REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed KHIDER – BISKRA

Faculté des Sciences et des Sciences de l'ingénieur

Département d'informatique

N° d'ordre :……
Série :……

Mémoire
Présenté en vue de l'obtention du diplôme

Magister en Informatique

Option: **Intelligence artificielle et systèmes d'information avancés**

SUJET DU MÉMOIRE :

Une méthode de conception et de réalisation des processus métiers basés sur les composants et les services Web

Présenté le : 10/04/2008

Par : **KERDOUDI Mohammed Lamine**

Composition du jury:

Mr. DJEDI NourEddine	Président	(Professeur à l'Université de Biskra)
Mr. BOUFAIDA Mahmoud	Rapporteur	(Professeur à l'Université Mentouri de Constantine)
Mr. KAZAR Okba	Examinateur	(Maître de Conférence à l'Université de Biskra)
Mr. LAHLOUHI Ammar	Examinateur	(Maître de Conférence à l'Université de Biskra).

Remerciements

C’est un agréable devoir de s’acquitter des dettes de reconnaissance cumulées tout au long de ce travail de recherche. Je voudrais adresser mes sincères remerciements à tous ceux qui m’ont aidée à préparer le mieux possible ce travail.

Je remercie vivement mon directeur de thèse, Mr. BOUFAIDA Mahmoud, pour avoir dirigé et encadré mon travail de recherche, ainsi que pour leurs conseils certainement très utiles pour mon avenir. Je suis aussi très reconnaissant de leur soutien et leur encouragement permanent.

J’exprime également ma sincère reconnaissance à Mr. DJEDI NourEddine qui me fait l’honneur de présider le jury, ainsi que aux membres du jury, Mr. KAZAR Okba et Mr. LAHLOUHI Ammar, qui ont donné de leur temps pour évaluer ce travail.

Je tiens à remercier tous mes amis (Tarek, Fouzi, Chawki, Okba, Yacine, Nabil etc.) qui m’ont aidé de prés ou de loin à mener ce travail à son terme.

Enfin, je ne pourrais pas terminer sans remercier infiniment mes parents pour leur éternel soutien, ainsi que mes frères, mes sœurs et toute ma famille pour leurs encouragements.

SOMMAIRE

LISTE DES FIGURES

Introduction générale

Aujourd'hui, il n'est plus à démontrer que les entreprises doivent tenir compte de plus en plus d'informations provenant des multiples applications hétérogènes. Le système d'information d'une entreprise doit donc masquer l'hétérogénéité des applications logicielles en proposant une vision unifiée et intégrée aux différents acteurs de l'entreprise. L'EAI[1] est un concept qui est la conséquence du besoin d'intégrer des systèmes d'information pour former un tout cohérent et opérationnel à la fois au sein des entreprises et entre entreprises. Cette intégration d'applications au niveau données a été tout à fait populaire pendant un certain temps avant que les entreprises aient compris l'importance des processus d'affaires pour maintenir une croissance continue et en bénéficié face aux conditions les plus extrêmes du marché [12]. Pour aligner la vision des affaires d'une organisation avec les processus d'affaires, les analystes ont vu l'intégration d'applications à un niveau d'abstraction plus élevé, appelé l'intégration de processus métiers d'entreprises. Ce modèle permet l'échange d'informations entre les applications comme parties d'un processus d'affaires qui est contrôlé par un BPM[2]. Avec les approches traditionnelles d'intégration d'applications la plupart des analystes d'affaires se plaignent que les affaires sont conduites par des modèles préparés par le service de la technologie d'information (Information Technologie Department) [12], qui ne regroupe pas toutes les parties intégrantes d'une affaire. Mais avec l'approche d'intégration de processus métiers les vues multiples d'analystes d'affaires, des développeurs, des clients et des partenaires se traduisent en un modèle simple de processus affaire. Cette approche d'intégration donne aux analystes d'affaires un contrôle plus rigide, facilitant la modélisation, la surveillance et l'optimisation des processus d'affaires. Ceci permet de réduire les coûts des processus inutiles et assure une plus grande satisfaction du client. En outre cette approche s'assure que les données sont au bon endroit et au bon moment réduisant la durée du cycle de tout le processus.

Les développeurs s'efforcent à construire des modèles plus riches pour concevoir les processus métiers, permettant de répondre à des critères de fiabilité soulevés à travers des problèmes du monde réel qui s'imposent, et que les concepteurs essaient de les respecter plus fidèlement. En effet les travaux actuels ne permettent pas de calquer exactement le déroulement du processus métier comme il est perçu par le concepteur. La problématique sous-jacente est liée à la définition des fonctionnalités et des structures de données utilisées. L'organisation de ces processus métiers conçus doit être cohérente et conforme aux besoins des développeurs. Notre objectif dans ce mémoire est de proposé une méthode à suivre pour aboutir à une bonne conception de processus métiers. Pour cela nous nous sommes basés sur une nouvelle approche de développement de systèmes, celle de la réutilisation. Actuellement la réutilisation est essentielle, afin de pouvoir construire des systèmes modulaires à partir des éléments (objets, composants, etc.) existants éprouvés et réutilisables. Cela permet, en particulier, de réduire le temps et les coûts d'un projet de développement. Elle fut évoquée suite à un échec patent de la part des développeurs pour livrer des logiciels sur mesure, rapidement et à des prix compétitifs. Les travaux de la réutilisation ont progressivement évolués ces dernières années, passant de la réutilisation

[1] Entreprise Application Integration

[2] Business Process Management

de code à la réutilisation de connaissances et de démarches de raisonnement de différents niveaux. Pour mettre en œuvre cette approche de réutilisation la tendance des travaux actuelle va vers l'utilisation des composants réutilisables.

Dans le domaine de la réutilisation des composants il existe plusieurs types de composants réutilisables dont nous allons éclaircir par une synthèse plus loin dans ce mémoire. Citons quelques uns : les composants d'analyse [11], les patrons processus [11], les patrons de conception [7], les patrons d'architecture [13], et les composants d'implantation comme CCM (CORBA Component Model) [9], EJB (Entreprise Java Beans) [10]. Vu que notre travail à pour but de contribuer dans la conception de processus métier, nous avons choisi d'utiliser les patrons de conception (Design Pattern) comme étant des modèles de conceptions. Ces derniers facilitent la réutilisation des solutions de conception efficaces [7]. Ils fournissent un vocabulaire commun aux concepteurs pour communiquer, documenter, et explorer les différentes solutions d'une conception. En outre ils font paraître un système moins complexe, en permettant d'en parler à un plus haut niveau d'abstraction, que celui qu'autorisent une notation de conception, ou un langage de programmation. Le catalogue complet des patrons de conception a été publié dans [7]. Il comporte 23 patrons de conception, extraits de plusieurs cadres d'applications, nous nous intéressons dans ce travail à un de ces patron qui est l'observateur[3], est une solution à un problème qui nous classons similaire à notre problématique. Nous montrons dans la suite du mémoire les motivations et la manière de leur utilisation.

Le fait de vouloir réaliser l'intégration des applications inter-entreprises, nous amené à réfléchir sur la manière de franchir les frontières de cette dernière de la façon la plus appropriée, pour pouvoir partager les processus métiers sur l'Internet et les rendre disponibles aux partenaires. L'arrivée des services Web a révolutionné la communication entre les applications de l'entreprise. Ces types de services permettent d'étendre les applications tout en facilitant beaucoup la communication entre les applications disparates.

Les services Web changent fondamentalement les règles du commerce électronique. Ils relient des applications entre eux à travers les points éloignés sur l'Internet, en transportant de grandes quantités de données plus efficacement par la transmission de messages standardisé de type XML basée sur le protocole SOAP (Simple Object Access Protocol). Le résultat est une communication plus rapide, meilleure et plus de productivité pour les entreprises et ainsi le consommateur lui-même [31]. Un service Web peut être implémenté en tant qu'un processus d'affaires. Il peut se composer de plusieurs processus d'affaires (public et privé). Chaque processus d'affaire peut être implémenté comme un service Web lui-même. Chaque activité qui fait partie d'un processus métier est logiquement liée à un service Web. Cependant, Nous allons utiliser dans ce travail une composition de services Web pour réaliser notre processus métier. Tant que le Pattern Observer définie dans [7] est conçu pour des applications Orienté Objet, alors la question qui se pose est comment l'adapter aux processus métiers. La solution qui apparaît évidente est d'avoir une structure du Pattern Observer s'adapte aux services Web, ce travail à été initié par le Pattern proposé dans [48]. L'objet de notre travail c'est l'adaptation de ce Pattern aux processus métiers, tout en proposant un modèle qui traite les anomalies d'exécution des processus métiers.

[3] Nous utilisant le terme anglais Le Pattern Observer dans le reste du mémoire

L'objectif de ce mémoire est de proposer une méthode de conception et de réalisation des processus métiers en utilisant les Design Patterns et les services Web. Cette conception permet de répondre aux critères qui sont bien définit par les concepteurs. Ces critères décrivent les mécanismes de recouvrement en cas des problèmes d'exécution des processus métiers. Un processus métier est représenté par un ensemble de services (activités) qui s'exécutent. L'échec de l'exécution d'un de ces services nécessite un traitement qui concerne l'annulation ou la compensation des effets des autres services. Ce traitement est spécifié par le concepteur en termes des critères simple. En d'autre terme, les critères du concepteur seront définit sous la forme d'une paire : un service doit être échoué, et une liste de services qui doivent être traité (compenser ou annuler). Le Pattern Observer en résumé contient deux éléments essentiels qui sont : un "Subject" signifiant une classe notifiant des observateurs lors de ses modifications, et un "Observer" qui représente une classe recevant des notifications d'un "Subject". Un "Subject" peut jouer le rôle du service qui doit être échoué, et un "Observer" peut jouer le rôle d'un service qui doit être annulé ou compensé. En effet nous avons décidé d'utiliser ce pattern pour concrétiser les critères du concepteur par l'intégration de ce pattern avec le modèle du processus métier, tout en adaptant les méthodes de chaque classe du pattern à notre contexte sans perdre l'essence du Pattern. Ensuite nous ajouterons de nouvelles structures et de nouvelles méthodes.

Premièrement, nous offrons dans ce travail une flexibilité parfaite et simple aux concepteurs pour la définition de leurs besoins en termes de critères simple. Deuxièmement, nous intégrons le Pattern Observer dans notre modèle de conception des processus métiers, tout en respectant les critères des développeurs. Enfin, nous implémentons un exemple concret d'un processus métier en se basant sur les services Web tout en respectant notre méthode proposée.

La suite de ce document est organisée en cinq chapitres :

- Le premier chapitre représente un état d'art sur le domaine de la gestion des processus métier et de la notion du processus métier (Business Process, BP).
- Le deuxième chapitre nous montre les mécanismes de réutilisation dans l'ingénierie de logiciels et les différents critères de classification des composants réutilisables.
  Nous conclurons ce chapitre par une discussion sur les Design Patterns.
- Le troisième chapitre sera consacré aux services Web et nous montrera la correspondance avec les processus métiers.
- Quatrièmement nous présenterons les différentes phases de notre méthode et les motivations de l'utilisation du Pattern Observer.
  Dans une étape suivante nous montrerons notre modèle de conception des processus métiers, ensuite nous utiliserons le diagramme de séquence d'UML pour démontrer le fonctionnement du modèle, tout en montrant la manière d'agir aux problèmes d'exécution.
- Dans le chapitre final nous utiliserons les services Web pour réaliser un processus métier, tout en suivant notre méthode proposée.
  Nous avons choisi une agence d'importation comme exemple de processus métiers, avec laquelle nous démontrerons la manière de spécification des critères du concepteur et comment réagir aux problèmes d'exécution selon ces critères.
- Nous finirons par une conclusion générale avec ses perspectives.

Chapitre I - Gestion des processus métiers (BPM)

1. Introduction

Dû aux changements contextuels tels que l'économie du marché, les attentes des clients qui se produisent à un rythme de plus en plus rapide, les entreprises sont appelées à s'adapter à son environnement. Dans le contexte concurrentiel que nous vivons, une implémentation presque instantanée des changements est la clé du succès. Cependant, les entreprises doivent adapter leurs processus pour refléter les objectifs fixés. Pour ce faire il faut l'adoption d'une solution de gestion des processus métiers. En effet, une mauvaise compréhension des processus d'affaires résulte inévitablement en une exécution inconsistante et inefficace. En outre, dans une situation normale, l'efficacité d'un processus métier décroît avec la complexité et le temps nécessaire à sa réalisation. Dans ce cas, les coûts augmentent car ils sont inversement proportionnels à l'efficacité. Grâce à l'utilisation d'un outil de gestion des processus métiers, cette tendance s'inverse. En effet, la coordination des différentes étapes des processus entre les différents systèmes informatiques, les personnes et les partenaires commerciaux, apportent une grande valeur ajoutée à l'entreprise en améliorant l'efficacité du processus métiers (moins de temps et moins d'erreurs).

Au cours de la dernière décennie, les entreprises ont porté une attention croissante aux processus métiers, à leurs descriptions, leurs automatisations et leurs managements. La représentation et l'automatisation des processus métiers possèdent son propre champ d'étude, le BPM. Ce terme est populaire aussi bien dans le domaine métier que scientifique, qui relève des questions qui concernent, notamment la conception, l'analyse, la modélisation, l'exécution et le contrôle des processus métiers. Ce chapitre présente le domaine de la gestion des processus métiers autour de la présentation des concepts de base des processus métiers et de leur gestion par des systèmes informatiques.

2. Notion du processus métier

Avant de discuter sur la gestion des processus métiers (BPM), nous nous essayons dans cette section de comprendre la notion du processus métier et leur impact sur l'entreprise.

2.1 Définition d'un processus métier

Plusieurs définitions du terme processus sont présentées dans la littérature. Il est vrai que cette notion est suffisamment générale pour être utilisée dans différents domaines scientifiques ou applicatifs. L'étude de ces définitions permet d'avoir une idée claire de ce qui est désigné par un «processus ». Selon Michael Havey « Un processus effectue des actions pendant un intervalle de temps afin de réaliser, ou progresser à un certain objectif » [26].

Cependant, le concept processus métier (BP, Business Process) a été défini « en tant qu'un enchaînement d'activités corrélées ou interactives dans l'entreprise. Un processus reçoit des objets en entrée et leur ajoute de la valeur, par le moyen de ressources, tout en fournissant des objets de sortie (services) remplissant les besoins et les exigences d'un client (atteindre les objectifs) internes ou externes à l'entreprise. Il ne peut être déclenché que par des événements internes et/ou externes à l'entreprise. Chaque processus est en communication avec d'autres et peut être décomposé en sous-processus. Une activité transforme des entrées en sorties par l'influence d'objets de contrôle et en utilisant les ressources requises et disponibles pendant une durée bien définie » [40]. Nous nous n'essayerons pas d'étendre ou de raffiner cette définition d'autant que d'autres définitions ont été proposées.

Simplement dit, un processus métier définit comment les employés effectuent leurs tâches quotidiennes [39]. De nombreuses transactions commerciales mettent en jeu une succession complexe d'informations et de décisions où chaque participant peut affecter la suite de déroulement du processus.

Cependant, le terme qui apparu important dans la définition est celui de l'activité, il représente une étape dans un processus métier. Chaque activité a un nom, un type, des pré et des post conditions et des contraintes d'ordonnancement. Chaque activité a un conteneur des données en entrée et un conteneur des données résultats.

2.2 Place et rôles des processus métiers dans une entreprise

Une entreprise qui est basée sur ses processus métiers a clairement identifié et défini les procédures qui entrent en ligne de compte dans ses relations commerciales. Lorsqu'une entreprise atteint cet état, elle a une base solide pour évaluer son cadre de travail et adopter des changements bénéfiques. Toute tâche qui n'appartient pas ou ne supporte pas un processus métier est considérée comme redondante et peut être éliminée. Tout processus peut être amélioré pour diminuer les coûts et augmenter les performances [39].

Dans une entreprise basée sur les processus métiers l'employé se préoccupe uniquement des parties des processus qui lui sont attribuées. Les avantages d'une approche par processus métiers sont des cycles de transaction plus courts, une plus grande facilité pour l'utilisateur de savoir ce qu'il a à faire ainsi que des besoins moindres de formation. Les utilisateurs sont assignés à des rôles spécifiques et ne remplissent que les tâches dévolues à leurs rôles. En effet, un groupe de personnes peut prendre en charge des rôles pour éviter tout goulot d'étranglement et permettre une distribution des tâches [39], cela est vient grâce aux avancées technologiques comme les outils de communication sans fil, les changements culturels dans les organisations ont permis une grande mobilité du personnel.

Pour garder une structure fluide, donc les processus doivent être définis non pas à partir des personnes mais au travers de rôles. Le travail n'est donc plus attribué à une personne précise mais à une des personnes capables de jouer un certain rôle. Un système de BPM permet de suivre les utilisateurs, leur charge de travail et leur disponibilité pour optimiser la charge de travail et le temps d'exécution.

2.3 Modélisation des processus métiers

Modéliser un processus métier est une étape essentielle de n'importe quel processus de développement d'un BP. Il permet aux analystes de capturer les grandes lignes et les procédures de l'affaire. Ce modèle fournit une vue d'ensemble où le système proposé considéré adapter à la structure organisationnelle et les activités quotidiennes. Cependant, quand on commence à s'intéresser à la modélisation des processus métiers d'une organisation, on se rend rapidement compte que le champ d'application de cette discipline est très large. On se retrouve rapidement noyé sous une pluie d'abréviations, dans un océan de termes qui semblent vouloir dire la même chose mais qui en fait sont bien différents.

2.3.1 Utilisation du Pi-calcul

Le Pi-calcul est langage formel développé par l'Ecossais mathématicien Robin Milner dans les années 90, il permet définir des processus concourant et en communication avec d'autres [26]. Un processus Pi-calcul est écrit en tant qu'un ensemble d'équation en utilisant une syntaxe particulière. Le Pi-calcul offre trois caractéristiques principales pour la modélisation d'un processus métier [26]:

- **Flot de Contrôle**

Un flot de contrôle est défini comme un modèle de processus. Il décrit chaque unité de travail, mais aussi la séquence adéquate d'unités de travail pour atteindre un certain objectif. Un flot de contrôle décrit l'ordre dans lequel les activités sont exécutées en spécifiant des connecteurs de contrôle entre les activités [22]. Dans le Pi-calcul, le comportement séquentiel, parallèle, conditionnel, et récursif d'un processus peut être déclaré succinctement.

- **Communication basé messages**

Le cœur du Pi-calcul est sa syntaxe et sa sémantique propres pour transmission et réception de messages.

- **Mobilité**

Il exige la capacité de changer des adresses dynamiquement. Quand un processus envoie une information à un autre, il inclut le nom du canal à utiliser par le processus destinataire pour sa réponse. Ce nom est de nature variable; il peut être changé si les conditions se changes. Le changement du canal désigné sous le nom de la mobilité. L'adressage dynamique, ou la mobilité est une caractéristique la plus distinctive du Pi-calcul.

2.3.2 Utilisation des réseaux de Pétri

Les réseaux de Pétri ont apparu en 1962 par le mathématicien Carl Adam Petri, est un langage formel de modélisation graphique des systèmes. Il est considéré convenable pour la modélisation des processus métiers [43]. Grâce à sa sémantique il permet la validation par la simulation des modèles construits. Cependant un réseau de Pétri peut être utilisé pour implémenter la sémantique du flot de contrôle, y compris les branchements de base et les règles de jointure, aussi bien que des scénarios plus compliqués de synchronisation [26]. Un réseau de pétri ordinaire est un assemblage des cercles, des rectangles, des line fléchés et des points noirs à l'intérieur des cercles. Ces formes représentent respectivement des places, des transitions, des arcs et des jetons. Le rôle de chacune est le suivant :

- **Place** : est un point d'arrêt dans un processus, représente (généralement) l'accomplissement d'une étape importante ;
- **Transition** : représente un événement ou une action ;
- **Jeton** : est un point noir résidant dans une place. Collectivement, l'ensemble des jetons représente l'état actuel du processus. Pendant l'exécution du processus, le jeton se déplace d'une place à une autre ;
- **Arc** : est un lien d'une transition à une place ou d'une place à une transition.

La figure 1 représente un exemple de modélisation d'un processus métier utilisant les réseaux de Pétri. Cet exemple montre comment un site, comme http://zopa.com, fonctionne.

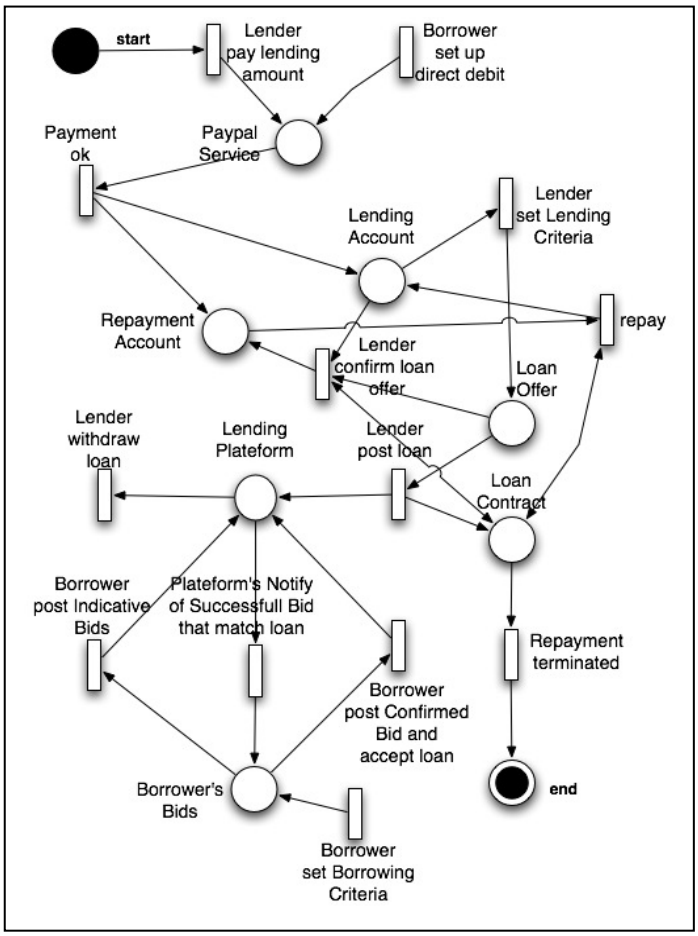


Fig. 1 Flot de contrôle du Zopa [44]

Ce site Web permet aux utilisateurs d'emprunter, ou de prêter de l'argent, entre eux, comme on le ferait avec un système bancaire traditionnel. Mais la grande différence est que ce sont les utilisateurs qui choisissent eux-mêmes, entre eux, la somme, la durée, le risque et le taux d'intérêt du prêt [44]. Le site créé par des spécialistes issus essentiellement de la finance il utilise les technologies Internet pour court-circuiter les systèmes bancaires traditionnels, et apporter un nouveau modèle BP. La modélisation du flot de contrôle de Zopa, faite sous la forme d'un réseau de Pétri illustré par la figure 1. Les détails de fonctionnement du modèle sont expliqués clairement dans [44].

2.3.3 Utilisation d'UML

UML (Unified Modeling Language) comme il indique son nom, un langage de modélisation n'est pas une méthode ou un processus. UML se compose d'une notation très spécifique et des règles grammaticales relatives pour construire des modèles logiciels [21]. UML ne proscrit pas ou ne donne pas un avis sur la façon d'utilisation de cette notation dans un processus de développement logiciel. Il ne participe pas d'une partie d'une méthodologie de conception orientée objet. Il propose des diagrammes spécialisés (dont les diagrammes d'activité, de séquence, etc.) ayant chacun une fonction précise. Il n'existe pour le moment pas de diagramme UML spécialisé pour la modélisation des processus métiers [19]. 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 [19].

Les diagrammes d'activités UML (DA UML) sont des cas spéciaux de diagrammes d'état UML, qui sont à leur tour des représentations graphiques des machines d'état [45]. Dans un DA UML 2,0 l'unité fondamentale des spécifications du comportement est une action ou « Une action prend un ensemble d'entrées et les convertit en un ensemble de sorties, bien que l'un ou l'autre ou les deux ensembles peuvent être vide » [46], les actions peuvent également modifier l'état du système.

Nous nous rappelons qu'un modèle de processus métiers (flot de contrôle) doit définir pratiquement les éléments suivants:

- Le but ou la raison du processus;
- Entrées spécifiques;
- Sorties spécifiques;
- Ressources consommées;
- Activités qui sont exécutées dans un certain ordre;

Un processus métier est une collection d'activités conçues pour produire un résultat spécifique pour un client ou un partenaire particulier. Il implique une emphase forte sur la façon dont le travail est effectué dans une organisation. Un processus représente aussi l'ordre spécifique des activités à travers le temps et l'emplacement, avec un début, une fin, et les entrées et les sorties sont clairement définies [21].

Considérant comme un exemple de processus métier les procédures de réclamations d'assurance présentée dans [26]. La figure 2 représente le flot de contrôle qui est modélisé utilisant le diagramme d'activité d'UML.

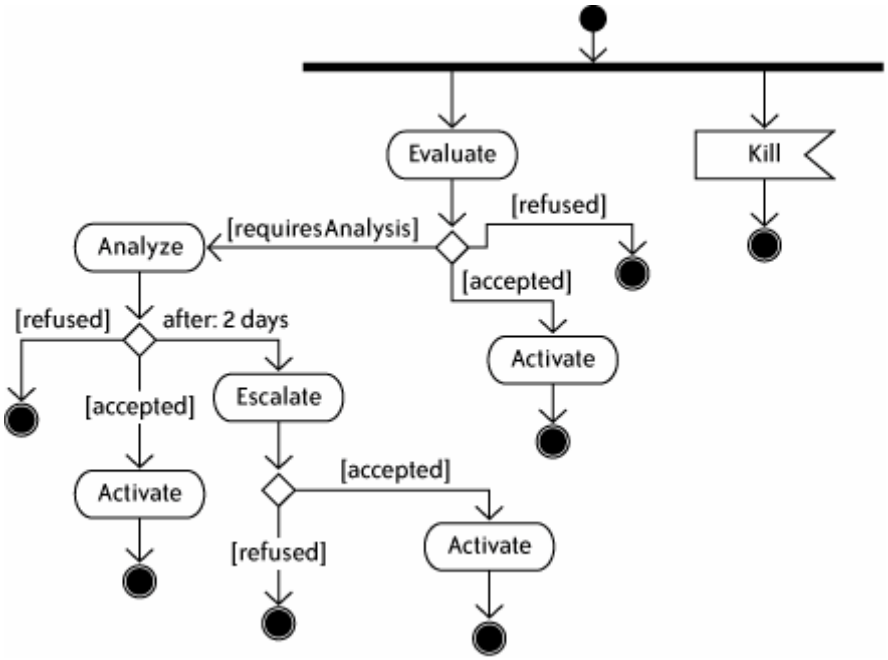


Fig. 2 Processus de réclamations d'assurance utilisant le DA UML [26]

Les formes dans le diagramme sont : des activités (rectangle à coins arrondis), des points de décision (Losanges), des transitions (flèches), l'état de début (boule pleine), les états d'arrêt (boule pleine avec un cercle externe), une bifurcation (trait horizontal en gras), et un récepteur de signal (rectangle avec la queue en forme de V).

Le processus commence en bifurquant dans deux directions: un qui annule le processus global à la réception d'un signal de mise à mort, l'autre contenant le traitement des lignes principales. Une nouvelle réclamation doit être d'abord évaluée (l'activité 'Evaluate'), ou elle peut être refusé, accepté ou passé vers un analyseur (basé sur les conditions émané par une décision qui suit l'activité Evaluate). Le refus arrête immédiatement le processus, quand une transition accepté l'activité 'Activate' se déclenche, elle représente le paiement et tout autre étape de traitement de fermeture d'une réclamation, qui mène à l'accomplissement et l'arrêté d'écoulement; les chemins de refusions et d'acceptations se produisent dans d'autres endroits dans le processus, et ils se comportent identiquement. Le chemin de 'requiresAnalysis' mène à l'activité 'Analyse', ayant trois résultats: « refusé accepté, ou l'expiration du temporisateur de deux jours ». Sur l'expiration, l'activité 'escalation', un traitement prioritaire de l'étape précédente d'analyse, peut-être par un autre analyseur ou superviseur, le résultat final doit être soit refusé ou accepté.

Cet exemple n'illustre pas les activités parallèles et de synchronisations qui sont utilisées largement dans les processus métiers. Cependant nous étudions le parallélisme et la synchronisation dans un autre exemple de processus métier qui constitue d'une agence

de voyage. La figure 3 montre la modélisation de processus métier de l'agence de voyage. Dont trois activités se déroulent en parallèles (réservation de vol et d'hôtel et la location de voiture) pour accomplir un voyage, si une des activités échoue le voyage devra être échoué [26].

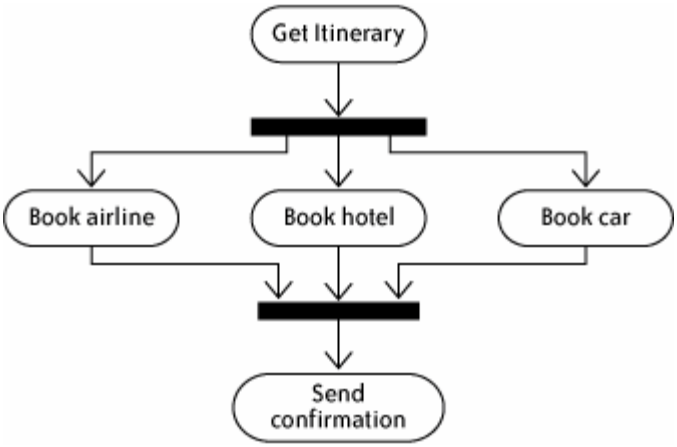


Fig. 3 Processus d'agence de voyage utilisant le DA UML [26]

Cependant, les diagrammes d'activité UML ne remportent pas beaucoup de succès dans la communauté du BPM comme ils ont été dans l'analyse et la conception de logiciel [26]. La limitation de cette approche est tout de même de creuser un gap entre la modélisation des processus et leurs exécutions. Les langages d'exécution de processus tels que BPEL sont directement interprétables par les moteurs des systèmes BPM. Il est donc nécessaire de créer une correspondance entre la représentation graphique des processus (UML ou autre) et leur langage d'exécution (tel que BPEL), afin de permettre un déploiement direct des processus modélisés sans passer par une phase d'implémentation.

2.3.4 Utilisation du BPMN

Le BPMI (Business Process Modeling Initiative) est une organisation son but est la construction des standards et une architecture commune pour le BPM. Elle a commencée par Intalio en 2000, et a crû pour inclure une variété d'organisations, y compris BEA, Fujitsu, IBM, IDS Scheer, Pegasystems, PeopleSoft, SAP, SeeBeyond, Tibco, Virtria, et WebMethods. BPMI est elle-même un membre de plusieurs organisations principales, y compris W3C, OASIS, OMG, et les WfMC [26].

BPMI a proposé la spécification du standard BPMN (Business Process Modeling Notation). Il a été développé pour permettre à l'utilisateur d'affaires de développer une représentation graphique standard et aisément compréhensible des processus métiers [47]. Les spécifications de BPMN 1.0 ont été libérées au public en Mai, 2004. En Février, 2006 le BPMN 1.0 a été adopté comme standard d'OMG [47].

La figure 4 montre une modélisation en BPMN de l'exemple précédent du processus de l'agence de voyage.

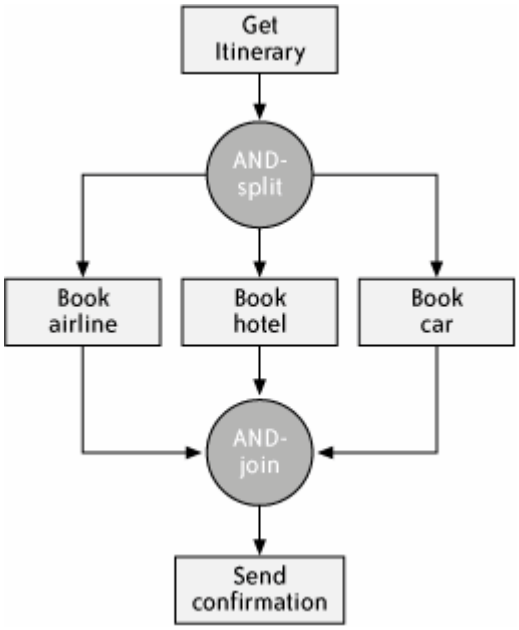


Fig. 4 Processus d'agence de voyage utilisant le BPMN [26]

L'avantage principal du BPMN est le fait qu'il fournit un mapping complet vers les langages d'exécutions. Il est possible de générer automatiquement, et de manière standard, le processus BPEL à exécuter par le moteur de processus [26].

2.3.5 Discussion

Le Pi-calcul et les réseaux de Pétri (ou une certaine combinaison des deux) fournissent les bases théoriques pour la plupart des principaux standards de BPM, quatre principaux standards contemporains sont influencé par le Pi-calcul sont WS-CDL, WSCI, BPML et XLANG. Et d'autres sont dériver à partir des réseaux de Pétri tel BPMN, YAWL et WSFL. Finalement le BPEL est un mélange, hérite les base du Pi-calcul et des réseaux de Pétri, à partir de ses langages parents respectivement XLANG et WSFL [26].

Cependant, Robin indique qu'il y a un nombre restreint d'analystes d'affaires ou de développeurs logiciels pourraient survivre s'il y a lieu pour composer leurs processus métier par des lignes de code du Pi-calcul [26]. D'autre part les réseaux de pétri avec ses techniques d'analyses en permettant la vérification du modèle conçu, les résultats de cette analyse sont utilisés pour évaluer le système et en permettre la modification et/ou l'amélioration dans le cas échéant. En outre les réseaux de pétri permettent la validation des ses modèles par la simulation. L'inconvénient major des réseaux de pétri pour la modélisation des processus métier, est le fait qu'il est difficile pour les analystes d'affaires, ou même par l'équipe technique d'apprendre facilement à modéliser un processus métier complexe. Un autre désavantage ou, un modèle RdP complexe peut représenter un simple processus métier. Notant que les réseaux de Pétri de haut niveau (RdP coloré…) étend la modélisation classique considérée précédemment et facilite la construction de plus grands et plus complexes processus.

Cependant, il y avait également des problèmes plus fondamentaux comme l'absence d'une méthode unifiée de modélisation de processus métiers [22]. La plupart des problèmes techniques ont été résolus à ce jour. Cependant, des problèmes plus conceptuels demeurent non résolus. Un des problèmes est l'exigence d'une conception efficace de processus métier, c'est-à-dire, qu'un concepteur doit construire un modèle détaillé décrivant exactement le déroulement du travail. La conception des processus métiers est loin d'être insignifiante. Elle exige une connaissance profonde du processus métier réel et des paramètres d'évolution au cours du temps. Les besoins d'évolution et d'adaptation aux nouveaux contextes peuvent se manifester particulièrement pendant l'exécution et influencer l'efficacité et la fiabilité du processus métier [22].

3. Gestion des processus métiers

Dans une entreprise idéale, les analystes se chargent d'automatiser les activités de l'entreprise en se basant sur les processus métiers.

3.1 Définition du BPM

La gestion des processus métiers (BPM : Business Process Management) est définie comme « l'art de la compréhension, codification, automatisation, et amélioration la manière d'une compagnie comment effectuer ces affaires » [41]. Le BPM coordonne et gère les relations entre les applications existantes de l'entreprise, les employés ainsi que toutes les relations commerciales en dehors de l'entreprise. En outre il inclut, des techniques, et des outils pour supporter la conception, l'exécution, la gestion, et l'analyse des processus métiers opérationnels [22].

La gestion de processus métiers a rien de nouveau, mais l'utilisation de la technologie de contrôler et améliorer l'exécution des processus métiers est plus récente. L'implémentation technologique de la gestion de processus métier s'appelle un système de gestion de processus métier (BPMS) [41].

3.2 Composants du BPM

Une vraie plate-forme (ou système) BPM doit inclure :

- Un environnement de modélisation des processus où les processus sont construits et modifiés de manière graphique et où sont définis les rôles (et les employés qui peuvent remplir ces rôles), les règles (qui définissent comment doit se dérouler les processus) et les liens entre les différents processus [26].
- Un moteur de gestion des processus qui est un environnement d'exécution où les processus définis dans l'environnement de modélisation sont démarrés, suivis puis terminés avec toute la gestion des différents intervenants (applications ou humains) des délais et des exceptions [26].
- Un environnement de gestion des processus, où, pour un utilisateur donné, sont répertoriées les tâches à exécuter avec toutes les informations nécessaires à la réalisation de sa partie du processus global.
- Un environnement de contrôle des processus qui permet de suivre en temps réels le déroulement des différents processus métier en cours et qui permet également de fournir des analyses de performances synthétiques à l'usage des managers [19].

3.3 Avantages d'un système BPM

Une solution (un système) BPM permet aux entreprises :

- Un déroulement en temps réel des transactions commercial;
- Une gestion en temps réel des prix de vente ou des stocks ;
- D'accélérer grandement les étapes de décisions en apportant en temps réel les informations pertinentes aux bonnes personnes ;
- D'enlever ou de diminuer la participation humaine à un processus ;
- D'implémenter rapidement des changements des règles et des objectifs économiques, alors le BPMS doit fournir la technologie exigée pour implémenter les changements et pour s'assurer que les gestionnaires peuvent réagir rapidement aux conjonctures économiques changeantes [41] ;
- De traverser les frontières pour collaborer avec les partenaires commerciaux comme les fournisseurs et les distributeurs, en intégrant les applications, les utilisateurs finaux et les bases de données... ;
- Un analyse et création de rapport sur les processus : les analystes d'affaires ont besoin d'indicateurs complets sur les différentes instances des processus qu'ils suivent (en cours ou terminés). En basant sur des analyses solides, l'amélioration continue des processus est rendue possible [39].

3.4 Causes de l'inefficacité de certains processus métiers

Comme les processus sont entremêlés, un processus mineur déficient peut par cascade induire de grandes inefficacités et donc devenir coûteux pour l'entreprise. Il y a bien sûr de profondes raisons inhérentes à l'entreprise qui expliquent pourquoi il est difficile de tenir à jour les processus métiers et les faires évoluer avec les demandes du marché.

La plupart des points sensibles dans les processus métiers se trouvent à l'intersection entre les personnes, les systèmes informatiques et les processus eux-mêmes. Les causes d'inefficacités principales sont :

a) Rupture dans les processus

Des processus métiers élaborés passent par plusieurs applications qui sont rarement reliées ou intégrées. Les passages qui se font par les employés entre les applications sont souvent manuellement. Ces manipulations sont bien entendu peu efficaces et surtout induisent de nombreuses erreurs, que ce soit lors de la collecte d'informations depuis un système, dans le traitement manuel de l'information ou dans la retranscription de ces informations traitées dans un autre système.

b) Changements de l'environnement commercial

L'évolution ou le changement radical des manières de faire du commerce entre l'entreprise et ses partenaires conduit à devoir ajouter de nouveaux processus métiers ou de prendre en compte de nouvelles exceptions.

c) Gestion des exceptions dans les processus métiers

La plupart des applications automatisent les parties des processus métiers qui se déroulent normalement mais, ne sont pas du tout prévues pour gérer les exceptions qui peuvent se produire. Celui-ci se retrouve la plupart du temps à devoir résoudre manuellement l'exception ce qui passe souvent par une étape de recherche manuelle des informations nécessaires [39].

d) Processus non automatisés ou non optimisés

Même si nous pouvons automatiser la plus part des étapes à l'intérieur du processus métier, en trouve souvent des étapes ne passent pas par les applications de l'entreprise. Par exemple, même si une grande partie du processus de commande se fait de manière électronique, la commande finale et souvent transmise par fax avant d'être introduite dans le système (au lieu de prévoir un système de commande électronique complet pour des acheteurs authentifiés).

3.5 Différents paradigmes d'intégration

Plusieurs solutions précèdent le BPM, les plus récentes parmi celles sont résumées dans cette section, en montrant leurs limitations pour répondre aux besoins et aux objectifs des entreprises.

3.5.1 Le Workflow

Le Workflow peut être défini comme « l'automatisation de processus métiers par échange de documents, informations et tâches entre acteurs pour action » [42]. Le Workflow a pour objectif la coordination automatisée de tâches réalisées par des intervenants humains. Cette approche pragmatique a l'avantage d'efficacité, les concepts sont clairs, les outils relativement aisés à mettre en place, par contre :

- Les participants au processus sont par définition des utilisateurs humains, on ne tient pas compte des applications du système d'information. L'intégration du Workflow aux systèmes est une tâche difficile qui nécessite beaucoup de code propriétaire [19] ;
- Les documents et les tâches ne sont pas suffisants pour l'automatisation des processus métiers. Il est nécessaire d'avoir un niveau d'abstraction supplémentaire, où l'on parle plus généralement de services, et d'informations [19] ;

3.5.2 L'EAI – Enterprise Application Integration

A l'origine, la solution EAI avait pour vocation la collaboration des applications d'une entreprise pour accomplir des objectifs métiers, mais dans les faits leur utilisation est beaucoup plus technique.

Les fonctions principales de l'EAI sont [29]:

- Connectivité : fournir les interfaces d'accès aux applications, généralement par l'utilisation de connecteurs propriétaires difficilement maintenables ;

- Transformation : fournir les services de transformation de données permettant de créer un niveau d'abstraction au dessus des applications du SI, un format pivot pour représenter les données du SI (factures, bons de commande, etc.), et des transformations pour les mapper vers les formats propriétaires attendus par les applications ;
- Routage : fournir les services permettant de localiser dynamiquement le destinataire d'un message en fonction de son contexte.

L'EAI a eu l'avantage d'apporter une réponse au souci de réutilisabilité des entreprises, en leur permettant de capitaliser sur les applications existantes. En outre l'EAI constituer un support efficace au déroulement des processus de l'entreprise en résolvant les problèmes de cohérence entre les systèmes qui interagissent. Mais malheureusement cela ne suffit pas car [39]:

- Les fonctionnalités de gestion des processus métiers des outils d'EAI sont généralement complexes à utiliser et dissociées de l'offre technique d'intégration ;
- Les processus sont définis à un niveau technique, et les décideurs n'ont aucune visibilité sur leur SI ;
- La plupart des outils d'EAI ne possèdent pas de fonctionnalités de Workflow qui permettent de prendre en compte les interactions avec les utilisateurs.

3.5.3 B2Bi – Business to Business integration

Les outils de B2Bi visent à définir les processus de collaboration entre entreprises partenaires [19]. Le résultat est que ces outils fournissent surtout un moyen technique de surveiller une collaboration avec un partenaire : gestion de la sécurité des échanges, fiabilité du transport, etc. Cela pose un certain nombre de problèmes [39]:

- Ces outils doivent pouvoir collaborer avec les outils d'EAI, les processus B2B se déclinent en un processus interne par participant de la collaboration ;
- Pourquoi avoir deux outils différents pour les processus internes et externalisés ? Ces deux types de processus participent en réalité au même processus métier.
- Cependant, l'intégration dans le cadre du B2B pose plus de problèmes à cause de la nature dynamique de l'environnement externe. En effet, les interactions intra-entreprise sont plutôt de nature statique contrairement aux coopérations inter-entreprises qui évoluent dans un environnement dynamique ;

3.5.4 Discussion

Touts ces outils sont indispensables pour automatiser les processus métiers de l'entreprise. Par contre, ces solutions doivent obligatoirement être couplées à d'autres outils, permettant de gérer les processus [19].

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'évolutivité des processus métiers via un seul outil agrégeant les différentes visions.

4. Impact de l'Internet sur le BPM

Un des grands catalyseurs des changements dans une entreprise c'est Internet. D'un côté, il apporte le premier model de travail possible pour une entreprise étendue, permettant de construire des organisations basées sur la collaboration. D'un autre côté, il a réduit les durées et augmenté l'efficacité. A mesure que l'on s'approche de marchés où l'instantanéité devient la règle, les entreprises semblent prendre peur de la pression et de la vitesse, peur de mettre en péril leur intégrité. La raison est simple, la plupart des entreprises n'ont pas été capable d'accélérer leur processus métiers internes pour répondre à temps à la complexité présentée dans les liens externes avec les clients et les partenaires commerciaux [39].

Ces dernières années ont vu l'avènement de la standardisation dans le monde informatique. Le temps où chaque éditeur proposait une solution monolithique, totalement intégrée et propriétaire est révolu. L'exemple le plus récent est celui des services Web : les éditeurs proposant des solutions concurrentes coopèrent désormais pour la définition des spécifications permettant à leurs outils de communiquer facilement. Le BPM n'échappe pas à cette règle, cependant il devient difficile de traiter le BPM sans parler des services Web. Ces derniers trouvent aux entreprises la justification de leurs coûts dans les connecteurs propriétaires qu'ils proposent. Le succès des services Web est grâce aux protocoles standards qui sont à la base du standard XML. Alors grâce aux services Web les entreprises peuvent suivre facilement l'évolution du marché en intégrant facilement leurs processus métiers avec les autres dans le monde, en proposant d'autres services par une intégration et une collaboration inter-entreprise. Nous allons détailler la notion de services Web et leurs impacts sur les processus métiers dans le chapitre 3.

5. Evolution des méthodologies de conception des processus métiers

Dans cette section nous allons présenter l'évolution des méthodologies de conception des processus métiers, tout en démontrant l'approche de conception classique la plus commune, et une approche transactionnelle pour la composition fiable de services Web.

5.1 Approche Classique

Dans cette partie, nous présentons la méthodologie de conception classique des processus métiers qui est la plus commune des méthodologies de conception de processus métiers. Quatre étapes successives illustrées dans le schéma de la figure 5 composent cette méthodologie [22]: la planification des besoins, la construction du modèle du processus métier, l'intégration et finalement la vérification du modèle conçu.

A. Différentes phases de l'approche classique

La phase d'analyse et la planification des besoins constituent la première étape qui consiste à analyser la situation actuelle, et en particulier les besoins et le produit du processus métier à concevoir afin de définir les objectifs que le modèle doit atteindre. Les clients fournissent cette entrée qui est reprise par les concepteurs sous la forme de spécifications. Ces objectifs peuvent être quantifiés pour être repris par la suite dans la dernière phase comme outils de mesure pour la vérification des performances du modèle conçu.

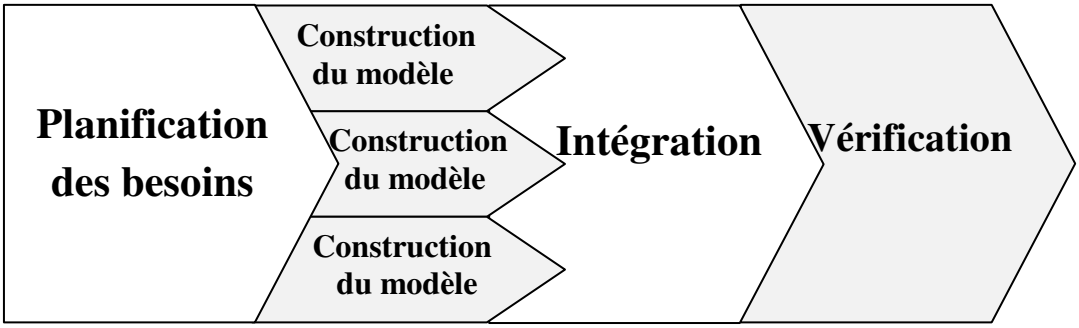


Fig. 5 Approche classique de conception des processus métiers [22]

En se basant sur les spécifications de la phase d'analyse et la planification des besoins, la deuxième étape de construction du modèle. Pendant cette étape de construction, les différentes spécifications du processus sont mises noir sur blanc sous la forme d'un modèle de processus métier en utilisant l'outil d'édition du système de gestion des processus métiers. Notons qu'on peut être amené à diviser l'étape de construction en plusieurs étapes parallèles. En effet, le processus est souvent trop grand pour être conçu dans sa totalité. Par conséquent, il peut être utile de développer le modèle dans un certain nombre d'étapes séparées. Chaque étape se termine avec la livraison d'un nouveau schéma de processus qui est une amélioration/extension du précédent.

Ces différentes versions sont par la suite intégrées dans la troisième étape d'intégration. Cette étape peut se faire dans le cadre d'une collaboration étroite entre les concepteurs du modèle d'une part, et ses utilisateurs d'autre part, sous la forme d'un développement commun.

Avant que le modèle conçu soit mis en pratique, il est indispensable de vérifier, d'une part, que le modèle conçu ne contient pas d'erreurs conceptuelles ou d'erreurs sémantiques et de s'assurer, d'autre part, que l'ensemble des objectifs définis initialement est atteint dans la pratique. La dernière étape de vérification analyse du modèle conçu et s'assure qu'il retranscrit fidèlement et sans erreurs les spécifications de la première phase. Cette étape peut inclure des techniques de vérification et d'analyse pour parfaire un diagnostic qui peut détecter des problèmes de correspondance conceptuelle. Ces techniques permettent en particulier d'expérimenter, de mesurer et d'évaluer les performances en utilisant des métriques prédéfinies pour assurer que l'ensemble des objectifs définis initialement est atteint dans la pratique.

Du point de vue du développement de système, il est important de valider une conception de processus avant son exécution. Il est bien connu que les erreurs de conception qui sont trouvées tard dans le projet sont très coûteuses à corriger [22].

B. Discussion sur les approches classiques

Cependant ces techniques manquent de mécanismes de correction permettant d'optimiser la structure du modèle conçu. En plus ces techniques se caractérisent par une approche statique, du fait qu'elles analysent un modèle conçu figé ou rigide et ne traitent pas de l'évolution ou du changement dû à l'utilisation future du processus [22]. En outre ils ne permettent pas d'atteindre des modèles efficaces répondant à des critères bien définis par les concepteurs. Ces critères décrivent la manière d'agir contre les échecs d'exécution de certaines activités composants les processus métiers. Cependant notre travail dans ce

mémoire permet d'enrichir les méthodologies classiques, tout en proposant un modèle efficace qui répond aux exigences des développeurs.

5.2 Approche transactionnelle

Dans les travaux dans le domaine du développement des processus métiers nous distinguons l'approche proposée dans [14], où le problème aborder présente des similitudes avec notre problématique et qui nous ont été sources d'inspiration. L'objectif de son travail est de proposé une approche transactionnelle pour la composition fiable de services Web. Noter que la composition de services Web partage plusieurs points communs avec la gestion de processus métiers. L'objectif majeur de l'étude est de permettre au concepteur de décrire des compositions fiables en indiquant : la structure globale de la transaction et les états de terminaison corrects. Cependant, il a enrichit la description des services Web avec des propriétés transactionnelles pour mieux exprimer leurs comportements. Ensuite il a développé un modèle de composition de services Web. Ce modèle et qui étend et fusionne les systèmes de Workflow et les modèles transactionnels avancés (MTA). Les MTA visent à assurer un certain niveau de correction des exécutions d'un ensemble d'opérations encapsulées dans une même unité de traitement appelée transaction. Tandis que les systèmes de Workflow s'intéressent plutôt à l'aspect coordination et organisationnel des processus métiers. Et ils ont ignoré les problèmes liés à la fiabilité des exécutions en cas d'échecs. Dans le contexte des processus métiers, les échecs des activités ont été résolus de façon ad-hoc, ou cas par cas et éventuellement via des interventions humaines [14]. C'est pourquoi, les systèmes de Workflows ont été toujours critiqués à cause de ce manque de fiabilité. Dans le modèle proposé par [14], un service composé décrit à la fois un aspect de coordination et un aspect transactionnel. D'une part il peut être considéré comme un Workflow de services. D'autre part, il peut être considéré comme une transaction structurée où les services composants sont des sous-transactions et les interactions sont des dépendances transactionnelles. Ce modèle permet de combiner la flexibilité des Workflows et la fiabilité des modèles transactionnels.

Le problème traité dans le travail de [14] partage certaines similarités avec notre problématique. En effet nous allons traiter les problèmes d'exécutions des processus métiers, en fiabilisant leurs déroulement. Nous allons proposer dans ce travail un modèle efficace de conception des processus métiers basé sur la réutilisation du pattern Observer tiré du catalogue du GOF [7]. Ce dernier représente des modèles de conception efficaces pour résoudre des problèmes spécifiques réalisés par des experts, et favorisent leurs réutilisation. En effet, une approche de réutilisation simplifie le développement et la maintenance à la fois.

6. Conclusion

Nous avons présenté dans ce chapitre la notion du processus métier et son impact sur les entreprises. Les processus métiers sont le sang de n'importe quelle organisation. C'est la visibilité et l'efficacité de ces processus qui permettent aux entreprises d'atteindre et excéder leurs buts et rester en concurrence. Nous avons présenté d'une manière non exhaustive une variété des langages de modélisation des processus métier. Cette dernière représente une étape essentielle dans un processus de développement d'un BP. La deuxième section dans ce chapitre est consacrée à l'illustration du concept BPM, est une solution récente aux entreprises pour décrire, automatiser, gérer et améliorer leurs

processus métiers. Ensuite ce chapitre a présenté l'intérêt des de l'Internet pour l'intégration inter-entreprise. Aujourd'hui l'Internet devient un support pour la publication des processus métiers et la communication inter-entreprise. Néanmoins, cette utilisation accrue d'Internet ne se fait pas sans heurts. Plusieurs problèmes restent à résoudre afin de faciliter l'utilisation de l'Internet et de profiter pleinement des bénéfices du commerce électronique. Considérons les difficultés associées aux échanges des informations de différent format, la standardisation est devient indispensable dans ce contexte. Les services Web viennent pour surmonter ces problème, grâce aux eux, il devient possible de communiqué avec n'importe qu'elle plateforme. Pour cela le chapitre 3 présente les différentes technologies utilisées dans les services Web et leurs impacts sur les processus métiers. Finalement nous avons présenté l'évolution des méthodologies pour la conception des processus métiers.

Notre objectif dans ce travail est proposé une méthode de conception des processus métiers, tout en répondant aux problèmes qui concernent les échecs d'exécution des processus métiers. Notre méthode basée sur l'approche de réutilisation des composants conceptuels. Cette approche permet aux développeurs de construire des modèles efficaces pour construire des logiciels de qualité avec un faible coût. Nous nous essayons dans le chapitre suivant de présenter les grands axes de cette approche.

Chapitre II - Mécanismes de réutilisation dans l'ingénierie des logiciels

1. Introduction

Le but principal de l'ingénierie logiciel est la qualité logicielle. Grâce à l'encapsulation, le polymorphisme, l'héritage et la délégation, l'orientation objet fournit un support important pour améliorer la qualité logicielle. Cette dernière n'est pas une idée aussi simple, elle est vue comme une notion à multiples facettes, décrite par un ensemble de facteurs ou de critères. Les facteurs clés pouvant influés la qualité logicielle sont la compréhensibilité et la réutilisabilité [2].

La compréhensibilité consiste à comprendre aisément un module logiciel indépendamment du reste du logiciel. La réutilisabilité consiste à produire des modules logiciels cohérents, autosuffisants pouvant être réutilisés à tout moment sans avoir à les recréer à zéro (from scratch).

Une bonne modularisation, lisibilité et compréhension des logiciels, la possibilité de réutiliser ceux-ci, de les faire évoluer facilement et d'une manière fiable, sont des objectifs très recherchés en génie logiciel.

Cependant, la complexité croissante des systèmes et leurs évolutions de plus en plus rapide, ont motive un intérêt accru pour une approche de développement plus efficace utilisant des modèles et des méthodes de réutilisation. Cette approche permet de construire un système nouveau à partir des éléments (objets, composants, etc.) existants, éprouvés et réutilisables [1]. Introduite d'abord pour améliorer la productivité des équipes de programmation. Les tendances actuelles exploitent la réutilisation dans toutes les phases dans cycle de développements logiciels : la phase d'expression des besoins, et en particulier lors des phases d'analyse et de conception.

Etant une forme particulière de composants réutilisables, les patrons d'ingénierie constituent en ce sens une des voies les plus pertinentes en matière de réutilisation. Ils répondent, en effet, à ce besoin tout en capitalisant d'importantes connaissances acquises et approuvées par plusieurs développeurs experts. De telles connaissances sont réutilisables des la phase de définition des besoins jusqu'à la phase d'implémentation. Ainsi, une grande variété de patrons a été proposée. D'ailleurs, plusieurs techniques et méthodes ont

été définies pour l'ingénierie de ces patrons et leurs réutilisations dans le développement des logiciels. Nous nous intéressons plus particulièrement, dans le cadre de ce mémoire aux patrons de conception. Ces patrons permettent de réduire la complexité d'un grand nombre de problèmes spécifiques à la conception [1]. Ils présentent des solutions génériques permettant de résoudre certains problèmes de structuration et d'organisation des programmes.

Ce chapitre présente un état de l'art sur la réutilisation et les patrons d'ingénierie en général, et sur les patrons de conception par objets en particulier. La première section définit la réutilisation et présente brièvement le processus de réutilisation. La deuxième section traite des patrons d'ingénierie. La troisième section introduit les patrons de conception en particulier celle du E. Gamma.

2. Notion de réutilisation

La réutilisation a était venue pour surmonter l'infirmité de développeurs pour livrer des logiciels sur mesure, rapidement et à des prix compétitifs [5]. Elle vise trois objectifs essentiels pour répondre à la compétitivité et à la concurrence sur les marchés économiques: diminuer les coûts de développement et de maintenance, réduire le temps de mise sur le marché (time-to-market), et améliorer la qualité du logiciel. La réutilisation se définit comme une nouvelle approche de développement de systèmes, qui propose notamment de construire un système nouveau à partir de composants logiciels sur étagère (COTS, Commercial Off The Shelf) [16]. Elle s'oppose ainsi aux approches traditionnelles de développement, dans lesquelles la construction d'un nouveau système part de rien (from scratch) et nécessite de réinventer de grandes parties d'applications à chaque fois. Les recherches sur la réutilisation ont progressivement évolue ces dernières années, passant de la réutilisation de code à la réutilisation de connaissances et de démarches de raisonnement de différents niveaux. Cette pratique présente un enjeu supplémentaire pour garantir une meilleure qualité de développements [1]. Cette nouvelle tendance est aujourd'hui à l'origine de plusieurs types de composants et diverses méthodes de développement basées sur la ceux-ci.

La réutilisation dans l'ingénierie des systèmes concerne en général deux dimensions : la dimension produit et la dimension processus [1]. La dimension produit concerne les différents types de composants réutilisables. La dimension processus est considérée, d'une part, les différentes activités spécifiques à la production de ces composants, et d'autre part, les activités spécifiques à la mise en œuvre d'une approche de développement par réutilisation de ces composants. Dans ce qui suit nous nous intéressons à la notion de composants réutilisables.

3. Notion du Composant réutilisable

A l'heure actuelle, il n'existe pas de normalisation de la notion de composant réutilisable [16]. La diversité des définitions revient naturellement au domaine d'application et en fonction de la compréhension qui peut être attachée aux composants. Dans le domaine de la réutilisation du logiciel par exemple, un composant est vu comme n'importe quelle entité qui peut être réutilisée. Les concepteurs logiciels voient les composants comme des éléments de conception. Dans d'autres considérations, un composant peut être également vu comme un élément d'un projet, comme une

configuration ou même comme une documentation. Du fait de l'absence d'une définition commune des composants en se réfère à une définition générale qui défini composant réutilisable comme :

« Une unité de conception (de n'importe quel niveau d'abstraction) identifiée par un nom, avec une structure définie et des directives de conception sous la forme de documentation pour supporter sa réutilisation » [3]. La documentation d'un composant illustre le contexte dans lequel il peut être utilisé en spécifiant les contraintes et les autres composants dont il a besoin pour offrir sa solution. Une définition dans [8] a vu le composant comme une solution testée et acceptée pour résoudre un problème fréquemment rencontré lors du développement de systèmes. Toutefois, plusieurs critères permettent de distinguer les très nombreux types de composants proposés aujourd'hui [8], dont nous nous essayons d'illustrer certains de ces critères dans la section suivante.

3.1 Critères de classification de composants

Nous présentons dans cette section un ensemble de critères permettant de caractériser les composants réutilisables. Selon Conte [8], il est possible de caractériser les composants selon un certain nombre de critères, à savoir le type de connaissance, la portée, la couverture, la nature de la solution, la technique de réutilisation, l'ouverture, la granularité et l'architecture.

a)- Types de connaissances

Les composants peuvent capitaliser des connaissances de type produit ou processus. Un composant de type produit est une partie cohérente d'un modèle qui peut être réutilisée avec d'autres composants produit pour assembler un modèle complet [3]. Un produit correspond au but à atteindre lorsqu'on utilise la solution offerte par le composant produit. Ils sont destinés à être réutilisés et intégrés directement dans le système en cours de développement. Les patrons de conception définissaient dans [7], les architectures logicielles et les bibliothèques de fonctions sont des exemples de composants produits. Un composant processus est une partie cohérente d'un processus qui peut être réutilisée avec d'autres composants processus pour assembler un processus complet. Un processus correspond au chemin à parcourir pour atteindre la solution offerte par le composant processus. Par exemple, les patrons d'Ambler [11] sont exemples de composants processus.

b)- Portées

La portée d'un composant détermine la phase du cycle de développement du logiciel concernée par celui-ci. Nous distinguons par exemple les composants d'analyse, de conception ou encore d'implémentation. Les Framework d'architecture et les patrons du [7] sont par exemple des composants de conception. De manière générale, la plupart des composants actuels sont dit « orientés activité » et sont dédies exclusivement à une phase particulière du cycle de développement [1]. Il existe cependant des composants qui concernent plus d'une phase, le catalogue proposé par Buschmann couvre trois phases dans le cycle de développement logiciel [4].

c)- Couvertures

La couverture caractérise le degré de généralité des composants. Certains peuvent être généraux, d'autres plus spécifiques à un domaine ou à une entreprise. C'est le problème traité par le composant qui détermine sa couverture et donc son type. Si le problème est très fréquent dans de nombreux domaines d'application celui-ci est dit général. D'autre part si le problème est spécifique à un domaine particulier, ou à une entreprise particulière alors est un composant de domaine ou d'entreprise. Les Framework spécifiques à la comptabilité sont des exemples de composants de domaine.

d)- Natures de solutions

La solution d'un composant peut être de nature conceptuelle ou logicielle. Les composants logiciels correspondent à la forme la plus ancienne et la plus répandue de composants réutilisables. Ils permettent de réutiliser des fragments de programmes destinés à être intégrés directement dans le code (au niveau de la phase d'implémentation). Les composants CCM [9] et EJB [10] sont des exemples de composants logiciels. Les patrons de conception [7] sont considérés comme des composants conceptuels, ils proposent des petites de structures ou des architectures permettant de spécifier les systèmes.

e)- Technique de réutilisation

Un composant peut être réutilisé par adaptation (imitation), par spécialisation, par composition etc. [3]

f)- Niveaux de granularité

La granularité mesure le plus souvent le nombre d'unités modulaires constituant le composant. Elle spécifie le degré de complexité de sa structure interne ou de l'architecture qu'il propose. Elle peut être faible (moins de 10 entités), moyenne (moins de 100 entités) ou forte (plus de 100 entités). La plupart des composants proposés sont de faible granularité, ils sont généralement définis par quelques entités modulaires atomiques telles que les classes. Les patrons de conception du GoF [7], ou ceux d'analyse de Coad offrent, par exemple, des petites structures de deux à six classes [3].

g)- Ouvertures

L'ouverture d'un composant caractérise son niveau de transparence. Selon que son utilisation nécessite ou non la modification de sa structure interne, il convient de distinguer plusieurs types de composants qui vont de la boite noire (seule l'interface du composant est visible, ou la structure interne est invisible et non modifiable), à la boite blanche (ou la structure interne est visible et modifiable), en passant par la boite en verre (ou la structure interne est visible mais non modifiable) ou la boite grise (ou la structure interne est visible et modifiable par paramètres).

h)- Architectures

L'architecture d'un composant peut être distribuée ou locale.

3.2 Modèle de composants

Un modèle de composants consiste en un ensemble de conventions à respecter dans la construction et l'utilisation des composants [3]. L'objectif de ces conventions est de permettre de définir et de gérer d'une manière uniforme les composants. Elles couvrent toutes les phases du cycle de vie d'un système d'information à base de composants : la conception, l'implantation, l'assemblage, le déploiement et l'exécution. Concrètement, un modèle de composants décrit certains aspects des composants comme la définition de composants à partir d'objets (classes, modules, etc.), les relations entre les composants, les propriétés fonctionnelles et non fonctionnelles de chaque composant, les techniques d'assemblage des composants, le déploiement et l'exécution d'un système d'information à base de composants, etc.

Dans la pratique, un modèle de composants donné est spécifique à une phase du cycle de vie du composant et du système d'information. On trouve ainsi des modèles de composants pour la phase de conception (patrons de conception), et d'autres pour les phases d'implantation et de déploiement (EJB, CCM, etc.).

4. Processus de réutilisation

Pour faciliter et imposer la réutilisation dans l'ensemble du processus de développement des systèmes il est nécessaire d'avoir un cadre technique, méthodologique et organisationnel. Pour ce faire, deux types de processus complémentaires doivent cohabiter : l'ingénierie de SI par réutilisation de composants et l'ingénierie de composants réutilisables (Figure 6) [3]:

4.1 L'ingénierie de SI par réutilisation

Un tel développement nécessite de nouveaux environnements intégrant de manière systématique la réutilisation de composants. Ces nouveaux environnements doivent intégrer des fonctionnalités de recherche, de sélection, d'adaptation et d'intégration de composants pour composer progressivement le système final [3].

4.2 L'ingénierie de composants réutilisables

Une ingénierie de composants est nécessaire afin de créer, d'enrichir puis de maintenir un référentiel de composants réutilisables. Le développement d'un système de réutilisation doit mettre en œuvre les fonctionnalités d'identification, de spécification, de qualification, d'organisation et d'implémentation de composants. Ce développement suit un processus coopératif et itératif demandant la participation et la mise en accord d'experts du domaine. L'ensemble de ces tâches est essentiel, puisqu'il a pour objectif de produire les ressources nécessaires à la mise en œuvre d'une approche d'ingénierie par réutilisation. Toutefois, il n'existe pas d'approches spécifiquement dédiées à l'ensemble de ces tâches [1].

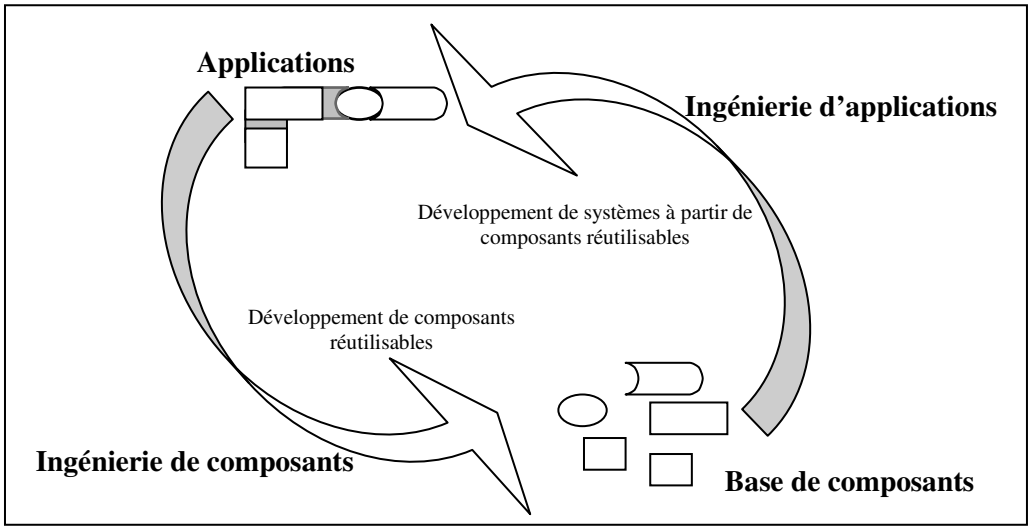


Fig. 6 Deux aspects de la réutilisation [3]

D'après les sections précédentes nous pouvons comprendre la nation des composants réutilisable, et les différents critères de leurs classifications. Dès que notre travail est à la base des patrons de conception cette section est consacrée à la présentation des patrons et en particulier les patrons de conception.

5. Réutilisation des patrons

Les patrons constituent une voie très pertinente dans l'ingénierie des logiciels. Il est désormais essentiel de s'intéresser à leur utilisation, afin d'être en mesure de développer efficacement des systèmes de qualité. Réellement le terme Patron utiliser dans une diversité de domaines, supposant un jeu d'échecs, un patron est un ensemble de mouvements qui peuvent être appliqués pour une stratégie donnée ; et en manufacture, un patron est une forme ou un style d'une pièce à fabriquer enfin, de nombreux autres domaines tels que, l'aviation, linguistique, décoration utilisent la notion de patrons. Dans le domaine de l'informatique, d'après P. Coad un patron est une forme entièrement réalisée, originale ou modèle accepté ou proposé pour une imitation, quelque chose qui est vue comme exemple normatif pouvant être copié ou utilisé [4].

5.1 Définition des patrons

Dans la littérature nous trouvons plusieurs définitions de patrons. On se réfère à une définition plus générale à être introduite dans [5] « un patron a pour but de décrire avec succès des solutions récurrentes à des problèmes logiciels communs dans un certain contexte, et d'aider les gens à réutiliser des pratiques vraies et résolues ». Pour mieux comprendre cette notion de patron, nous étudions d'autres définitions :

« Un Pattern est une solution générale à un problème ou à une issue commun, dont une solution spécifique peut être dérivée »[4] [11].

« Chaque patron décrit un problème qui se manifeste constamment dans notre environnement, et donc décrit le cœur de la solution de ce problème, d'une façon telle que

[4] Signifie en anglais «A pattern is a general solution to a common problem or issue, one from which a specific solution may be derived »

l'on peut réutiliser cette solution des millions de fois, sans jamais le faire deux fois de la même manière » (Alexander, 1979). [7]

« Un Pattern est une idée qui a été utile dans un contexte pratique et sera probablement utile pour d'autres »[5] (Fowler, 1997). [3]

Toutes ces définitions s'accordent sur le fait qu'un patron est un couple formé par un problème et par une solution. Fowler introduit de plus la notion de contexte. Finalement, la définition devient [7]: « Un patron est une solution à un problème dans un contexte ».

5.2 Description d'un patron

Pour pouvoir réutiliser un patron, il faut également exprimer les orientations, les alternatives, et les compromis qui conduisent à l'utiliser. Les exemples concrets sont importants également, car ils permettent de constater le fonctionnement du modèle. L'utilisation d'un format unique pour décrire un patron rendant les patrons plus faciles à apprendre, à comprendre et à utiliser. Un patron est souvent présenté comme « une solution à un problème dans un contexte » [7], dans cette phrase chaque mot à un sens. *Contexte* réfère à un ensemble de situations récurrentes dans lesquelles les patrons sont appliqués. *Problème* exprime un ensemble de forces (buts et contraintes) qui ont lieu dans ce contexte. Enfin, S*olution* réfère à un modèle pour la conception que l'on peut appliquer pour résoudre ces forces. De nombreux formalismes de représentation d'un patron intègrent ces trois aspects et sont globalement équivalents les uns par rapport aux autres [5]. Nous nous limitons à présenter ici trois formalismes. Le premier a été introduit par C. Alexander pour décrire les patrons nécessaires à la construction de bâtiments dans le domaine architectural. Les deux suivants sont ceux que nous considérons comme les plus représentatifs des formalismes utilisés pour représenter des patrons dans le domaine de l'analyse, la conception et l'implantation orientées objets. En particulier, le formalisme utilisé par le Gang of Four s'impose actuellement.

5.2.1 Formalisme de Christopher Alexander

Bien que C. Alexander ait fait état de patrons en matière d'édifices et de ville. En général selon C. Alexander, un patron possède quatre composants essentiels [7].

- Le **nom de patron**

Est un moyen de décrire en un nom ou une phrase courte un problème de conception. Donner un nom à un patron permet de travail à un haut niveau d'abstraction, trouver de bons noms, est une tâche difficile. Par exemple Alcôve, Entrée Principale, Fenêtres Intérieures, etc. ;

- Le **problème**

Une description des forces essentielles du patron et des contraintes sous lesquelles il s'applique, ainsi que les interactions entre ces deux éléments ;

[5] Signifie en anglais «A pattern is an idea that has been useful in one practical context and will probably be useful for others»

- Le **contexte**

Les situations dans lesquelles le patron s'applique ;

- La **solution**

La manière de résoudre le problème, composée de relations statiques et de règles dynamiques décrivant comment construire les artefacts en accord avec le patron. Souvent, des variantes sont proposées ainsi que des manières d'ajuster la solution en fonction des circonstances. Parfois, la solution nécessite l'utilisation d'autres patrons.

5.2.2 Formalisme de P Coad

P. Coad offre 148 stratégies et 31 patrons destiné à guider un concepteur dans la construction de modèles objets effectifs et complets [5]. La description d'un patron nécessite l'inclusion des rubriques données dans la table suivante :

Rubrique		**Signification de la rubrique**
Nom		Nom du patron constitue des noms des différents acteurs composant le patron.
Catégorie		Catégorie du patron précisant le type du patron selon son contexte. Elle est choisie parmi cinq types définie par l'auteur : le patron fondamental, les patrons de transaction, les patrons d'agrégation, les patrons de plan et les patrons d'interaction. Elle peut éventuellement être une nouvelle catégorie.
Gabarit		Représentation graphique type Object Modeling Technique (OMT) montrant les classes composant le patron et les relations entre ces classes.
Interactions		Interactions entre les objets des classes composant le patron.
Exemples d'utilisation		Instanciations possibles des différentes classes composant le patron.
Combinaisons		Combinaisons possibles avec d'autres patrons.
Remarques		Remarques éventuelles complémentaires.

Tableau 1 : **Rubriques du formalisme de représentation de P. Coad [5]**

5.2.3 Formalisme du Gang of Four (GOF)

E. Gamma, R. Helm, R. Johnson et J. Vlissides, regroupés sous le patronyme du Gang of Four, proposent 23 patrons, en général le formalisme du patron GOF possède les éléments suivants [7]:

- Nom de patron

Le nom du patron contient succinctement son principe. Un bon nom est vital, car il va faire partie du vocabulaire du concepteur.

- **Intention**

C'est une courte déclaration qui répond aux questions suivantes : que fait effectivement le patron de conception ? Quelle est sa raison d'être et son but ? Quel cas ou quel problème particulier de conception concerne-t-il ?

- **Alias**

Quelques autres noms reconnus du patron, s'il en existe.

- **Motivation**

C'est un scénario qui illustre un cas de conception, et qui montre comment la structure de classe et d'objet du modèle contribuent à la solution de ce cas. Ce scénario aide à comprendre les descriptions plus abstraites du modèle dans les sections qui suivent.

- **Indication d'utilisation**

Quels sont les cas qui justifient l'utilisation du modèle de conception ? Quelles situations de conception peu satisfaisantes peuvent tirer avantage de l'utilisation du modèle ? Comment reconnaître ces situations ?

- **Structure**

C'est une représentation graphique des classes du modèle, qui utilise une notation issue de la méthode OMT. Nous utilisons également les diagrammes d'interaction pour représenter les séquences de requêtes et la coopération entre objets.

- **Constituants**

Les classes et/ ou les intervenant dans le modèle de conception avec leurs responsabilités.

- **Collaborations**

Comment les constituants collaborent-t-ils pour assumer leurs responsabilités ?

- **Implémentation**

De quels pièges, astuces, ou techniques, faut-t-il être averti lors de l'implémentation du modèle, y a-t-il des solutions typiques du langage utilisé ?

- **Exemples de code**

Ce sont des extraits de programmes, qui illustrent la façon qu'il convient d'employer pour développer le modèle en langage C++ ou Smalltalk.

- **Utilisations remarquables**

Exemples de modèles appartenant à des systèmes existants. Nous présentons au moins deux exemples issus de domaines différents.

- **Patrons apparentés**

Quels patrons de conception sont en relation étroite, avec celui traité ? Quelles sont les différences importantes ? Avec quels autres modèles peut-il être employé ?

5.2.4 Comparaison entre les formalismes

La plupart de ceux-ci sont quasiment équivalents. Ils diffèrent, le plus souvent, par le nombre et le degré de détail (plus ou moins élève) de leurs rubriques ainsi que par le type des patrons qu'ils décrivent. Toutefois, chaque formalisme doit décrire obligatoirement pour chaque patron, le contexte, le problème, et la solution proposée pour celui-ci.

- Le formalisme original proposé par C. Alexander recouvre très bien celui proposé par E. Gamma, puisque chacune des rubriques de C. Alexander y est présente sous une forme

plus détaillée (par exemple, la Solution est détaillée en cinq rubriques dans le formalisme de GOF).

- Le formalisme de P. Coad est assez peu détaillé et il est difficile de retrouver des rubriques de ce formalisme dans chacun des deux autres formalismes, en particulier celui de C. Alexander.

- Enfin, toutes les rubriques proposées par le formalisme d'E. Gamma sont présentes dans l'un ou l'autre des formalismes de C. Alexander et de P. Coad [5].

5.3 Intérêts de l'approche à base de patrons

L'utilisation systématique de patrons facilite la conception d'un modèle en permettant à une démarche de conception de procéder par assemblages et connexions d'instances de patrons [1]. En effet, une approche à base de patrons s'intègre à une méthode de conception orientée objet classique comme OMT et permet en particulier de découvrir les classes d'objets et les associations pertinentes et d'organiser le modèle du système d'information en sous modèles. Le modèle devient ainsi organisé et plus facile à comprendre. La qualité de la conception en est augmentée, les patrons mettant en œuvre des mécanismes éprouvés. Une telle approche est donc prometteuse quel que soit le domaine d'ingénierie et permet en particulier [5]:

- D'archiver et de réutiliser des solutions tant conceptuelles que techniques éprouvées dans un même domaine ou dans des domaines différents. La réutilisation se fait par « bloc » et il est ainsi possible de conserver et de réutiliser des schémas de plus haut niveau que ceux de classes et de types offerts par les modèles objets traditionnels ;

- De communiquer en des termes compréhensibles par les acteurs du domaine. En particulier, on ne parlera plus de classes ou de types, mais de ressources, de contrats, de contrôles, de rôles, etc. ;

- De guider une activité d'ingénierie en organisant hiérarchiquement et fonctionnellement les problèmes et leurs solutions.

Il n'existe pour l'instant pas de règles précises concernant l'utilisation de la notion du patron. Une méthode guidant l'utilisation des patrons pour concevoir des applications est nécessaire.

5.4 Techniques de réutilisation des patrons

Il est désormais essentiel de s'intéresser à l'utilisation des patrons, afin d'être en mesure de développer efficacement des systèmes de qualité. De nos jours, les patrons sont généralement réutilisés de deux manières différentes : Réutilisation directe de patrons et Adaptation ou spécialisation de patrons plus génériques [5]. Nous présentons brièvement, dans ce qui suit, un ensemble de techniques existantes offrant de bonnes potentialités pour la réutilisation des patrons:

A. Réutilisation directe de composants

La réutilisation directe de composants concerne le cas où les besoins cernés pour l'application correspondent exactement à des patrons prédéfinis. Il suffit dans ce cas de trouver le ou les patrons correspondant aux besoins et d'appliquer la solution proposée.

B. Adaptation ou spécialisation de composants plus génériques

Dans ce cas, il n'existe pas de patrons exactement adaptés aux besoins de l'application. Il faut alors adapter ou spécialiser des patrons existants pour qu'ils correspondent aux besoins cernés pour l'application. Dans ce cas, la solution proposée ne correspond pas exactement aux besoins de l'application, et il faut faire aussi l'adaptation (l'imitation est l'équivalant de l'adaptation [1]) ou la spécialisation la solution.

L'imitation d'un patron consiste à dupliquer puis adapter sa solution à un contexte spécifique. Adapter un duplicata d'un patron revient, à renommer, redéfinir, ajouter ou encore supprimer une ou plusieurs propriétés des classes de la solution de ce patron [1].

Pour illustrer cette opération d'imitation, considérons l'exemple tiré de [1] dont, un concepteur désire spécifier, dans le contexte d'un système bancaire, une opération de création d'un nouveau compte. Il est supposé que le concepteur dispose d'un catalogue de patrons, qui contient le patron « nouvelle opération bancaire ». Pour obtenir le modèle résultat, il s'agit tout simplement de réaliser une duplication de la solution de ce patron, suivie d'une adaptation concrète des 3 classes génériques (figure 7).

Cette adaptation consiste, par exemple, à renommer les classes Agrégat-Produit, Produit et Opération-Bancaire qui deviennent respectivement Agence, Client et Compte. Cette adaptation peut se poursuivre par la redéfinition, l'ajout ou la suppression d'autres propriétés. C'est le cas, par exemple, de la méthode verser() ajoutée à la classe Compte, ou encore, le cas de l'argument Compte ajouté à l'opération créer-opération-bancaire().

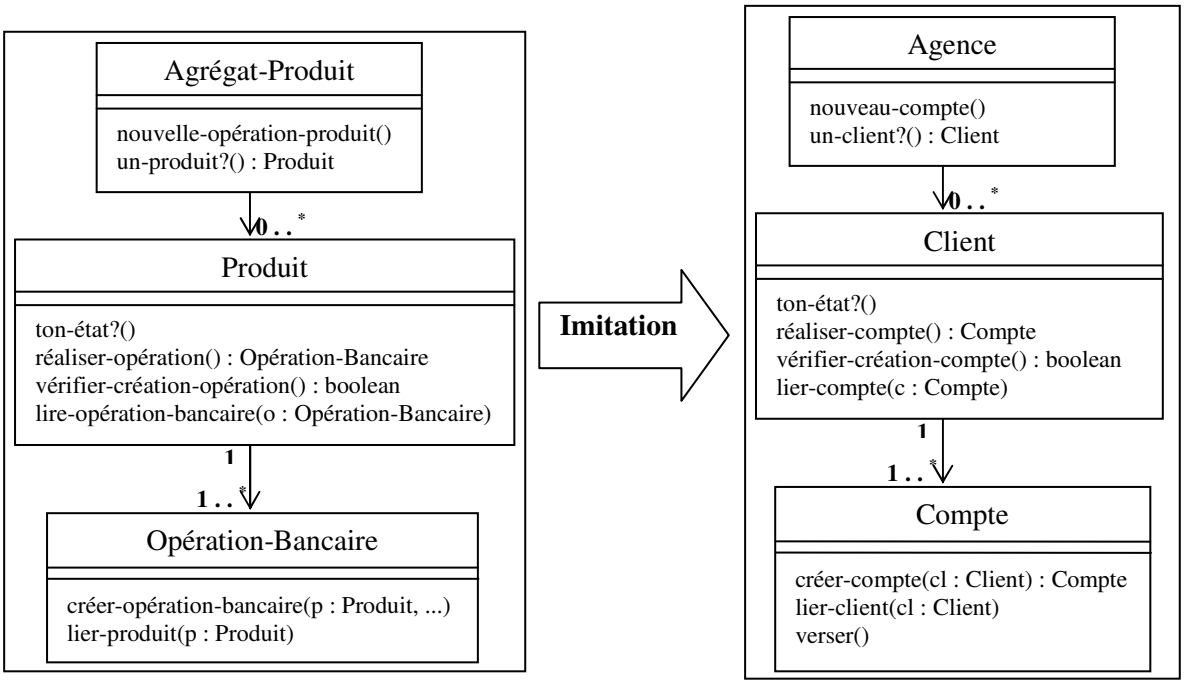


Fig. 7 Exemple d'imitation d'un patron [1]

5.5 Classification des patrons

Trois critères sont particulièrement intéressants pour classifier les composants de type patrons: type de connaissance, couverture et portée (Figure 8). Les autres critères prennent des valeurs uniques caractérisant ainsi ce type de composants. Par exemple, les systèmes de patrons les plus connus, ceux de P. Coad et d'E. Gamma, concernent des *patrons produit généraux* dédiés à une et une seule étape du processus de développement (analyse pour ceux de P. Coad, conception pour ceux de E. Gamma). Les patrons de S.W. Ambler sont par contre des *patrons processus généraux* couvrant l'ensemble du cycle de développement des logiciels. Ils fournissent des collections de techniques, d'actions et/ou de tâches à suivre pour le développement des logiciels. Les patrons proposés par L. Gzara sont spécifiques d'un *domaine*. Ils couvrent les étapes d'analyse et de conception en combinant *patrons produits* et *patrons processus* [8].

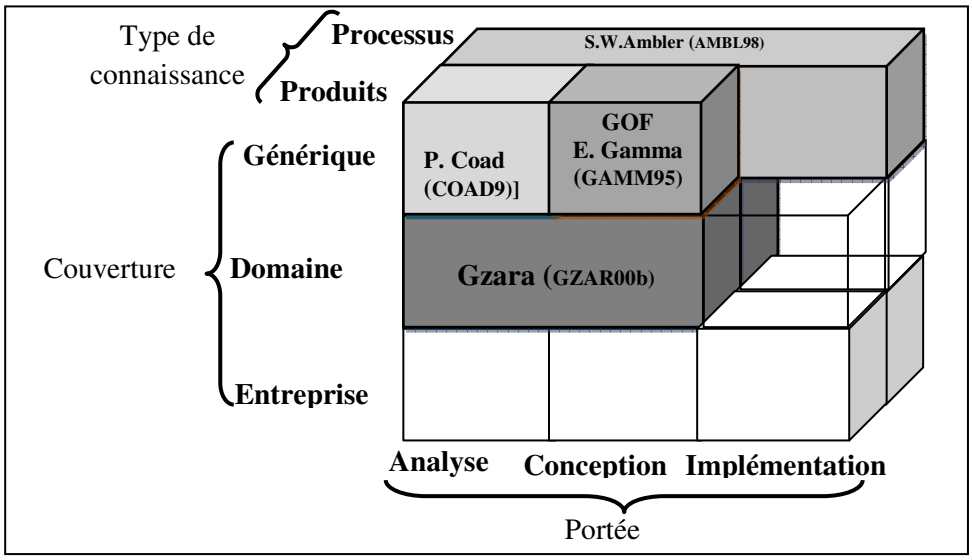


Fig. 8 Différents types de patrons [8]

5.6 Réutilisation de patrons de conception

On s'accorde depuis plusieurs années à reconnaître l'utilité des patrons de conception par objets (Object-Oriented Design Patterns) en tant que solutions génériques à des problèmes récurrents en conception par objets [7]. Les patrons de conception facilitent et accélèrent le développement, en fournissant notamment un langage commun aux différents acteurs de ce développement, et favorisent l'évolution, l'adaptation et la réutilisation.

Les solutions conceptuelles proposées par les patrons de conception sont exprimées, en général, sous la forme de spécifications semi-formelles qu'il s'agit d'adapter pour réutiliser. Ces spécifications décrivent un savoir ou un savoir-faire capitalisant des connaissances réutilisables, ayant des forces et des faiblesses, et précisent les moyens pour adapter ces connaissances. Les patrons de conception par objets présentent plusieurs intérêts et sont largement utilisés dans l'ingénierie des SI, mais également par plusieurs autres types de composants tels que les frameworks et les composants logiciels [1], par exemple. En effet, les frameworks sont les plus souvent conçus à l'aide de patrons de conception par objets, qui facilitent la compréhension de leur conception et améliorent par conséquent leur maintenance et leur évolution. Les composants tels que CORBA (CCM) [9], EJB [10] utilisent également les patrons de conception dans leurs conceptions internes.

Par exemple la définition des propriétés du composant Java Beans suit le patron de conception Accesseur. Le mécanisme de notification d'événements suit le patron de conception Observer [6]. Tans que nous nous intéressons dans ce mémoire aux patrons de conception les plus connus, ceux du GoF [7] alors il est indispensable de présenté un peu de détails sur eux.

5.6.1 Définition des Patrons de conception

Un patron de conception par objets identifie un problème devant être résolu dans un contexte spécifique, et propose une solution possible et approuvée à ce problème. Une définition donnée par E. Gamma [7] pour les patrons de conception est la suivante :

« Un patron de conception donne un nom, isole et identifie les principes fondamentaux d'une structure générale, pour en faire un moyen utile à l'élaboration d'une conception orientée-objets réutilisables » [7]

5.6.2 Patrons de conception du GOF

Les patrons de conception par objets ont été présentés d'une façon uniforme dans un catalogue complet de 23 patrons (notamment par E. Gamma dans [7]) à partir de solutions éprouvées, issues de divers développement, à divers problèmes posés par les limites de la programmation par objets. Dans un but d'intégrer et d'améliorer la réutilisation au niveau de la phase de conception, les auteurs de ce catalogue ne proposaient pas une nouvelle approche de développement. Il s'agit plutôt d'une nouvelle forme complétant les approches classiques de développement à base d'objets. Ce catalogue est désormais convaincant ; la communauté de recherche en génie logiciel a très bien accepté l'ensemble des 23 patrons ainsi proposés [1]. Ils ont été, utilisés et intègres à plusieurs méthodes de développement à base d'objets.

Une classification a été élaborée pour le catalogue de par les auteurs du livre [7]. La classification aide à apprendre plus rapidement les patterns du catalogue, et elle permet également d'orienter les efforts pour trouver des nouveaux patrons [7]. L'organisation se fait selon deux critères. Le premier appelé Rôle, traduit ce que fait le patron. Les patrons peuvent avoir un rôle soit créateur, soit structurel, soit comportemental. Les patrons créateurs concernent le processus de création d'objets. Les patrons structuraux s'occupent de la composition de classes ou d'objets. Les patrons de comportement spécifient les classes et les objets interagir, et se distribué les responsabilités. Le deuxième critère, appelé Domaine, spécifier si le Patron s'applique principalement aux classes ou aux objets. Nous explicitons dans cette section les détails d'un des ces patrons nommé le Pattern Observer. Ce pattern sera intégrer dans notre méthode de développement des processus métier.

5.6.3 Description du Pattern Observer

Le Pattern Observer représente un des patrons de conception les plus utilisés dans l'ingénié des logiciels. Puisque, que notre travail est à la base du Pattern Observer, il est indispensable de faire un survol sur ce Pattern, c'est l'objet de cette section.

1) Intention

Ce pattern est un pattern comportemental. Il définit une dépendance « un à plusieurs » entre les objets de façon à ce que si un objet change d'état, tous les objets dépendants de lui sont mis à jour automatiquement.

Le pattern Observer décrit comment établir ces relations. Les objets principaux dans ce modèle sont un sujet 'Subject' et un observateur 'Observer'. Un Subject peut dépendants de n'importe quel nombre d'Observers. Tous les Observers ont annoncés chaque fois que le Subject subit un changement d'état. Dans la réponse, chaque Observer synchronisent son état avec l'état du Subject.

Ce patron est l'un des plus connus et des plus utilisés. Un observateur maintient une référence vers le Subject observé, les attributs nécessaires à la représentation de l'état de ce Subject et l'opération de mise a jour de cette représentation. Un Subject maintient des références vers plusieurs Observers, et assure la responsabilité de notifier à ceux-ci tout changement de son état.

2) Structure

La figure 9 illustre la structure de ce Pattern utilisant le diagramme de classe d'UML

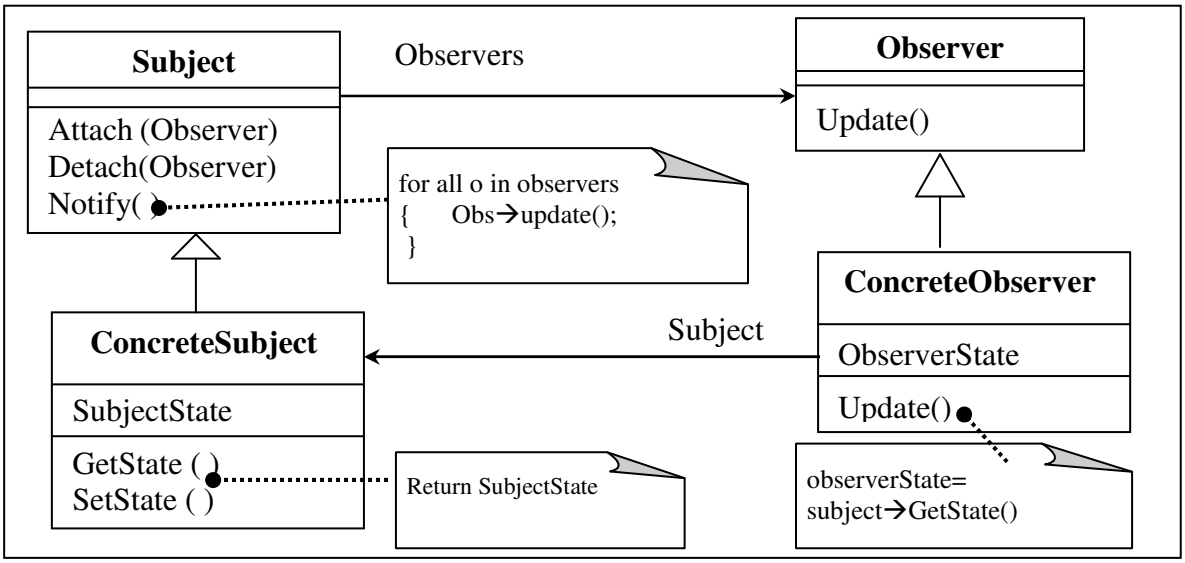


Fig. 9 La structure du Pattern Observer [7]

3) Les participants dans la structure.

- **Subject**

Connait ses observateurs. Un nombre quelconque d'observers peut observer un Subject. Il fournit une interface pour attacher et détacher les objets observers.

- **Observer**

Définit une interface de mise à jour pour les objets qui doivent être notifies de changements dans un Subject.

- **ConcreteSubject**

Mémorise les états qui intéressent les objets ConcreteObserver. Il envoie une notification à ses observers lorsqu'il change d'état.

- **ConcreteObserver**

Gère une référence sur un objet ConcreteSubject et mémorise l'état qui doit rester pertinent pour le Subject. Il fait l'implantation de l'interface de mise à jour de l'Observer pour conserver la cohérence de son état avec le Subject

4) Indications d'utilisation

On utilisera le patron Observer dans les situations suivantes [7]:
- Quand un concept a deux représentations, l'une dépendante de l'autre.
- Quand la modification d'un objet nécessite de modifier les autres, et que l'on ne sait pas combien ils sont.
- Quand un objet doit être capable de faire une notification à d'autres objets sans faire d'hypothèse sur la nature de ces objets.

5) Conséquence

Le Pattern Observer permet de modifier les Subject et les Observers indépendamment. Parmi les avantages le l'utilisation de ce pattern nous citons les suivants :

1. Isoler le couplage entre le Subject et leurs Observers : Le Subject ne connaît aucune classe concrète d'Observers. De ce fait le couplage entre le Subject et les Observers est abstrait et minimal.
2. Support la diffusion : A la notification du Subject est automatiquement diffusée à tous les objets intéressés. Le Subject ne s'occupe pas au nombre des objets intéressés. Ceci donne la liberté s'ajouter ou de retrancher, à tout instant des Observers. C'est à l'Observer de donner suite à une notification ou de l'ignorer

5.6.4 Travaux sur la réutilisation du Pattern Observer

Il existe plusieurs travaux dans la pratique sur la réutilisation du pattern Observer. La première implémentation de ce pattern a été démontrée sur une application orientée objet présenté dans le catalogue du Gang-of-For [7]. En effet, trois d'interfaces graphiques présentent différents aspects (cellules, histogramme, secteurs) pour un ensemble de données (voir la figure 10). Le 'Subject' (sujet) représente l'ensemble de données, et les différents aspects de présentation sont considérés comme des 'Observers' (observateurs).

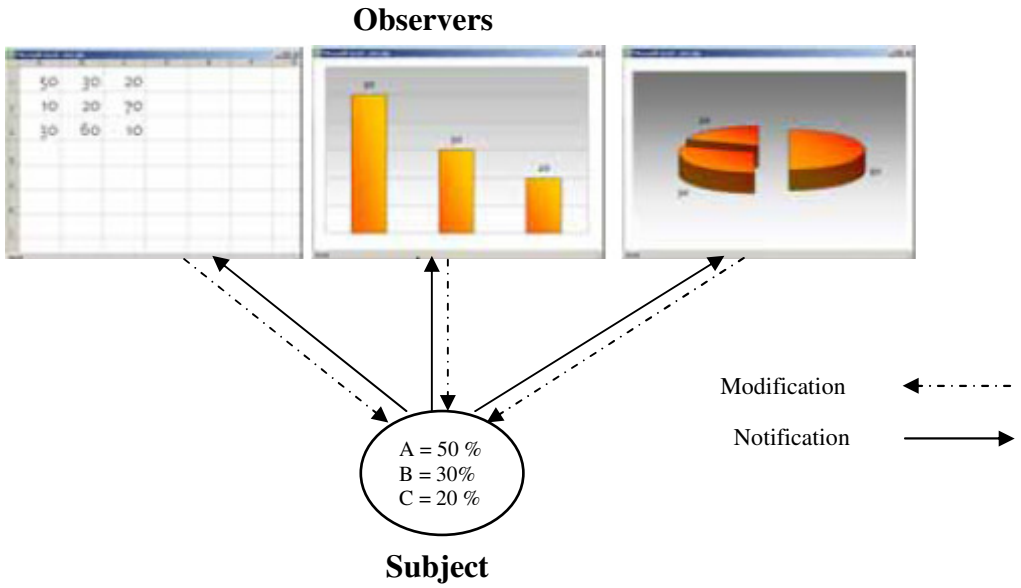


Fig. 10 Exemple d'utilisation du Pattern Observer [7]

Dans cet exemple si un changement subit les données du Subject, ce dernier doit envoyer une notification sur ce changement aux différentes interfaces de présentation. Par conséquents, Chaque interface mis à jour sa forme selon les nouvelles données.

Il existe d'autres situations où le pattern Observer peut être utilisé [48]. En effet, l'annuaire UDDI définit les mécanismes permettant de répertorier des services Web (la section 2.3.4 du chapitre 3 présente les détails d'UDDI). Ce standard régit donc l'information relative à la publication, la découverte et l'utilisation d'un service [31]. Cet annuaire utilise le Pattern Observer pour notifier les clients sur les changements dans leurs structures de données (Business Entity, Business services, Binding Templates ou des tModèles) (voir la figure 11). Sans l'utilisation des événements du Pattern Observer, les clients des données de l'annuaire UDDI doivent de temps en temps examiner l'annuaire pour assurer les changements aux éléments sur lesquels ils se référencent. En effet, la structure originale du pattern sera définie par des interfaces vers les implémentations concrètes des classes ConcreteSubject et ConsreteObserver. Les interfaces doivent être publiées sur l'UDDI. Les clients peuvent alors récupérer le fichier WSDL et construire leur propre implémentation de l'Observer et plus tard, l'enregistrer comme un service Web Subject.

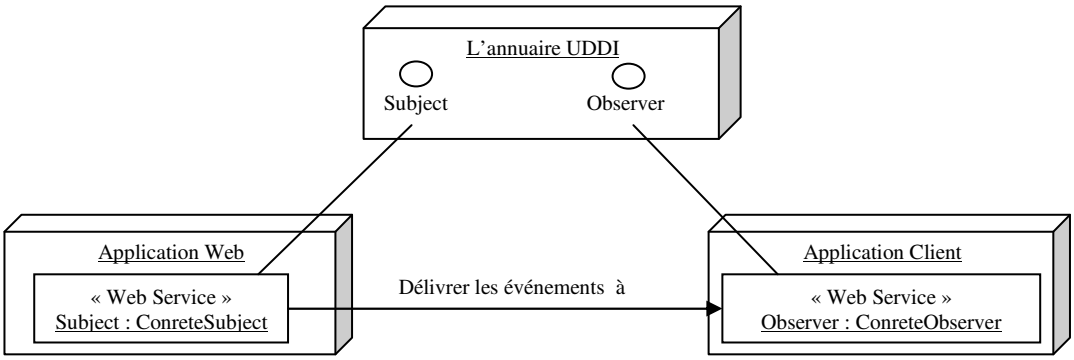


Fig. 11 Scénario de déploiement du Pattern Observer dans les services Web [48]

Ces deux exemples d'utilisation du pattern Observer nous ont données une idée claire sur l'intention du pattern Observer et le contexte de son utilisation. Le deuxième exemple illustre une utilisation du pattern dans le contexte des services Web.

Les processus métiers définis dans notre approche sont réalisés par les services Web. L'utilisation d'une structure basée des interfaces devient une solution adéquate à notre contexte. Nous avons adapté le pattern Observer définit dans [8] aux processus métiers, tout en proposant un modèle réutilisable qui traite les problèmes d'exécution des processus métiers.

6. Conclusion

Ce chapitre nous a permis, dans un premier temps, de montrer l'importance de la réutilisation de composants dans l'ingénierie des systèmes. Nous nous sommes intéressés, ensuite, aux approches de développement de systèmes basées sur la réutilisation de patrons d'ingénierie, une forme particulière de composants réutilisables. Nous avons montré et expliqué l'importance de l'utilisation de telles approches. Enfin, nous avons considéré plus particulièrement les patrons de conception par objets, propos de nos travaux. Les patrons de conception par objets sont des solutions réutilisables au niveau de la conception à des problèmes de conception récurrents, ils sont d'une grande utilité pour structurer la solution. Ils permettent d'améliorer et de pallier certains problèmes de conception pouvant nuire à l'évolution, à la maintenance, ou encore à la réutilisation des applications. Toutefois, ils introduisent à leur tour d'autres problèmes. Nous nous identifions dans le reste de ce mémoire les problèmes liés à l'utilisation du Pattern Observer pour le développement des applications orienté services Web. Ce Pattern sera intégrer dans notre méthode de conception de processus métier avec une adaptation aux services Web. Avant de présenté ce travail nous nous illustrons dans le chapitre suivant peu de détails sur le domaine des services Web qui consiste actuellement la base pour la publication et l'exécution des processus métiers sur le Web.

Chapitre III – Caractéristiques des services Web

1. Introduction

Le système d'information (SI) de l'entreprise est l'un des moyens dont elle dispose pour améliorer ses performances économiques. Il doit être assez souple pour absorber rapidement des nouveaux besoins fonctionnels avec des coûts toujours réduits, tout en accompagnant les évolutions technologiques. Pour respecter leurs objectifs économiques, les entreprises font souvent évoluer leur système d'information en liant les applications entre elles.

Les SI complexes sont donc forcément hétérogènes en termes d'architectures de plates-formes, de bases de données, d'environnements de développement, et ils utilisent des applications répondant à des besoins fonctionnels précis. Les solutions traditionnelles n'abordent pas le problème de l'intégration entre applications que par les données : transferts périodiques de fichiers, partage de base de données, réplication et transformation des données utilisées par les applications … Par suite une évolution vers des solutions d'intégration spécifiques capables de répondre rapidement au besoin d'intégration : les applications se communiquent alors en "point à point" via des interfaces qui doivent être paramétrées et maintenues une à une. Il s'agit d'une architecture spaghetti qui peut répondre rapidement à des exigences d'intégration. Mais elle exige alors des programmeurs maîtrisant les divers protocoles de transmissions, des langages de programmation, les différentes plates-formes de base de données, …

Surmonter les problèmes de cette architecture spaghetti a été plus tard par le développement de la solution EAI (Entreprise Application Integration) [29]. EAI est une solution pour l'intégration des systèmes distribués incompatibles, produisant l'interopérabilité entre les applications disparates dans l'organisation, et améliorent spectaculairement la fiabilité et la flexibilité [49]. Malgré ces avantages les solutions EAI sont fréquemment échouer, où sont tendu à être complexes et chères pour les implémentés parce qu'elles se fondent sur des propriétés technologiques qui sont chère à acquérir, à fonctionner et à maintenir [37]. Une alternative qui semble très prometteuse a récemment vu le jour adressent ces défauts, il s'agit des services Web, ce sont des applications qui relient des programmes, des objets, des bases de données utilisant XML et des protocoles Internet standard [31] et sont considérés comme l'extension de EAI parce qu'ils standardisent la communication, la description et la découverte.

Plus que l'intégration des applications des entreprises les services Web viennent pour surmonter le besoin de partager les processus d'affaires de l'entreprise sur l'Internet et les rendre disponibles aux partenaires. Ce besoin devient très important après l'intégration des applications. Il permet de suivre l'évolution du marché, réduire les coûts et rester en compétition. Alors, avec l'essor des services Web, il devient de plus en plus intéressant pour les entreprises d'encapsulais ses processus métiers en des services Web publiées sur l'Internet et interagir avec d'autre services Web en fournissant des processus métiers inter-entreprise. Tandis que notre travail dans ce mémoire consiste en une méthode pour le développement d'un processus métier basés sur les services Web, il est indispensable de détailler les différents concepts et technologies utiliser pour le développement des services Web, ce qui est l'objectif de ce chapitre. Nous commençons par la compréhension de la notion de service Web.

2. Notion de services Web

Les services Web fournissent un ensemble de protocoles qui permettent aux applications de mettre leurs fonctions et leurs données à la disposition d'autres applications sur Internet. Ce sont pour l'essentiel des composants logiciels présents sur le réseau. Comme les précédents protocoles d'informatique distribuée (comme DCOM, CORBA et Java RMI) les services Web fournissent une ossature pour invoquer un service à distance et échanger des données. Mais en fait, les services Web sont très simples [30]. Ils établissent un langage et une syntaxe indépendante de la plate-forme, pour l'échange de données complexes par le biais de messages. Les éléments internes des services Web sont réalisés à l'aide du langage XML, de sorte qu'il est facile à n'importe quelle plate-forme de prendre en charge cette technologie**.**

2.1 Définition de services Web

Plusieurs acteurs définissent les services Web par des caractéristiques technologiques distinctives, si on se réfère à la définition de [28] « Un service Web est une application logicielle, légèrement couplée, a une interaction dynamique, identifiée par un URI (*Uniform Resource Identifier*), pouvant interagir avec d'autres composantes logicielles et dont les interfaces et les liaisons ont la capacité d'être publiées, localisées et invoquées via XML et l'utilisation des protocoles Internet communs. Ils sont les bases permettant de construire des systèmes distribués et ouverts sur Internet, utilisant des technologies indépendantes des plates-formes » certains éléments principaux nous apparaissent importants:

- Une application logicielle identifiée par un URI:

Un URI est la façon d'identifier un point de contenu sur le Web, que ce soit un fichier texte, audio ou vidéo. L'URI le plus connue est l'adresse d'une page Web, un service Web est donc accessible en spécifiant son URI.

- Capacité des interfaces et liaisons (*bindings*) d'être publiées, localisées et invoquées via XML :

Un service Web peut être publié dans un registre situé à l'intérieur ou à l'extérieur d'un SI (Système d'information). Un service Web peut être localisé en interrogeant le

registre qui l'héberge. Une fois localisé, il peut être invoqué (même par un autre service Web) en envoyant une requête appropriée.

• Capacité d'interagir avec d'autres composantes logicielles via des éléments XML et utilisant des protocoles de l'Internet :

L'une des bases des services Web est l'utilisation de protocoles standards de l'Internet tels que HTTP (*HyperText Transfer Protocol*, le protocole du Web) et SMTP (*Simple Mail Transfer Protocol*, le protocole du courriel électronique) et d'XML. Les services Web peuvent donc traverser les pare-feu conventionnels sans problème.

Contrairement à une page Web ou à une application de bureautique, les services Web ne sont pas destinés à une interaction humaine directe. Ils sont plutôt conçus pour être consommés par d'autres logiciels.

• Composante logicielle légèrement couplée à interaction dynamique :

Par légèrement couplée, on veut dire que le service Web et le programme (le consommateur de service Web) qui l'invoque peuvent être modifiés indépendamment l'un de l'autre. Cela veut aussi dire que, contrairement à une composante logicielle qui serait fortement couplée, il n'est pas nécessaire de connaître la machine, le langage, le système d'exploitation ou tout autre détail, pour qu'une communication puisse avoir lieu. Cela offre une flexibilité qui permet aux entreprises d'éviter les coûts engendrés par l'intégration via des communications fortement couplées.

Par interaction dynamique, on signifie que le consommateur de services Web peut localiser et invoquer ce dernier au moment de l'exécution du programme sans avoir à programmer cette habileté à l'avance. Cela présuppose aussi que l'on peut modifier le service Web sans avoir à modifier le logiciel de chacun des utilisateurs potentiels.

Tous ces éléments permettent de mieux comprendre ce que sont les services Web. Chacun de ces éléments exprime une facette de ce qu'il est maintenant convenu d'appeler services Web.

La définition des services Web ne serait pas complète si l'on n'évoquait pas ses principaux standards. SOAP, WSDL et UDDI sont les technologies dominantes des services Web. Avant de présenter ces technologies il est indispensable de comprendre d'abord les architectures de services Web.

2.2 Architectures de services Web

L'architecture la plus avancée et la plus complète pour un service Web en général est basés sur des caractéristiques courantes du W3C (World Wide Web Consortium) est les spécifications développées par le groupe de travail d'architecture de services Web de W3C [32].

2.2.1 Architecture de base des services Web

Une architecture d'un service Web permet le développement des services Web qui encapsulent tous les niveaux de fonctionnalité d'une entreprise [36]. En d'autres termes, un

service Web peut être très simple, comme un service qui retourne la température courante, ou peut être une application complexe. L'architecture permet également à des multiples services Web d'être combinés pour créer la nouvelle fonctionnalité. Elle a trois rôles distincts: un fournisseur (Provider), un demandeur (Requester), et annuaire de service (Broker). Le fournisseur crée le service Web et le rend disponible aux clients qui veulent l'utiliser. Un demandeur est une application client qui consomme le service Web. Le service Web demandé peut également être un client pour d'autres services Web. L'annuaire, est un enregistrement de service Web, fournit une manière d'interaction entre le fournisseur et le demandeur d'un service Web.

Les trois rôles du fournisseur, demandeur, et l'annuaire de services interagissent l'un avec l'autre par les opérations suivantes : publier (publish), chercher (Find), et lier (Bind).

La figure 12 montre l'ensemble d'interaction entre les trois rôles dans l'architecture de base de services Web. Le fournisseur informe l'annuaire de services de l'existence du service Web, utilisant l'opération 'publier', une interface offert par le fournisseur pour rendre le service accessible aux clients. L'information éditée décrit le service et indique où il se trouve (adresse). Le demandeur consulte l'annuaire pour localiser un service Web par l'opération 'chercher'. Avec les informations sur le service Web obtenue, qui ont enregistrés dans l'annuaire, le demandeur peut se lié au fournisseur en suite il invoque le service visé.

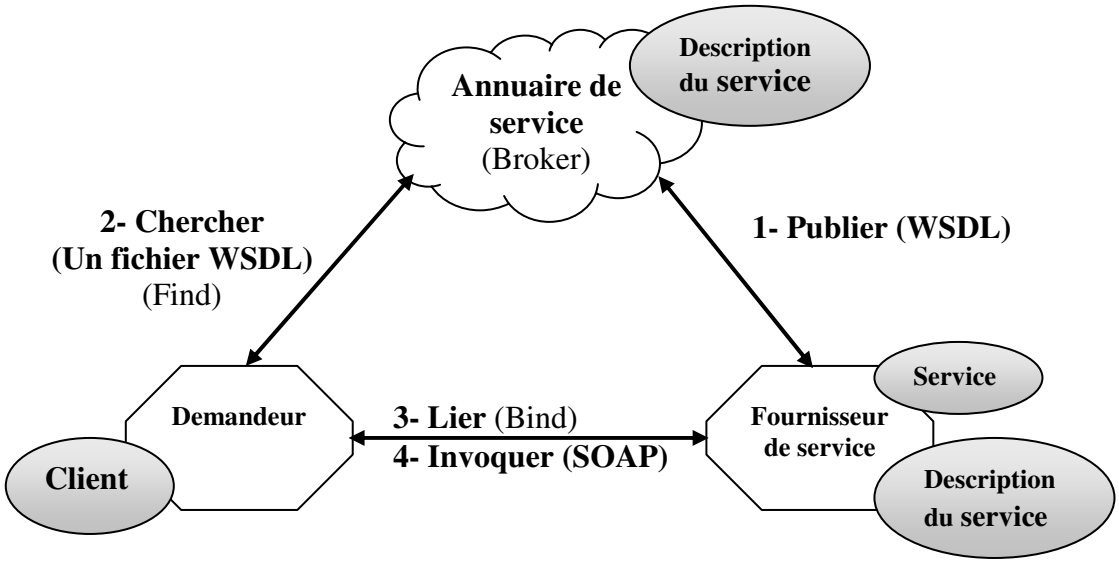


Fig. 12 Architecture de base des services web [32]

2.2.2 Architecture étendue des services Web

Les architectures étendues des services Web impliquent un support des modèles plus compliqués d'échange de message (MEPs, Message Exchange Patterns) qui créent une multicouche de transaction à partir des mécanismes simples d'appel et de réponse [32]. Ces transactions peuvent inclure la sécurité et l'authentification, les événements pour d'autres services Web. Les architectures étendues des services Web profitent également des développements plus récents dans les spécifications du W3C et d'UDDI, telles qu'inclure des attachements dans des documents SOAP pour représenter plusieurs transactions

accumulés ou concaténer durant une transaction multicouche compliquée, authentifiant les messages via l'identification ID d'utilisateur et du mot de passe…

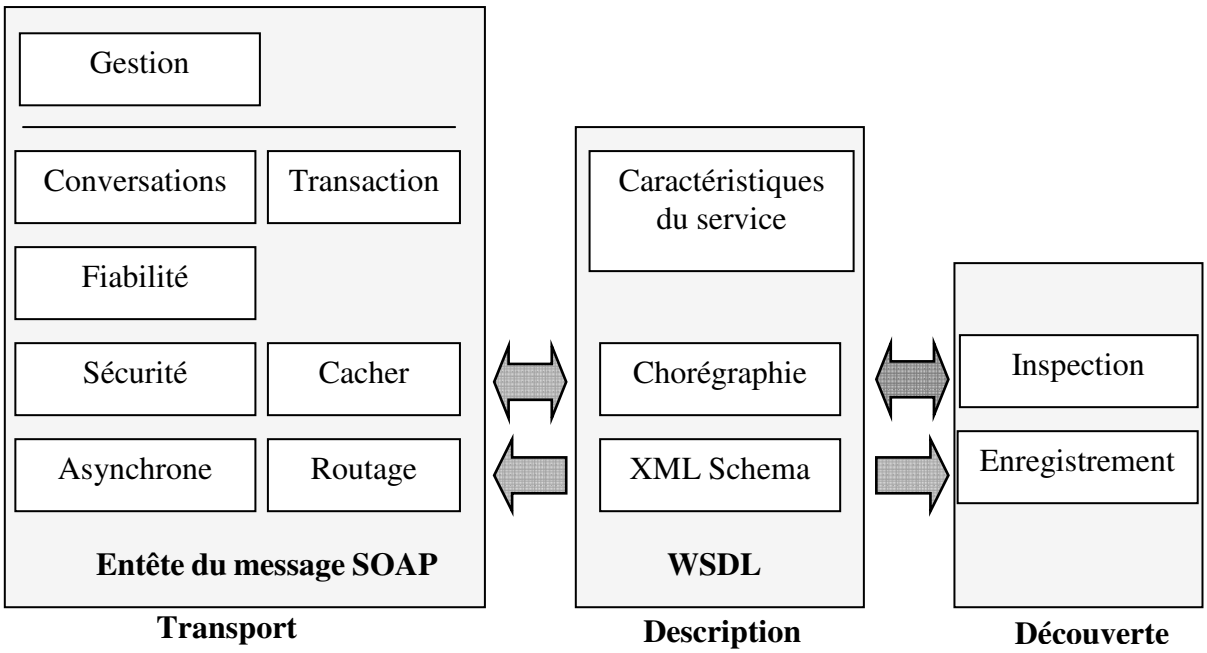


Fig. 13 Architecture étendu des services web [32]

Après avoir présenté les architectures des services Web, il ne reste que de comprendre les technologies utiliser pour implémenter ces architectures. Cela représente le contenu de la section suivante, qui va présenter un aperçu de ces technologies.

2.3 Les technologies des services Web

Fondamentalement, SOAP, WSDL et UDDI sont des technologies issues de l'intérêt parmi des membres de la communauté Internet à développer un mécanisme pour échanger des documents XML sur le Web entre systèmes d'information. Avant de discuter sur ces technologies il est indispensable de faire un survol sur l'XML qui la base de ces technologies.

2.3.1 Un survol sur XML

XML est la base sur laquelle des services Web sont établis. Il fournit la description, le stockage, et le format de transmission pour des données échangées via des services Web. Il est utilisé pour décrire des documents et des données dans un format standard qui peut être facilement transporté par l'intermédiaire des protocoles standard d'Internet. En plus il permet de séparer la structure d'un document, sa présentation et son contenu, de telle sorte ou le contenu d'un document XML peut être présentés avec plusieurs manières, utilisant des feuilles de style différentes.

Il a été développé pour surmonter les limitations de HTML pour l'améliorer [32]. Pour mieux répondre aux besoins des utilisateurs, le World Wide Web Consortium (W3C) a proposé en 1998 l'outil XML qui est capable d'adresser ces besoins. Celui-ci repose sur la norme SGML (Standard Generalized Markup Language) et conserve ces fonctionnalités, l'un de leurs avantages est le fait qu'il est un méta-langage qui permet de construire

d'autres langages propres à des domaines d'activités, XML profite de ce bénéfice est définit d'autres technologies pour des services Web [31]. La syntaxe d'XML est utilisé dans les services Web, indiquant comment les données sont génériquement représentées (WSDL), définit comment et avec quelles qualités de service les données sont transmis (SOAP), et détaille comment les services sont publié et découverts (UDDI). Par conséquent, il est important de connaître comment on peut employer XML lui-même, particulièrement dans le contexte de la façon dont il est utilisé pour définir et mettre en œuvre des services Web.

a) Les éléments et les attributs d' XML

Les documents bien formés de XML peuvent contenir des éléments, des attributs, et le texte [32].

- **Les éléments** : Comme cet exemple, un élément a toujours une balise d'ouverture et autre de fermeture :

  <element></element>

  Le nom de l'élément peut contenir des lettres et des nombres…
  Un élément ne peut pas commencer par un non alphabétique

- **Les attributs** : Contiennent des valeurs qui sont associées à un élément et sont inséré dans la partie du balise ouverte d'un élément :

  <element **attribute** ="value"></element>

- **Les textes** : Est localisé entre deux balises, généralement représente la donnée associe au élément :

  <element attribute ="value"> **Texte** </element>

b) Un espace de nom :

Un espace de nom (Namespace) est une méthode pour séparer et identifier des noms dupliqués d'éléments dans un document XML. Un espace de nom peut également être utilisé comme identifiant pour décrire des types de données et toute autre information [32]. Des déclarations des espaces de noms peuvent être comparées à définir un nom variable court pour une longue variable (telle que pi=3.14159....) dans les langages de programmation.

Afin d'identifier des déclarations des espace de nom contre d'autres types de déclarations d'attribut, le préfixe réservés « xmlns : » est utilisé pour déclarer un nom et une valeur d'un espace de nom. Les espaces de nom sont des composants facultatifs des documents de base d'XML. Cependant, des déclarations des espaces de nom sont recommandées si vos documents XML offrent n'importe quelles possibilités intéressantes actuelles ou futures de partage avec d'autres documents XML qui a les mêmes noms d'élément. En outre, les nouvelles technologies basées XML telles que les XML schémas, le SOAP, et le WSDL se font par l'utilisation lourde des espaces de nom pour identifier les types d'encodage de données et les éléments importants de leur structure. Nous illustrons plus loin dans ce chapitre comment utilisé ces espaces de nom.

Tous ces objets, les attributs, les éléments et les espaces de nom sont arrangés selon certaines règles structurales et syntaxiques pour construire un document XML bien formé.

2.3.2 Le protocole SOAP

SOAP est un protocole de la famille XML servant à l'échange d'informations dans un environnement distribué et décentralisé, indépendant de toute plate-forme. Il est considéré comme la technologie la plus importante des services Web. Le standard SOAP a été proposé au W3C par Microsoft, IBM, DevelopMentor et UserLand [33].

Les messages SOAP sont des documents XML qui contiennent trois éléments composant un message : l'enveloppe, l'entête (Header) et le corps du message (Body) (voir la figure 14). Une enveloppe, qui contient une description du contenu d'un message (mais pas le message), les règles d'encodage pour les types de données spécifiques à l'application, sont placées dans l'en-tête, servent à exprimer et définir le mécanisme de représentation des données. C'est le mécanisme principal par lequel un SOAP peut s'étendre pour inclure des nouveaux caractéristiques et des fonctionnalités additionnelle, telle que la sécurité, les transactions, et d'autres attributs de qualité de service associés au message. Le corps du message permet de transmettre les requêtes et les réponses entre les systèmes [32].

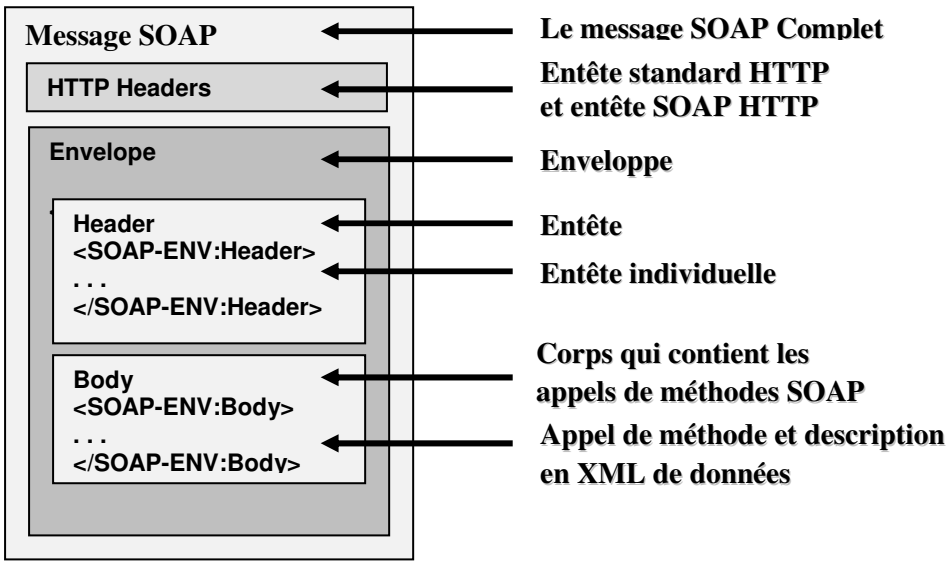


Fig. 14: Structure d'un message SOAP [31][36]

Les messages du SOAP sont envoyés et reçus par des clients SOAP et des serveurs des services Web. Le client SOAP génère et envoie des requêtes SOAP au serveur SOAP via le protocole HTTP. Le serveur SOAP reçoit ces requêtes et génère les réponses appropriées utilisant le protocole HTTP au client (voir figure 15).

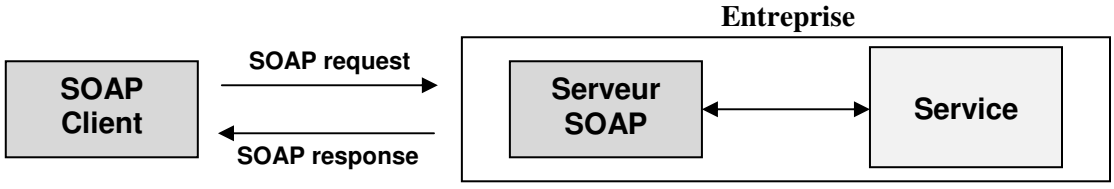


Fig. 15 : Architecture SOAP [36]

L'exemple suivant illustre la structure d'un message simple du SOAP encapsulé dans une requête HTTP [31] :

*// **Entête HTTP** est générer basé sur des valeurs dans le dossier WSDL pour le service Web.*

POST /soap HTTP/1.1

Host: localhost

Content-Type: text/xml; charset="utf-8"

Content-Length: **length**

<?xml version='1.0' ?>

<SOAP: **Envelope** *// **Enveloppe du message***

// Espace de nommage pour l'enveloppe SOAP :

xmlns:SOAP ="http://schemas.xmlsoaporg/soap/envelope/"

// Le type d'encodage du message (entête) :

SOAP:**encodingStyle**="http://schemas.xmlsoap.org/soap/encoding/"

// Espace de nommage pour les types de variables :

xmlns : xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns : xsd=" http://www.w3.org/2001/XMLSchema"

//Espace de nommage pour l'appel des méthodes :

xmlns :tns ="http.//soapinterop.org/">

< SOAP: **Body>** *// **Corps** qui contient le message lui-même :*

// Appel à la fonction getNombreEtudiant qui retourne le nombre d'étudiant d'une class

<getNombreEtudiantRequest>

<nameclass xsi:type="xsd:string"> B </nameclass> *// Paramètre de la fonction*

</getNombreEtudiantRequest>

< SOAP*:* **Body>**

</SOAP: **Envelope>**

La réponse sur ce message sera :

Response header:

HTTP/1.1 200 OK

Content-Type: text/xml; charset="utf-8"

Content-Length: **length**

<?xml version='1.0' ?>

<SOAP: **Envelope**

… // les espaces de nom

< SOAP: **Body>**

<getNombreEtudiantResponse>

<NombreEtudiant xsi:type="xsd:int "> 50 </NombreEtudiant>

</getNombreEtudiantResponse>

< /SOAP: **Body>**

</SOAP: **Envelope>**

Quand une erreur se produit pendant le traitement, la réponse au message SOAP est un élément d'erreur (fault) qui est inséré dans le corps du message, et retourné à l'émetteur.

Le Contenu du corps des messages d'erreur dans SOAP sont les suivants [31]:

<*Faultcode*>: Le nom de l'erreur exercé par émetteur ou récepteur, selon si le message est avéré incorrect à la réception ou une erreur s'est produite pendant le traitement du message.
<*Faultstring*>: Lié au code d'erreur, un texte explicatif qui décrit le problème.
<*Faultactor*>: Un URI identifiant l'adresse du processeur de SOAP qui a produit l'erreur
<*Detail*>: Informations sur l'erreur à l'application spécifiques, nécessaire quand le message ne pourrait pas être traité avec succès.

2.3.3 Le protocole WSDL

WSDL est un langage de description de services Web, utilisant le format XML. WSDL a été développé principalement par Microsoft et IBM et a été soumis à W3C par 25 compagnies. Il décrit les services Web comme un ensemble d'opérations et de messages abstraits. Il a pour but de décrire la fonctionnalité d'un service Web et de spécifier la manière dont d'autres applications peuvent accéder à ce service via une interface [34].

En clair, WSDL définit, de manière abstraite et indépendante au langage de programmation, l'ensemble des opérations et des messages qui peuvent être transmis vers et depuis un service Web donné.

❖ **Les éléments d'un service Web**

Le fichier WSDL décrit l'interface d'un service Web contient les éléments suivants [31]:

- **Types :** Contient les définitions de types décrits par un schéma XML;
- **Message :** Décrit les noms et types d'un ensemble de champs à transmettre (Paramètres d'une invocation, valeur du retour…)
- **Opération :** Correspond à une abstraction décrivant une action implémentée par un service Web

Les Types d'opérations :

1- ***One-way*** : Le point d'entrée reçoit un message (<input>)
2- ***Request-response*** : le point d'entrée reçoit un message (<input>) et retourne un message corrélé (<output>) ou un ou plusieurs messages de faute (<faut>).
3- ***Solicit-response*** : Le point d'entrée envoi un message (<output>) et reçoit un message corrélé (<input>) ou un ou plusieurs messages de faute (<faut>).
4- ***Notification*** : Le point d'entrée envoie un message de notification (<output>)

- **Les types de port (PortType) :** Décrit un ensemble d'opérations, Chaque opération à zéro ou plusieurs messages en entrée, zéro ou un message en sortie ou une faute (exception) ;
- **Liaison (binding) :** Spécifie une liaison d'un <PortType> à un protocole concret (SOAP, HTTP, MIME (Multipurpose Internet Mail Extensions)…). Un PortType peut avoir plusieurs liaisons ;

- **Port :** Définit un point d'entrée (endpoint) comme la combinaison d'une <Liaison> et d'une adresse Internet (réseau) ;
- **Service Web :** Est une collection de points d'entrée.

Donc en résume, le WSDL décrit quatre ensembles des données importants [34]:

- Information de type de donnée pour toutes les requêtes de message et requêtes de réponse,
- Information d'interface décrivant toutes les fonctions disponibles publiquement,
- Information de liaison sur le protocole de transport utilisé,
- Information d'adresse pour localiser le service spécifie.

Ces différentes parties peuvent être développées en tant que documents séparés et puis doivent être combiné ou réutilisé pour former les dossiers complets de WSDL.

❖ **Exemple de spécification du WSDL**

L'exemple suivant figure un service Web qui a pour but de connaître le nombre d'étudiants dans une promotion. Ce service à une méthode unique appelée *getNombreEtudiant* qui prend un paramètre de type "string" (*nomClasse*) et retourne le nombre d'étudiant dans la classe correspondant au paramètre. Cette méthode est déployée en utilisant le protocole SOAP sur HTTP.

Sémantiquement, l'interface de ce service est définie tout simplement comme suivant :

```
Interface
{
   Service EtudiantService
   {
      Operation getNombreEtudiant(entrée nomClasse: string) retour NombreEtudiant: integer;
   }
}
```

Utilisant WSDL, nous avons quatre ensembles principaux présentés en XML comme suit :

- **Type de données et les messages,** Ils sont spécifies dans les éléments "types et messages":

```
<types>
    <!-- Définition des types de données -->
</types>
<message name="getNombreEtudiantRequest">
      <part name="parameters" element="tns:getNombreEtudiantRequest"/>
</message>
<message name="getNombreEtudiantResponse">
      <part name="parameters" element="tns:getNombreEtudiantResponse"/>
</message>
```

- **Interface décrit la fonctionnalité du service :**

```
<portType name="EtudiantServicePortType">
```

```
        <operation name="getNombreEtudiant">
            <input message="tns:getNombreEtudiantRequest"/>
            <output message="tns:getNombreEtudiantResponse"/>
        </operation>
    </portType>
```

- **Liaison sur le protocole de transport.** Dans cet exemple, on utilise le protocole SOAP/HTTP :

```
    <binding name="EtudiantServiceSoap"
            type="tns:EtudiantServicePortType">
        <soap:binding
            transport=http://schemas.xmlsoap.org/soap/http
            style="document"/>
        <operation name="getNombreEtudiant">
            <soap: operation soapAction="urn: #getNombreEtudiant"/>
            <!-- Entrée et sortie de l'operation -->
        </operation>
    </binding>
```

- **Nom du service et de son adresse :**

```
    <service name="EtudiantService">
        <port name="EtudiantServiceSoap"
                    binding="tns:EtudiantServiceSoap">
            <soap: address
                location="http://localhost:8081/WebServiceEtudiantService.asmx"/>
        </port>
    </service>
```

Description complète du service avec WSDL :

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions  xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://new.Webservice.namespace"
    targetNamespace="http://new.Webservice.namespace">

<!-- Définition des types de données -->

  <types>
      <xs:schema elementFormDefault="qualified"
      targetNamespace="http://new.Webservice.namespace">
          <xs:element name="getNombreEtudiantRequest">
              <xs:complexType>
                  <xs:all>
                      <xs:element name="nomClasse" type="xs:string"/>
                  </xs:all>
```

```
            </xs:complexType>
        </xs:element>
        <xs:element name="getNombreEtudiantResponse">
            <xs:complexType>
                <xs:all>
                    <xs:element name="NombreEtudiant" type="xs:int"/>
                </xs:all>
            </xs:complexType>
        </xs:element>
    </xs:schema>
</types>
```

< !-- Le reste de la définition de messages dans la partie précédente -->

2.3.4 Le protocole UDDI

UDDI définit les mécanismes permettant de répertorier des services Web. Ce standard régit donc l'information relative à la publication, la découverte et l'utilisation d'un service Web [28]. En fait, UDDI définit un registre des services Web sous un format XML. Ce registre peut être public, privé ou partagé. Il a pour but de permettre d'automatiser les communications entre fournisseurs et clients. UDDI est une création du trio Microsoft, IBM, et Ariba. L'initiative UDDI doit permettre à un logiciel de reconnaître automatiquement les services dont il a besoin et de s'interfacer avec eux. Une fois un service Web est développé, il faut publier sa description et faire un lien vers elle dans un catalogue UDDI. Les entreprises publient les descriptions de leurs services Web dans cet annuaire sous forme de fichiers WSDL, les clients peuvent ainsi rechercher plus facilement les services Web dont ils ont besoin en interrogeant le registre UDDI. Lorsqu'un client trouve une description de service Web qui lui convient, il télécharge son fichier WSDL depuis le registre UDDI, ensuite à partir des informations inscrites dans le fichier WSDL, notamment la référence vers le service Web, le client peut invoquer les fonctionnalités du service Web [36].

- **La structure d'un annuaire UDDI**

Pour entrer en contact avec les entreprises pour commander quelque chose, vous avez besoin d'une manière de trouver des informations sur ces entreprises : adresse, numéro de téléphone, site Web, ou adresses des services Web. Vous pouvez obtenir ces informations directement d'un représentant d'entreprise peut-être sous forme de carte de visite, de note manuscrite, ou un e-mail. Or vous pouvez également rechercher un nom d'une entreprise dans un annuaire de téléphone et obtenir leur adresse et numéro de téléphone. UDDI est semblable dans le concept à un annuaire de téléphone. Des entreprises peuvent également découvrir des informations sur des services Web spécifiques dans l'enregistrement, pratiquement elles trouvent un URL pour un dossier de WSDL qui fait référence au service Web du fournisseur [31].

Pour cela les informations d'UDDI sont souvent divisées en trois catégories principales d'information d'affaires [31]:

1). **Pages blanches :** Inclut, un nom d'entreprise et l'adresse, information de contact (numéro tel, fax…), site Web, et système de numération universel de données (DUNS, Data Universal Numbering System) ou autre nom d'identifiant.

2). **Page jaune :** Type d'affaires, endroit, et produits, y compris de diverses taxonomies de catégorisation pour l'endroit géographique, type d'industrie, identification d'affaires…

3). **Page verte :** Les informations techniques sur les services, tels que la façon d'agir avec eux, des définitions de processus métier….Un pointeur vers le dossier WSDL du service, le cas échéant serait placé ici.

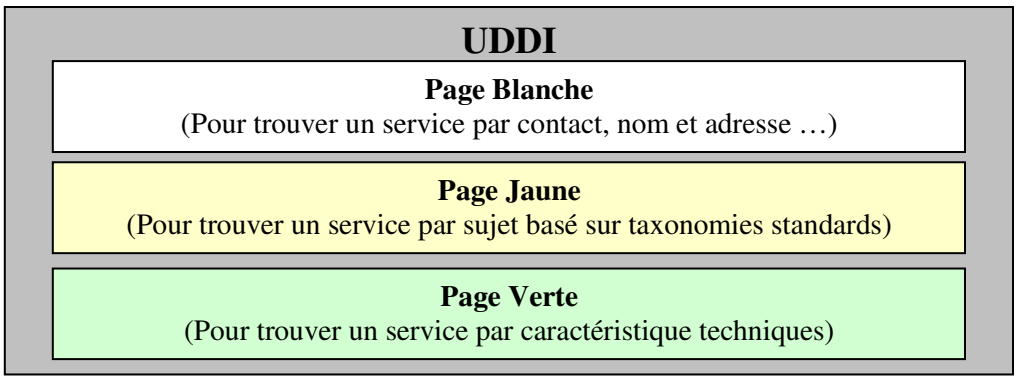


Fig. 16 : Structure UDDI

- **La structuration d'informations UDDI**

Les informations d'UDDI sont décrites dans les structures de données (entités) suivantes [31]:

1. **Business Entity :** une structure de haut niveau, décrit le fournisseur (nom, contact, idantifiant …) ou toute autre entité auxquelles l'information est enregistrée. Les autres structures sont attachées à cette structure par l'intermédiaire des références.
2. **Business services :** Le nom et la description du service publié.
3. **Binding Templates:** L'information sur le service, y compris une adresse d'un point d'entrée pour accéder au service. Elle est attachée à la structure « Business services ».
4. **tModel (Technology Model):** Une collecte d'informations identifiant uniquement les spécifications du service (définition abstrait). Il donne aussi un supporte pour des recherches de haut niveau.
5. **Publisher Assertion :** C'est une structure de relation, met dans une association deux ou plus de structures « BusinessEntity » un type spécifique de relation, tel que « subsidiaire à » ou « département de ».

La figure 17 montre cette structuration d'information dans l'UDDI.

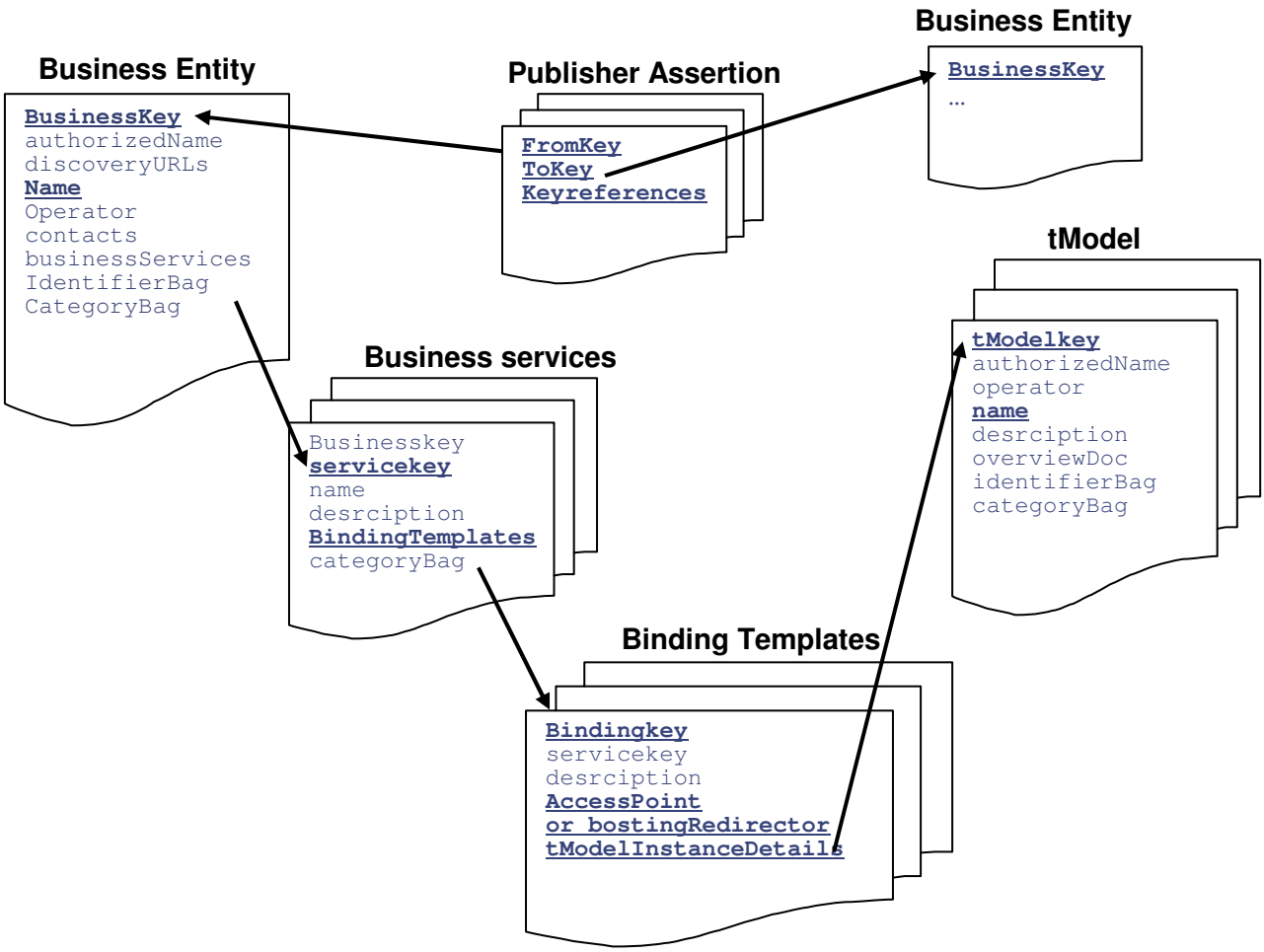


Fig. 17 : Structuration d'information dans l'UDDI [31]

Voici une vue de certains éléments dans une entrée de businessService [32]

```
<businessService serviceKey="..."> // identifiant du service
    <name> SymbolService </name> // nom du service
    <description>Description of the service here</description> // description du service
    <bindingTemplates>
        <bindingTemplate>
            <accessPoint // point d'entrée pour accéder au service.
                urlType="http">
                http://mycompany.com/myservice
            </accessPoint>
            <tModelInstanceDetails>
                <tModelInstanceInfo tModelKey="12345678"/>
            </tModelInstanceDetails>
        </bindingTemplate>
    </bindingTemplates>
</businessService>
```

Un exemple de tModel : [32]

```
<tModel
  authorizedName="..." operator="..."
  tModelKey="UUID:12345678>
```

```
<name>Our special service</name>
<description xml:lang="en">
        WSDL description of our service
</description>
<overviewDoc>
        <description xml:lang="en">WSDL source document.
        </description>
        <overviewURL>     // URL du fichier WSDL
          http://mycompany/ourservice/service1.wsdl
        </overviewURL>
</overviewDoc>
<categoryBag>
        <keyedReference tModelKey="UUID:987654"
        keyName="uddi-org:types" keyValue="wsdlSpec"/>
</categoryBag>

</tModel>
```

Par exemple, pour rechercher une compagnie appelée XYZ dans l'UDDI, une requête SOAP contenant dans le corps de leur message la séquence suivante [32]:

```
<find_business generic="1.0" xmlns="urn:uddi-org:api">
    <categoryBag>
        <keyedReference tModelKey="uuid:70a80f61-77bc-4821-a5e2-2a406acc35dd"
    keyName="Advertising" keyValue="7310" />
    </categoryBag>
</find_business>
```

La réponse en message SOAP qui revient d'UDDI contient toutes les entreprises qui accordent les critères (celles écrites en gras) de recherche et les services enregistrés pour chaque entreprise, la réponse sera :

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <soap:Body>
            <serviceList generic="1.0"
                    operator="Microsoft Corporation"
                    truncated="false" xmlns="urn:uddi-org:api">
            <serviceInfos>
                <serviceInfo serviceKey= "d5b180a0-4342-11d5-bd6c-002035229c64"
                    businessKey="ba744ed0-3aaf-11d5-80dc-002035229c64">
                <name> XMethods Barnes and Noble Quote </name>
            </serviceInfo>
            </serviceInfos>
            </serviceList>
        </soap:Body>
  </soap:Envelope>
```

3. Intégration des services Web dans les Processus métiers

L'approche processus métier n'est pas nouvelle, elle résulte des approche processus métier sont modélisés par des objets métier coopérants. L'OMG (Object Management Group) propose d'ailleurs un standard pour modéliser les processus métier appelé EDOC (Extended Distributed Object Computing) [38]. EDOC spécifie un cadre de modélisation objet basé sur UML et définit une architecture de collaboration. Il est plutôt complémentaire aux approches XML. Cependant, les services Web vont devenir à terme une base solide pour le partage de processus métier gérés par les outils de BPM. Un service Web permet la syndication de processus métier sous la forme de composants qu'il devient possible de partager, acheter ou vendre facilement. Dans ce contexte un processus métier sera abstrait par un document XML décrivant les interactions entre participants. Une composition de services Web s'apparente à la composition de procédures dans les langages de programmation pour réaliser un sous-programme complexe. Chaque service est décrit par un document WSDL qui décrit son interface avec des messages entrant et sortant. La composition de services permet d'agréger un enchaînement de services de façon à fournir un nouveau service, aussi décrit par une interface. L'agrégation peut être simplement une séquence, mais aussi une exécution parallèle ou conditionnelle, ou une alternative. La composition de service permet de définir les activités participant à un processus métier, une activité pouvant être un service simple ou un service composé [38].

4. Composition de services Web

WSDL, UDDI et SOAP constituent les technologies de base pour les interactions entre les services Web. Cependant, elles sont insuffisantes à elles seules pour répondre aux besoins d'intégration des applications inter-entreprise [35]. En effet, ces trois propositions permettent respectivement la description, la publication et l'invocation d'un service Web. Cependant, un processus métier nécessite d'invoquer un ensemble de services Web dans un ordre précis et selon une logique bien définie. Or SOAP, WSDL et UDDI ne s'intéressent pas à ce problème et se situe plutôt au niveau transport et données [27]. A ce propos, le développement d'applications à base de services Web est s'appuie sur la composition de ces derniers. Ce qui a relancé les travaux de recherche sur les problèmes d'interopérabilité et de coordination de services Web, ce qui a donné naissance aux termes : *Orchestration* et *Chorégraphie*, et à la définition de plusieurs langages de coordination de services Web.

4.1 Définition de l'Orchestration et de la Chorégraphie

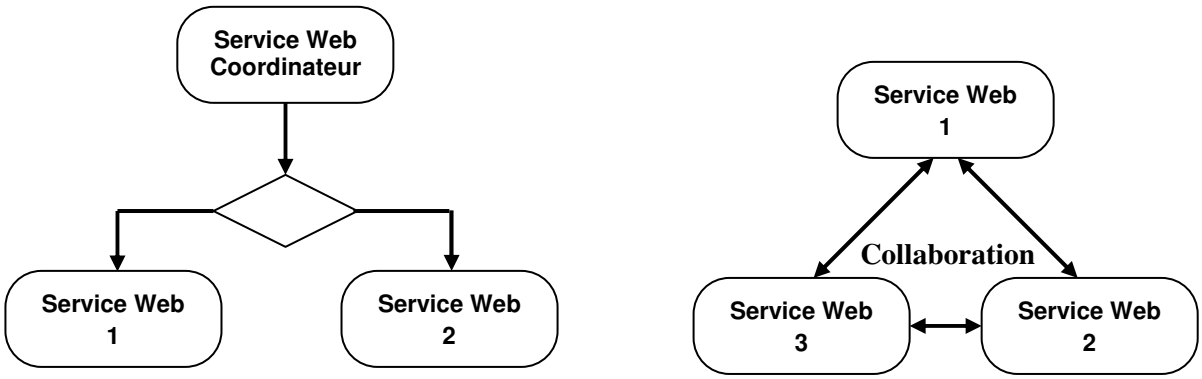
L'**orchestration** de services permet de définir l'enchaînement des services selon un canevas prédéfini, et de les exécuter à travers des "scripts d'orchestration". Ces scripts sont souvent représentés par des processus métier inter/intra-entreprise. Ils décrivent les interactions entre applications en identifiant les messages, et en branchant la logique et les séquences d'invocation (voir figure 18.a) [24].

La **chorégraphie** trace la séquence de messages pouvant impliquer plusieurs parties et plusieurs sources, incluant les clients, les fournisseurs, et les partenaires. La chorégraphie est typiquement associée à l'échange de messages publics entre les services Web, plutôt qu'à un processus métier spécifique exécuté par un seul partenaire (voir la figure 18.b).

Il y a une différence importante entre l'orchestration et la chorégraphie de services Web [35].

L'orchestration se base sur un processus métier exécutable pouvant interagir avec les services Web internes ou externes. L'orchestration offre une vision centralisée, le processus est toujours contrôlé du point de vue d'un des partenaires métier.

La chorégraphie est de nature plus collaborative, chaque participant impliqué dans le processus décrit le rôle qu'il joue dans l'interaction.



a. Orchestration de services Web **b. Chorégraphie de services Web**

Fig. 18 Composition de services Web [24]

Beaucoup de standards se sont intéressés initialement soit à l'orchestration soit à la chorégraphie. La section suivante sera consacrée à la présentation de quelques langages de composition de services Web.

4.2 Langages d'orchestration et de chorégraphie des services Web

De nombreux langages de coordination de services Web sont apparus. Certains de ces langages étaient plutôt centrés sur l'orchestration, d'autres plutôt sur la chorégraphie.

4.2.1 Les langages XLANG, WSFL, et BPEL4WS

Au début de l'année 2001, Microsoft a crée le langage XLANG (XML business process language), il contient la description WSDL, et y ajoute les éléments suivants [35]:

- **Le comportement (*behavior*) :**

Le comportement spécifie l'enchaînement des activités par des balises propres à XLANG. La partie comportement contient :

- Les actions XLANG : operation, delayFor, delayUntil, et raise ;
- Le contrôle XLANG : qui définit l'enchaînement de l'exécution des actions, et qui peut être l'une des commandes : empty, sequence, switch, while, all, et pick.

- **Le contexte (*context*)**

Le contexte spécifie les transactions éventuelles du processus et les références utilisées dans le document. Il permet de définir :

- Des variables locales, constituées de corrélations et de références aux ports.
- Des transactions.
- Des signaux envoyés en cas de défaillance.

- **Le contrat (*contract*)**

Le contrat spécifie les relations entre services XLANG. Il est spécifié dans un fichier à part. Ce fichier importe les fichiers XLANG des services participant à ce contrat, et il précise la manière dont les ports de ces services sont liés.

- En mai 2001, IBM a proposé le langage WSFL (Web Services Flow Language). Ces deux langages ont été ensuite remplacés par la spécification BPEL4WS (Business Process Execution Language for Web Services), habituellement appeler (BPEL) réalisée par IBM, Microsoft et BEA. Il a été positionné comme une extension des processus métier aux standards existants de services Web. Précédemment, les interactions de services Web sont limitées, BPEL et d'autres langages de processus et de chorégraphie montre une conversation de processus métier à travers les services Web. Il définit la coordination des interactions entre l'instance de processus et ses partenaires [26].

Le BPEL4WS permet de modéliser deux types de processus :

- Le processus *abstrait* : spécifie les échanges de messages entre les différentes parties, sans spécifier le comportement interne de chacun d'eux ;

- Le processus *exécutable* : spécifie l'ordre d'exécution des activités constituant le `processus`, des partenaires impliqués dans le `processus`, des messages échangés entre ces partenaires, et le traitement de fautes et d'exceptions spécifiant le comportement dans les cas d'erreurs ou d'exceptions.

4.2.2 Le langage BPML

L'organisation BPMI a développé le langage BPML (Business Process Modeling Language) publié en mars 2001. Est un langage de définition de processus métier (Business Process) en XML. Il décrit la représentation structurale d'un processus et la sémantique de son exécution. En plus il gère la coordination de toute sorte de participants, et non pas de services Web uniquement [35]. Le code d'un processus BPML peut contenir des boucles, des décisions, des chemins parallèles, des variables, et des exceptions, il est facile à comprendre par un programmeur [26].

4.2.3 Le langage WSCI

De leur coté, Sun, SAP, BEA, et Intalio ont proposé le langage WSCI (Web Services Choreography Interface), décrivant les interfaces de services Web, pour réaliser une chorégraphie. Ce langage offre une vision différente de la collaboration, qui prend une forme de conversation entre plusieurs services Web. La collaboration se passe d'une

manière décentralisée, sans utiliser un processus principal, mais plutôt plusieurs processus distribués [35].

4.2.4 Le langage WSCL

Hewlett-Packard a soumis au World Wide Web Consortium (W3C), le 14 mars 2002, le langage de conversation WSCL. Ce langage ne s'occupe que de la description de conversations entre paires de services Web. Il est donc plus léger que WSCI qui décrit des conversations entre plusieurs services Web, et non pas seulement entre deux services Web.

Mais contrairement à WSCI, WSCL utilise une forme de processus pour décrire la conversation, et l'exécution de la conversation n'est pas centralisée [35].

4.3 Discussion sur l'orchestration et la chorégraphie

Les langages présentés sont les plus connus pour la composition de services Web, certain de ces langages ont des similitudes et des différences. Le principe de certains de ces langages, comme BPML, est de construire un processus centralisé, se chargeant de coordonner les services Web. Ces langages-là font donc de l'orchestration.

D'autre langages, comme WSCL il permet de décrit une conversation entrer un service Web d'un participant avec d'autre participant pour atteindre un objectif commercial. Les étapes de conversation appeler interactions, cela sans spécifier le contenue des messages échangées. Cette description de conversation prend une forme de processus, mais son exécution n'est pas centralisée. La collaboration des deux services Web entre donc dans ce cas-là, dans le domaine de la chorégraphie.

Dans des langages comme XLANG et WSCI, chaque service Web connaît son comportement, et la réaction qu'il aura suite à chaque opération et à chaque message arrivant. Dans les deux langages, la collaboration entre services Web se passe d'une manière décentralisée. Ces langages font donc de la chorégraphie.

Il y a aussi des langages, comme BPEL4WS, qui permettent de décrire deux types de collaboration :

La coordination centralisée, lorsqu'il s'agit de décrire l'enchaînement des appels d'opérations de services Web (le "processus exécutable" dans le langage BPEL4WS). Ce qui correspond à faire de l'orchestration ;

Et la collaboration décentralisée, lorsqu'il s'agit de décrire le contrat d'interactions entre deux services Web, ou ce qu'on a appelé "la conversation" (le "processus abstrait" dans BPEL4WS). Ce qui correspond à faire de la chorégraphie.

5. Conclusion

Les services Web permettent d'intégrer, de gérer et d'automatiser plus facilement et plus rapidement les processus métiers intra et inter-entreprise en échangeant des informations au format XML. On peut donc, par ce biais, intégrer dans le système d'information les différentes activités constituant les chaînes de valeur de l'entreprise.

L'assemblage des différents services Web peut être orchestré par un moteur de d'exécution, qui contrôle l'exécution de l'application en fonction du contexte d'exécution et des règles de déroulement du processus métier. Cette manière d'intégrer les applications est beaucoup plus simple et coûte de loin moins cher (en délais et coûts de développement) que le développement classique. Ce dernier nécessitait la programmation d'adaptateurs spécifiques pour intégrer les applications couvrant les divers processus métiers de l'entreprise.

Cependant la composition des services Web pour réaliser des processus métiers enlève le problème d'intégration intra et inter-entreprise. Mais l'intégration seulement reste insuffisante pour traiter d'une manière flexible les problèmes qui s'exposent durant l'exécution de ces processus métiers. Il est important d'avoir des modèles efficace pour traiter les anomalies dans le déroulement des processus métiers. Dans ce travail nous nous essayons de traiter ce problème. Le chapitre suivant démontre notre méthode de conception des processus métiers, dont nous allons proposer un modèle efficace qui repend aux problèmes d'exécution des processus métiers.

Chapitre IV- Conception d'un processus métier basé sur le Pattern Observer

1. Introduction

Ce chapitre consiste en la présentation d'une méthode pour le développement d'un processus métier flexible, et facilement adaptable aux divers changements d'état pouvant surgir durant sa vie. Les méthodes et les techniques de développement d'un processus métier ont atteint une certaine maturité. Cependant ces méthodes restent inadaptées dans de nombreux contextes, car elles ne permettent pas d'atteindre des modèles efficaces répondant à des critères bien définis par les concepteurs. Ces derniers essaient de les respecter plus fidèlement pour augmenter ainsi les performances des processus métiers (c'est-à-dire augmenté la satisfaction des clients et la fiabilité d'exécution des processus métiers). Pour répondre à ces critères et satisfaire les exigences conceptuelles et faire face aux problèmes rencontrés, nous proposons un modèle de conception qui est à la base construit et fondé sur l'utilisation d'un patron de conception nommé le Pattern Observer, dont les détails seront présentés plus loin dans ce chapitre.

2. Conception d'un processus métier

Il n'existait jusqu'à récemment pas de méthode standard permettant le développement d'un processus métier [19]. Les méthodologies classiques de conception des processus métiers sont régies par un ensemble de principes et une philosophie commune décrite par un ensemble d'étapes successives qui sont : la planification des besoins, la construction du modèle du processus métier, l'intégration et finalement la vérification du modèle conçu [22]. Comme nous avons cité précédemment, ces approches ne permettent pas de calquer exactement le déroulement du processus métier comme il est perçu par les concepteurs. Notre travail est de ne pas proposer une nouvelle méthode pour le développement des processus métier. Néanmoins nous allons enrichir les méthodes existantes pour qu'elles permettent de générer des processus performants satisfaisants les objectifs des concepteurs. Généralement pour réaliser un bon BP, nous nous examinons l'environnement du projet: nous devons d'abord comprendre le problème. Le concepteur doit regrouper les différentes informations concernant le processus métier (les fonctionnalités, les contraintes...). Ces informations doivent être définies de façon compréhensible par les utilisateurs et par l'équipe de développement.

L'étape suivante consiste à utiliser ces informations pour objectifs d'identifier les différents processus de l'entreprise. Deux approche sont utilisé pour ceux-ci, l'approche « top-down » en partant du plus générale et en allant vers le plus particulier par raffinement successif, la deuxième approche est celle du « bouttom-up » démarre par la conversation avec les employés de l'entreprise, en identifiant les différentes activités journalières. Cette étape se termine par une définition des différentes activités qui participent dans le processus métier et l'ordre de leurs s'exécutions.

Cependant en passe à l'étape la plus importante qui consiste en la description du processus métier. Nous avons décomposé cette étape en deux phases : la première consiste en la définition du flot de contrôle, en modélisant la succession d'activités qui composent le processus métier, en utilisant des outils de modélisations, nous citons par exemple UML, BPMN…, généralement via une interface graphique. La phase suivante dans cette étape consiste en la spécification des critères des concepteurs, qui concernent les mécanismes de recouvrement et de traitement des problèmes d'exécution des processus métiers. Ces critères sont en langage naturel, qui soient transformés en un document XML structuré selon un DTD sera présenté par la suite. Ces critères doivent être se respecté par le processus métier en cours de son exécution.

L'étape suivante consiste à l'intégration des différents modèles du processus métier en plus le modèle du Pattern Observer qui permet de concrétiser les critères des concepteurs. Ce modèle sera détaillé au plus tard.

Après l'intégration des différents modèles et la vérification de ceux-ci, l'étape l'avant dernières concerne l'implémentation de différentes parties du processus métier, en respectant la conception proposé et enfin le déploiement de la solution. Les utilisateurs exécutent le processus métier et les concepteurs se préoccupent de la surveillance et l'analyse des résultats des exécutions, et ils essaient d'améliorer le processus en cas d'inefficacité, ou de faire des modifications face aux changements du marché économique. La section suivant dans ce chapitre sera consacrée aux détails de notre méthode de conception efficace d'un BP.

3. Différentes phases de la méthode

Le processus de développement d'un processus métiers qui nous proposons dans ce travail suit les étapes suivantes (voir figure 19):

3.1 Analyse et compréhension du problème

Pour comprendre comment réaliser un bon BP, nous devons d'abord comprendre la dimension du problème à résoudre. Les exigences principales pour réaliser un BP sont la capacité de concevoir, exécuter, surveiller et administrer les processus d'affaires qui incorporent des interactions humaines et des systèmes. À la différence de la conception orienté objet, la conception des processus est établis en collaboration entre les analystes affaires et les analystes techniques. Les analystes techniques sont invités à comprendre les aspects d'affaires et analyser les objectifs de l'entreprise et son organisation afin d'être en mesure de décomposer l'ensemble de son activité en processus métier.

3.2 Identification des processus

Dans cette étape, il est préconisé de devoir répertorier les processus de l'entreprise, en distinguant les processus qui sont au cœur de son métier et les processus supplémentaires, et en identifiant ceux qu'il convient d'informatiser et ceux qui seront difficilement informatisables. Pour identifier un processus métier, il y a deux approches possibles : « top-down » ou « bottom-up » [19].

Typiquement, l'approche « top-down » commence par l'organigramme. En partant du plus général et en allant vers le plus particulier, par raffinements successifs. Cela liste les responsabilités de chaque département et identifie les processus de haut niveau supportés par ces responsabilités. Ces processus sont ensuite décomposés en sous processus et ainsi de suite jusqu'à ce qu'il ne soit plus possible de les décomposer en sous processus [19]. L'avantage de cette approche est qu'elle fournit une large vision d'ensemble de l'organisation. L'inconvénient est qu'elle manque de détail ainsi qu'un douteux degré de précision.

L'approche « bottom-up » démarre par l'interview des employés à propos de leurs activités journalières et la tentative d'intégrer ces informations dans des processus cohérents mis bout à bout [18], et en partant de parties de processus déjà modélisées, isolées comme des composants, on construit un processus plus global par « assemblage » de composants de processus [20]. Cette approche est extrêmement précise mais l'on peut facilement se perdre dans des détails [19].

3.3 Description des processus métiers

Cette étape est divisée en deux phases, dont la construction du flot du contrôle en suite la définition des critères de fiabilités par les concepteurs :

3.3.1 Construction du flot de contrôle

C'est l'étape la plus importante dans le processus de développement, dans cette étape, l'entreprise doit formaliser son savoir-faire dans des descriptions des processus sous forme des spécifications. La spécification permet de décrire d'une façon abstraite, les différentes parties du processus métier durant sa conception. La meilleure façon pour un analyste de décrire les spécifications d'un processus métier consiste à utiliser un ensemble de modèles [15], les analystes du processus métier doivent représenter les spécifications par un modèle nommé à flot de contrôle. Il est représenté comme un algorithme, où une séquence d'étapes, dont des conditions, des boucles…etc. Il représente l'ordre dans lequel les activités sont exécutées en spécifiant des connecteurs de contrôle entre les activités. Un modèle est une représentation d'un aspect quelconque du processus métier en cours de développement. Nous utilisons différents types de modèles pour montrer le processus métier à divers degrés de détails (ou niveaux d'abstraction). Le flot de contrôle est réalisé grâce à des outils spécifiques de modélisation généralement avec une interface graphique. Parmi ces outils de modélisation, nous citons UML, BPMN, RdP qui ont été introduits dans le chapitre 1.

3.3.2 Définition des critères du concepteur

Une fois l'expression de besoin en langage naturel rédigée et validée, elle est décrite sous forme d'un modèle (ou flot de contrôle). Il est nécessaire de montrer la manière de réagir aux problèmes du monde réel qui apparaissent en cours d'exécution du processus métier. Ces problèmes que nous allons traiter dans la suite sont liés aux échecs d'exécution des activités composante d'un processus métier.

Les concepteurs spécifient en plus du flot de contrôle des mécanismes de recouvrement. Si un service dans un processus métier est échoué, le concepteur doit spécifier quels sont les services qui sont en relation avec cet état (échoué). Ces services devront être notifiés, dont chacun doit mettre à jour son état lui même (annuler si il est en cours d'exécution, ou le compenser s'il se termine son exécution) dès qu'il reçoit la notification.

Exemple

Voici un exemple de critères du concepteur correspondant au processus métier de l'agence de voyage de figure 3 du chapitre 1 :

- En cas d'échec du service de réservation de vol il faut annuler (si le service est en cours d'exécution) ou compenser (s'il a terminé son exécution) immédiatement les services de Réservation d'hôtel et de location de voiture.
- En cas d'échec du service de Réservation d'hôtel il faut annuler ou compenser immédiatement les services de Réservation de vol et de location de voiture.

A cet effet nous avons en général un service échoué et des services qui devront être notifiés. Le problème est comment concrétisé ces critères, c'est-à-dire le processus conçu doit s'exécuter, et traite les problèmes rencontrer lors de son exécution selon ces critères. Ce problème est récurrent donc dans la plupart des processus métiers dans le monde réel.

Il faut avoir une solution conceptuelle à ce problème, d'une manière où ce modèle de conception devient réutilisable à chaque fois les analystes veulent réaliser un nouveau processus métier, ce qui exige d'avoir une conception efficace qui répond à tout genre de processus métier.

Cependant, notre processus métier est une orchestration de service Web, ou chaque service peut prendre un des quatre états suivants :

- En cours : touts les services du BP prennent l'état 'en cours', une fois le processus est lancé.
- Terminer : une fois le service se termine son exécution avec succès, il change son état de l'état 'en cours' vers l'état 'terminer'.
- Annuler : quand un service en cours d'exécution est annulé, il prend l'état 'annuler'.
- Compenser : quand un service se termine son exécution avec succès, mais il devra annuler leurs effets, il prend l'état 'compensé'.

Nous rappelons qu'un processus métier peut être réalisé par une composition de services Web, chacun peut être définis comme un programme modulaire, indépendant et auto-descriptif qui peut être découvert et invoqué via Internet ou un intranet [31]. L'indépendance des services Web implique un faible couplage, cela sans doute est un avantage, car il permet aux entreprises d'éviter les coûts engendrés par l'intégration via des communications fortement couplées, malgré cela, le processus de conception efficace d'une composition fiable devient difficile. Nous essayons dans ce chapitre de proposé un modèle de conception efficace qui doit favoriser leur réutilisation tout en fournissant l'ensemble des informations utiles et nécessaires à leur bonne utilisation.

Comme nous l'avons déjà précisé dans le chapitre 2, il n'existe pas jusqu'à récemment d'approches spécifiquement dédiées à l'ingénierie de modèle réutilisable. Mais ce qui est évident, nous pouvons réutiliser un autre modèle de conception efficace qui traite un problème similaire à notre problème. Alors il suffit de trouver ce modèle et de chercher une manière pour l'adapter à notre contexte. Comme a été discuté dans le chapitre mentionnons les Design Patterns tirés du catalogue GoF [7] sont des modèles de conception efficace à des problèmes de conception récurrente. En outre le Pattern Observer décrit dans ce catalogue est un modèle de conception efficace d'un problème dont nous classons comme similaire à notre problème, dont les détails des motivations de choix seront discutés par la suite. Alors nous pouvons réutiliser ce pattern pour construire notre modèle qui devient lui même un modèle réutilisable durant la conception des processus métier. Néanmoins l'utilisation de ce pattern ce n'est pas de calquer exactement le pattern sur notre modèle, car le Pattern Observer est conçus pour réaliser des logiciels orientés objet et n'ont pas orientés service Web comme c'est le cas de notre processus métier. Alors il est indispensable de le modifier et l'adapter à notre contexte c'est ce que nous détaillons plus loin dans ce chapitre.

Cette étape se termine avec la livraison des modèles de conception du processus. L'étape suivante dans la succession d'étapes sera consacrée à l'intégration des différents modèles.

3.4 Intégration

Les différents modèles qui se construisent sont par la suite intégrées dans cette étape. Elle peut se faire dans le cadre d'une collaboration étroite entre les concepteurs du processus métier, d'une part et ses utilisateurs d'autre part, sous la forme d'un développement commun. Les utilisateurs peuvent ainsi soumettre des amendements aux propositions des concepteurs pour coller au mieux aux objectifs définis dans la première et la deuxième étape.

Avant que le processus métier conçu soit mis en pratique, il est indispensable de vérifier, d'une part, que le modèle conçu ne contient pas d'erreurs conceptuelles ou d'erreurs sémantiques et de s'assurer, d'autre part, que l'ensemble des objectifs définis initialement sont atteint dans la pratique. La simulation et la visualisation sont des outils important pour la validation, simuler le processus avant qu'il soit implémenté, ce qui permet une amélioration (optimisation) avant le déploiement

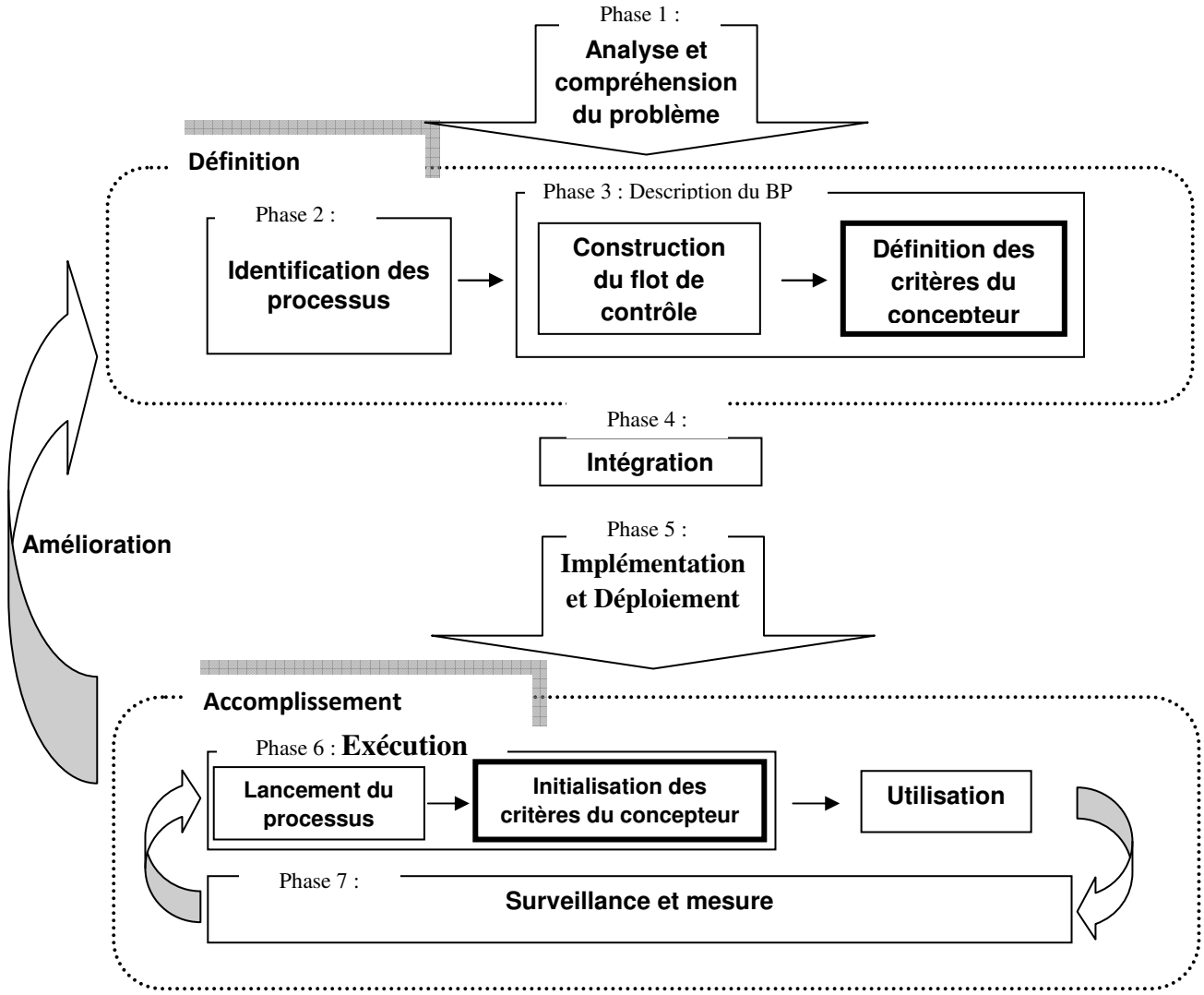


Fig. 19 Notre Méthode de développement des processus métiers

3.5 Implémentation et déploiement

Cette étape consiste en la mise en œuvre d'une solution de BP, reliée les services Web de l'entreprise et entre les entreprises pour former un processus conformément à la conception définie, répondant aux objectifs de l'entreprise. On passe à l'étape de déploiement, qui est le processus qui couvre des activités telles que la configuration, l'installation, la mise à jour, la reconfiguration et la désinstallation. Ce processus n'est souvent pas considéré de manière explicite dans le processus de développement, il se situe après la livraison du processus métier et est souvent laissé à la charge des utilisateurs.

3.6 Exécution des processus

Il s'agit de la phase opérationnelle de la solution BP. Elle permet le déroulement des tâches définis dans le processus. Comme exemples de tâches on peut citer le transfert de fichier, le lancement d'un script, la lecture de message dans une file de messages, la transformation du type de données, une demande de confirmation d'un employé. Les tâches sont basées sur des transactions qui mettent en jeux des applications ou la livraison

asynchrone à un utilisateur ou une application externe. Cette étape permet aux utilisateurs d'exploiter la solution.

3.7 Surveillances et mesure

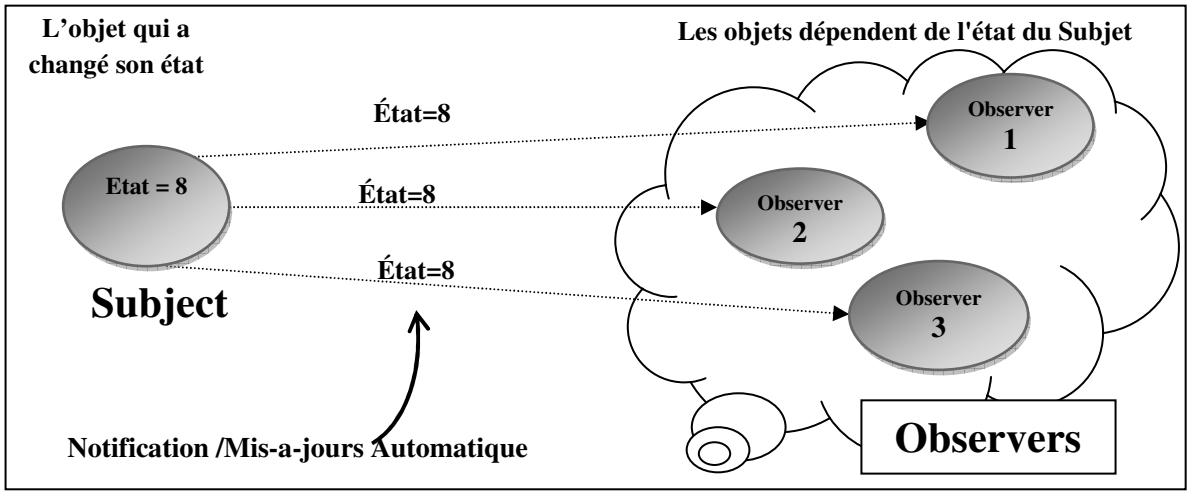
La surveillance des processus est fondamentale. Un des buts majeurs du BPM est de permettre un contrôle permanent et une amélioration constante des processus. Dans cette étape nous analysons l'état des processus en présentant leurs performances.

Avec le suivi de cette succussion d'étapes et en utilisant le modèle de conception présenté dans ce travail nous permettons d'atteindre des processus métiers permettant de concrétisé les désirs des concepteurs et enfin nous augmentons la satisfaction des clients. La section suivante sera consacrée aux détails de l'utilisation du Pattern Observer pour construire notre modèle de conception, nous commençons par les motivations de son utilisation.

4. Motivation du choix du Pattern Observer

L'intention de ce pattern est qu'il définit une dépendance « un à plusieurs » entre les objets de façon à ce que, si un objet change d'état, tous les objets qui en dépendent, en soient notifiés et mis-à-jour automatiquement [7]. Ce pattern permet une interaction entre plusieurs observateurs sans exiger aucune communication directe entre elles. Sa structure présentée dans le chapitre 2 décrit comment établir les relations entre les éléments du modèle. Les objets principaux dans ce modèle sont le sujet (Subject) et l'observateur (Observer). Un Subject peut dépendre de n'importe quel nombre d'Observers. Tous les Observers reçoivent une notification chaque fois que le Subject subit un changement de son état. En réponse, chaque Observer interrogera le Subject pour synchroniser son état avec lui [7]. A cette fin ce Pattern permet de modifier les Subjects et les Observers indépendamment. Par conséquent on peut ajouter un observateur sans avoir modifie les autres et sans modifier le Subject.

Il n'est plus à démontrer que ce problème est similaire à notre problème. Nous avons effectué une imitation comme celle utilisé par l'exemple présenté dans la section 5.4 du chapitre 2. Cependant, le contenu de la figure 20 montre l'imitation du pattern Observer définie dans [7] vers notre problème. Tout simplement nous pouvons dire que les services échoués deviendront des Subject et les Observers deviendront les services à notifier par le service échoué. Nous ajoutons que chaque service Web dans le processus métier peut jouer les deux rôles Subject et Observer, selon les spécifications des concepteurs. Le premier rôle dans le cas où il échoue et le deuxième s'il reçoit une notification par un autre service qui a changé son état. Nous avons montré les motivations qui nous ont menés à choisir ce pattern pour construire notre modèle de conception, nous décrirons la manière de son utilisation dans la section suivante.



Interaction entre les éléments du Pattern Observer [17]

Technique d'imitation

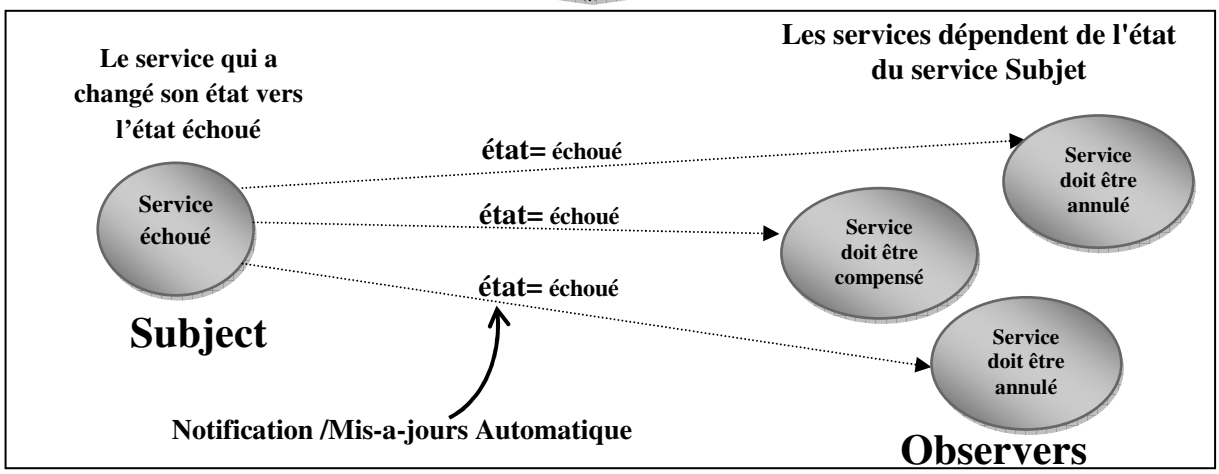


Fig. 20 Un aspect d'imitation du Pattern Observer vers notre problème

5. L'utilisation du Pattern Observer

L'utilisation du pattern Observer avec sa définition originale dans [7] pour la conception d'un processus métier réalisé par une composition de services Web est insuffisante, elle nécessite une adaptation.

5.1 Besoin d'une adaptation du Pattern Observer aux processus métiers

Les travaux actuels sur la réutilisation distinguent deux manières d'utilisation des patrons, que nous avons introduites dans le chapitre 2. La première consiste à réutiliser directement le patron, qui concerne le cas où les besoins cernés pour l'application correspondent exactement à des patrons prédéfinis. La deuxième manière concerne l'adaptation ou la spécialisation des patrons, il faut alors adapter ou spécialiser des patrons existants pour qu'ils correspondent aux besoins cernés pour l'application. Dans notre conception, la solution proposée par le Pattern Observer ne correspond pas exactement à nos besoins d'application. Il faut donc adapter ou spécialiser la solution. Nous utilisons la deuxième manière de réutilisation par l'adaptation du pattern Observer. Néanmoins ce pattern présente en revanche des limites et des problèmes liés à son utilisation avec sa

structure originale définie dans [7], plus particulièrement lors de son imitation et implémentation dans le contexte des processus métiers. Nous présentons dans ce travail une adaptation de ce patron aux processus métiers sans perdre l'essence du patron. Avant de présenter les détails de l'adaptation il faut d'abord expliquer l'inadéquation de son utilisation avec sa structure originale.

5.2 Inadéquation de l'utilisation de la structure originale du Pattern Observer

Nous pouvons citer dans les points suivants les problèmes liés à l'utilisation du Pattern Observer pour concevoir un processus métier réalisé par une composition de services Web :

A. Le pattern Observer est conçu pour définir une dépendance spécifique « un à plusieurs » entre objets, alors que, les processus métiers définit dans notre approche sont réalisés par les services Web. Ces derniers sont considérés par erreur comme des objets distribués. Un service Web est juste comme un objet distribué. Tellement les similitudes sont fortes entre les services Web et les objets distribués, il est compréhensible que beaucoup de gens croient qu'elles sont les mêmes [25].

B. L'interaction entre les services Web du processus est réalisée par l'envoi des messages dans un format standard, sous forme de documents XML qui peuvent être facilement transporté par l'intermédiaire des protocoles standard d'Internet (SOAP). Par contre le Pattern Observer fournit un mécanisme d'interaction entre les objets « Subject » et « Observer » basé sur des appels de méthode d'objet local ou à distance se qui ne s'adapte pas aux services Web. Cependant, échanger des documents XML et très différent d'appeler un objet instancié, invoqué une méthode d'une instance spécifique et recevoir le résultat de l'invocation dans une réponse et libérant l'instance après plusieurs échanges [25].

C. Les méthodes Attach (Observer) et Detach (Observer) du Pattern Observer qui permettent respectivement la sauvegarde et la suppression du côté Subject des références aux objets Observers (cela peut être un simple pointeur si les deux méthodes s'exécutent dans le même espace mémoire, dans le cas d'un objet situé sur une autre machine la référence contient alors des informations indiquant, par exemple, l'adresse réseau de la machine distante [23]). Néanmoins, un service Web est invoqué via son URL.

5.3 Discussions

Il est clair que l'utilisation de la structure originale du Pattern Observer pour concevoir un processus métier pose un problème lors de son utilisation dans le contexte des services Web, car comme déjà cité plus haut un service n'est pas un objet. La solution évidente qui s'impose peut être par l'adaptation de Pattern Observer pour modéliser les services Web. L'idée de cette adaptation a été initiée par un travail réalisé par [48] qui consiste en l'implémentation du Pattern dans le contexte des services Web. Nous avons réutilisé le Pattern de [48] pour proposer un modèle qui traite les échecs d'exécution des services Web composant un processus métier. La section suivante démontre les détails de notre modèle proposé.

6. Adaptation du Pattern Observer pour la conception des processus métiers

Dans cette section nous présentons les détails de l'intégration du Pattern Observer dans un modèle de conception des processus métiers composé par des services Web. Nous illustrons la structure du modèle utilisant le diagramme de classe d'UML. Ce dernier représente l'aspect statique du système. Ce diagramme est largement utilisé pour décrire les types d'objets dans un système et leurs relations, est une collection d'éléments déclaratifs, tels que les classes et les interfaces ainsi que les relations entre elles, reliées comme un graphe. Nous définissons par la suite la collaboration entre ces éléments utilisant le diagramme de séquences d'UML. Ce diagramme est un modèle d'interaction entre les instances, illustre l'aspect dynamique du processus métier. Il montre les objets qu'ils participent dans l'interaction et les messages échangés entre eux. Noter que l'UML ne proscrit pas ou ne donne pas un avis sur la façon d'utilisation de leur notation dans un processus de développement de logiciel, et il ne participe pas d'une partie d'une méthodologie de conception orientée objet [21].

6.1 Structure du modèle utilisant le Pattern Observer

L'adaptation du pattern Observer aux services Web sera par l'implémentation des éléments du pattern en tant que services Web, les méthodes de la classe Subject seront encapsulé dans un service Web appelé WSSubject, de la même façon les méthodes de la classe Observer seront encapsuler dans un autre service Web appelé WSObserver. L'adaptation consiste en la représentation la structure du pattern sous forme d'interfaces et des classes concrets implémentant ces interfaces. Nous rappelons qu'un service Web est composé de deux parties : Une interface et une implémentation. L'interface est standard, l'implémentation quand à elle est propriétaire, et dépend de la plateforme utilisée, cela a peu d'importance à partir du moment où l'interface est standard, et que cette implémentation dialogue avec ses clients par échange de messages sous forme de documents XML. L'entreprise qui procède à réaliser un processus métier par orchestration de services Web, elle doit fournier un coordinateur prend le contrôle de tous les services Web impliqués et coordonne l'exécution des différentes opérations des services Web (voir la figure 21).

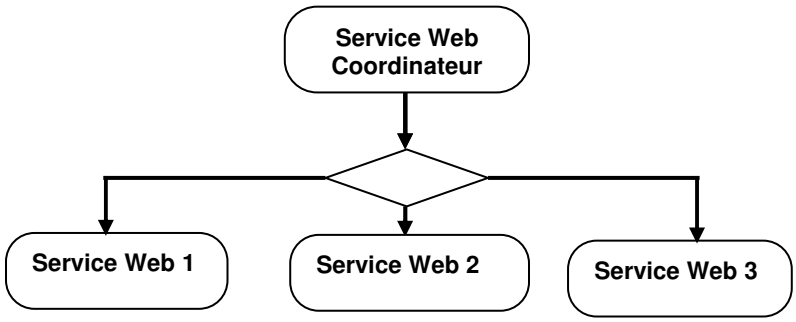


Fig. 21 Orchestration de services Web [24]

La figure 22 consiste en la représentation d'une vue globale de la structure de notre modèle d'une conception efficace des processus métiers qui est à la base du pattern Observer. Ce modèle de conception contient un ensemble d'interfaces à des services Web et leurs implémentations concrètes, Une interface ne décrit aucune structure ni aucune implémentation. Elle ne peut donc pas contenir d'attributs, ni de méthodes fournies sous la forme d'une implémentation. Une interface est identifiée par l'indication du stéréotype «

Interface » accolé à son nom. Nous essayons dans cette section de comprendre ces composants et leurs rôles dans ce modèle.

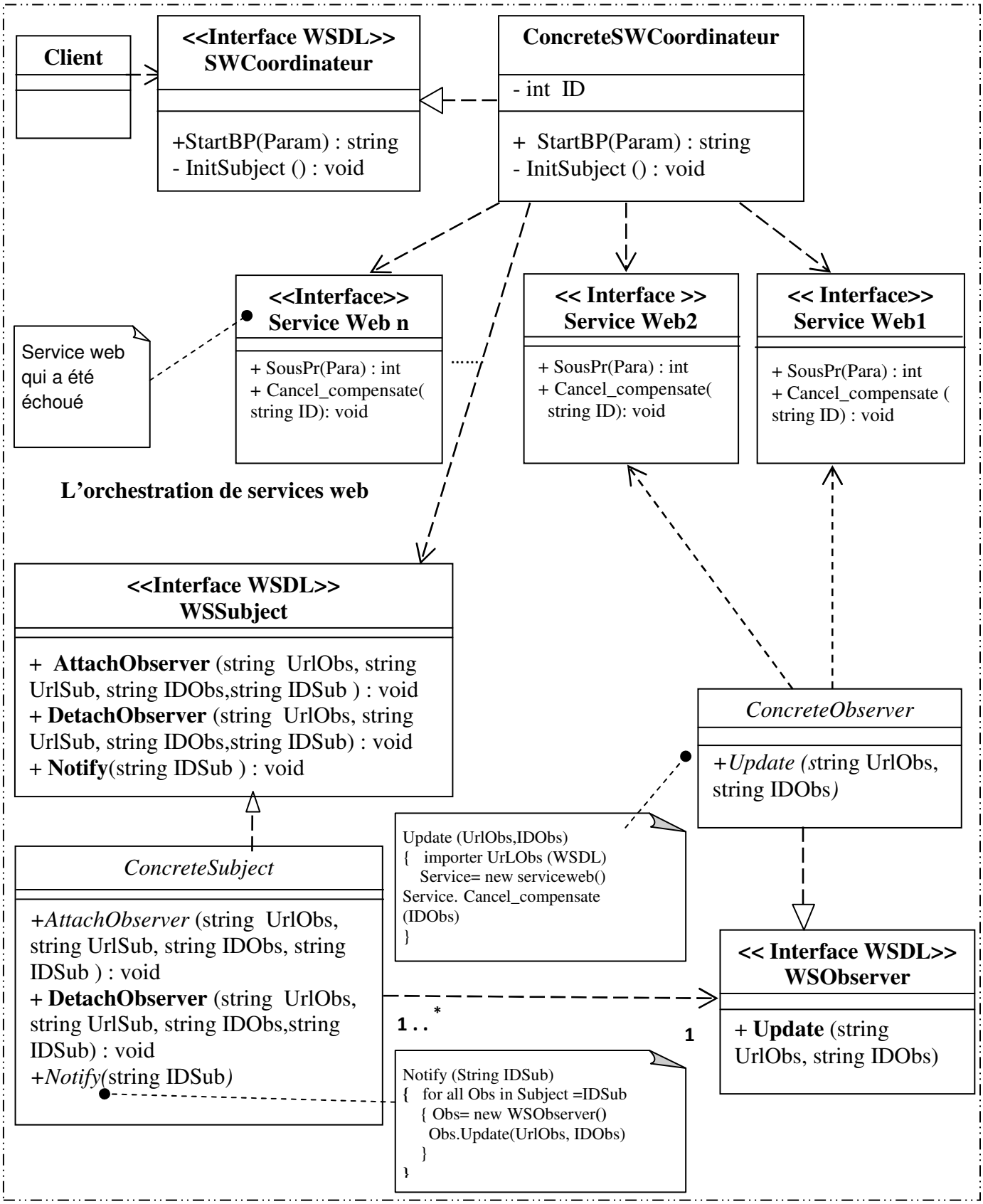


Fig. 22 Structure du notre modèle de conception du processus métier utilisant le Pattern Observer

- **Interface WSSubject**

C'est une interface d'un service Web Subject (dont le document WSDL qui représente l'interface de ce service Web sera présentée dans la section suivante), qui maintient des paires sous la forme (Url Subject, liste d'Urls d'Observers), ces paires représentent les spécifications du concepteur, c'est à dire: l'URL du service Web échoué (Subject) et les URL des services Web qui devront être notifiés (Observers), l'interface comporte trois opérations primaires sont : 'AttacheObserver', 'DetacheObserver' et 'Notify'. L'opération 'AttacheObserver' permet d'attacher un Observer à un Subject prend quatre paramètres le premier paramètre 'UrlSub' représente l'URL du service échoué, le deuxième paramètre 'UrlObs' représente L'URL d'un service doit être se notifier (Observer), le troisième paramètre est 'IDObs' représente l'identification d'une instance de service Web Observer attacher à leurs Subject utiliser dans la notification en cas d'échec du service Subject alors le dernier paramètre 'IDSub' représente l'identificateur de l'instance du service qui doit être échoué, l'opération 'DetachObserver' permet de détacher un Observer d'un Subject utiliser si le concepteur effectuer un changement dans ses spécification. L'opération 'Notify' permet de notifier les observateurs lors d'un changement d'état subit un service Web Subject, elle a un seul paramètre 'IDSub' qui représente l'identificateur d'une instance du service qui doit être échoué, utiliser pour identifier leurs Observers, lors de l'appelle du service Web Observer utilisant son URL, ensuite il pointe sur l'opération 'update'.

- **ConcreteSubject**

Ce composant représente l'implémentation concrète des opérations du service Web (Subject). Ce dernier maintient la liste des URL et des identificateurs des services Web observateurs qui ont en relation avec ce Subject.

- **Interface WSObserver**

C'est une interface du service Web Observer (le fichier WSDL de cette interface est présentée dans la section suivante), à partir de laquelle son implémentation fait appel aux aux services Web qui devront être notifiés sur le échec d'exécution du service Subject. Elle ne contient qu'une seul opération 'update'. L'implémentation du service Subject fait appelle à cette opération lors de la notification.

- **ConcreteObserver**

Ce composant représente l'implémentation de l'opération 'update' celle publie dans interface de l'observateur. Il contient deux paramètres 'UrlObs', l'URL du service Web qui doit être annulé ou compensé, et 'IDObs', un identificateur qui permet d'identifier l'instance du service Web à traiter.

- **Interface SWCoordinateur**

C'est l'interface du service Web coordinateur qui fait appelle aux autre services qui compose le processus métier, utilisant l'opération 'StartBP'. En outre Ce service joue le rôle du gestionnaire, il permet d'appeler les opérations du service Web Subject à travers son implémentation, utilisant l'opération 'InitSubject', cette opération est invoqué quand le concepteur ayant déjà introduit ses critères de recouvrement. Elle invoque la méthode « AttachObserver » du service Web Subject, pour affecter pour chaque service Web Subject leur liste de services Observateurs, tout en respectant les critères bien définit par le développeur.

- **Concrete SWCoordinateur**

Ce composant représente l'implémentation concrète des opérations de l'interface du SWCoordinateur. Après la réception d'un échec d'exécution d'un service Web, la classe Concrète **SWCoordinateur** invoque l'opération 'notify' du service WSSubject pour informer les observateurs sur ce changement. Cette classe contient un seul attribut représente l'identificateur de chaque instance de processus métier.

- **Interface ServiceWeb i**

Représente l'interface des services Web 'i' qui entre dans la composition du processus métier, où chaque service peut jouer le rôle du Subject quand il échoue, comme il peut jouer le rôle d'Observer s'il est en relation avec un autre service qui doit être échoué, ces rôles sont introduits par le concepteur du processus métier. Ce service fournie l'opération 'Anuuler_compenser' s'il joue le rôle d'un Observer, qui permet d'annuler ou compenser le service, qui est invoqué par l'implémentation du service Observer par son opération 'Update'.

- **Client**

Le classe client représente le client du service Web Coordinateur. Elle fait invoque la méthode 'StartBP' définie dans le service Web Coordinateur pour lancer le processus métier.via interface graphique.

De cette manière nous avons présenté l'aspect statique du processus métier par un diagramme du classe, contient les composants du modèles et les relations entre eux. La section suivante est consacrée à la démonstration de l'aspect dynamique.

6.2 Collaboration entre les composants du modèle

Les collaborations dans modèle sont illustrés par un diagramme de séquence d'UML, Un diagramme de séquence à deux dimensions: la dimension verticale représente le temps, la dimension horizontale représente différents instances de services Web. Le temps procède en bas de la page. Il n'y a aucune signification à l'ordre horizontale des instances. La figure 23 collecte les différents composants du modèle en utilisant le diagramme de séquence d'UML. Noter que l'ordre horizontal des lignes est arbitraire, les flèches d'appel sont souvent arrangées pour procéder dans une direction à travers la page, mais ce n'est pas toujours possible et l'ordre ne donne pas l'information.

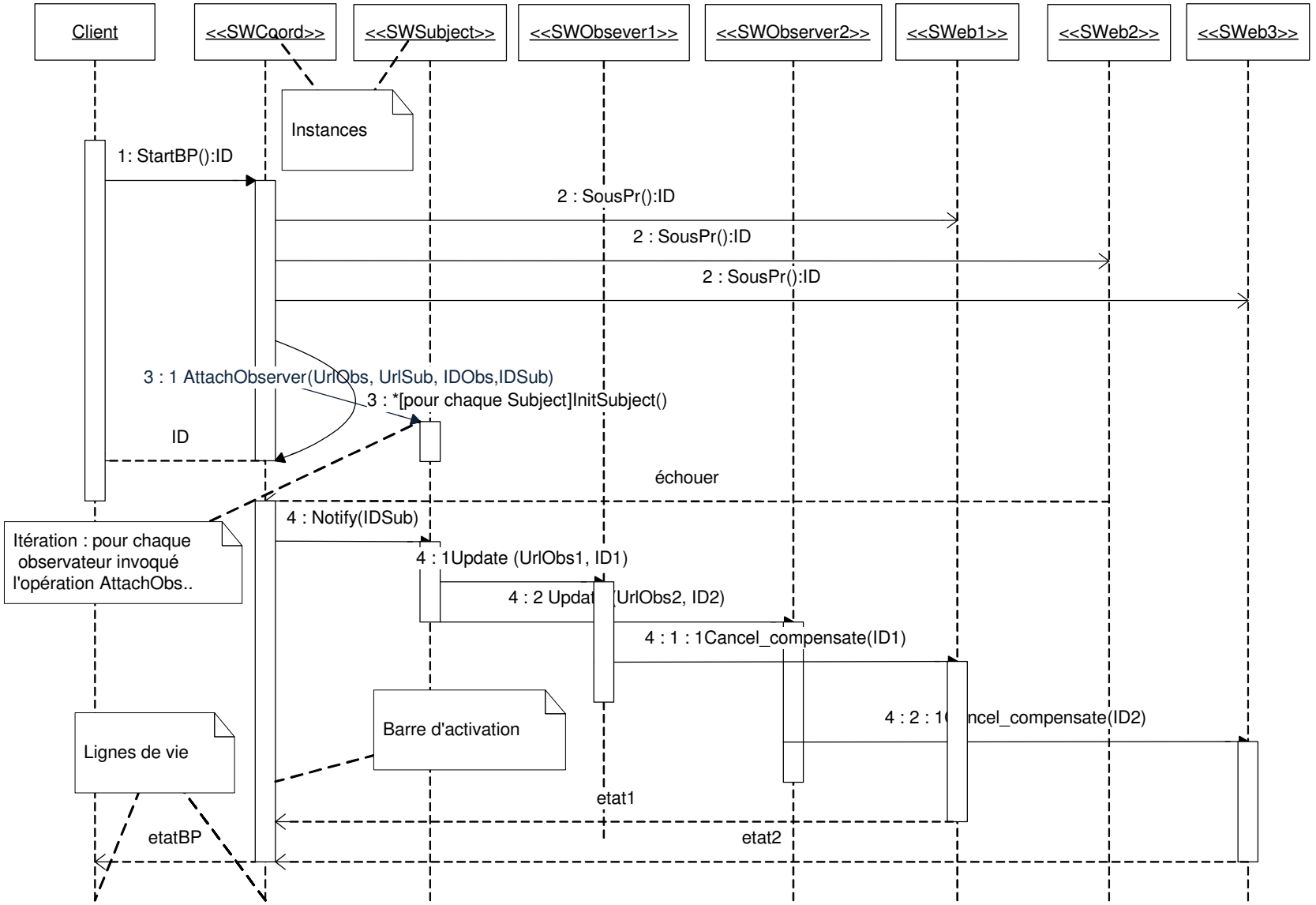


Fig. 23 Collaboration entre les composants du modèle utilisant le diagramme de séquence d'UML

Le client commence ce diagramme par le lancement d'un processus métier via une interface graphique au service Web coordinateur. Il invoque l'opération StartBP() du service Web coordinateur, qui fait à son tour des appels aux différents sous-processus qui compose le processus métier, qui ont encapsulé dans trois services Web dans cet exemple. Chaque sous-processus retourne un identificateur de l'instance du service Web lancé. Le service coordinateur invoque localement l'opération 'InitSubject' où le contenu de cette opération invoque la méthode AttachObserver(UrlObs, UrlSub, IDSub, IDObs) publié dans le service Web Subject, les paramètres de cette méthode sont l'URL du service Subject, l'URL du service sui joue le rôle d'Observer et un identificateur de leur instance et le dernier paramètre est l'identificateur du service qui joue le rôle du Subject. l'opération 'InitSubject' pour rôle d'initialiser les critères du concepteur pour l'instance en cours Mettant la correspondance entre chaque Subject et ses observateurs en sauvegardant le résultat dans une base de données XML par exemple. Voici un exemple de DTD (Document Type Definition) qui représente la structure de ce document XML :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT Analyste_Criteria (Criteria)*>
        <!ELEMENT Criteria (Subject, Observers)>
                <!ELEMENT Subject (NameSubject,UrlSubject,IdSubject)>
                    <!ELEMENT NameSubject (#PCDATA)>
                    <!ELEMENT UrlSubject (#PCDATA)>
                    <!ELEMENT IdSubject (#PCDATA)>

                <!ELEMENT Observers (Observer)*>
                   <!ELEMENT Observer (NameObserver,UrlObserver,IdObserver)>
                       <!ELEMENT NameObserver (#PCDATA)>
                       <!ELEMENT UrlObserver (#PCDATA)>
                       <!ELEMENT IdObserver (#PCDATA)>
```

Dans un moment un des ces services peut échouer. Alors si le cas, il envoi son état au service coordinateur. Ce dernier par la suite invoque la méthode 'Notify' du service Subject, incluant dans ses paramètres l'identificateur du service échoué. Le service Subject va instancier pour chaque observateur de ce Subject des services Web Observer, et en invoquant l'opération 'Update'. Les paramètres de cette opération sont l'URL de service Observer et l'identificateur de l'instance du service à notifié, l'Url utiliser pour importer l'interface du service Web à notifié, et l'identificateur utilisé pour identifier l'instance du service qui doit être notifié. Après l'importation de l'interface en invoquant la méthode 'Cancel_compensate' du service concerné, un seul paramètre dans cette méthode qui est l'identificateur de l'instance. Avec cette méthode le service doit changer son état soit vers annuler s'il est en cours d'exécution soit vers compensé s'il est terminé son exécution, et il effectué leur traitement qui concerne le changement d'état. Chaque service change son état il envoi au service coordinateur le nouveau état. Le service coordinateur prend la décision de l'état du processus global en se basant sur les différents états prévenant de différents services Web.

6.3 Obtention des documents WSDL des Services Web Subject et Observer

Le document WSDL du service Web Subject (Interface WSSubject) qui à été discuté plus haut est le suivant : qui contient les opérations Attach et Detach et enfin Notify.

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://www.BPDesignMethod.com//WSSubject"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://www.BPDesignMethod.com//WSSubject"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
<s:schema elementFormDefault="qualified"
targetNamespace="http://www.BPDesignMethod.com//WSSubject">
<s:element name="AttachObserver">
<s:complexType>
<s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="NameSub" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="UrlSub" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="IDSub" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="NameObs" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="UrlObs" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="IDObs" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
<s:element name="AttachObserverResponse">
  <s:complexType />
  </s:element>
<s:element name="Notify">
<s:complexType>
<s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="IDSub" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
<s:element name="NotifyResponse">
  <s:complexType />
  </s:element>
<s:element name="DetachObserver">
<s:complexType>
<s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="NameSub" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="UrlSub" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="IDSub" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="NameObs" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="UrlObs" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="IDObs" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
```

```
<s:element name="DetachObserverResponse">
 <s:complexType />
 </s:element>
 </s:schema>
 </types>
   <message name="AttachObserverSoapIn">
      <part name="parameters" element="s0:AttachObserver" />
   </message>
   <message name="AttachObserverSoapOut">
       <part name="parameters" element="s0:AttachObserverResponse" />
   </message>
   <message name="NotifySoapIn">
     <part name="parameters" element="s0:Notify" />
   </message>
   <message name="NotifySoapOut">
      <part name="parameters" element="s0:NotifyResponse" />
   </message>
   <message name="DetachObserverSoapIn">
     <part name="parameters" element="s0:DetachObserver" />
   </message>
   <message name="DetachObserverSoapOut">
      <part name="parameters" element="s0:DetachObserverResponse" />
   </message>
<portType name="WSSubjectSoap">
  <operation name="AttachObserver">
    <documentation>Add new Criteria</documentation>
     <input message="s0:AttachObserverSoapIn" />
     <output message="s0:AttachObserverSoapOut" />
 </operation>
<operation name="Notify">
    <input message="s0:NotifySoapIn" />
    <output message="s0:NotifySoapOut" />
</operation>
<operation name="DetachObserver">
    <input message="s0:DetachObserverSoapIn" />
    <output message="s0:DetachObserverSoapOut" />
</operation>
</portType>
<binding name="WSSubjectSoap" type="s0:WSSubjectSoap">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
<operation name="AttachObserver">
 <soap:operation
soapAction="http://www.BPDesignMethod.com//WSSubject/AttachObserver"
style="document" />
<input>
   <soap:body use="literal" />
  </input>
<output>
  <soap:body use="literal" />
   </output>
```

```
</operation>
<operation name="Notify">
  <soap:operation soapAction="http://www.BPDesignMethod.com//WSSubject/Notify"
style="document" />
<input>
  <soap:body use="literal" />
    </input>
   <output>
  <soap:body use="literal" />
    </output>
  </operation>
<operation name="DetachObserver">
  <soap:operation
soapAction="http://www.BPDesignMethod.com//WSSubject/DetachObserver"
style="document" />
<input>
  <soap:body use="literal" />
  </input>
<output>
  <soap:body use="literal" />
  </output>
  </operation>
  </binding>
<service name="WSSubject">
<port name="WSSubjectSoap" binding="s0:WSSubjectSoap">
  <soap:address location="http://192.168.0.3/WSerSubject/WebSSubject.asmx" />
  </port>
  </service>
  </definitions>
```

Ce document WSDL décrit l'interface d'un service Web Subject, il contient les éléments suivants :

- Les définitions de types décrits par un schéma XML;
- Une partie des messages, décrit les noms (AttachObserverSoapIn, NotifySoapIn, DetachObserverSoapIn, AttachObserverSoapOut, NotifySoapIn , DetachObserverSoapIn) et les types (*Request-response*) d'un ensemble de champs à transmettre ;
- Une partie Décrit les opérations du service Web (Attach, Detach, Notify). Chaque opération de ceux-ci à un message d'entrée et un massage de sortie ;
- Une partie pour spécifier une liaison d'un <PortType> à un protocole concret (SOAP) ;
- La dernière partie décrit le nom du service Web (WSSubject) et définit le Port qui représente un point d'entrée comme la combinaison d'une Liaison et d'une adresse Internet (réseau).

Le document WSDL du service Web Observer (Interface WSObserver) qui a été expliqué dans les sections précédent, est le suivant :

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://www.BPDesignMethod.com/WSObserver"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://www.BPDesignMethod.com/WSObserver"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
<s:schema elementFormDefault="qualified"
targetNamespace="http://www.BPDesignMethod.com/WSObserver">
<s:element name="Update">
<s:complexType>
<s:sequence>
 <s:element minOccurs="0" maxOccurs="1" name="UrlObs" type="s:string" />
 <s:element minOccurs="0" maxOccurs="1" name="IdObs" type="s:string" />
 </s:sequence>
 </s:complexType>
 </s:element>
<s:element name="UpdateResponse">
 <s:complexType />
 </s:element>
 </s:schema>
 </types>
<message name="UpdateSoapIn">
 <part name="parameters" element="s0:Update" />
 </message>
<message name="UpdateSoapOut">
 <part name="parameters" element="s0:UpdateResponse" />
 </message>
<portType name="WSObserverSoap">
<operation name="Update">
 <input message="s0:UpdateSoapIn" />
 <output message="s0:UpdateSoapOut" />
 </operation>
 </portType>
<binding name="WSObserverSoap" type="s0: WSObserverSoap">
 <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
<operation name="Update">
 <soap:operation soapAction="http://www.BPDesignMethod.com/WSObserver/Update"
style="document" />
<input>
 <soap:body use="literal" />
 </input>
<output>
 <soap:body use="literal" />
 </output>
 </operation>
```

```
  </binding>
<service name="WSObserver">
<port name="WSObserverSoap" binding="s0:WSObserverSoap">
  <soap:address
location="http://192.168.0.4/WebServiceObserver2/WebServiceObs.asmx" />
  </port>
  </service>
  </definitions>
```

Ce document WSDL décrit l'interface d'un service Web Observer, il contient les éléments suivants :

- Les définitions de types décrits par un schéma XML;
- Une partie des messages, décrit les noms ('UpdateSoapIn' et 'UpdateSoapOut') et les types (*Request-response*) d'un ensemble de champs à transmettre ;
- Une partie Décrit la seule opération nommée à 'Update'. Cette opération à un seul message en entrée (UpdateSoapIn), et un seul message en sortie (UpdateSoapOut) ;
- Une partie pour spécifier une liaison d'un <PortType> à un protocole concret (SOAP) ;
- La dernière partie décrit le nom du service Web (WSObserver) et définit le Port qui représente un point d'entrée comme la combinaison d'une Liaison et d'une adresse Internet (réseau)

7. Conclusion

Dans ce chapitre nous avons présenté une méthode de conception d'un processus métier, permettant de fournir des modèles efficaces répondant aux exigences des développeurs. En outre nous avons proposé dans ce travail un modèle de conception qui à la base de la réutilisation du pattern Observer. Le recours à une approche de réutilisation avait pour objectif d'augmenter l'efficacité du modèle, tout en facilitant l'évolution et la maintenance des processus métiers. En effet, le concepteur représente la réalité en fonction de sa perception et de la compréhension de celle-ci, du paradigme utilisé et des concepts qu'il renferme ainsi que des objectifs qu'il souhaite atteindre. De ce fait, le modèle n'est pas une représentation totalement conforme à la réalité mais une représentation plus ou moins proche de celle-ci. Cependant, un des objectifs majeurs des développeurs est d'étudier comment calquer exactement le déroulement des systèmes réels sur les modèles proposés. Nous avons illustré dans notre modèle la façon de répondre à quelques problèmes d'exécution des processus métiers selon les exigences du concepteur. L'effet du modèle proposé est indiqué dans le chapitre suivant qui montre leur utilisation avec un exemple concret d'un processus métier.

Chapitre V - Réalisation d'un processus métier basé sur les services Web

1. Introduction

Nous avons présenté dans le chapitre précédant notre méthode pour la conception des processus métiers efficaces et fiables, dont nous avons présenté notre modèle de conception des processus métiers qui est à la base du Pattern Observer. Ce modèle permet de traiter les problèmes concernant les échecs d'exécution des activités composants d'un processus métiers lors de son déroulement. Pour démontrer les effets de ce modèle il est indispensable d'utilisé un exemple concret d'un processus métier. Ce chapitre consiste en l'application de notre méthode sur un exemple concret. Une démonstration à la fin du chapitre de l'implémentation de cet exemple utilisant sur les services Web. D'abord nous présentons une manière pour la définition flexible des critères des concepteurs, et comment les respectés durant l'exécution du processus métier. Nous avons choisi comme exemple une agence d'importation, qui permet d'assurer l'achat des produits et la réservation de transport externe entre les pays (réserver dans le bateau) et la réservation du transport local en niveau du pays du client (d'après le port jusqu'à l'adresse du client).

2. Rappel sur les différentes phases de la méthode

Pour réaliser un processus métier la première phase dans notre méthode consiste en l'analyse et la compréhension du problème. Dans cette phase nous analysons les besoins de l'agence d'importation qui concernent les services à fournir aux clients. Alors le résultat de cette phase c'est l'ensemble de services qui peuvent être fournit par l'agence et l'ensemble d'information concernant les entrés et les sorties de chaque service qui doivent être introduites par le client. La deuxième phase consiste en l'identification des processus, dont nous formaliserons l'ordre dont lequel s'effectuent les invocations des services, tout en fournissant les entrer et les sortie de chaque un service. Nous arrivons à la phase de modélisation du flot de contrôle utilisant le DA UML. En suite la définition de nous critères qui concernant le traitement des échecs d'exécution de certains service. La phase suivant démontre le modèle complet du processus de l'agence. La dernière phase consiste en la réalisation du processus par une orchestration de services Web. Finalement le test et l'analyse des résultats obtenue par un scénario d'exécution, tout en démontrant le traitement des anomalies des d'exécution selon les critères que nous avons spécifié.

3. Application de la méthode sur l'exemple

Dans cette section nous appliquons notre méthode sur l'exemple motionné, nous commençons par l'analyse et la compréhension du problème.

3.1 Analyse et compréhension du problème

Supposant un client habite en Algérie essai d'importer des produits à partir de l'Europe, il doit contacter leurs fournisseurs en Europe pour soumettre sa commande, d'autre part il doit contacter les agences de transports pour transporter le produit qui a été acheter. Ces agences doivent fournir le transport externe et le transport interne. Le transport externe c'est de porter le produit d'un pays à un autre (par exemple de la l'Italie jusqu'à l'Algérie). Le transport Interne c'est de porter le produit au niveau d'un pays (d'après le port si le produit est arrivé en bateau par voie de mer ou d'après l'aéroport s'il est arrivé en avion jusqu'à l'adresse du client).

Avec l'arriver du Web le client effectue toutes ces réservations avec un seul click !, Notre exemple consiste en une agence qui fournit touts ces services. Cependant, les services achat du produit et la réservation du transport (interne ou externe) sont liée l'une avec l'autre. En d'autre terme les trois services sont obligatoires. Pour accomplir une importation les trois services doivent être accomplit. Si un des ces service échoue l'importation ne soit pas pratique et elle doit s'annuler.

En conclusion l'agence d'importation doit fournir au client trois services (achat du produit, transport interne et externe) et un service coordinateur. En plus de ces trois services il faut disposer aux clients une interface graphique pour soumettre ses informations concernant la procédure de l'achat du produit. Nous ajoutant que chaque service doit avoir une base de données concernent les informations sur le service à fournir et les informations sur les clients et ses instances lancés. Nous allons utiliser des bases de données XML dans ce travail pour chaque service. L'étape suivante représente l'identification du processus et l'ordre dans les quelles les services sont invoqués.

3.2 Identification du processus

Après la compréhension du problème, l'étape suivante consiste en l'identification du processus métier, voici les étapes pour qu'un client veut importer un produit :

1. Le client saisit ses informations concernant la commande d'un produit via une interface graphique.
2. Pour chaque produit il doit spécifier le nom du produit, le pays d'achat, la quantité demandée, la date de livraison et l'adresse où le produit sera transporté.
3. L'agence invoque les trois services et attend les résultats.
4. Si les trois services sont accomplit le client reçoit une confirmation.

Alors d'après ces séquences d'activités nous pouvons identifiés trois sous processus métiers et un processus global qui lance et contrôle et coordonne ces derniers. Dans ce travail nous avons réalisé chaque sous processus par un service Web, et le processus global lui-même est un service Web coordinateur. La figure 24 représente graphiquement l'architecture de notre exemple.

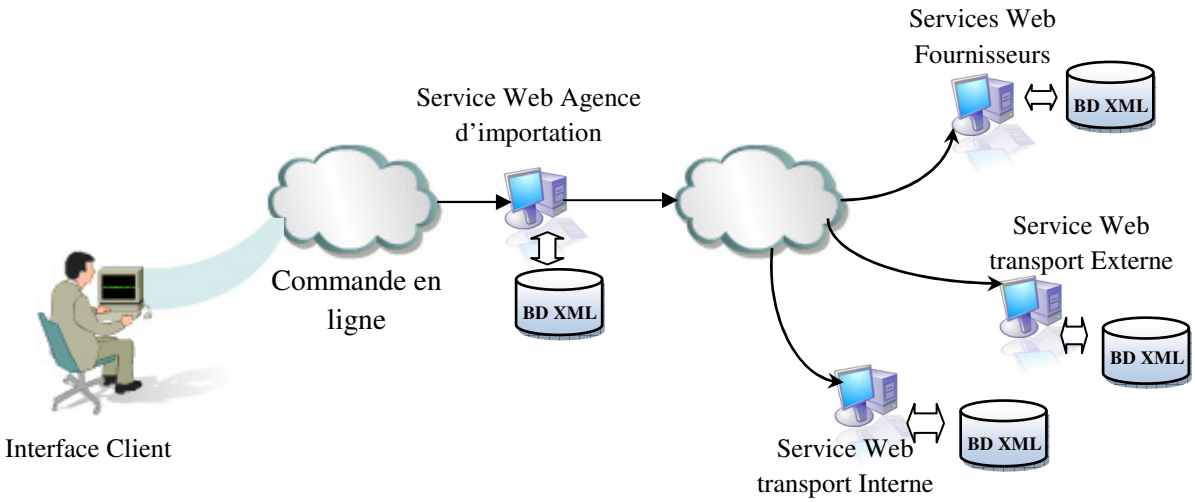


Fig. 24 Représentation graphique de l'exemple

3.3 Modélisation du flot de contrôle du processus métier

Comme il a été discuté dans les chapitres précédents le flot de contrôle est définit comme un modèle de processus décrit chaque unité de travail, mais aussi la séquence adéquate d'unités de travail pour atteindre un certain objectif. Un flot de contrôle décrit l'ordre dans lequel les activités sont exécutées en spécifiant des connecteurs de contrôle entre les activités.

Comme il est représenté dans la figure 25, nous avons utilisé le diagramme d'activité d'UML, par le recours à un outil graphique qui est Microsoft Visio, pour modéliser notre processus.

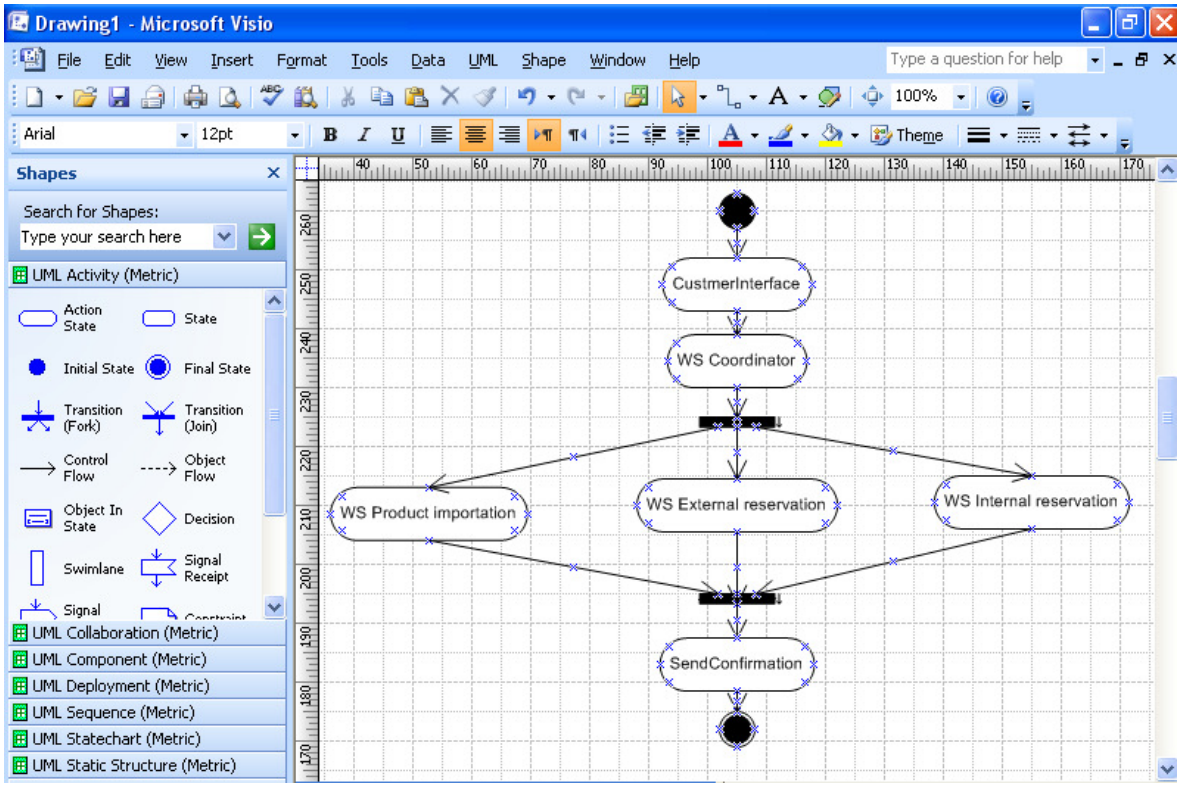


Fig. 25 Flot de contrôle du processus métier de l'agence d'importation

Le client lance un BP, via une interface graphique il introduit ces informations concernant la commande du produit et leur transport (externe ou interne). La requête du client sera envoyée au service Web de l'agence. Ce dernier lance en parallèle trois sous processus métiers encapsulé en des services Web. Le résultat c'est la jointure des résultats de chaque service. Enfin l'agence prend la décision en se basant sur la combinaison de jugement et d'information provenant de ces trois services. En suite le client reçoit la confirmation.

3.4 Définition des critères du concepteur

Après la définition du flot de contrôle le concepteur il doit spécifier les critères qui concernent l'exécution du BP. Ce travail traite les critères qui concernent la réaction aux échecs d'exécution de certains services. Voici les critères que nous avons choisis trois critères pour notre exemple de processus métier :

a. En cas d'échec du service réservation de transport externe il faut annuler les services d'achat et de transport interne s'ils sont en cours d'exécution. Si ces services sont terminé leurs exécutions ils devront être compensés.
b. En cas d'échec du service d'achat il faut annuler les services de transport externe et de transport interne s'ils sont en cours d'exécution. Si ces services sont terminé leurs exécutions ils devront être compensés.
c. En cas d'échec du service réservation de transport interne il faut annuler les services d'achat et de transport externe s'ils sont en cours d'exécution. Si ces services sont terminé leurs exécutions ils devront être compensés.

La durée d'exécution des processus métier peut prendre des heures ou des jours ou même des fois des mois. Si un service échoue les services qui sont en relation avec se service ils devront être annulés immédiatement et en abandonnant l'exécution du processus global. Si le produit à importer avec une durée de vie minimal (par exemple : la viande, les médicaments, ...) et dans le cas ou le service de transport interne peut être échoué, il ne faut pas continue l'exécution des autres services. Ils devront d'être annuler immédiatement, sinon le produit sera gâté, par conséquent une dégradation de la satisfaction des clients. Nous montrons dans le reste de ce chapitre comment notre méthode traite ce problème.

Après la spécification des critères du concepteur. Si en revient à notre modèle proposé, en trouve que le service qui va échouer devient un Subject et les services qui vont être annulé ou compensé sont des Observers. D'après les critères que nous avons spécifiés pour notre exemple, en découvert trois Subjects et pour chaque Subject il y a deux observateurs.

Nous avons réalisé une application simple pour que le concepteur spécifier d'une manière flexible ses critères sous la forme d'un langage naturelle. Ces critères seront transformés et structurer dans un document XML. Ce document sera utilisé directement en cours d'exécution des processus métiers d'une manière où, si le concepteur change ses critères les changements seront pris en compte directement par les nouvelles instances des processus métiers. La figure 26 montre l'interface de cette application.

La DTD suivant définie la structure de ce document XML, qui représente les critères du concepteur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT Analyste_Criteria (Criteria)*>
        <!ELEMENT Criteria (NameSubject, Observers)>
                <!ELEMENT NameSubject (#PCDATA)>
                <!ELEMENT Observers (NameObserver)*>
                        <!ELEMENT NameObserver (#PCDATA)>
```

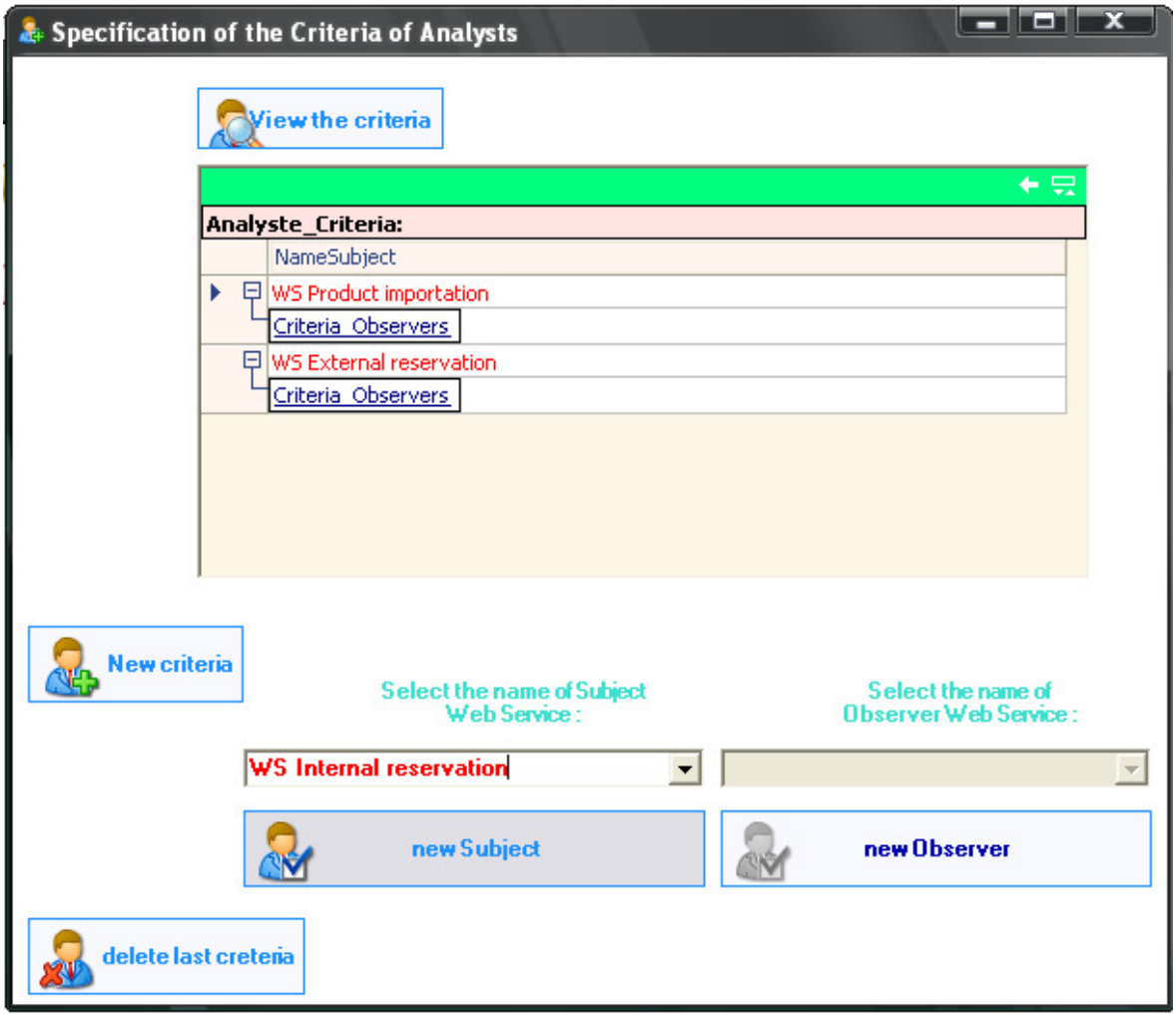


Fig. 26 Interface pour la spécification des critères des concepteurs

Le document XML résultat est un ensemble d'éléments 'Criteria' ou chaque élément représente un des critères du concepteur. L'élément contient lui-même deux autres éléments 'NameSubject' qui représente le nom du service Web Subject (qui va échouer) et 'Observers' représente la liste des noms d'observateurs de ce Subject qui devront être notifiés.

L'interface de l'application contient un bouton 'View the criteria' utilisé pour voir les critères introduits. Pour ajouter un nouveau critère en clic sur 'New Criteria' en ajoutant le nom du service Web, ensuite en clic sur le bouton 'new Sujbect'. en passe à l'insertion des services Web Observers un par un, en clic sur 'new Observer'.

Pour voir les nouveaux critères toujours en clic sur le bouton 'View the criteria'. Le résultat est apparu sur le tableau au dessous de ce bouton.

La figure 27 montre un document XML valide (conforme au DTD) qui représente les critères de notre exemple, qui sont spécifié au début :

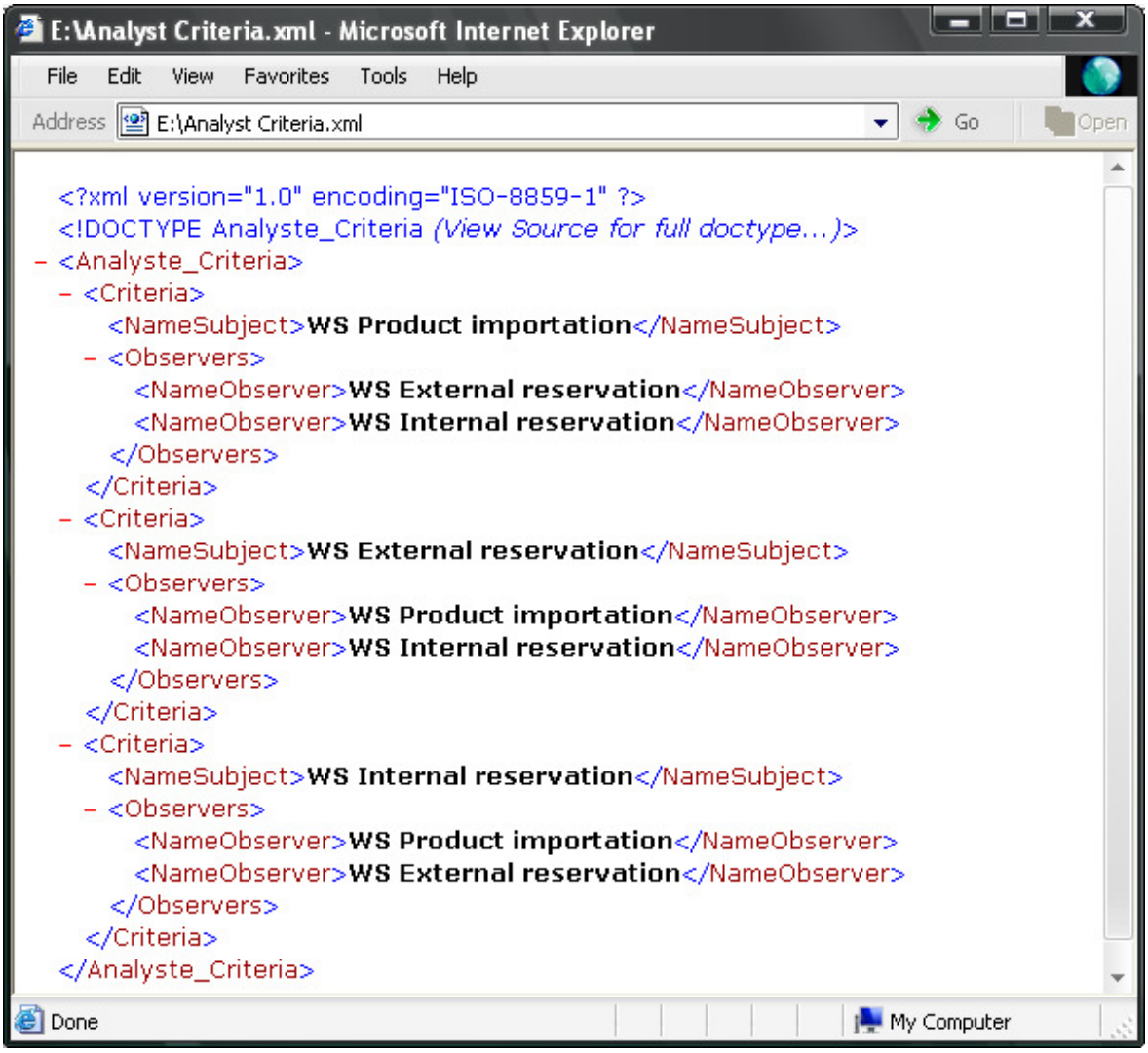


Fig. 27 Formalisation des critères de concepteurs en un document XML

3.5 Structure de notre modèle du processus métier

Le modèle du processus métier de l'agence d'importation est présenté dans figure 28. Dans ce modèle nous avons illustre si le service Web 'WS Product importation' qui préoccupe de l'achat du produit a échoué, alors selon les critères que nous avons spécifiés plus haut il doit annuler ou compenser les deux autres services Web.

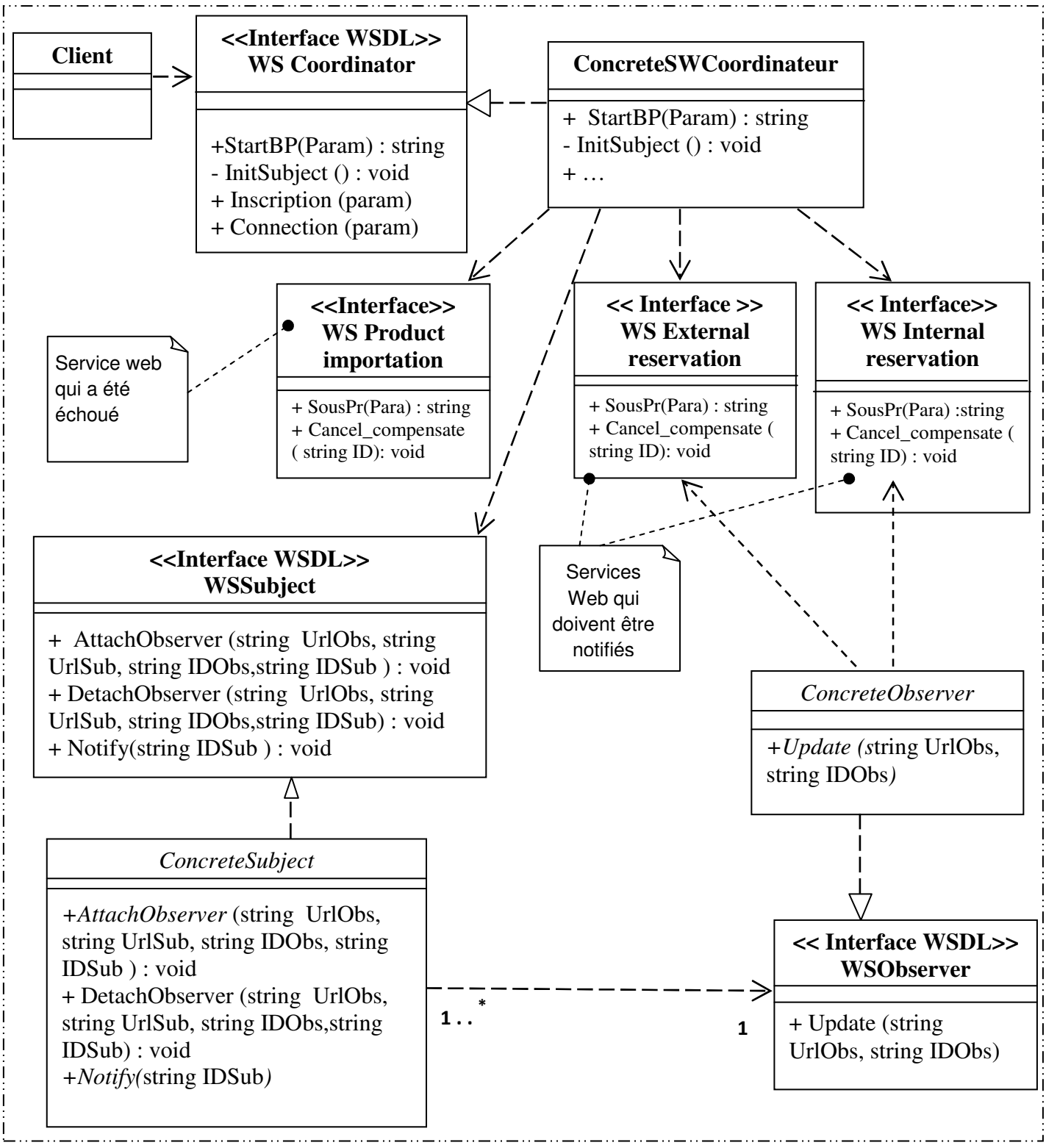


Fig. 28 Structure du modèle de conception du processus métier de l'agence d'importation

3.6 Implémentation de l'exemple

L'implémentation du processus métier consiste en la réalisation de chaque service Web. Finalement en reliant ces services selon la conception proposée.

Les services Web qui participe dans notre exemple sont :

1- Le service Web coordinateur « **WS Coordinator** » qui coordonne l'exécution du processus métier est fournit une interface au client.
2- Le service Web qui préoccupe de d'achat du produit « **WS Product importation** »
3- Le service Web qui offre la réservation du transport externe « **WS External reservation** ».
4- Le service Web qui offre la réservation du transport externe « **WS Internal reservation** ».
5- Le service Web Subject « **WS Subject** »
6- Le service Web Observer « **WS Observer** ».

Nous avons réalisé ces services utilisant l'Environnement de Développement Intégrer (IDE) Borland Developer Studio 2006. Il fournit plusieurs outils pour la construction des applications C++, C#, Delphi et des services Web utilisant le ASP .NET. Après la construction des services Web, pour tester les résultats nous avons déployé notre processus métier sur un réseau local.

La meilleure façon d'expliquer le problème traiter c'est par le déroulement d'un scénario d'exécution d'une instance de processus métier pour un client, dont nous essayons de présenter les codes des méthodes les plus importantes pour traiter le problème.

❖ **Description d'un scénario d'exécution**

La figure 29 représente une interface fournit aux clients pour commander un produit.

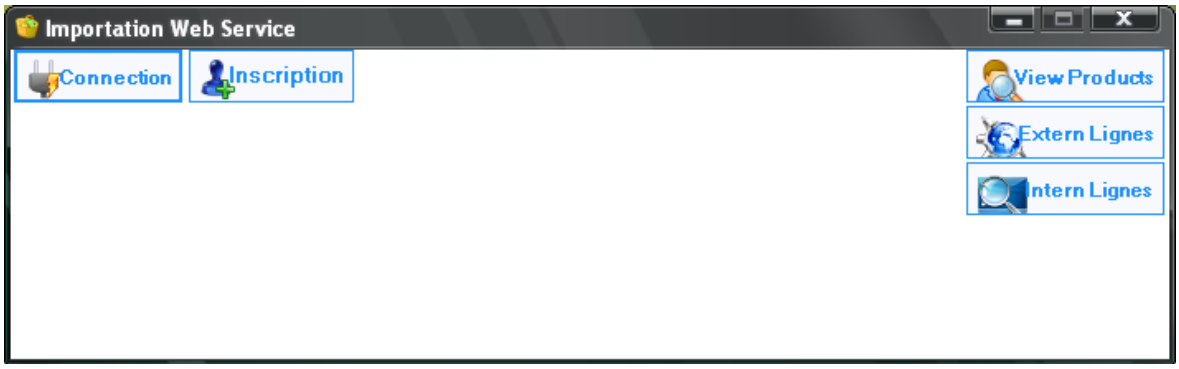


Fig. 29 Interface client

Cette figure représente une interface au service Web Coordinateur de l'agence d'importation. Les méthodes de ce service Web sont présentées dans la figure 30. Avant que le client commence à commander un produit il doit s'inscrire dans l'agence via cette interface utilisant le bouton 'inscription' (figure 31). S'il a déjà inscrit, il va connecter utilisant leur mot de passe et leur identifiant. Le client consulte les produits à vendre et les lignes de transport externe ou interne qui sont fournis par l'agence d'importation. Il utilise pour ceux-ci respectivement les trois boutons superposés dans l'interface.

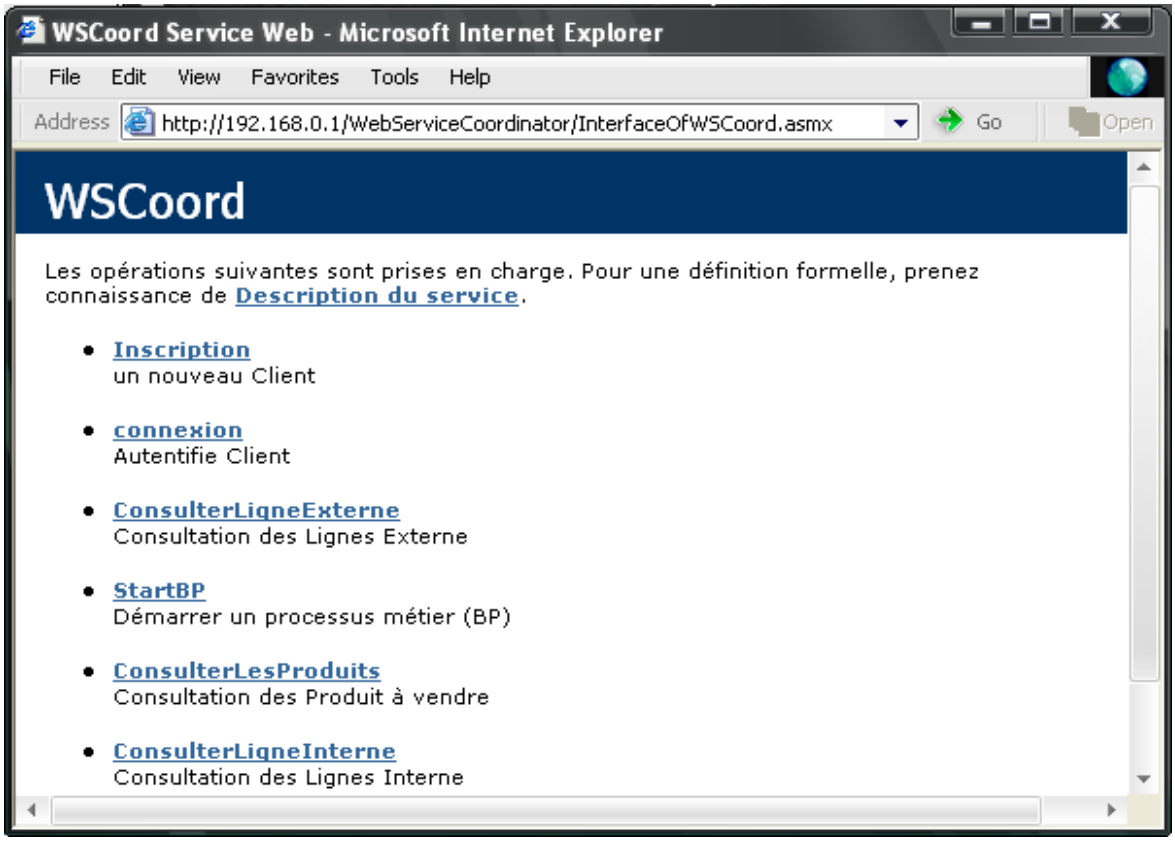


Fig. 30 L'interface du service Web Coordinateur

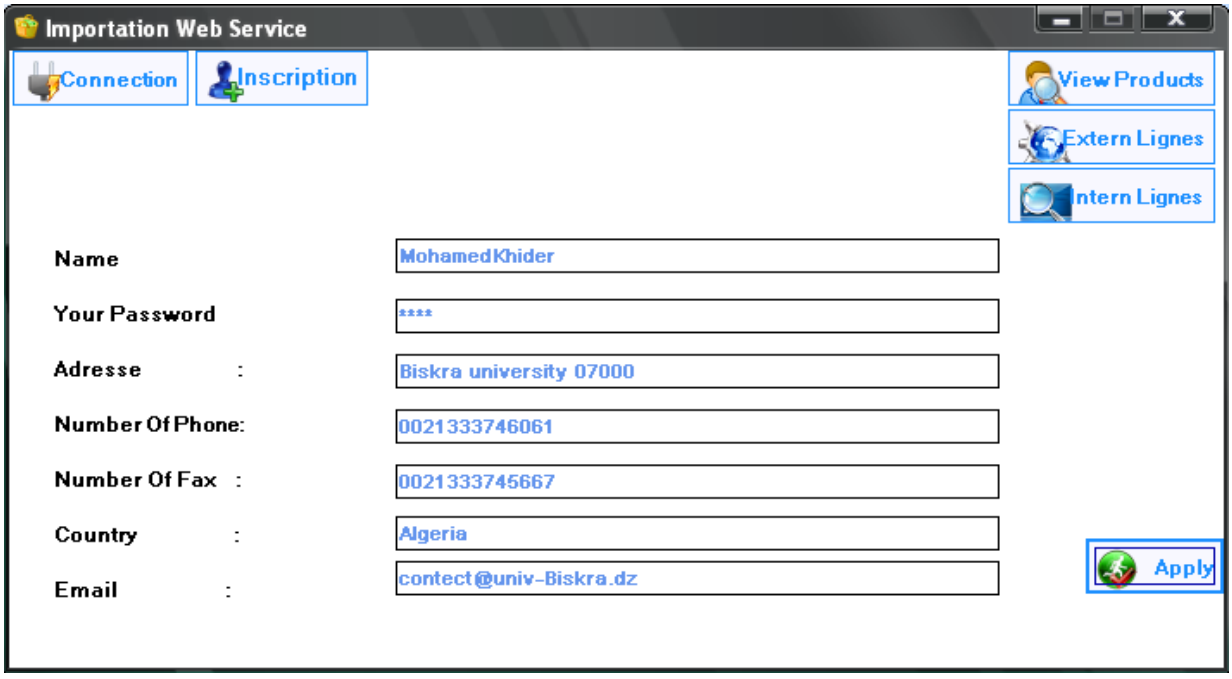


Fig. 31 Interface pour l'inscription du client

Dès que le client consulte les produits, il lance son processus métier, en clic sur le bouton 'Start the BP'. La figure 32 montre un exemple de saisi d'une commande.

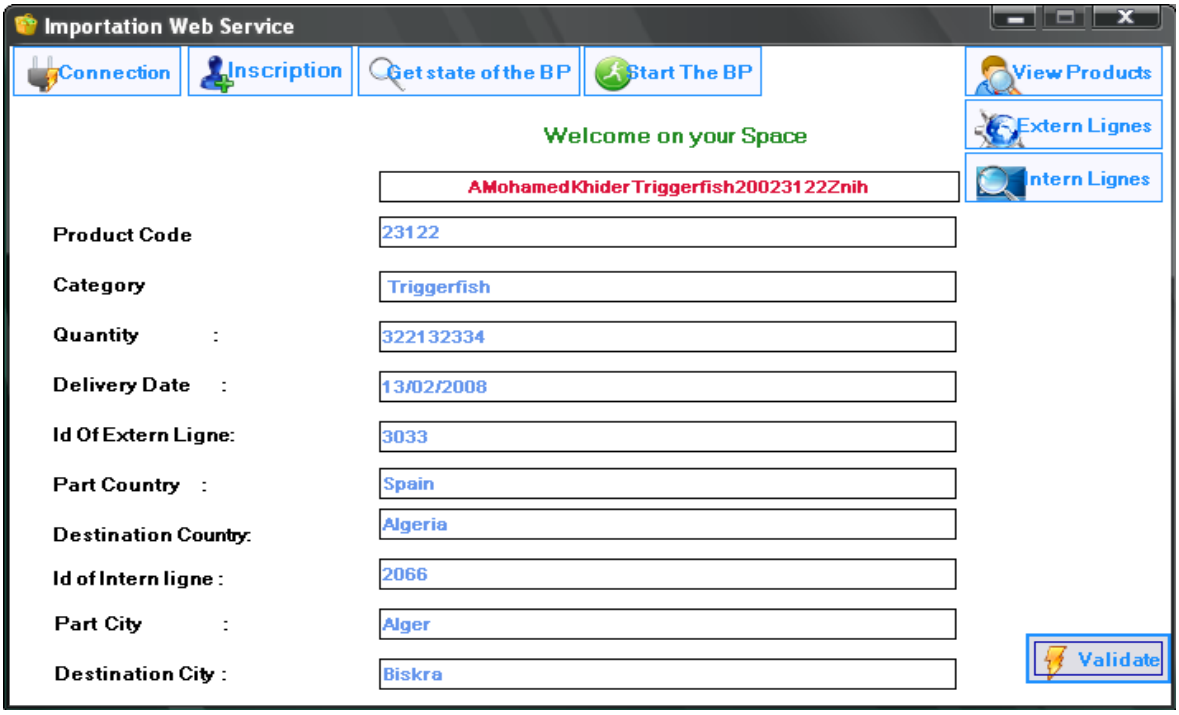


Fig. 32 Interface utilisateur pour dérouler un processus

Le client introduit leur commande et les lignes de transport. Il clic ensuite sur le bouton 'validate', qui invoque la méthode publié sur le service Web Coordinateur 'StartBP'. Cette méthode lance trois sous processus par l'invocation de la méthode 'demarrerSousPr' de chaque service. Chaque service doit retourner un identificateur (ID) de l'instance qui a été lancé. En attendant le résultat du service qui sera retourné après une certaine durée spécifique à chaque service. A ce moment chaque sous-processus lancer il retourne un 'ID'. Le service coordinateur en plus de l'invocation de ces trois sous processus il invoque la méthode 'InitSubject'. Cette dernière est utilisée pour initialiser les critères du concepteur pour cette instance du BP. Ces critères sont spécifiés dans le document XML (voir la figure 27). Voici un exemple de code en C# de la méthode 'InitSubject' :

Code 1 :

```
[WebMethod]
Void InitSubject (string id1, string id2, string id3)
{     txtReader = new XmlTextReader(DocXML_Fig26);
      reader = new XmlValidatingReader(txtReader);
      reader.ValidationType = ValidationType.DTD;
      XmlDocument doc = new XmlDocument();
      doc.Load(reader);
      reader.Close();  txtReader.Close();
       XmlElement root = doc.DocumentElement;
      WSSubject wsSub = new WSSubject()    // instance du service Web Subject
         XmlNode cre;
         XmlNode observers;
         string  idSub,urlSub,idObs,urlObs;
```

```
        for (int i = 0; i < root.ChildNodes.Count; i++) // for all criteria in element of doc XML
        { cre= root.ChildNodes[i];                    // cre prend un critère
            nameSub = cre.FirstChild.InnerText;
          idSub = returnIdSub(nameSub, id1,id2,id3);
          urlSub = returnUrlObs (nameSub);
          observers = cre.LastChild;        // les observers du subject
          for (int j = 0; j < observers.ChildNodes.Count; j++) // for all observers in Childs of criteria
        {   nameObs = observers.ChildNodes[j].InnerText;
            idObs = returnIdObs(nameObs, id1,id2,id3);
            urlObs = returnUrlObs (nameObs);
          wsSub →AttachObserver(urlObs, urlSub, idObs, idSub, nameObs , nameSub)
        }
        }
}
```

Les paramètres de la méthode correspondent aux identificateurs de chaque sous processus lancé. La première boucle parcourt les critères du document XML, pour chaque critère en prend le nom du Subject en lui associant leurs Url et l'ID approprié parmi les trois paramètres. La deuxième boucle parcourt les observateurs de ce Subject qui sont définis dans le document XML, en lui aussi associant leurs ID et leurs Url. En suite en invoquant la méthode attachObserver du service Web Subject (figure 33). Cette méthode doit créer un document XML qui doit être conforme au DTD définie dans section 5.2 du chapitre précédent, un exemple d'un des critères dans ce document pour le client introduit précédemment est le suivant :

```
<Criteria>
 <Subject>
    <NameSubject>WS Product importation</NameSubject>
     <UrlSubject>http://192.168.0.3//WebServiceProduct//WebSerPro.asmx</UrlSubject>
     <IdSubject>AMohamedKhiderTriggerfish20023122Z</IdSubject>
</Subject>
 <Observers>
   <Observer>
     <NameObserver>WS External reservation</NameObserver>
      <UrlObserver>http://192.168.0.2//WebServiceBateau//WSExterneTransport.asmx
</UrlObserver>
       IdObserver>ZSpainAlgeria200MohamedKhider3033A</IdObserver>
   </Observer>
    <Observer>
     <NameObserver>WS Internal reservation</NameObserver>

<UrlObserver>http://192.168.0.4//WebServiceApplication1//WSLocalTransport.asmx</UrlObserver
>
      <IdObserver>KAlgerBiskra200MohamedKhider2066I</IdObserver>
  </Observer>
 </Observers>
</Criteria>
```

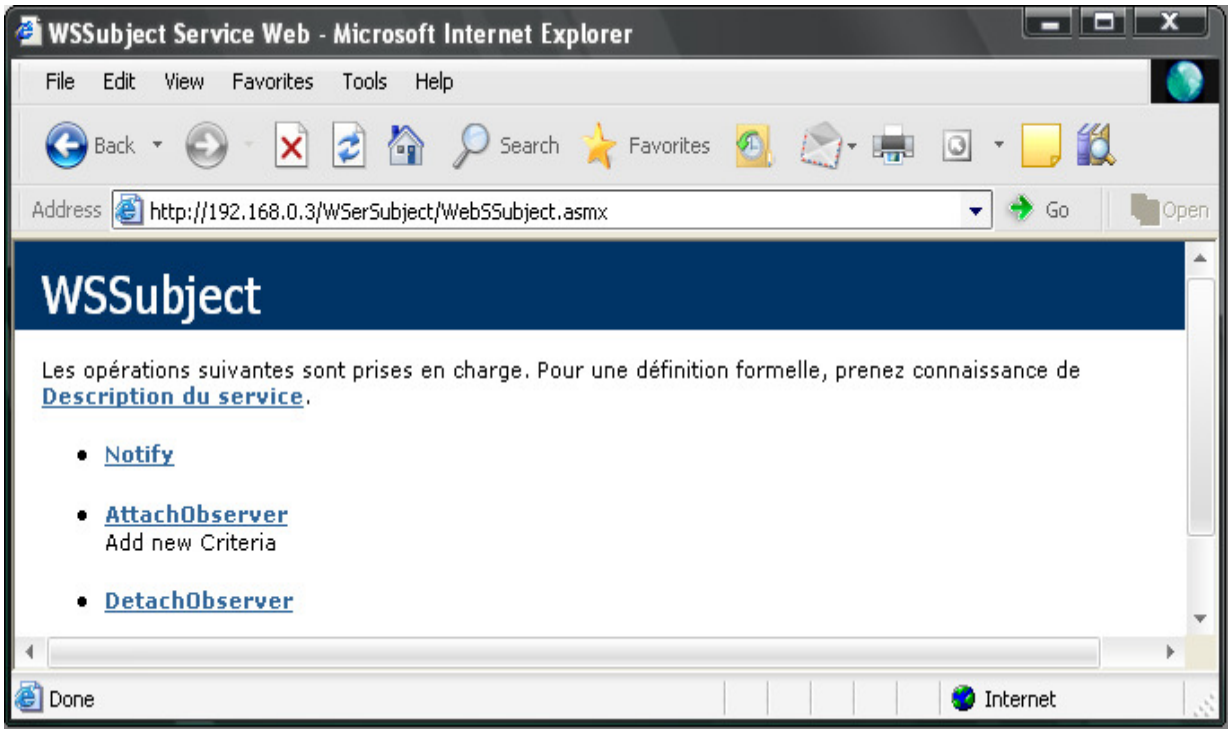
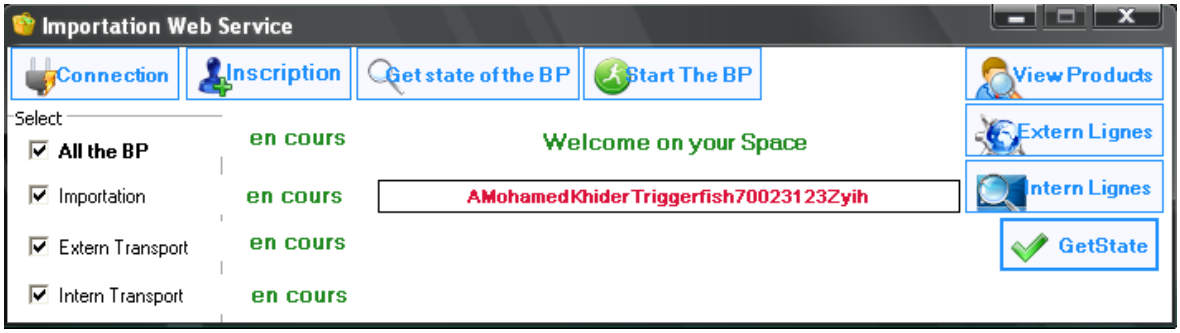


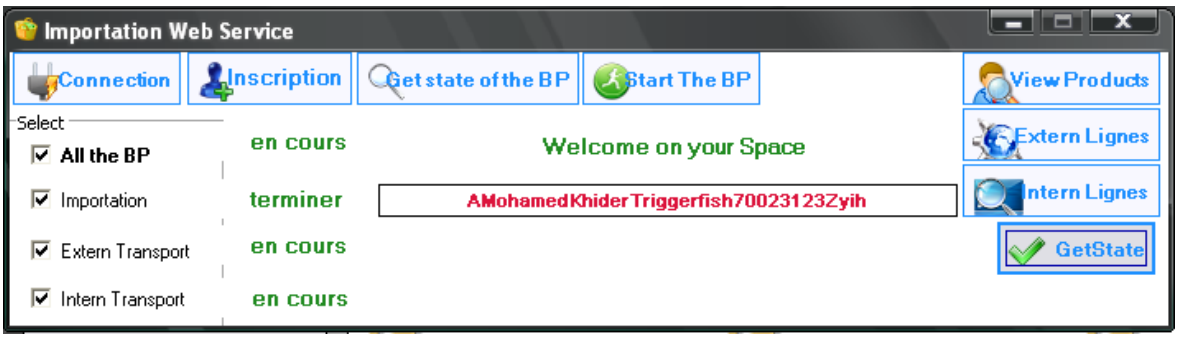
Fig. 33 Interface du notre service Web Subject

Après l'initialisation des critères du concepteur la méthode 'StartBP' (dans la figure 30) envoi au client un identificateur du BP, pour notre exemple il est apparu en rouge dans la figure 32. Le client attend le résultat de chaque service. Supposant que le service d'achat du produit prend deux semaines pour retourner leur résultat du service et le service de réservation du transport externe et interne prennent respectivement trois semaines et quatre semaines. Dans le cas normal ou touts les services se terminent avec succès la séquence de figure 34 montre les changements d'états de chaque service, initialement les trois services prennent l'état 'en cours'.

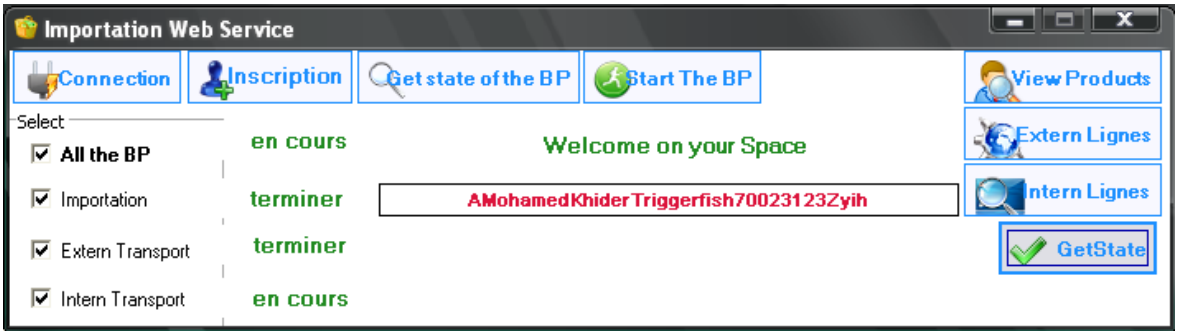
L'état initial :



Après deux semaines :



Après trois semaines :



Après quatre semaines :

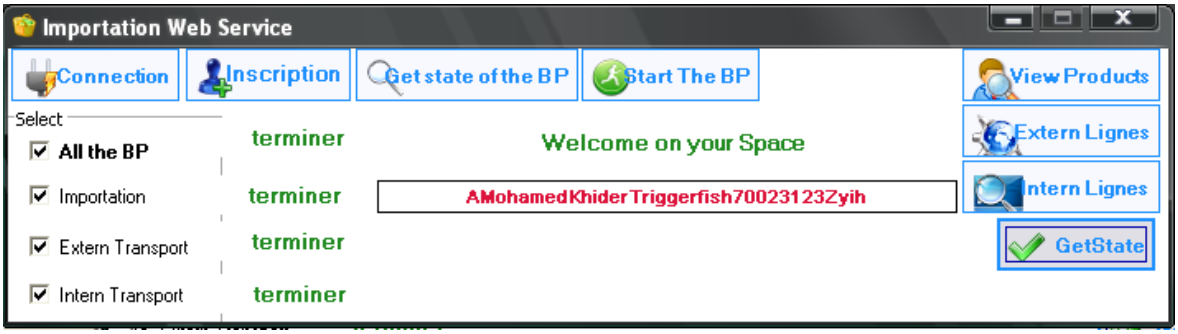


Fig. 34 Changement d'état d'une instance BP qui se termine sans échecs

Cependant, si l'exécution du service d'achat du produit (WS Importation) est échouée, le processus doit être abandonné, et en annulant les deux autres services immédiatement, la figure 35 montre le résultat après deux semaines pour une autre instance BP:

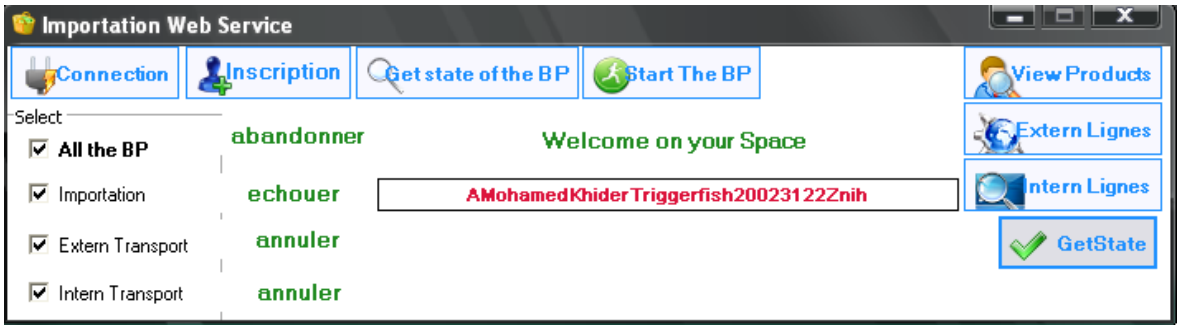


Fig. 35 L'état d'une instance qui se termine par un échec

Cependant, avec ce résultat nous avons arrivé traité le problème l'échec du service d'achat, tout en respectant les critères celles que nous avons définis précédemment. Réellement, le déroulement de l'exécution est de la manière suivante : le service d'achat du produit renvoi son état d'échec au service coordinateur, ce dernier abandonne l'exécution du BP et invoque la méthode 'notify' du service Web Subject (figure 33). Cette méthode prend en paramètres l'identificateur du service qui a été échoué, ensuite elle cherche dans le document XML (le résultat de l'invocation de la méthode 'InitSubject') l'élément qui contient un IdSub égal à ce paramètre. Pour chaque observateur de ce Subject en invoque la méthode 'update' (figure 36) par l'instanciation du service Web Observer. La méthode 'update' prend en paramètre l'URL du service à notifié et l'identificateur de l'instance. Voici un exemple de code qui explique mieux la méthode 'notify' :

Code 2 :

```
[WebMethod]
Void notify (string IdSub)
{
    txtReader = new XmlTextReader(DocXMLCreteria);
    reader = new XmlValidatingReader(txtReader);
    XmlDocument doc = new XmlDocument();
     reader.ValidationType = ValidationType.DTD;
       doc.Load(reader);  reader.Close(); txtReader.Close();
      WSObserver wsObs;
       XmlElement root = doc.DocumentElement;
      XmlNode cre, subject, idsubject, observers, observer, urlobserver, idobserver;
      for (int i = 0; i < root.ChildNodes.Count; i++)
        {
            cre = root.ChildNodes[i];
            subject = cre.ChildNodes[0];
            idsubject = subject.ChildNodes[2];
            if (idsubject.InnerText == IdSub)  //comparé avec l'Id du Subject dans le paramètre
            {    observers = cre.ChildNodes[1];
                 for (int j = 0; j < observers.ChildNodes.Count; j++) // parcourt la liste des Obsvateurs
                 {
                       observer = observers.ChildNodes[j];
                       urlobserver = observer.ChildNodes[1];
                       idobserver =  observer.ChildNodes[2];
                       WsObs = new WSObserver(); // instance du service Web Observer
                       WsObs.Update (urlobserver.InnerText,idobserver.InnerText);
                 }
                      i =root.ChildNodes.Count;
              }
        }
}
```

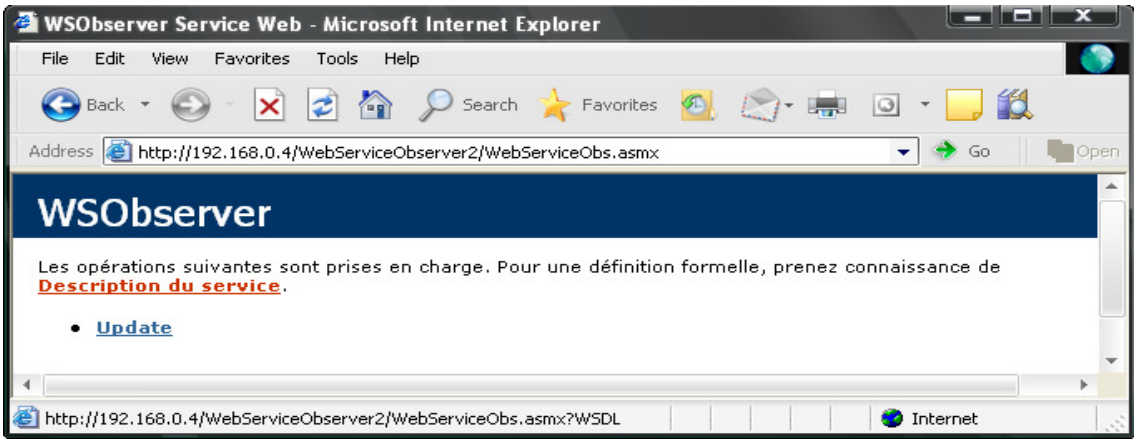


Fig. 36 Interface du notre service Web Observer

La méthode 'update' utilise le paramètre 'UrlObs' qui représente l'URL du service à notifier. Après l'importation de leur interface la méthode 'update' invoque elle-même l'opération 'Cancel_compensate' du service à notifié. Selon le paramètre de la méthode 'Cancel_compensate' qui est l'identificateur de l'instance du service Web, ce service réalise la mis-à-jours automatique de leurs état. Si elle est en cours d'exécution l'état devient 'annuler'. Si il termine sont exécution il prend le service l'état 'compenser' et fait le traitement local de compensation.

En fin chaque sous processus envoi leur nouvelle état au service Web coordinateur. Ce dernier informe le client que le processus de l'importation est annulé. De cette manière nous arrivons à terminer le scénario d'exécution d'un BP.

4. Conclusion

Nous avons présenté dans ce chapitre, un exemple concret d'application de notre méthode pour la conception des processus métier. L'exemple que nous avons choisi c'est une agence d'importation qui permet aux clients de effectuent des importations via une interface graphique sur le Web. À travers cet exemple nous avons démontré notre modèle qui est à la base du Pattern Observer pour la conception de ce processus métier. Nous avons réalisé ce processus métier par une par orchestration de services Web. En outre nous avons réalisé une application simple pour faciliter la spécification des critères des concepteurs d'une manière flexible, dont nous avons démontré la manière d'agir aux échecs d'exécution de certains services selon les critères que nous avons définis.

Cependant, une entreprise suit notre méthode pour le développement de ses processus métiers, elle minimise les coûts de développement, augmenter l'efficacité de la solution et enfin elle gagne la satisfaction des clients et en minimisant les pertes. Ces critères économiques sont très importants pour qu'une entreprise souhaite prospérer.

Conclusion générale et perspectives

L'objectif de ce travail était de proposer une méthode de conception des processus métiers basé sur Pattern Observer. Les processus considérés dans ce travail sont réalisés par la composition des services Web. Puisque, les approches classiques de développement des processus métiers ne permettent pas d'atteindre des modèles efficaces répondant à des critères bien définis par les concepteurs, ces critères décrivent la manière d'agir contre les échecs d'exécution de certaines activités composants les processus métiers. Notre méthode permet d'assurer un traitement automatique des problèmes qui apparaissent durant l'exécution de ces processus. La solution était un modèle basé sur la réutilisation du Pattern Observer, ce dernier représente une solution conceptuelle efficace pour une classe de problèmes donnée. Nous avons réutilisé le Pattern Observer pour proposer un modèle efficace et réutilisable qui traite les échecs d'exécution des services Web composants un processus métier.

Nous nous somme menés dans la première partie de ce mémoire à étudier le domaine des processus métiers (BPM), ainsi que les composants réutilisables, plus particulièrement les composants de type patron pour construire des solutions conceptuelles réutilisables et efficaces (efficace car elles sont réalisés et approuvées par plusieurs développeurs experts), dans une étape suivante nous nous somme analysé les services Web qui sont des standards utilisés principalement pour traiter les problèmes d'hétérogénéités.

Cependant, la complexité du domaine du BPM présente un frein aux concepteurs pour développer des processus métier fiable. Néanmoins, la réutilisation des modèles conceptuels qui peuvent décrire ces processus est une solution prometteuse pour maîtriser cette complexité.

Partant de ce constat, les objectifs fixés s'articulent autour des points principaux : fournir une démarche pour orienter le concepteur dans le développement de ces processus métiers, offrir une flexibilité aux concepteurs pour spécifier leurs besoins en termes de critères de fiabilité, proposer un modèle basé sur le Pattern Observer permettant de concrétiser les critères des concepteurs et enfin, de permettre l'utilisation des services Web pour la réalisation des processus métiers.

L'approche proposée a été validé en l'appliquant sur un exemple concret d'un processus métier. Cet exemple concerne la gestion d'une agence d'importation qui fournit trois services Web. En outre, nous avons réalisé une application pour la définition des critères des concepteurs d'une manière simple. Enfin, nous avons démontré la manière d'agir aux problèmes d'exécution selon un ensemble de critères que nous avons définis.

Une perspective au travail proposé consiste en l'enrichissement de notre méthode pour traiter d'autres types de critères du concepteur. Par exemple après l'échec d'un des services composants un processus métier, avant que le service Subject notifie les autres services sur cet échec, nous ajoutons une opération au service Subject, qui consiste en la recherche dans UDDI d'un autre service Web qui retourne un service identique au service échoué pour le remplacer. S'il y un résultat après la recherche, le processus continue son exécution. Dans

le cas contraire, on invoque le la méthode ‘notify’ du service Subject pour annuler les autres services et abandonner le processus. En outre, notre travail futur vise à fournir un outil formel de simulation pour vérifier et valider formellement notre modèle avant que le processus conçu soit implémenté.

Bibliographie

[1] **Ouafa Hachani** « Patrons de conception à base d'aspects pour l'ingénierie des systèmes d'information par réutilisation » Thèse de doctorat au sein de l'université JOSEPH FOURIER-Grenoble I, soutenue le 4 Juillet 2006.

[2] **Assia AIT ALI SLIMANE, Muhammad Usman BHATTI** « Utilisation des services et des aspects pour la réutilisabilité du logiciel d'un automate pour l'analyse de plasma » Université Paris 1 Panthéon Sorbonne 2006

[3] **Oualid KHAYATI** « Modèles formels et outils génériques pour la gestion et la recherche de composants » Thèse de doctorat de l'institut national polytechnique de Grenoble (INPG), Le 17 décembre 2005

[4] **Mhamed SAIDANE** « Formalisation de familles d'architectures logicielles coopératives une démarche, modèles, outils » Thèse de doctorat au sein de l'université JOSEPH FOURIER-Grenoble I, soutenue le 1 Décembre 2005.

[5] **Agnès FRONT** « Développement de systèmes d'information à l'aide de patrons » Thèse de doctorat au sein de l'université JOSEPH FOURIER-Grenoble I, soutenue le 13 Décembre 1997

[6] **Robin Passama**« Conception et développement de contrôleurs de robots, Une méthodologie basée sur les composants logiciels » Thèse préparée au sein du Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, soutenue le 30 Juin 2006.

[7] **E. Gamma, R. Helm, R. Johnson, and J. Vlissides,** « Design Patterns: Elements of Reusable Object-Oriented Software», Addison Wesley, Reading, MA, 1995

[8] **Conte A., Fredj M., Giraudin J-P, Rieu D., P-Sigma** : « Un formalisme pour une représentation unifiée de patrons », Inforsid'01, Martigny, Juin 2001.

[9] **Object Management Group** (OMG) - Corba Component Model (CCM) Specification 3.0. http://www.omg.org/technology/documents/formal/components.htm, 2002.

[10] **Sun** Microsystems - Enterprise JavaBeans (EJB) Specification 2.1
http://www.sun.com/, 2003.

[11] **Ambler S. W,** «An Introduction to Process Pattern», AmbySoft White paper, 1998. http://www.Ambysoft.com.

[12] **Ashutosh Raut, Ashwin Basavaraja** « Enterprise Business Process Integration», TENCON 2003 IEEE page 1549/1553

[13] **Douglas C. Schmidt,** « Using Design Pattern to develop Reusable Object-Oriented communication Software», October 1995/vol. 38, no. 10 Communications of the ACM, pages 65-74.

[14] **S. Bhiri** « Approche Transactionnelle pour Assurer des Compositions Fiables des services Web » Thèse au sein de l'université Henri Poincaré Nancy 1, soutenue le 06/10/2005.

[15] **John W. Satzinger, Robert B. Jackson, Stephen B. Burd** « Analyse et conception de systèmes d'information » REYNALD GOULET, deuxième édition 2003

[16] **Abdelmadjid KETFI,** « Une approche générique pour la reconfiguration dynamique des applications à base de composants logiciels», Thèse au sein de l'Université Joseph Fourier de Grenoble, soutenue le 10 décembre 2004.

[17] **Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates** «Head First Design Patterns» O'Reilly 2007 www.oreilly.com

[18] **Christain, DONZEL** « Identification des besoins et choix d'outils informatiques : réponses aux impératifs des utilisateurs de Sécheron SA » Mémoire en vue de l'obtention du diplôme post-grade en informatique et organisation, Université de Lausanne 2002-2004.

[19] **Tanguy Crusson** « Business Process Management, De la modélisation à l'exécution Positionnement par rapport aux Architectures Orientées Services » 2003 www.intalio.com

[20] « La maîtrise des données est acquise. Et si l'avenir reposait sur la maîtrise des Processus » Livre Blanc Akazi technologies www.akazi.com

[21] **Geoffrey Sparks** « An introduction to modelling software systems using the UML: The Business Process Model » *Geoffrey Sparks 2000* www.sparxsystems.com.au

[22] **Walid Gaaloul** « La Découverte de Workflow Transactionnel pour la Fiabilisation des Exécutions » Thèse au sein de l'université Henri Poincaré Nancy 1, soutenue le 03/11/2006.

[23] **Jean-Marc Geib, Christophe Gransart, Philippe Merle** « CORBA des concepts à la pratique » DUNOD, Paris, deuxième édition 2000

[24] **Nerea Arenaza**, « Composition semi-automatique de Services Web » SIN Projet de Master Février 2006, Ecole Polytechnique Fédérale de Lausanne

[25] **Werner Vogels** « Web Services Are Not Distributed Objects » IEEE Internet Computing, Published by the IEEE Computer Society November - December 2003 pages 59-66.

[26] **Michael Havey,** «Essential Business Process Modeling», O'Reilly Media, Inc., 2005.

[27] **Ashok K. Harikumar & Roger Lee, Hae Sool Yang, Haeng-Kon Kim, Byeongdo Kang,** « A Model for Application Integration using Web Services», Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICCIS'05), 2005 IEEE

[28] **Gilbert Babin, Michel Leblanc,** « Les Web services et leur impact sur le commerce B2B», Rapport Bourgogne, CIRANO (Centre Interuniversitaire de Recherche en Analyse des Organisations), (Août 2003).

[29] «L'EAI (Enterprise Application Intégration) », www.seralia.com

[30] **Jeremy Allaire,** « Macromedia MX : Composants et services Web » White paper Avril 2002

[31] **Eric NewComer,** «Understanding Web Services- XML, WSDL, SOAP and UDDI» (Addison Wesley) 2002

[32] **Brian Benz, John R. Durant, Tod Golding,** «XML Programming Bible » 2003 by Wiley Publishing, Inc., Indianapolis, Indiana, www.wiley.com

[33] **Pierre-Yves Cloux David Doussot Aurélien Géron,** « Technologies et architectures Internet Corba, COM, XML, J2EEE, NET, WEB services », 2ème édition, DUNOD, Paris 2002.

[34] **Nguyen Duycu,** « Accès aux contenus du système iClass grâce à un Adaptateur utilisant la "SQI" » Mémoire de fin d'études, Bruxelles, Belgique – 2005

[35] **Sonia JAMAL,** « Environnement de procédé extensible pour l'orchestration Application aux services Web » Thèse préparée au sein du Laboratoire LSR – Equipe ADELE (Grenoble I), soutenue le 13 décembre 2005

[36] **Borland Software Corporation** « (Java - Borland) Jbuilder 9 - Web Services Developer's Guide», 2002-2003, www.borland.com

[37] **Mike Rosen and Jim Boak** « Developing a Web Services Strategy » *E*AI Journal January 2002 pages 39-43.

[38] **Georges Gardarin** « XML Des bases de données aux services Web » Dunod, Paris 2002.

[39] **Philippe Giaccari** « Gestion des Processus Business et Modélisation des Processus Business d'une Start-up de Type Cybermédiaire » Mémoire en vue de l'obtention du Diplôme postgrade en informatique et organisation, Université de Lausanne 2001/2002.

[40] **ABDMOULEH Anis** « Composants pour la Modélisation des Processus Métier en Productique, basés sur CIMOSA » Résumé de thèse de l'Ecole Nationale d'Ingénieurs de METZ (ENIM), soutenue le 15 septembre 2004.

[41] **Geert Van de Putte, Tony Benedetti and al** « Intra-Enterprise Business Process Management » IBM Corporation, International Technical Support Organization October 2001.www.ibm.com/redbooks.

[42] « E-Business–Workflow, Introduction au Workflow » L'encyclopédie en ligne www.commentcamarche.net

[43] **Thomas Erwin** « Performance analysis of Business Process », In Proceedings of the Conference on Petri Nets and Business Processes, 06/07, - 10/07/1998. Page 20

[44] **Phong Nguyen** « Cours de Modélisation et Vérification, TP de Business Process Modeling » CUI, Université de Genève 2007.

[45] **Marlon Dumas and Arthur H.M. ter Hofstede** « UML Activity Diagrams as a Workflow Specification Language » In proceeding of the UML'2001 Conference.

[46] **P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell,** « Pattern-based Analysis of UML Activity Diagrams » BETA Working Paper Series, WP 129, Eindhoven University of Technology, Eindhoven, 2004.

[47] **Stephen A.** « Introduction to BPMN » White, BPM Architect, IBM Software Group, WebSphere software, October 16, 2006.

[48] **Paul B. Monday** « Web Service Patterns: Java Edition» Apress, first edition, 2003.

[49] **Piyush Maheshwari** «Enterprise Application Integration using a Component-based Architecture» Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC'03), 2003 IEEE