République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 08 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 : Informatique Académique

Thème :

Analyse d'impact de l'évolution des protocoles de services web

Encadré par :

Mr. KHEBIZI Ali

Présenté par :

DIARRA Djibril

MAHAMAT NOUR Adam Moussa

Juin 2013

Dédicace

« Ce que je sais c'est à mon ignorance que je le dois » - Sacha Guitry

« Connaître son ignorance est la meilleur part de la connaissance » - Lao Tseu

« La réflexion est le travail le plus dur qu'il soit, ce qui explique sans doute pourquoi si peu de gens s'y adonnent » - Henri Ford

« On ne peut rien enseigner à autrui. On ne peut que l'aider à le découvrir lui-même » - Galilée

« Devant une situation complexe, il y a toujours une réponse simple : elle est mauvaise » - Umberto Eco

« L'expérience ce n'est pas ce arrive aux hommes, c'est ce que les hommes font de ce qui leur arrive » - Aldous Huxley

« Il s'agit plus de penser et d'agir autrement que davantage » – Patrick ZIMBARDO

A nos parents...

Remerciement

Nous rendons grâce au bon DIEU de nous avoir donné une santé de fer et beaucoup de courage. Nous remercions nos parents pour leur soutien incessant sans condition, nos amis pour leurs encouragements et tous les enseignants qui ont contribué à notre formation depuis l'école primaire. Mention spéciale à notre encadreur **Mr. KHEBIZI Ali,** pour ses conseils fructueux et sa disponibilité illimitée durant tout notre travail.

Nos remerciements les plus vifs s'adressent à monsieur le président ainsi qu' aux membres de jury d'avoir accepté d'examiner et d'évaluer ce modeste travail.

Et nous ne saurons terminer sans lancer un grand merci à tous nos camarades étudiants de la promotion du département de informatique. En particulier ceux de notre spécialité INFORMATIQUE ACADEMIQUE promotion 2013. En bref, nous remercions tous ceux qui nous ont aidé de près ou de loin que ce soit par leur soutien moral ou physique pour la réalisation de ce projet.

Résumé

Dans l'environnement versatile qu'est le Web, les différents acteurs d'un système d'information peuvent interagir d'une manière dynamique, abstraction faite sur leur localisation géographique, leur plate-forme. Différentes entreprises interagissent avec des clients de plus en plus nombreux ou inter opèrent avec d'autres entreprises dans un cadre de partenariat ou d'intégration de leurs systèmes d'informations. Ces interactions s'effectuent à travers les processus métiers qui expriment les règles de gestion de l'entreprise et qui doivent être modélisés et implémentés avant d'être mis en service. Ces dernières années, beaucoup de techniques d'intégration ont vu le jour. Parmi ces techniques figure l'architecture orientée service qui semble la plus adéquate actuellement. Cette technique est basée sur les services Web et un service Web est toujours associé à des processus métier qui constituent son protocole métier.

A tout moment, un service publié est exécutée par un nombre important de clients. Dans son environnement d'évolution toute entreprise est soumise à des changements qui sont reflétés par le changement de ses processus métiers. Or, tout changement de protocole d'un service influe sur les clients qui exécutent ce service. **Le travail démandé dans ce mémoire consiste à analyser l'impact du changement des protocoles de service Web sur ses clients en cours d'exécution qu'on appelle instances actives. Cette analyse consiste à déterminer si une instance active peut continuer son exécution sans erreur après l'évolution du protocole du service qu'il exécutait.** Pour cela :

Nous modéliserons les protocoles métiers par des automates d'état finis déterministes, ceci facilite leur gestion (création, modification, évolution, ...).

Nous effectuerons une simulation de l'exécution des services par des clients à travers la génération des traces sur les protocoles définis, ces traces seront stockées dans une base de données relationnelle.

A travers cette base de données et l'approche qu'on proposera, nous déciderons de la migration des traces générées sur un protocole après une évolution de ce dernier.

Mots-clés : Système d'information, intégration, architecture orientée service, service Web, processus métier, trace d'exécution, automates d'états finis deterministe, migration.

Table des matières

Table des figures

Liste des tableaux

Abréviations et Acronymes

API : Application Programming Interface.

BP : Business Process.

BPM : Business Process Management.

BPMN : Business Process Management Notation.

BPEL : Business Process Execution Language.

BPEL4WS : Business Process Execution Language for Web Services.

B2B : Business To Business.

CORBA : Common Object Request Broker Architecture.

CICS :Customer Information Control System.

DB2 :Base de données relationnelle.

DOM :Document Object Model.

DTD :Document Type Definition.

EAI : Enterprise Application Intergration.

Ebxml : Electronie bisnes XML.

EDI : Electronice Data Interchange.

ERP : Enterprise Ressources Planning

FTP :File Transfer Protocol.

HTML : Hypertext Markup Language.

HTTP : HyperText Transfer Protocol.

HP :Hewlett-Packard.

IDL : Interface Definition Language.

IBM :International Business Machines.

IP : Internet Protocol.

IMS :Information Management System.

IS : Information Systemes.

JAXP :Java API for Processing.

JDBC : Java Data Base Connectivity

JDOM :Java Document Object Model.

LDD :Langage de Définition des Données.

LCD :Langage de Contrôle de l'accès aux Données.

LMD :Langage de Manipulation des Données.

MCD :Modèle Conceptuel de Données.

MVS :Multiple Virtual Storage.

MIME :Multipurpose Internet Mail Extensions.

NAA :Not An Abreviation.

OASIS : Organization for the Advancement of Structured Information Standards.

OMG : Object Management Group.
OMT : Object Modelling Technique.
OOSE : Object Oriented Software Engineering.
OWL-S : Ontology Web Language for Services.
PAIS : Process Aware Information Systemes.
PHP :Personal Home Page.
RDB :Relational DataBase.
RPC : Remote Procedure Call.
RdP : Réseau de Pétri.
SAX :Simple API for XML.
SGBDR :Systéme de Gestion de Base de Bonnées Relationnel.
SOA : Service Oriented Architecture.
SMTP : Simple Mail Transfer Protocol.
SOAP : Simple Object Acces Protocol.
SQL :Structured Query Language
SVG :Scalable Vector Graphics.
WS : Web Services.
WSDL : Web Service Description/Definition Language.
XML : eXtensible Markup Language.
URL : Unified Ressources Locator.
UDDI : Universal Description, Discovery and Integration.
UML : Unified Modeling Language.
W3C : World Wid Web Consortium.
WS-I : Web Services Interoperability Organization.
WSFL : Web Service Flow Language.
WSCI : Web Service Choreography Interface.
WS-CDL :Web Service Choreography Language.
WFM :WorkFlow Management.
XSD : XML Schema Definition.

Introduction générale

L'avènement de l'Internet a révolutionné les modes de communications entres les individus et les organisations. Les systèmes d'informations des entreprises doivent, par conséquent, s'adapter aux nouveaux besoins d'ouverture sur le monde extérieur, d'interopérabilité avec d'autres systèmes et aux besoins d'évolution. La technologie des services Web répond dans une large mesure à ces nouvelles exigences des systèmes d'informations de l'entreprise.

Un service Web est décrit par son interface de service et par son protocole métier (Business Protocole). Leur environnement d'évolution est dynamique et caractérisé par un nombre important d'instances qui peuvent invoquer un service publié. En même temps, le processus métier associé au service est impacté par les changements de l'environnement (modification de lois, nouveaux besoins, raisons de maintenance, raisons de fusion des activités, ...). Une conséquence immédiate de ce changement est la modification du protocole du service qui est déjà publié. Dans ce cadre, les instances actives ne peuvent pas continuer leur exécution conformément à l'ancienne version du protocole et doivent impérativement s'adapter aux nouvelles spécifications.

Donc la question de la continuité des exécutions en cours s'impose et une analyse des impacts de l'évolution est inévitable.

Notre travail porte sur une analyse formelle de ce problème. Une approche sera proposée afin de filtrer les instances d'exécutions continuables et celles qui ne le sont pas. Ceci se fait sur la base de l'analyse du schéma des protocoles, modélisés avec des Automates d'Etats finis Déterministes (AEFD) et sur la base des traces d'exécutions enregistrées dans les journaux Logs, en occurrence une base de données des traces. Dans ce travail on aborde la problématique de l'analyse d'impact de l'evolution des protocoles de service Web sur ses traces en cours.

Ce mémoire est constitué de deux parties essentielles, composée chacune de trois chapitres.

La première partie présente le contexte général de notre travail à travers ses trois chapitres. Le premier chapitre intitulé :Systèmes d'informations distribués relate une description brève des Systèmes d'informations classiques ainsi que leurs insuffisances face aux nouvelles besoins de l'entreprise, présente succinctement les systèmes d'informations distribués à travers quelques uns de ses concepts. Le problème d'intégration des différents composants logiciels d'un système d'information distribué nous conduit rapidement à l'architecture orienté service dont l'élément de base est le service Web. Ce dernier fait partie des essences de notre sujet avec les processus métier. Par conséquent la technologie des services Web y sera minutieusement détaillée. Le second chapitre portera sur la nouvelle tendance des systèmes d'information : les PAIS, sur les processus métiers qui sont des composants inhérents aux services Web ainsi que sur les traces qui désignent l'exécution d'une activité par un client dans le système. Le chapitre trois sera consacré à l'évolution des processus métiers : les raisons des évolutions,

les types d'évolutions ainsi que les conséquences induites par ces évolutions.

Les trois chapitres de la seconde partie concernent la conception et l'implémentation de notre approche. Dans le premier, nous évoquerons les modèles (patterns) d'évolutions et la problématique de notre sujet. Le second portera sur l'approche proposée pour la résolution de la problématique posée dans le précédent ainsi que les éléments nécessaires à sa conception. Le troisième sera consacré à la réalisation concrète de cette approche en termes de programmation. Nous présenterons les différents outils utilisés dans l'implémentation de notre application ainsi que ses interfaces principales avec leurs grandes fonctionnalités. Et enfin nous terminerons par une conclusion générale.

Première partie

Etat de l'art

Chapitre 1

Systèmes d'Information Distribués

Introduction

Les problématiques de communication et d'intégration entre les applications hétérogènes au sein d'une entreprise ou inter-entreprise sont un vrai défi de l'informatique contemporaine. En effet, chaque entreprise dispose de son Système d'Information (S.I) mais, en même temps, a besoin d'échanger des informations avec d'autres S.I dans une perspective d'interopérabilité.

Dans ce chapitre, nous aborderons les systèmes d'informations tout en montrant comment ils sont souvent complexes et peu cohérents, pour bien illustrer le besoin de la nouvelle technologie des services Web.

1.1 Systèmes d'Informations

A l'ère de l'information et des technologies de communication, consciemment ou inconsciemment, chacun de nous, est en contact quasi-permanent avec un ou plusieurs S.I. Les appréciations et les points de vue peuvent varier, mais l'impact des S.I sur la société, l'économie et la vie quotidienne de chacun de nous est incontestablement perceptible. Historiquement, les SI ont débuté avec les outils de gestion. Il était alors question de "robotiser", à l'aide de l'informatique, des tâches difficiles et répétitives liées au traitement des données, afin de gagner en rapidité et fiabilité.

1.1.1 Définitions

En informatique et en télécommunications, et plus généralement dans le monde de l'entreprise, le terme système d'information (SI) possède les significations suivantes :

a- Reix, 1995 :

Ensemble organisé de ressources : matériel, logiciel, personnel, données, procédures ; permettant d'acquérir, traiter, stocker, communiquer des informations (données, textes, images,

sons...) dans les organisations.[1]

b- Mason et Mittrof, 1973 :

Tout système d'information concerne un individu, pourvu d'un profil psychologique donné, confronté à un problème précis, dans un contexte organisationnel déterminé.[1]

c- Alter, 1991 :

Une combinaison de pratiques de travail, d'informations, de personnes et de technologies de l'information organisée pour atteindre des objectifs dans une organisation.[1]

d- Un système ou sous-système d'équipements d'informatique ou/et de télécommunication, interconnectés dans le but de l'acquisition, du stockage, de la structuration, de la gestion,du déplacement, du contrôle, de l'affichage, de l'échange (transmission ou réception) de données sous forme de textes, d'images, de sons.[2]

A notre avis, la définition (a) s'avère plus concise et pertinent dans notre contexte.

1.1.2 Structure d'un système d'information

Plusieurs méthodologies de conception des S.I existent et sont plus au moins utilisées en fonction des domaines d'application.

En dépit de la complexité et de la variété des S.I, nous pouvons résumer les caractéristiques liées à l'aspect de leur conception dans la fig 1.1. La représentation en couche permet de faciliter la conception et le déploiement du S.I.

- Couche présentation :

C'est un utilisateur ou un programme qui veut effectuer une opération sur le système. Les clients interagissent avec le système à travers cette couche de présentation.[12]

- Couche logique d'application :

Détermine ce que le système fait en réalité. Elle prend soin de faire respecter les règles métier et établir les processus métier.

La logique de l'application peut prendre plusieurs formes : les programmes, les contraintes, les Workflows, ...etc. [12]

Dans ce mémoire, la problématique abordée se positionne au niveau de cette couche logique.

- Couche gestionnaire de ressources (Resource Management Layer) :

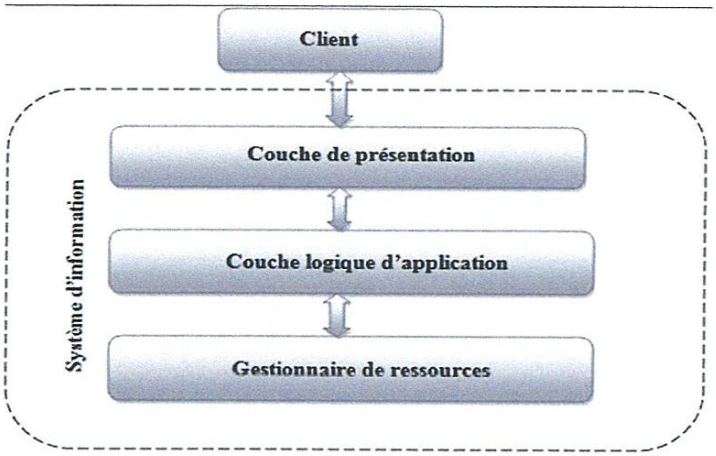

FIG. 1.1 – Les différentes couches d'un SI[12].

S'occupe de l'organisation (stockage, indexation, la recherche) des données nécessaires au soutien de la logique applicative. Il s'agit généralement d'une base de données, mais il peut aussi être un système de récupération de texte ou tout autre système de gestion de données fournissant des fonctionnalités de requêtes et de persistance. [12]

1.1.3 Insuffisances des systèmes d'informations centralisés

L'expansion et la généralisation du Web fait que les organisations et entreprises visent à atteindre plus de profit, mettre leurs produits à la disposition du maximum de clients et nouer un cadre de partenariat économique assez étendu. Pour repondre à ces exigences, ils doivent élargir leur champ de circulation d'information, et quoi de plus approprié que le Web pour ça ? Les applications sur un seul site central (les SI classiques) souffrent d'une interface utilisateur en mode caractères et la cohabitation d'applications exploitant des données communes n'est pas évident. Ainsi les SI classiques sont fragmentés en sous systèmes stables pour plus de disponibilités, de fiabilté, de sécurité et donc plus de gains.

Les SI sont alors condamnés à ouvrir leur environnement donnant ainsi naissance au SI distribues.

1.2 Systèmes d'informations distribués

1.2.1 Définition

Un système d'information distribuée est un système d'information basé sur une structure de système distribué. Les différentes couches (composants, sous systèmes) d'un tel système sont

dispersées sur de sites différents et sa réalisation nécessite une synergie de communications de données et des ressources informatiques entre ces composants.

Dans la suite nous présenterons les caractéristiques de cette synergie de communications.[3]

1.2.2 Problématique de l'intégration des S.I Distribués

L'intégration, un terme actuellement très en vogue, est devenue pour beaucoup d'entreprises une nécessité incontournable pour faire face aux exigences sans cesse évolutives du marché. Les contraintes qui pèsent de nos jours sur les entreprises sont souvent directement répercutées sur leurs systèmes d'information, à qui on demande d'être sans cesse plus agiles afin de pouvoir mieux soutenir la stratégie de l'entreprise. Pour faire face à ces nouvelles exigences, les entreprises ont très souvent recours au concept d'intégration, et parfois de façon plus spécifique au concept d'interopérabilité afin d'interconnecter leurs applications informatiques.[4]

1.2.3 Problématique de l'interopérabilité

Les partenaires ont besoin d'échanger des données syntaxiquement et sémantiquement interopérables. Il faut donc disposer de formats et d'une ontologie en commun pour pouvoir appréhender ces informations sur le plan syntaxique et sémantique. Selon le type de processus (conception, production, gestion globale...), et l'acteur en charge de la décision, différents types d'informations doivent être partagées. Pour cela, un format d'échange général et une politique d 'intégration des contraintes de sécurité doivent être définis. Une technique simple est réalisée par l'interconnexion de tous les éléments d'information à partager dans une structure faiblement couplée en se basant sur des objets métier communs. Ce faible couplage (interconnexion via le partage d'information) donne une grande souplesse du système d'information global.[4]

1.2.4 Intégration et échange : techniques existantes

Dans cette section, nous traitons les notions liées à l'intégration et l'échange de données entre les systèmes d'information distribués au sein de l'entreprise. L'objectif est de montrer la possibilité de se servir des techniques existantes pour la mise en oeuvre du système d'information inter-entreprises visé. Nous introduirons l'EDI (Electronic Data Interchange) ou en français, Echange de Données Informatisées. L'EDI offre un moyen pour échanger les informations entre des entités de business séparées géographiquement. Ensuite nous présenterons les technologies d'EAI (Enterprise Application Integration) et CORBA (Common Object Request Broker Architecture). En effet, CORBA peut servir comme un middleware dont l'objectif est de regrouper des applications réparties par l'intermédiaire d'un bus général de communication. Les détails de ces techniques sont traités dans les paragraphes suivants.[6]

Premiers pas : l'EDI

L'EDI peut être défini comme l'échange d'ordinateur à ordinateur de données concernant des transactions en utilisant des réseaux et des formats normalisés. Les informations issues du système informatique de l'émetteur transitent par l'intermédiaire de réseaux vers le système informatique du partenaire pour y être intégrées automatiquement. Ceci induit le traitement des points suivants :[6]

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

Un des défauts de l'EDI est son coût assez élevé, lié au besoin d'une infrastructure de réseau adaptée et à l'organisation de processus contraignant avec des interfaces « lourdes ». Ce défaut peut être évité avec la technologie du Web EDI que nous présentons dans le paragraphe suivant.[6]

Web EDI

Dans sa forme la plus simple, le Web EDI permet aux petites et moyennes entreprises de créer, gérer, recevoir et envoyer des documents électroniques en utilisant des technologies et outils du Web. Pour cela, un service dédié doit transformer de façon transparente les données de l'entreprise en format échangeable (respectant un standard EDI) et les transmettre aux partenaires commerciaux. Des formats simples sont utilisés pour permettre aux entreprises d'échanger avec leurs partenaires commerciaux. En utilisant une interface Web convenable, les transactions de type EDI peuvent être effectuées aussi facilement par le biais du courrier électronique. Le Web EDI permet également de recevoir et d'envoyer des documents métier sans recourir à un logiciel particulier. Son avantage est le fait qu'il est accessible partout dans le monde sans avoir besoin d'installer des logiciels métiers ou de mettre en place une infrastructure de communication particulière. Il offre un moyen efficace pour échanger les données entre partenaires. En revanche, comme l'EDI classique, il n'a pas trouvé la reconnaissance méritée dans le domaine du business inter-organisationnel. Cela dû au fait qu'il est dédié à l'échange de données et non aux règles de décision, règles dont on a besoin pour gérer la chaîne de partenaires.[6]

Communication entre les briques logicielles : EAI et CORBA

Comme nous l'avons déjà mentionné, les systèmes d'information d'entreprise sont composés de plusieurs "briques applicatives ". Souvent développées indépendamment et recourant à

des technologies parfois incompatibles, ces outils ne sont que peu interopérables. Dans une logique d'intégration (ou tout au moins de couplage entre ces briques), il convient d'ajouter des systèmes intermédiaires qui facilitent la démarche d'interfaçage, voire d'intégration. C'est le rôle des outils d'EAI (Enterprise Application Integration). Outre la gestion des formats d'échange, l'EAI doit aussi prendre en compte le flux des événements entre les applications. Dans ce contexte, un système de gestion de Workflow devient un élément important de l'EAI, car il est capable de coordonner et de contrôler le flux des évènements entre les applications. Ceci conduit donc à définir un type particulier d'EAI : l'EAI basé sur le Workflow. Les systèmes de ce type peuvent être classés en deux catégories [6] :

EAI statique :

L'exécution du processus de Workflow suit un chemin précis et prédéfini. Il n'est pas possible d'ajouter un processus ou de modifier la configuration de la définition du processus en cours d'exécution.[6]

EAI dynamique :

La liaison entre les services des applications se fait en cours d'exécution selon des règles spéciales. La configuration des interfaces des services peut alors être changée et de nouveaux systèmes peuvent être ajoutés.[6]

L'EAI permet de résoudre le problème d'interopérabilité entre les briques applicatives constituant les applications propres à l'entreprise. Elle permet de spécifier les informations à échanger mais avec une localisation statique de ces informations. En revanche, et en se basant sur les facilités de CORBA, un middleware générique peut être mise en place permettant d'accéder aux informations quelle que soit leur localisation. CORBA (Common Object Request Broker Architecture) représente une solution d'intégration pour des applications développées indépendamment [6]. Il s'agit d'un middleware orienté objet proposé par l'Object Management Group (OMG, *un consortium international créé en 1989 et a pour objectif de faire émerger des standards pour l'intégration d'applications distribuées hétérogènes à partir des technologies orientées objet*). Les concepts clés mis en avant sont la réutilisabilité, l'interopérabilité et la portabilité de composants logiciels. CORBA est un bus d'objets répartis qui offre un support d'exécution masquant les couches techniques d'un système réparti (système d'exploitation, processeur et réseau). Elle prend en charge les communications entre les composants logiciels formant les applications réparties hétérogènes (voir fig 1.2). Le bus CORBA propose un modèle orienté objet client/serveur pour la coopération entre les applications réparties. Chaque application peut exporter certaines de ses fonctionnalités (services) sous la forme d'objets CORBA : c'est la composante d'abstraction de ce modèle.

Les interactions entre les applications sont réalisées par l'invocation à distance de méthodes associées aux objets. C'est la partie coopération. La notion client/serveur intervient

uniquement lors de l'utilisation d'un objet : l'application implantant l'objet est le serveur, l'application utilisant l'objet est le client. Selon les traitements effectués, on notera qu'une application peut tout à fait être à la fois cliente et serveur. Dans la fig 1.2, on identifie les éléments suivants : [6]

- L'application cliente est un programme qui invoque les méthodes et des objets à travers le bus CORBA.
- La référence d'objet est une structure désignant l'objet CORBA et contenant l'information nécessaire pour le localiser sur le bus.
- L'interface de l'objet est le type abstrait de l'objet CORBA définissant ses opérations et attributs. Cette interface est définie par l'intermédiaire du langage OMG-IDL
- La requête est le mécanisme d'invocation d'une opération ou d'un accès à un attribut de l'objet.
- Le bus CORBA achemine les requêtes de l'application cliente vers l'objet en masquant tous les problèmes d'hétérogénéité (langages,systèmes d'exploitation, matériels, réseaux).
- L'objet CORBA est le composant logiciel cible. C'est une entité virtuelle gérée par le bus CORBA.
- L'activation est le processus d'association d'un objet implémentant un traitement à un objet CORBA.
- L'implantation de l'objet est l'entité codant l'objet CORBA à un instant donné et gérant un état de l'objet temporaire.
- Le code d'implantation regroupe les traitements associés à l'implantation des opérations de l'objet CORBA.
- L'application serveur est la structure d'accueil des objets d'implantation et des exécutions des opérations.

Le standard CORBA offre une solution ouverte et évolutive avec une architecture modulaire garantissant l'interopérabilité entre des composants hétérogènes, tout en gardant un choix libre pour les technologies d'implantation. Cette norme offre la possibilité d'encapsuler l'existant, ainsi on peut réutiliser totalement des applications complètes en les encapsulant dans des objets CORBA, objets utilisables pour bâtir de nouvelles applications.

Par contre, du fait de sa complexité et de son coût de mise en oeuvre (ad hoc pour chaque entreprise) assez élevé, CORBA n'a pas été adopté par les géants de logiciels comme Microsoft et IBM. Il en résulte un manque d'outils de support pour cette technologie.

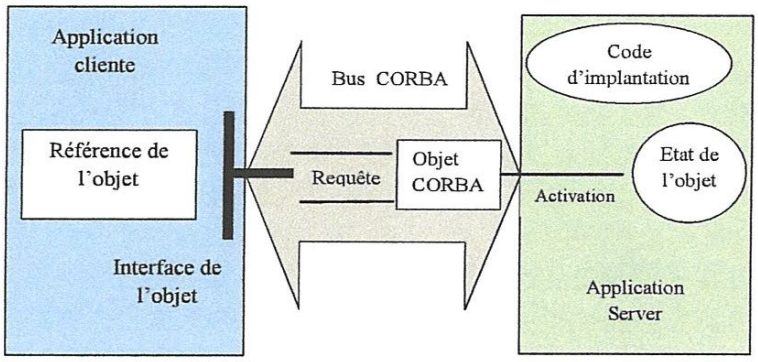

FIG. 1.2 – Notion client/serveur avec le bus de CORBA.[4]

Les ERP : Enterprise Ressources Planning

Définition : Le principe fondateur d'un PGI/ERP est de construire des applications informatiques (gestion des commandes, des stocks, de la paie, de la comptabilité, etc.) de manière :

- modulaire et intégrée au niveau des traitements offerts (les différents modules qui le composent sont indépendants mais parfaitement compatibles entre eux) ;
- rigoureuse et cohérente au niveau des données gérées (partage d'une base de données unique et commune).[16]

Cela comble une lacune importante :

- Dans la situation préexistante aux PGI/ERP, des applications sur mesure d'origine diverse, co-existent tant bien que mal, ne partagent pas ou peu leurs données et ne sont pas forcément toujours prévues pour travailler simplement et correctement ensemble. Les équipes informatiques ont alors fort à faire pour construire et mettre en place des interfaces "ad hoc" dont le fonctionnement pratique peut réserver des surprises.[16]
- Les PGI/ERP sont des garantis modulaires par leur concepteur. Par exemple : le module achat est garanti compatible avec le module stock, qui est garanti compatible avec le module gestion de commandes. Les données sont désormais supposées standardisées et partagées, ce qui élimine les saisies multiples et évite (en théorie) l'ambiguïté des données multiples de même nature (exemple : société TRUC, TRUC SA et Sté TRUC...).[16]

L'autre principe qui caractérise un PGI/ERP est l'usage systématique de ce qu'on appelle un moteur de workflow (qui n'est pas toujours visible de l'utilisateur), et qui permet,

lorsqu'une donnée est entrée dans le système d'information, de la propager et d'offrir des vues logiques pertinentes dans tous les modules du système qui en ont besoin, selon une programmation prédéfinie.

Ainsi, on peut parler de PGI/ERP lorsqu'on est en présence d'un système d'information composé de plusieurs applications partageant une seule et même base de données, par le biais d'un système automatisé prédéfini éventuellement paramétrable (un moteur de workflow).[16]

Les avantages

- optimisation des processus de gestion (flux économiques et financiers) ;
- cohérence et homogénéité des informations (un seul fichier articles, un seul fichier clients, etc.) ;
- intégrité et unicité du Système d'information ;
- partage du même système d'information facilitant la communication interne et externe ;
- minimisation des coûts : pas d'interface entre les modules, synchronisation des traitements, maintenance corrective simplifiée car assurée directement par l'éditeur et non plus par le service informatique de l'entreprise (celui-ci garde néanmoins sous sa responsabilité la maintenance évolutive : amélioration des fonctionnalités, évolution des règles de gestion, etc.) ;
- globalisation de la formation (même logique, même ergonomie) ;
- diminution du nombre de salariés ayant pour mission principale la saisie comptable (aide-comptable) ;
- maîtrise des coûts et des délais de mise en œuvre et de déploiement ;

Ce dernier point est essentiel et la mise en œuvre d'un ERP/PGI dans une entreprise est fréquemment associée à une révision en profondeur de l'organisation des tâches et à une optimisation et standardisation des processus, en s'appuyant sur le « cadre normatif » de l'ERP/PGI.[16]

Les inconvénients

- mise en œuvre pouvant être complexe si le périmètre est mal déterminé ou trop mouvant ou le projet mal piloté ;
- coût élevé de 300 000 € minimum pour un progiciel fiable et de qualité, mais pouvant rapidement monter beaucoup plus haut, en fonction de l'industrie et de la complexité

du projet. L'option fonctionnellement riche des solutions de logiciels libres si elle réduit les coûts de licence, ne supprime pas les coûts d'accompagnement et de formation ;

- périmètre fonctionnel souvent plus large que les besoins de l'organisation ou de l'entreprise (le progiciel est parfois sous-utilisé) ;
- périmètre fonctionnel pouvant ne pas couvrir l'ensemble des besoins ;
- lourdeur et rigidité de mise en œuvre ;
- difficultés d'appropriation par le personnel de l'entreprise ;
- nécessité d'une bonne connaissance des processus de l'entreprise (par exemple, une petite commande et une grosse commande nécessitent deux processus différents : il est important de savoir pourquoi, de savoir décrire les différences entre ces deux processus de façon à bien les paramétrer et à adapter le fonctionnement standard du PGI/ERP aux besoins de l'entreprise) ;
- nécessité parfois d'adapter certains processus de l'organisation ou de l'entreprise au progiciel ;
- nécessité d'une maintenance continue ;
- captivité vis-à-vis de l'éditeur : le choix d'une solution est souvent structurant pour l'entreprise et un changement de PGI peut être extrêmement lourd à gérer.[16]

Evolution : Les défauts des techniques précédentes les laissent indésirées et l'intégration étant une nécessité en vue aux nouvelles exigences des entreprises, il faut une technique convenable pour tous, qui remédie aux défauts des précédentes. Cette technique est l'utilsation de SOA (Service Oriented Architecture, *en français Architecture Orientée Service*). Nous détaillerons dans les sections suivantes comment cette approche peut apporter l'interopérabilité entre les systèmes d'informations des partenaires au sein de la chaîne d'interaction et d'échanges.

1.3 Architecture orientée service

Les architectures orientées services sont présentées actuellement comme la solution miracle à tous les problèmes d'intégration du SI.

C'est une architecure logiciel dont l'objectif est de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées services fournies par des composants et de décrire finement le schéma d'interaction entre ces services. Ces services sont indépendants de tout

langage de programmation et peuvent être hébergés sur différentes plates-formes avec divers modèles de sécurité.

1.3.1 Pourquoi l'architecture orientée service ?

La SOA offre une capacité de réaction flexible durant le cycle de vie de l'organisation associée à la chaîne logistique : tous les processus accessibles par les clients et les partenaires sont intégrés dans un système collaboratif basé sur les règles métier de la chaîne logistique. Ceci permet aux entreprises membres de développer leur stratégie marketing en temps réel en se basant sur les communications et les transactions réalisées en partenariat avec d'autres entreprises. Cette fonction est assurée au niveau du socle de l'architecture, en synthétisant et en analysant les flux de données, pour créer de nouvelles pratiques métier.

En plus, la SOA permet d'utiliser des services génériques. En se basant sur ces services, les entreprises peuvent s'adapter en temps réel afin de répondre aux besoins du marché. L'utilisation de standards ouverts permet une adoption rapide et moins coûteuse des meilleurs pratiques métiers au sein de l'organisation et de son écosystème.

1.3.2 Définitions

Il existe différentes définitions de l'SOA selon différents acteurs de l'entreprise :

- Dirigeants et analystes métiers : Un ensemble de services qu'une entreprise souhaite exposer à leurs clients et partenaires, ou d'autres parties de la même entreprise.
- Architectes : Un style architectural qui, basé sur un fournisseur, un demandeur et une description de service, supporte les propriétés de modularité, encapsulation, découplage, réutilisation et compossibilité.
- Développeurs : Un modèle de programmation avec ses standards, paradigmes, outils et technologies associées.
- Intégrateurs : Un intergiciel (middleware) offrant des fonctionnalités en termes d'assemblage, d'orchestration, de surveillance et de gestion des services.

Pour nous, c'est une architecture logicielle s'appuyant sur un ensemble de services simples. L'idée sous-jacente est de cesser de construire la vie de l'entreprise autour d'une seule grande application centrale et faire en sorte de construire une architecture logicielle globale décomposée en services correspondant aux processus métiers de l'entreprise ».

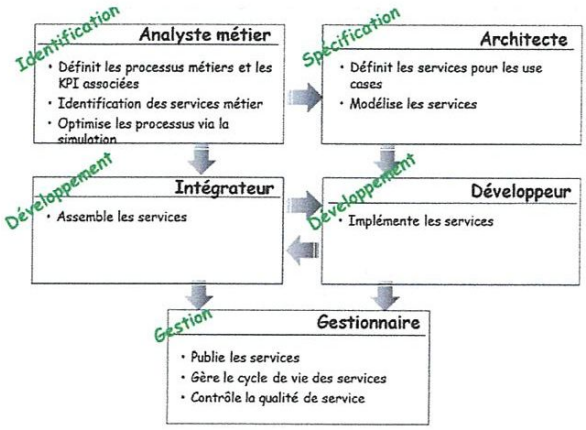

FIG. 1.3 – Cycle de vie d'un service

1.3.3 La base de l'SOA : le service

Définition et cycle de vie

Le service est le composant clef de la SOA. Il consiste en une fonction ou fonctionnalité bien définie. C'est aussi un composant autonome qui ne dépend d'aucun contexte ou service externe. Il est divisé en opérations qui constituent autant d'actions spécifiques que le service peut réaliser. En d'autre terme, c'est une fonction encapsulée dans un composant que l'on peut interroger à l'aide d'une requête composée d'un ou plusieurs paramètres et fournissant une ou plusieurs réponses. Idéalement chaque service doit être indépendant des autres afin de garantir sa réutilisabilité et son interopérabilité.

Le cycle de vie d'un service est constitué de quatre (4) grandes phases à savoir : L'identification, la spécification, le développement et la gestion.voir fig 1.3

1.3.4 Technologie d'application des SOA

Parmi les technologies utilisant les SOA, les services Web répresentent une implémentation concrète de ce paradigme et d'ailleurs c'est la technologie la plus utilisée. Nous la decortiquons dans la section suivante

1.3.5 Quelques atouts des SOA

Les architectures orientées services offrent une flexibilité et une meilleure résilience, notamment en termes de disponibilité, au système d'information. Ces architectures apportent trois avantages majeurs :

- Réactivité : la création des processus métiers interentreprises obtenu par la composition

de services distribués permet d'accélérer la mise en oeuvre d'une solution pour répondre à un nouveau besoin métier engendré par l'évolution des conditions de marché.

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

Cet aspect entre en ligne direct avec notre problématique mais la gestion des traces actives n'est pas prise en charge par la SOA.

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

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

Dans la prochaine section, nous nous intéresserons aux services Web, leurs standards et leurs fonctionnements.

1.4 Les Services Web

Le paradigme de services Web (*en Anglais, Web Service* **WS**) a émergé comme un mécanisme puissant pour intégrer les systèmes d'information distribués. Combinant les meilleurs aspects du développement dans le domaine de construction des systèmes d'information à base de composants avec les facilités offertes par le Web, les services Web sont utilisés pour implémenter le concept de SOA. Un service en général (et plus particulièrement un service web) représente une fonctionnalité qui peut être facilement réutilisée sans avoir à connaître ni les détails ni la façon dont le service est construit. Puisque les protocoles du Web sont complètement indépendants des fournisseurs, des plateformes et des langages de mise en oeuvre, les applications résultant de la mise en oeuvre des technologies de services Web sont à la fois intégrables dans l'environnement de business, et suffisamment flexibles pour faire face aux modifications potentielles dans un monde très évolutif tel que celui du e-commerce.[6]

Les applications construites en se basant sur les services Web offrent les mêmes fonctionnalités que celles qui sont mises en oeuvre selon une architecture monolithique mais en plus on peut constater les bénéfices suivants : Une extension plus facile pour les règles de business afin de les appliquer aux nouvelles fonctionnalités "coopératives ". La flexibilité de modifications sans avoir besoin de reconstruire la structure de base. La réduction du coût d'intégration en utilisant des protocoles de référence.[6]

1.4.1 Définition

Un service Web est un service d'application (c'est-à-dire une fonction accessible depuis le réseau) qui peut être consulté en utilisant des protocoles standards du web. Les services Web indiquent entre autres l'ensemble des standards qui assurent l'interopérabilité entre les différents services. Le concept de service Web inclut les caractéristiques suivantes : [6]

- L'accessibilité depuis l'Internet : les services Web communiquent via des protocoles liés au Web, indépendants d'une plateforme ce qui facilite l'intégration entre des environnements hétérogènes.
- Les standards des services Web définissent une interface et un protocole de communication permettant l'enregistrement de ces services sur un serveur, pour faciliter l'invocation de leurs fonctionnalités par une application cliente.
- Le langage de définition de services Web WSDL (Web Services Definition Language) met en place une couche d'abstraction entre l'implémentation et l'interface, fournissant une application configurable. Ceci garantit la flexibilité de l'application résultante.

1.4.2 Les avantages de services web

L'adoption à une large échelle des technologies des services Web aux entreprises, d'effectuer une transition incrémentale vers une architecture orientée service avec des risques minimums et donc de limiter les coûts. Outre, les services Web ont les attouts suivants :[4]

- **L'interopérabilité** : C'est la capacité des services Web d'interagir avec d'autres composantes logicielles via des éléments XML et utilisant des protocoles de l'Internet.
- **La simplicité** : Les services Web réduisent la complexité des branchements entre les participants. Cela se fait en ne créant la fonctionnalité qu'une seule fois plutôt qu'en obligeant tous les fournisseurs à reproduire la même fonctionnalité à chacun des clients selon le protocole de communication supporté.
- **Une composante logicielle légèrement couplée** : L'architecture modulaire des services web, combinée au faible couplage des interfaces associées, permet l'utilisation et la réutilisation de services qui peuvent facilement être recombinés à différentes autres applications.
- **L'hétérogénéité** : Les services Web permettent d'ignorer l'hétérogénéité entre les différentes applications. En effet, ils décrivent comment transmettre un message (standardisé) entre deux applications, sans imposer comment construire ce message.

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

1.4.3 Architecture et Fonctionnement des services web

La définition de l'architecture des services Web consiste à mettre en évidence ses concepts, leurs relations ainsi que l'ensemble des contraintes existantes entre eux. Les principaux concepts intervenant dans l'architecture des services Web sont [4] :

- **le fournisseur du service** : désigne le serveur qui héberge les services déployés ;
- **le client du service** : représente l'application cliente qui invoque le service ;
- **le service** : désigne les fonctionnalités d'un agent logiciel qui implémente le service ;
- **la description du service** : c'est la spécification du service exprimée dans un langage de description interprétable par les machines, c'est-à-dire une description technique dans laquelle le service est vu en termes de messages, de types, de protocoles de communication et d'une adresse physique ;
- **les messages** : c'est la plus petite unité d'échange entre les clients et les services. La structure des messages qui permettent l'invocation d'un service doit être exprimée dans la description du service ;
- **la ressource** : désigne l'identifiant du service, c'est-à-dire son URI. Bien que le Web soit constitué de plates-formes totalement hétérogènes ou les intérêts des différents acteurs du marché s'entremêlent, ceci ne l'a pas empêché de se développer et d'être universel. Ce succès est du essentiellement à l'édition d'un ensemble de standards ouverts, dont les plus connus sont le protocole HTTP et le format MIME. Ensemble, ils offrent un mécanisme d'échange de données de toutes sortes quelle que soit la nature des plates-formes impliquées. Le développement des services Web a suivi la même approche en mettant en place une campagne de standardisation qui a touché les aspects les plus importants du développement d'un modèle d'intégration d'applications hétérogènes.

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

FIG. 1.4 – Architecture des services Web étendue

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

L'originalité de l'infrastructure des services Web consiste à mettre en place ces services exclusivement sur la base des protocoles les plus répandus sur Internet. Ces protocoles sont répartis selon quatre axes représentés dans la figure 1.4 :

- **Couche de transport** : cette couche s'occupe de transporter les messages entre applications en utilisant les protocoles HTTP, FTP ou SMTP ;
- **Message XML** : il s'agit de formaliser les messages à l'aide d'un vocabulaire XML commun (SOAP) ;
- **Description des services** : c'est la description de l'interface publique des services Web (WSDL) ;
- **Recherche de services** : la centralisation des services et de leur description dans un référentiel commun (UDDI). Au delà de cette dernière couche, on a deux autres couches suplémentaires pour les services Web étendues. La prémière (**Sémantique Web Service**) ne nous intéresse pas. Notre travail se déroule au niveau de la dernière couche (**BPM**) qui décrit le processus métier associé au service Web.

Fonctionnement des Services Web

Le cycle de vie d'un service Web se déroule de la façon suivante : une fois crée, le service est déployé sur le réseau (local ou Internet). Puis un utilisateur en cas de besoins recherche un service correspondant à ses besoins à l'aide d'un annuaire spécialisé(UDDI). Une fois le service

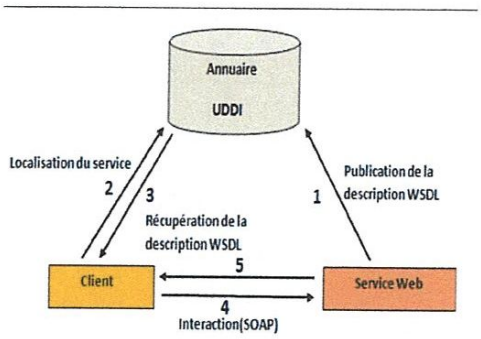

FIG. 1.5 – Fonctionnement des services web.

trouvé, l'utilisateur l'invoque. Ainsi une communication se met en place entre l'utilisateur et le service Web. Ce cycle de vie, représenté dans la figure 1.5, fait appel à trois grandes technologies : SOAP, WSDL, UDDI.[4]

1.4.4 Les standards utilisés par les services web

Les services Web sont bâtis en se basant sur des standards ouverts et largement adoptés comme http et XML. Ces standards sont maintenus par des organisations indépendantes sans but lucratif. Il en résulte des standards ouverts et gratuits selon lesquels des applications et des outils compatibles avec les services Web sont construits.

Dans les paragraphes suivants, nous allons introduire les standards qui jouent un rôle vital dans l'environnement des services Web.

a) XML : eXtensible Markup Language

XML concerne les spécifications liées à la customisation des types de documents, en utilisant des formats lisibles par l'Homme. Les spécifications de XML renforcent les règles de construction des documents. XML permet aux développeurs de créer leurs propres balises, permettant la définition, la transmission, la validation, et l'interprétation de données entre les applications et les organisations. Les services Web communiquent en utilisant XML (interfaces et messages de XML interprétables facilement par les applications) pour décrire leurs interfaces et encoder leurs messages et en s'appuyant sur les protocoles standards de web, comme les protocoles SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery and Integration), et WSDL (Web Services Description Language) pour réaliser l'interaction.[6]

Exemple 1.1 ***code XML d'un protocole métier de notre application :***

```
< ?xml version="1.0" encoding="UTF-8" ?>
<Automate>
  <Etats>
    <Etat Type="Initial" Position="(185,53)">Départ</Etat>
    <Etat Type="Simple" Position="(159,151)">Command envoyée</Etat>
    <Etat Type="Final" Position="(192,237)">Payé</Etat>
  </Etats>
  <Transitions>
    <Transition Etat_Source="Départ " Etat_Destination="Command
          envoyée " Position="(153,95)">Commanderbien</Transition>
    <Transition Etat_Source="Command envoyée "Etat_Destination="Payé"
          Position="(144,182)">Effectuerpaiment</Transition>
    <Transition Etat_Source="Command envoyée" Etat_Destination="Payé"
          Position="(155,198)">Effectuerpaiment</Transition>
    <Transition Etat_Source="Départ" Etat_Destination="Command
          envoyée" Position="(168,108)">Commandcrbien</Transition>
  </Transitions>
</Automate>
```

b) SOAP : Simple Object Access Protocol

SOAP est un standard qui représente une « enveloppe » légère contenant la charge utile "payload" des messages échangés entre les producteurs et consommateurs de services. C'est un standard basé sur XML qui décrit le contenu du message échangé ainsi que la façon de traiter ce contenu. Ce protocole permet en outre une association avec les protocoles de transport. Il offre aussi un ensemble de règles d'encodage pour exprimer les instances des types de données définies par les applications et une convention pour représenter les invocations à distance et les réponses des applications.[6]

Structure d'un message SOAP SOAP définit un format pour l'envoi des messages. Les messages SOAP sont structuré en un document XML et comporte deux éléments obligatoires : une enveloppe et un corp ainsi qu'une entête facultative. Le protocole SOAP est une surcouche de la couche application du modèle OSI des réseaux. Le protocole applicatif le plus utilisé pour transmettre les messages SOAP est HTTP, mais il est également possible d'utiliser les protocoles SMTP ou FTP, la norme n'impose pas de choix.[4]

Principe de fonctionnement de SOAP

*** Coté client**

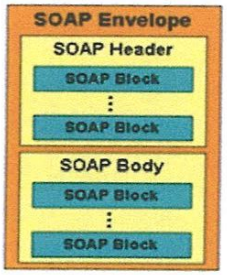

FIG. 1.6 – Structure de base des messages SOAP.[4]

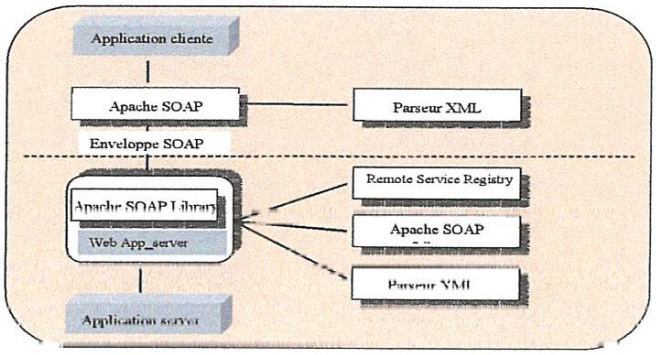

FIG. 1.7 – Principe de fonctionnement de SOAP.[4]

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

*** Coté serveur**

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

Exemple 1.2 ***requête SOAP/HTTP :***

Poste/ stockQuote http/1.1
Host : www.stockquoteserver.com
Content-type :text/xml;charset="utf-8"
Content-length :nnnn
SAOPAction :"URI"
<soapenv :Envelope
Xmlns :soapenv–http ://schemas.xmlsoap.org/soap/envelope/">
<Soapenv :Body>
<m : GETLastTradePrice xmlns :m="Some-URI">
<m : tickerSymbol>DIS</m : tickerSymbol>
</m : GETLastTradePrice>
</Soapenv :Body>
</ soapenv :Envelope>.[4]

Exemple 1.3 ***réponse SOAP/HTTP***

HTTP/1.1 800 OK
Content-Type : text/xml; charset="utf-8"
Content-Length : nnnn
<soapenv :Envelope xmlns :soapenv="http ://schemas.xmlsoap.org/soap/envelope/">
<soapenv :Body>
<m :GetLastTradePriceResponse xmlns :m="Some-URI">
<m :price>34.5</m :price>
</m :GetLastTradePriceResponse>

</soapenv :Body>
</soapenv :Envelope>.[4]

c) WSDL : Web Services Description Language

WSDL est un langage au format XML utilisé pour la description des services offerts sur un réseau. Ce langage décrit les interfaces des messages contenants des information orientées documents ou orientées procédures. Les opérations et les messages sont décrits de manière abstraite. Ces descriptions sont ensuite couplées au système de transport (spécifiant le format des messages et les protocoles réseau) pour définir les points de connexion aux services afin de définir une extrémité. Les messages sont associés au protocole SOAP et au protocole de transport HTTP. La nature abstraite de WSDL (défini comme un outil de description pour les services) fournit la flexibilité nécessaire pour décrire des applications complexes des services web. Un document WSDL utilise les éléments suivants pour la définition des services :[6]

- **Types** : conteneur pour la définition de type de données comme XSD (XML Schema Definition).

<types>
<schema targetNamespace="http ://example.com/stockquote.xsd" xmlns="http ://www.w3.org/2000/10/XMLSchema">
<element name="TradePriceRequest">
<complexType>
<all>
<element name="tickerSymbol" type="string"/>
</all>
</complexType>
</element>
<element name="TradePrice">
</element>
</schema>
</types>.[4]

- **Message** : définition typée abstraite des données communiquées.

<message name="GetLastTradePriceInput">
<part name="body" element="xsd1 :TradePriceRequest"/>
</message>
<message name="GetLastTradePriceOutput">
<part name="body" element="xsd1 :TradePrice"/>
</message>.[4]

- **Opération** : description abstraite pour une action supportée par le service
- **Type de port** : groupe d'opérations abstraites supportées par une ou plusieurs extrémités.

<portType name="StockQuotePortType">
<operation name="GetLastTradePrice">
<input message="tns :GetLastTradePriceInput"/>
<output message="tns :GetLastTradePriceOutput"/>
</operation>
</portType>.[4]

- **Association** : protocole concret ainsi qu'une spécification pour un format de donnée dédié à un type de port précis.
- **Port** : extrémité définie comme une combinaison d'une association avec une adresse réseau.
- **Service** : collection d'extrémités reliées.

<service name="StockQuoteService">
<documentation>My first service</documentation>
<port name="StockQuotePort" binding="tns :StockQuoteBinding">
<soap :address location="http ://example.com/stockquote"/>
</port>
</service>.[4]

d) UDDI : Universal Description, Discovery and Integration

UDDI est une norme d'annuaire de services Web appelée via le protocole SOAP. Pour publier un nouveau service web, il faut générer un document appelé Business Registry qui sera enregistré sur un UDDI Business Registry Node. Le Business Registry comprend 3 parties :[4]

- **Pages blanches** : noms, adresses, contacts, identifiants des entreprises enregistrées
- **Pages jaunes** : informations permettant de classer les entreprises, notamment l'activité, la localisation, etc.
- **Pages vertes** : informations techniques sur les services proposés.

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

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

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

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

Comme pour la certification, il est possible de constituer des annuaires UDDI privés, dont l'usage sera limité à l'intérieur de l'entreprise.[4]

Conclusion

Les concepts fondamentaux inhérents à ce prémier chapitre étant clarifiés, le suivant portera sur deux éléments prépondérants de notre sujet à savoir : les processus métiers (Business Process) et les traces.

Chapitre 2

PAIS, BP et Traces d'exécutions

Introduction

Au cours des deux dernières décennies , la conception des systèmes d'information d'entreprise a évolué du simple système "orienté donnée" au système "orienté processus". Pour assurer le besoin de controler les processus métiers, le SI de l'entreprise doit se rendre compte de ses processus et de leur contexte d'organisation. Ceci a donné naissance à une nouvelle génération de systèmes d'informations nommés **Systèmes d'information à processus conscient** (*en anglais* **Process Aware Information Systems, PAIS).** Des exemples prototypiques de systèmes se sont les systèmes de controle de flux opérationnels ou flux de travaux (*en anglais* **WorkFlow Management, WFM**). Plus recemment, les fournisseurs préfèrent le terme systèmes de contrôle de **système métier** (*en anglais* **Business Process Management, BPM**). Les systèmes BPM ont une portée plus large que les systèmes classiques de WFM et ne se concentrent pas simplement sur l'automatisation de processus, ils tendent de fournir plus d'appui pour différentes formes d'analyse (par exemple, la simulation) et de gestion (par exemple, la surveillance). WFM et BPM visent ensemble à soutenir les processus opérationnels souvent rapportés aux simples flux de travaux (workflow). Nous emploierons le terme générique PAIS pour nous référer aux systèmes qui contrôlent et exécutent de tels workflows. Les processus que doivent controler les PAIS constituent les règles de gestions de l'entreprise ou l'organisation dans un cadre d'interopérabilité ou de management éfficace de ses ressources. Nous désignerons par **Business Process (BP)** ces processus.[7]

Dans l'architecture orienté service(SOA) le système d'information est vue comme un ensemble de services reliés. Un PAIS ne peut être réalisé qu'en utilisant une telle architecture et en général, il est très naturel de voir les processus comme des services reliés. L'ajustement entre la SOA et le PAIS est illustré par des normes naissantes telles que BPEL (**Business Process Execution Langage**) et BPMN (**Business Process Management Notation**). La focalisation sur les services Web et la SOA a haussé l'enthousiasme vers l'orienté processus. En conséquence on s'attend à ce qu'à l'avenir le PAIS générique commence à jouer un rôle plus

important. Cependant, en même temps il n'est pas à négliger que la plupart des PAIS sont consacrés vers un domaine particulier d'application ou même une compagnie spécifique.[7]

Vue le changement constant des réalités de l'environnement de l'entreprise, ses SI doivent réfleter ces réalités à tout moment, donc ils sont également sujets à des changements constants. Pour renforcer l'efficacité du controle des SI ainsi que des clients afin de les garder et par conséquent faire plus de profit, il est imprératif d'avoir une historique de leurs utilisations par les clients. Dans la suite nous désignerons cette historique par la notion de **traces.**

2.1 BP (Business Process)

Le processus métier (ou Business Process) est un composant élémentaire du processus global dans la chaîne logistique.

Un exemple typique de processus métier est le traitement d'un ordre d'achat. Les activités constituant ce processus commencent par la prise en compte de la commande et se terminent par la livraison au client. Entre ces deux extrémités, plusieurs activités intermédiaires se déroulent selon un chemin **connu à l'avance** afin de garantir le fonctionnement correct et cohérent de ce processus.

2.1.1 Définitions

Plusieurs définitions ont été données comme les suivants :

- Le processus métier est défini comme un groupe d'activités qui permettent d'atteindre collectivement un objectif de business dans un contexte inter organisationnel.[6]
- Un Business Process ou Processus Métier est un ensemble de règles métiers (business), de définitions d'activités et d'événements déclencheurs qui fournissent un contexte d'échange d'information.[4]
- Un Processus Métier est une unité persistante d'un travail pouvant avoir un nombre important de transactions. Il est déclenché par un événement métier tel que l'invocation, requête pour proposition ou une requête pour un transfert de fonds. Le processus est conduit par des règles métiers qui déclenchent les tâches et les sous-processus, avec chaque transition d'état exécutée dans une transaction et auditée pour des raisons métiers. Les tâches et les sous-processus sont assignés à des ressources qui sont des unités organisationnelles capables et autorisés à jouer des rôles spécifiques dans le processus. La définition des règles, des tâches, des sous-processus, et des stratégies de ressources constitue une description de processus. L'exécution d'un processus métier consiste en l'invocation des services métiers existants, qui peuvent résider n'importe où.[4]

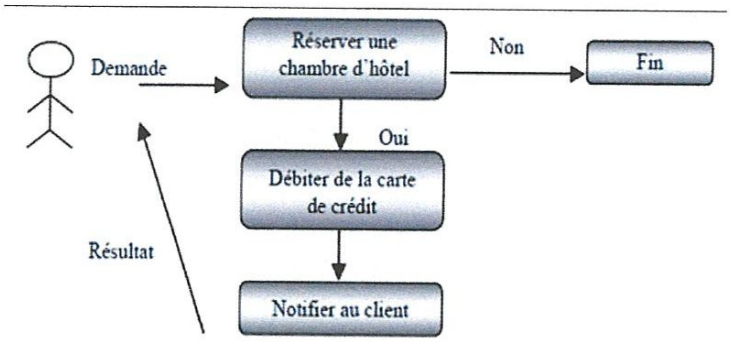

FIG. 2.1 – Exemple de processus métier.[4]

- Un Business Processus est une séquence d'activité relative à une ou plusieurs organisations permettant d'exprimer une procédure de gestion particulière.[Mr Khebizi Ali,2012][16]
- Un Processus Métier est un ensemble d'activités conçues pour produire des outputs spécifiques pour un marché ou un client particulier. Il peut être défini sur la base de trois dimensions :

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

Objets : les processus sont le résultat d'une manipulation d'objet. Ces objets peuvent être physiques ou informationnels comme l'illustre la fig 2.1 (un exemple de BP).

- **Activités** : les processus peuvent impliquer deux types d'activités : les activités opérationnelles et les activités de gestion.[4]

2.1.2 Qu'est ce qu'un protocole, un processus ?

Un protocole, en général, désigne un ensemble de règles régissant l'utilisation d'une chose. En informatique on peut le définir comme un ensemble de règles de communication qui permet à deux ou plusieurs entités (ordinateurs, applications logicielles, périphériques d'ordinateur, etc.) d'échanger des données entre elles. Par analogie, un protocole peut être comparé à une langue : pour que deux personnes puissent se comprendre, il faut qu'elles utilisent la même langue, et de la même manière, pour que deux entités puissent échanger des données, elles doivent impérativement utiliser des protocoles compatibles.[8]

Dans notre contexte, un processus est une description informelle de ces règles. C'est pourquoi on dit qu'un protocole est un processus formel. Dans la suite, nous désignérons par protocole métier (BP) un processus métier formalisé.

2.1.3 Classification des processus métier

Un processus métier peut être classifié selon [6] :

La fréquence d'exécution

- Processus ad hoc : exécuté une fois, par exemple la construction d'une chaîne logistique.
- Processus de production : haute fréquence, par exemple la construction d'une série de maisons de même type.
- Processus de production de masse : très haute fréquence, par exemple, la production de voitures.

Le classement des processus métier nous permet d'analyser les différentes exigences concernant la modélisation des processus selon leur type. Par exemple, quand il s'agit d'un processus de production, l'accent est mis sur la logique de construction afin de minimiser le temps d'exécution. La logique de construction ou d'assemblage est aussi importante pour la composition du processus métier commun. En revanche dans ce dernier cas, l'objectif sera de garantir son bon fonctionnement. A cet effet là, un outil efficace de management pour le processus métier est nécessaire afin de faciliter l'intervention des acteurs et utilisateurs sur les différents niveaux de processus. La sous section suivant porte sur la gestion du processus métier ou Business Process Management (BPM).

2.1.4 Business Process Management

Un processus métier est modélisé en plusieurs niveaux, et plus généralement en trois niveaux :[9]

- **Le niveau métier** : vue métier et haut niveau du processus, définissant ses principales étapes et l'impact sur l'organisation de l'entreprise. Ce niveau est défini par les décideurs, et les équipes méthodes de l'entreprise.
- **Le niveau fonctionnel** : formalisation des interactions entre les participants fonctionnels du processus, où sont formalisées les règles métiers conditionnant son déroulement. Ce niveau est modélisé par les équipes fonctionnelles.
- **Le niveau technique** : lien entre les activités / participants modélisés dans le niveau fonctionnels, et les applications / services du SI, ainsi que les tâches utilisateurs (Workflow). Ce niveau est réalisé par les architectes et les équipes techniques de l'entreprise.

  Ces modélisations par niveau sont effectuées par différents acteurs et faites par des outils différents, ce qui rend difficile voire impossible l'utilisation d'un modèle, mise en oeuvre à un niveau, par le niveau suivant.

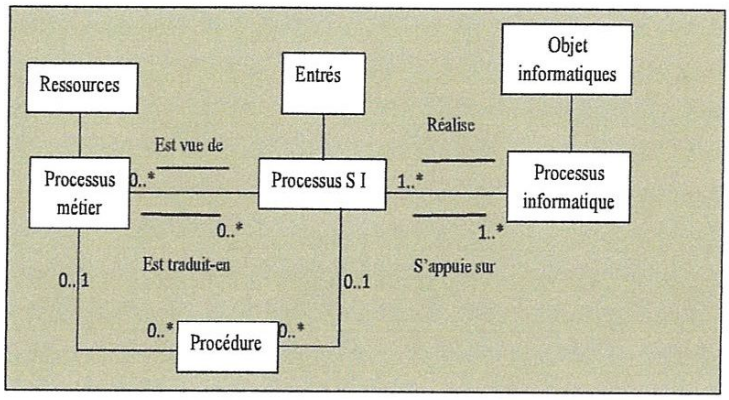

FIG. 2.2 – Processus métier, processus SI, processus informatique.[4]

L'enjeu du BPM est d'unifier sous un seul outil toutes ces visions, pour fournir à l'entreprise la possibilité de définir ses processus au niveau métier, et de faire intervenir les utilisateurs et les applications de l'entreprise en tant que partie prenante à ces processus. L'objectif est de permettre aux décideurs, analystes métiers, équipes fonctionnelles et équipes techniques de collaborer pour la définition et l'évolution des processus métiers via un seul outil aggrégeant les différentes visions.[9]

Un des objectifs du BPM est de capitaliser sur les applications du système d'information : le mot d'ordre est la réutilisabilité. Nous verrons dans la suite de ce document que, contrairement aux initiatives précédentes architectures objets, dans les services Web la réutilisabilité est rendue effectivement possible, tant au niveau technique (connecteurs, règles techniques, transformation de données) que fonctionnel (réutilisation d'un processus de facturation dans un processus de gestion de bons de commande par exemple).

2.1.5 Processus métier et Systèmes d'Informations

Les processus métiers permettent d'organiser et de répartir les tâches entre les différents acteurs d'une entreprise. Ceci contribue à la réalisation des objectifs stratégiques de celle-ci. Ces processus métiers obéissent également à des règles et procédures représentant le point de vue organisationnel de celle-ci. Un processus métier est mis en œuvre à l'aide du SI de l'entreprise. Ainsi, un processus métier peut être décomposé en un ou plusieurs processus SI. [4]

Ces processus SI représentant le tout ou une partie du processus métier participent aux mêmes objectifs mais se focalisent sur la mise à disposition et le traitement de l'information. Ils sont mis en œuvre par des processus informatiques, soit un ensemble d'activités logicielles, exécutées par des machines et dialoguant éventuellement avec des humains. L'articulation de ces différents concepts est représentée par la fig 2.2.

2.1.6 Processus métier et Service Web

La sous section ci-dessus nous révèle que les processus métier permettent d'organiser et de répartir les tâches entre les différents acteurs d'une entreprise. Cette dernière est composée d'une ou plusieurs processus métiers, son SI doit recouvrir l'ensemble de ces BP. En supposant que le SI de l'entreprise adopte la tehnologie des services Web, la seule manière recouvrir l'ensemble des BP est de représenter chacun par un ou plusieurs services Web. Chaque service Web correspond à un processus métier de l'entreprise, ainsi l'exécution d'un processus métier consiste à exécuter le service associé. Un service Web est décrit par une interface (WSDL *voir chapitre I*) exposant l'ensembles de ses règles d'utilisation. Des expériences d'utilisation de services ainsi que la composition interne même de cette description classique nous démontrent qu'elle est insuffisante pour une bonne gestion du dit service. L'ordre d'exécutions des différentes tâches (opérations) composantes du service n'est pas décrit par cette interface. Or, pour assurer la cohérence dans l'exécution du service, il faut un certain agencement entre ses différentes opérations. Par exemples :

Exemple 2.1 *Soit un service du domaine de e-commerce constitué des tâches suivantes :* ***s'identifier, commander, confirmer, payer, livrer****. Un client quelconque, sans une description de l'organisation des tâches, peut les exécuter dans n'importe quel ordre. Ce qui eventuellement suiscite des incohérences dans l'exécution du service web.*

Exemple 2.2 *A cela on peut ajouter la notion du temps dans l'exécution des différentes tâches. Supposons que dans l'exécutions les tâches précedentes, la paye doit être effectué au bout d'un temps* ***t*** *après la confirmation. Si le client n'est pas informé de cela dans la description du service, l'exécution du dit service peut s'averer fastidieux pour lui avec des initialisations infinies. Ce qui peut être cause de baisse de profit, vis à vis de l'entreprise, due à des pertes de clients.*

Pour remédier à ces insuffisances du WSDL, on a recours à l'utilisation des processus métier qui constituent un compléments descriptifs de l'interface des services web. Outre, cette agencement des opérations permet une facilité d'inter opérabilité entre différents services dans une perspective de composition de ces derniers. Cet aspect de composition ne fait pas l'objet de ce mémoire.

2.1.7 Interopérabilité intra/inter-organisationnelle avec les processus métier

Construire des processus métier inter-organisationnels requière une intégration à des niveaux différents, et plus spécialement aux niveaux communication et processus métier. La couche

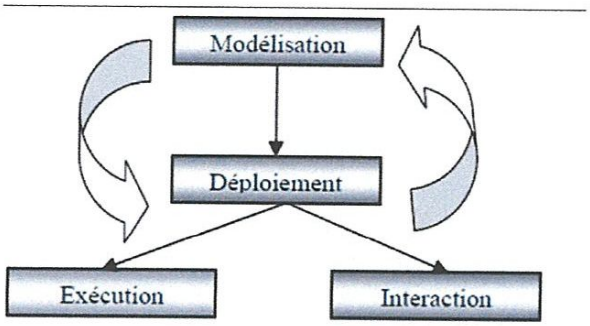

FIG. 2.3 – Cycle de vie d'un Processus Métier.

communication est concernée par l'échange de messages entre partenaires. L'objectif d'intégration de cette couche est de permettre des interactions transparentes entre partenaires. Celle-ci, étant une tâche difficile, peut être assurée en utilisant les API (Application Programming Interface) et les workflows. Pour ces raisons, les services Web offrent un cadre conceptuel et une plate forme technologique adéquate pour l'interopérabilité.

En effet, comme décrit plus haut dans la section consacrée aux services web, l'utilisation de ces derniers pour l'interopérabilité est bénéfique car il favorise la réutilisabilité et facilite la maintenance du système naissant, en ne se focalisant que sur le service concerné.

2.1.8 Le cycle de vie d'un processus métier

Selon « Tanguy Crusson » le cycle de vie d'un processus métier est présenté sur quatre étapes. Ces étapes sont illustrées dans la fig 2.3.[4]

- **Modélisation du processus métier :** Un modèle de processus métier consiste en un ensemble de modèles d'activités et de contraintes d'exécution entre eux. Elle représente la phase où l'importation des définitions des processus métiers dans les outils techniques est possible, afin d'ajouter le lien avec les applications du SI. Les capacités de modélisation des outils devraient permettre de déployer directement les processus modélisés sans passer par une phase d'implémentation.
- **Déploiement du processus métier :** Consiste à déployer le code réalisé durant la phase de modélisation
- **Exécution du processus métier :** Consiste en l'exécution des applications existantes de l'entreprise.
- **Interaction du processus métier :** Désigne l'interaction des utilisateurs avec ce processus. Une fois le processus déployé, la phase d'interaction permet de l'optimiser.

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

2.1.9 Modélisation des processus métiers

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

Dans cette section, nous analyserons ces différents modèles.

a) Les réseaux de pétri

Les Réseaux de Pétri (RdP) ont été introduits par Carl Adam Pétri, en 1962, afin de modéliser et d'analyser les comportements des systèmes d'une manière graphique en se basant sur un modèle mathématique. Les RdP sont largement utilisés pour la modélisation des processus métier. Comme l'illustre le RdP de la figure 2.4 , un réseau de Pétri est un graphe orienté et biparti c'est à dire qu'il a deux types de nœud : **les places et les transitions**. Une place modélise une condition ou l'état d'une ressource du système et une transition désigne une action à effectuer. Un réseau de pétri est défini alors par le tuple SN (P, T, W, i, o, l) avec :[4]

P : ensembles de places,

T : ensemble de transitions représentant les opérations du service

$W \subseteq (P*T) \cup (T*P)$: ensemble d'arcs

i : le point d'entrée du service, $i = \{x \in P \cup T (x,i) \in W\} = \emptyset$,

o : le point de sortie du service, $o = \{x \in P \cup T (o,x) \in W\} = \emptyset$,

$l : T \rightarrow A \in \{t\}$ avec :

A : l'ensemble des noms des opérations

$t \notin A$, t est une opération qui ne porte pas de nom

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

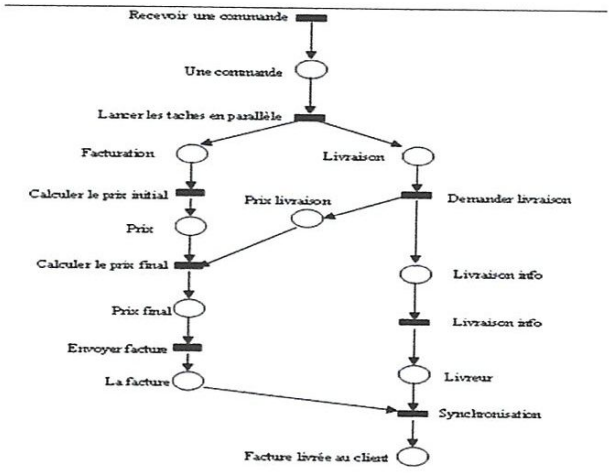

FIG. 2.4 – Le processus de traitement de commande modélisé en RdP.[4]

b) MCT : Model Conceptuel de traitement de Merise

Le MCT issu de Merise modélise les activités du domaine, activités conditionnées par les échanges avec l'environnement, sans prise en compte de l'organisation. Ainsi, chaque activité (nommée opération) regroupe un ensemble d'activités élémentaires réalisables au sein du domaine, sans autres informations extérieures (on n'a pas besoin de s'arrêter pour attendre des informations extérieures). De ce fait, une opération du MCT présente une vision macroscopique qui en fait l'intérêt dans l'analyse des processus métier, en particulier dans le Business Process Reengineering .[4]

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

c) Unified Modeling Language (UML)

UML est un langage pour spécifier, visualiser, construire, et documenter les artefacts des systèmes logiciels, ainsi que pour la modélisation d'entreprise et des systèmes non logiciels Comme son nom l'indique, ce langage est né de la fusion de plusieurs langages de modélisation, notamment de la méthode de Grady Booch particulièrement adaptée à la conception et à l'implémentation, de la méthode OOSE (Object Oriented Software Engineering) de Ivar Jacobson : qui permettait essentiellement l'expression des besoins, et de la méthode OMT (Object Modelling Technique) de James Rumbaugh : pour l'analyse et applications orientées données.[4]

UML propose des diagrammes spécialisés (dont les diagrammes d'activité, de séquence,

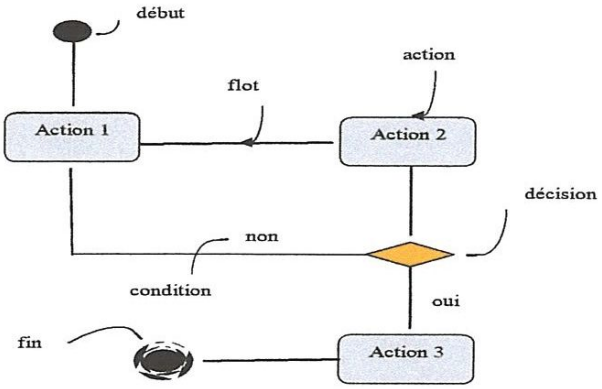

FIG. 2.5 – diagramme d'activité UML d'un BP.

de classes. . .) ayant chacun une fonction précise. Il n'existe pour le moment pas de diagramme UML spécialisé pour la modélisation des processus métiers. UML propose cependant un mécanisme d'extensibilité permettant de spécialiser chaque diagramme pour une utilisation particulière. Il est par exemple possible de spécialiser les diagrammes d'activité pour la modélisation des processus métiers.[4]

Remarque 2.1 *:*

La panoplie de diagrammes de modélisation qu'offre UML permet de satisfaire les différents niveaux de modélisation des processus métiers, depuis la description générale jusqu'à la conception détaillée. Cependant le manque d'outils d'analyse rigoureux dans UML peut réduire sa capacité. Pour cela nous recourons aux réseaux de Pétri pour remédier à cet inconvénient. On peut constater que les réseaux de Pétri et UML possèdent des caractéristiques réciproques :[4]

- UML est convivial alors que les réseaux de Pétri possèdent une rigueur formelle.
- UML peut décrire les systèmes de façon efficace pendant que les RdPs peuvent les analyser strictement.
- Le modèle UML peut être facilement implémenté tandis que le modèle des RdPs est plus adéquat pour la simulation.

d) OWL-S (Ontology Web Language for Services)

Aussi appelé DAML-S, il vise à réaliser une vision de Web sémantique pour rendre les ressources Web disponibles aussi bien par contenu que par mots clés. OWL-S organise les services Web en ontologies. Une ontologie est définie par une spécification formelle et explicite d'une

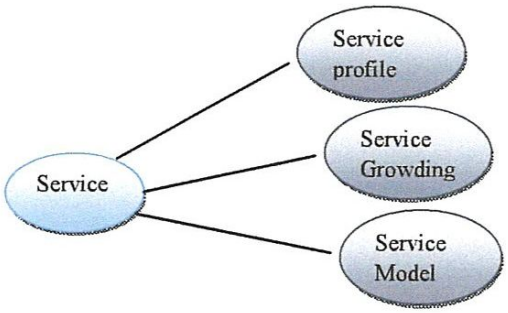

FIG. 2.6 – Exemple d'OWL-S.[4]

conception partagée. OWL-S divise les descriptions de service en profil de service, modèle et fondement. Le profil de service fournit une description haute niveau d'un service Web. Il exprime l'entrée requise du service et les sorties que le service va fournir au demandeur. Le modèle de service définit les opérations et leurs flots d'exécution dans le service Web. Le fondement de service fournit un mapping entre OWL-S et le standard WSDL et décrit comment le service est invoqué. C'est un modèle pour la description déclarative de processus de services web. Leur transformation en systèmes d'états transitions permet de les soumettre à un moteur de planification (et ensuite pour les auteurs de comparer les performances entre une approche de composition sémantique vs. exécutable). La tâche de composition consiste alors à trouver un plan satisfaisant un but de composition dans le domaine des services disponibles. Les plans générés par l'algorithme de planification sont des automates pouvant être traduits en code exécutable.[4]

e) Le processus BPEL abstrait

L'utilisation du langage BPEL est la définition de protocoles métiers entre plusieurs partenaires également désignés comme « processus abstraits ». Un processus abstrait ne décrit que les aspects comportementaux entre processus BPEL indépendamment de leur exécution plaçant l'ensemble dans un mode de fonctionnement plus proche de la chorégraphie que de l'orchestration de services Web. Pour rappel, un protocole définit un mode de communication en établissant des règles d'échange de messages entre plusieurs intervenants. La définition d'un protocole métier nécessite d'adresser plusieurs points en particulier :[4]

- La description du comportement en ajoutant éventuellement l'utilisation de constructions logiques et temporelles de traitement des informations.
- La description des conditions d'exceptions, leurs conséquences et leurs activités de recouvrement.

- La description des interactions de longues durées et la coordination du succès ou de l'échec des composantes à différents niveaux de granularité.

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

f) BPMN (Business Process Modeling Notation)

Le BPMI issu d'un regroupement d'entreprises, a développé le standard Business Process Modeling Notation (BPMN). Ce dernier a pour objectif primaire de fournir une notation compréhensible par tous les utilisateurs métiers ; des analystes métiers qui créent des drafts initiaux de processus, aux développeurs techniques responsables de l'implémentation de la technologie qui exécute ces processus, et finalement, aux gestionnaires et ceux qui surveillent ces processus. Il crée un lien standardisé entre la conception et l'implémentation des processus.[4]

g) Automates

Un automate est un modèle très connu dans le domaine de la spécification formelle de systèmes. Un automate est composé d'un ensemble d'états ou nœuds, un ensemble d'actions et un ensemble de transitions étiquetées entre les états. Les actions sont modélisées par des étiquettes et une transition étiquetée modélise ainsi l'exécution de l'action lors du franchissement de celle-ci.[4]

Définition Un automate est représenté formellement par un 5-uplet (Q, Σ, δ, q_0 , F), où :

- Q est un ensemble d'états
- Σ est un ensemble fini de symboles, appelé alphabet ou langage reconnu par l'automate (un automate peut reconnaître un langage formel)
- δ est la fonction de transition, telle que $\delta : Q \times \Sigma -> Q$
- q_0 est l'état initial, c'est à dire l'état dans lequel est l'automate lorsqu'aucune entrée n'a encore été traitée, où $q_0 \in Q$
- F est un ensemble d'états de Q (c'est à dire $F \subseteq Q$) appelés états accepteurs. Ils ont aucune transition sortante.

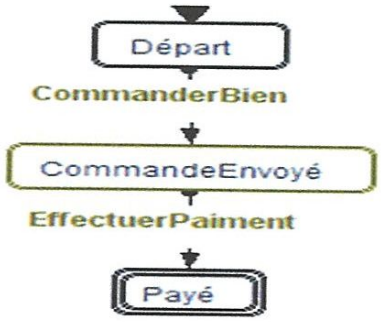

FIG. 2.7 – Modélisation avec les automates.

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

Type d'automate De nombreux types d'automates existent, entre autres nous avons : les automates à états finis, Entrée/Sortie, temporisés... Les modèles à base d'automates peuvent être utilisés pour décrire, spécifier et vérifier la composition de services. Pour modéliser la composition de service à l'aide d'un automate, on assigne généralement un état à chaque invocation de service, un événement à chaque réponse d'un service et un deuxième état pour indiquer la fin de l'activité d'invocation. Un exemple d'une telle modélisation est fourni dans la figure 2.7

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

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

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

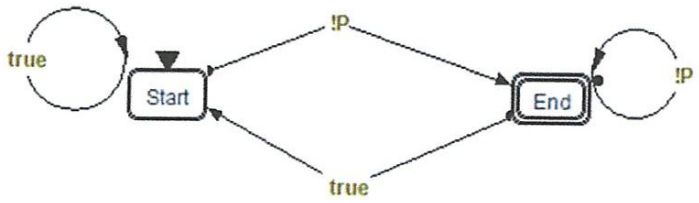

FIG. 2.8 – Représentation graphique d'un automate d'états déterministe.

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

Automate à états finis déterministe (AEFD) Dans ce type d'automate la fonction de transition Π est définie par Π : Q×X ⟶ Q telle que Π (q, x)=q' q ∈ Q et x ∈ X, c'est à dire que l'image par Π d'un couple (q, x) est un état unique de l'automate.

Remarque 2.2 :

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

Dans notre travail, les protocoles d'un service Web seront représentés par des AEFD.

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

Les automates à entrées/sorties (E/S) Ce sont des automates où les actions peuvent être des entrées, des sorties où des comportements internes. Ils sont conçus pour permettre la modélisation de systèmes à événements discrets formés de composants concurrents. Les automates E/S ont initialement été introduits pour modéliser les calculs distribués dans les réseaux asynchrones et pour la preuve d'algorithmes distribués. Ils sont désormais largement utilisés pour modéliser tous types de systèmes distribués et réactifs.

Choix du modéle : l'utilisation des AEFD pour la modélisation de nos protocoles métiers est motivée par les raisons suivantes :

* Ils sont facile à comprendre pour les utilisateurs,
* appropriés pour décrire le comportement réactif,
* utiles pour le contrôle d'exécution de services,
* ce sont des outils populaires et très répandus,
* l'existence d'outils logiciels pour leur vérification,
* assis théorique très solide.

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

2.1.10 Notion de scénarios

Littérairement un scénario défini un enchainement de faits qui constituent la structure d'un recit. Par analogie à cette définition, on comprend qu'un processus métier constitue le scénario d'une ou plusieurs activités dans un domaine bien précis. Par exemple dans le e-commerce, le e-learning, le e-gouvernement. En d'autres termes, le scénario est spécifique au domaine d'application. Dans notre mémoire on s'intéressera à deux scénarios e-commerce et sécurité sociale : retraite.

2.2 PAIS (Process-Aware Informations Systems)

2.2.1 Définition

Traduit en français par Systèmes d'Information à Processus Conscient, c'est un système logiciel qui contrôle et exécute des processus opérationnels impliquant des personnes, des applications, et/ou des émetteurs d'informations sur la base de modèles de processus.[7]

Bien que pas explicitement indiqué dans cette définition, il devrait être noté que les modèles de processus mentionnés sont habituellement représenté dans certains langage graphique, par exemple, un réseau de Pétri, automates d'états finis, ... Les modèles sont en général instanciés plusieurs fois (par exemple, pour la commande de chaque client) et chaque exemple est manipulé d'une manière bien définie (probablement avec des variations).

2.2.2 Exemples

Des exemples classiques de PAIS sont les systèmes de gestion de flux d'opérations (WFM) et les systèmes de contrôle de processus d'affaires (BPM). Ces systèmes soutiennent des processus opérationnels d'affaires et sont conduits par une représentation explicite de processus. Etant donné la définition ci-dessus, on peut voir qu'un éditeur de texte n'est pas (orienté processus) étant donné qu'il est employé pour faciliter l'exécution des tâches spécifiques sans rien connaitre sur le processus auquel ces tâches font partie. Un commentaire semblable peut être fait sur les clients d'e-mail utilisés pour envoyer et recevoir le courrier électronique. Une tâche en processus peut être inclus dans un e-mail envoyé sans que le client d'e-mail soit au courant du processus déployé dedans. À tout moment, on peut envoyer un e-mail à n'importe qui, sans être soutenu ni limité par le client d'e-mail. Les éditeurs de texte et les clients d'e-mail (sauf ceux d'actualité) sont des applications supportant des tâches et non des processus. Le même s'applique à un grand nombre d'applications utilisées dans le contexte des systèmes d'informations.[7]

2.2.3 Avantages

La conscience de processus est une propriété importante pour des systèmes d'information et le décalage des tâches vers les processus apporte un certain nombre d'avantages :[7]

- L'utilisation des modèles de processus explicites fournit des moyens pour la communication entre les personnes.
- Les systèmes conduits par des modèles plutôt que des code ont moins de problèmes à traiter le changement, c'est à dire, si un système d'information est conduit par les modèles de processus, seulement les modèles doivent être changés pour soutenir l'évoluant ou des processus naissants d'affaires.
- La représentation explicite des processus soutenus par une organisation permet leur établissement automatisé. Ceci peut mener à une meilleure performance.
- La représentation explicite des processus permet l'appui de gestion au niveau de conception, c'est à dire, un support de modèles de processus.
- La représentation explicite des processus permet également l'appui de gestion au niveau de commande. Les équipements de surveillance et d'extraction .

2.3 Traces d'exécutions

2.3.1 Définition

Le dictionnaire Robert donne la définition suivante : "Trace : suite d'empreintes ou de marques que laisse le passage d'un être ou d'un objet".[10]

L'enregistrement systématique et temporaire d'un certain nombre d'informations caractérisant chaque transaction, appelées traces, est une contribution importante pour la sécurité et le contrôle des systèmes d'information et des moyens de communication.

Dans un système d'information, toute action d'un utilisateur ou d'un composant est susceptible de laisser des traces. Celles-ci se présentent sous la forme d'enregistrements stockés en cours d'exécution dans des fichiers. Elles peuvent provenir de tout composant du SI

On peut citer comme exemples d'actions traçables :[10]

- les connexions réussies à un système,
- les requêtes applicatives,
- les accès aux serveurs de bases de données, aux services Intranet ou Internet,
- les actions de télémaintenance,
- les tentatives de prise de contrôle d'un équipement informatique,
- les tentatives d'intrusion dans le système d'information de l'entreprise,
- etc.

Ces traces peuvent être utilisées soit :

- immédiatement, par exemple par un système de détection d'intrusions,
- de manière différée, par des équipes d'analyse et de recherche.

2.3.2 Pourquoi le traçage informatique ?

Pour l'entreprise, le traçcage est le moyen de mémoriser l'ensemble des tâches effectuées ou des décisions prises par une personne. Elle témoigne donc d'une démarche de qualité dans le fonctionnement de l'entreprise. Par exemple :[11]

Exemple 2.3 *Prenons l'exemple du contrôleur aérien. Lorsqu'il est dans sa tour de contrôle, il donne un ensemble d'indications aux avions pour leur donner les ordres d'atterissages, de décollages ou de manoeuvres à effectuer. On comprend aisément que tous ses faits et gestes, dans le cadre de son travail, doivent être mémoriser. En cas de problème, les données enregistrées permettront aisément de comprendre les raisons du problème survenu.*

Exemple 2.4 *Le boursier qui passe des ordres sur le marché. Chaque fois qu'il passe une commande d'achat et de ventes de transactions, il souhaite que l'on mémorise toutes ses actions. Dans le cas d'un problème X ou Y, il sera content, grâce aux systèmes informatiques, de retrouver l'ensemble des transactions qu'il a effectué. Elle lui serviront de preuve et témoigneront de ses actions.*

C'est une mesure de sécurité destinée à detecter (voire protéger le SI de l'entreprise contre) les menaces qui pèsent sur lui, entre autres nous avons :

- L'accès non autorisé qui est une menace permanente, et peut engendrer des perturbations du fonctionnement du système ou bien donner accès à des informations confidentielles.
- Les malversations produites, par exemple, par une action frauduleuse sur des données d'entrée.
- Les denis de service, ...

Il se révèle également être un attout incontestable pour la reprise du système après une panne comme le démontre les deux exemples ci-dessus, donc une mesure de sécurité corrective aussi.

Au premier chef, elle doit déterminer les finalités de sa gestion des traces. Dans notre contexte, la gestion des traces permettra d'assurer la continuité de leurs exécutions en termes de migration.

2.3.3 Différents types de traces

En général, on distingue les types de trace suivant :[10]

a) Traces applicatives

Désignent les obligations légales, réglementaires, professionnelles vis à vis du SI de l'entreprise par ses usagers. Elles n'existent que par la volonté des équipes de projet, qui sont supposées accomplir tous les actes notamment administratifs nécessaires, à l'occasion de la mise en oeuvre d'une application.

Dans le contexte de notre étude, ce type de trace n'est pas pris en compte.

b) Traces techniques

Représentent toutes les autres traces, notamment les traces d'exécution qui désignent la séquence des opérations effectuées par un utilissateur du SI.

C'est à ce type de trace qui nous interesse dans notre travail. Pour la plupart d'entre elles, il s'agit d'événements enregistrés, notamment par :

- les systèmes d'exploitation (ex : Windows, Unix, IBM/MVS),
- les sous-systèmes techniques : moniteurs de transactions (ex : Tuxedo, IBM/CICS), systèmes de gestion de bases données (ex : Oracle, Informix, IBM/IMS), composants réseaux (ex : routeurs, switchs, frontaux de communication IBM, équipements de partage de charge),
- les infrastructures des opérateurs de réseaux privés, des fournisseurs de services Internet.

Dans notre cas les traces d'exécution sont enregistrées par l'infrastructure de service Web.

Les caractéristiques et la granularité des évènements enregistrés sont paramétrables en fonction des besoins. Ces équipements et logiciels disposent d'un paramétrage par défaut, fourni par le fabricant, qui très souvent s'avère ne pas correspond aux besoins exacts de l'entreprise, et peut lui faire courir des risques, dès la mise en service du produit, à savoir :

- des risques organisationnels, par saturation des ressources techniques,
- des risques juridiques, pour non-déclaration de traitements nominatifs,
- des risques de sécurité par diffusion non souhaitée de certaines informations.

C'est pourquoi il est impératif de définir une politique de traces associée à tout nouvel équipement intégré dans le système d'information de l'entreprise, afin de l'adapter à ses besoins et aux risques à couvrir.

Traces d'exécution

Notre étude est impossible sans les traces d'exécutions, il s'agit des séquences d'opérations exécutées par chaque client sur un processus. C'est à travers leurs traces qu'on peut décider de la continuité des client dans leurs exécutions d'un processus donné après une évolution de ce dernier. Ici nous présentons les différents attributs d'une telle trace. Elle est caractérisée par :

- **un identifiant de trace** : C'est un entier permettant d'identifier chaque trace des autres.
- **l'heure et la date du debut d'exécution :** Ces attributs nous permettent de savoir à quelle date, à la seconde près, l'exécution à commencé.
- **la séquence d'opérations** : Il s'agit de l'ensemble des activité qu'a exécuté un client depuis qu'il a commencé sa session. Dans notre contexte, une opération avec un entier i entre parenthèses signifie que l'opération a été exécutée i fois.
- **le protocole exécuté** : Il s'agit de l'identifiant du processus que le client exécute.

trace_ID	opSequence	etatCourant	datDebut	heureDebut	etatTrace	protocol_ID
308	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur	NiveauChoisi	2013-04-17	22:36:00	0	E-learning
309	Loggin.ChoisiSupport	SuportChoisi	2013-04-17	22:36:00	0	E-learning
310	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur	NiveauChoisi	2013-04-17	22:36:00	0	E-learning
311	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur	NiveauChoisi	2013-04-17	22:36:00	0	E-learning
341	Connexion.Commander.Commander(2)	EntrainDeCommander	2013-05-21	11:35:48	0	E-commerce
342	Connexion.Commander.Commander(3).Confirmation	CommandeConfirme	2013-05-21	11:36:15	0	E-commerce
345	Connexion.Commander.Commander.Confirmation	CommandeConfirme	2013-05-21	11:39:45	0	E-commerce
346	Connexion.Commander.Commander(3).Confirmation.Annu...	Fin	2013-05-21	11:40:30	1	E-commerce
347	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur	NiveauChoisi	2013-05-21	11:41:31	0	E-learning
353	Connexion.Commander.Confirmation	CommandeConfirme	2013-05-21	11:46:49	0	E-commerce
352	Inscription.Confirmation	FinInscription	2013-05-21	11:46:49	1	E-commerce
351	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur.Apprent...	ApprentissageTerine	2013-05-21	11:45:15	0	E-learning
350	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur.Apprent...	Fin	2013-05-21	11:43:36	1	E-learning
349	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur.Apprent...	NiveauChoisi	2013-05-21	11:42:04	0	E-learning
348	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur.Apprent...	ApprentissageTerine	2013-05-21	11:41:46	0	E-learning
344	Inscription.Confirmation	FinInscription	2013-05-21	11:38:06	1	E-commerce
331	ApplicationFilled.CheckingExistance.AgeControl.Car...	CareerReconstructed	2013-05-21	11:13:33	0	pension
332	ApplicationFilled.CheckingExistance.AgeControl	Checked	2013-05-21	11:14:06	0	pension
333	ApplicationFilled.CheckingExistance.AgeControl.Car...	CareerReconstructed	2013-05-21	11:14:20	0	pension
361	ApplicationFilled.CheckingExistance.AgeControl.Car...	Payment	2013-05-21	11:51:13	1	pension
335	ApplicationFilled.CheckingExistance	Submitted	2013-05-21	11:15:51	0	pension
336	ApplicationFilled.CheckingExistance.AgeControl	Checked	2013-05-21	11:15:51	0	pension
360	ApplicationFilled.CheckingExistance.AgeControl.Car...	Rejected	2013-05-21	11:49:53	1	pension
359	ApplicationFilled.CheckingExistance.AgeControl.Car...	CareerReconstructed	2013-05-21	11:49:10	0	pension
358	ApplicationFilled.CheckingExistance	Submitted	2013-05-21	11:48:58	0	pension

FIG. 2.9 – Base des traces d'exécution (tirée de l'application).

- **l'état atteint par la trace** : A tout moment, le client atteint un état donné suite à l'exécution d'une opération du processus. Cet attribut garde cet état. Il nous permet de savoir si la trace est active (*désigné par 0, c'est à dire que l'état atteint n'est pas un état final, comme valeur de l'attribut "etatTrace"*) ou non (*désigné par 1 comme valeur de l'attribut etatTrace*).

Toutes ces informations relatives à une trace sont structurées et enregistrées dans une base de données (voir fig 2.9).

2.3.4 Finalités des traces

La notion de finalité est essentielle. Il s'agit de l'usage que l'entreprise compte faire de ses traces. Or, dans le cas de traces contenant des informations à caractère personnel, la loi fait obligation à l'entreprise de déclarer toutes ces traces ainsi que les finalités. A titre d'exemples, voici quelques finalités de l'usage des traces dans une entreprise :[10]

- surveiller le bon fonctionnement des composants du système d'informations,
- élucider des incidents : le plantage, le non respect des règles...etc.

- surveiller le respect des règles, le comportement des usagers, les attentes, le temps.
- détecter des situations anormales : les blocages, les actions non autorisées.
- rassembler des éléments à valeur probantes : Bouclage, tentatives...etc.
- imputer des actions frauduleuses :assurer la non répudiation et la responsabilité.
- contrôler les coûts, faire des statistaques, etc.
- assurer la migration des traces actives lors de l'évolution des applications (*notre contexte*).

Les traces sont donc utiles à toutes sortes de services de l'entreprise, notamment : les exploitants, les informaticiens, les services utilisateurs, les responsables d'unité, les services d'inspection ou d'enquête internes, etc. Elles sont utilisées même par les partisans de la loi à savoir la police et la justice.

En somme les finalités des traces peuvent être regroupées en trois grandes rubriques :

- la recherche de la **qualité de service** dont s'occupent principalement les exploitants informatiques et les responsables d'unités,
- la **protection du système d'information** de l'entreprise, dont s'occupent principalement les responsables de sécurité,
- le domaine des **enquêtes internes, administratives ou judiciaires** menées par l'inspection interne de l'entreprise, les services de Police ou de Justice.

Conclusion

Dans ce chapitre, nous avons exposé les PAIS, la nouvelle tendance des systèmes d'informations dans l'entreprise ; les processus métiers qui representent le comportement visible externe des services Web et aussi des traces qui désignent l'historique d'exécutions du service. Nous avons vu aussi que dans l'environnement versatile dans lequel évolue l'entreprise, elle est consatamment assujettie à des changements qui doivent à tout moment être refletés dans le comportement de son SI. Par conséquent, les protocoles métiers associés aux services du SI évoluent aussi, ceci suicite un problème de gestion des clients en cours d'exécutions du service. Le chapitre suivant de ce mémoire porte sur cette évolution des processus métiers et la gestion de ses conséquences sur les traces actives.

Chapitre 3

Processus métiers : Evolution et gestion des changements

Introduction

Toute organisation (entreprise, administration, etc) est un système ouvert qui est influencé par l'environnement dans lequel elle évolue. Les lois du marché font que cet environnement subi constamment des changements de différentes natures ; pour y survivre, l'organisation doit s'adapter à ces changements. L'une des plus importantes facteurs de performence d'une entreprise est l'agilité de son SI, c'est à dire, la capacité de ce dernier à réfleter les réalités de l'entreprise à tout moment. Ainsi, pour des raisons d'adaptation, le SI de l'entreprise subi obligatoirement des changements suite à ceux de son environnement d'evolution.[3]

Dans cet environnement dynamique, ces changements sont effectués sur ses services en les faisant évoluer. Ce qui ce traduit par l'évolution des protocoles métiers qui leurs sont associés pour donner une nouvelle version de ces processus. Les motifs de ces changements sont variés : [3]

- Nouvelles applications : par extension de celles déjà existantes.
- Nouvelles stratégies commerciales : changement des procédures, améliorations.
- La concurrence : allégement et raffinement des procédures existants.
- Stratégies commerciales changeantes : prise en compte de nouveaux paramètres de l'environnement.
- Nouvelles lois (Changement de règlement) : intégration de nouvelles règles de gestion.
- Fixer les problèmes trouvés dans la définition de protocole : maintenance des processus métier déjà existants.

- Redressement/correction d'anciennes versions : suite à des améliorations, fixer les anomalies.
- Collaboration : par la fusion et /ou coordination avec d'autres partenaires.
- D'une manière générale, l'organisation du travail, l'urbanisation du S.I, le déplacement des structures de décision, les nouvelles et complexes architectures technologiques, la pression des directions financières, l'administration de la sécurité, la relation client, l'obligation de visibilité sur chaque activité, etc. peuvent induire des remises en ordre des processus métier par leur changement.

Ces évolutions ont un impact direct sur les instances actives du service. Dans la suite, nous détaillerons les concepts y afférents.

3.1 Evolution des processus métiers :

Dans le contexte de l'évolution des protocoles, on peut distinguer deux aspects :[3]

3.1.1 L'évolution statique d'un protocole :

Se rapporte au problème de modifier la définition du protocole, sans qu'il soit en ce moment des instances en cours d'exécution. Pour cela, il est nécessaire de fournir un ensemble d'opérations de changement qui permettent la modification progressive d'un protocole existant sans besoin de le redéfinir à partir de zéro.Cet aspect aura des conséquences sur les futures instances seulement. Comme, il n'y a pas d'instances actives et du fait que touts les instances ont déjà terminé leur exécutions, il n'y aura pas d'impact majeur lié à l'évolution du protocole.

3.1.2 L'évolution dynamique d'un protocole :

Se rapporte au changement de la définition du protocole alors qu'il y a des instances en cours d'exécution au moment du changement et qui ont déclenché leur exécution conformément à l'ancienne version du protocole. Ces instances sont dites **actives**. Le besoin de fournir des mécanismes pour qu'un concepteur de protocole puisse manipuler ces instances en cours, pour répondre aux nouvelles exigences induites par le changement, est alors de rigueur.

L'un des problèmes les plus difficiles dans la modification d'un protocole est la gestion de ses instances actives. La resolution de ce problème est motivée par plusieurs raisons :

- Certains protocoles qui décrivent les services pourraient être de longue durée de plusieurs jours à une année, comme un service de demande de citoyenneté, un service de réclamation d'assurance et ainsi de suite. Si toutes les transactions en cours sont

abandonnées suite à certains changements, tous les clients actuels auraient à perdre des quantités considérables de travail voire d'argent. On ne peut non plus faire passer (on dira migrer dans la suite) tous ces clients vers la nouvelle version du processus, ceci induira des incohérences et des erreurs d'éxécutions, ce qui peut s'averer coûteux pour l'organisation. **Alors que faire ?**

Le besoin de minimiser la perturbation de clients actuels en veillant à ce que le nouveau protocole soit appliqué est nécessaire. Il n'est pas suffisant de gérer manuellement les instances en cours, étant donné que les protocoles peuvent être très complexes et qu'il pourrait y avoir un nombre énorme d'instances actives.

- Certains protocoles supportent des services qui sont de nature critique, ne peuvent être arrêté, comme dans les industries pharmaceutique, chimique et industrielles. Dans ce cas, il est impossible d'arrêter les services afin de répondre aux exigences de certaines modifications.

Pour les raisons ci-dessus, simplement abandonner ou annuler les instances actives n'est pas adéquat. Pour cela, des approches plus pragmatiques doivent être prises en compte sur la façon de traiter le problème d'évolution des protocoles. **Notre travail vise à résoudre ce problème.On ne s'intéressera pas aux raisons du changement mais uniquement à ses conséquences sur les instances actives.**

Nous visons à aborder le problème, en spécifiant les états de migration les plus pertinents (adéquats).

3.2 Les opérations d'évolution :

Elles désignent les différentes actions qu'on peut effectuer sur le protocole afin de l'adapter aux changements de spécification du service associé. Nous avons récensé les opérations suivantes :

3.2.1 Ajout de message :

Cette opération ajoute une transition étiquetée avec un message entre un état source et état cible.Voir fig 3.1

3.2.2 Suppression d'un message :

Cette opération supprime une transition étiquetée avec un message entre un état source et état cible. Voir fig 3.2

3.2.3 Ajout d'un état :

Cette opération ajoute un nouvel état étiqueté avec un message..voir fig 3.3

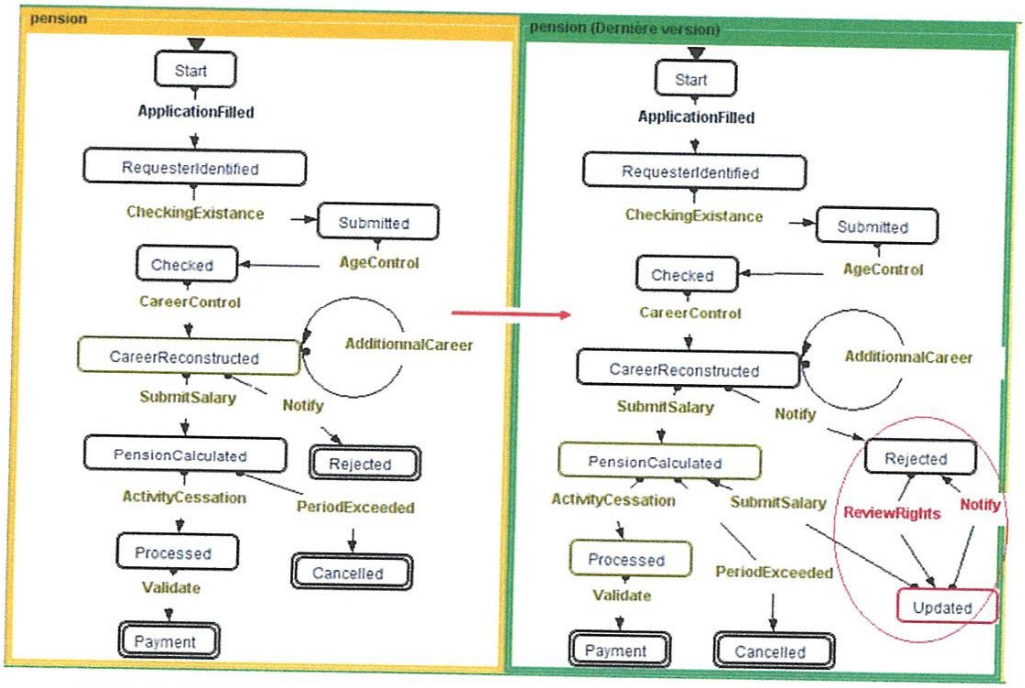

FIG. 3.1 – Exemple d'évolution par ajout de message .

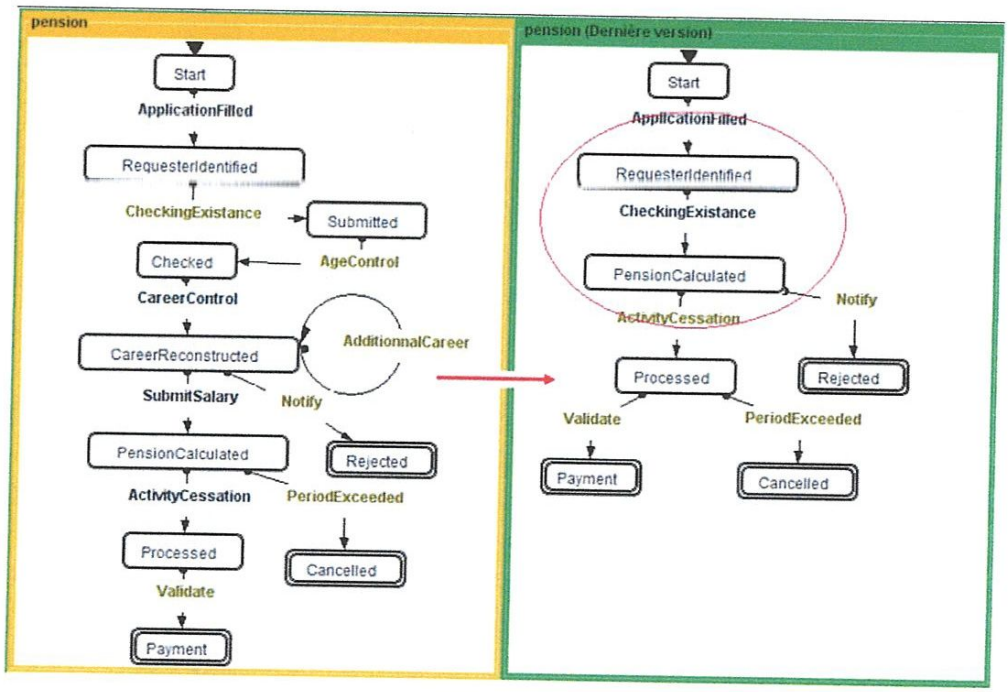

FIG. 3.2 – Exemple d'évolution par suppression de message.

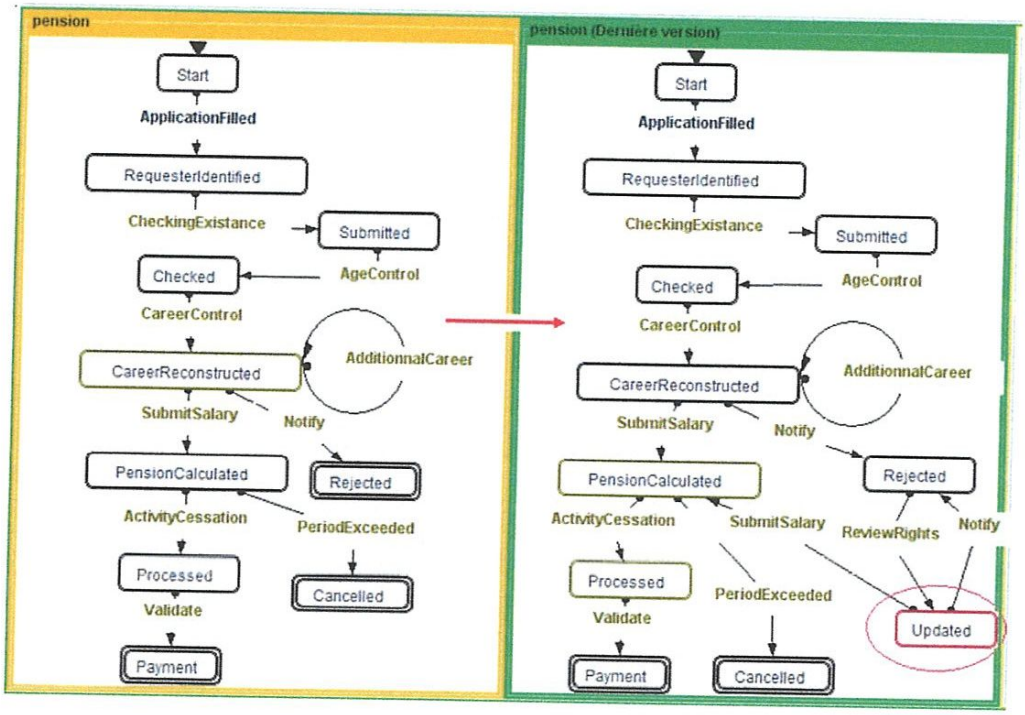

FIG. 3.3 – Exemple d'évolution par ajout d'état.

3.2.4 Suppression d'un état :

Cette opération enlève un état du protocole. Après que l'état soit enlevé, toutes les transactions sortantes et entrantes de cet état sont enlevées. Voir fig 3.4

3.2.5 Ajout de boucle sur un Etat/sous protocole :

Cette opération ajoute une transition étiquetée avec un message entre un état qui est la source et la cible.Voir Fig 3.5

3.2.6 Suppression de boucle sur un Etat/sous protocole :

Cette opération supprime une transition et son message entre l'état qui est la source et la cible de cette transition. Voir Fig 3.6

3.2.7 Ajouter un sous chemin

Consiste en une sequence d'ajout d'etats et de messages en alternative pour aboutir soit un nouvel état final, soit à celui existant tout en préservant la cohérence du protocole (C'est à dire pas de sous chemin, ni d'état isolé). Voir Fig 3.7

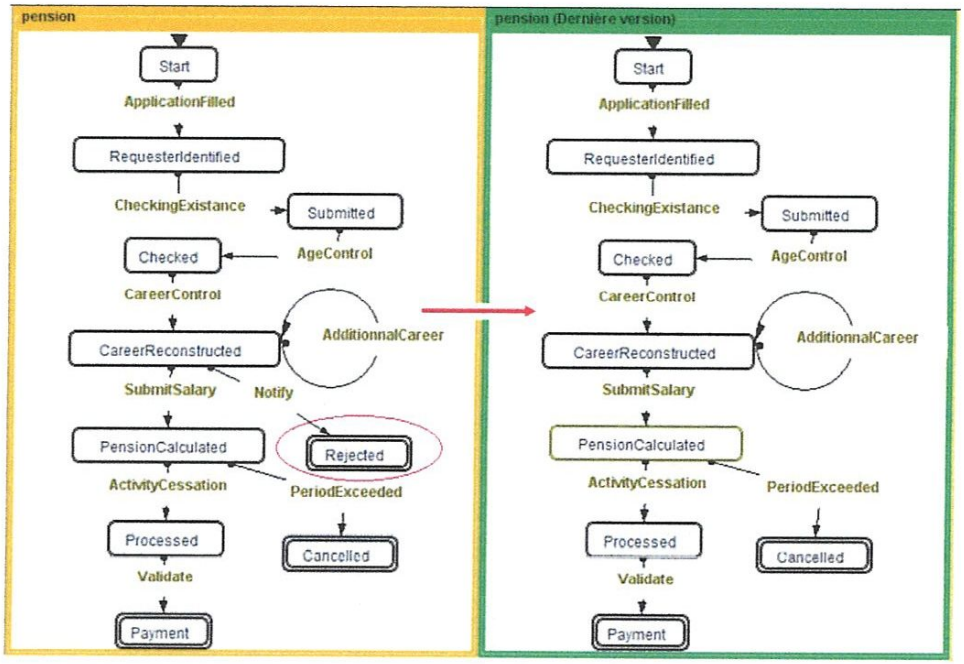

FIG. 3.4 – Exemple d'évolution par suppression d'état.

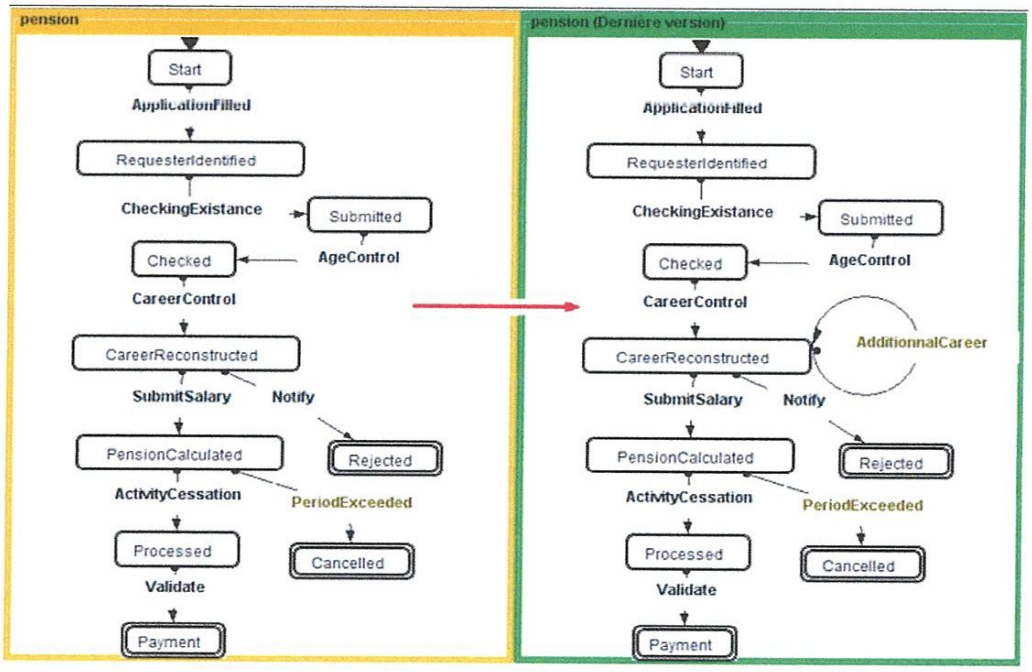

FIG. 3.5 – Exemple d'évolution par ajout d'une boucle.

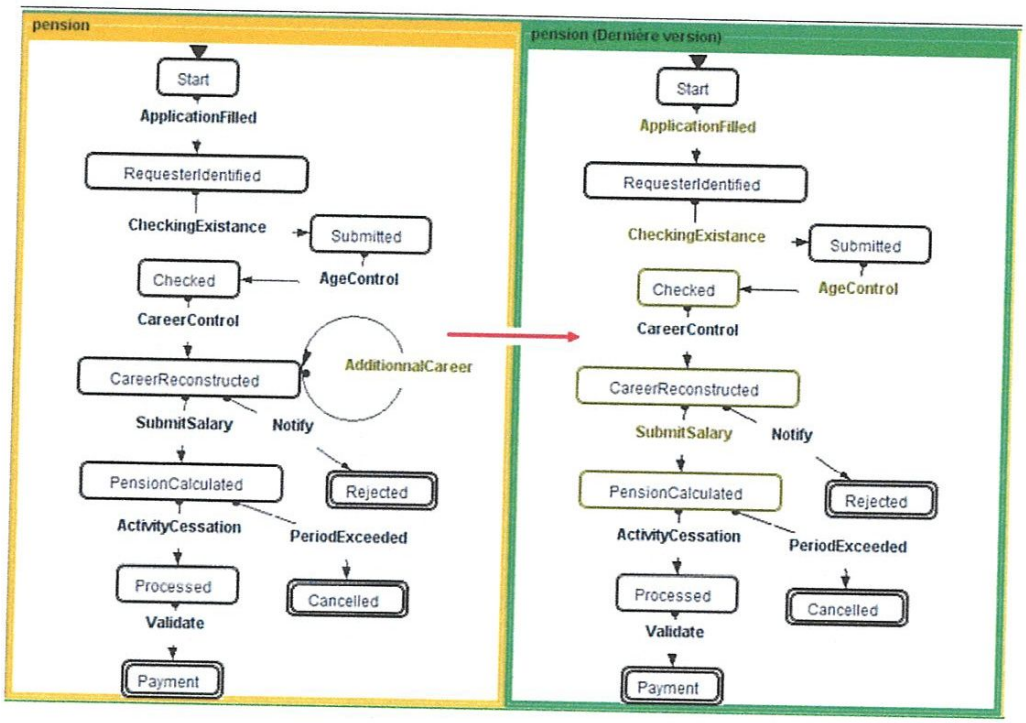

FIG. 3.6 – Exemple d'évolution par Suppression d'une boucle.

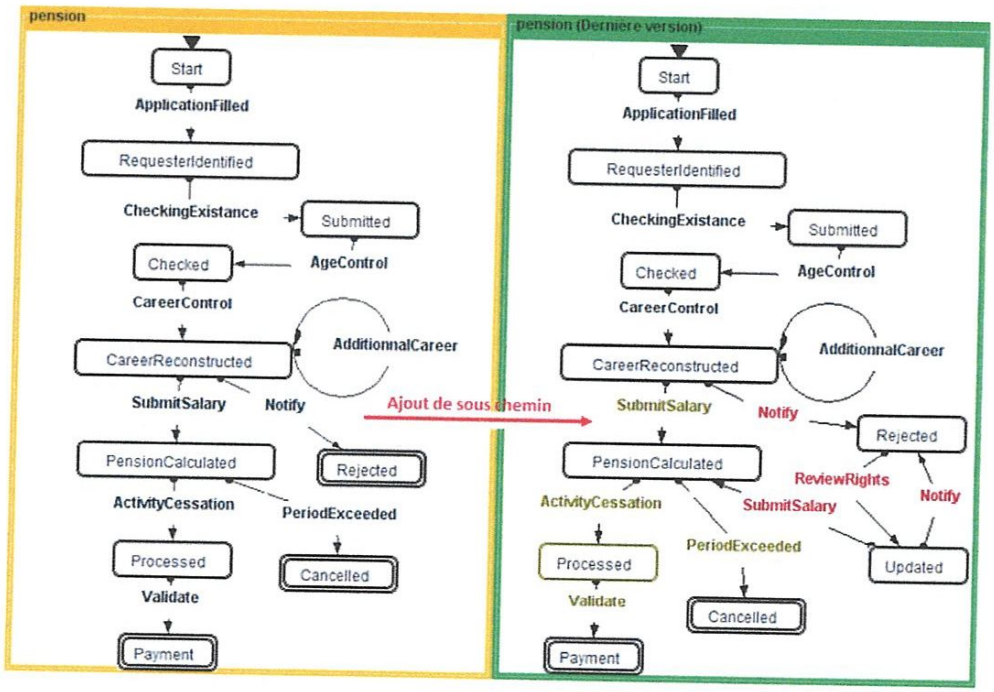

FIG. 3.7 – Exemple d'évolution par ajout un sous chemin.

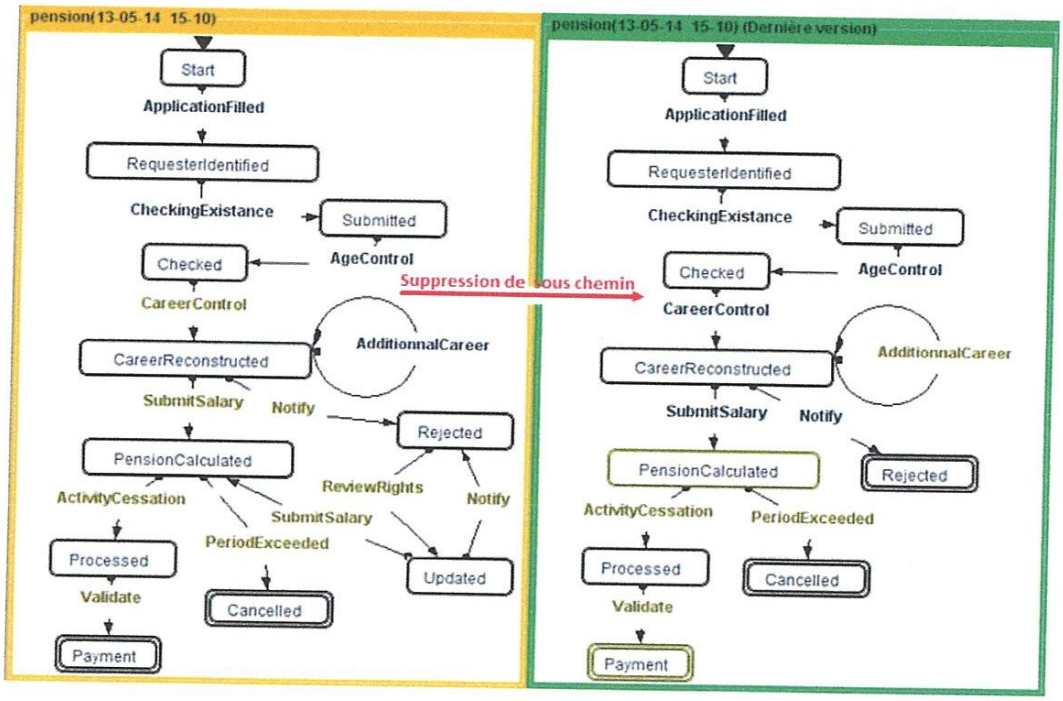

FIG. 3.8 – Exemple d'évolution par suppression un sous chemin.

3.2.8 Supprimer un sous chemin

Il s'agit d'éliminer un sous chemin.Voir Fig 3.8

3.2.9 Modifier un état/Message

Cette opération consiste à modifier le nom d'un état ou d'un message suite à un réordonnancement, une substitution, Voir Fig 3.9

3.3 Conséquences de l'évolution :

Suite aux évolutions des processus métiers, le besoin pour toute organisation de gérer efficacement l'impact de ces évolutions est une nécessité afin de satisfaire les contraintes et les possibilités proposées par les changements. L'exécution des services par les clients sont tracées, ceci pour diverses raisons (*voir le chapitre sur les traces*). Ces traces d'exécutions désignent les instances d'exécution du service, elles peuvent être actives ou terminées.

Toute évolution d'un protocole de services impactera surement ses instances actives. Certaines d'entre elles pourront continuer leur exécution sans problèmes tandisque que d'autres se trouveront devant une situation ambigüe pour continuer leur exécution ; en ce sens, il est

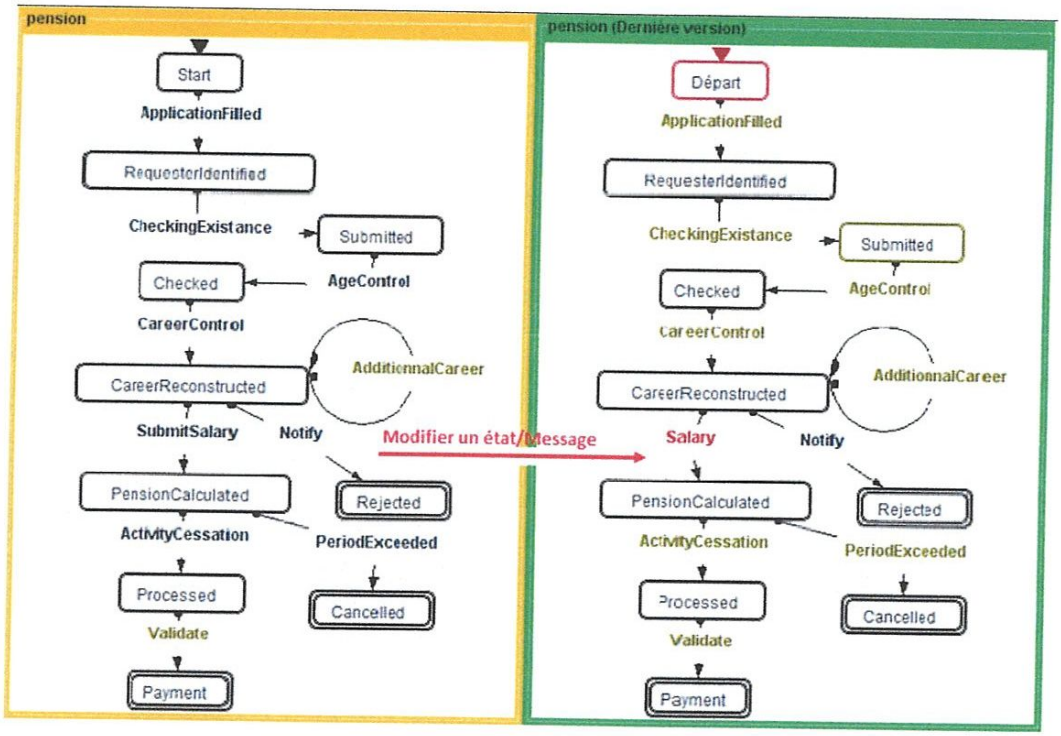

FIG. 3.9 – Exemple d'évolution par modification d'un état/Message.

impératif d'analyser l'impact de l'évolution sur ces instances en cours. Le principe consiste à faire une classification des instances actives en instances migrables et instances non migrables. Nous présentons ci-desous la problématique ainsi que les raisons qui motivent son étude.

3.4 Problématique et motivations :

Le sujet de notre travail est l'analyse de l'impact de l'évolution des protocoles des service Web sur les instances actives. Autrement dit, soit un service Web dont le comportement externe est décrit par un protocole. Ce service peut être exécuté par plusieurs clients. Pendant que certains de ces exécutions sont en cours ce protocole peut subir des évolutions, ce qui donne une nouvelle version du protocole. La problématique est : **conformément à la nouvelle version du protocole les clients en cours d'exécutions peuvent-ils continuer leur exécution ?** (Voir Fig 3.10, l'exemple d'un processus metier d'e-commerce)

Au passage, il faut savoir que la solution d'initialiser toutes les instances n'est pas appropriée car elle peut susciter des pertes de temps et/ou de ressources.

L'analyse du problème est motivée par plusieurs raisons à savoir :

a- Aider les gestionnaires de protocoles et développeurs de service à assurer la maintenance de leurs services.

b- Il est inadéquat de demander à des clients de recommencer l'exécution de leur processus à partir de zéro, suite à un changement du protocole du fournisseur car cela engendre une perte du travail déjà effectué.

c- Certains services Web sont de longue durée (plusieurs heures, voire plusieurs jours, mois ou années (demande de citoyenneté par exemple)) donc c'et des applications consommatrices de ressources. Dans ce cas la reprise d'un service (qui a évolué) sera catastrophique pour ses clients.

d- Dans les services Web grand public (applications e-Commerce, e-gouvernement, bibliothèque électronique.....etc.) le nombre d'instances actives est très élevé. Alors, la gestion manuelle de la migration vers la nouvelle version d'un protocole est très lourde, voire impossible vue l'urgence des décisions à prendre.En ce sens, la prise en charge avec un outil automatique est primordiale.

e- Pour les clients, et même en cas d'évolutions des protocoles fournisseurs, il est intéressant de spécifier, les nouveaux services équivalents afin d'assurer la continuité d'exécution en cas de panne.

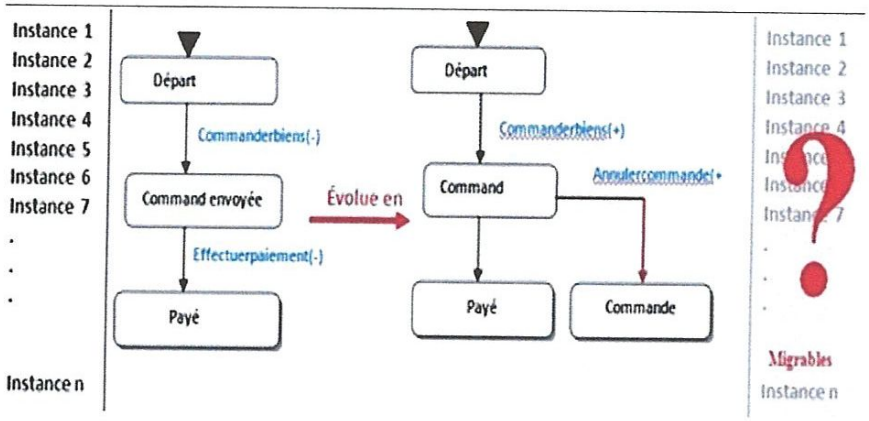

FIG. 3.10 – Problématique de l'évolution des protocoles

f- Pour les processus critiques (services de type : industriels, médicaux, centrales nucléaires, bourses, contrôles aériens) l'arrêt brutal du service ne peut être opéré car il est catastrophiques pour les clients actifs. Il faut trouver une migration transparente.

Les instances actives ont commencé leurs conversations conformément à l'ancienne version, après évolution peuvent-ils continuer leur interaction sans erreurs ni incompatibilités ?

Ces diférentes motivations nous montrent la nécessité d'une analyse d'impact de cette évolution du protocole sur les instances actives du service Web qui lui est associé.

3.5 Travaux connexes :

L'évolution de protocole est liée à quatre autres problèmes d'évolution : l'évolution de schéma de base de données, l'évolution de composant logiciel, l'évolution des Workflows et le service Web versionning. Cette section aborde une analyse comparative de ces différents problèmes d'évolution par rapport au problème d'évolution des protocoles métiers.

3.5.1 Évolution de schéma de base de données :

La communauté de base de données a considéré le problème de contrôler l'évolution de schéma, principalement dans le domaine des bases de données orientées objet. [3]

Pour répondre aux nouvelles exigences des applications de base de données, la définition de schéma est modifiée avec le temps, en ajoutant ou en enlevant des éléments de schéma. Le travail dans ce secteur a développé plusieurs techniques qui soutiennent la cartographie des éléments de schéma du vieux au nouveau schéma. De telles approches incluent la conversion, qui transforme les données de la base de données pour les rendre conformes au schéma

modifié, et la nouvelle version de la classe, qui permet aux vieilles applications commencées de continuer à employer le vieux schéma. Cependant, l'évolution de protocole métier diffère de l'évolution de schéma de base de données de deux manières significatives :

D'abord, la base de données ne peut pas être accédée et les transactions en cours sont bloquées pendant la réorganisation de base de données, alors que l'évolution du protocole doit transférer les instances en cours pour le nouveau protocole sans arrêter le système.

En second lieu, dans le cas des versions de classe, il est acceptable pour les anciennes applications à fonctionner selon l'ancien schéma, alors il pourrait y avoir des situations où il n'est pas possible pour les instances en cours pour continuer à fonctionner selon l'ancien protocole, par exemple, dans le cas où il y a des trous de sécurité dans la définition du protocole de sécurité. Par conséquent, nous ne pouvons pas utiliser les techniques d'évolution de schémas de bases de données dans le contexte de l'évolution des protocoles de services Web.

3.5.2 Évolution des composants logiciel :

L'évolution des composants logiciel a été considérée importante pour obtenir les avantages des développements logiciels à base de composants, tels que la réutilisation de composants, la maintenance facile, et la plus grande flexibilité. Les composants peuvent évolués puisque les logiques métiers aussi changent. L'évolution est obligatoire pour satisfaire les nouvelles exigences de l'application, allant des changements de structure de logiciels à des problèmes et des bugs à fixer. La plupart des solutions à ce problème sont basées sur les mécanismes de « versioning ». [3]

En ce sens, et afin de donner les informations spécifiques de version, les techniques de versioning s'appuient sur l'amélioration des noms de fichier des bibliothèques par les numéros de version ou les bibliothèques par métafichiers spéciaux (i.e., les fichiers manifestes d'un format XML). Ainsi, de tels mécanismes permettent à des versions multiples d'un composant d'exister dans un système et de permettre à des applications d'employer différentes versions d'un composant. Cependant, ces approches ne s'appliquent pas à notre problème, parce que l'objectif de notre travail est différent de ce scénario, du fait que l'ancienne version d'un protocole de service est considérée comme obsolète dès qu'une nouvelle version apparait.Par ailleurs, nous voulons assurer la continuité des instances actives en se basant seulement sur la dernière version de processus métier(BP).

3.5.3 Évolution des Workflows :

On appelle workflow la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier (aussi appelé processus opérationnel ou bien procédure d'entreprise).

L'évolution du protocole a quelques similitudes avec l'évolution de Workflows. [3]

Pour aborder la question des changements de flux de travail dynamiques dans leur travail, les workflows ont exploités une abstraction Petri net pour la modélisation des changements dynamiques qui signifie le changement «à la volée », tandis que des instances de workflow sont en cours d'exécution. Leur approche est basée sur « les région de changement » qui contiennent les parties du réseau de Petri directement touchées par le changement. Toutefois, leur approche ne fournit pas une méthode de calcul de la région de changement, à savoir, la région doit être déterminée manuellement.[3]

Présentent un ensemble d'opérations de base qui permettent la modification d'un schéma de Workflow et préservent l'exactitude structurale et comportementale quand ils sont appliqués à la modification de Workflow. Ils proposent également des politiques d'évolution applicables aux instances en cours de Workflow. Cependant, le groupement de l'instance faite selon les politiques est laissé au concepteur de Workflow. Comparé à leur travail, le groupement et la classification des instances peuvent être faits automatiquement avec les méthodes d'analyse statique et dynamique.

Pour gérer les instances actives de workflow cette approche, propose des politiques de modification qui peuvent être adoptées par le concepteur de workflow. Et puis, il lance un processus de modification de trois phases consistant : en définissant, en se conformant et en adoptant la modification. Dans la phase de définition, la modification du schéma de flux de travail est effectuée. Au cours de la deuxième phase, les instances actives sont regroupées suivant la conformité avec le nouveau schéma. En conséquence, les instances conformes et instances non conformes sont déterminées. Pour les instances non conformes, il est proposé le graphe de conformité qui agit comme un pont entre les anciens et les nouveaux schémas. La troisième phase du processus de modification est d'adopter la modification et la migration des instances.

Cependant, il n'est pas détaillé comment déterminer la conformité des instances avec le nouveau schéma. En outre, par rapport à leurs deux types de méthodes de regroupement, notre cadre permet à un concepteur de protocoles d'effectuer plus de partitionnement à grain fin et choisir les plus diverses stratégies de migration.

3.5.4 Service Web versioning :

Dans le cadre des services Web, quelques travaux récents ont proposé les techniques versioning pour traiter le problème de l'évolution de service de Web.Proposé une approche basée sur l'utilisation du namespace de version et l'utilisation du numéro de version dans l'entrée d'UDDI. L'approche permet des versions multiples d'un service Web par le client. Ces versions de soutien dépendent des versions antérieures du service. Cette approche présente une technique de conception appelée « Chain of Adapters» pour traiter le problème de contrôler

la version de service Web et pour réaliser la compatibilité ascendante avec des clients ayant entamé leur travaux avec des versions plus anciennes du service Web concerné.

Conclusion

Après avoir exposé le concept d'évolution des protocoles métiers et cerné ses conséquences vis à vis des instances actives du service, nous nous focaliserons dans le chapitre suivant sur une notion inhérente à la gestion de ces conséquences, en analysant l'impact de cette évolution sur les instances actives exprimées par leurs traces d'exécution.

Deuxième partie

Conception et réalisation

Chapitre 4

Identification et formalisation des patterns d'évolution

Introduction

Dans ce chapitre, on va s'intéresser à la problématique de la migration des traces actives. Pour atteindre cet objectif, nous proposerons une modélisation formelle de cette notion de migration ainsi qu'une identification de quelques patterns associés aux évolutions des protocoles. Chaque cas sera décrit formellement et illustré d'un exemple.

En effet, notre étude des BPs relatifs à des domaines de gestion spécifiques, nous a permis de cerner quelques scénarios de changement qui permettent de préserver la compatibilité entre ancienne et nouvelle version des protocoles. Les domaines d'analyse aux quels nous nous sommes intéressés sont la gestion de le retraite des assurés sociaux, le domaine du commerce électronique et la gestion de la scolarité.

4.1 Notion de pattern en Génie logiciel

En génie logiciel, un patron de conception (*design pattern en anglais*) est un concept destiné à résoudre les problèmes récurrents suivant le paradigme objet. Ils décrivent des solutions standards pour répondre à des problèmes d'architecture et de conception des logiciels. Ce n'est pas un algorithme, mais une structure générique qui permet de résoudre le problème identifié[13]. Il est indépendant du langage de programmation, il est "standardisé", suffisamment pour que tous puissent s'y référer. C'est l'expérience qui montre que la solution est valide. Ce qui nous ramène à la définition suivante : Les design pattern ou motifs de conception sont des recueils de bonnes pratiques de conception pour un certain nombre de problèmes récurrents. Ces problèmes sont généralement d'ordre programmatiques entre autres garder une forte cohésion entre les composants d'une même classe, un couplage faible entre des classes distinctes [14]. Les design pattern permettent d'assurer la pérénité dans le code dans le temps.[15]

Dans notre contexte les patterns d'évolution, analogiquement aux design pattern du génie logiciel, sont des notions abstraites que doit suivre toute évolution d'un protocole donné afin de garder une compatibilité et une cohérence entre les deux versions du dit protocole. Ce sont des motifs d'identification de compatibilité entre traces lors de la migration de ces dernières. Leur identification et formalisation est donc necessaire. Ils sont caracterisés par :

- **Nom :** Désignation du pattern enfin de l'identifier des autres. Pour des exemples, voir en bas la section intitulé "Identification des patterns d'evolution des protocoles".
- **Description** : Elle consiste à donner le principe de fonctionnement du pattern. En mettant l'accent sur les éléments préservés et ceux qui ont changé.
- **Paramètres** : Désignent les différents éléments de l'automate, en occurrence les messages, qui entrent en jeu dans le pattern. Ils sont regroupés en deux ensembles, l'un pour l'ancienne version du protocole et l'autre pour la nouvelle. Ces deux ensembles sont faits de telle sorte qu'ils peuvent toutes les deux être vides dans certains patterns ou le contraire dans d'autres, le cas intermédiaire se produit également dans certains situations. Il est à signaler qu'une évolution de protocole peut satisfaire plusieurs patterns.

Avant d'aborder l'indentification des patterns, nous commencons par spécifier formellement la notion de migration de trace.

4.2 Migration des traces : définition du modèle

Soient deux versions P et P' d'un protocole donné tel que P' soit une évolution de P suivant un pattern quelconque, la migration d'une trace active dans P, consiste à determiner la trace correspondante ainsi que l'état qu'elle peut atteindre dans le nouveau protocole P'. Cette trace et l'état qu'elle atteint dans $P\prime$ sont appélés les attributs de la migration. C'est ainsi qu'on determinera si la trace donnée dans P peut oui ou non continuer son exécution après une évolution du protocole. Une trace est dite migrable si elle peut continuer son exécution dans $P\prime$ et dans le cas contraire, on dit qu'elle est non migrable. La modélisation du problème se présente comme ci-dessous.

Problème : Input : $\{ P, P', \delta, pattern, q_s \}$

Output : $\{ q_t, \beta \}$

Avec :

P = Ancienne version du protocole, P' =Nouvelle version du protocole, δ =Trace dans P

$pattern$ =Pattern d'évolution de P vers $P\prime$, q_s =L'état atteint par δ ou encore l'état source,

q_t =L'état atteint par β ou encore état cible, β =Trace correspondante à δ dans P' ou trace témoin

Nous formalisons cette migration par :

$$(q_s, \delta) \xrightarrow[\text{paramètres1, paramètres2}]{\text{nom du pattern}} (q_t, \beta)$$

4.3 Identification des patterns d'evolution des protocoles

Les patterns d'évolution sont inhérents à notre travail, ils présentent un cadre de compatibilité entre deux protocoles différents sans quoi aucune notion de migration n'est envisageable. C'est ainsi que nous avons identifiés quelques uns que nous décrivons ci-dessous.

Nous signalons que cette identification n'est pas exhaustive. L'objectif est de recenser les patterns les plus fréquents et résultant des actions courantes de changement opérées par les gestionnaires de protocoles dans les domaines de gestion de nos centres d'intéret à savoir : la retraite des assurés sociaux, le commerce électronique et la gestion de la scolarité.

Cependant, dans ce travail nous nous interresserons aux patterns unique sans combinaisons. C'est à dire qu'on ne tiendra pas compte de la combinaison de plusieurs patterns dans une évoluion de protocole. Les patterns que nous avons identifié sont les suivants :

4.3.1 Pattern strict

Exemple

Dans cet exemple de pattern strict on a supprimé les opérations (*annuler* et *sortir*) de P, mais on exige que la migration soit selon le pattern strict (c'est le choix du gestionnaire de protocole)(voir Fig 4.1).

Description et utilité

Dans ce pattern, tout élément figurant dans l'ancienne version du protocole doit se trouver dans la nouvelle et vis-versa. Autrement dit les deux versions sont les mêmes. Il n'a pas de paramètres car rien ne change. Les deux ensembles de paramètres sont vides.

Le pattern strict intervient dans le cas suivant :

Panne de fourniseur de service : Si le fournisseur de service que client exécute tombe en panne alors que l'exécution est active, alors le client aura besoin d'un autre service qui a au moins les mêmes fonctions que le précédent. Dans ce cas, nous nous intéressons à exactements les mêmes fonctions.

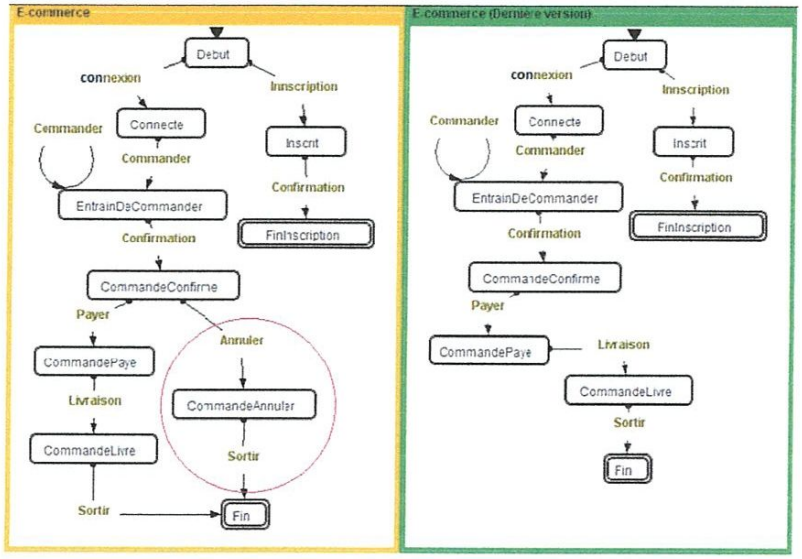

FIG. 4.1 – Exemple d'évolution pour le pattern Strict d'un BP d'e-commerce.

Numéro	Etat atteint	Active	Traces d'exécutions
1	EntrainDeCommander	True	Connexion.Commander.Commander(1)
2	FinInscription	False	Inscription.Confirmation
3	Inscrit	True	Inscription
4	CommandeAnnuler	True	Connexion.Commander.Commander(5). Annuler
5	Fin	False	Connexion.Commander.Commander(4).Annuler. Sortir
6	CommandePaye	True	Connexion.Commander.Commander(6).Payer

TAB. 4.1 – Exemple de migration de traces pour le cas du pattern strict.

Formalisation

$$(q_s, \delta) \xrightarrow{Strict} (q_t, \beta) \quad et \quad \delta = \beta \tag{4.1}$$

Exemple de migration

Pour qu'une trace actice puisse migrer dans la nouvelle version du protocole conformément à ce pattern, il faut impérativement la même séquence d'opérations. Ainsi dans les six traces citées en exemple dans le tableau 4.1, on a :

- les traces 2 et 5 sont inactives,
- les traces 1, 3, 4 et 6 sont de telle sorte que : 1, 3 et 6 peuvent migrer sans difficulté en gardant les mêmes attributs, mais 4 ne peut pas migrer.

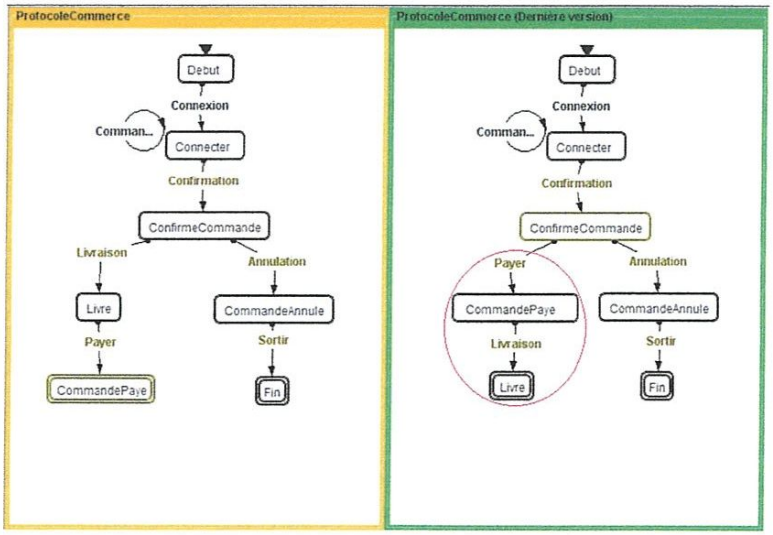

FIG. 4.2 – Exemple d'évolution pour le pattern de réordonnancement.

- *(EntrainDeCommander, Connexion.Commander.Commander(1))* → *(EntrainDeCommander, Connexion.Commander.Commander(1))* ;
- *(Inscrit, Inscription)* → *(Inscrit, Inscription)* ;
- *(CommandePaye, Connexion.Commander.Commander(6).Payer)* → *(CommandePaye, Connexion .Commander.Commander(6).Payer)* ;

4.3.2 Pattern de réordonnancement des opérations

Exemple

Dans cet exemple de pattern de réordonnancement d'opération on a ordonné les opérations (*Livraison*, *Payer*) dans l'ancienne version du protocole en (*Payer*, *Livraison*) dans la nouvelle version (voir Fig 4.2).

Description et utilité

Ce pattern consiste à réordonner, comme son nom l'indique, un ensemble d'opérations de l'ancienne version du protocole pour établir la nouvelle. Ses deux ensembles de paramètres sont α' dans P et α dans P', les opérations à réordonnaner. Ils ne sont pas vides. Le réordonnancement est justifié par des choix de gestion consistant à faire anticiper un contrôle (une action).

Son cadre d'utilité est le besoin de réordonnancement. Par exemple, soit un BP d'e-commerce constitué des opérations suivantes : *connexion, commander, confirmer,* ***livrer, payer***.

Numéro	Etat atteint	Active	Traces d'exécutions
1	Connecter	True	Connexion.Commander(2)
2	Fin	False	Connexion.Commander(6).Confirmation.Annulation. Sortir
3	Livre	True	Connexion.Commander(3).Confirmation.Livraison
4	CommandePaye	False	Connexion.Commander(3).Confirmation.Livraison. Payer
5	ConfirmeCommande	True	Connexion.Commander(3).Confirmation
6	CommandeAnnule	True	Connexion.Commander(6).Confirmation.Annulation

TAB. 4.2 – Exemple de migration de traces pour le cas du pattern de réordonnancement.

Après un moment d'interaction, le fournisseur se rend compte que certains clients se rétractent après la livraison, ce qui cause d'énorme pertes au fournisseur. Pour rémedier à ce problème, il décide de faire avancer une ou plusieurs opérations rapport à d'autres ou le contraire. par exemple *connexion, commander, confirmer,* ***payer, livrer***, serait de rigueur.

Formalisation

$$(q_s, \delta) \xrightarrow[\alpha \,:\, \alpha']{\textit{Réordonnancement}} (q_t, \beta), \ni \delta_0, \delta_1 \textit{ tel que } \delta = \delta_0\alpha\delta_1 et\ \beta = \delta_0\alpha'\delta_1, \alpha' = \otimes\alpha \tag{4.2}$$

avec$\otimes$ ≡Shuffle(c'est-à-dire toute permutation possible)

Exemple de migration

Dans ce pattern, pour avoir la trace correspondante dans la nouvelle version du protocole, on réordonne les opérations qui constituent les paramètres du pattern. Parmi les six traces du tableau 4.2, on a :

- les traces 2 et 4 sont inactives,
- les traces 1, 3, 5 et 6 peuvent migrer comme suite (dans la même ordre) :

- *(Connecter, Connexion.Commander(2))* → *(Connecter, Connexion.Commander(2))*;
- *(Livre, Connexion.Commander(3).Confirmation.Livraison)* → (***CommandePaye***, *Connexion .Commander(3).Confirmation.****Payer***);
- *(ConfirmeCommande, Connexion.Commander(3).Confirmation)* → *(ConfirmeCommande, Connexion.Commander(3).Confirmation)*;
- *(CommandeAnnule, Connexion.Commander(6).Confirmation.Annulation)* → *(CommandeAnnule, Connexion.Commander(6).Confirmation.Annulation)*.

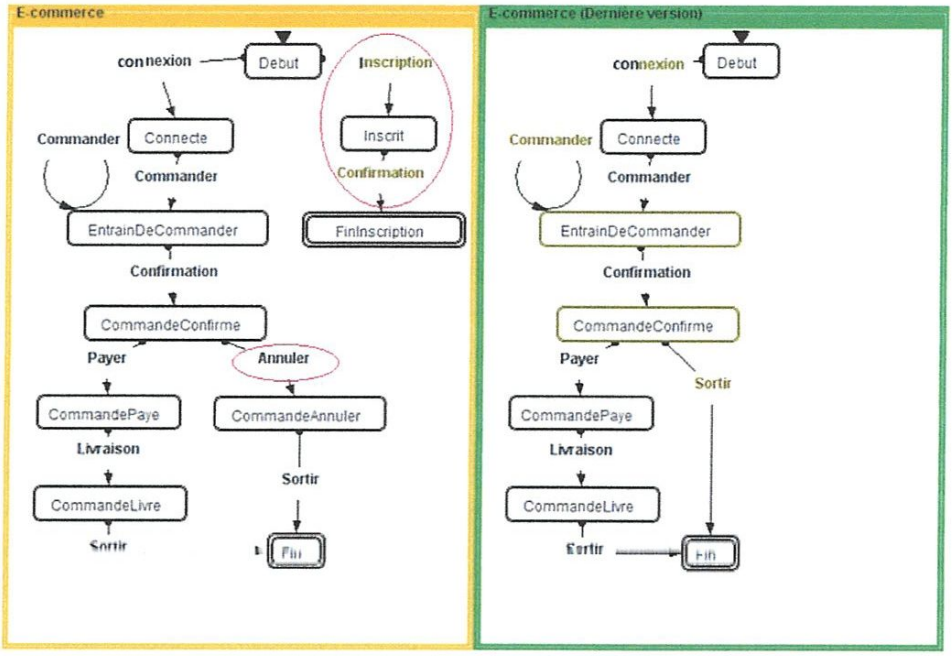

FIG. 4.3 – Exemple d'évolution pour le pattern de réduction d'un BP d'e-commerce

4.3.3 Pattern de la réduction

Exemple

Dans ce pattern de la réduction on a supprimé les opérations (*Annuler*, *Inscriprion* et *Confirmation)* qui suit l'inscription dans l'ancienne version du protocole pour faire la nouvelle version (voir Fig 4.3).

Description et utilité

Ce pattern consiste à supprimer un ensemble d'opérations dans l'ancienne version du protocole pour établir la nouvelle. L'un de ses deux ensembles de paramètres est vide, celui dans P' et celui dans P, α est constitué des opérations supprimées.

On a recours à ce pattern lorqu'on juge inutile une opération dans la séquence, ou encore lorsqu'on veut subdiviser un BP en plusieurs autres BP.

Formalisation

$$(q_s, \delta) \xrightarrow[\alpha]{Réduction} (q_t, \beta) \ni \delta_0, \delta_1, \alpha \; tel \; que \; \delta = \delta_0 \alpha \delta_1 \; et \; \beta = \delta_0 \delta_1 \qquad (4.3)$$

Numéro	Etat atteint	Active	Traces d'exécutions
1	Inscrit	True	Inscription
2	Fin	False	Connexion.Commander.Commander(3). Confirmation.Annuler.Sortir
3	CommandeConfirme	True	Connexion.Commander.Commander(4).Confirmation
4	CommandeAnnuler	True	Connexion.Commander.Commander(5). Confirmation.Annuler
5	Fin	False	Connexion.Commander.Commander(6). Confirmation.Payer.Livraison.Sortir
6	Fin	False	Connexion.Commander.Commander(5). Confirmation.Annuler.Sortir

TAB. 4.3 – Exemple de migration de traces pour le cas du pattern de réductiont.

Exemple de migration

Avec ce pattern, pour obtenir la trace témoin d'une trace active dans P, il faut annuler les opérations supprimées dans ce dernier figurant dans cette trace. Pour les six traces du tableau 4.3 nous avons :

- les traces 2, 5 et 6 qui sont inactives,
- les traces 1, 3, 4 sont actives, concernant leur migrabilité on a dans la même ordre :

- *(Inscrit, Inscription)* ne peut pas migrer car elle n'a pas de trace témoin dans P',
- *(CommandeConfirme, Connexion.Commander.Commander(4).Confirmation)* $\rightarrow$ *(CommandeConfirme, Connexion.Commander.Commander(4).Confirmation)* ;
- ***(CommandeAnnuler**, Connexion.Commander.Commander(5).Confirmation.**Annuler**)* $\rightarrow$ ***(CommandeConfirme**, Connexion.Commander.Commander(5).**Confirmation**)*.

4.3.4 Pattern de la substitution

Exemple

Dans ce pattern de substitution d'opérations, on a remplacé les opérations *Confirmation* et *Confirmation* dans l'ancienne version de protocole par respectivement *ConfirmationCmd* et *ConfirmationInscri* dans la nouvelle version (voir Fig 4.4).

Description et utilité

Dans ce pattern, c'est le changement d'un élément par un autre du même type dans l'ancienne version du protocole qui constitue la nouvelle. L'élément peut être un état ou un message. Mais, nous nous interesserons qu'aux messages car l'état peut être considerer comme fictif,

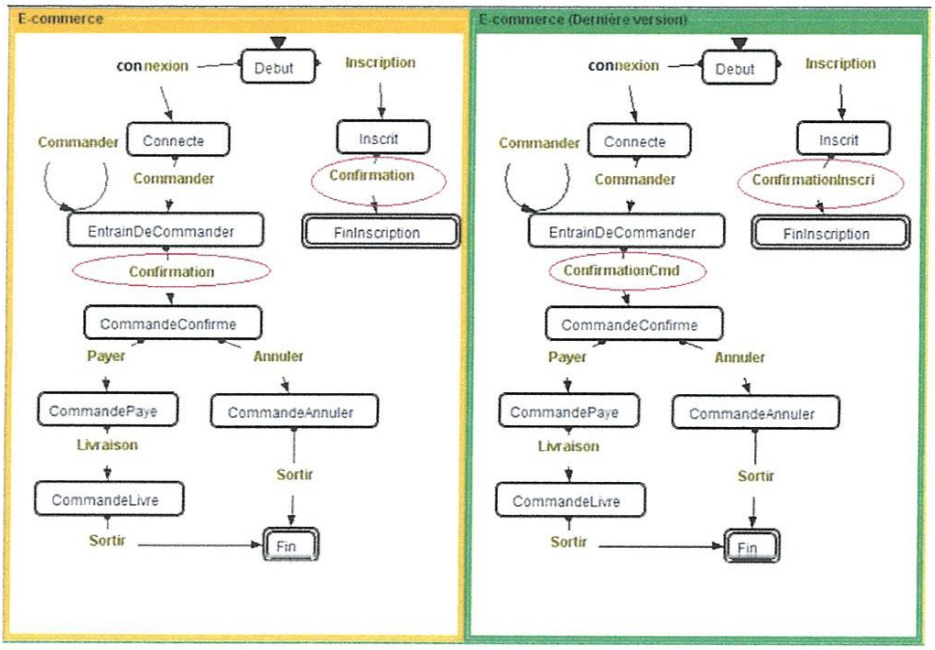

FIG. 4.4 – Exemple d'évolution pour le pattern de substitution d'un BP d'e-commerce.

donc sa substitution est sans grande importance. Les deux ensembles de paramètres de ce pattern sont respectivement dans P et P' les opérations substituées α et celles qui vont le remplacer α'.

Le pattern de substitution est utile lors du regroupement d'un ensemble d'opérations en une autre pour plus d'abstraction, ou l'inverse ou encore lors du remplacement d'une activité par une autre.

Formalisation

$$(q_s, \delta) \xrightarrow[\alpha/\alpha']{Subtitution} (q_t, \beta), \ni \delta_0, \delta_1 tel\ que\ \ \delta = \delta_0\alpha\delta_1 et\ \beta = \delta_0\alpha'\delta_1 \qquad (4.4)$$

Exemple de migration

De la même manière que pour le pattern de réduction, il faut, dans ce pattern, substituer les opération à remplacer dans une trace active dans P pour obtenir sa trace témoin. Pour les traces dans le tableau 4.4 nous avons :

- les traces 1, 2,3 et 5 qui peuvent migrer comme suite (dans la même ordre) :

- *(CommandeAnnuler, Connexion.Commander.Commander(3).***Confirmation***.Annuler)* →
  *(CommandeAnnuler, Connexion.Commander.Commander(3).***ConfirmationCmd***.Annuler)*

Numéro	Etat attient	Activte	Traces d'exécutions
1	CommandeAnnuler	True	Connexion.Commander.Commander(3). Confirmation.Annuler
2	CommandeConfirme	True	Connexion.Commander.Commander(1). Confirmation
3	CommandeLivre	True	Connexion.Commander.Commander(4). Confirmation.Payer.Livraison
4	FinInscription	False	Inscription.Confirmation
5	EntrainDeCommander	True	Connexion.Commander.Commander(2)
6	Fin	False	Connexion.Commander.Commander(6). Confirmation.Annuler.Sortir

TAB. 4.4 – Exemple de migration de traces pour le cas du pattern de substitution.

- *(CommandeConfirme, Connexion.Commander.Commander(3).**Confirmation**)* → *(CommandeConfirme, Connexion.Commander.Commander(3).**ConfirmationCmd**)*;
- *(CommandeLivre, Connexion.Commander.Commander(3).**Confirmation**.Payer.Livraison)* → *(CommandeAnnuler, Connexion.Commander.Commander(3).**ConfirmationCmd**.Payer*
- *(EntrainDeCommander, Connexion.Commander.Commander(2))* → *(EntrainDeCommander, Connexion.Commander.Commander(2))*.

  - les traces 4 et 6 sont inactives.

4.3.5 Pattern de l'extension

Exemple

Dans cet exemple de pattern de l'extension on a ajouté les opérations (*choixAnnuler*, *Sortir*) à l'ancienne version de protocole (voir Fig 4.5).

Description et utilité

C'est le pattern dual de celui de la réduction. Donc l'ensemble de paramètres vide ici est celui dans P celui qui ne l'est pas, α', est constitué des opérations ajoutées à P pour former P'.

Ce pattern est utile dans les cas suivants :

- Panne de fournisseur de service : (voir le pattern strict), le client aura besoin d'un service qui a au moins les mêmes fonctions que son précédent.
- Fusion de BPs.

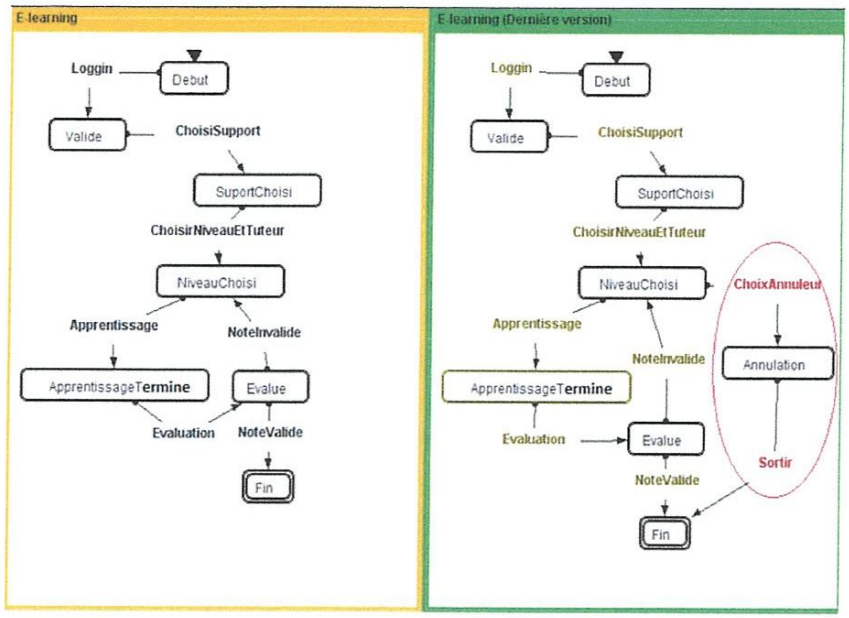

FIG. 4.5 – Exemple d'évolution pour le pattern d'extension d'un BP d'e-learning.

Formalisation

$$(q_s, \delta) \xrightarrow[\alpha']{Extension} (q_t, \beta), \ni \delta_0, \delta_1, \alpha' tel\ que\ \delta = \delta_0\delta_1 et\ \beta = \delta_0\alpha'\delta_1 \qquad (4.5)$$

Exemple de migration

Ce dernier pattern fonctionne de même façon que le pattern strict. Si une trace active dépasse les opérations ajoutées, elle n'a pas besoin d'y revenir. Sa trace témoin reste identique à l'initiale. Mais si c'est pas le cas, elle pourrait exécuter ces opérations ajoutées. Ainsi dans

Numéro	Etat atteint	Active	Traces d'exécutions
1	Fin	False	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur. Apprentissage.Evaluation.NoteValide
2	Fin	False	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur. Apprentissage.Evaluation.NoteValide.
3	SuportChoisi	True	Loggin.ChoisiSupport
4	Evalue	True	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur. Apprentissage.Evaluation
5	ApprentissageTermine	True	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur. Apprentissage
6	NiveauChoisi	truee	Loggin.ChoisiSupport.ChoisirNiveauEtTuteur

TAB. 4.5 – Exemple de migration de traces pour le cas du pattern d'extension.

les traces du tableau 4.5, nous avons :

- les traces numero 1 et 2 sont inactives, nous la migration les concernent moins,
- les 4 dernièers (3 à 6) peuvent migrer aux mêmes états cibles que les états sources et avec les mêmes traces témoins que les initiales. Par exemple pour la trace 6, on a :

- (*NiveauChoisi, Loggin.ChoisiSupport.ChoisirNiveauEtTuteur*) → (*NiveauChoisi, Loggin.ChoisiSupport.ChoisirNiveauEtTuteur*).

4.4 Prise en compte des patterns lors de la migration

Nous exploiterons les différents patterns identifiés et formalisés en amont pour décider de la migration d'une trace active δ qui a atteint un état q_s dans l'ancienne version du protocole. La question de la migration revient à trouver les deux arguments suivants : q_t et β.

En effet lors de la migration, une trace active δ peut migrer vers une nouvelle trace β si la contrainte de compatibilité des protocoles est respectée. Or, les choix de gestion du gestionnaire de protocole peuvent lui donner un degré de liberté supplémentaire par la relaxation de cette contrainte souvent forte. En ce sens, la migration peut être possible même si l'ancienne et la nouvelle version de protocoles ne sont pas strictement compatibles. L'objectif d'identification des patterns est justement, pour pouvoir les exploiter lors de la migration. On dit par la suite qu'une trace active δ peut migrer vers la nouvelle version du protocole avec une trace correspondante β qui satisfait un des patterns d'évolution cités ci-dessus.

Conclusion

L'attention portée sur la notion de design pattern dans ce chapitre est juste contextuel, c'est dans la perspective de cerner le besoin des patterns d'évolution dans notre étude. A la suite de l'identifiication et la formalisation de ces patterns, on peut entamer la question de la migration des traces. La conception et la structure de l'approche pemettant d'y repondre sont étalées dans le prochain chapitre.

Chapitre 5

Approche pour la gestion de la migration des traces actives

Introduction

L'impact de l'évolution des protocoles du service Web sur ses instances actives nécessite la mise en oeuvre d'une approche globale permettant la gestion de la migration automatique de ces instances conformément aux patterns d'évolution identifiés dans le chapitre précédent. Dans ce chapitre, nous exposerons la démarche conceptuelle de l'approche que nous avons proposé et qui compose trois étapes :

5.1 Définition et Calcul du produit asynchrone d'automates

5.1.1 Définition

La prémière étape de notre approche consiste à concevoir une structure de données permettant d'avoir notre premier attribut de migration (q_t). Cette structure se nomme le produit cartésien asynchrone d'automates représentant les deux versions du protocole. Ce produit prend en entré deux automates et fourni comme sortie, un autre automate qui donne toutes les possiblités de combinaison de couple d'états des deux automates donnés en entrée. Il constitue une plate forme exprimant touts les similitudes et touts les différences entre les deux automates en entrée. Formellement soient :

$P = \{Q, M, q_0, Q_f, R\}$ exprime l'ancien protocole, avec

Q = l'ensemble des états de l'automate P, M = l'ensemble des messages, (l'aphabet dans la théorie des automates), q_0= l'état initial,

Q_f = l'ensemble des états finaux, R = l'ensemble des transitions dans P.

$P' = \{Q', M', q'_0, Q'_f, R'\}$ le nouveau protocole. Les attributs sont les mêmes mais concernent $P\prime$.

La structure du produit cartésien **asynchrone enrichi** de ces deux automates se présente comme suite :

$P \otimes P\prime = \{Q_c, M_c, q_{c0}, Q_{cf}, R_c, L\}$, les cinq premiers attributs ont la même signification que pour les automates ordinaires formalisés précédemment, mais pour le produit d'automate. L est une fonction qui spécifie de quel automate vient l'opération exécutée.

$L(m_i) = \{^{-\ \text{if}\ m_i \in M \setminus M'}_{+\ \text{if}\ m_i \in M' \setminus M}\}$

Le produit cartésien est dit asynchrone car il englobe toutes les possibilités de combinaison d'état (q_s, q_t). Il existe d'autre type de produit d'automate en occurrence le produit synchrone mais ce dernier ne nous interesse pas car il ne donne pas toutes ces possibiltés.

Il est dit enrichi car chaque attribut a des extensions que celui d'un automate normal n'a pas. Ces extensions sont pour :

- **Les états de l'automate :**

$Qc = Q \times Q\prime \times \Sigma^* \times \Sigma'^*$, et $|Qc| = |Q| \times |Q'|$ c'est à dire, le nombre d'états du produit est égale au produit des nombres d'états des deux automates de entrée.

Chaque état est caractérisé par 4 attributs :

- L'état associé à l'automate de l'ancienne version du protocole : $q_i \in P$
- L'état associé à l'automate de la nouvelle version du protocole : $q'_i \in P'$
- La séquence d'opération exécutée dans l'automate de l'ancienne version du protocole, de son état initial jusqu'à son état courant. Elle est notée $\omega_i^- \in \Sigma^*$ (ω_i^- *est une expression régulière et* Σ^* *: ensemble des mots)*
- La séquence d'opération exécutée dans l'automate de la nouvelle version du protocole, de son état initial jusqu'à son état courant. Elle est notée $\omega_i^+ \in \Sigma'^*$ (ω_i^+ *et une expression régulière)*

Donc, les états du produit d'automate se présentent sous cette forme : $(q_i\ ,\ q'_i\ ,\ \omega_i^-\ ,\ \omega_i^+)$

- **L'état initial :** $q_{c0} \in Q_c = (q_0\ ,\ q'_0\ ,\ \varepsilon\ ,\ \varepsilon)$

- **Les états finaux :** $Q_{cf} = Q_{\mathbf{f}} \times Q'_{\mathbf{f}} \times \Sigma^* \times \Sigma'^*$

- **Les messages :** $M_c = M \times \{-\} \oplus M' \times \{+\}$ avec $\oplus$ = l'opérateur du "ou exclusif"

Soit deux nouveaux alphabets notés '-' et '+' permettant respectivement de savoir si l'opération exécutée est de l'ancienne version du protocole ou de la nouvelle afin de polariser les transition par la source du message.

- **La fonction de transition :** $R_{\mathbf{c}} = ((q_i, q'_i, \omega_i^-, \omega_i^+), (q_{i+1}\ ,\ q'_{i+1}\ ,\ \omega_{i+1}^-\ ,\ \omega_{i+1}^+), \alpha_i)$

Exprime la transition d'un état source $(q_i, q'_i, \omega_i^-, \omega_i^+)$ pour atteindre un état terminal $(q_{i+1}$, q'_{i+1} , ω_{i+1}^- , $\omega_{i+1}^+)$ en exécutant l'opération α_i qui est soit de P, soit de P'.

5.1.2 Les caractéristiques du produit d'automates

Dans le produit d'automates proposé, on dégage les caratéristiques suivants :

- Pour chaque état, un attribut descriptif de l'état et la séquence d'opération exécutée dans l'une et l'autre version du protocole de l'état initial jusqu'à l'état courant.
- Chaque opération est polarisée soit par '-' ou '+' selon qu'elle provient respectivement de l'ancienne ou de la nouvelle version du protocole. Cette notation est conventionnelle et est omise de l'opération lors de la comparaison ou la recherche.
- L'historique des exécutions d'opérations (dans chaque version du protocole) est exprimé sous forme d'expression régulière nommée : ω_i^- et ω_i^+. Ces expressions régulières utilisent seulement deux opérations : la concaténation et l'étoile de Kleene. L'opérateur d'union n'est pas permis car il n'y a pas d'alternatif de choix pendant la transition et l'historique d'opérations est unique. Ceci permet d'avoir un automate resultant déterministe. Les expressions régulières sont construites sur les opérations polarisées des protocoles : $M \times \{-\} \oplus M' \times \{+\}$. C'est en se basant sur cette historique que nous determinerons notre trace correspondante (β) dans la nouvelle version du protocole.

5.1.3 Comment construire le produit asynchrone enrichi d'automates ?

Le produit cartésien asynchrone est consruit par un algorithme qui manipule les syntaxes formelles des deux automates. Avant de passer à cet algorithme, présentons un exemple de produit asynchrone enrichi. Soit deux protocoles P (*le cadre gauche*) et P' (*le cadre droit*) représentés par les automates de la figure 5.1 . Le produit de ces deux automates est donné par la figure 5.2.

Le protocole initial P (*ancienne version*) évolue en P'(*nouvelle version*) suivant le pattern de suppression avec comme paramètres les opérations "*Confirmer, Sortir, Annuler*".

Dans ce produit d'automates (*voir* Fig 5.2), vous constaterez que :

- le nombre d'états est égal à 12 =(4*3) et que ce produit couvre toutes les combinaisons de couple (q_s, q_t) d'états.
- les caracteristiques de chaque état sont uniques.
- les historiques d'exécution d'opération dans l'une et l'autre version sont désigné par (ω_i^-, ω_i^+), comme dans l'exemple ci-dessous.

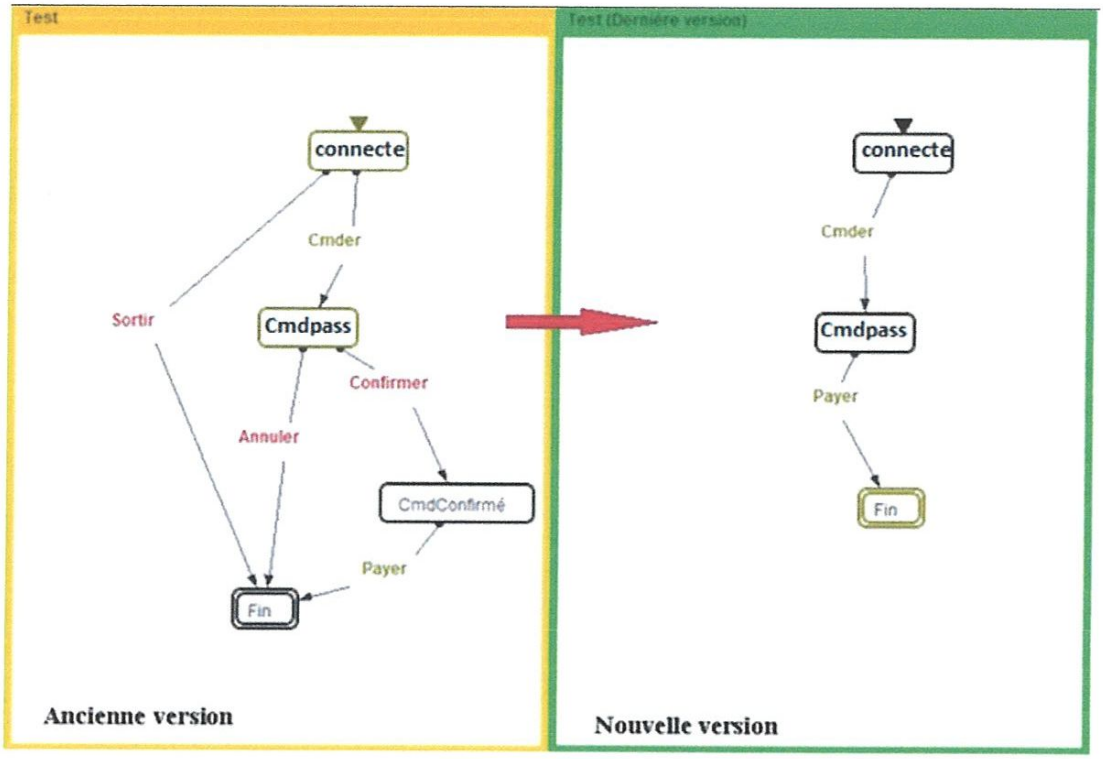

FIG. 5.1 – Exemple d'automates opérandes du produit cartésien asynchrone.

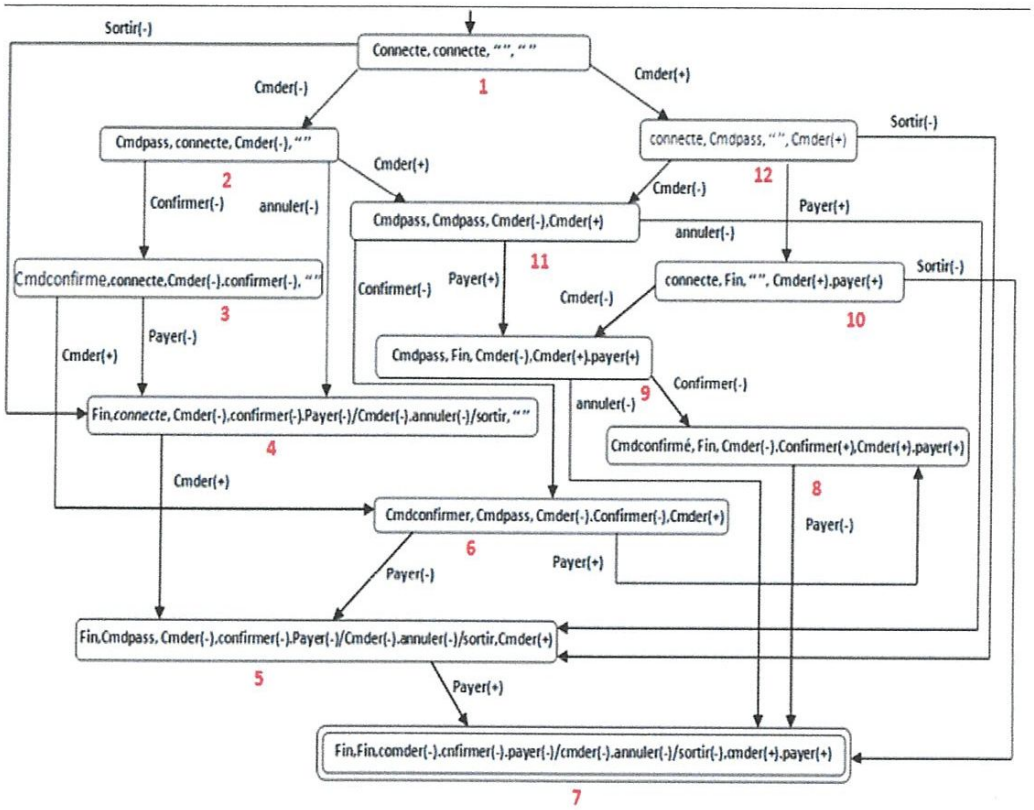

FIG. 5.2 – Le produit cartésien synchrone enrichi des deux automates de la figure 5.1

requête SQL permettant de faire cette recherche est la suivante :

R2*=SELECT *etatNew
FROM *R1*
WHERE δ IN *exprOld*
AND β IN *exprNew.*

Si on trouve cet état, dans notre exemple c'est l'état numéro **6**, alors l'état cible de la migration est l'attribut ***état du nouveau automate*** de cet état du produit. Ce qui dans notre exemple donne q_t = **CmdPass**

(*CmdConfirmé*, *Cmder.Confirmer*) $\rightarrow$ (*CmdPass*, *Cmder*).

- Trace **3,** on applique la même démarche :

Aucune opération du pattern d'évolution ne figure dans la trace d'exécution, donc β = δ =Cmder et q_s =CmdPass. La selection des états primaires du produit d'automates donne la liste d'états suivante : **2, 9, 11**. La reconnaissance de δ (resp. β) par ω^- (resp. ω^+) nous conduit à l'état **9**. Donc q_t = **CmdPass**.

(*CmdPass*, *Cmder*) $\rightarrow$ (*CmdPass*, *Cmder*).

5.3 Recherches des attributs de la migration dans le produit

Après son calcul, notre produit d'automates sera stocké dans une base de données relationnelle. L'interrogation de cette dernière via un langage déclaratif d'interrogation, notamment le langage SQL, nous permet d'obtenir nos attributs de migration. Avant de passer à l'exploitation de cette base de données, nous présentons sa conception et les liens sémantiques qui relient les différents objets du domaine.

5.3.1 Conception de la base de données

Toute application informatique manipule d'une manière ou d'une autre un ensemble de données qui sont généralement stockées dans une base de données. La notre est constituée de huit entités dont les liens sont présentés dans le MCD (Modèle Conceptuel de Données) de la figure 5.3.

Soit le protocole P (l'ancienne version) de la figure 5.1. Nous determinons la migrabilité des traces du tableau 5.1

Etant donné qu'aucun modèle de conception n'a été imposé, nous avons opté pour le modèle Entité/Association (E/A) de Merise au détriment du diagramme des classes d'UML. Ce choix est du en grande partie à la facilité et à notre maitrise du modèle choisi. En plus, même si l'UML est plus adéquat aux langages orientés objets, l'utilisation de son diagramme

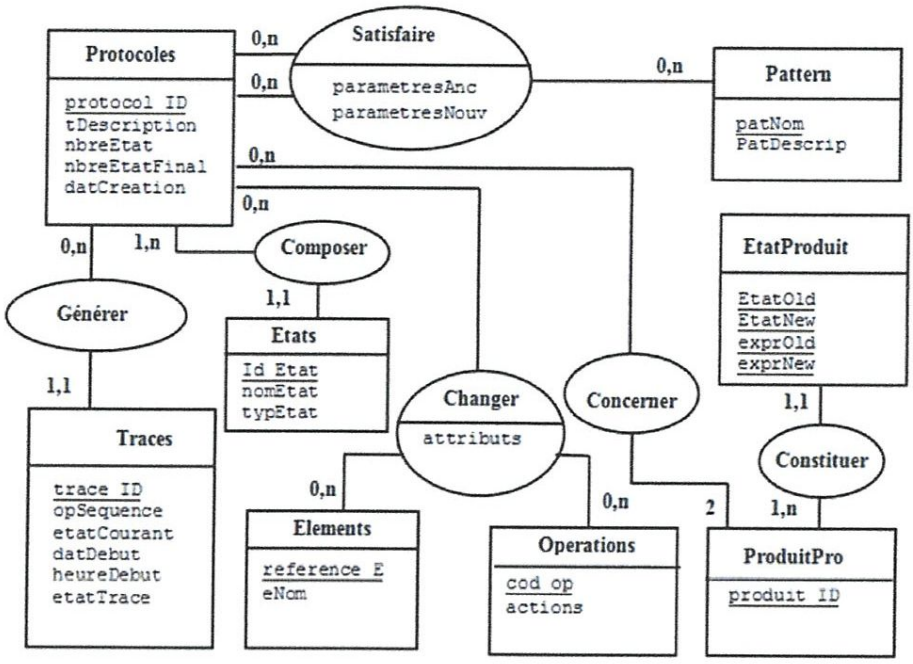

FIG. 5.3 – Le MCD décrivant le domaine.

des classes n'est pas assez bénéfique ici. Il sera sous exploité vu qu'aucune table de la base n'a de méthode encapsulée (voir Fig 5.3).

Pour réaliser la base de données, on doit transformer nos entités/associations en schémas relationnels (MLD) qui seront traduits et implémenté sous un Système de Gestion de base de données (SGBD). Le SGBD utilisé dans notre application est MySQL. Là encore la facilité d'utilisation influe beaucoup sur ce choix en plus de sa gratuité bien sur. Ces schémas relationnels sont :

Protocoles *(protocol_ID, tDescription, nbreEtat, nbreEtatFinal, datCreation),*

Traces *(trace_ID, opSequence, etatCourant, datDebut, heureDebut, etatTrace, #protocol_ID),*

Etats *(Id_Etat, nomEtat, typEtat, #protocol_ID),*

Pattern *(patNom, PatDescrip),*

Satisfaire *(protocol_ID1, protocol_ID2, patNom, parametresAnc, parametresNouv),*

ProduitPro *(produit_ID, #protocol_ID1, #protocol_ID2),*

EtatProduits *(EtatOld, EtatNew, exprOld, exprNew, #produit_ID),*

Elements (reference_E, eNom),

Changer (reference_E, cod_op, protocol_ID, attributs),

Operations (cod_op, actions).

5.3.2 Exploitation de la base de données pour la recherche des attributs de la migration

Il s'agit d'interroger la base de données via des requêttes SQL afin de trouver les attributs q_t *et* β de migration d'une trace active δ qui atteint un état q_s dans P. Cette dernière sera également chargée de la même base de données. En guise d'exemples de requête, voir plus haut, dans l'exemple 5.2 de ce chapitre.

Conclusion

Après avoir identifié le problème que pose l'evolution d'un protocole de service Web sur ses instances, nous avons dans ce chapitre étalé la conception de l'approche qui nous permettra de le résoudre. Au bout des trois étapes de cette approche, nous pourrons effectivement décider de la migration, dans la nouvelle version du protocole, d'une trace active relative à l'ancienne version. Nous ne saurons terminer cette conclusion sans noté que l'algorithme, du calcul de produit d'automate, proposé n'est pas exhaustif. Il est ouvert à tout ajout positif dans un souci d'amélioration.

En informatique, on dit toujours que la théorie sans pratique est incomplète. Alors pour compléter notre étude du problème, nous avons mis en pratique l'approche proposée. Cette implémentation fait l'objet du dernier chapitre qu'est le prochain.

Chapitre 6

Implémentation

Introduction

En vue d'atteindre les objectifs fixés au début de ce projet et concrétiser en termes de programmation l'algorithme proposé dans le chapitre précédent, nous abordons dans ce chapitre l'aspect pratique relatif à notre travail. Nous rappelons que ces objectifs sont les suivants :

Modéliser les protocoles métiers par des automates d'état finis déterministes.

Effectuer une simulation de l'exécution des services par des clients à travers la génération des traces sur les protocoles définis.

Décider de la migration des traces générées sur un protocole après une évolution de ce dernier.

Nous y présenterons l'environnement de travail, les outils utilisées dans cet environnement, l'architecture générale de notre application ainsi que ses interfaces principales avec les grandes fonctionnalités.

6.1 Présentation de l'environnement de travail

Dans le développement de notre application, les différentes fonctionnalités convoitées requièrent différents outils. Ces outils seront présentés dans cette section ainsi que les raisons qui ont motivé leur choix.

6.1.1 Le choix d'Eclipse et JAVA

Comme environnement de programmation nous avons opté pour le couple java/Eclipse.

a. Eclipse

Eclipse est un environnement de développement intégré (IDE) gratuit. Il permet de développer des programmes en plusieurs langages : java, PHP, C,... Il utilise le principe intéressant

des plugins, ce qui permet d'ajouter des fonctionnalités selon les besoins (*l'extensibilité*). C'est un IDE à complétion contextuelle c'est-à-dire qu'il aide à corriger les erreurs syntaxiques du code.

Il se présente sous différentes versions, dans notre travail la version qu'on a utilisé est la 3.6. Elle a vu le jour en juin 2010 et s'intitule ***Eclipse Helios***.[17]

b. Java

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

Il est à signalé que malgré la grandeur de la bibliothèque native de java, certaines fonctionnalités, tel que la manipulation des document XML, l'accès à une bas de données et bien d'autres, nécessitent des APIs qui ne sont pas intégrées dans java par défaut, on les appele les APIs externes. Donc il faut les intégrer et cela se fait d'une manière très simpliste à travers l'IDE Eclipse, c'est d'ailleurs un de ses points forts.

Dans notre travail, les APIs externes utilisées sont :

1. Jdom

JDOM est une API permettant la manipulation (création, modification, suppression, ...) de documents XML via une structure arborescente. Il présente quelque similitude avec DOM (Document Objet Model). Cependant, il s'en distingue parce que JDOM est spécifiquement conçu pour JAVA et que du point de vue du développeur JAVA, il s'avère beaucoup plus pratique à utiliser (à ce propos, il convient de noter que contrairement à ce qui est parfois écrit, le J de JDOM ne renvoie pas à Java, et que JDOM suit la nomenclature NAA - not an abreviation - de Sun : JDOM veut ainsi dire JDOM et rien d'autre). Sans plus détaillé, la raison qui nous a poussé à l'utiliser est sa simplicité et la possiblité de sa manipulation avec java.

Pour plus d'informations sur JDOM et DOM consulter la reférence [19]

2. mysql-connector

L'API qui permet l'accès aux bases de données sous java se nomme JDBC. Son fonctionnement requiert un drivers qui peut être de différents types (il existe 5 type de drivers, voir [20]). Comme nous utilisons MySQL en guise de SGBD alors le type 4, pour plus de

performances en termes de vitesse et de d'utilisation de ressources, est adéquat à notre application. ***mysql − connector*** est une API qui fourni ce drivers de type 4 conforme à MySQL. Chaque SGBD à son propre drivers.

C'est une ficher exécutable jar (Java ARchive) qui englobe toutes les classes nécéssaires à l'accès à la base de données.

6.1.2 Le choix de SQL

Le besoin, à divers niveaux de notre application, d'accéder à la base de données nécessite un langaged'interrogation de base de sonnées. Et pour cela nous avons choisi SQL. Accronyme de *"Structured Query Language* " c'est-à-dire "Langage d'interrogation structuré", SQL est un langage complet de gestion de bases de données relationnelles. Il a été conçu par IBM dans les années 70 et est devenu le langage standard des systèmes de gestion de bases de données (SGBD) relationnelles(SGBDR).[18]

6.1.3 Le choix de XML

XML est devenu omniprésent dans le monde de l'informatique. De nombreux standards sont apparus et permettent à des applications différentes de stocker mais surtout de partager des documents. Ce succès de XML est en grande partie du à ses qualités qui sont :

Séparation stricte entre contenu et présentation,

Simplicité, universalité et extensibilité,

Format texte avec gestion des caractères spéciaux,

Structuration forte,

Modèles de documents (DTD et Schémas XML),

Format libre : facilité d'exporter un document XML sous divers autres format : pdf, html, ...

Existance d'outils de manipulation : JDOM, DOM, ...

Dans notre application, nous l'utilisons pour la représentation des automates qui modélise les BP.

Après avoir présenté l'environnement de réalisation de notre application, nous passons à la présentation de son l'architecture.

6.2 Vue globale de l'application

Au bout de notre étape de conception, l'application qui implémentera l'approche proposée a la vue globale 6.1. On l'a bâptisé **PCIA** (**P**rotocol **C**hange **I**mpact **A**nalyser) et elle est consitutée de deux bases de données :

- une base de donnée relationnelle (BDDR) pour faciliter manipulation d'informations via les clause SQL. Dans cette base seront stockées les traces d'exécutions des clients, les protocoles métiers sous forme de tuples, le produit d'automates, ... (Voir 5.3 du chapitre 5).

- et une base de données XML (BDD XML), en fait c'est un ensemble de fichiers xml répresentant les protocoles, ceci facilite la manipulation des éléments (états et messages) de ces derniers.

1≡ L'utilisateur, en occurrence le gestionnaire de protocoles, entre dans l'application. Il crée les BP, les modifie, les évolue. Il simule l'exécution des protocoles par des clients, cette exécution est répresentée par une trace d'exécution qui sera enregistrée dans la BDDR. Et enfin il peut décider de la migration des traces sur un protocole, après l'evolution de ce dernier, en donnant pour chaque trace migrable δ la trace témoin β ainsi que l'état cible q_t atteint dans la nouvelle version du protocole

2≡Après avoir créé ou modifié un protocole, ce dernier est enregisté dans la base XML sous format xml d'états/Transition (*voir Exemple 1.1 de chapitr 1*). C'est dans cette base que le gestionnaire de protocoles peut, partir de l'application PCIA, charger un protocole pour diverses manipulation ainsi que pour la simulation de l'exécution par les clients.

2'≡En même temps que l'enregistrement dans la BDD XML lors de la création, la modification ou l'évolution d'un BP, des informations sur ce dernier sont également stockées dans BBDR, dans les tables protocoles et etats (Voir Fig 5.3 du chapitre 5). Ces informations sont également assujetties à des modications.

3≡Lors de la simulation de l'exécution du protocole par des clients, les traces générées sont stockées dans la BDDR. Ceci permet de faciliter leur visualisation et la déterminaison de leur migrabilité. Le resultat du calcule du produit asynchrone est également enregistré dans cette même BDDR.

4≡Pour déterminer cette migrabilité d'une trace active, le gestionnaire de protocole charge cette dernière à partir de la BDDR enfin de lui appliquer la démarche proposée dans la section 5.2 du chapitre précédent.

En plus de ces 5 points capitaux, notre application propose au gestionnaire différents tableaux de bord sur les protocoles. Ces statistiques lui permettent de savoir l'état d'avancement d'un protocole, par exemple le nombre traces actives sur ce dernier, son nombre de versions, les patterns pris en compte lors des évolutions, ...

La figure 6.2 présente un zoom de notre application dans sa vue globale. A travers cette figure, on constate qu'elle est consituée de cinq principaux menus :

GP≡ Gestion de Protocoles, Ce menus à travers ces huit items permet de modéliser les processus métiers (création, évolution statique, enregistrement) et les stocker dans BDD XML. C'est le centre de toute l'application et il est inhérent aux autres menus.

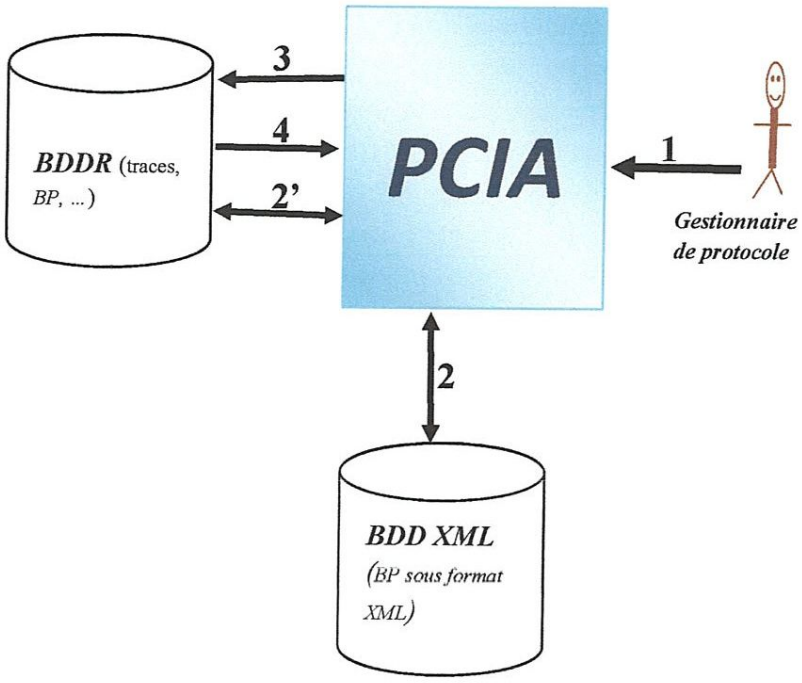

FIG. 6.1 – Une vue globale de notre application.

EP≡ Evolution de Protocoles, ce menus permet de faire une évolution dynamique d'un protocole déjà créé par le menu précédent. Il dépend du menu GP (la flèche ***a***).

ST≡ Suivi des traces. A travers ce menus, le gestionnaire de protocoles peut générer et stocker dans BDDR des traces relatives à un protocole ainsi que de visualiser ces traces. Il présente deux formes de génération : la génération manuelle, permet de créer une trace précise par selection manuelle des opérations effectuées ; et la génération aléatoire qui facilite la création d'un grand nombre de traces. Ce menu a également besoin de protocole déjà modélisé, donc du menu GP (flèche ***b***).

AM≡ Analyse de la Migration, c'est dans ce menu que se passe la fonctionnalité principale de l'application à savoir décider de la migrabilité d'une trace active. Il permet de calculer le produit asynchrone de deux protocoles (protocole initial et évolué) chargés de BDD XML (flèches ***c*** et ***d***) et stocker le resultat dans BDDR. C'est à partir de ce résultat qu'on peut determiner si une trace active chargée de BDDR (flèche ***f***) est migrable oui ou non. Ce menu interagit avec les trois précédents.

Statistiques≡ Enfin ce dernier menu propose au gestionnaire de protocoles, dans un cadre supervision, quelques tableaux de bord relatifs au protocoles (flèche ***e***).

A titre additif, l'application inclu un menu **Aide** qui ne figure dans ce zoom. Il présente un explication brève de comment utiliser l'application.

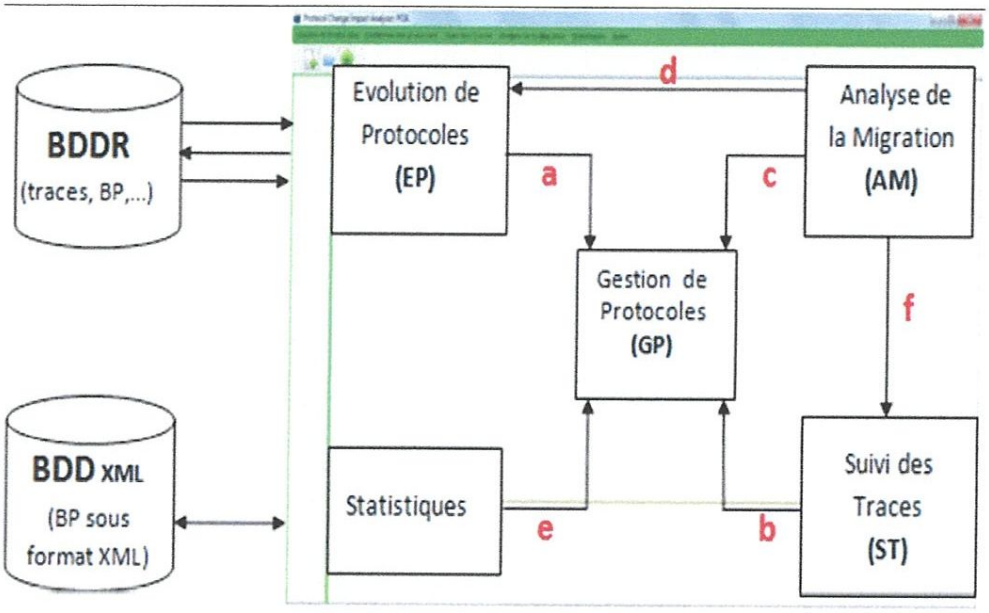

FIG. 6.2 – Zoom sur PCIA dans la vue globale de notre application.

Remarque 6.1 $X \underset{\rightarrow}{u} Y \equiv$ *Le menu X depend et interagit avec le menu Y.*

6.3 Description de l'application

Cette section est consacrée à la présentation, en termes d'image de capture, les principales interfaces de l'application. Entre autres nous avons :

1. Création, modification ou évolution d'un protocole métier

La figure 6.3 représente l'interface de la création d'un nouveau protocole métier. A l'ouverture de l'application, l'interface est dépourvu de contenue à part la barre de menu (**2**). A partir du menu de ***Gestion de protocoles***, en cliquant sur l'item ***nouveau***, une boîte d'outils de création de protocole (**1**) s'ajoute à cette interface. C'est à partir de cette boîte qu'on peut créer, modifier ou faire évoluer un protocole métier. En même temps que la création, modification ou évolution d'un BP, un analyseur vérifie la présence d'erreur l'automate, donne sa syntaxe formelle, sa réprésentation XML, les différents chemins présents dans l'automate, la description du protocole ainsi que la liste de ces version d'évolution s'il y en a. Ces différentes informations sont présentées dans le panneau d'en bas (**3**).

A l'ouverture d'un BP pour une modification (menu ***Gestion de protocoles*** $\rightarrow$ ***ouvrir***) ou pour une évolution (menu ***Evolution des protocoles*** $\rightarrow$ ***Evolution d'un protocole***),

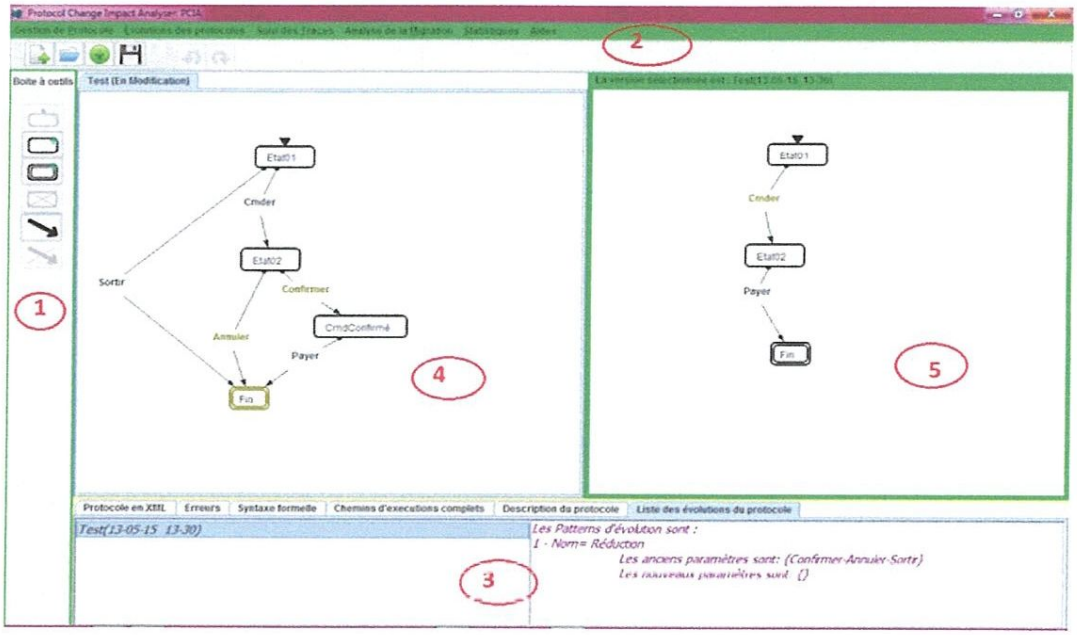

FIG. 6.3 – Interface pour la modélisation et l'évolution d'un protocole.

on a une visualisation directe (4) du protocole chargé. De même, si on clique sur une version d'évolution (dans le dernier onglet du panneau d'en bas), on a une visualisation de ce dernier dans (**5**).

Dans la barre d'outils, en dessous de barre de menu, figurent des icones pour l'accès rapide aux opérations les plus fréquentes : création de BP, ouverture d'un BP en modification ou en évolution, enregistrement d'un BP, ...

Remarque 6.2 *(i) désigne une partie sur la figure courante, indiquée par le cercle numéro* ***i*** *sur cette figure.*

2. Génération et visualisation des traces d'exécution

Les figures 6.4 et 6.5 représentent les interfaces pour la gestion des traces d'exécution. Dans la première, on charge un protocole au niveau de (**1**) et ce dernier sera visible en (**2**) ainsi que sa dernière évolution s'il y en a. Selon le mode de génération choisi : manuelle ou aléatoire, on respectivement dans la deuxième figure (**3** et **3′**) ou (**4** et **4′**) qui s'affichent.

Pour la génération manuelle, on choisi les opérations du protocole à exécuter, ces opérations sont indiquées par le menu déroulant de (**3′**). Pour la génération aléoitre, on indique juste le nombre de traces à générer dans le champs de texte de (**4′**).

Une visualisation sous différents aspects (par protocole, par état, par période, par latence) est également présente dans ce menu (menu ***Suivi des traces*** → ***Visualisation***). La

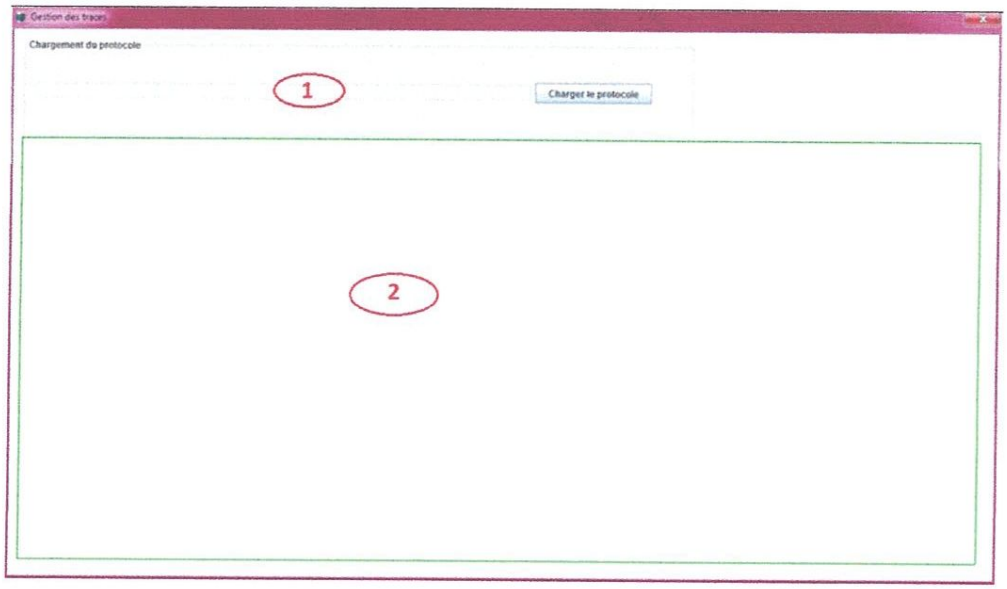

FIG. 6.4 – Interface initiale pour la génération des traces.

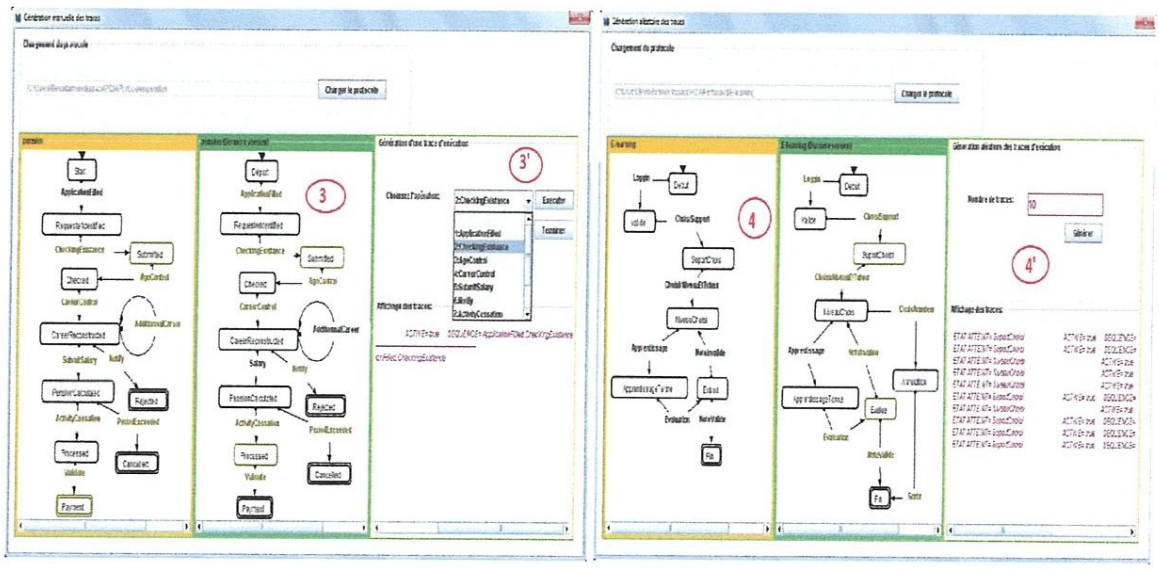

FIG. 6.5 – Interface pour la génération manuelle ou aléatoire des traces.

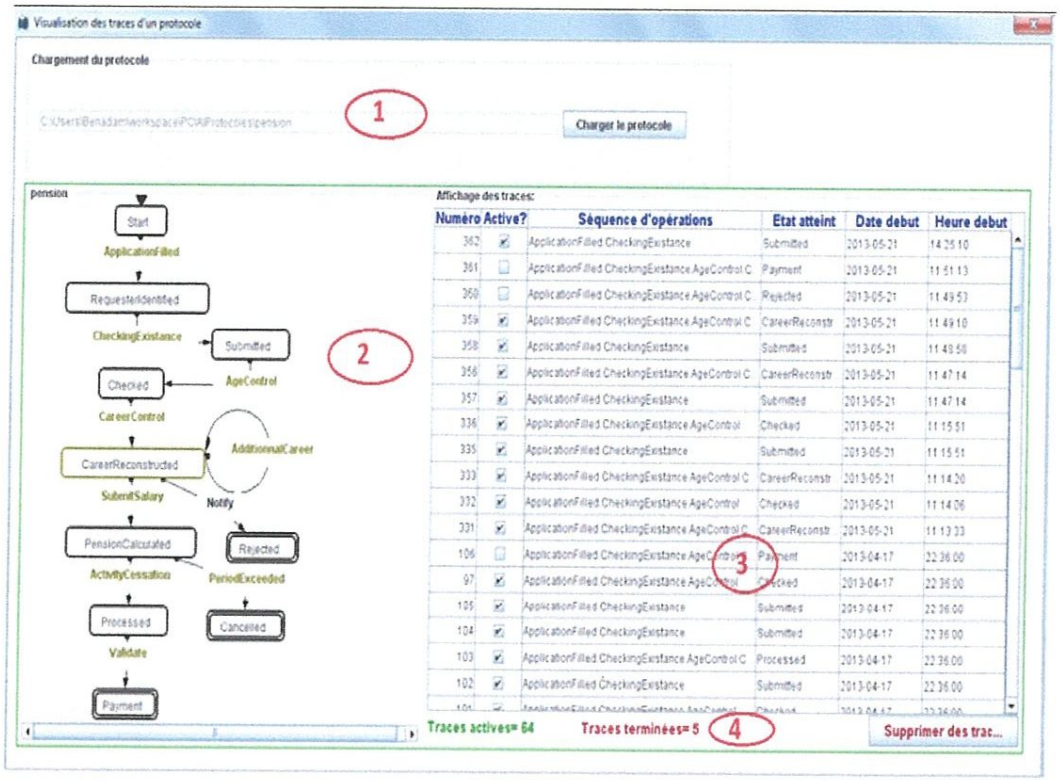

FIG. 6.6 – Interface pour la visualisation des traces d'un protocole.

figure 6.6 présente l'interfacc corrcspondante. Une fois le protocole chargée dans (**1**), qui s'affiche dans (**2**) ; selon de mode visualisation choisi, on a les résultats qui s'affichent dans (**3**). Si une trace s'avère incohérente, on peut également la supprimer à partir du boutton ***Supprimer des traces*** de (4)

4. Calcul du produit asynchrone de protocoles et décision de la migration d'une trace active

L'interface qui prend en charge le calcul du produit asynchrone de protocoles est présentée par la figure 6.7. Les deux protocoles sont chargés au niveau de (**1**) et affichés dans (**2**) et (**3**). Ensuite, en cliquant sur le boutton ***calculer le produit***, une boîte de dialogue nous indique que le calcul est teminé.

Une fois ce produit calculé, le gestionnaire de protocoles peut alors charger des traces actives d'un protocole et déterminer de la migrabilité de ces dernières conformément à la version choisi de ce protocole. L'interface qui le prend en charge est celle de la figure 6.8. Le gestionnaire charge les deux versions du protocoles dans (**1**), ils sont affichés dans (**2**) et (**3**). Ensuite il selectionne dans la liste de (4) les traces à migrer. Une fois ces traces selectionnées,

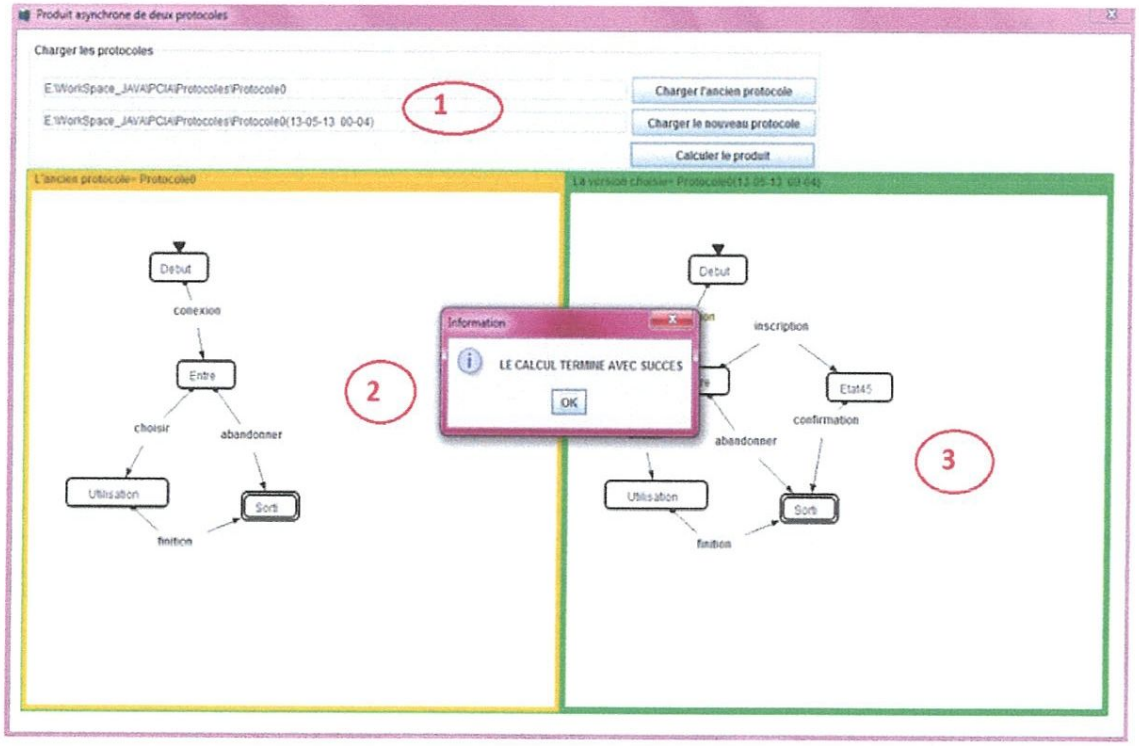

FIG. 6.7 – Interface pour le calcul du produit asynchrone de protocoles.

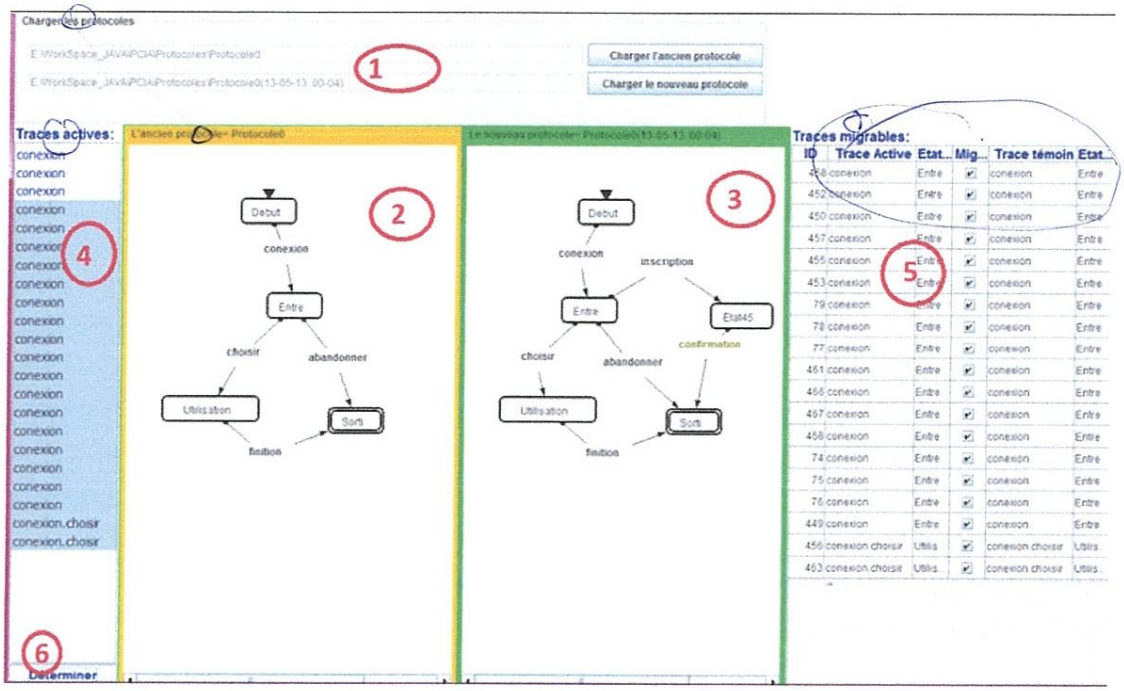

FIG. 6.8 – Interface pour la migration de traces d'exécution

il clique sur le boutton ***Determiner*** de (**6**). Ce clique permet de décider de la migrabilité des traces selectionnées. Pour chacune de ces dernières, les informations relatives à la migration, *l'identifient de la trace, la séquence d'opérations, l'état atteint, la décisison de migration (oui ou non), la trace témoin et l'état cible,* sont affichées dans le tableau de (**5**).

5. Etat d'avancement d'un protocole

Dans le menu ***Statistiques***, l'item ***Sur un protocole***, le gestionnaire de protocoles peut avoir une vue sur l'état d'avancement d'un BP. C'est-à-dire le nombre de traces à chacun de ses états. Ceci lui permet d'évoluer un BP sans causer beaucoup problèmes de migration. L'interface qui correspond à cette fonctionnalité est présentée dans la figure 6.9.

6. Aide sur comment utiliser l'application.

C'est une fonctionnalité supplémentaire d'auto description de l'application. Elle permet de savoir comment utiliser l'application et la dépendance entre ces différents menus. On y accède par le menu ***Aides***, item ***comment utiliser?***

Conclusion

Aux termes de cette étape de notre projet, nous sommes parvenus à réaliser une application qui répond brillamment aux objectifs fixés au début. Mais comme toute première version

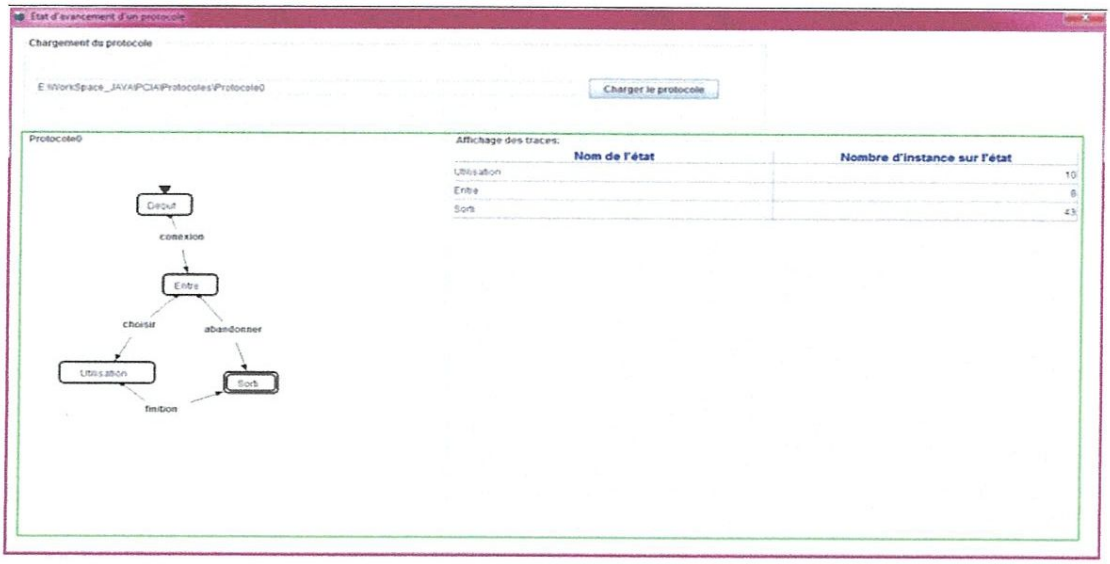

FIG. 6.9 – Interface pour visualiser l'état d'avancement d'un protocole.

d'une application, la notre est assujettie à des imperfections que nous comptons résoudre avec le temps.

Conclusion générale

Dans ce mémoire, nous avons fait le parcours d'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. Notre attention s'est axée sur un niveau particulier de cette technologie : c'est le niveau des protocoles métier associé aux services. Ce niveau confère plus de cohérence à un service web vis-à-vis de ses utilisateurs.

Nous somme partie du général, à savoir le besoin des protocoles métiers dans les services Web, son cycle de vie (modélisation, déploiement, ...), pour aboutir à un problème particulier qui est l'impact du changement de ce protocole vis-à-vis des clients qui exécutent le service Web. Après avoir identifié le problème, nous avons mise en place une conception rigoureuse qui en théorie le résout idéalement. Enfin nous avons expérimenté cette conception en le mettant en pratique à travers la partie implémentation dans un langage complet et d'actualité à savoir java.

En guise de perspectives, nous proposons :

- La combinaison des patterns d'évolution lors de la spécification des patterns que satisfait une évolution de protocole. Cela donne plus de rigueur à la conception et plus de possibilité de migration des clients.
- L'ajout d'autres patterns car, comme déjà précité, les patterns qu'on a identifiés ne recouvrent pas tous les cas.
- Dans nos patterns, l'élément considéré est le message (c'est-à-dire la transition dans les automates). Nous proposons d'étendre la granularité, en considérant toute un segment comme élément au lieu du message simple.

Sur le plan personnel, ce mémoire nous a permis de :

- Capitaliser nos acquis depuis la licence jusqu'à maintenant entre autre les techniques de conception de logiciel, de gestion de projet, conception de base de données, ...
- Découvrir les techniques d'intégration de SI en général et la technologie des services Web et les standards y afférents en particulier.
- Comprendre le mode fonctionnement de XML et améliorer nos connaissances en Java.
- Et enfin, prendre conscience encore plus des bienfaits du travail en équipe.

Nous espérons que la lecture de ce mémoire a été amusante, intéressante et sans beaucoup d'ambiguïtés. Car comme un adage le dit : « ***La seule chose qu'on est sûr de connaitre est qu'on ne connait rien*** ».

Bibliographie

[1] **Systèmes d'Information et Méthode MERISE,** http://www2.lirmm.fr/IC/Supports/FMIN 114-ConduiteProjets/Supports_ThereseLibourel/MERISE_2010.pdf

[2] Encyclopédie moulin.

[3] **Chemakhi et Bekakria, Mr Khebizi Ali**, Mémoire de fin d'études pour l'obtention du diplôme de Master2, 2011, Thème: Analyse de l'impact des changements de protocoeles de services sur la compatibilité et l'équivalence des services.

[4] **Gerard et Fadel, Mr Khebizi Ali**, Mémoire de fin d'études pour l'obtention du diplôme de Master2, 2012, Thème : Fouille Des Processus Métiers Approche Déclarative .

[5] **Taleb et Layada, Mr Khebizi Ali**: Mémoire de fin d'études pour l'obtention du diplôme de Master2, 2010, Thème : L'analyse de la compatibilité et de l'équivalence des protocoles des services Web enrichis par les effets transactionnels.

[6] docinsa.insa-lyon.fr/these/2009/jaber/these.pdf

[7] **W.M.P. van der Aalst**, Process-Aware Information Systems: Lessons to be Learned from Process Mining, wwwis.win.tue.nl/~wvdaalst/publications/p522.pdf, Novembre 2008

[8] Microsoft ® Encarta ® 2009. © 1993-2008 Microsoft Corporation.

[9] **Tanguy Crusson**, Business Process Management, www.urba-ea.org/telecharge.php?doc=ref040 31258401284.pdf..., Octobre 2003

[10] **Pascal**, Forum sur les compétences en sécurité des SI, http://www.forum-des-competences.org /files/resourcesmodule/@random4e15889c4f3d1/1310034176_livrable_trace.pdf, Décembre 2004

[11] www-igm.univ-mlv.fr/~dr/XPOSE2002/.../traces_informatiques.htm.

[12] **Gustavo Alonso and Al**, Web Services:Concepts, Architectures and Applications, Springer,2004

[13] **Gauthier Picard**, Design Patterns, Outils pour la Gestion de Projets, http://www-igm.univ-mlv.fr/~cherrier/download/imac/Poo5.pdf, Novembre 2011

[14] **P. Laroque**, Design patterns: Eléments de conception réutilisables, http://2011.rmll.info/IMG/pdf/RMLL-oop-patterns-2011.pdf, Octobre 20011

[15] Jean-Christophe Routier, Principes de conception et Design Patterns, http://www.commentcamarche.net/contents/genie-logiciel/design-patterns.php3, Octobre 2012

[16] **Khebizi Ali**, http://www.informatik.uni-trier.de/~ley/pers/hd/k/Khebizi:Ali.html, Octobre 2012

[17] **Rémi Forax**, Eclipse pour les null, http://igm.univ-mlv.fr/ens/Master/M1/2007-2008/JavaAvance/pdf/EclipsePourLesNull.pdf, Septembre 2007

[18] **HANS-PETTER HALVORSEN**, Tutorial: LanguageStructured Query Language, http://home.hit.no/~hansha/documents/database/documents/Structured%20Query%20Language.pdf, Avril 2012

[19] **Cyril Vidal**, Tutorial: Dom et Jdom, http://www-lium.univ-lemans.fr/~lehuen/master1/xml/doc/Tutoriel%20DOM%20et%20JDOM.pdf, Avril 2008

[20] **Jean Michel DOUDOUX**, Développons en java, chapitre 40, Octobre 2009 pdf

[18] http://home.hit.no/~hansha/documents/database/documents/Structured%20Query%20Language.pdf

[19] http://www-lium.univ-lemans.fr/~lehuen/master1/xml/doc/Tutoriel%20DOM%20et%20JDOM.pdf

[20] **Jean Michel DOUDOUX**, Développons en java, chapitre 40, Octobre 2009