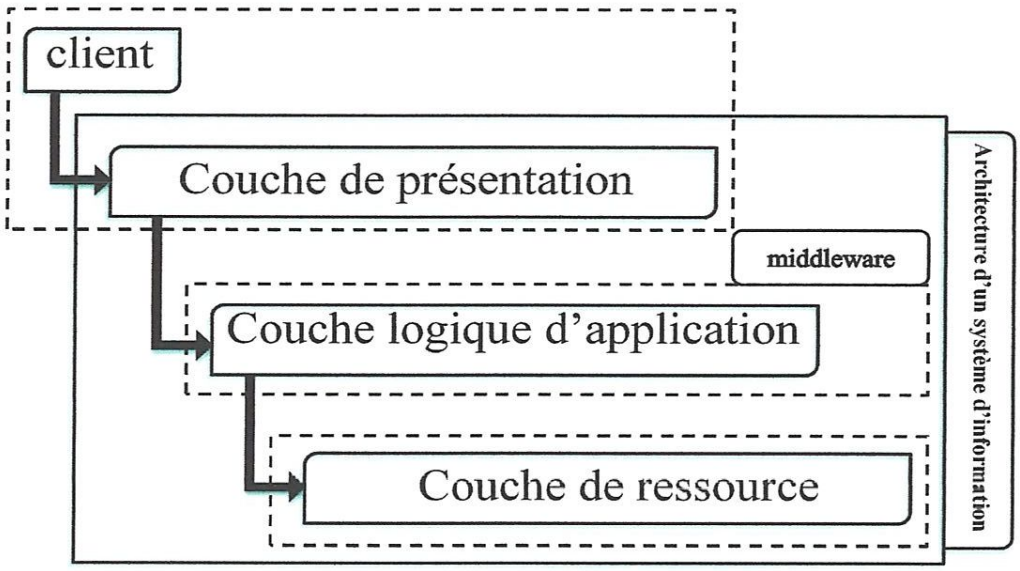


Figure 1.4 : *Architecture des middlewares.*

Evaluation :

Avantage :

- Les applications serveurs sont délocalisées, c'est à dire que chaque serveur est spécialisé dans sa tâche.
- Le poste client ne supporte plus l'ensemble des traitements, il est moins sollicité et peut être moins évolué, donc moins coûteux.
- La fiabilité et les performances de certains traitements se trouvent améliorées par leur centralisation, il est relativement simple de faire face à une forte montée en charge en renforçant le service applicatif
- Une plus grande sécurité : compte tenu du fait que l'on peut définir la sécurité pour chaque service.
- De bonnes performances : due au fait que les tâches sont partagées entre plusieurs serveurs.

Inconvénients :

- Le serveur se trouve souvent fortement sollicité et il est difficile de répartir la charge entre client et serveur.
- Les solutions mises en œuvre sont relativement complexes à maintenir et la gestion des sessions compliquée
- Les contraintes semblent inversées par rapport à celles rencontrées avec les architectures deux tiers : le client est soulagé, mais le serveur est fortement sollicité.

3.4. Architecture N-Tiers : Application Web

C'est la généralisation de l'architecture 3-tiers on utilisant l'internet comme canal d'accès. Cette architecture apparaissent dans deux séances génériques: liaison de systèmes différents et en ajoutant de la connectivité à travers l'Internet.

Dans cette architecture la couche de gestionnaire de ressource ne contient pas seulement des simples ressources comme les bases de données mais aussi des systèmes à deux, trois et multiples tiers. **(Fig. 1.5)**

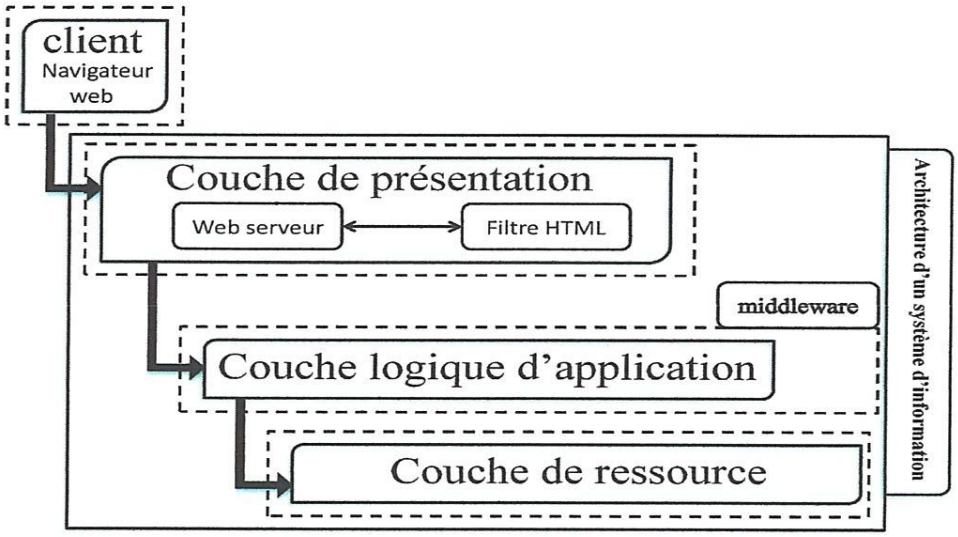


Figure 1.5 : *Architecture N-tiers.*

Evaluation :

Avantage :

- Une meilleure évolutivité.
- Augmentation de la tolérance aux pannes.

Inconvénients :

- Trop de middlewares impliqués, souvent avec des fonctionnalités redondantes.
- les coûts de développement, de maintenance et d'évolution de ces systèmes augmentent presque exponentiellement avec le nombre de tiers. [2]

4. Les systèmes d'information distribués : (SID).

4.1. Définitions d'un SID :

Définition1 : Un système distribué est un ensemble d'entités autonomes de calcul (ordinateurs, PDA, processeurs, processus, processus léger etc.) interconnectées et qui peuvent communiquer. Exemples: un réseau physique de machines ou un logiciel avec plusieurs processus sur une même machine.

Définition 2 : Un ensemble d'ordinateurs indépendants qui apparaît à un utilisateur comme un système unique et cohérent ou les machines sont autonomes et les utilisateurs ont l'impression d'utiliser un seul système.

Définition 3 : Un système réparti est un système qui vous empêche de travailler quand une machine dont vous n'avez jamais entendu parler tombe en panne !!

Définition 4 : un système d'information distribué est un ensemble de ressources :

- Logicielles : logicielles application, etc.

- Base de données.

- Ressources humaines : utilisateurs, informaticiens, gestionnaires…

-ressources matérielles : machines, équipements réseaux…

Qui peuvent être géographiquement dispersées (plusieurs sites).

Nous retiendrons la définition la définition 4qui s'avère plus clair et plus pertinente dans notre contexte. [3]

4.2. Intérêt des systèmes distribués :

- Mise en commun d'un grand nombre de ressources à faible coûts :
  – Bon rapport performance/prix.
  – Puissance globale virtuellement illimitée, supérieure à celle des gros calculateurs.
- Disponibilité et flexibilité :
  – Un élément peut tomber en panne sans bloquer tout le système.
  – Distribution de la charge.
- Réponse à la distribution géographique de certains systèmes existants :
  – Réplication locale des ressources globales.
- Partage de ressources coûteuses entre plusieurs utilisateurs/machines :
  – Accès à distance à une ressource indisponible en local.
  – Accès aux mêmes ressources depuis tous les points d'accès du système. [4]

4.3. Modélisation d'un système distribué :

Dans le schéma ci-dessous on présente la démarche de modélisation d'un système distribué avec tous les modules impliqués dans ce modèle. **(fig.1.6)** [4]

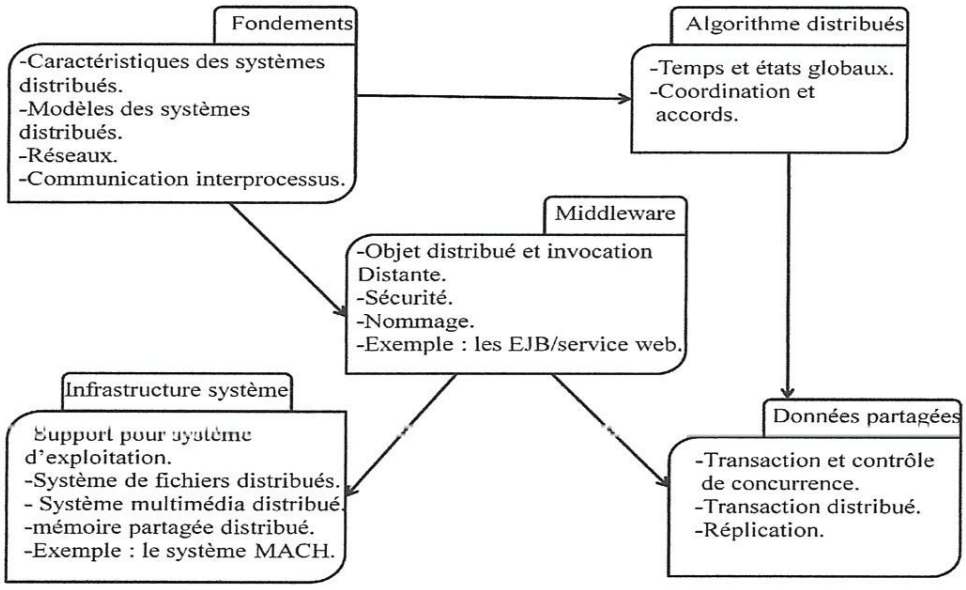


Figure 1.6 : *Démarche de modélisation des SID.*

4.4. Objectifs d'un système distribué :

Les SID visent à atteindre les objectifs suivants :

a- Transparence (Masquer la répartition)

– Uniformité des accès locaux et distants : La séparation physique entre machines et les différences matériels/logiciels pour les accès sont invisibles par l'utilisateur.

– Localisation des ressources non perceptible.

– Migration des ressources possible sans interférence avec la localisation physique (ex. transférer un objet uniquement par son nom logique sans modification de ce nom et sans modification de l'environnement d'un utilisateur).

– Réplication de ressources non visible.

– Concurrence d'accès aux ressources non perceptible (ex. accès à un même fichier ou une table dans une base de données: mécanisme de verrou ou de transaction).

- Invisibilité du parallélisme offert par l'environnement d'exécution.

- Tolérance aux pannes permettant à un utilisateur de ne pas s'interrompre (ou même se rendre compte) à cause d'une panne d'une ressource. [5]

b- Ouverture :

– Services offerts selon des règles standards qui décrivent la syntaxe et la sémantique de ces services (Interfaces publiées, ex. IDL)

– Interopérabilité des matériels (de fournisseurs différents)

– Portabilité

– Flexibilité (facilité d'utilisation et de configuration)

– Extensibilité (ajout/MAJ de composants sans en affecter les autres)

c- Mise à l'échelle (scalability) :

– fonctionne efficacement dans différentes échelles:

– Des postes de travail et un serveur de fichiers

– Réseau local avec plusieurs centaines de postes de travail et serveurs de fichiers

– Plusieurs réseaux locaux reliés pour former un Intranet.

d- Sécurité :

– Confidentialité (authentification)

– Intégrité (protection contre les falsifications et les corruptions)

– Disponibilité (accès aux ressources)

e- **Tolérance aux pannes :**

– Pannes franches

– Pannes byzantines

– Détection de pannes (difficulté et même impossibilité de détection pour certains systèmes, suspicion de machines)

– Correction d'erreurs (de fichiers/messages corrompus)

– Reprise sur pannes (techniques de journalisation dans les BDD) (Éventuellement système dégradé).

5. Les middlewares conventionnels :

5.1. Définition et rôle des middlewares :

Les Middleware sont une classe de technologies logicielles conçues pour aider à gérer la complexité et l'hétérogénéité inhérente à des systèmes distribués. Elle est définie comme une couche de logiciel au-dessus du système d'exploitation, mais inférieur au programme d'application qui fournit une abstraction de programmation commune à travers un système distribué.

Ce faisant, il fournit un bloc de construction de niveau supérieur pour les programmeurs que les interfaces de programmation d'applications (API) telles que les sockets qui sont fournies par le système d'exploitation.

Cela réduit considérablement la charge pesant sur les programmeurs d'applications en les soulageant de ce type de programmation fastidieuse et source d'erreurs.

Un Middleware est parfois officieusement appelé «Plomberie», car il relie les parties d'une application distribuée avec des tuyaux de données, puis transmet les données entre eux. **(fig.1.7)** [6]

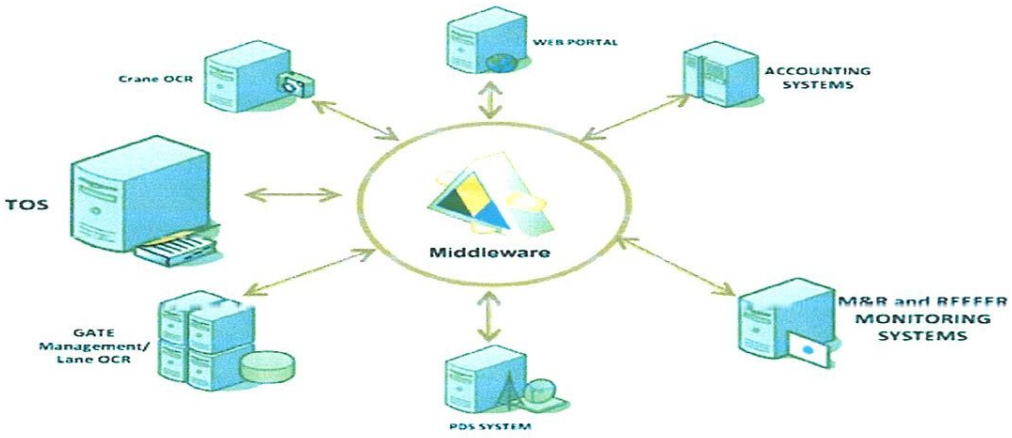


Figure 1.7 : *Position d'un Middleware dans un SID.*

5.2. Historique d'évolution des middlewares conventionnels :

Historiquement, la technologie des middlewares à connu une évolution croissante. On distingue :

a. RPC : Remote Procedure Call(RPC) : est un protocole que l'on programme peut utiliser pour demander un service à partir d'un programme situé dans un autre ordinateur dans un réseau sans avoir à comprendre les détails du réseau. (Un appel de procédure est aussi parfois connu sous le nom un appel de fonction ou d'un appel de routine.) RPC utilise le modèle client/serveur.

b. MOM : Message-Oriented Middleware (MOM) est une infrastructure logicielle centrée sur l'envoi et la réception de messages entre systèmes distribués.MOM permet aux modules d'application pour être distribué sur les plates-formes hétérogènes, et réduit la complexité du développement d'applications qui couvrent plusieurs systèmes d'exploitation et protocoles réseau en isolant le développeur de l'application à partir des détails du système d'exploitation différents et des interfaces réseau

c. CORBA : Commune Object Request Broker Architecture, Une spécification Object Management Group, qui fournit une interface standard de messagerie entre les objets distribués. Son architecture est respectée dans des composants client et serveur qui communiquent à travers une Object Request Broker (ORB). Lorsqu'un ORB détermine que la requête est à un objet distant, il peut exécuter la demande en communiquant avec l'ORB distance.

d. DCOM : C'est une interface ActiveX destinée aux applications réseau de composants distribués. Le modèle DCOM offre également une transparence au niveau du réseau et une automatisation des communications, de sorte que les communications peuvent intervenir entre des objets sans qu'un objet connaisse nécessairement l'emplacement de l'autre objet. Les objets peuvent se situer dans des processus différents sur la même machine ou sur des machines distinctes.

Cependant, les services Web sont la nouvelle technologie d'intégration.

Conclusion :

Faire interagir des programmes différents en réseau a toujours été une affaire complexe, notamment si on souhaite standardiser certains aspects de cette interaction.

Plusieurs travaux ont été réalisés pour permettre une intégration cohérente entre les différents environnements hétérogènes et distribués.

Avec l'évolution rapide des besoins de plus de connectivité entre systèmes, les solutions réaliser non pas pus couvrir touts les alternatives, mais les services Web ont offert cette possibilité.

Dans le prochain chapitre, on va faire la lumière sur cette nouvelle technologie émergente permettant d'atteindre des SID intégrés.

Chapitre II :

Les Services Web

Introduction :

Les dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et tout particulièrement par la démocratisation de l'accès à Internet.
Cette évolution du monde informatique a entrainé le développement de nouveaux paradigmes d'interaction entre applications. Notamment, l'architecture orientée service (Service-Oriented Architecture, ou SOA) a été mise en avant afin de permettre des interactions entre applications distantes. Cette architecture est construite autour de la notion de service, qui est matérialisée par un composant logiciel assurant une fonctionnalité particulière et accessible via son interface.
Dans ce chapitre, on va entamer la notion de services Web : leur apparition, leurs objectifs et leur fonctionnement. Après on détaillera les technologies utilisées par ces services et leur structuration.

1. Présentation des Services Web :

1.1. Historique et fondement des services Web :

L'histoire commence en 1975 lorsque l'informatique souffrait encore de peu de standardisation et que les constructeurs et éditeurs se rendent compte de la nécessité d'uniformiser les échanges de données. Ils firent alors le vœu pieux de l'interopérabilité afin de standardiser la communication entre application au travers d'un réseau. C'est la naissance l'EDI (Electronic Data Interchange ou Echange de Données Informatisées), l'ancêtre des Web Services.
L'émergence du Web à tout bouleversé, elle a passé par la création du protocole HTTP et sa standardisation en 1997 comme protocole de transfert de données hyper texte et qui s'appuie sur TCP/IP.
Il ne manquait alors qu'un format d'encapsulation et d'échange de message pour finaliser la chaîne de l'interopérabilité, et la en 1998, année de sa standardisation en version 1.0 par le W3C, XML viendra compléter la chaîne de l'interopérabilité en s'imposant comme format d'échange et de description des données, indépendant de toute plateforme. [8]

1.2. Définition des services Web :

En terme général, les services Web sont des composants d'applications distribuées qui se conforment à des standards permettant de les rendre publique, et de résoudre les problèmes d'interopérabilité entre les applications hétérogènes.
Une définition plus précise est fournie par le W3C :

« Un service Web est un système logiciel destiné à supporter l'interaction ordinateur-ordinateur sur le réseau. Il a une interface décrite en un format traitable par l'ordinateur (Particulièrement WSDL). Autres systèmes réagissent réciproquement avec le service Web d'une façon prescrite par sa description en utilisant des messages SOAP, typiquement transmis avec le protocole HTTP et une sérialisation XML, en conjonction avec d'autres standards relatifs au Web». [9]

En d'autres termes, les services Web sont des applications auto descriptives, modulaires et faiblement couplées qui fournissent un modèle de programmation et de déploiement d'applications, basé sur des normes, et s'exécutant au travers de l'infrastructure Web.

On déduit que 'un service Web est un composant implémenté dans n'importe quel langage, déployé sur n'importe quelle plateforme, afin qu'il soit recherché et invoqué par d'autres services grâce à des standards (SOAP, WSDL et UDDI).

En d'autre terme : Un service Web est une unité de logique d'application qui fournit des données, et des services aux autres applications, les applications accèdent aux services Web via des protocoles, et des formats de données omniprésentes telles que HTTP, XML et SOAP.

1.3. But des services Web : les objectifs des services Web sont :

- Fournir une architecture générale pour les applications réparties sur internet : Ils permettent d'interconnecter différentes entreprises, matériels, applications et clients.
- la réutilisabilité : Les services Web peuvent être réutilisés dans différentes applications internes utilisés par plusieurs clients pour réaliser des fonctions spécifiques pour cela le niveau de granularité (faible, moyen, fort) du service est un facteur déterminant pour la réutilisabilité.
- Interopérables : basé sur des standards ouverts sans composant spécifique à un langage ou un système d'exploitation.
- Couplage faible : limiter au maximum les contraintes imposées sur le modèle de programmation des différents éléments de l'application par exemple ne pas imposer un modèle objet.
- Supportant la montée en charge : par exemple en n'imposant pas un modèle de type RPC.
- L'automatisation des processus métiers : Les services Web permettent d'intégrer, gérer et automatiser rapidement les processus métier intra et interentreprises en composant des services élémentaires et en échangeant des informations au format XML. [12]

2. Architecture et fonctionnement des services Web :

2.1. Les acteurs :

Trois acteurs réagissent dans cette architecture :

a. Le fournisseur de service (service provider):C'est lui qui définit le service, publie sa description dans l'annuaire de service contenant les informations commerciales, les services offerts ainsi que les informations nécessaires à l'invocation du service sous un format préalablement défini. D'un point de vue technique, il est constitué par la plate-forme d'accueil du service.

b. L'annuaire (Registry) : Reçoit et enregistre les descriptions de services publiées par les fournisseurs, reçoit et répond aux recherches de services lancées par les clients.

Un des buts initial des annuaires était d'offrir une place de marché universelle où chaque entreprise, quels que soient sa taille, son domaine d'activité et sa localisation, pourrait publier les services dont elle dispose, et trouver de façon dynamique les services dont elle a besoin.

c. Le client (service requestor) :C'est le consommateur de service. Il obtient la description du service grâce à l'annuaire après l'avoir sélectionné en interagissant directement avec le fournisseur. D'un point de vue technique, il est constitué par une application (ou un utilisateur) ou même un autre service Web. [10]

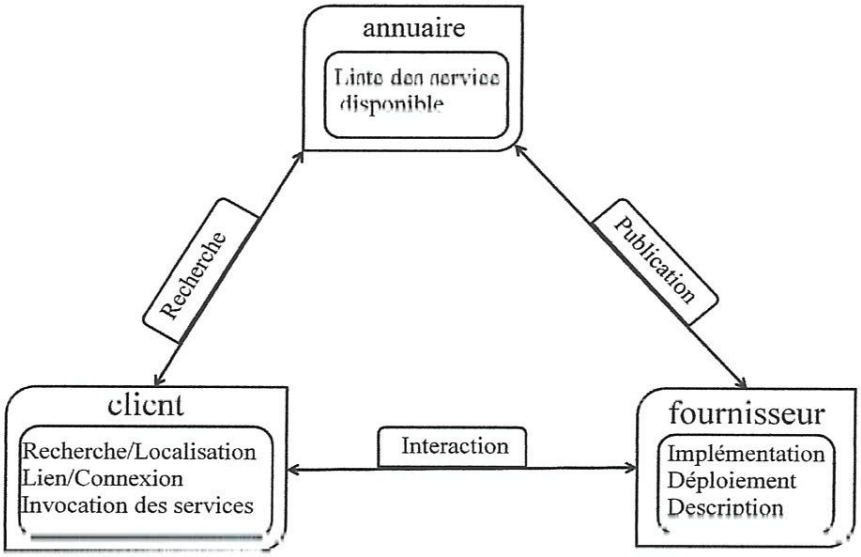


Figure 2.1 : *Fonctionnement des services Web.*

2.2. Les opérations associées aux services Web :

On distingue trois opérations relatives aux SW :

a. La publication : Le fournisseur de service doit le publier dans l'annuaire de services (UDDI) pour le mettre à la disposition des utilisateurs. Cette publication requiert quelques connaissances quant au modèle de données utilisé, dont voici les principaux éléments :

BusinessEntity : chaque service appartient à une organisation, qui est identifiée par un nom, une description, des personnes de contact, une ou plusieurs catégories, etc. .

BusinessService : inclut les informations non techniques relatives au service, comme le nom et une brève description.

BindingTemplate : définit où (URL) et comment (protocole) accéder au service.

T_Model : comprend des liens vers les informations techniques du service, comme par exemple un fichier WSDL (Web Service Description Language) décrivant les différentes méthodes disponibles.

b. La Recherche:

Il permet aux utilisateurs de chercher les services et de les localiser sur la base d'un ensemble de critères. Cette recherche se fait selon le scénario suivant :

- Selon ces besoins, le client interroge l'annuaire (UDDI) par l'envoi d'une requête encapsulée dans une enveloppe SOAP.
- Le résultat de la recherche dans l'annuaire sera transmis dans un message SOAP au client, contenant la description du service et les informations techniques nécessaires pour son invocation.
- Dans le cas de plusieurs réponses des techniques de sélection de services sont offertes (premier trouvé, aléatoirement) ou sur la base de paramètres techniques telle que la Qualité de service (QoS).

c. L'interaction (Binding) :

Une fois le service localisé, le Binding consiste à l'invocation du service par le client et la réponse du fournisseur à cette requête dans un mécanisme d'interaction. Il définit les détails techniques nécessaires pour consommer le service:

- Style de consommation des opérations
- L'encodage des messages requêtes et réponse [11]

3. Technologies des Services Web: Les standards

Une caractéristique qui a permis un grand succès de la technologie des services Web est qu'elle est construite sur des technologies standards de l'industrie.

XML, SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) et UDDI (Universal Description, Discovery and Integration) sont les technologies les plus dominantes.

Nous exposons ici les descriptions de chacune de ces technologies :

3.1. XML : Extensible MarckupLanguage:

XML est un langage de balisage générique qui dérive du SGML. À l'origine conçu pour la publication électronique à grande échelle, XML joue aussi un rôle de plus en plus important dans l'échange d'une large variété de données sur le Web et ailleurs.

W3C recommande depuis 1998 XML en tant que standard de description de données. XML est un méta langage permettant d'identifier la structure d'un document. Un document est composé d'une définition de sa structure et d'un contenu. La structure d'un document XML est souvent représentée graphiquement comme un arbre. La racine du document constitue le sujet du document, et les feuilles sont les éléments de ce sujet. De ce fait, XML est alors flexible et extensible, et est devenu rapidement le standard d'échange de données sur le Web.

a. Comment XML facilite l'utilisation des services Web?

L'avantage de son utilisation dans les services Web est compris essentiellement dans les points suivant :

– La séparation du contenu d'un document, de sa structure et de sa représentation facilite l'échange d'informations entre les différents partenaires.

– Permet la composition de services plus complexes à travers la définition des enchaînements entre les services grâce à BPEL4WS…

– Favorise l'émergence de standards d'industrie dans la mesure où les structures de données sont réutilisées et réutilisables.

b. DTD : Document Type Définition

C'est le document spécifiant la structure, la syntaxe et le contenu d'un document

XML, il peut être inclue dans le fichier XML, mais pour une meilleure réutilisabilité, la DTD est séparée du document XML qu'elle représente.

Elle peut contenir des déclarations, des notations, des éléments, des listes d'attributs, des commentaires…

Mais en réalité la DTD n'est pas un document XML, il est difficile à faire évoluer, Il n'y a pas de limites sur les caractères et la notion d'espace de nommage n'existe pas.

c. Les name spaces :

Consiste à regrouper les noms d'éléments et les noms d'attributs dans une collection qui sera identifiée par une URI. Cette technique permet d'éviter des conflits de noms dans un document XML et permet d'adapter les traitements au contenu du document.

Les name spaces favorisent la réutilisation des standards déjà définis dans des domaines d'activités spécifiques.

d. Schéma XML :

Il permet de spécifier la syntaxe d'une classe de documents XML en définissant les éléments et attributs ainsi que les contraintes sur celle-ci.(Mêmes objectifs qu'une DTD).

Nous allons prendre un exemple simple pour visualiser graphiquement quelle serait la hiérarchie du document XML correspondant et quelle est la structure du document.

Exemple XML :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE annuaire SYSTEM "annuaire.dtd">
<annuaire>
  <personne type="étudiant">
    <nom>HEUTE</nom>
    <prenom>Thomas</prenom>
    <email>webmaster@xmlfacile.com</email>
  </personne>
  <personne type="chanteur">
    <nom>CANTAT</nom>
    <prenom>Bertrand</prenom>
    <email>noir@desir.fr</email>
  </personne>
</annuaire>
Voici ce que pourrait-être sa DTD:
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT annnuaire    (personne*)>
<!ELEMENT personne  (nom,prenom,email+)>
<!ATTLIST personne type (étudiant | professeur | chanteur | musicien) "étudiant">
<!ELEMENT nom      (#PCDATA)>
<!ELEMENT prenom   (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

***Figure 2.2** : Exemple d'un document XML.*

3.2. SOAP : Simple Object Acces Protocol

SOAP est un protocole pour l'échange d'information dans un environnement repartie, basé sur le standard *XML*. Ce protocole consiste en trois parties : une enveloppe qui définit un canevas pour décrire le contenu du message et comment le traiter, un jeu de règles de codage à exprimer les cas de types de données défini par l'application et une convention pour représenter des appels de procédure distante (*RPC*) et ses réponses.

SOAP n'est pas lié à aucun système d'exploitation ni langage de programmation. Il est indépendant du langage d'implémentation des applications client et serveur. En plus, il peut potentiellement être utilisé avec une variété de protocoles (e.g. HTTP, HTTP Extension Framework).

a. Objectifs de SOAP:

- Offrir un format de message pour les communications entre partenaires, décrivant comment les informations à échanger sont structurées dans un document XML.
- Définir les conventions nécessaires à l'appel de procédures à distance pour invoquer un service par un client, en envoyant un message SOAP, et comment le service répond par un autre message SOAP.
- Décrire comment un message SOAP doit être transporté au-dessus de HTTP ou SMTP.

b. Fonctionnement de SOAP :

Les messages SOAP sont des transmissions fondamentalement à sens unique d'un expéditeur à un récepteur. Lorsqu'une transmissions d'un message commence (e.g. invocation d'un service Web), un message SOAP (ou document XML) est généré. Ce message est envoyé à partir d'une entité appelé le SOAP sender, localisé dans un SOAP nœud. Le message est transmis à zéro ou plusieurs nœuds intermédiaires (SOAP intermediates) et le processus fini lorsque le message arrive au SOAP receiver. Le chemin suivi par un message SOAP est nommé message path. La figure 2.3 montre les éléments qui participent au processus.

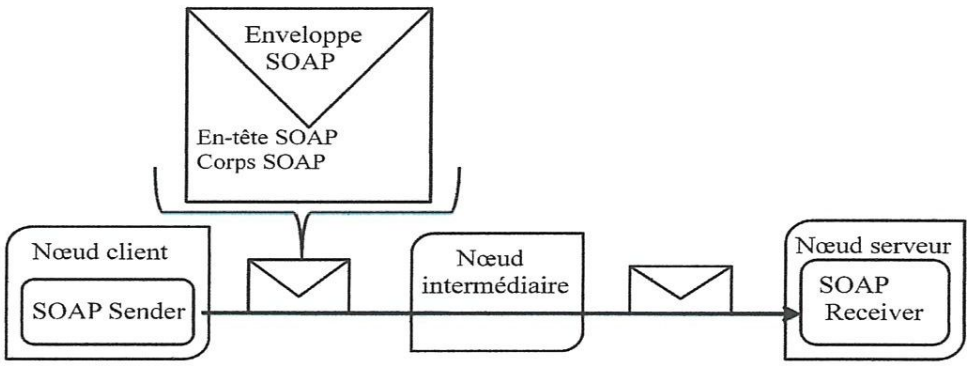


***Figure 2.3** : fonctionnement de SOAP.*

c. Structure d'un message SOAP :

Enveloppe : C'est un élément obligatoire dans un message SOAP, elle permet de spécifier la version de SOAP utilisée, en utilisant un espace de nom (http://www.w3.org/2003/05/soap-envelope), elle permet aussi de spécifier les règles d'encodage (sérialisation et dé sérialisation) mises en œuvre dans le message (encoding Style).

Entête : Optionnel, permet de spécifier certaines directives pour le traitement du message.

Peut contenir trois types d'éléments

- Actor : Permet de préciser le destinataire final du message (message path).
- MustUnderstand (0,1) : Spécifie que le récepteur du message doit obligatoirement comprendre cet élément. Si ce n'est pas le cas, le récepteur arrête tout traitement.
- Encodingstyle : les règles d'encodage (sérialisation et dé sérialisation) mises en œuvre dans le message.

Corps **:** Contient les données (paramètres) utilisées pour un appel de procédure distante effectué par le destinataire final et permet l'envoi de données non-XML. [11]

Exemple:

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
soap:encodingStyle="http://schemas.xmlsoap.org/
soap/
encoding/">
<soap:Body>
<soap:Fault>
<faultcode>soap:MustUnderstand</faultcode>
<faultstring>Mandatory Header error.</faultstring>
<faultactor>http://www.wrox.com/heroes/endpoint.asp</
faultactor>
<detail>
<w:sourcexmlns:w="http://www.wrox.com/">
<module>endpoint.asp</module>
<line>203</line>
</w:source>
</detail>
</soap:Fault>
</soap:Body>
</soap:Envelope
```

Figure 2.4 *: Exemple d'une enveloppe SOAP.*

3.3.WSDL : Web Services Description Language

WSDL décrit les services Web et particulièrement leur interface dans le format XML. Il décrit de manière abstraite et indépendante du langage de programmation, l'ensemble des fonctionnalités offertes par un service. Il permet de connaître les protocoles, les serveurs, les ports, le format des messages, les entrées, les sorties, les exceptions possibles et les opérations réalisées par un service Web. (fig.2.5) [11]

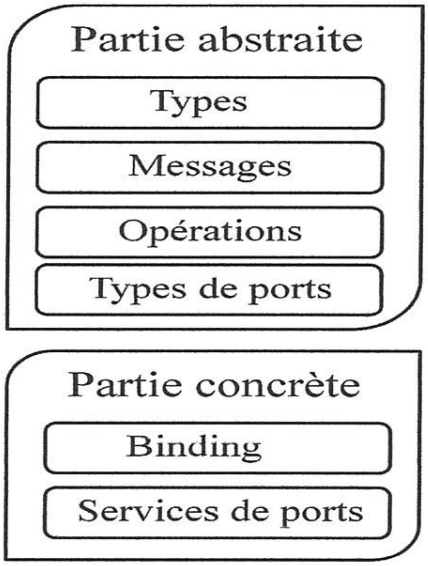


***Figure 2.5** : Structure d'un fichier WSDL.*

Les objectifs de WSDL sont :

- Décrire les interfaces des services en précisant les opérations offertes et leur signature (Paramètres d'Entrée/Sorties et types). Ces interfaces constituent les contrats que les clients doivent respecter pour pouvoir interagir avec le service.
- Préciser le (s) protocole (s) d'accès au service (HTTP, SMTP…).
- Fournir la localisation du service où il peut être invoqué (URI).
- Définir une description de la sémantique du service par la fourniture des informations permettant, éventuellement, aux développeurs de traiter la sémantique des services.

Cependant le W3DL est incapable de décrire des organisations complexes. Les services Web sont des tâches simples, il sera donc nécessaire de définir des enchaînements plus ou moins complexes pour réaliser des tâches élaborées.

3.4. UDDI: Universal Description Discovery and Integration

UDDI est né de la collaboration entre plusieurs entreprises dont IBM, Microsoft et Ariba. Assurant le rôle d'un médiateur centralisé pour l'ensemble des partenaires désirant faire des échanges à l'échelle du Web.

Le noyau d'UDDI travaille avec la notion de "business registry", qu'est un service sophistiqué de noms et répertoires. Plus précisément, UDDI définit des structures de données et APIs pour publier les descriptions des services dans le registre et pour interroger le registre afin de chercher des descriptions publiées. Parce que les APIs de UDDI sont aussi spécifiés en WSDL avec une attache SOAP, le registre peut être invoqué comme un service Web (en conséquence, ses caractéristiques peuvent être décrites dans le même registre, comme un autre service).

Les spécifications du « registry » UDDI ont deux buts principaux en ce qui concerne la découverte d'un service: le premier, soutenir les développeurs dans la découverte d'informations sur les services. Le deuxième, permettre la liaison dynamique et en conséquence habiliter les clients pour interroger le registre et obtenir des références aux services d'intérêt.

L'annuaire UDDI est composé de

- Pages blanches : Contiennent des informations sur l'entreprise comme le nom de la société, l'adresse.
- Pages jaunes : Contiennent la description des services Web, au format WSDL, déployés par l'entreprise.

Pages vertes : Contiennent les informations techniques détaillées sur les services fournis (processus métier, description de service….).[12]

Conclusion :

Les services Web sont autonomes, basés sur les technologies standard de l'industrie de WSDL (description), UDDI (publication), et SOAP (pour communiquer). Ils permettent aux utilisateurs de connecter des composants différents à travers les frontières organisationnelles d'une manière indépendante de la plate-forme et du langage. Cependant, aucune de ces normes ne permet de définir la sémantique des processus métier associée aux services Web. Ainsi, les services Web d'aujourd'hui sont isolés et opaques. Briser l'isolement des moyens reliant les services ce fait en précisant comment les collections de services Web sont utilisées conjointement pour réaliser les fonctionnalités plus complexes, ce qui est la interprétation d'un processus métier. En se sens, il est impératif d'ajouter la sémantique et la description du comportement ainsi que la spécification des processus métiers associés aux services Web.

Dans le prochain chapitre, on s'intéressera à l'aspect comportement externe des services Web par la description des processus métiers.

Chapitre III :

Processus métiers : Modélisation et Evolution

Chapitre III : Processus métiers : Modélisation et Evolution

Introduction :

La modélisation des processus métiers et de leur organisation est aujourd'hui considérée comme un préalable nécessaire à la conception d'un système d'information organisationnel. Or, en un demi-siècle, d'abord sous l'influence de l'environnement concurrentiel, puis pour répondre aux opportunités apportées par la mondialisation et le commerce électronique, l'organisation des activités productives a évolué en profondeur s'appuyant d'une part sur une approche processus, et intégrant d'autre part des principes de travail collaboratif et en mode projet. Il existe cependant un décalage important entre ces nouvelles formes d'organisation du travail et les possibilités de représentation offertes par les modèles de processus métier. De plus, les modèles une fois conçus s'adaptent difficilement aux évolutions.

Dans ce chapitre, on abordera en détails les notions de processus métiers, leurs modèles de représentations ainsi que leur évolution.

1. Processus métier :

1.1. Définitions :

Plusieurs définitions du terme processus métiers sont présentées dans la littérature. Il est vrai que cette notion est suffisamment générale pour être utilisée dans différents domaines scientifiques ou applicatifs. L'étude de ces définitions permet d'avoir une idée claire de ce qui est désigné par un «processus métier». [32]

Les définitions d'un processus retenues dans la littérature sur la gestion des processus métiers focalisent le plus souvent sur trois éléments : le résultat à atteindre, le découpage en travaux à exécuter et la structure logique d'ordonnancement de ces travaux, suivant en cela la définition normalisée par ISO9000:2000 qui le présente comme **un ensemble d'activités corrélées ou interactives qui transforme des éléments d'entrée en éléments de sortie**. [33]

Une autre définition précise que les processus métier sont les processus représentatifs des activités de l'entreprise indépendamment des moyens humains et techniques. Ces processus interfèrent de manière transversale dans le système d'information et peuvent même traverser les frontières organisationnelles internes de l'entreprise. Les processus métier traversent même les limites de l'entreprise pour collaborer avec les partenaires comme les fournisseurs et les distributeurs.

La définition la plus simple que nous allons retenir est la suivante :

Un processus métier est un enchaînement d'actions réalisées par différents acteurs collaborant pour délivrer un résultat tangible et une valeur ajoutée métier pour l'entreprise. [34]

Le terme de processus métier est souvent utilisé à tort et à travers pour désigner des notions différentes : processus exécutable, processus abstrait, processus collaboratif, etc. Prenons le temps de définir ces différentes notions :

Un processus métier est une chorégraphie d'activités incluant une interaction entre participants sous la forme d'échange d'informations. Les participants peuvent être :

- Des applications / services du SI
- Des acteurs humains
- D'autres processus métiers. [36]

Il faut préciser qu'un processus métier peut être interne à une entreprise ou mettre en jeu des entreprises partenaires.

1.2. Cycle de vie d'un processus métier :

Le cycle de vie d'un processus est composé de quatre étapes : modélisation du processus – métier, fonctionnelle et technique – déploiement, exécution se basant sur les applications existantes de l'entreprise, et interaction des utilisateurs avec ce processus (contrôle d'exécution, optimisation). [37]

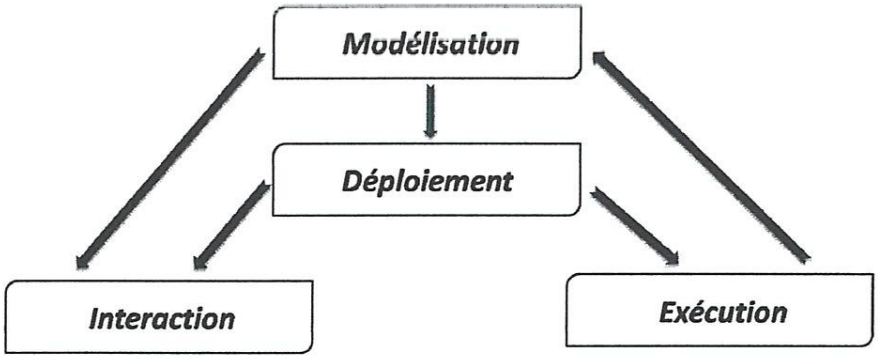


***Figure 3.1**: Cycle de vie d'un processus métier.*

1.3. Caractéristiques des processus métier :

Une entreprise atteint ses objectifs en réalisant une ou plusieurs activités, et par activités on veut dire un ensemble d'actions et d'opérations visant un objectif déterminé et généralement identifié par domaine, métier, secteur, localisation ou type de produit manufacturé.

Par exemple, une banque peut exercer plusieurs activités comme la gestion patrimoniale de clients privés, les crédits hypothécaires, la vente de produits d'assurances, etc.

L'entreprise fondée sur une démarche de gestion des processus métiers présente un mode de fonctionnement privilégiant la collaboration de ses employés affectés sur des enchaînements déterminés de tâches conduisant à réaliser les objectifs de la direction.

Un processus métiers est un concept abstrait qui n'existe qu'au travers le travail des employés assistés de systèmes automatisés présentant diverses caractéristiques :

- La portée d'un processus métier est étendue en intégrant largement les flux de matières et d'information de l'organisation dans la production de biens et de services.
- Le processus métier est dynamique en répondant aux besoins et attentes des clients tout en s'accommodant des changements du marché ou contextuel de l'organisation.
- Le processus métier est distribué et offre une certaine souplesse d'adaptation en intégrant d'autres processus métiers situés en dehors de la frontière de l'organisation.
- L'exécution d'un processus métier est réalisée sur un temps plus ou moins long en considérant son éventuelle suspension et sa reprise ultérieure.
- L'exécution d'un processus métier est transactionnelle. Le cheminement des tâches déclenché continue jusqu'à l'état final (Atomicité). La transaction laisse toujours les résultats intermédiaires et finaux dans des états valides (Cohérence). Chaque instance d'un processus est indépendante des autres instances (Isolation). Le résultat du processus métier est durable (Durabilité).
- L'exécution du processus métier est partiellement ou totalement automatisée en remplaçant les tâches routinières, répétitives et de faibles valeurs ajoutées par des systèmes automatiques.
- Le processus métiers est défini suivant une stratégie de réalisation des objectifs fixés.
- Le processus métier consomme des ressources afin de traiter ou transformer des entrées en sorties formellement identifiées.
- L'exécution du processus métier est mesurable.
- Le processus métiers entretient la collaboration entre les individus ayant des objectifs communs.

1.4. Les protocoles métier dans les services Web :

Il n'est plus besoin de justifier que, la transition des systèmes d'information centrés-données vers des systèmes d'information dirigés vers les processus, prenne une place prépondérante

dans les domaines de recherche actuels. La tendance actuelle est celle des architectures orientées services.

En effet, cette nouvelle tendance augmente les besoins des entreprises en termes de flexibilité et d'adaptabilité. Ce besoin est ressenti autant au niveau conceptuel qu'au niveau opérationnel.

An niveau conceptuel, le besoin de flexibilité est reflété par le développement de langages pour la composition de Web services tels que : BPEL4WS2 et BPEL 3.

Au niveau opérationnel, de nouveaux middleware, tels que : BizTalk, IBM Web sphère Choreographer, servent de plate-forme pour l'exécution des services (flux).

Dans cette section on va décrire l'ensemble des caractéristiques des processus métiers :

Les business protocoles constituent une partie nécessaire de la description de service Web. Ils assurent la spécification de la conversation supportée par les services.

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 analyses 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, les vérifications de conformité, la correspondance statique ou dynamique, tout en prenant l'avantage de la description des protocoles.

Nous définissons trois types d'analyses de protocoles: analyses de compatibilités, analyses de remplacements et analyses consistantes.

Les analyses de compatibilités consistent à vérifier si deux services peuvent interagir correctement, bases sur leurs définitions de protocoles (si une conversation peut être établie entre les services considères).

Les analyses de remplacement référent aux vérifications si deux protocoles peuvent supporter les mêmes conversations (c'est-a-dire si un service peut remplacer un autre en général ou en interagissant avec un des clients spécifiques).

Finalement, l'analyse consistante est considérée pour supporter les changements dans les protocoles et leurs influences aux demandeurs.

2. Modélisation des processus métier :

Les méthodes de modélisation des processus métier sont performantes pour décrire les procédures de gestion.

2.1. Principes de la modélisation :

a. Intérêt :

La modélisation par processus, de l'entreprise, est néanmoins l'apport conceptuel le plus important pour la gestion des entreprises. Les modèles de processus qu'ils proposent intègrent généralement plusieurs vue (données, fonctions, organisation, produits, contrôles …) mais la perspective 'fonction' de l'organisation demeure le pivot de l'analyse dans la plupart des méthodologies, car elle se rapprocherait le plus de la définition du processus de gestion.

Le processus est alors décrit du point de vue des fonctions à exécuter et leur séquence, c'est-à-dire comme un flux de fonctions piloté par un flux de contrôle qui décrit les évènements déterminant le déclenchement et la séquence 'logique' des fonctions.

Cette logique est déterminée par les objectifs existants de l'organisation et par des contraintes technologiques liées au métier, plus que part les pratiques professionnelles réelles des acteurs de l'organisation. Pour modéliser, nous allons définir le concept de modèle.

b. Définitions d'un modèle :

Un modèle est une abstraction de la réalité. L'abstraction est un des piliers de la modélisation.

- Il s'agit d'une démarche qui consiste à identifier les caractéristiques intéressantes d'un domaine, en vue d'une utilisation précise.
- L'abstraction désigne aussi le résultat de ce processus, c'est-à-dire l'ensemble des caractéristiques essentielles d'une entité, retenues par un observateur.

On peut dire aussi qu'un modèle est une vue subjective, mais pertinente de la réalité.

- Un modèle définit une frontière entre la réalité et la perspective de l'observateur. Ce n'est pas "la réalité", mais une vue très subjective de la réalité.
- Bien qu'un modèle ne représente pas une réalité absolue, un modèle reflète des aspects importants de la réalité, il en donne donc une vue juste et pertinente. [37]

A titre d'exemples : une image, une maquette, un prototype, cartographie sont des modèles.

c. Caractéristiques fondamentales des modèles :

Le caractère abstrait d'un modèle doit notamment permettre :

- De faciliter la compréhension du système étudié : il réduit la complexité.
- De simuler le système étudié : un modèle représente le système étudié et reproduit ses comportements.
- Un modèle doit être : fiable, claire et concis.
- Technologies de processus métier
- Construire des processus métier inter-organisationnels requière une intégration à des niveaux déférents, 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 workfows.

2.2. Les technologies de gestion des processus métiers :

Dans cette section, nous exposerons les familles de technologies qui implémentent et gèrent les processus métier.

- *Techniques Ad hoc*

Les processus métier peuvent être implémentés en utilisant les langages de programmation tels que C et Java. Dans cette approche, le processus métier est modélisé (par exemple comme un document Word). Ensuite, les activités principales sont implémentées par des programmes. L'interaction entre ces programmes est supportée par des interconnexions entre des protocoles de communication appropriés (par exemple, les sockets TCP/IP).

Le problème critique de cette solution est que tous les protocoles de communication utilisés doivent être compris. En même temps, toute modification dans le protocole de communication ou la logique métier requière la recompilation des programmes.

- *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.

- *Services Web :*

Le concept de service Web est devenu récemment très populaire. Un service Web (ou simplement service) est une application autonome ou un composant, i.e., une fonctionnalité sémantiquement bien définie, uniquement identifiée par un identificateur uniforme de ressource (ou Uniform Resource Identifier URI). Comme exemples typiques de services Web, nous citons les réservations de billets en lignes, la gestion des relations entre clients, la facturation, la comptabilité et la chaîne de la provision.

Les services Web présentent l'application la plus concrète de SOA (Service Oriented Architecture).

Nous allons nous intéressés au modèles qui permettent de décrire les processus métier dans le contexte des services Web.

3. Les approches de modélisations des processus métier :

3.1. Modélisation par réseaux de PETRI

Ainsi l'approche réseau de Petri est un cadre théorique pour la vérification de la composition de services Web. Cette approche propose d'utiliser plusieurs operateurs algébriques tels que: séquence, choix alternatif, arbitraire pour la vérification et l'analyse de la composition.

En effet, un réseau de Petri est un graphe dirigé biparti connecte dont chaque nœud est soit une transition soit une place.

Les réseaux de Petri, qui expriment naturellement la "vraie concurrence", possèdent une représentation graphique intuitive et très visuelle ainsi que de nombreux outils de vérification associes.

Ceci est du grâce aux similitudes existantes entre les concepts de services Web et les réseaux de Petri. Les services Web sont considères comme un ensemble d'opérations ordonnées et un ensemble d'états. Les opérations sont les transitions et les états constituent les places. [28]

a. Formalisme :

Un réseau de Petri est un graphe orienté comportant :

Un ensemble fini de *places* P = {P*1*,*P2*,...,*Pn*} : conditions ou statuts

- Place : 

Un ensemble fini de *transitions* T = {T*1*, *T2*, ..., *Tn*} : évènements ou actions

- Transition : 

Un ensemble fini d'arcs orientés et unidirectionnels :

Liaisons : 

b. Marquage d'un réseau de Petri :

Le marquage d'un réseau représente son état.

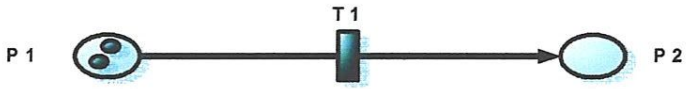


Figure 3.2 : *Marquage d'un réseau de Petri*

- Pour ce faire, on place à l'intérieur des places un nombre fini (plusieurs ou nul) de marques ou jetons.
- Condition logique (processus) : le jeton indique que cette condition est vraie (place marquée) ou fausse
- Ressources : le jeton indique la quantité de ressource en stock.
- Marquage initial : M0, correspond à la distribution initiale des jetons dans chacune des places : état initial du système.

c. Dynamique d'un réseau de Petri :

Figure 3.3 : *La dynamique d'un réseau de Petri*

- Places d'entrée : d'où sont issus les arcs orientés vers la transition (e1, e2)
- Places de sortie : où aboutissent les arcs orientés issus de la transition (s1, s2)

d. Règles de parcours d'un réseau de Petri :

- RdP ordinaire : Le franchissement d'une transition validée consiste à enlever un jeton de toutes les places d'entrée de la transition et à en déposer un dans toutes les places de sortie de cette même transition.

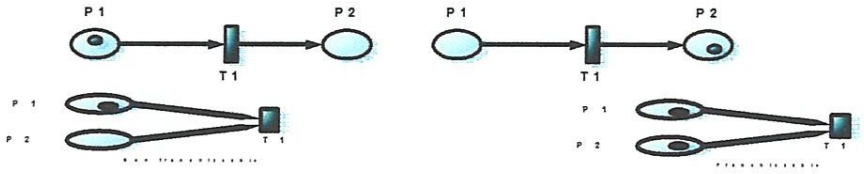


Figure 3.4 : *RDP ordinaire*

- RdP généralisés : permet de représenter un déplacement de ressources ou un stock. Dans ce type de RdP, les arcs sont le plus souvent values.

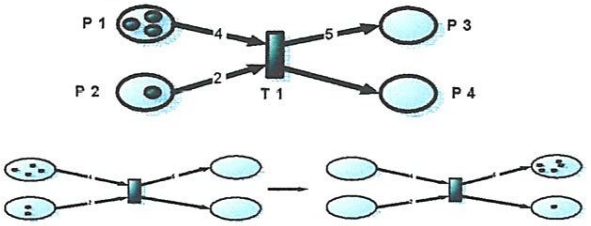


Figure 3.5 : *RDP généralisé*

Exemple : Réservation en ligne

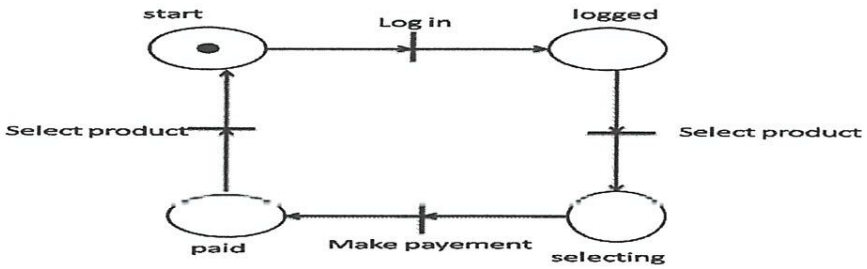


Figure 3.6 : *Exemple d'un réseau de Petri du service réservation en ligne.*

3.2. Modélisation par automates d'états finis : (AEF)

Cette approche utilise la notion d'automates à états finis pour représenter le comportement des services Web et consiste à modéliser un service Web par un ensemble d'états et de transitions. Mais avant on doit connaitre la notion d'automate :

a. Automate fini :

C'est une machine abstraite qui est un outil fondamental dans la modélisation de processus, le contrôle, les protocoles de communication, la vérification de programmes, la théorie de la calculabilité, dans l'étude des langages formels et en compilation.

Les automates finis reconnaissent exactement des langages rationnels. Ce sont les machines les plus simples dans la hiérarchie de Chomsky, et par conséquent ils sont moins puissants que les automates à pile et les machines de Turing.

Un automate est constitué d'états et de transitions. Son comportement est dirigé par un mot fourni en entrée : l'automate passe d'état en état, suivant les transitions, à la lecture de chaque lettre de l'entrée.

L'automate est dit « fini » car il possède un nombre fini d'états : il ne dispose donc que d'une mémoire bornée. On peut très bien considérer des automates sans limitation sur le nombre d'états.

b. Formalisme :

Un automate à états fini est un quintuplé A = {S, S0, F, M, R} avec

S : Ensemble fini d'états.

S0 : Etat initial.

F : Ensemble d'états finaux.

R C (SxSxM) : Ensemble de transitions.

M : Ensemble fini de messages.

c. Présentation graphique

Un automate à états finis est représenté par un graphe orienté dont les nœuds représentent les états et les arcs, annotés par des éléments de l'alphabet, décrivent les transitions.

Exemple

On donne comme exemple l'automate caractérisé par *:*

- Q= {q0, q1, q2} ;
- R= {a, b, c, d} ;
- q0 ϵ Q0 est l'état initial ;
- q2 ϵ F est l'état final ;

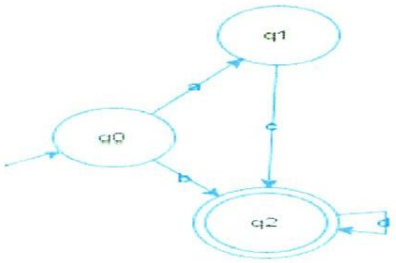


Figure 3.7 : *Exemple d'automate représentant un service à trois états.*

Définition de l'automate :

Caractéristiques d'un automate à états finis

• Un automate est constitué des éléments suivants:

– Un ensemble fini d'états: Q= {q0;q1;q2}

– Un message (un ensemble de symboles) S= {V;D}

– Un ensemble de transitions: T= {(q0,V,q1);(q1,D,q2)}

– Un état initial: q0

– Un ensemble d'états terminaux (accepteurs): F= {q2}

d. Automate déterministe

Un automate est dit déterministe ssi

$$card(S_0) = 1,$$
$$\forall s \in S. \forall a \in E. card(T(s,a)) \leqslant 1.$$

Ou on peut dire qu'un automate est déterministe si a partir d'un état on ne peu avoir plus d'une transition pour le même message.

3.3. BPEL-Abstract: Abstract Business Process Execution Language

Ce standard offre un langage d'exécution des processus métier distribues et transactionnels. Il supporte l'orchestration de multiples processus indépendants mais communiquant entre eux. En effet, BPEL est la représentation XML d'un processus exécutable, qui peut être déployée sur n'importe quel moteur de processus métier. L'élément atomique d'un processus BPEL est une "activité", qui peut être l'envoi d'un message, la réception d'un message, l'appel d'une opération ou une transformation de données. BPEL offre les fonctionnalités suivantes :

- Organisation des activités : activités séquentielles, parallèles, Switch conditionnel . . .
- Gestion des erreurs : il est possible de définir des gestionnaires d'erreurs pour des erreurs spécifiques, métier ou techniques. Des erreurs peuvent être déclenchées, traitées ou ignorées ;
- Gestion des transactions : il existe deux modes de transactions : les transactions courtes,
- XA ou 2-phase commit, et les transactions longues, ou transactions compensées ;
- Organisation du processus en sous processus.

Un processus BPEL **(figure 3.8)** définit, en XML, les activités réalisées dans le cadre d'exécution du processus métier. Toutes les informations techniques nécessaires sont décrites.

```
<process>
      <partners/>          → Définition des partenaires (WebServices)
      <containers/>        → Définition des conteneurs de données
      <sequence>
            <receive/>     → Réception d'une requête
            <assign/>      → Transformation de données
            <invoke/>      → Appel de Web Service
            <assign/>      → Transformation de données
            <reply/>       → Envoi de la réponse
      </sequence>
</process>
```

***Figure 3.8** : Exemple de BPEL-Abstract*

Cette description fournit les informations nécessaires pour composer le processus avec d'autres services Web. BPEL soutient cela en fournissant des dialectes distincts pour spécifier les processus abstraits et exécutables.

Malheureusement, BPEL n'empêche pas que les complexes calculs soient inclus dans un processus abstrait.

Cela complique la description du protocole, révèle inutilement les détails de mise en œuvre, et le rend difficile à analyser son exactitude. [12]

3.4. OWL_S: Ontology Web Language for Services

OWLS est un langage de description de services basé sur XML utilisant le modèle des logiques de descriptions (une proposition d'ontologie des services Web). Il est aussi un langage de haut niveau pour la description et l'invocation des services Web dans lequel la sémantique est incluse et interface avec UDDI/WSDL/SOAP.

Il propose une sémantique des données, et pas seulement des champs prédestinés par la structure des standards ou par des "feuilles de styles" utilisées pour décrire les services. OWL utilise des logiques de descriptions pour modéliser les services, donc, il nous permet une grande puissance d'expression, que ne possèdent pas les autres systèmes.

Composants principaux de OWLS Ontologie

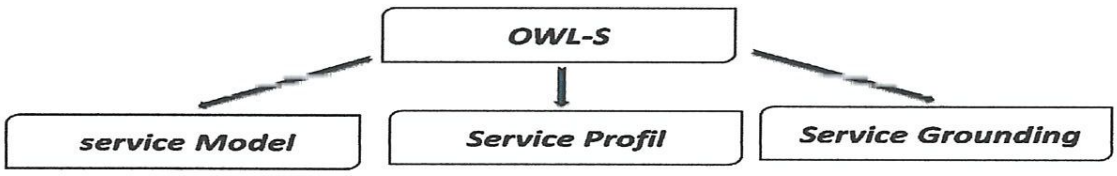


***Figure 3.9:**Structure d'ontologies dans OWLS.*

a. Service Profile :

- Présenté par un service
- Représenter ce que le service fournit
- Usage Principales:
  - Publicité des capacités
  - Demande de service Web qui a donnée un ensemble de sa capacité.

b. Process Model :

- Décrit comment ce service fonctionne: les processus internes du service.
- Spécifie le protocole d'interaction de service.
- Spécifie les messages abstraits: l'information transmis sur type d'ontologie.
- Faciliter l'invocation et la composition des services, la surveillance de l'interaction.

c. Service Grounding :

- Fournit des spécifications d'information d'accès de service.
- Service Model+Grounding donne tous les choses nécessaire pour utiliser ce service.
- Construire sur WSDL pour définir la structure de message et la couche physique.

Spécifier le protocole communication, mécanismes de transport, langages de communication, etc.. [38]

3.5. MCT : Le modèle conceptuel des traitements de MERISE

Le MCT est une représentation de la succession des règles de gestion dont le service veut se doter pour répondre aux événements auxquels il doit faire face, du fait de son activité et de son environnement.

Il décrit formellement les activités exercées par chaque processus déterminé lors du recueil de l'existant. Cette description se fait indépendamment de toute considération organisationnelle. Les différents processus (paie, facturation, livraison, ...) vont êtrc dócrits sans se soucier de savoir QUI les réalise, QUAND et OÙ ils sont réalisés.

a. Formalisme :

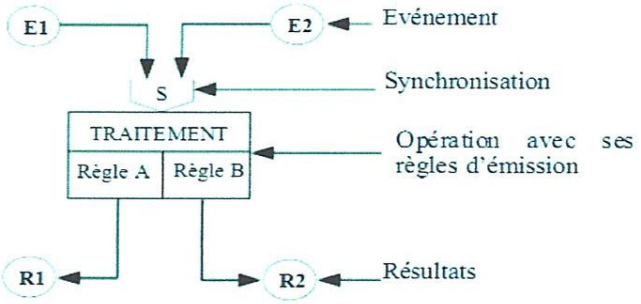


Figure 3.10 *: Formalisme du modèle MCT pour représenter des procesus.*

- L'événement est un fait nouveau pour le SI ou le processus concerné. Il est externe au processus, provoque une réaction de celui-ci (exécution d'une ou plusieurs opérations) qui peut se traduire par l'émission de résultats
- Une opération est un ensemble d'actions (traitements) accomplies par le processus en réaction à un ou plusieurs événements déclencheurs
- Une condition d'émission est une formalisation traduisant les règles de gestion à laquelle est soumise l'émission du résultat d'une opération.
- Le résultat est le produit d'une opération.

La synchronisation est la représentation d'une condition (expression booléenne) qui doit être vérifiée au déclenchement d'une opération. [31]

Exemple **:** Processus élaboration devis :

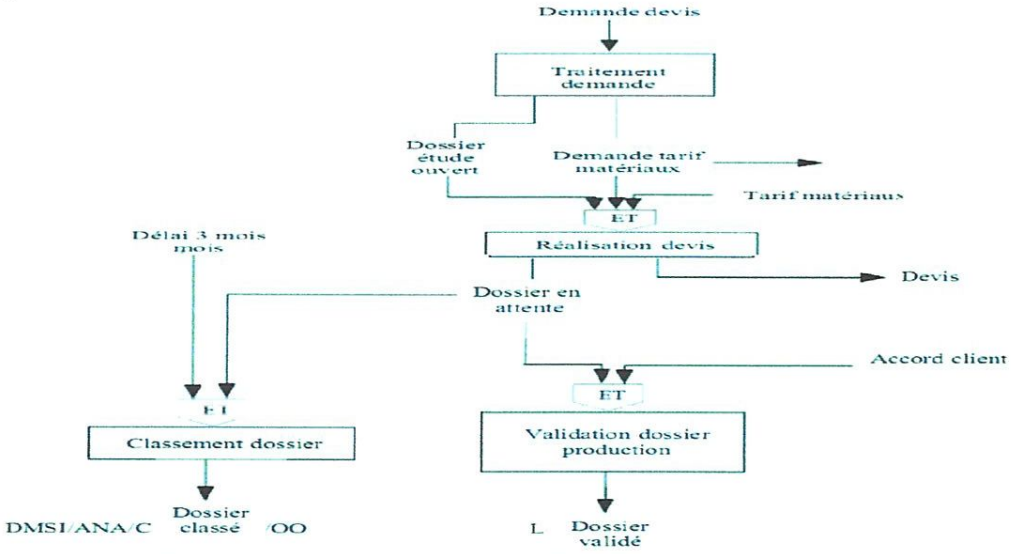


Figure 3.11 : *Exemple d'un modèle MCT d'élaboration devis*

3 .6. Diagramme de séquence UML

UML (Unified Modeling Language) est un langage de modélisation graphique à base de pictogrammes. Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes.

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont :

- Activité d'un objet/logiciel
- Acteurs
- Processus
- Schéma de base de données
- Composants logiciels
- Réutilisation de composants

Grâce aux outils de modélisation UML, il est également possible de générer automatiquement une partie de code, par exemple java, à partir des divers documents réalisés.

UML n'étant pas une méthode, leur utilisation est laissée à l'appréciation de chacun, même si le diagramme de classes est généralement considéré comme l'élément central d'UML ; des méthodologies, telles que l'Unified Process, axent elles l'analyse en tout premier lieu sur les diagrammes de cas d'utilisation (Use Case).

De même, on peut se contenter de modéliser seulement partiellement un système, par exemple certaines parties critiques.

UML se décompose en plusieurs sous-ensembles

- Les *vues* : Les vues sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet.
- Les *diagrammes* : Les diagrammes sont des éléments graphiques. Ceux-ci décrivent le contenu des vues, qui sont des notions abstraites. Les diagrammes peuvent faire partie de plusieurs vues.
- Les *modèles d'élément* : Les modèles d'élément sont les briques des diagrammes UML, ces modèles sont utilisés dans plusieurs types de diagramme. Exemple d'élément : classe, association, etc.

Les points forts d'UML

UML est un langage formel et normalisé : gain de précision, gage de stabilité et encourage l'utilisation d'outils UML est un support de communication performant : Il cadre l'analyse, facilite la compréhension de représentations abstraites complexes en plus son caractère polyvalent et sa souplesse en font un langage universel.

Les points faibles d'UML

La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation. UML n'est pas à l'origine des concepts objets, mais en constitue une étape majeure, car il unifie les différentes approches et en donne une définition plus formelle. Le processus (non couvert par UML) est une autre clé de la réussite d'un projet.

Or, l'intégration d'UML dans un processus n'est pas triviale et améliorer un processus est une tâche complexe et longue. L'acceptabilité industrielle de la modélisation objet passe d'abord par la disponibilité d'un langage d'analyse objet performant et standard. [38]

Deus types de diagrammes UML peuvent représenter les processus métiers : Diagramme de séquence et diagramme d'activités.

Exemple :

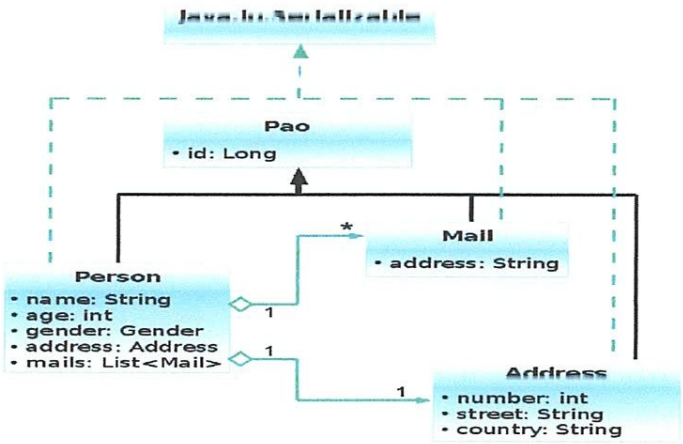


Figure 3.12 : *Diagramme UML des objets de communication ou de transfert.*

4. Etude comparative des différents modèles de représentation des BP existante :

	Outil logiciel	*Boucle*	*Temps*	*Formalisme*	*Contrôle d'exécution*	*Facilité de composition*	*Expressivité du model*
Réseaux de PETRI	***Oui***	***Oui***	*Minimale*	*Graphe*	*Bon*	+	+
Automate	***Oui***	***Oui***	*Dépend des circuits de données*	*Graphe*	*Elevé*	++	++
BPEL-Abstract	***Oui***	***Oui***	*Dépend du volume du document*	*Document XML*	*Bon*	±	-
OWL_S	***Oui***	***Oui***	*Minimale*	*Document XML*	*Elevé*	±	-
MCT	***Oui***	***Oui***	*Minimale*	*Graphe*	*Bon*	++	++
UML	***Oui***	***Oui***	*Minimale*	*Graphe*	*Elevé*	++	++

Tableau 3.1 : *Tableau de comparaison entre les différents types de modèles BP.*

Nous pouvons conclure de l'analyse précédente que :

Les protocoles de services constituent une partie nécessaire de la description de service Web. Ils assurent la spécification de la conversation supportée par les services.

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.

Pour notre travail on a choisi la modélisation des nos business process par des automates d'états fini déterministes et cela est justifié par :

- Ça compréhension facile par les utilisateurs.
- Ils sont appropriés pour décrire le comportement réactif.
- Utile pour le contrôle d'exécution de services.
- Existence d'outils logiciels pour la vérification.
- Ils sont basés sur une assise théorique solide ce qui leur confèrent un formalisme rigoureux.

5. Evolution des protocoles de service Web

Les services Web appartiennent à un environnement ouvert alors leurs comportement n'est pas fixe; ils peuvent êtres exposés à de différents changements de logique métier, selon les besoins de gestionnaire de ces protocolcs.

Les raisons des changements :

Diverses raisons sont à l'origine d'actions de changement des processus métiers. On peut citer les plus importantes, à savoir :

Ce changement est du à plusieurs raisons :

- La maintenance : certaines opérations de maintenance nécessite une modification du déroulement des protocoles utilisé.
- Evolution : pour plus de bénéfice ; il doit y avoir des changements de protocole pour perfectionner et avoir plus de gain.
- Alignement des processus: c'est la démarche consistant à redessiner les structures organisationnelles et systèmes d'information et de production afin qu'ils soient en parfait accord avec la stratégie élaborée.
- Fusion d'entreprises : le but de ces fusions est de rendre l'entreprise plus compétitive dans son métier de base en joignant ces efforts à d'autres partenaires.
- Adaptation aux lois : c'est l'adaptation des protocoles utilisés aux législations qui change selon l'évolution des spécificités de chaque société.
- Rachat d'entreprise : le rachat d'une entreprise nécessite une réorganisation de cela et une adaptation des anciennes ressources à la nouvelle politique de gestion.

6. Problématique et motivation:

Les business processus en tant que composant de services Web peuvent maintenant être identifiés, mesurées, gérés, alignés et évolués selon les changements de leurs environnements d'exécution ouverte à toutes nouvelles technologies engendrant une augmentation d'intérêts.

Les modifications que subi ces processus peuvent provoquer des violations de plusieurs contraintes et endommager l'exécution de ces protocoles et compromettre les intérêts des utilisateurs de ces services.

Donc la conception de ces protocoles doit satisfaire plus d'agilité et d'adaptabilité dans la gestion de ces changements, pour cela on va procéder à la modélisation par les automates d'états finie et extraire les différences engendré de l'évolution qui a eu lieu.

Mais comment on va exprimer ce changement ?

D'abord on va essayer de concevoir un système de gestion des protocoles Web qui facilite leur mise à jour et l'extraction des changements engendrés.

Après on va tenter la présentation de ces changements sous un format standard qui modélise le plus fidèlement tout les propriétés des différances entre les anciens et nouveaux protocoles.

A ce stade on essayera de donner une méthode qui facilitera la gestion de l'impact de ces changements sur les instances en cours en les classant en deux catégories.

Conclusion

Dans ce chapitre nous avons présenté les processus métier avec leurs différents type de modélisations et après une évaluation de chaque type nous avons choisi les automates d'états finis déterministes pour les utilisé dans la présentation des protocoles de notre système.

Dans le chapitre prochain, et à partir des protocoles présenter on AEF, on va essayer de proposé un modèle de présentation des changements apporté à ces protocoles pour faciliter la gestion de l'impact de cette évolution sur les instances en cours.

Chapitre IV :

Analyse et Conception

Introduction

Les services Web, sont rapidement devenus le choix privilégié pour le développement d'applications distribuées et hétérogène et les protocoles métier (Business Process) ont rapidement gagné une position importante comme une partie nécessaire de la description du service.

Comme on a vu dans le chapitre précédent, les processus métier peuvent êtres exposés à de différents changements de logique métier qui implique leurs mise à jour, mais ces modifications peuvent provoquer des violations de plusieurs types de contraintes.

Pour la gestion de ce changement on va procéder a une modélisation des ces protocoles et leur mise a jour on utilisant les automates d'états finis déterministe (AFD) et on essayera d'extraire le changement qui a eu lieu.

Mais comment on va exprimer ce changement ?

On va utiliser le concept de Pattern, et particulièrement les patterns d'évolution.

Dans un premier temps, on commence par expliquer la notion de pattern.

1. Les patterns : Concepts, modèles et utilité

1.1. Définitions d'un pattern :

Le mot anglais « pattern » est souvent utilisé pour désigner un modèle, une structure, un motif, un type , simplifié qui représente la structure d'un phénomène complexe. Il s'agit souvent d'un phénomène ou d'une organisation que l'on peut observer de façon répétée lors de l'étude de certains sujets, auquel il peut conférer des propriétés caractéristiques.

Ces patterns sont établis à partir des réponses à une série homogène d'épreuves et se présentant sous forme schématique. [19]

Un pattern constitue donc une solution générique à un type de problème fréquemment rencontré, en décrivant et formalisant les concepts sous-jacents à cette solution.

Exemple de Pattern : Domaine de Génie logiciel

Commande (Command, Action ou Transaction) : [17]

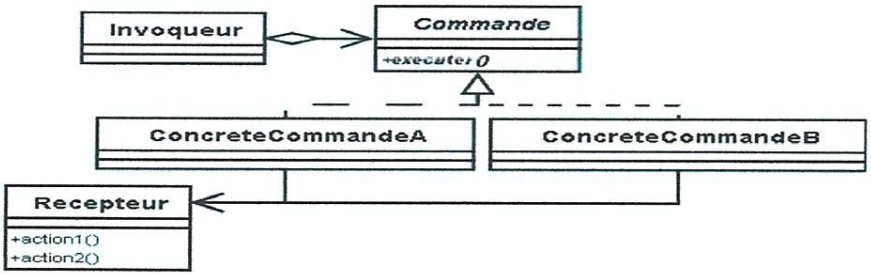


Figure 4.1 : *Exemple simple de pattern.*

L'objectif de ce pattern est d'encapsuler une requête sous la forme d'objet et la paramétrer facilement ont permettant des opérations réversibles sur ces objets.

L'invoqueur peut exécuter la méthode ' executer ()' contenue dans l'objet 'commande' et par conséquent géré deux instance de cette méthode A-B.

1.2. Caractérisation d'un pattern

Les principales caractéristiques sont les suivants :

• Un pattern peut être répété une ou plusieurs fois dans la même application ou dans des applications différentes.

• Aide a trouvé la solution la plus souple, extensible, pratique et compréhensible pour un concepteur.

• Défini un schéma (un modèle) que l'on va pouvoir réutiliser : Toujours un certain degré d'adaptation, par exemple. : Une boucle peut avoir différents formes, initialisations, conditions, mais ça reste une boucle. [20]

1.3. Evaluation des patterns

a. Avantage des patterns

Les avantages de l'utilisation des patterns sont les suivants :

– Un vocabulaire commun.

– Une capitalisation de l'expérience.

– Un niveau d'abstraction plus élevé.

– Réduction de la complexité.

– Guide/catalogue de solutions.

b. Inconvénients des patterns

Ce pendant les patterns n'échappent pas à certaines limites, à savoir :

– Effort de synthèse : reconnaître, abstraire sont des tâches complexes.

– Apprentissage, expérience : Effort de modélisation considérable.

– Des patterns différents mais souvent proches : Peuvent engendrer des confusions.

– Manque de modèles universels pour les représenter. [18]

1.4. Représentation des changements avec les patterns d'évolution :

Quand il y'a une opération de changement de protocole, il aura des différences qui doivent être extraits et classifiés selon leur type, on distingue les opérations de changement typiques suivantes: [24]

a. *Insertion d'un état* : Un état X est ajouté au protocole S.

Comment est-ce que le nouveau état X est inséré dans le schéma du protocole ?

- X est insérée entre deux activités directement suivantes.
- X est insérée entre deux ensembles de l'activité :
  - Sans condition supplémentaire (en cas parallèle).
  - Avec condition supplémentaire (en cas conditionnel).

Exemple :

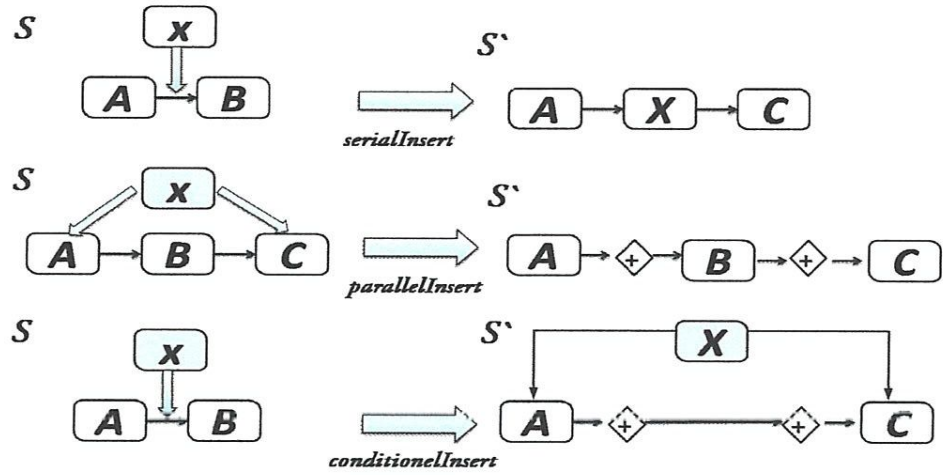


Figure 4.2 : *Insertion d'un état dans un protocole*

b. Suppression d'un état :

Un état peu être ignorer ou supprimer d'un protocole S.

Exemple :

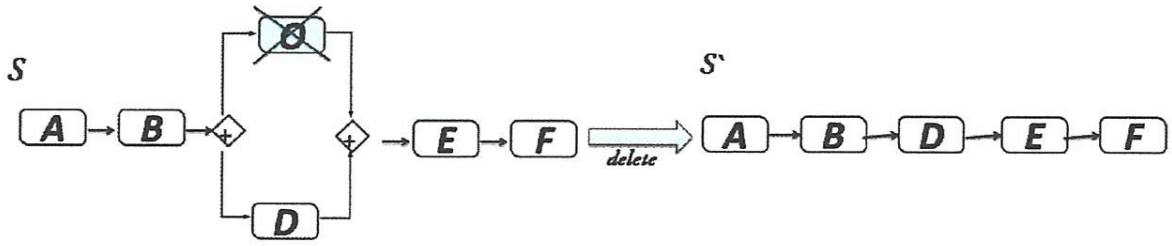


Figure 4.3 *: Suppression d'un état d'un protocole.*

c. Déplacement d'un état:

Un état peut changer de position dans le même protocole.

Comment est-ce que le nouveau fragment d'état est X a intégré dans le schéma du processus?

- X est insérée entre deux activités directement suivantes (en cas de succession).
- X est insérée entre deux ensembles de l'activité (en cas entre ensembles des nœuds) :

- Sans condition supplémentaire (en cas parallèle).
- Avec condition supplémentaire (en cas conditionnel) .

Comment est-ce que le fragment d'état changé est X ré-encadré dans le nouveau protocol?

- X est réinsérée entre deux activités directement suivantes (mouvement d'une série).
- X est réinsérée entre deux ensembles de l'activité (mouvement entre ensembles du nœud) :
  - Sans condition supplémentaire (mouvement parallèle).
  - Avec condition supplémentaire (mouvement conditionnel).

Exemple :

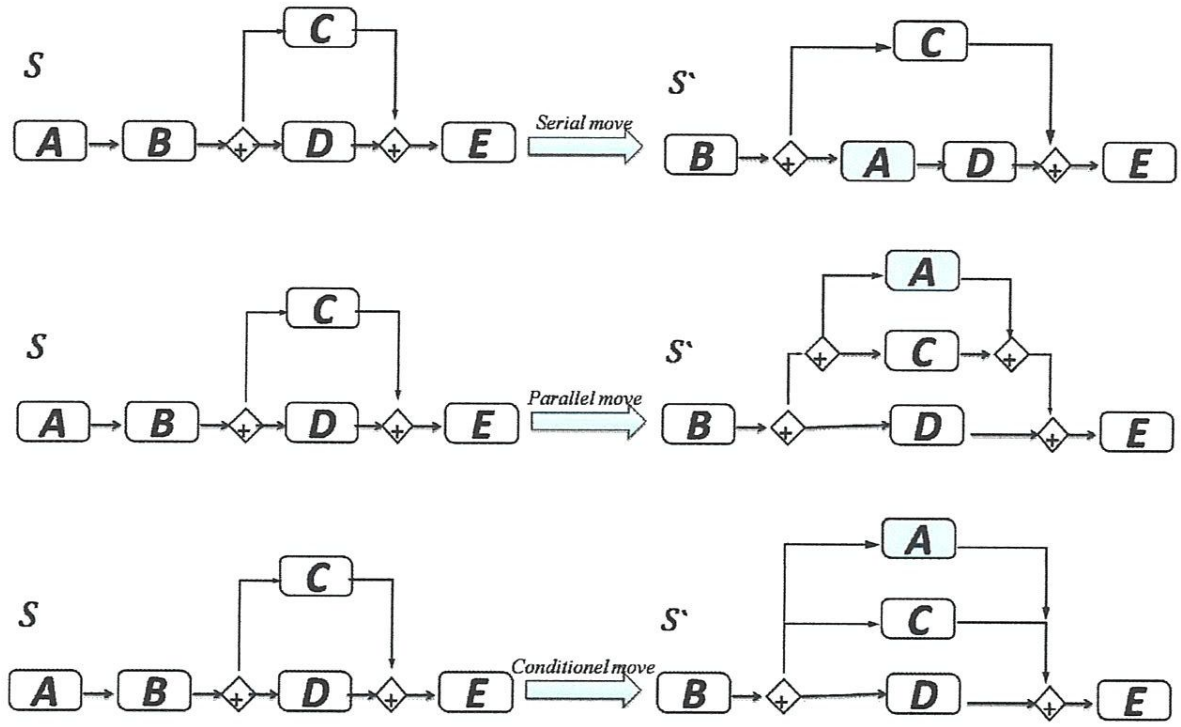


Figure 4.4 : *Déplacement d'un état dans un protocole.*

d. *Remplacement d'un état* :

Un état peut être remplacé par un autre état.

Exemple :

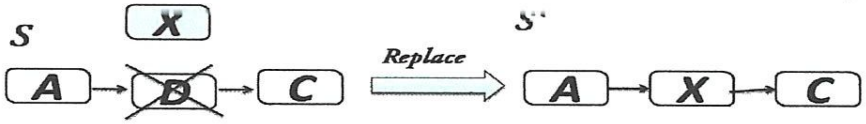


Figure 4.5 : *Remplacement d'un état dans un protocole*

e. Permutation d'état :

Deux états déjà existant dans le protocole peuvent permuter leur position.

Exemple :

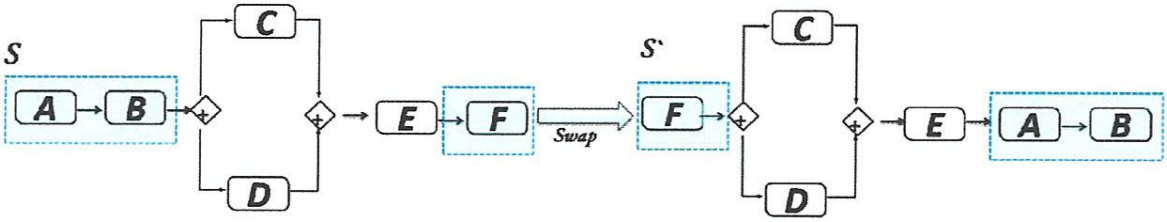


Figure 4.6 : *Permutation d'états dans un protocole.*

Ces différents types de changement doivent être présentés sous une forme qui reflète le plus fidèlement la différence résultante.

Cette représentation permettra, par conséquent de modéliser les différences (les ressemblances sont implicite) et nous pouvons utiliser les paternes afin d'exprimer les changements. [24]

1.5. Exemple de changement d'un protocole de: P, P'

Soit P un protocole d'un service Web, qui subi des opérations de changement pour donner une nouvelle version P'.

Nous utilisons comme exemple le cas de service Web relatif aux inscriptions à une formation en ligne :

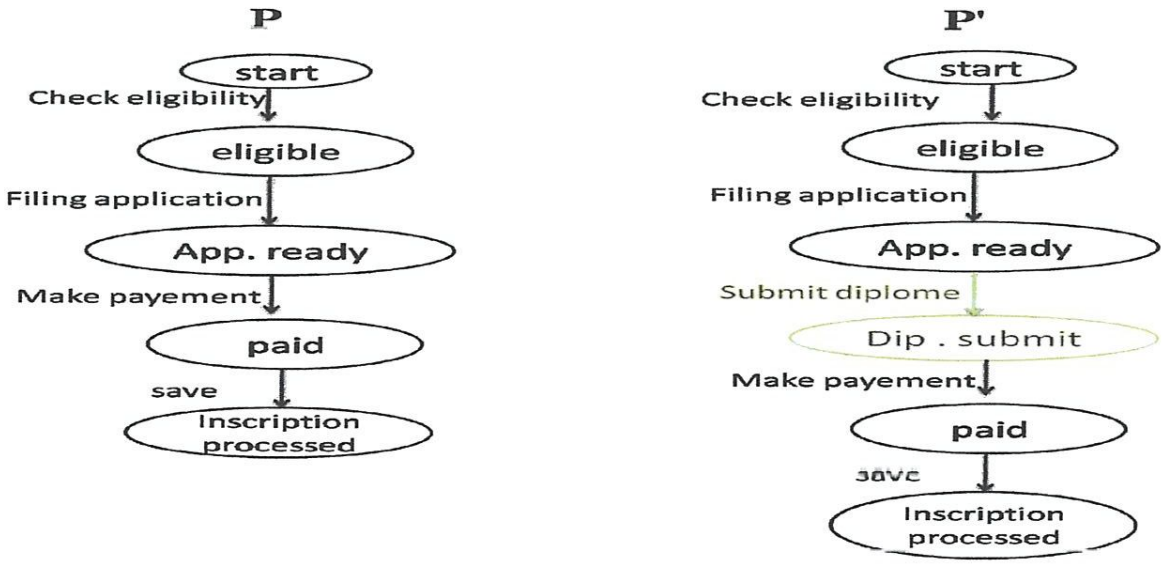


Figure 4.7 *: Exemple de changement de protocole d'inscription à une formation en ligne.*

1.6. Pattern matching et mapping

Notre travail se situe au niveau modèles de protocoles et patterns d'évolution, donc nous utilisons les concepts patterns matching et mapping.

a. Le matching

Le matching permet de vérifier si l'objet du filtrage possède une structure donnée, s'il s'y trouve telle ou telle sous-structure spécifique et laquelle, pour y retrouver des parties par l'organisation de leur contenu, et/ou pour substituer quelque chose d'autre aux motifs reconnus.

b. Le mapping

Permet de vérifier qu'une instance donnée satisfait un modèle particulier.

Nous allons utiliser ce concept pour s'assurer que les traces déjà exécutées, ou en cour d'exécution sont conformes aux changements (vérifient le pattern).

1.7. Quel modèle utiliser pour représenter les patterns de changement ?

La nécessité de formalismes visuels et simple (au lieu de logiques puissants mais souvent lourdes) lorsqu'on traite des événements de base exige une spécification de l'ensemble du comportement attendu d'un système, même sous une forme abstraite, est habituellement une tâche difficile.

Nous croyons que l'utilisation des scénarios est une stratégie clé pour faire face au problème de l'expression basées sur les propriétés des événements.

On a choisie pour la modalisation de nos patternes le Visual Timed Event Scenarios (VTS).

Nous exposons, plus tard les raisons de ce choix.

2. Visual Timed Event Scenarios VTS

2.1. Présentation de VTS :

VTS est un langage d'expression des contraintes temps réel. Il est basé sur une relation d'ordre partielle et la notion d'évènements interdits.

C'est un formalisme graphique et visuel pour représenter des scénarios interdits. Il est basé sur les éléments suivants :

- Points, arcs, annotations.
- Ordre partiel.
- Négation d'événements.
- Contraintes quantitatives de temps.

Un scénario VTS exprime les exigences temps réel d'un système, en termes de scénarios négatif. Donc en termes d'interdiction. Au lieu de représenter tout le système, des scénarios simples et parcellaires sont utilisés.

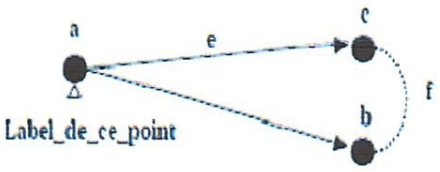
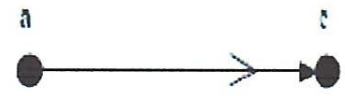
Un scénario VTS est vu comme un pattern générique qui représente des situations récurrentes.

La Sémantique de VTS est basée sur la notion de reconnaissance du scénario dans une trace. Le Matching entre scénarios et traces permet de vérifier si une fonction associant les points du scénario à certains événements de la trace, et devant posséder de bonnes propriétés.

Une trace satisfait un scénario s'il existe un matching entre eux. Donc les scénarios VTS sont des patterns représentant les situations interdites, donc qui violent les spécifications d'un système.

Il permet, donc d'exprimer les infractions aux exigences d'une évolution d'un système donné ou des violations à des contraintes de sécurité du système. [25]

2.2. Notation VTS (Formalisme):

Scénario Graphique	**Signification du Pattern**	**Exemples de Traces correctes et incorrectes**
	Ordre partiel: b,c dans le futur de a. Événements interdits: pas e entre a et c, pas f entre c et b (quel que soit leur ordre d'occurrence)	a x y z c y b : scénario reconnu : correcte … a c x y z f b .. scénario non reconnu : non correcte
 Ou   Ou	Premier c après a. Pas de boucle sur c. Dernier a avant c Pas de boucle sur a	aac : Reconnue :correcte acc: non reconnu : non correcte accc : Reconnu : correcte aac : non Reconnu : non correcte

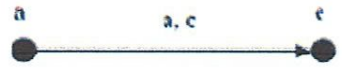
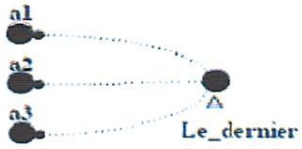
		
 Ou 	a et c sont interdits entre premier a et le premier c. A et c sont consécutifs.	…a xaxyzcc.: reconnu : correcte …axaxyzacc….:non correcte
	Premier b ou premier c après a.	ab, ac : Reconnu : correcte abc : Non reconnu : non correcte
	1er a de l'exécution	
	Dernier a de l'exécution	
	Le premier d'une exécution	a1 x y z a3 c a2 **a1 : le premier**
	Le dernier d'une exécution	… a1 x y z a3 c a2 a2 : Le dernier

Tableau 4.1 : *Tableau du formalisme VTS*

2.3. Evaluation du formalisme VTS :

a. Avantages de VTS :

- Un point fort de VTS, par rapport aux AFD est l'expression de l'interdiction (Forbidden events) qui permet d'exprimer les transitions interdites entre deux ou plusieurs évènements.

- En plus, les contraintes temporelles entre deux évènements peuvent êtres exprimées aisément.
- Le nombre d'évènement peut être limité en jouant sur le premier et le dernier évènement (pas de boucle).
- Expression du premier et du dernier évènement (the first and the last).
- Procédure pour construire un automate qui reconnait le scénario.
- Connexion à des outils de model-checking.
- Sémantique clair et intuitive basée sur la notion de reconnaissance du scénario dans une trace.
- La possibilité de proposer une stratégie de migration pour chaque pattern.

b. Points faibles de VTS :

- L'algorithme de construction de l'automate n'est pas donné explicitement.

2.4. Le matching en VTS :

Toute la sémantique de VTS est basée sur le matching entre scénarios et traces d'exécution. On dira qu'un point d'un scénario P, peut matcher une position dans une trace si, l'évènement correspondant à cette position est un évènement parmi ceux qui sont permis par la fonction d'étiquetage associés à ce point.

Propriétés du matching scénarios-traces

Si on trouve une fonction telle que:

• **M1:** Tout point p du scénario étiqueté par un ensemble d'événements l(p) doit être associé à une position i de la trace telle que si ∈ l(p) .

• **M2:** Deux points concrets (i.e., qui ne représentent pas le premier ou dernier d'un ensemble de points) ne peuvent avoir la même image.

• **M3:** Si p < q dans le scénario (arc orienté entre les deux), alors p < q dans la trace.

• **M4:** S'il y a des événements interdits entre p et q (arc annoté avec événements interdits), alors ces événements n'apparaissent pas dans la sous-séquence $s_{p+1} \ldots s_{q-1}$.

• **M5:** S'il y a des événements interdits entre début et p, alors ces événements n'apparaissent pas dans la sous-séquence $s_1 \ldots s_{p-1}$. De même, s'il y a des événements interdits entre p et fin, alors ces événements n'apparaissent pas dans la sous-séquence $s_{p+1} \ldots s_{[|s|]}$.

• M6 et M7 relatives aux contraintes temporelles quantitatives.

• **M8:** Si p représente le premier d'un ensemble de points {ri}, alors p = min (ri). De même, si p représente le dernier d'un ensemble de points {ri}, alors p = max (ri).... alors la trace satisfait le scénario.

Au niveau traces d'exécutions:

Il est évident que tout système va générer un ensemble (peut être infini) de traces. Les traces sont classées en deux catégories :

- Celles qui vérifient le Scénario VTS : Donc elles sont conformes aux propriétés de l'interdiction, alors interdites par le système.
- Celles qui violent le scénario VTS : Donc ne vérifient pas les interdictions et par conséquent sont conformes au système.

Exemple de matching: [25]

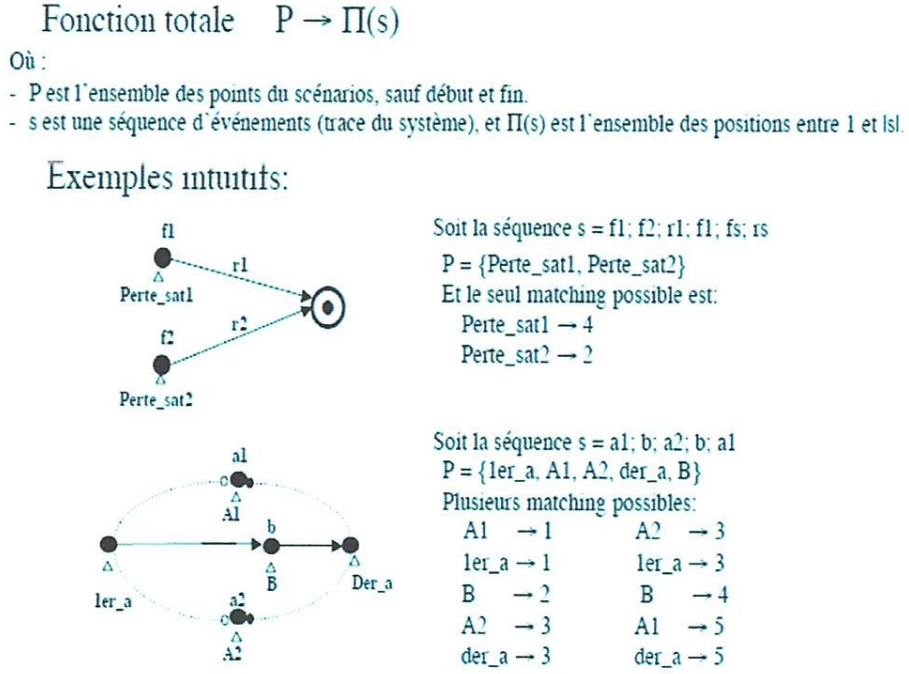
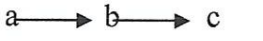
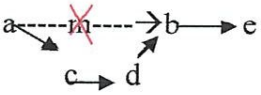
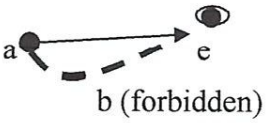
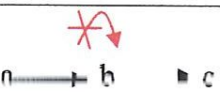
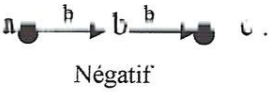
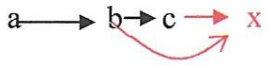
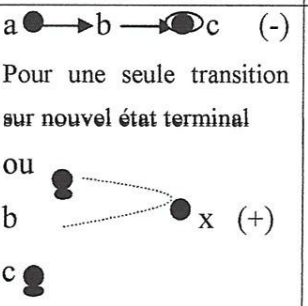
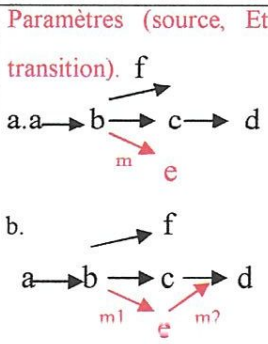
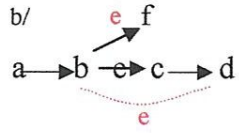
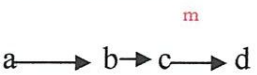
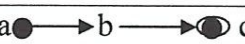
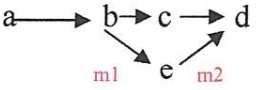
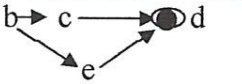
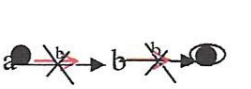


***Figure 4.8** : Exemple du matching.*

3. Présentation de l'évolution avec les patterns en VTS

Actions sur les patterns / *Référence Pattern*	*Opération de changement*	*Exemple*	*Pattern VTS sénario Négatif (forbidden)*	*Remarques*
Pattern 1	Suppression d'état Terminal	a----→b		La trace a suivi de b (final) ne doit pas figurer dans le système.
Pattern 2	Suppression d'état Intermédiaire		Préserver la cohérence et revenir à 1.	La trace a suivi de b ne doit pas figurer dans le système. Donc a et c ne sont pas séparés par b.
Pattern 3	Suppression d'une transition			La trace a suivi de e ne doit contenir la transition b à l'intérieur (cible du message m supprimé).
Pattern 4	Suppression de boucle sur un état. *Si b se produit, il n'y*		 Négatif	Les flèches étiquetées par des noms d'evnts indiquent que ces events sont

	aura pas d'autres b qui seront activés.			interdits de b (pas d'autres b entre a et b pas de b entre b et c: ab* ni b*c). (pas de ab*c).
Pattern 5	Ajout d'un état en fin de protocole			Les doubles cercles concentriques indique que c est un état final. Ni b ni c sont terminaux mais x.
Pattern 6	Ajout d'un état en milieu de protocole Engendrant une seul transition Engendrant 2 transition (ou +)			e ne doit pas figurer après b et avant les états cibles de b.
Pattern 7	Ajout d'une transition en fin de protocole			Ajout de message, suite à l'ajout d'états (un à la fin ou deux messages au milieu)
Pattern 8	Ajout d'une transition en milieu de protocole			L'état b est suivi de c ou de e.
Pattern 9	Ajout d'une boucle sur un			La première flèche ouverte de a vers b

	état	VTS avant boucle: avant boucle : +ieurs b peuvent être avant c.	le premier b qui est avant c, et c est le seul qui suit b.	indique une exécution unique de la séquence a,b, donc elle ne doit pas figurer dans le scénario.

***Tableau 4.2** : Tableau des différents types d'évolution des patterns VTS.*

VTS permet aussi d'exprimer les contraintes temporelles.

Nous nous n'intéresserons pas aux patterns associées par manque de temps

4. Analyse de l'impact de l'évolution sur les instances actives :

Une des questions les plus provocatrices dans la modification d'un protocole est diriger les utilisateurs du protocole en cour par rapport à l'ancien protocole.

Avorter simplement ou annuler tout les instances actives n'est pas une solution adéquate.

Donc, il est impératif de déceler et de filtrer, parmi les instances en cours, celles qui sont conformes aux changements de celles qui ne le sont pas .

On parle de migration de traces ou d'instances actives.

4.1. Base des traces :

Une trace c'est l'historique des événements engagés par une instance depuis son déclenchement jusqu'à son état actuel.

Formellement c'est le chemin d'exécution d'un protocole du début jusqu'à l'état courant.

L'assemblage des trace d'instance d'un système nous donne une base de données des traces ; qui sont enregistrées par le système (journaux logs).

Par exemple : Les traces d'instances utilisant le protocole de l'exemple figure 4.7 :

Num.	*Instances*	*Trace*
1	Gerard Nguessan	Chek iligibility ; Filing application;
2	Mohamed Fadel	Chek iligibility ; Filing application; submit diplome; make payement;
3	Kenouz Saliha	Chek iligibility ; Filing application; make payement; save;
4	Azzouz Samah	Chek iligibility ; Filing application; submit diplome; make payement; save;

***Tableau 4.3 :** Exemple de base d'instances*

4.2. Analyse d'impact de l'évolution d'un protocole sur les traces :

Il est nécessaire de fournir des services avec la possibilité d'évoluer et la capacité de minimiser l'impact d'une telle modification et éviter une réexécution de tout le protocole et l'interruption des instances en cours qui est indésirable pour les raisons suivantes :

- les transactions dans les environnements ouverts et dynamique tel que les services Web sont de longue durée, alors une modification ou une réexécution de protocole impliquent une grande perte de travail.
- Une utilisation non rentable des ressources soit : le temps, l'argent, la bande passante …
- La gestion manuelle des utilisateurs (leurs traces) est irréalisable puisque chaque un doit reprendre le protocole d'un état spécifique et il y'a un nombre important et croissant d'utilisateur pour chaque protocole.

Pour ces raisons, il est impératif de mener une réflexion profonde sur le sort des instances actives :

Peuvent- elles continuer leur exécution, tout en restant conformes aux modèles de protocoles déjà sélectionnés ?

La continuité de l'exécution peut-elle engendrer des incohérences et des incompatibilités ?

Pour répondre à ces deux questions notre démarche est la suivante :

Démarche :

1- Partir d'une ancienne version de protocole.
2- Enregistrer les exécutions des instances actives.
3- Opérer les divers changements (opération de mise à jour).
4- Extraire le pattern d'évolution associée et le représenter en VTS.
5- Faire le mapping entre patterns et traces :
   - Les traces qui vérifient le pattern ne sont pas migrable.
   - Les traces qui ne respectent pas le scénario du pattern sont migrable.
6- Donnez des statistiques (outil d'aide à la décision) pour le gestionnaire de protocole.

5. Les stratégies de migration:

Dans ce qui suit, on va aborder l'analyse des stratégies de migration des instances actives associées aux patterns d'évolution, déjà identifiées précédemment.

Cette analyse est basée sur le matching qui est la recherche de correspondance entre un ensemble d'occurrences (traces) et un modèle déjà conçu (pattern).

Patterns	***Migration***	***Explication***
Pattern 1 : Suppression d'état Terminal	Non migrable	Si l'exécution de a est terminé alors le pattern est non migrable ; si non il est migrable
Pattern 2 : Suppression d'état Intermédiaire	Non migrable	Si l'exécution de a est terminé et il a passé à c par b alors le pattern est non migrable ; Sinon il est migrable
Pattern 3 : Suppression d'une transition	Non migrable	Si l'exécution de a est terminé et il passe à e par b alors le pattern est non migrable ; Si non il est migrable
Pattern 4 : Suppression de boucle sur un état.	Non migrable	Si l'exécution de a et b est terminé et il boucle sur b alors le pattern est non migrable ; Si non si il passe directement à c après le premier b alors le pattern est migrable.
Pattern 5 : Ajout d'un état en fin de protocole	Migrable	L'ajout d'un état a la fin ne peu pas influencé l'exécution du protocole.
Pattern 6 : Ajout d'un état en milieu de protocole	Non migrable	Si l'exécution de b est terminé et se suit d'un e quelque soit l'état suivant, le pattern est non migrable ; Si non le pattern est migrable.
Pattern 7 : Ajout d'une transition en fin de protocole	Non migrable	Si b est suivie directement de c par un seul b alors le pattern est non migrable. Si non le pattern est migrable.
Pattern 8 : Ajout d'une transition en milieu de protocole	Migrable	b peut êtres suivie de c ou de e
Pattern 9 : Ajout d'une boucle sur un état	Migrable	b peut être exécuté une ou plusieurs fois

Tableau 4.4 : *Tableau des stratégies de migrations proposées.*

6 .Architecture du système proposé :

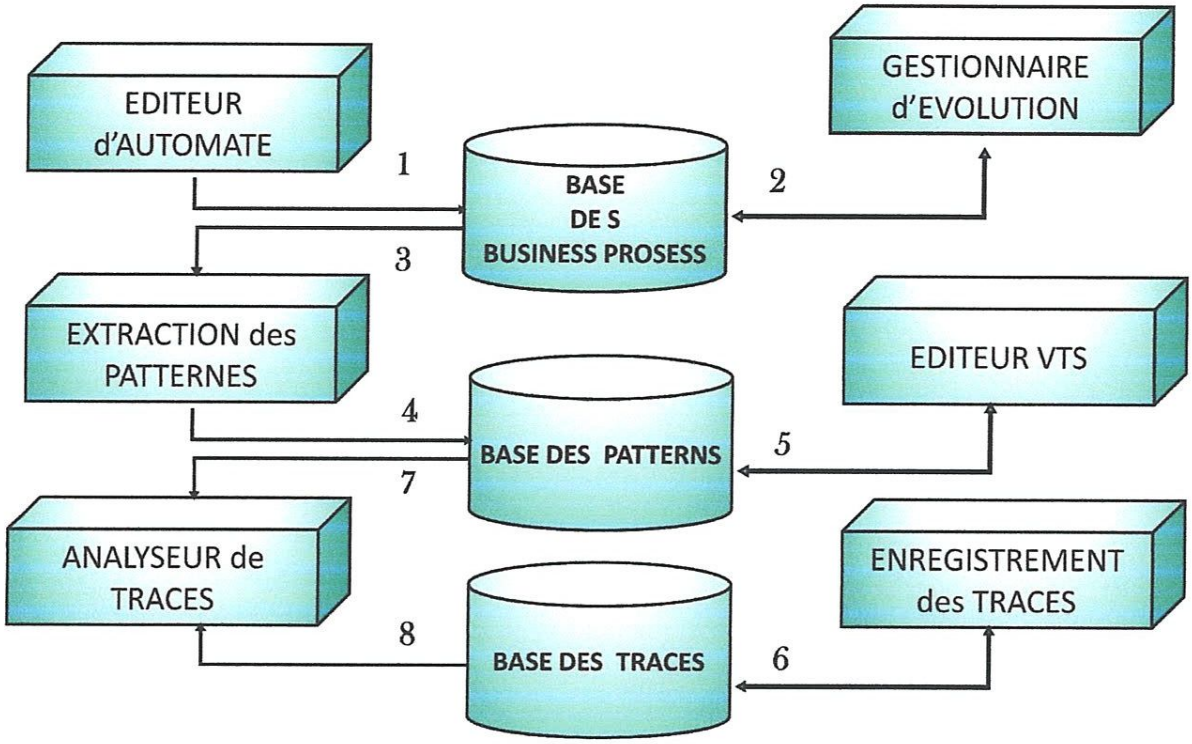


Figure 4. 9 :*L'architecture du système proposé.*

Notre système s'articule au tour de trois bases de données, qui sont les suivantes :

a. Base de données des BP : Contient tous les protocoles relatifs aux services Web.

Ils sont stockés sous forme de fichiers XML.

b. La base des traces : Contient toutes les traces décrivant le chemin des instances en cours d'exécution des BP de la base de données précédente. Chaque BP peut être exécuté par un nombre quelconque de clients, donc il génère un nombre correspondant de traces.

c- La base des patterns : contient touts les motifs d'évolution extraite de la comparaison entre un protocole et sa version évoluée.

Les transformations et interactions des modules logiciels sont détaillées ci-dessous :

1: l'interface offre un module permettant de créer, de décrire et de stocker l'automate du BP en question dans la BDD des automates.

2 : l'interface offre un module qui permet d'opérer un changement précis dans un protocole en évolution.

3 : Le module principal de ce système permet de charger un protocole de la base des BP et le comparer avec ca version d'évolution.

4 : Après l'extraction des patterns d'évolution, ces derniers seront stockés dans la base des patterns sous forme de fichier XML.

5 : Ce système offre un éditeur graphique manuel pour concevoir des scénarios VTS, qui seront stockés sous forme d'un fichier XML.

6 : L'interface permette à l'utilisateur d'alimenter la base des traces avec les utilisateurs des protocoles du service, en enregistrant la trace d'exécution.

7-8 : L'interface propose un analyseur de traces qui prend un pattern d'évolution en analysant son impact sur une instance et donne le résultat de cet impact.

Conclusion :

Dans ce chapitre nous avons présenté les différents concepts qu'on a utilisé pour cette étude, on commençant par les patterns d'évolution ensuite leur présentation on modèle VTS, puis nous avant proposé notre stratégie de migration par apport a ce model pour chaque pattern présenté on se basant sur l'étude d'impacte de chaque changement sur l'instance en cours.

Pour le résultat des stratégies de migration il y'a deux cas :

- Les instances migrable : Peuvent continuer leur exécution dans le nouveau protocole sans interruption.
- Les instances non migrable : ils doivent subir des changements pour leur adaptation au nouveau protocole avec des adaptateurs spécifiques.

Chapitre V :

Implémentation

Introduction :

Dans le but de présenter des solutions concrètes aux problèmes expliqués précédemment et selon l'analyse qu'on a faite dans le chapitre IV, nous aborderons dans ce chapitre l'aspect pratique relatif à l'implémentation de notre conception.

Dans un premier temps, nous avons programmé une implémentation des protocoles métiers représentés par des automates finis déterministe, avec la possibilité d'une représentation graphique des automates avec la possibilité de modification et de mise à jour de ces protocoles.

En suite, nous avons réalisé un algorithme de comparaison et d'extraction du changement et qui sont présentés et stockés sous forme de fichier XML.

Cette implémentation a conduit à la réalisation d'un outil logiciel d'aide à la gestion d'impact de l'évolution des protocoles de services Web sur les instances en cour d'exécution, et par la suite distinguer entre celles qui peuvent migrer dans les nouvelles version des protocole et celles qui doivent subir une adaptation avant d'être migrées.

Il s'agit de mettre en œuvre une application qui a pour principaux rôles de :

1. Créer et sauvegarder un ensemble d'automates représentant les protocoles en format *XML*.
2. Faire la consultation d'un automate et procéder à son éventuelle évolution (*modification du schéma*).
3. Affichage graphique des automates, avec la possibilité de faire des mises à jour de ces automates (*suppression, ajout, évolution, …etc.*).
4. Extraire les patterns d'évolution des différentes versions des protocoles.
5. La représentation de ces patterns sous forme de scénario VTS.
6. Générer les chemins d'instance en cour d'exécution (jeu d'essai simulé).
7. Analyse d'impact de l'évolution des protocoles sur les instances en cour.

1. Présentation de l'environnement de développement :

Nous avons choisi d'utiliser l'environnement de développement JAVA, alors il est intéressant d'indiquer les deux principaux choix technologiques que nous avons faits, à savoir le langage de programmation Java et la plateforme Eclipse IDE.

Nous avons utilisé le langage Java SE 7 pour le codage de toutes les classes de notre application. Le langage Java a été choisi pour obtenir une plus grande portabilité, en restant prêt de nombreux outils dans le domaine des services Web, développés principalement en Java. Les composantes de cette plateforme sont développées de manière très modulaire et

générique. Par ailleurs, l'intégralité des fonctionnalités actuelles et futures de cette plateforme sera exposée sous forme de plug-ins Eclipse IDE.

Le choix de la plateforme Eclipse IDE est justifié par le fait que c'est un environnement de développement intégré (IDE) libre, extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.

Nous avons utilisé une machine simple pour implémenter nos algorithmes en java qui n'ont pas des calculs importants. La machine c'est un portable avec un processeur Intel-Pentium-Dual core, une RAM de 4GB doté du système Windows 7.

2. Description de l'application :

2.1. Présentation de l'Interface et fonctionnalités :

L'interface principale de notre application est constituée des principales fonctions représentées par le menu principal suivant :

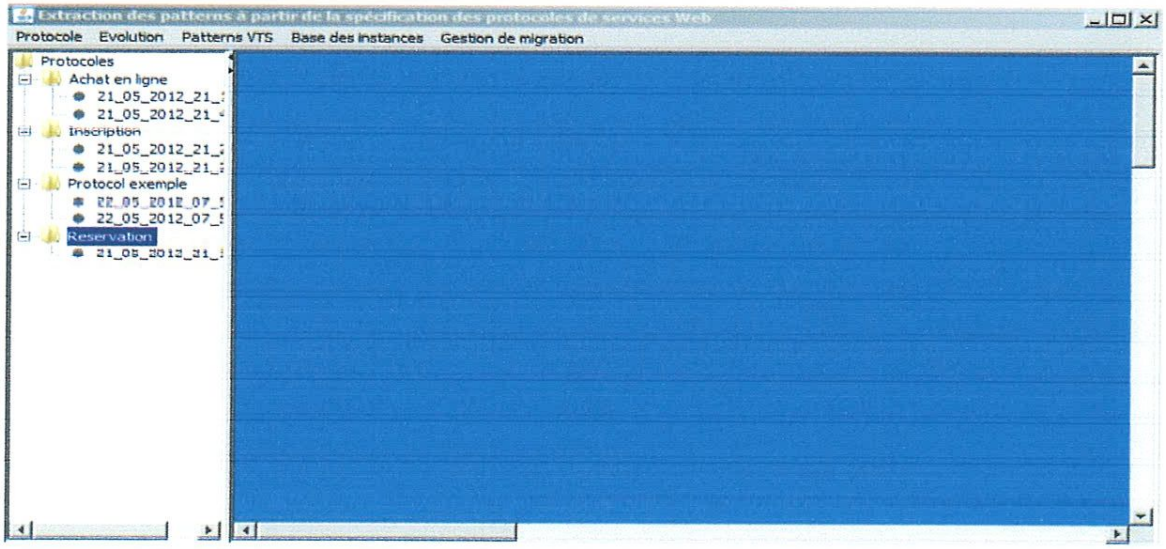


Figure 5.1 *: Interface principale de l'application.*

2.2. Menu Protocole :

Ce menu se compose de cinq fonctionnalités :

Figure 5.2 : *Menu Protocole.*

Ce menu se compose de cinq fonctionnalités :

a. ***Nouveau :*** Permet de créer un nouveau protocole, cette option a comme entrée le nombre d'état du protocole et le nombre de ses transitions et nous donne la main de les nommer.

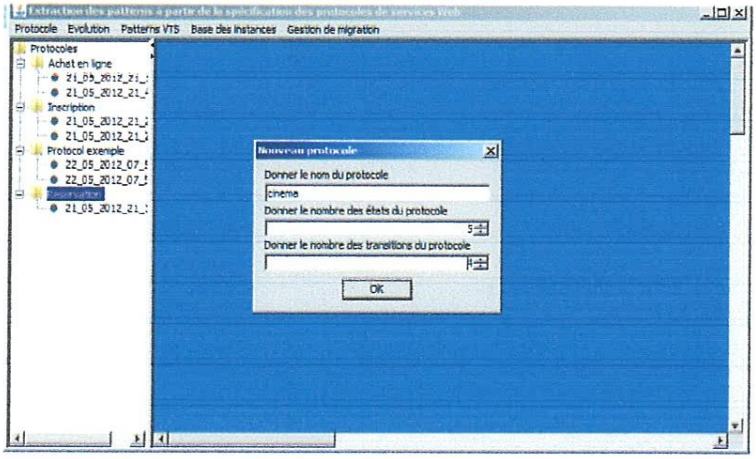


Figure 5.3 : *formulaire de saisie des éléments d'un Nouveau Protocole.*

Après on peut choisir les relations de notre protocole grâce au tableau suivant :

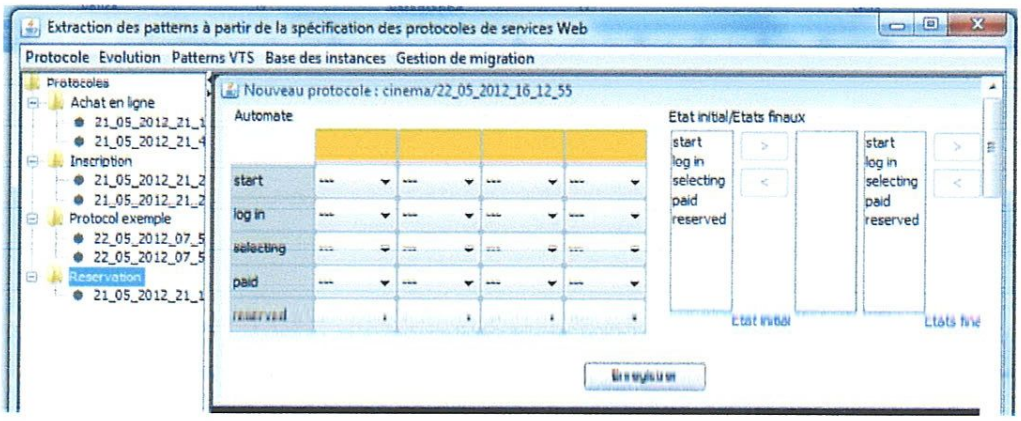


Figure 5.4 : *Formulaire de création d' un Nouveau protocole.*

b. ***Ouvrir:*** Permet d'ouvrir un protocole déjà existant ; comme on peut accéder à cette fonctionnalité par la fenêtre à gauche.

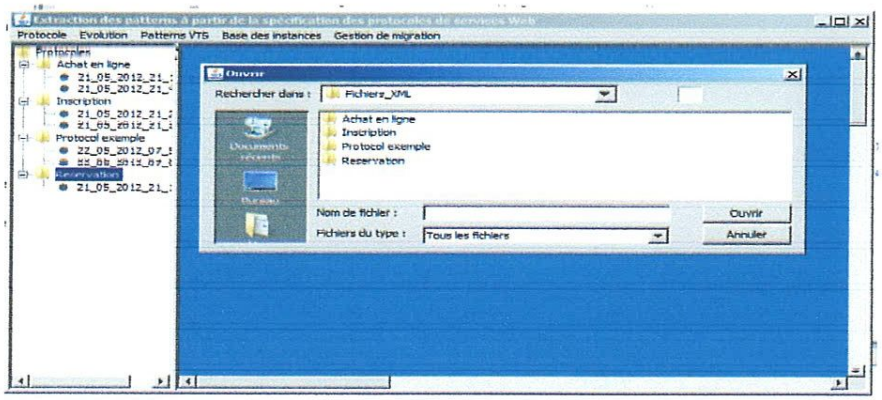


Figure 5.5 : *chargement d'un protocole existant.*

Après la sélection d'un protocole on obtient un affichage qui présente sa structure plus sa visualisation en AEF.

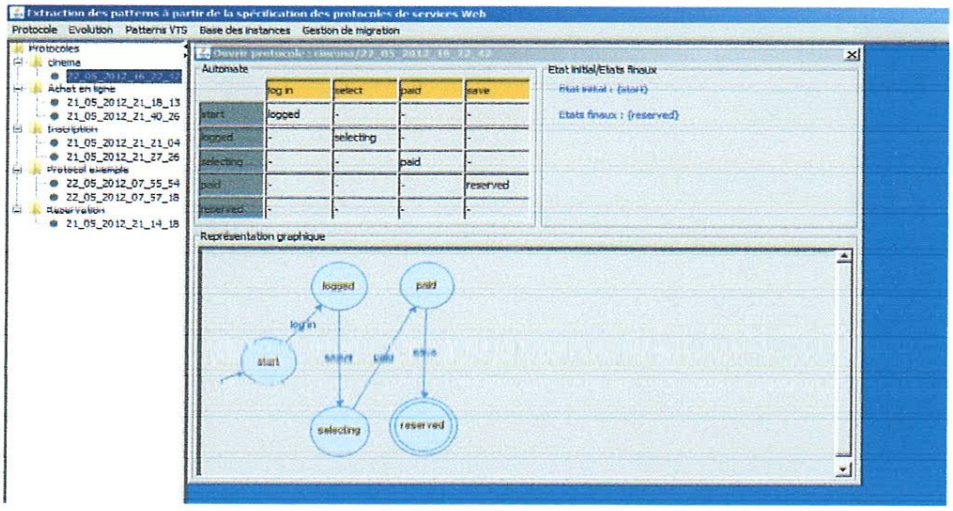


Figure 5.6 : *Affichage d'un protocole existant sous forme graphique.*

c. ***Modifier :*** Cette fonctionnalité permet d'établir des modifications sur les protocoles déjà existants. Une boite de dialogue est afficher pour choisir le protocole et la version sur la quelle on veut faire des modifications :

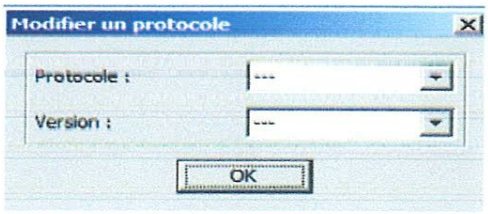


Figure 5.7 : *Boite de dialogue Modifié.*

Après le même tableau d'affichage de protocole apparait mais on donne la main à l'utilisateur pour modifier le protocole par les boutons suivants :

Ajouter état, Supprimer état Pour l'ajout et la suppression des états,

Ajouter étiquette , Supprimer étiquette Pour l'ajout ou la suppression d'une transition et

Mettre à jour Pour valider les modifications apporter au protocole.

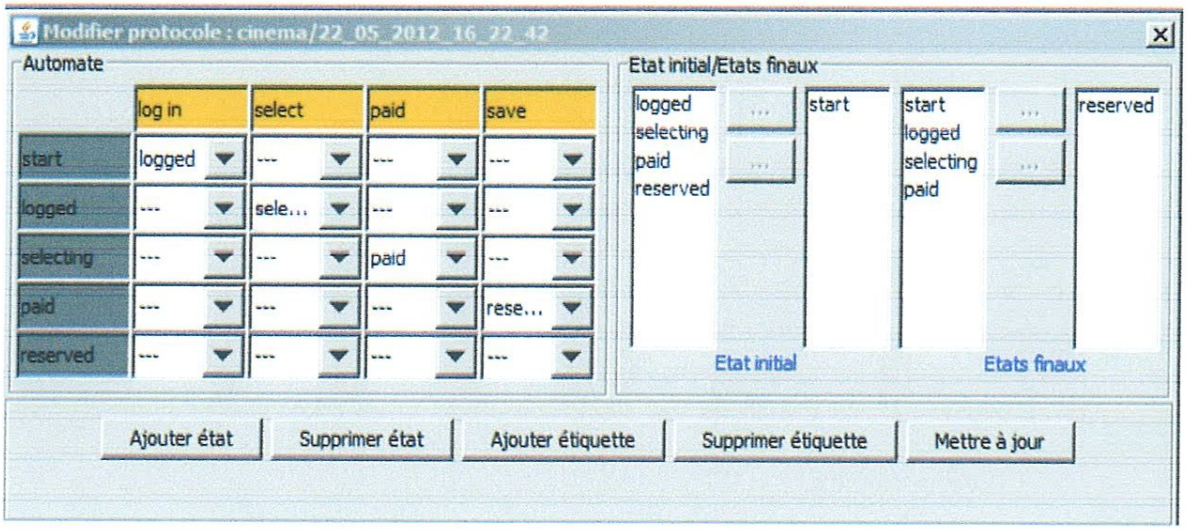


Figure 5.8 : *Modification de la description d'un protocole.*

d. ***Supprimer :*** Cette fonction nous donne la possibilité de supprimer un protocole entier ou une seule version d'un protocole

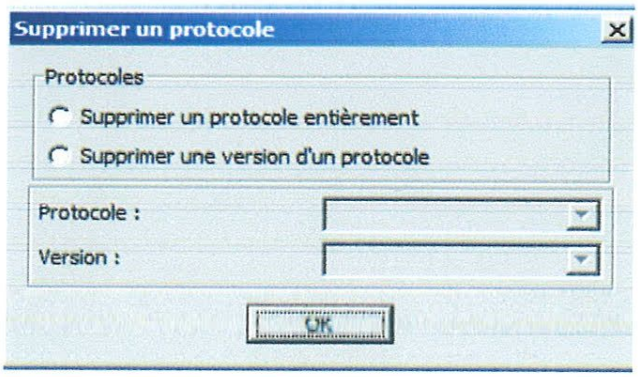


Figure 5.9 : *Suppression un protocole*

e. ***Quitter :*** Pour quitter l'application.

2.3. Menu Evolution :

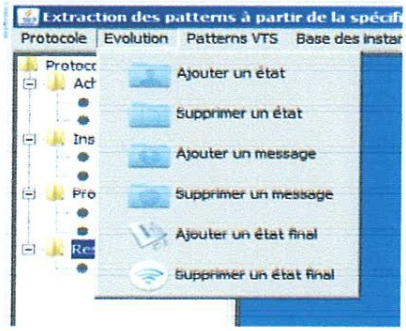


Figure 5.10 *: Menu Evolution*

Dans ce menu il existe six fonctions, chacune permet d'opérer un changement précis dans un protocole, en donnant la main à l'utilisateur de choisir le protocole et ses deux version sur les quelles il veut effectuer la comparaison :

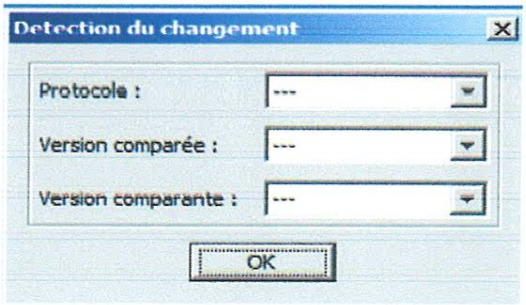


Figure 5.11 : *Choix des versions de protocoles a comparé.*

Selon la fonction choisie dans le menus la comparaison donnera le résultat dans un message soit qu'il existe un changement de ce type et le quel ou la négation dans le cas contraire.

2.4. Menu Patterns VTS :

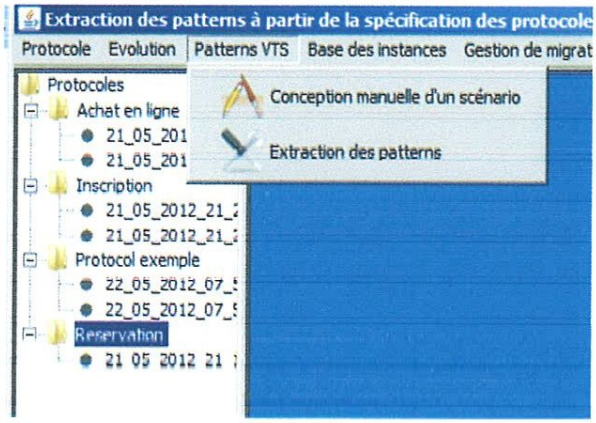


Figure 5.12 : *Menu Patterns VTS*

Ce menu se compose de deux fonctions :

a. Conception Manuelle d'un scénario :

C'est un éditeur graphique, qui permet de concevoir des scénarios VTS manuellement :

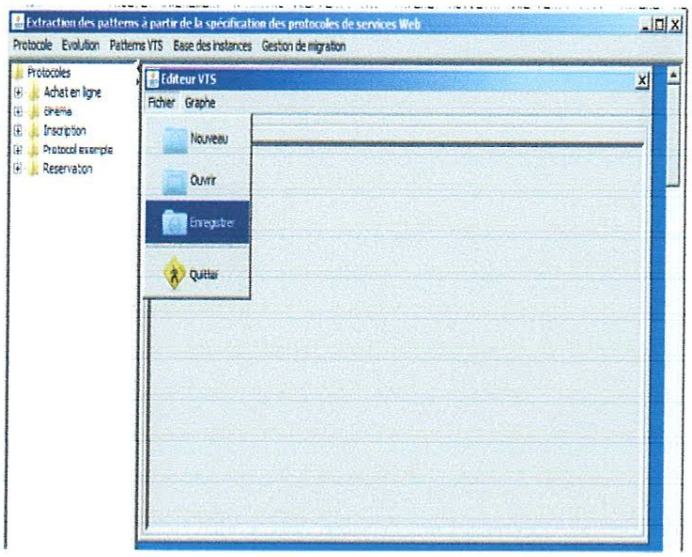


Figure 5.13 : *Editeur VTS pour les patterns.*

Son interface est composée de deux boutons :

- ***Fichier*** : On trouve les fonctions 

, 

, 

et 

.
- ***Graphe :*** C'est de la qu'on peut concevoir notre scénario par choisir ces différents composants :

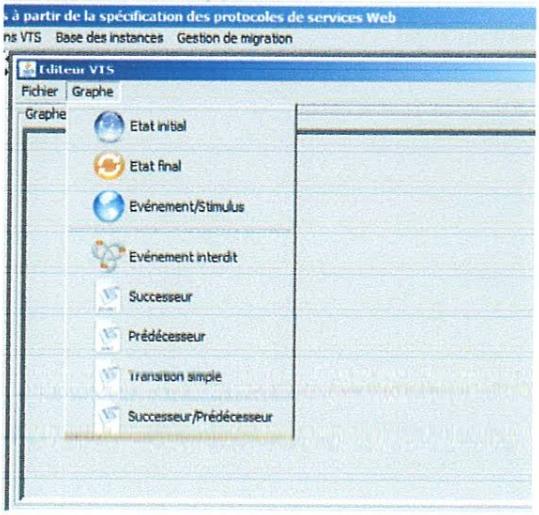


Figure 5.14 : *Les composants d'un pattern VTS*

Le scénario est donc présenté par la combinaison des composants suivants :

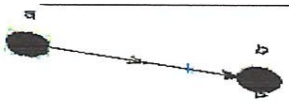
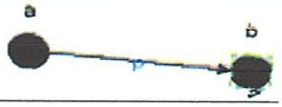
Élément	*Description*
Un état Initial	
Un état final	
Un événement stimulus	
Un événement interdit	
Successeur	
Prédécesseur	
Transition simple	
Successeur-Prédécesseur	

Tableau 5.1 : *Éléments VTS*

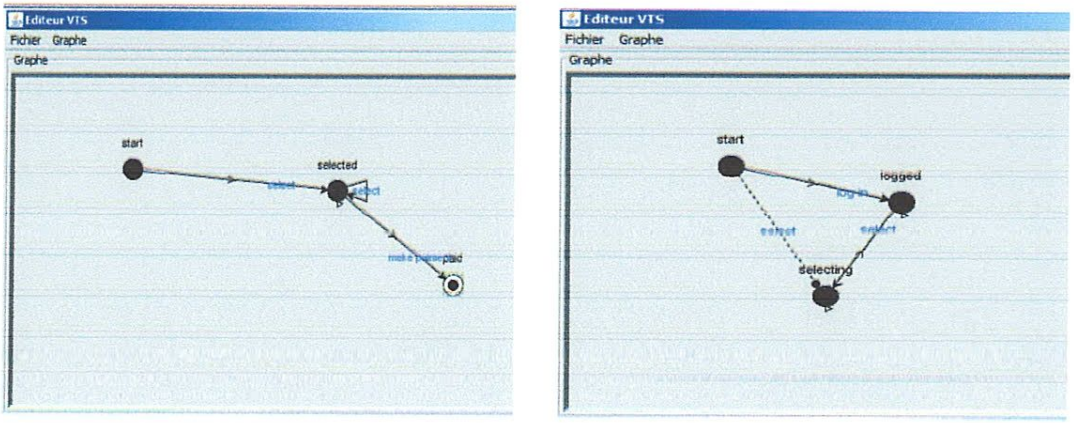


Figure 5.15 : *Conception d'un scénario VTS manuellement.*

b. Extraction des Patterns

En premier on a une fenêtre de choix qui permet de sélectionner le protocole et les versions a comparer et donner un nom au pattern extrait :

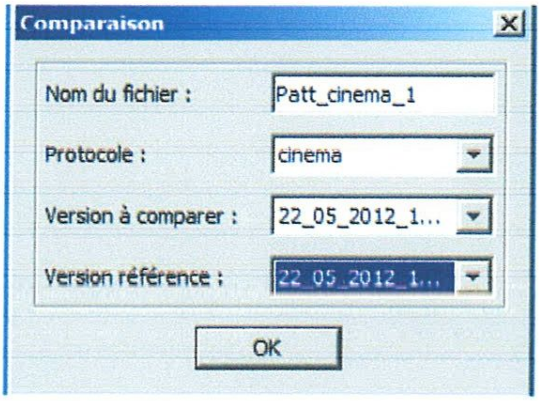


Figure 5.16 : *Extraction d'un pattern.*

Le résultat de la comparaison sera affiché dans un message est les données seront stockées dans un fichier XML dans la base des patterns:

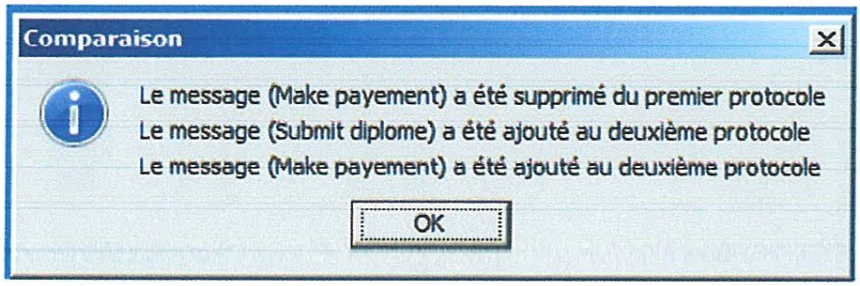


Figure 5.17: *Message résultat d'extraction des patterns.*

2.5. Menu Base des instances :

Ce menu est établi pour créer les instances et les stocker dans une base de données :

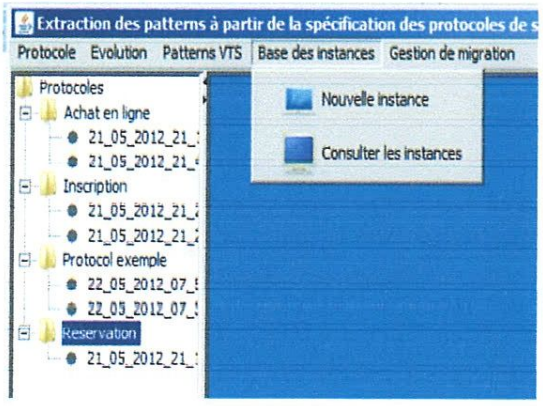


Figure 5. 18 : *Menu Base des instances*

Il engendre deux fonctionnalités :

a. ***Nouvelle instance :*** Permet de créer de nouvelles instances et leur affecter le chemin d'exécution correspondant.

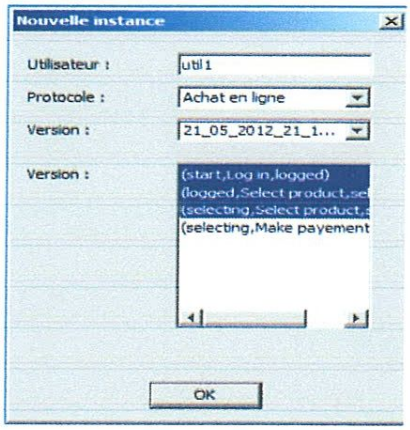
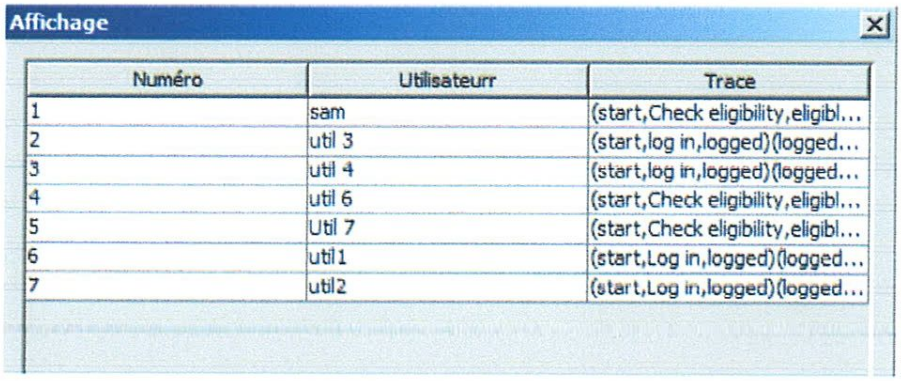


Figure 5.19: *Création d'une nouvelle instance*

c. Consulter les instances en cours :

Permet de nous afficher touts les instances en cours avec le chemin qu'elles ont déjà exécuté.



Numéro	Utilisateurr	Trace
1	sam	(start,Check eligibility,eligibl...
2	util 3	(start,log in,logged)(logged...
3	util 4	(start,log in,logged)(logged...
4	util 6	(start,Check eligibility,eligibl...
5	Util 7	(start,Check eligibility,eligibl...
6	util1	(start,Log in,logged)(logged...
7	util2	(start,Log in,logged)(logged...

Figure 5.20 : *Affichage du contenu de la Base des instances actives.*

2.6. Menu Gestion de la migration :

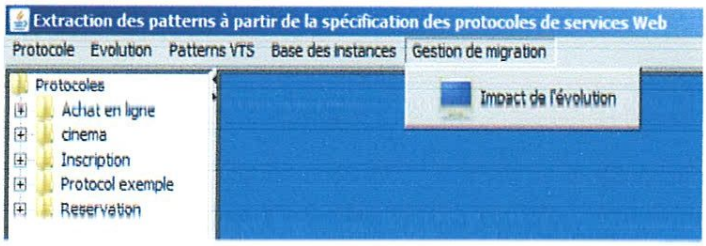


Figure 5.21 : *Menu Gestion de la migration.*

Ce menu contient la fonction d'analyse des instances choisies et les comparer avec le pattern d'évolution du protocole exécuté et avoir comme résultat un message qui précise si elle est migrable ou non.

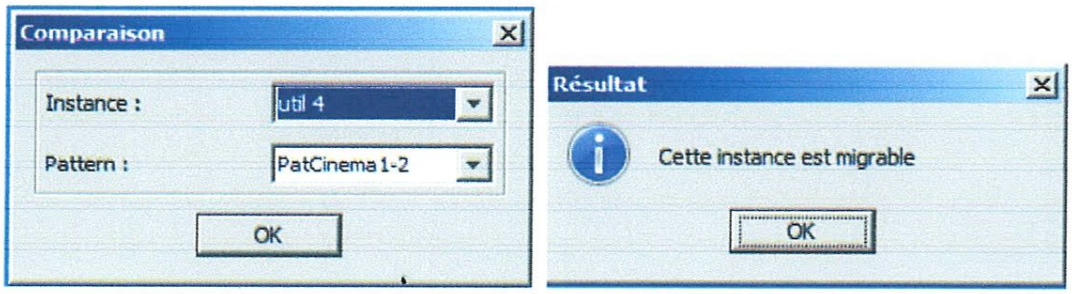


Figure 5.22 : *Analyse d'instance et résultat sur la migration.*

3. Les structures de données manipulées :

Nous avons choisi d'enregistrer les protocoles (automates) dans un fichier au format XML, afin de simplifier la manipulation des données existantes dans chaque automate.

3.1. Pour les protocoles :

Le document XML contient un élément racine "*<Automate>*", l'élément racine contient trois éléments fils. L'élément *<Etats>* ,l'élément*<Transitions> et l'élément<Etiquet>* .

- L'élément *<Etats>* : comporte trois attributs *<Nom>,<Etat_Initial> et <Etat_Final>*.
- L'élément *<Transitions>* : comporte trois attributs *<Debut>,<libelle><Fin>*.
- L'élément *<Etiquet>* : comporte l'attribut <Intitule>.

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<Automate>
 <Transition>
  <Debut>start</Debut>
  <Libelle>log in</Libelle>
  <Fin>logged</Fin>
 </Transition>
 <Etat>
  <Nom>start</Nom>
  <Etat_Initial>oui</Etat_Initial>
  <Etat_Final>non</Etat_Final>
 </Etat>
 <Etat>
  <Nom>logged</Nom>
```

```
  <Etat_Initial>non</Etat_Initial>
  <Etat_Final>oui</Etat_Final>
 </Etat>
 <Etiquete>
  <Intitule>log in</Intitule>
 </Etiquete>
</Automate>
```

3.2. Pour les instances et les patterns :

Le document XML contient un élément racine "*<Instance>*", l'élément racine contient un élément fils *<Transitions>*.

L'élément *<Transitions>* : comporte trois attributs *<Debut>*,*<libelle>**<Fin>*.

3.3. Pour les scénarios VTS :

Le document XML contient un élément racine "*<VTS>*", l'élément racine contient un élément fils *<Attributs>*.

L'élément *<Attributs>* : comporte quatre attributs *<Nom>*,*<x>*,*<y>* *et* *<Type>*.

Conclusion

La phase d'implémentation été une étape critique durant le déroulement de notre projet.

Elle nous a permet de consolider l'approche proposée et de s'assurer de sa faisabilité

Le système proposé dans la phase conception a été développé, les algorithmes programmés et les bases de données créées et alimentées avec des jeux d'essai réels.

Cependant, nous avons trouvé des difficultés pour la génération automatique des patterns VTS à partir des deux versions de protocoles et leur représentation graphique.

En résumé, nous pouvons affirmer que notre système fonctionne d'une manière satisfaisante, bien que des raffinements restent à faire.

Conclusion générale

Conclusion générale

Dans ce mémoire, nous avons traité un sujet d'actualité relatif à une technologie d'intégration des systèmes d'information ouverts et distribués sur le Web qui est la technologie des services Web.

Nous nous sommes intéressés, particulièrement à l'évolution des protocoles exécutés dans ces services Web et l'impact de leur changement sur les utilisateurs de ces services.

Notre attention a été axée sur la conception d'un outil qui facilites la gestion des différents protocoles en évolution et à donner une sorte d'outil d'aide à la décision concernent l'instance en cours d'exécution.

Pour cela, nous sommes partis d'un modèle de protocole en évolution et nous avons analysé et comparé ces différentes versions pour arriver à modéliser ce changement dans un format standard.

Ce format nous a permet et faciliter l'analyse de l'impact de ces changements sur les instances en cour d'exécution.

Cette analyse nous a permis de distingué les instances qui peuvent migrer de celles qui doivent subir une adaptation avant la migration.

Au terme de ce projet de fin d'étude nous pouvons dire que les acquis capitalisés par la conduite de ce projet sont :

1. Nous avons appliqué et maitrisé la méthodologie scientifique pour la conduite d'un projet.
2. Nous avons découverts la technologie des services Web et nous nous sommes familiarisés avec les standards y afférents.
3. Nous avons pris conscience des problématiques liées à l'intégration des systèmes d'information.
4. Sur le plan pratique, nous avons approfondi nos connaissances sur le standard XML et nous avons maitrisé l'exploration de ce standard lié au service Web dans l'environnement java.

Nous espérons que ce travail important sera exploité par la communauté activant dans le domaine des services Web. Nous souhaitons, aussi qu'il sera pris en charge par les promotions futures pour l'utiliser et le raffiner.

Bibliographie

BIBLIOGRAPHIE

[1] Les apports de la methode MDM dans la performance du SI des entreprises ; *par* Axel KAMALAK ; EPSI - Ingénieur 2008.

[2] Introduction aux Systèmes d'Information Répartis, Lê Hồng Phương.

[3] Méthode de Conception des Systèmes d'Information, NGANG BILOUNGA Jean Jacques.

[4] Systèmes distribués, Tarak Chaari, l'institut supérieur d'électronique et de communication.

[5] Architectures de Systèmes d'Information, Laurent Avignon, Gilles Laborderie, Rémy Mathieu- Daudé, et Pierre Pezziardi. ,Novembre 2002 , Livre Blanc.

[6] Les services Web ; Jeremy Fierstone ; SAR5 – Novembre 2002.

[7] LES *WEB SERVICES* ET LEUR IMPACT SUR LE COMMERCE B2B ;Gilbert Babin Fellow, CIRANO Michel Leblanc; (Août 2003).

[8] Web Services ; Mohamed Kawtharany

[9] Découverte et Sélection de Services Web pour une application Mélusine ; Ricardo DE LA ROSA-ROSERO ; septembre 2004 ; Genève (Suisse).

[10] Introduction aux Web Services ; Rahee Ghurbhurn.

[11] Services Web – publication et découverte ; Fabrice Rossi ; Université Paris-IX Dauphine

[12] CONTRIBUTION À L'INTEGRATION DES PROCESSUS MÉTIER : APPLICATION À LA MISE EN PLACE D'UN RÉFÉRENTIEL QUALITÉ MULTI-VUES ; Anis FERCHICHI ; Juillet 2008 ; Ecole Centrale de Lille et Recherche Opérationnelle Innovation – Sylis.

[13] Business Process Management ;De la modélisation à l'exécution ; Positionnement par rapport aux Architectures Orientées Services ,Tanguy Crusson.

[14] Gestion des processus métier et travail collaboratif ; Eddie Soulier1, Myriam Lewkowicz1, Nicolas Corouge2 ; Université de Technologie de Troyes.

[15] Une approche pour l'amélioration de la flexibilité des processus métier basée sur les techniques du process mining ; Mounira ZERARI ; Université Mentouri – Constantine.

[16] Business Process Analytics ; Michael zur Muehlen.

[17] Design Patterns du Gang of Four appliqués à Java.

[18] Erich Gamma et al., "Design Patterns – Elements of Reusable;Object-Oriented Software", Addison-Wesley, 1995.

[19] Design PatternsElements of Reusable Object-Oriented Software; Produced by KevinZhang.

[20] Design patterns ; Lionel Seinturier ;Université des Sciences et Technologies de Lille.

[21] Representing, analysing and managing Web service protocols; Boualem Benatallah , Fabio Casati , Farouk Toumani .

[22] Business-to-business interactions: issues and enabling technologies; Brahim Medjahed, Boualem Benatallah, Athman Bouguettaya, Anne H. H. Ngu, Ahmed K. Elmagarmid; April 3, 2003

[23] A Computer Scientist's Introductory Guide to Business Process Management (BPM) ;by Ryan K. L. Ko.

[24] Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems; Barbara Weber a, Manfred Reichert, Stefanie Rinderle-Ma; 9 May 2008.

[25] Visual Timed Event Scenarios;A. Alfonso, V. Braberman, N. Kicillof; Departamento de Computaci´on – FCEN;Universidad de Buenos Aires, Argentina.

[26] Web Service Conversation Modeling ; Boualem Benatallah; Fabio Casati; Farouk Toumani.

[27] Framework for Managing the Evolution of Business Protocols in Web Services ;Seung Hwan Ryu, R´egis Saint-Paul, Boualem Benatallah, Fabio Casati; School of Computer Science & Engineering 2 Intelligent Enterprise Technology Lab ;University of New South Wales HP Laboratories Sydney, Australia

[28] Réseaux de Petri , catheryne Trembely

[29] Learning Information Extraction Patterns from Examples ; Scott B. Huffman ; Price Waterhouse Technology Centre, 68 Willow Road, Menlo Park CA 94025, USA.

[30] Limits of Formalism in business process modelling; An Essay

[31] Les Modèles de traitements (MCT et MCTA) ; S. Laporte.

[32]Thèse Pour l'obtention du Doctorat en Sciences Spécialité : Informatique Par ADLA BENTELLIS Epouse FEDJKHI

[33]Un modèle de processus métier pour les nouvelles formes d'organisation des activités Improved modelling for flexible and collaborative business Process

[34] Thèse préparée dans le Laboratoire de Génie Industriel de Lille – Ecole Centrale de Lilleet Recherche Opérationnelle Innovation - Sylis
Ecole Doctorale SPI 287 (EC Paris, EC Lille)
tel-00295306

[35]Le Guide Pratique des Processus Métiers Copyright Softeam
Copyright Softeam 2008 21 avenue Victor Hugo, 75016 PariS

[36]Business Process Management - Loulem

[37]Langage de modélisation objet unifié - TsTRI.Com . 100% Réseaux .

[38]Bringing Semantics to Web Services:The OWL-S Approach ; Department of Computer Science, University of Toronto,Toronto, Ontario, Canada;

[38] http://fr.wikipedia.org/wiki/Unified_Modeling_Language