Contribution des “scénarios patterns” et du raisonnement à partir de cas à la modélisation conceptuelle des applications logicielles

Bilal Hussein

Université de Valenciennes et du Hainaut-Cambrésis

Numéro d'ordre : 08-27

THESE

présentée à

L'UNIVERSITE DE VALENCIENNES ET DU HAINAUT-CAMBRESIS

pour l'obtention du

DOCTORAT
Sciences et Technologies Informatique

par

Bilal HUSSEIN
Ingénieur d'état

Contribution des « *scénarios patterns* » et du raisonnement à partir de cas à la modélisation conceptuelle des applications logicielles

Application à la gestion bancaire

Soutenue publiquement le 9 décembre 2008 devant le Jury composé de :

Alain MILLE	Université Claude Bernard Lyon 1	Rapporteur
Abdelhamid MELLOUK	Université de Paris 12 Val-de-Marne	Rapporteur
Imad SALEH	Université de Paris 8	Examinateur
Patrick MILLOT	Université de Valenciennes	Directeur de thèse
Patrice CAULIER	Université de Valenciennes	Co-directeur de thèse
Youssef MONSEF	Université Libanaise	Co-directeur de thèse
Yahia RABIH	Université Libanaise	Invité

Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines
UMR CNRS 8530

Avant-propos

Le travail présenté dans ce mémoire a été réalisé au Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines de l'Université de Valenciennes et du Hainaut-Cambrésis, au sein de l'équipe Système Homme-Machine et en collaboration avec Monsieur Youssef MONSEF, Professeur des Universités à la Faculté de génie de l'Université Libanaise.

Je tiens à remercier chaleureusement Monsieur le Professeur Patrick MILLOT, Vice-Président Recherche de l'Université de Valenciennes et du Hainaut-Cambrésis, d'avoir accepté d'être le directeur de ma thèse.

Je remercie le directeur de l'équipe Système Homme-machine Monsieur Frederic VANDERHAEGEN, Professeur des Universités, pour m'avoir accueilli dans son équipe.

Mes remerciements les plus chaleureux vont tout naturellement à mes co-directeurs de thèse Monsieur Patrice CAULIER, Maître de conférences à l'Université de Valenciennes et du Hainaut-Cambrésis et Monsieur Yousef MONSEF.

Je tiens à remercier Monsieur Alain MILLE, Professeur des Universités à l'Université Claude Bernard Lyon 1, et Monsieur Abdelhamid MELLOUK, Professeur des Universités à l'Université de Paris 12 Val-de-Marne, d'avoir accepté d'être les rapporteurs et qui par leurs remarques et leur conseil ont permis l'aboutissement de cette thèse.

Je remercie aussi Monsieur Imad SALEH, Professeur des Universités à l'Université de Paris 8, d'avoir accepté d'examiner ma thèse.

Je suis encore reconnaissant envers les personnes qui ont contribué par leurs idées et leurs assistances à ma recherche : Yahia RABIH, Maître de conférences à l'ISAE de l'Université Libanaise, Aref MHANNA Enseignant Chercheur à l'Université Polytechnique de Bucarest et Abdallah EL-ASMAR ingénieur en informatique à l'ISAE.

Au-delà de tout, je dois à ma famille, ma femme Mme. Zahia TARMOUL, mes filles Sarah et Mélissa beaucoup de reconnaissance. Ses encouragements m'ont aidé à surmonter les étapes les plus difficiles et à continuer sans peine le projet de cette thèse. Ainsi, je leur dédie cette thèse comme expression de ma profonde reconnaissance.

Valenciennes, le 9 décembre 2008.

Table des matières

Introduction générale

Contexte du travail et motivations

Situé dans le domaine de l'ingénierie des besoins, la thèse concerne la contribution de la notion de « *scénario pattern* » et du Raisonnement à Partir de Cas (RàPC) à la construction automatisée du modèle d'analyse statique des applications logicielles complexes.

Le thème de ce travail vise à développer un environnement de modélisation intelligent à base de connaissances et à raisonnement à partir de cas. Le but de cet environnement est de permettre à un ingénieur système de générer d'une façon automatique un modèle de conception objet, formalisé en UML, à partir de l'expression des besoins utilisateurs.

L'idée de base de cette thèse est de trouver une façon de remédier à la problématique définie par la préexistence d'un fossé entre les besoins réels vis-à-vis du système et les services finalement rendus par l'application informatique.

En effet, un grand nombre d'études [Lubars *et al.* 93], [Standish 95], [McGraw et Harbison 97], [Leishman et Cook 02], [Sánchez-Alonso, Murillo et Hernández 04] ont montré que les échecs dans la mise en œuvre et l'utilisation des systèmes informatiques sont dus à une mauvaise compréhension des besoins auxquels ces systèmes tentent de répondre.

L'ingénierie des besoins constitue le cadre méthodologique pour résoudre ce type de problématique. Elle comporte différentes méthodes et modèles aidant la capture et l'analyse des besoins. Ces outils reposent sur des concepts et des vues différentes :

- L'utilisation de différents points de vue sur le système pour effectuer l'analyse.
- La modélisation des besoins à travers la description des flux de données et de leurs traitements (*Data-flow diagrams*).
- L'identification et la modélisation des données inhérentes aux besoins (modèle entités - relations).
- L'analyse orientée objet [Rumbaugh et *al.* 98], [Booch 94], [Coad et Yourdon 91].
- La création de dictionnaires des données et des traitements.
- L'utilisation d'un langage dérivé de la programmation.
- La capture des besoins par prototypages successifs.
- Les méthodes formelles telles que : Z, VDM et les algèbres de processus, ...

La spécification des besoins par l'une des méthodes citées ci-dessus fournit un modèle des besoins utilisateurs et décrit ce que le système futur doit faire. Ce modèle

des besoins peut ne pas représenter le système tel qu'il est dans le monde réel et ceci est dû à une spécification incomplète.

Un état de l'art sur les méthodes d'ingénierie des besoins (chapitre 1) a permis de mettre en évidence l'absence de définition précise d'une démarche d'aide à la construction de modèles de besoins.

Problématique de la thèse

L'analyse des besoins, la spécification, la conception, la validation et la vérification sont parmi les plus grands défis que doivent surmonter les ingénieurs système. Dans de nombreux énoncés des besoins du logiciel, les exigences sont incomplètes, ambiguës et contradictoires. Dans beaucoup de cas, les besoins sont spécifiés à un niveau trop haut ou aussi trop bas du système ou au niveau de la conception, et non pas au niveau de l'analyse des exigences du logiciel. Si ces défis sont relevés le risque de développer des systèmes qui ne satisfont pas les exigences sera atténué.

L'évolution permanente des besoins métier de l'entreprise et des technologies rend les systèmes informatiques de plus en plus complexes et rapidement obsolètes. En outre, cela génère des coûts de maintenance et de migration insupportables pour la plupart des entreprises [Kadima 05].

Il est reconnu que ce sont les premières étapes du développement du logiciel qui sont cruciales pour parvenir à la meilleure adéquation entre les besoins exprimés et la réalisation proposée [Kadima 05]. Différentes approches sont proposées pour aller dans ce sens avec leurs avantages et leurs inconvénients respectifs.

La notation UML [UML 03] a le bénéfice de la diffusion, de concepts ayant fait leur preuve dans le monde industriel, de notations graphiques, mais aussi l'inconvénient de l'absence de sémantique formelle ce qui limite la portée de tout guide méthodologique.

Les spécifications formelles pallient ce problème et, par leur expression précise, conduisent à se poser des questions qui font progresser dans la compréhension du problème à traiter. Toutefois, un écueil demeure celui de la taille et/ou de la complexité de la spécification, qui rend malaisée la compréhension synthétique et peut égarer dans la démarche [Kadima 05].

Les "patterns" (schémas ou patrons) [Choppy et Heisel 04] présentent des modèles de conception "bien connus" nous faisant profiter de l'expérience. Les "patterns" peuvent être vus comme un moyen élaboré de réutiliser des connaissances acquises par l'expérience. Toutefois, les design patterns [GoF *et al.* 95] n'offrent qu'une réutilisabilité informelle. Ils nomment et décrivent une solution (ici un design) générique à des problèmes récurrents. Il appartient au programmeur de les référencer, de les suivre, de les adapter et de les utiliser au sein de son application [Rapicault 99].

Enfin, le développement dirigé par les modèles MDA (Model driven Architecture) apparaît comme un moyen élaboré pour construire des modèles métiers indépendamment de toute plate-forme et assure la transformation de ces modèles en modèle technique dépendant de plates-formes [Miller et Mukerji 03]. Les modèles métiers proposés par MDA sont représentés par des cas d'utilisation au formalisme UML. Toutefois, les problèmes liés à l'incompréhension des scénarios associés aux cas d'utilisation, l'oubli de certains cas d'utilisation ou, encore, la mauvaise représentation de quelques contraintes restent toujours posés [Hussein et *al.* 06 (b)].

Aujourd'hui, il n'existe aucune technique permettant de valider le choix de la méthode ou le modèle de développement avant que les résultats du système ne soient disponibles.

Des questions récurrentes demeurent, quelle méthode peut-on choisir pour ce type de problème? Quel modèle est-il le plus pertinent pour la méthode choisie (spirale, cascade, V, W, prototypage, 4eme génération, etc.....)? Pourquoi a-t-on besoin toujours de commencer un développement de logiciel dès le début de son cycle de vie?

Objectifs de la thèse

Face à cette problématique, le principal objectif de cette thèse est de proposer un environnement de développement permettant de guider l'ingénieur système dans l'élaboration automatique de modèles de conception objets (modèle d'analyse). Cet Environnement de Modélisation Intelligent (EMI) inclut une base de connaissances des scénarios métier et des outils de Raisonnement à Partir de Cas (RàPC).

L'EMI permet à un ingénieur système de générer d'une façon automatique un modèle de conception objet à partir d'un ensemble de besoins standards prédéfinis sous forme de scénarios « besoins patterns ».

L'environnement permet de prendre en compte les différents profils utilisateurs, de vérifier la prise en compte correcte des besoins utilisateurs lors de la modélisation conceptuelle UML, de gérer des versions, d'assurer la traçabilité de la modélisation, de valider le modèle de conception vis-à-vis des besoins utilisateurs, de prendre en compte les attentes utilisateurs en terme d'IHM et non pas seulement leurs besoins fonctionnels et d'intégrer des librairies de « besoins standards ».

La thèse présente aussi l'approche et la démarche utilisées dans la réalisation de l'EMI. Elle explique les principes de représentation des connaissances scénario métier, la construction de la base de cas et l'exploitation des outils de RàPC dans le processus de développement des logiciels. Ainsi, l'approche est justifiée et validée par une application réelle présentée dans le chapitre 4 « Mise en œuvre de l'EMI ».

Approche

L'EMI est basé sur une nouvelle méthodologie de développement. Celle-ci constitue notre approche de l'ingénierie intelligente des logiciels. Cette approche renforce le rôle des experts métier pendant le processus de standardisation des besoins. Ce rôle consiste à transformer les connaissances métier informelles (micro-besoins ou fragments de scénario métier) en connaissances semi formelles (scénarios prototypes) et à les regrouper dans une base de connaissances (base de cas) intégrée dans l'EMI.

L'EMI fournit pour les experts métier un langage semi formel pour décrire les connaissances métier sous forme de scénarios prototypes appelés « *scénarios patterns* ».

Ces scénarios patterns sont complétés par des schémas de sous modèles objets standards appelés patterns de conception qui, ensemble, forment la base de cas d'un système de RàPC. Une entrée de la base de cas est le couple scénario pattern (problème) et schéma sous modèle objet (solution). Le problème et la solution génèrent un cas de la base de cas.

L'ingénieur système, à partir des nouvelles exigences du client et de la base de cas, construit d'une façon automatique un modèle de conception objet (modèle d'analyse) conforme à ces nouvelles exigences.

Après la construction de la base de cas, la réalisation du modèle de conception objet, à partir de l'EMI, suit ces étapes (figure 1):

1- L'ingénieur système et le client construisent un pré modèle des besoins comme un nouveau cas pour le système RàPC.

2- L'ingénieur système génère, à l'aide d'un outil de RàPC, un modèle d'analyse (ex. diagramme de classes) équivalent à ce nouveau cas. Cette génération procède par un test de similarité entre le pré modèle des besoins du client et la base de cas et fournit un nouveau modèle des besoins riche en scénarios patterns. Ainsi, les sous modèles objet (patterns de conception) relatifs aux scénarios patterns de la base de cas seront rassemblés dans un modèle objet unique (modèle d'analyse).

3- L'ingénieur système peut adapter et valider le modèle d'analyse précédent pour en fournir le modèle d'analyse final.

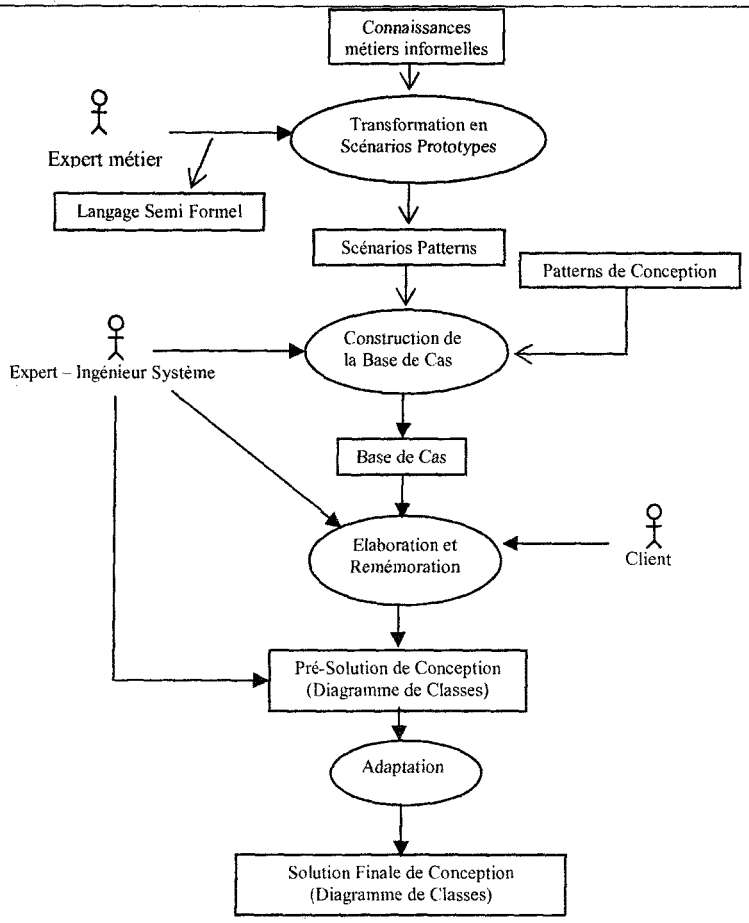


Figure 1: Elaboration d'une solution de conception

Organisation du mémoire de thèse

La thèse est composée d'une introduction générale, de cinq chapitres et d'une conclusion générale.

Le chapitre un est consacré à l'ingénierie des besoins (IB), les récents travaux dans le domaine et les différentes méthodes de recueil des besoins. A la fin du chapitre nous avons présenté les problèmes liés à l'ingénierie des besoins, l'impact de ces problèmes sur l'échec ou la réussite d'une spécification et les limites de l'IB.

Le chapitre deux exposera l'état de l'art de la spécification des besoins, les méthodes et les modèles utilisés dans ce domaine. Dans ce chapitre nous avons présenté aussi l'aspect logiciel des méthodes de spécification sur les systèmes informatiques et les inconvénients des méthodes de spécification en fonction de l'évolution et le changement de l'organisation. Comprendre les impacts d'un changement organisationnel sur les systèmes informatiques existants reste un problème très ouvert. La prise en compte de l'évolution des besoins métier durant la phase de développement d'un système reste elle aussi problématique. Dans ce but, et à la fin du chapitre nous avons présenté la nécessité

de faire recours aux expériences passées pour les capitaliser afin de les réutiliser. Les connaissances métier réutilisables seront le noyau de notre EMI futur environnement de développement d'applications informatiques.

Le chapitre trois est consacré pour détailler notre nouvelle approche méthodologique. Ce chapitre a débuté par une présentation générale des systèmes de Raisonnement à Partir de Cas (RàPC). Ensuite, nous avons montré l'intégration du RàPC dans l'EMI, sa structure et fonctionnalité. La standardisation des connaissances métier à l'aide des scénarios prototypes est la méthode adoptée par notre approche. Ces scénarios prototypes sont dits scénarios patterns et ils constituent les cas problèmes de notre RàPC dans l'EMI. Le langage LSP fournit un moyen semi-formel pour la rédaction des scénarios patterns. Il possède des analyseurs lexical, syntaxique et sémantique permettant de rédiger des scénarios patterns opérationnels.

Les cas problèmes de la base de cas appelés, cas sources, sont complétés par des solutions sous forme de patterns de conception. Le couple scénario pattern et pattern de conception est un cas mémorisé dans le RàPC de l'EMI. Le RàPC implante des algorithmes de description des cas cibles et de remémoration. Ces algorithmes sont aussi décrits dans ce chapitre et il termine avec un exemple qui valide l'utilisation de notre méthodologie.

Le chapitre quatre exposera une description détaillée de l'EMI, son architecture, sa plate-forme, ses interfaces et ses outils. Les interactions homme-machine à travers les interfaces de l'EMI sont expliquées à l'aide des diagrammes de séquences. « Project Developer 1.0 » est le nom du logiciel donné à l'EMI, il est décrit à la fin du chapitre avec ses outils de génération et d'édition.

Le chapitre cinq s'intéresse à l'évaluation expérimentale, à l'intégration et à l'extension de l'EMI. L'évaluation expérimentale est effectuée sur une application de « Gestion bancaire », les résultats fournis par l'EMI sont tous détaillés et commentés. L'application montre l'intégration de notre méthodologie dans l'EMI depuis la construction de la base de cas jusqu'à la génération du modèle d'analyse (exemple diagramme de classes). La deuxième partie de ce chapitre est consacrée pour l'intégration et l'extension de l'EMI. Les perspectives d'intégration concernent les améliorations techniques à apporter à l'EMI de façon en accroître les performances. Et enfin, l'extension de la méthodologie à d'autres domaines d'applications.

Ingénierie des besoins (IB) : Synthèse et Limites
Chapitre 1

Dans ce chapitre nous exposons l'état de l'art concernant l'ingénierie des besoins (IB) et les récents travaux et publications dans le domaine. L'état de l'art est suivi par une synthèse critique décrivant les problèmes liés à l'ingénierie des besoins et ses limites. A la fin de ce chapitre, et dans sa conclusion nous montrons que l'absence de définition précise d'une démarche d'aide à la construction de modèles de besoins nous a poussé à introduire une nouvelle façon pour construire les modèles des besoins.

1.1 Introduction

La naissance de l'ingénierie des besoins résulte de la nécessité de réussir les projets informatiques qui restent, malgré l'évolution des méthodes de développement et de conception, face à un vrai défi. Les statistiques ont montré que le pourcentage de succès des projets informatiques est très faible ; en effet, une étude réalisée par Standish Group en 1994 [Standish 94] sur 8000 projets menés par 350 compagnies, a montré que seulement 16% des projets ont été considérés réussis.
En 1999 les résultats, d'une autre étude faite sur le « Department of Defense Software (DoD) » [Jarzombek et Stanley 99], a montré que seulement 2% des projets informatiques réalisés par (DoD) ont été utilisés et que 75% de ces projets sont abandonnés ou jamais utilisés. Les 23% restant des projets ont été utilisés après modification.

Leishman et Cook [Leishman et Cook 02] renvoient ceci à la présence d'une inadéquation entre les services fournis par le système après sa réalisation et les besoins utilisateurs qui, normalement, ont été fixés au début du processus de développement. Cette inadéquation est due à l'inexistence d'un langage commun entre le client d'une part et l'ingénieur système d'autre part.

Dans le cycle de vie d'un projet informatique on peut distinguer trois grandes périodes: conception, réalisation et maintenance. La période de la maintenance consiste à prendre en compte les évolutions apparaissant après le lancement opérationnel et les

modifications de l'application initiale pour l'adapter à des changements imprévus dans les besoins [Nancy et Espinasse 96].

Malheureusement, la réalité suggère qu'un pourcentage excessif de ressources de développement de logiciel est dépensé pour la préservation de programmes (maintenance) [Booch 94]. Les méthodes actuelles de conception des systèmes d'information ne permettent pas de concevoir des systèmes d'information artificiels dotés de capacités d'auto-adaptation et d'auto-évolution et qui leur permettent d'accompagner l'évolution de l'entreprise et de son système d'information naturel [Nancy et Espinasse 96].
Pour diminuer le coût de la maintenance les recherches ont été focalisées d'une part sur l'introduction de nouvelles approches dans le cycle de vie et d'autre part sur l'amélioration et l'exploitation des techniques et méthodes d'ingénierie de besoins.

" L'ingénierie des besoins est l'activité qui transforme une idée floue en une spécification précise de besoins, souhaits, exigences exprimés par une communauté d'usagers et donc définit la relation existante entre un système et son environnement " [Origine du terme IEEE 85]. L'ingénierie des besoins est l'une des méthodes permettant de minimiser le risque d'inadéquation du produit ou service aux attentes des clients.

Les importants travaux de recherches sur l'ingénierie de besoins sont réalisés par le groupe RENOIR (Groupe International de travail sur l'ingénierie de besoins). Selon RENOIR, l'ingénierie de besoins s'attache à porter des améliorations au niveau du cycle de vie du développement du système. Elle fournit les outils, les concepts et les méthodes qui mettent en relation les fournisseurs de services et de produits de technologies de l'information [Costanzo 00].

Ces travaux de l'ingénierie de besoins proposent plusieurs scénarios de processus pour identifier et recueillir les besoins. Ces processus sont guidés par plusieurs méthodes d'ingénierie de besoins. Ils ont été critiqués par un travail effectué en 1992 intitulé « Issues in requirements elicitation » [Christel et Kang 92]. Ce travail focalise sur la problématique de l'identification des besoins et propose ensuite un cadre méthodologique conceptuel combinant les avantages des différentes méthodes d'ingénierie. Ce cadre méthodologique propose un processus de recueil de besoins en cinq étapes qui constitue un outil que nous pourrons réemployer pour critiquer les autres méthodes :

- Relevé des faits.
- Collecte des besoins et classifications.
- Évaluation et rationalisation.
- Hiérarchisation.
- Intégration et validation.

Plusieurs techniques et méthodes ont été développées selon ce cadre méthodologique, que nous présentons ci-après, et en particulier les méthodes PREVIEW, NATURE, MAP, CREWS, CREWS-l'Ecritoire.
Le processus de recueil des besoins selon les méthodes mentionnées n'est pas le même car chaque modèle de processus repose sur un concept différent de l'autre.

1.2 Techniques Avancées de l'IB

1.2.1 Méthode PREVIEW

La méthode PREVIEW est une méthode d'exploration des besoins par exploitation des points de vue. Elle a été développée par l'équipe de chercheurs en Informatique du Computing Department of Lancaster University, UK. Ceux-ci l'ont transcrite sous forme de guide à l'attention des développeurs de projets PREVIEW1 [Sommerville et Sawyer 97].
La méthode développée s'appuie sur un modèle flexible de points de vue qui peut s'adapter à une large gamme de contextes industriels.
Un point de vue, selon PREVIEW, se compose des éléments suivants :

- **Le nom**, qui l'identifie;
- **Le domaine** du point de vue, est détaillé en trois types : perspective de l'acteur impliqué, perspective de personnes indirectement concernées, champ de description du domaine;
- **Le problème** sur lequel s'exprime le point de vue;
- **Les sources**, qui identifient les sources des besoins associés au point de vue;
- **Les besoins** du point de vue;
- **L'historique** du point de vue.

PREVIEW définit un processus qui débute avec l'identification et l'élaboration des problèmes qui sont alors utilisés pour mener les activités d'ingénierie des besoins. Il faut souligner que PREVIEW n'est pas une méthode d'ingénierie de besoins fournissant des lignes de conduite et des directives, dans la mesure où les utilisateurs possèdent déjà souvent leurs propres manières de travailler.
Le processus générique de PREVIEW est formé de l'itération d'un processus de mode spirale (figure 1.1) : les différentes activités conduisant à l'identification de besoins se déroulent successivement selon le bras de la spirale, tandis que les informations émergeant à la fin du processus influencent le processus à l'itération suivante.
Les trois activités conduites à chaque cycle sont :

-Identification des besoins : Etant donné les attentes globales du domaine, différentes sources sont consultées afin de comprendre le problème et le domaine d'application. Les besoins sont alors exprimés sous une forme générale et vague.
-Analyse des besoins : elle permet l'identification des besoins manquants.
-Négociation des besoins : les analystes et les décideurs considèrent la problématique afin d'atteindre un consensus.

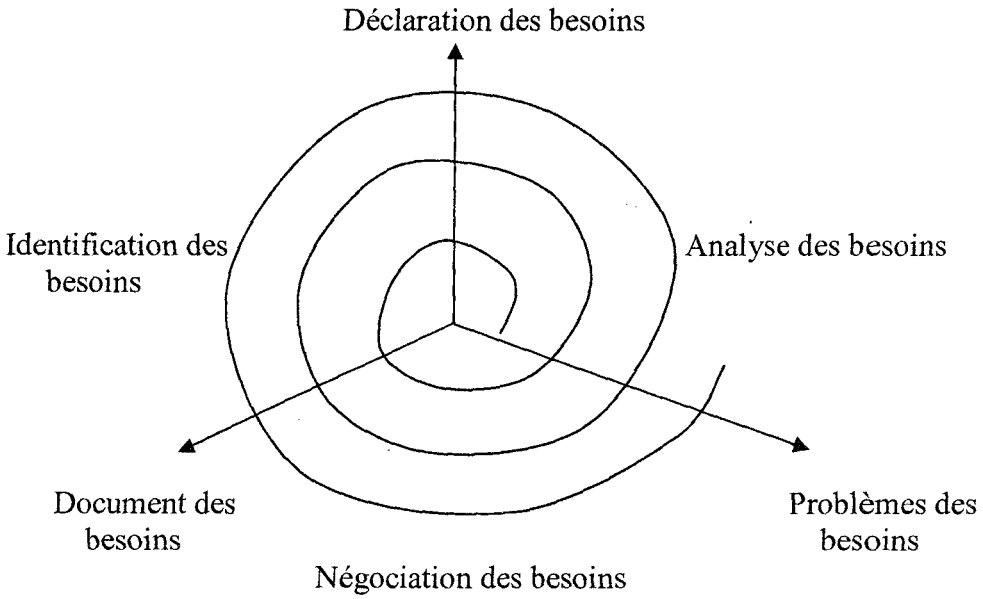


Figure 1.1 : Processus PREVIEW en spirale

Un point-clé du processus est l'identification des points de vue qui correspondent à l'application recherchée. Le processus d'identification des points de vue s'appuie sur le modèle opérationnel (points de vue utilisateurs) et le modèle organisationnel (points de vue direction). Enfin, il faut prendre garde à ce que le nombre de points de vue et de problèmes identifiés reste suffisamment bas pour pouvoir gérer le processus. En pratique, il convient de se limiter à 5 problèmes au maximum, et à 10 points de vue. Si ces nombres sont dépassés, il convient alors d'effectuer des regroupements en termes plus génériques.

Deux premières remarques sont à formuler concernant cette méthode :
- D'une part, elle a pour objectif de s'intéresser aux systèmes en sûreté de fonctionnement, donc souvent complexes et gouvernés par de fortes contraintes temporelles.
- D'autre part, elle a pour ambition de pouvoir être élargie à tous les domaines de l'ingénierie de besoins, donc en particulier au nôtre.
Le souci exprimé de "transformer des objectifs vagues en propriétés des systèmes à concevoir" semble s'appliquer à la problématique des systèmes complexes à objectifs généraux comme le nôtre. Ceci se trouve confirmé par l'approche qualifiée de «

pyramidale », qui reconstitue en quelque sorte la hiérarchie des bénéficiaires comme celle des décideurs.
La notion de besoin, telle qu'elle est définie dans la méthodologie, contient également des éléments s'adressant à notre problématique.
Le processus en lui-même, qui comprend deux phases intitulées « Analyse des interactions entre points de vue » et « Résolution des incohérences », pourrait s'appliquer au cas des systèmes à forte composante informelle. Partant d'un élément quelconque du processus informel (le point de vue d'un acteur par exemple), on peut alors identifier des besoins, identifier de nouveaux points de vue et les exposer à d'autres acteurs du processus qui en souligneront les incohérences, de manière à préciser les besoins.
PREVIEW n'admet pas une vraie stratégie de recueil des besoins lorsque le nombre de points de vue est très important.

1.2.2 Méthode NATURE

La méthode NATURE utilise un processus méta-modèle orienté décision qui permet une modélisation du processus sous forme de quadruplets <Situation, Décision, Argument, Action>. Le méta-modèle comporte un ensemble des concepts formellement définis [Rolland et *al.* 99a] [Rolland et Schwer 99]. Il s'agit d'un méta-modèle à caractère *prescriptif* plus particulièrement adapté à des activités de développement à caractère créatif telles que celles de l'ingénierie des besoins. Le paradigme proposé est contextuel. Il part de la constatation suivante : à tout moment, l'ingénieur d'ingénierie des besoins se trouve dans une *situation* qu'il perçoit subjectivement, avec une certaine *intention* à l'esprit. Sa réaction dépend à la fois de la situation dans laquelle il se trouve et de l'intention qu'il a; en d'autres termes, elle dépend du *contexte* dans lequel il se trouve. On peut donc dire que l'ingénieur réagit contextuellement, souvent par analogie avec les situations dans lesquelles il s'est déjà trouvé impliqué par le passé.

La méthode couple la modélisation du contexte de la décision à la décision elle-même. Le concept central est donc le *contexte* qui est formé de l'association d'une *intention (but)* et de la *situation* dans laquelle l'intention peut être atteinte et la décision prise. Le méta-modèle identifie trois types de contextes : plan, choix et exécutable. Un contexte *choix* offre des alternatives pour prendre une décision ; un contexte *exécutable* identifie l'action qui implémente la décision; un contexte *plan* fournit un plan de prise de décision [SiSaid et *al.* 96]. On trouvera dans la thèse de Véronique Plihon [Plihon96], une panoplie d'une vingtaine d'instances du méta-modèle pour décrire des méthodes telles que OMT, E/R, MERISE, REMORA, O* etc. On y montre en quoi le méta-modèle aide à apporter de la flexibilité à des démarches définies par leurs auteurs de façon rigide et de quelle manière il peut capturer de la connaissance méthodologique heuristique.

La méthode NATURE, validée par un logiciel appelé MENTOR [Si-Said et *al.* 99], [Rolland et *al.* 99b], [Grosz 94], [Grosz et *al.*96] est à la fois un outil CASE et un outil CAME. Il aide à la construction de méthodes (et en particulier de leurs modèles de processus) mais aussi à l'usage de ces méthodes dans le développement d'applications. MENTOR vise à satisfaire les trois objectifs suivants :

- aider à la construction des modèles de processus
- guider la conduite de processus
- permettre l'évolution des modèles de processus à partir des expériences.

L'architecture présentée à la figure 1.2 schématise les fonctionnalités de MENTOR qui comporte trois composants: *un environnement pour l'ingénieur d'applications*, dans lequel le processus est guidé, exécuté et tracé, *un environnement pour l'ingénieur des méthodes* dans lequel le processus est défini et enrichi, et un *dictionnaire* qui lie les deux environnements précédents.

Le dictionnaire est organisé en trois niveaux : celui des traces de processus (l'enregistrement de ce qui s'est passé réellement pendant le développement), celui des modèles de processus, c'est à dire des démarches et celui du méta-modèle des processus. Chaque niveau contient de l'information sur le *processus* et sur le *produit* qu'il conduit à construire.

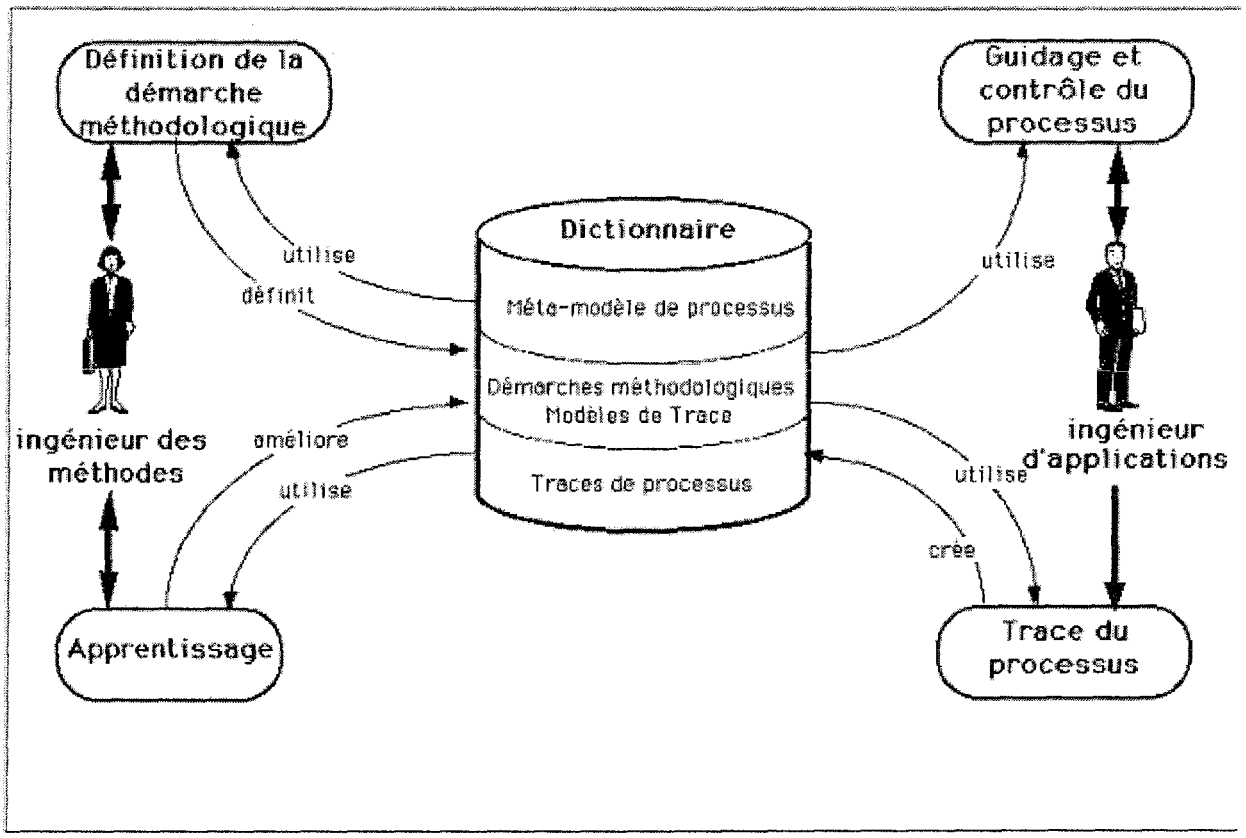


Figure 1.2 : Architecture générale de l'environnement MENTOR pour construire, tracer, guider et enrichir les processus [Si-Said et *al.* 99]

L'environnement de l'ingénieur d'applications guide et trace le processus. Dès qu'une démarche a été définie, le processus peut être guidé par le *moteur de guidage.* Ce moteur implémente l'approche contextuelle. Il aide l'ingénieur d'application à prendre la bonne décision dans une situation donnée. Une des caractéristiques essentielles du moteur de guidage est sa *généricité.* Il est indépendant des modèles de processus et, bien sûr, des processus. Il ne dépend que du méta-modèle de processus, de ses concepts et des relations qui les lient. Le moteur de guidage est similaire au moteur d'inférence d'un système expert qui réagit uniquement par rapport à la structure des règles. De la même façon, le fonctionnement du moteur de guidage ne dépend que des trois classes de contextes (les contextes exécutables, plans ou choix) du méta-modèle de processus et de leurs relations. Le moteur de guidage est un *processeur de contextes.* Il est capable de guider tous les processus modélisés comme un ensemble de contextes.

Le processus est tracé selon la structure définie dans le modèle de trace. Le module de trace est couplé au moteur de guidage. Ceci est nécessaire car le processus est à la fois guidé par les démarches et tracé selon la structure suggérée dans le modèle de trace. L'étude des divergences entre les traces réelles et les traces prévues sert de base au processus d'amélioration de la démarche [Prat 99]. Toutefois le module de trace peut également fonctionner indépendamment du guidage. Les traces obtenues au cours d'une session guidée pourront être utilisées pour guider d'autres projets dans le futur [Jarke 94].
L'environnement de l'ingénieur des méthodes guide la construction d'une démarche particulière en s'appuyant sur la *méta-démarche* (le modèle du méta-processus) et une bibliothèque *des motifs génériques.* Ceci évite la construction des démarches trop dépendantes du produit et négligeant les autres activités cognitives, sociales et consensuelles de l'analyse. Cela permet aussi d'accélérer et de faciliter la construction des démarches situationnelles, c'est à dire adaptées à des caractéristiques particulières d'un projet donné [Rolland et Prakash 96].

L'environnement guide l'ingénieur méthode dans son travail de construction de démarche de manière tout à fait similaire au guidage apporté à l'ingénieur d'applications. En effet comme la méta-démarche est formulée dans les termes du méta-modèle, elle est un modèle de processus comme les autres, et le moteur de guidage peut guider le méta-processus de la même manière qu'il guide les processus d'applications.

A la fin du projet, la démarche est enrichie en utilisant les résultats de l'étude des traces du processus. Les ingénieurs d'applications peuvent bénéficier de l'expérience passée. Une fois de plus, comme le *processus d'enrichissement* d'une démarche est modélisé comme un processus, le moteur de guidage peut guider l'ingénieur de méthodes dans sa tâche de modification du modèle de processus.

Le premier point faible de la méthode NATURE est le problème d'identification de situation par l'ingénieur d'application. Selon NATURE l'identification d'une situation parait un travail simple et par l'utilisation d'un processus de guidage de prise de décision, la décision devient elle-même simple. Alors que l'ingénieur d'application ne peut pas être jamais dans une situation simple pendant le développement puisque ce dernier est un processus complexe de point de vue des tâches et des activités qui les apporte. L'ingénieur d'application se trouve toujours dans des situations décomposables plutôt que des situations élémentaires. Enfin, l'ingénieur d'application doit se trouver dans un cas de raffinement successif de situation afin d'identifier exactement une situation élémentaire. La méthode NATURE ne supporte pas un modèle de processus pour la décomposition des situations ; La méthode compte beaucoup sur l'expérience des ingénieurs d'application et surtout lorsque le problème à traiter est très récurrent.

Le deuxième point faible de la méthode NATURE est le manque d'un processus de validation de décision. La décision prise par l'ingénieur d'application après l'identification d'une situation n'est pas validée dans le vrai contexte de l'ingénieur d'application. Il parait que la méthode NATURE utilise des situations élémentaires (enregistrées dans un dictionnaire) fréquemment utilisées par les ingénieurs d'application.

1.2.3 Méthode MAP

MAP, extension du projet NATURE, propose un modèle de conduite de plusieurs modèles d'ingénierie; l'une des caractéristiques principales des approches d'Ingénierie des Besoins (IB) définies dans le contexte de l'Ingénierie des Systèmes d'Information (ISI) est l'exploitation des connaissances du métier pour définir les exigences du système. La *vue métier* (évoquée traditionnellement en France sous la dénomination « maîtrise d'œuvre »), et la *vue système* (celle qui relève de la « maîtrise d'ouvrage ») ne s'opposent pas. Au contraire, elles doivent coopérer pour que les SI correspondent aux besoins de ses futurs utilisateurs.

Cependant, les modèles de définition de métier (ou « modèles métier ») et les modèles de spécification de systèmes sont habituellement exprimés dans des langages différents. D'un côté, les modèles métier reposent sur des concepts tels que but, processus, acteur et rôle. D'un autre côté les modèles de spécification de système utilisent les concepts d'opération, de classe, d'événement, et autres. Il y a donc un écart conceptuel entre les deux types de modèles rendant difficile la bonne coordination entre la maîtrise d'œuvre et celle d'ouvrage. [Arsajani et Alpigini 01] identifient cet écart sous le nom de « discordance conceptuelle » (« conceptual mismatch ») et montrent les dangers qu'il représente. Par exemple, ce problème se rencontre fréquemment dans le monde industriel dans le contexte de projets ERP où les experts du système et les experts

métiers emploient chacun leur documentation, conservent leur propre langage et, ce faisant, rencontrent des difficultés à se faire comprendre de l'autre partie. Le risque principal de cette incompréhension mutuelle est de produire un paramétrage de système qui ne corresponde pas aux besoins réels des futurs utilisateurs, et comme cela s'est vu à de nombreuses reprises, d'aboutir à un échec total du projet.
L'objectif de la méthode MAP est de définir un langage d'expression des besoins qui soit exploitable à la fois dans la vue métier et dans la vue système. Ce langage doit non seulement pouvoir être exploité par les deux parties, mais aussi dans divers types de projets.

La méthode MAP vise aussi à permettre la représentation dans un même modèle, de plusieurs façons de conduire un processus d'ingénierie. Il s'agit donc d'une représentation multi-modèle permettant la construction du modèle suivi pendant le déroulement d'un projet de manière dynamique à partir des multi modèles contenus dans la représentation. Le système MAP échappe à la linéarité des modèles de processus classiques, va plus loin que la possibilité de sélectionner une alternative parmi plusieurs pour exécuter une tâche telle que l'offre NATURE afin de permettre une construction dynamique du modèle de processus suivi.

Le paradigme sous-jacent à l'approche MAP est celui de carte : une carte contient une collection d'intentions (*buts*) reliées entre elles par des stratégies de réalisation de ces intentions lors du déroulement du processus. Une stratégie peut être une action exécutable qui aboutit à des transformations du produit ou être elle-même un processus modélisé par une sous-carte. Les cartes de processus ne contraignent pas l'utilisateur dans une démarche constituée d'étapes successives mais permettent au contraire, un grand degré de liberté dans l'ordonnancement de réalisation des intentions et dans le choix des techniques qu'il désire appliquer.

La méthode MAP, comme un projet récent, reste dans la phase de répondre et de résoudre le problème de l'évolution qui est le plus fréquemment rencontré dans la pratique, et le moins souvent traité par les approches d'Ingénierie des Besoins. Dans ce contexte différents problèmes doivent être résolus :
- cohérence de l'évolution : il s'agit de la difficulté à définir de manière systématique ce qui doit changer dans l'une des vues lorsque des besoins d'évolution sont exprimés dans l'autre vue.
- Structuration systématique de la documentation : dans le contexte de l'évolution, de même que dans le contexte du développement initial d'un système, tous les besoins sont habituellement spécifiés au même niveau. Les documents de spécification des exigences prennent alors la forme d'une liste de besoins faiblement structurée (ou structurée de manière non systématique), sans considération au regard du niveau d'abstraction et de détail, et au risque de rencontrer une explosion combinatoire du nombre de besoins à présenter au lecteur. Ce problème pose d'autant plus de difficultés dans le cadre de

l'évolution que l'on ne gère pas que les besoins d'un système, mais aussi divers besoins relatifs à divers systèmes dans divers horizons temporels.

- Organisation flexible des besoins : il est fréquent de rencontrer des bases de données de besoins dans lesquelles la collection des besoins est soit faiblement organisée à travers des liens de dépendance, soit fortement structurée à travers des liens sémantiquement riches tels que ceux offerts par le schéma de composition/décomposition. Dans le cadre de l'évolution, les liens entre exigences doivent à la fois comporter suffisamment de sémantique pour pouvoir définir des collections cohérentes de besoins, et en même temps être suffisamment flexibles pour permettre à des exigences d'évoluer sans que la structure de l'ensemble ne soit nécessairement remise en question.

- Langage spécifique pour l'expression des besoins d'évolution : les langages d'expression des besoins offrent chacun un moyen de définir soit ce qui est nécessaire dans la future organisation du métier, soit ce qui est nécessaire dans le futur système, sans lien explicite vers la situation présente. Or, dans le contexte de l'évolution, il ne s'agit pas simplement de définir ce que sera le futur système ou la future organisation du métier, mais aussi de caractériser ce en quoi on veut qu'elle diffère ou reproduise la situation présente. Nous pensons que des méta-modèles doivent être proposés pour définir spécifiquement ces deux aspects de l'évolution.

- Systématisation et guidage des démarches : le contexte de l'évolution nécessite des démarches des techniques et des outils qui soient adaptées aux problèmes spécifiques qu'il pose. Ces démarches techniques et ces outils doivent être identifiés, définis, évalués, développés et documentés avec le souci principal de répondre à ces spécificités que pose l'évolution.

1.2.4 Méthode CREWS

Le but du projet CREWS est de développer, d'évaluer, et de démontrer l'applicabilité de méthodes et d'outils pour l'identification et la validation des besoins basés sur des scénarios coopératifs [Tawbi et Souveyet 99].
Les spécifications des besoins décrivent un futur monde réel, habituellement dans une combinaison de représentations formelles, semi-formelles et informelles.
L'implication des utilisateurs et des clients pendant le développement d'une spécification des besoins est un but généralement accepté. CREWS perçoit le processus d'ingénierie des besoins comme un processus coopérant d'apprentissage dans lequel les responsables et les ingénieurs des besoins doivent se comprendre mutuellement pendant l'identification et la compréhension des besoins, la détection des erreurs en validant la spécification des besoins, et réconcilier leurs connaissances à des différents niveaux techniques et sociaux.

Les approches basées sur les scénarios supposent que l'apprentissage doit être fondé sur des exemples de niveau cas d'utilisation ou scénarios. Un scénario est défini comme une séquence d'actions ou événements pour un cas spécifique de quelque tâche générique qu'un système est destiné à accomplir. Il consiste en un comportement et un contexte qui décrivent les agents, les objets, et cadre de l'environnement du système sous développement.

Dans le projet CREWS quatre méthodes et outils comme support pour l'identification et la validation des besoins basées sur des scénarios coopératifs ont été développés. Ces méthodes et outils sont dirigés par plusieurs responsables engagés pour découvrir et négocier les besoins pour des vraies scènes du monde réel sous différentes perspectives. La compréhension du langage naturel est utilisée pour permettre aux responsables de découvrir les besoins à partir des scénarios décrits par des déclarations simples et courtes en langage naturel. La validation est facilitée par une animation des besoins coopératifs, et par comparaison systématique de la spécification avec des utilisations des scénarios testés et générés interactivement à partir d'un domaine de connaissance.

1.2.5 Méthode CREWS-L'ECRITOIRE

CREWS-l'ECRITOIRE est un concept basé sur le couplage entre buts et scénarios. C'est une approche d'extraction des besoins basée sur l'exploitation de scénarios textuels combinée au développement de réseaux d'objectifs [Tawbi et *al.* 00].
L'approche d'extraction des besoins proposée vise à :
- fournir une collection complète et organisée de buts et de scénarios écrits en langage naturel et spécifiant une partie des besoins à l'égard du système ;
- aider à sa production par un processus guidé, exploitant les buts et les scénarios de manière alternée et complémentaire.

L'approche « *L'ECRITOIRE* » vise à guider la découverte / l'identification des besoins à travers un couplage bidirectionnel des buts et des scénarios supportant la transition des buts aux scénarios et vice-versa [Rolland et Ben Achour 97] [Rolland et *al.* 98]. Pour chaque but découvert un scénario peut être rédigé. Le couplage but-scénario est alors exploité dans la direction avant, des buts aux scénarios. Une fois un scénario rédigé, il peut être analysé pour en tirer de nouveaux buts. La découverte de buts correspond donc à l'exploitation de la relation dans la direction inverse.
Les différentes étapes guidées lors de l'application du processus offert par « L'*ECRITOIRE* » sont les suivantes :

- Identification initiale de buts
- Analyse de but
- Rédaction de scénario
- Analyse de scénario / découverte de nouveaux buts

Nous pouvons donc constater que la découverte de buts et la rédaction sont des étapes complémentaires et que les besoins sont identifiés incrémentalement en répétant le cycle analyse de but, rédaction de scénario, découverte de buts. Chacune de ces étapes est supportée par un ensemble de mécanismes guidant pas à pas l'application de la démarche. Le mécanisme de guidage d'analyse des buts est basé sur une analyse linguistique de la structure des buts. Celui-ci aide à reformuler une description narrative du but en une expression de but structurée et normalisée.

Le mécanisme de rédaction des scénarios combine des directives de style et de contenu à une *panoplie d'outils linguistiques*. Les premières conseillent le rédacteur sur le meilleur style à adopter lors de la rédaction des scénarios, et sur leur contenu en termes de concept. Les secondes fournissent une aide semi-automatique permettant de *vérifier*, de *corriger*, de *conceptualiser*, et de *compléter* les scénarios.

Finalement, l'analyse des scénarios est automatisée à travers une *panoplie de règles d'analyse* supportant trois stratégies issues de la découverte de buts. La première (stratégie d'*affinement*) permet d'approfondir l'exploration à travers les niveaux d'abstraction en détaillant certains aspects des scénarios analysés. La seconde (stratégie de *complétude*) permet de découvrir des buts complémentaires à ceux attachés aux scénarios analysés.
La dernière (stratégie d'*alternative*) permet de découvrir de buts alternatifs à ceux attachés aux scénarios analysés. Ces différents éléments méthodologiques sont implémentés dans un prototype [Tawbi et *al.* 98] [Tawbi et *al.* 00].

La méthode CREWS-L'ECRITOIRE semble avant tout pouvoir être adaptée au cas des processus complexes. En effet, lorsqu'un but est identifié, la méthode et le logiciel qui la supporte proposent une aide à l'élaboration du scénario correspondant, ce qui permet de construire progressivement une représentation du processus et de l'environnement complexes.

En outre, l'usage de scénarios, qui en simulant la réalité mettent en relief les conséquences, notamment néfastes, d'un mauvais fonctionnement du système, a un effet pédagogique. Les bénéficiaires du système prennent alors plus facilement la mesure de l'importance de celui-ci, ce qui relance la demande. Néanmoins, d'après la méthode, le choix des scénarios en incombe au développeur. Il serait intéressant d'envisager la validation et le choix des scénarios proposés par les acteurs.

1.3 Problèmes liés à l'IB

L'IB souffre néanmoins de plusieurs problèmes:

- *Multiplicité des bénéficiaires, acteurs ou décideurs*

Les méthodes les plus préconisées sont celles s'appuyant sur des questionnaires, en raison de la facilité à toucher un grand nombre de personnes. Cependant, les auteurs soulignent que cette forme de recueil des données prend difficilement en compte la variété des approches des différentes personnes. C'est pourquoi sont également indiquées celles faisant appel à des points de vue ou interviews personnalisées, qui permettent de prendre en compte la variabilité des opinions lorsque de nombreuses personnes sont impliquées ou touchées par le processus d'ingénierie de besoins.
Les techniques de groupe constituent également un moyen, si le groupe n'est pas trop important, de gérer la multiplicité des intervenants.

- *Variabilité spatiale*

Ce problème s'appréhende de deux manières. D'une part, en recueillant indirectement des informations sur le domaine par le biais de questionnaires ou interviews des acteurs du domaine. D'autre part, en construisant des modèles de domaine.

- *Complexité du processus*

Face à ce type de problème, un large recours est fait aux études de terrain d'une manière, études de cas, ou encore extrapolation de points de vue. Celles-ci s'appuient toujours sur un processus itératif permettant une construction progressive du domaine et surtout des besoins.
En outre, Christel et Kang [Christel et Kang 92] soulignent que « plus le processus est flou, plus l'implication des bénéficiaires et des acteurs du domaine est nécessaire ». Ces processus itératifs s'appuient donc sur les personnes concernées par le projet.

- *Volatilité de la demande*

C'est la problématique la moins bien adressée parmi les méthodes étudiées. Il s'agit en effet d'un problème très spécifique de notre contexte, les méthodes offrant une mise en perspective historique des problèmes et des besoins semblent les plus appropriées pour faire émerger l'expression du besoin. Il en est ainsi de celles se fondant sur des situations passées, ou scénarios, ainsi que sur des points de vue de manière à recueillir l'expérience des personnes impliquées.

1.4 Limites de l'IB

Aujourd'hui, il est clair que les hypothèses de l'IB ne sont plus valides. En réponse aux pressions économiques et à l'émergence constante de nouvelles technologies, les organisations changent plus rapidement que par le passé. En conséquence, ce que les utilisateurs attendent du système évolue bien plus rapidement qu'auparavant. Leurs besoins ne sont donc pas stables [Harker 93].

Comprendre les impacts d'un changement organisationnel sur les systèmes informatiques existants reste un problème très ouvert [Lubars 93]. La prise en compte de l'évolution des besoins durant la phase de développement d'un système reste elle aussi problématique [Curtis 88]. Les besoins évoluant, il n'est plus possible de les considérer comme données a priori. Il est plutôt nécessaire de déterminer les nouveaux besoins pour les systèmes existants et développer des modèles qui représentent fidèlement les besoins et leurs évolutions, durant tout le cycle de développement. De plus, le rôle central de l'analyste système doit être reconsidéré, l'ingénierie des besoins requiert la participation d'un grand nombre d'acteurs de l'organisation, chacun apportant sa vision sur ce que le système devrait faire [Finkelstein 90].

Enfin, la validation des besoins doit être basée sur les besoins réels de l'organisation plutôt que sur les fonctionnalités du système. C'est seulement à cette condition que les systèmes informatiques seront capables de s'adapter aux changements organisationnels. Ainsi, pouvoir aller dans ce sens il faut admettre une nouvelle vision de recueillir les besoins et de ne pas les considérer comme des besoins qu'à tout moment il faut les découvrir, mais agir de manière à capitaliser les besoins en tenant compte des systèmes similaires et des connaissances déjà acquises. C'est à ce moment là où nous pouvons profiter de la réutilisabilité des besoins dans les contextes similaires.

1.5 Conclusion

Aucune des méthodes examinées ne s'adresse de manière explicite à la problématique que nous avons définie dans l'introduction. Compte tenu du caractère plus général de la seconde problématique, la variabilité spatiale, il peut être plus profitable dans un premier temps de s'adresser à celle-ci en particulier. Il semblerait ainsi :
- de tester l'applicabilité de méthodes par scénarios au cas des organisations et processus complexes, ainsi que des méthodes couplant problèmes et points de vue ;
- le cas échéant, de proposer des adaptations de ces méthodes permettant de prendre cet aspect en compte.

En s'attaquant aux problèmes de l'ingénierie des besoins déjà cités nous proposons alors un cadre méthodologique pour l'ingénierie des besoins:
- L'ingénierie des besoins ne se limite plus à prendre en compte ce que doit faire le système mais cherche à comprendre le "pourquoi" du système. On répond à la question du "pourquoi" dans des termes d'objectifs organisationnels et de l'impact de ces objectifs sur le système de l'organisation.
- L'ingénierie des besoins ne s'intéresse pas directement aux fonctionnalités d'un système, ni à ce qui doit être réalisé par le système tel que cela a été conceptualisé par un analyste en partant d'un ensemble de besoins donnés. Au contraire, ce sont les

utilisateurs potentiels du futur système qui sont les plus à même de fournir des points de vue utiles et réalistes sur le système à développer. En conséquence, il faut effectuer une exploration détaillée des différentes manières dont le système sera utilisé ainsi que les activités qu'il doit supporter. Afin de répondre à tous ces objectifs, une méthode pour l'ingénierie des besoins, comprenant une démarche, un ensemble de modèles et des outils logiciels, doit être développée.

Spécification des besoins : Synthèse et aspects logiciels Chapitre 2

Dans ce chapitre nous présentons une synthèse concernant la spécification des besoins ainsi que le rapport entre les aspects métiers et les aspects logiciels. Ce chapitre présente la complexité menée par les différentes méthodes de développement de logiciels et le grand fossé entre l'ingénierie des besoins (IB) et la spécification. Dans la conclusion nous visons à montrer l'importance de changer le mode de raisonnement des problèmes de conception afin de faire la réutilisabilité des besoins une approche rigoureuse dans le développement des logiciels.

2.1 Introduction

L'analyse et la spécification des besoins présentent la première phase du cycle de vie d'un logiciel. C'est dans cette phase où nous devons spécifier et définir le système à construire. Les modèles développés doivent décrire ce que le système doit faire répondant aux besoins exprimés. Il est important pendant cette phase de rester en contact avec les clients pour que le système cible soit conforme à leurs besoins. Au cours de cette phase nous cherchons à construire des modèles qui nous permettent à mieux comprendre le système.

Ainsi, les modèles construits sont totalement détachés des problèmes d'implémentation (langage, SGBD, matériel, etc.). C'est la définition exacte de la spécification, il faut modéliser sans contraintes d'implémentation. La cible est d'obtenir une spécification qui résolve notre problème dans les conditions idéales. Ainsi la spécification est uniquement orientée « résolution de problèmes », sans se compliquer l'existence avec des problèmes techniques. Le fait de ne pas se soucier de l'implémentation nous permet d'assurer que notre analyse est basée réellement sur le problème et non pas sur des considérations spécifiques aux conditions de développement.

La spécification des besoins fournit deux modèles le modèle des besoins et le modèle d'analyse. Le modèle des besoins doit définir les limites du système et ses

fonctionnalités. On développe un modèle des besoins en utilisant les termes et les notations de l'environnement réel où s'intègre le système. C'est là que les interfaces du système sont décrites. C'est là aussi que l'on décrit l'ensemble des cas d'utilisation (services) réalisés par l'ensemble des acteurs. Les acteurs sont l'environnement du système, les cas d'utilisation ce qui se passe à l'intérieur.

Le modèle d'analyse doit fournir une approche conceptuelle du problème. Le but de ce modèle est de définir une structure robuste et extensible qui nous servira de base pour la construction du système. Un modèle d'analyse comporte les objets réels du système et leurs interactions sous les différentes contraintes fonctionnelles.
Donc, le but général d'une spécification est de définir des modèles descriptifs du vrai monde réel et compréhensible par les acteurs du système et le client.

2.2 De la spécification orientée donnée vers la spécification orientée modèle

Pour développer des systèmes logiciels de qualité, il faut utiliser une méthode de développement. Avec la méthode de développement on construit une spécification des besoins. La spécification est la phase la plus importante et délicate du cycle de vie de logiciel. Plusieurs méthodes d'analyse et de conception ont été utilisées pour construire des spécifications pour les systèmes à modéliser. En réalité, la plupart des méthodes peuvent être classées dans l'une des catégories suivantes [Sommerville 04]:

- conception orientée par les données,
- conception structurée descendante,
- conception systémique,
- conception orientée objets,
- conception orientée composants,
- conception orientée modèles.

Jacobson a classé les méthodes d'analyse et de conception en deux catégories : les méthodes fonctions/données et les méthodes orientés objet [Jacobson 93].
Une étude détaillée des méthodes d'analyse et de conception est décrite en annexe C. Chacune de ces méthodes repose sur un concept et une démarche particuliers. Les systèmes développés à l'aide d'une méthode fonctions/données deviennent souvent difficiles à maintenir [Jacobson 93]. Un problème de fond des méthodes de la catégorie fonctions/données vient du fait qu'en principe, toutes les fonctions doivent savoir comment les données sont stockées, c'est-à-dire leur structure de données. L'autre problème des méthodes fonctions/données, c'est que l'on ne pense pas naturellement de la même manière que l'on structure. La spécification des besoins est formulée d'habitude en langage naturel. Elle décrit souvent le « quoi » : ce que le système doit effectuer, et de quelles fonctionnalités il doit disposer, et quels éléments il doit comporter. Puis on reformule souvent tout cela en terme de « comment » au moment de

la division fonctionnelle, où l'objectif est différent. De cette manière, on obtient une différence sémantique importante entre les vues internes et externes du système.

Les limites des méthodes orientées objets sont plusieurs et nécessitent une maintenance permanente pour les dépasser. La première limite est la complexité des systèmes dus à la présence de beaucoup d'interaction entre les objets du système. La deuxième limite concerne l'évolution des objets dans leur contexte et le changement de leur comportement qui encerclent les méthodes orientées objet. La conception des objets avant l'implémentation met les objets dans une forme statique et la modification de leur structure après l'implémentation devient un processus très cher et coûteux.

2.3 Aspects métiers et aspects logiciels

L'évolution des méthodes de spécification (voir annexe C) vise fondamentalement l'aspect logiciel des systèmes informatiques. Elles n'ont pas pris en considération l'évolution et le changement de l'organisation. Aujourd'hui les organisations changent plus rapidement que par le passé. En conséquence, ce que les utilisateurs attendent du système évolue bien plus rapidement qu'auparavant. Leurs besoins ne sont donc pas stables [Harker 93].

Comprendre les impacts d'un changement organisationnel sur les systèmes informatiques existants reste un problème ouvert [Lubars 93]. La prise en compte de l'évolution des besoins métier durant la phase de développement d'un système reste elle aussi problématique [Curtis 88]. Les besoins métier évoluant, il n'est plus possible de les considérer comme données à priori. Il est plutôt nécessaire de déterminer les nouveaux besoins métier pour les systèmes existants et de développer des modèles représentants fidèlement les besoins métier et leurs évolutions, durant tout le cycle de développement. Les besoins métiers font aujourd'hui un domaine de recherche très important. Selon [Dieng et *al.* 01] les besoins métiers ou connaissances métiers sont développés dans des « mémoires d'entreprise », une mémoire d'entreprise est la représentation persistante, explicite, désincarnée, des connaissances et des informations dans une organisation, afin de faciliter leur accès, leur partage et leur réutilisation par les membres adéquats de l'organisation.

Une autre recherche a spécifié les besoins métiers par des composants métier réutilisables. Un composant métier fournit plusieurs structures réutilisables pour un même objet de domaine ainsi qu'une aide à leur réutilisation [Ramadour et Cauvet 02]. Frey, Gomes et Sagot ont appliqué la méthode QFD pour la capitalisation des connaissances métier afin de les rendre facilement réutilisables par des novices [Frey, Gomes et Sagot 07]. D'autres méthodes offrant la possibilité de capitaliser les connaissances ont été élaborées telles que les mémoires projet [Djaiz et Matta 06], le modèle KnoVA-Sigma [Serrafero 02] ou la méthode GAMETH [Grundstein et *al.* 03].

Aujourd'hui la réutilisabilité des connaissances métier dans le monde informatique apparaît dans la conception des systèmes d'information de l'entreprise. Par exemple, l'ingénieur système peut réutiliser des schémas de conception (« design patterns ») dans son modèle d'analyse et il peut alors profiter des objets réutilisables. Mais, les connaissances métier ne sont pas réutilisables au niveau du modèle des besoins car l'ingénieur système définit son modèle des besoins en fonction du système informatique c'est-à-dire en fonction de moyens matériels et logiciels du système. Alors, les connaissances métier sont capitalisées en vue d'être exploitées par les acteurs de l'entreprise plutôt que par l'ingénieur système.

L'ingénieur système définit le modèle des besoins en décrivant tous les cas d'utilisation (ou les fonctions) que le système doit réaliser. Il doit aussi détailler les cas d'utilisation, selon sa compréhension et les désirs du client, en faisant apparaître les différents échanges de messages entre les acteurs du système et le système lui-même.
A ce stade nous constatons que l'ingénieur système refait la rédaction des connaissances métiers selon son point de vue et son système futur. Donc, la réutilisabilité des connaissances métier à ce niveau de spécification n'existe pas.
Il est clair que la réutilisabilité des connaissances métier au niveau de l'entreprise est limitée aux acteurs de l'entreprise. Les connaissances métier ne sont pas réutilisables dans le monde informatique telles qu'elles sont dans le monde réel.

2.4 Le fossé entre l'IB et la spécification

L'ingénierie des besoins fournit plusieurs méthodes et techniques pour l'exploration et la rédaction des besoins. Parmi les plus importantes celle des approches par scénarios [Mallet 05]. Ces approches utilisent le langage naturel pour l'expression textuelle des scénarios. Elles décrivent les scénarios en détail et à un niveau différent d'abstraction. Les connaissances métier sont alors définies par un ensemble de scénarios textuels détaillés (informels).
Pour réussir une base de scénarios métier, cette dernière nécessite des experts en ingénierie de besoins ayant une expérience forte dans le domaine métier concerné.
Une base documentaire de scénarios qui représente un contexte métier parait une base complexe pour les ingénieurs système lorsqu'ils concevaient leurs systèmes et surtout lorsqu'ils ne possédaient pas le même degré d'expertise que ceux qui ont rédigé les scénarios.

Les ingénieurs des besoins ont écrit leurs scénarii pour refléter exactement le monde réel sans prendre en considération ce que le système informatique peut faire et supporter. A son tour, l'ingénieur système définit sa spécification (généralement semi formelle, formelle ou graphique) qui représente les scénarios (besoins) en fonction du système informatique et avec un langage de spécification différent de celui utilisé dans la rédaction des scénarios métier.
Plusieurs points sont à soulever:

- Le premier point il s'agit d'un fossé entre l'ingénieur des besoins et l'ingénieur du système qui parlent deux langages différents (langage informel pour l'ingénieur des besoins et langage formel ou semi formel pour l'ingénieur système).
- Le deuxième point il s'agit d'une spécification trop liée aux besoins ce qui provoque un couplage très fort entre les besoins et la spécification. Ce couplage fort ne permet pas une évolution et un changement facile de la spécification ; un découplage, pour une spécification, devient une solution majeure.
- Le troisième point est le rythme d'évolution des besoins métier qui ne suit pas le même rythme d'évolution de la technologie en informatique.

2.5 Conclusion

L'une des problématiques majeure de l'ingénierie des besoins et des méthodes de spécification est l'absence de lien opérationnel entre les deux qui visent, d'une part à concevoir une spécification de besoins par un modèle de conception simple à lire et à comprendre, et d'autre part à définir une implémentation logique prête à être implantée par un langage de programmation approprié. Ce lien opérationnel nécessite la mise en œuvre d'une approche de réutilisabilité des connaissances métier au niveau de l'ingénierie des besoins. Bien que les connaissances métier, dans plusieurs contextes d'entreprises ou dans la même entreprise, évoluent d'une façon permanente, elles suivent souvent un rythme discontinu (figure 2.1). Cette discontinuité signifie que ces connaissances, sur une durée de temps limitée, sont standards. C'est seulement à ce point qu'on peut profiter de la réutilisabilité des connaissances métier. Cette nouvelle approche nécessite la transformation des connaissances métier en des connaissances métier formelles et réutilisables. Ceci requiert un langage semi formel pour la représentation des connaissances métier. Cette approche sera développée dans le chapitre 3 en faisant apparaître la notion des « scénarios patterns ».

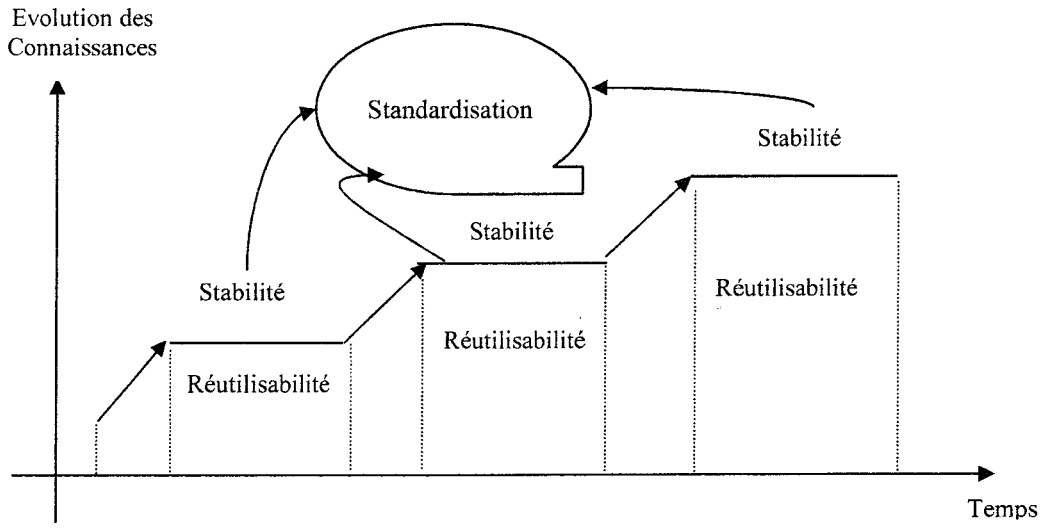


Figure 2.1: Evolution et réutilisation des connaissances métier

L'évolution des connaissances métier suit un rythme discontinu (figure 2.1). Cette discontinuité est due à la stabilité des connaissances métier dans le temps. Donc, la réutilisabilité des connaissances métier est une nécessité pour l'ingénierie des besoins et pour la conception pour plusieurs raisons :

- Atténuer le couplage fort entre les besoins et la spécification, afin de permettre une évolution et un changement facile de la spécification avec l'évolution et le changement des besoins.
- Inventer un langage semi formel pour la représentation des connaissances métier sous forme de scénarios patterns.
- Créer un environnement pour la réutilisabilité des scénarios patterns dans l'élaboration des modèles des besoins.
- Introduire un modèle de transformation du modèle des besoins en modèle d'analyse.

Environnement de Modélisation Intelligent (EMI) : Une nouvelle vision de développement des logiciels Chapitre 3

Le Raisonnement à Partir de Cas (RàPC), plus connu sous le nom de *CBR (Case Based Reasoning)*, est un paradigme de résolution de problèmes fondé sur la réutilisation d'expériences passées pour résoudre de nouveaux problèmes [Cordier et Fuchs 05]. Dans le RàPC les expériences passées sont décrites par des cas composés de problèmes et des solutions. Dans notre nouvelle vision de développement des logiciels, nous avons implanté les techniques des systèmes RàPC dans le noyau de l'Environnement de Modélisation Intelligent (EMI) qui constitue notre approche méthodologique de construction des logiciels. Le principe de base de l'EMI est de construire, pour des cas cibles, des solutions de conception en formalisme UML par la réutilisation des solutions de conception des cas similaires (cas sources) déjà testés et utilisés. Le but est de réduire les erreurs et le temps réservés pour l'élaboration des solutions de conception.

Dans ce chapitre nous présentons le principe et l'approche de notre nouvelle méthodologie de développement des logiciels, les concepts de base utilisés par cette méthodologie et le RàPC dans l'EMI.

3.1 Introduction

3.1.1 Principe et Approche

Le grand fossé entre l'ingénierie des besoins et la spécification préconise des coûts de maintenance et d'adaptation très élevés. Dans les deux chapitres précédents, nous avons présenté les points faibles de l'ingénierie des besoins ainsi les inconvénients de la spécification formelle, semi formelle ou graphique dus à l'absence d'un langage commun entre l'ingénieur système et le client.

Le génie logiciel a fourni beaucoup d'outils, de méthodes et de techniques d'aide à la modélisation et au développement des logiciels. Malgré le progrès en génie logiciel, la

réutilisabilité (capitalisation des connaissances par l'expérience) reste un moyen puissant rigide pour beaucoup des méthodes en génie logiciel.

La réutilisabilité peut se présenter tout au long du cycle de vie du logiciel. En programmation, les développeurs utilisent de plus en plus des composants opérationnels dans leurs applications. En conception, les patrons de conception (design patterns) furent un moyen efficace pour certains types d'application dont le but est de faciliter la réutilisabilité et la maintenance du code. Jusqu'à présent les patrons de conception ne sont pas normalisés même s'ils peuvent être classés en trois catégories ; les patrons de création, de structure et de comportement.

En analyse des besoins, l'approche de réutilisabilité n'est pas encore présente comme une vraie solution pour les problèmes de spécification et de conception. Malgré les méthodes de capitalisation des besoins que nous avons présentées dans le chapitre précèdent, le manque de la réutilisabilité des connaissances dans le début du cycle de vie du logiciel est toujours présent.

Notre approche s'inscrit dans le domaine de l'ingénierie intelligente des logiciels. Elle vise à développer un Environnement de Modélisation Intelligent (EMI) à base de connaissances et raisonnement à partir de cas (RàPC) et à faire contribuer les « scénarios patterns » dans l'élaboration d'une nouvelle vision de méthodologie de développement des logiciels. Le but de cet environnement est de permettre à un ingénieur système de générer d'une façon automatique un modèle de conception objet, en formalisme UML, à partir de besoins utilisateurs standards.

3.1.2 L'Environnement de Modélisation Intelligent (EMI)

L'EMI est basée sur une nouvelle méthodologie de développement qui constitue notre approche de l'ingénierie intelligente de logiciels. Cette approche donne plus de rôle pour les experts métier pendant le processus de standardisation des besoins. Ce rôle consiste à transformer les connaissances métier informelles (micro-besoins ou fragments de scénario métier) en connaissances semi formelles (scénarios prototypes) et à les regrouper dans une base de connaissances (base de cas) intégrée dans l'EMI.

L'EMI fournit pour les experts métier un langage de description des connaissances métier sous forme de scénarios prototypes appelés scénarios patterns. Ces scénarios patterns sont complétés par des schémas de sous modèles objets standard (patterns de conception) qui, ensemble, forment la base de cas d'un système RàPC. Une entrée de la base de cas est le couple scénario pattern (problème) et pattern de conception (solution), donc un cas est une paire (problème, solution).

L'ingénieur système, à partir des nouvelles exigences du client et la base de cas, construit d'une façon automatique un modèle de conception objet (modèle d'analyse) équivalent à ces nouvelles exigences.

La construction d'un modèle de conception objet à partir de l'EMI suit les étapes suivantes:

- L'ingénieur système et le client utilisent l'EMI pour construire un modèle des besoins relatif aux nouvelles exigences du client (nouveau cas cible).
- L'ingénieur système, applique les techniques des systèmes RàPC, pour sélectionner un cas similaire (cas source) au cas cible.
- L'EMI génère la pré-solution du cas similaire. Cette pré-solution est un pattern de conception représenté par un diagramme de classes (modèle d'analyse).
- L'ingénieur système peut adapter et valider la pré-solution pour en fournir le modèle d'analyse final.

Dans les paragraphes suivants nous allons montrer les concepts de base fondamentaux utilisés par l'environnement EMI, tels que les scénarios patterns, les patterns de conception, etc...

a) Besoins et scénarios

Les techniques d'Ingénierie des Besoins (IB) basées sur les scénarios visent à faire participer les utilisateurs et les néophytes de la conception aux processus d'Ingénierie des Besoins. En combinant l'approche méthodologique d'élucidation des besoins à des descriptions informelles, ils apportent une plus grande productivité par rapport aux techniques classiques de l'IB. Dans ce contexte, un problème important est l'intégration des approches par scénario dans les méthodes de l'IB [Ralyté et Ben Achour 97]. L'approche de Ralyté et Ben Achour propose une base de connaissances des différents modules de scénario. Cette base de connaissances permet d'identifier des scénarios appropriés au contexte du projet de l'IB et de les intégrer dans les méthodes de l'IB. L'identification du module de scénario est partiellement basée sur les propriétés internes des scénarios qui sont décrites dans un cadre de référence pour la classification des approches basées sur les scénarios.

Les premières étapes de tout processus de conception sont parmi les plus importantes. C'est à ce moment-là que se prennent des décisions majeures, presque instinctivement, à partir d'exigences informelles. Ces dernières sont souvent incomplètes, contradictoires, et en constante évolution [Amyot et *al.* 97].

Le passage d'exigences informelles à la spécification formelle d'une conception représente bien souvent un ravin dans lequel il est facile de trébucher. De la même façon qu'il n'est pas conseillé de « coder » un programme à partir des exigences, nous ne devrions pas « coder » directement une spécification à partir des exigences. Il faut tout d'abord s'assurer que ces dernières sont bien comprises. Afin de fournir un pont plus aisément franchissable aux concepteurs, des approches basées sur les scénarios, qui visent justement à morceler les exigences en parties plus compréhensibles, sont en pleine émergence.

Au cours des dernières années, nous avons remarqué une tendance accrue pour l'utilisation de scénarios dans les méthodes de conception de systèmes. L'émergence des scénarios d'usages (*use cases*) [Vallée et Roque 02] dans le monde de l'orienté objet confirma cette tendance. De nombreuses méthodologies sont maintenant proposées. Cependant, selon les approches, le terme « scénario » porte en son sein des sémantiques bien différentes les unes des autres. On associe parfois les scénarios à des traces (d'événements internes et/ou externes), à des échanges de messages entre composantes, à des séquences d'interactions entre un système et l'utilisateur, ou encore à des ensembles plus ou moins génériques de telles séquences, pour ne mentionner que ces quelques définitions. Nombreuses sont aussi les notations utilisées, qu'elles soient basées sur une grammaire textuelle plus ou moins formelle, sur des automates, ou sur des diagrammes d'échanges de messages semblables aux MSC [ITU 96]. Les approches diffèrent donc sur plusieurs points en fonction de la définition retenue.

Afin de bien établir le contexte de notre travail, nous présentons dans la section suivante la définition d'un scénario, suivi du détail de l'approche utilisée par notre méthodologie.

b) Définition de scénario

Un scénario est une séquence de responsabilités (événements et activités), internes ou externes, qui sont reliées de façon causale dans le but d'offrir une fonction précise du système [Amyot et *al.* 97].

Tout scénario n'est qu'un fragment de connaissance du système pour lequel les besoins sont spécifiés. Les relations entre scénarios, et les relations entre scénarios et besoins sont rarement rendues explicites par des définitions systématiques. La relation scénario - besoin n'est pas triviale : les scénarios sont-ils des besoins ? Si non, de quelle manière s'y rattachent-ils ? En sont-ils issus ou inversement les besoins font-ils partie des scénarios ? Les réponses à ces questions sont présentées dans les travaux réalisées en projet SPINOV basée sur l'approche CREWS [SPINOV 05] qui à partir de l'écriture des scénarios métier les besoins seront découverts.

c) Notion de scénario prototype et micro-besoin

Donc, la relation entre scénario et besoin est très dépendante ; la description d'un besoin par un scénario est une forme textuelle informelle que les hommes sont capables de la comprendre.

Lorsque les scénarios descriptifs des besoins deviennent très grands il faut les décomposer en scénarios réduits ; ceci aboutit à comprendre les besoins complexes par compréhension de leurs micros besoins. Un besoin est une connaissance particulière dans un domaine. Cette connaissance est stable pour une durée fixe de temps, ceci permet de la réutiliser avant son évolution. Le scénario est un fragment de connaissances réutilisables dans des contextes similaires ; donc il est standard. Alors, la réutilisabilité des scénarios métier est fortement possible [Hussein et *al.* 06 (a)].

Pourquoi rédigeons-nous des scénarios pour comprendre des besoins qui sont standards et qui peuvent appartenir à des domaines similaires. Notre approche vise à réutiliser les scénarios métiers dans le contexte de développement des applications informatiques.

Les scénarios métiers sont alors des scénarios prototypes créés pour être utilisés par des ingénieurs systèmes afin de construire un modèle des besoins qui répond aux exigences des clients.

d) Standardisation des besoins et Réutilisabilité

La standardisation des besoins, par des scénarios prototypes, montre l'efficacité de cette technique dans la réutilisabilité des scénarios. Les connaissances métier ne changent pas d'une entreprise à une autre ou d'un système à un autre mais elles peuvent se développer ou se présenter de façons différentes selon l'intellectualité ou l'expérience des gens. Pour cela, la réutilisabilité des scénarios métiers, dans des entreprises sous construction ou dans des nouveaux systèmes d'informations, réduit la possibilité d'inadéquation entre les besoins réels du métier et la spécification produite [Hussein et *al.* 08]. Donc, les connaissances métier, fournies par les experts métiers, construisent le cadre informationnel des scénarios métiers.

Pourquoi la réutilisabilité des scénarios métiers ?

La phase d'analyse qui débute le développement des applications informatiques possède l'impact essentiel sur la réussite ou l'échec d'une application informatique [Hussein et *al.* 06], car dans cette étape se mettent les points sur la compréhension des besoins par l'ingénieur système en faisant une certaine entente particulière avec les utilisateurs et les « stakeholders ».

Mais, en conséquence, ce que les utilisateurs attendent du système évolue bien plus rapidement qu'auparavant. Leurs besoins ne sont pas donc stables [harker et *al.* 93] Comprendre les impacts d'un changement organisationnel sur les systèmes informatiques existants reste un problème ouvert [Lubars et *al.* 93]. La prise en compte de l'évolution des besoins durant la phase de développement d'un système reste elle aussi problématique [Curtis 88].

Les besoins évoluant, il n'est plus possible de les considérer comme donnés a priori. Il est plutôt nécessaire de déterminer les nouveaux besoins pour les systèmes existants et développer des modèles représentants fidèlement les besoins et leurs évolutions, durant tout le cycle de développement. De plus, le rôle central de l'ingénieur système doit être reconsidéré, l'ingénierie des besoins requière la participation d'un grand nombre d'acteur de l'organisation, chacun apportant sa vision sur ce le système devrait faire [Finkelstein90]. Enfin, la validation des besoins doit être basée sur les besoins réels de l'organisation plutôt que sur les fonctionnalités du système. C'est seulement à cette condition que les systèmes informatiques seront capables de s'adapter aux changements organisationnels.

En s'attaquant à ces problèmes, le rôle de l'ingénieur système et des acteurs de l'entreprise doivent s'articuler autour des connaissances métier de base et non pas autour du système à développer. Alors, les connaissances de base ou les scénarios prototypes peuvent décrire le système à développer. Ainsi, les scénarios métiers prototypes de base, rédigés par les experts métiers, peuvent s'évoluer facilement avec le changement organisationnel.

De ce point de vue la partie analyse dans le développement des applications informatiques sera en question. Dès maintenant, il faut penser au moment de l'analyse de réutiliser les besoins sous formes de scénarios métiers.

e) Les Scénarios Patterns

Les scénarios patterns (SP) sont des schémas réutilisables de scénarios métier standard qui sont rédigés avec un langage semi formel dans le but est de permettre l'exploitation des SP dans des divers domaines d'application [Hussein et *al.* 08].

Les scénarios patterns sont des supports puissants pour la capitalisation des connaissances par l'expérience. Afin de faciliter la réutilisabilité des SP, les scénarios patterns sont regroupés dans des classes de cas d'utilisation. Par exemple dans le domaine bancaire, les scénarios patterns du cas d'utilisation « retrait argent » sont des instances de la classe de ce cas d'utilisation.

Selon le formalisme UML, les cas d'utilisation permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système. Ils centrent l'expression des exigences du système sur ses utilisateurs : ils partent du principe que les objectifs du système sont tous motivés.

Un cas d'utilisation est une abstraction de plusieurs chemins d'exécution. Une instance de cas d'utilisation est appelée : « scénario ». Chaque fois qu'une instance d'un acteur déclenche un cas d'utilisation, un scénario est créé (le cas d'utilisation est instancié). Ce scénario suivra un chemin particulier dans le cas d'utilisation. Un scénario ne contient pas de branche du type « Si condition ... alors » car pendant l'exécution, la condition est soit vraie, soit fausse, mais elle aura une valeur. Après la description des cas d'utilisation, il est nécessaire de sélectionner un ensemble de scénarios qui vont servir à piloter l'itération en cours de développement.

Le choix et le nombre de scénarios à retenir est une étape difficile à réaliser : l'exhaustivité est difficile, voire impossible à atteindre. Le nombre d'instances pour un cas d'utilisation peut être très important, voire infini. Les scénarios peuvent être classés en :

- scénarios principaux : il correspond à l'instance principale du cas d'utilisation. C'est souvent le chemin « normal » d'exécution du cas d'utilisation qui n'implique pas d'erreurs.

- scénarios secondaires : il peut être un cas alternatif (un choix), un cas exceptionnel ou une erreur.

Les scénarios sont utiles pour :

- analyser et concevoir le système,
- justifier les choix effectués (ils serviront à la documentation des cas d'utilisation),
- tester : les scénarios constituent le meilleur moyen de spécifier les tests.

f) Utilisation des cas d'utilisation

La portée des cas d'utilisation dépasse largement la définition des besoins du système. Les cas d'utilisation interviennent tout au long du cycle de vie du projet, depuis le cahier des charges jusqu'aux tests.

Intervenant	Utilisateur	Analyste	Architecte	Programmeur	Testeur
Rôle des cas d'utilisation	Exprimer	Comprendre	Concevoir	Réaliser	Vérifier

g) Les cas d'utilisation orientés sujet

Un cas d'utilisation orienté sujet regroupe des scénarios qui se rapportent à un même sujet dans un domaine particulier. Les acteurs (les utilisateurs du système futur) des cas d'utilisation jouent un rôle important dans l'expression des besoins du système, cette expression est représentée sous forme d'interaction de messages échangés avec le système (scénarios). En effet, ces utilisateurs ne sont pas des experts métier (domaine d'application), alors l'analyste pourra mal comprendre les exigences des utilisateurs. Donc, le modèle de besoins conçu peut ne pas répondre aux exigences des utilisateurs.

L'analyste, à l'aide de la notation UML, décrit les scénarios de cas d'utilisation sous différentes formes de diagrammes selon sa compréhension des besoins exprimés par les utilisateurs. Il est clair d'après le tableau précèdent qu'il y a un fossé entre l'utilisateur et l'analyste (entre l'expression et la compréhension), ceci est dû à l'absence d'un langage commun entre l'utilisateur et l'analyste. Pour remédier à cette problématique nous avons adopté un langage de description des connaissances métier compréhensible par les deux acteurs sous forme de schémas de scénarios (scénario Patterns). L'utilisateur et l'analyste peuvent se communiquer à travers une base de scénarios classés par des cas d'utilisation orientés sujet.

L'analyste utilise les schémas de scénarios patterns pour décrire les besoins informels de l'utilisateur. Donc, l'analyste transforme et adapte ces besoins informels aux scénarios patterns semi formels et structurés (modèle des besoins). L'architecte, après une compréhension approfondie du modèle des besoins définit par l'analyste, utilise des techniques de transformation de modèles pour concevoir un schéma de conception prêt à être réalisé par un programmeur et ensuite vérifié par un testeur.

Intervenant	Utilisateur	Analyste	Architecte	Programmeur	Testeur
Rôle des cas d'utilisation	Exprimer à l'aide des schémas de scénarios informels	Comprendre à l'aide des schémas de scénarios semi formels	Concevoir à l'aide de transformation des modèles	Réaliser	Vérifier

h) Base de connaissances et classes de cas d'utilisation

Les connaissances métier, qui sont formulées par des schémas des scénarios, constituent le contexte général de l'évolution du métier en fonction du progrès technologique. Ainsi, ces connaissances ne changent pas d'un environnement à un autre ou d'un pays à un autre, mais elles peuvent être exploitées partiellement ou totalement et selon l'expertise dans le domaine. Bien que les connaissances s'évoluent avec le temps elles resteront à tout moment prêtes à être réutilisées.

Les schémas de scénarios traduits par scénarios patterns peuvent fournir un moyen fort pour la description semi formelle des connaissances métier dans leur évolution et développement. Pour les gens travaillant dans le domaine de l'intelligence artificielle, la représentation des connaissances métier à l'aide des scénarios patterns est une technique très efficace. Elle assure un support évolutif et maintenable pour la représentation des connaissances métier.

3.1.3 De la connaissance métier vers le scénario pattern

Un métier fournit un ensemble de connaissances exploitées par les gens ou la machine. Lorsque l'homme travaille dans un métier, il exécute un ensemble d'activités attachées à ce métier. Pour faire réussir un métier, dans une entreprise, un établissement ou autre, il faut exécuter ces activités selon un modèle de processus bien expérimenté. Notons ici que l'expérience dans un domaine quelconque peut fournir de critères de qualité distingués.

Le recours aux activités métier expérimentées est le seul moyen pour mettre le métier au défi de la technologie et l'évolution rapide. Un modèle de processus qui modélise une activité dans un métier est donné dans la figure suivante (figure 3.1) qui montre le modèle de processus pour un scénario métier (scénario pattern) (SP) comme une chaîne d'opérations abstraites et indique les différents types possibles de SPs extraits de ce modèle.

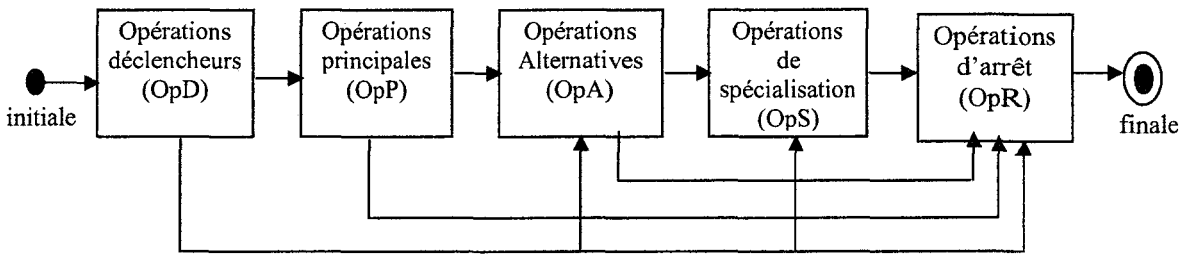


Figure 3.1 : Modèle de processus du scénario pattern (SP)

Le SP complet est construit par une chaîne d'opérations (OpP, OpA et OpS) et il est délimité par une opération initiale (OpD) et une opération finale (OpR). L'opération initiale appelée encore opération déclencheur indique le comportement du scénario pattern au sein du processus. L'opération d'arrêt indique le comportement du scénario pattern à la fin du processus. Chaque SP peut être caractérisé par un ensemble d'opérations principales, alternatives ou de spécialisation. Les opérations principales décrivent le(s) but (s) général (aux) du SP. Les opérations alternatives décrivent les buts alternatifs du SP et elles peuvent être traitées comme des compléments pour les opérations principales.

Les opérations de spécialisation décrivent les buts spécialisés ou privés des SPs. En général, les opérations de spécialisation représentent des points de vue différents pour les mêmes systèmes. Plusieurs instances du processus SP peuvent être extraites de ce modèle. Le SP complet, le SP réduit, le SP principal, le SP alternatif et le SP de spécialisation. Cet ensemble d'instances définit une classe qui modélise le cas d'utilisation du scénario pattern [Jacobson et *al.* 1992]. Le SP réduit est construit seulement par les opérations déclencheurs et les opérations d'arrêt. Le SP alternatif contient au moins une opération alternative (OpA), s'il est accompagné par des opérations principales (OpP) il devient alternatif spécialisé. Le SP alternatif est délimité par les opérations déclencheurs et les opérations d'arrêt. Le SP de spécialisation contient au moins une opération de spécialisation et il est délimité par les opérations déclencheurs et les opérations d'arrêt. Le SP principal contient au moins une opération principale (OpP) et il est délimité par les opérations déclencheurs et les opérations d'arrêt. Lorsqu'il est accompagné par au moins une opération alternative il devient principal alternatif et s'il est accompagné par au moins une opération de spécialisation il devient principal spécialisé (figure 3.2).

Exemples :
SP réduit: OpD, OpR
SP alternatif : OpD, OpA, OpR.
SP de spécialisation : OpD, OpS, OpR.
SP principal: OpD, OpP, OpR.
SP principal alternatif: OpD, OpP, OpA, OpR.
SP principal spécialisé: OpD, OpP, OpS, OpR
SP complet: OpD, OpP, OpA, OpS, OpR

Formalisme en UML des scénarios pattern à l'aide de classes de cas d'utilisation

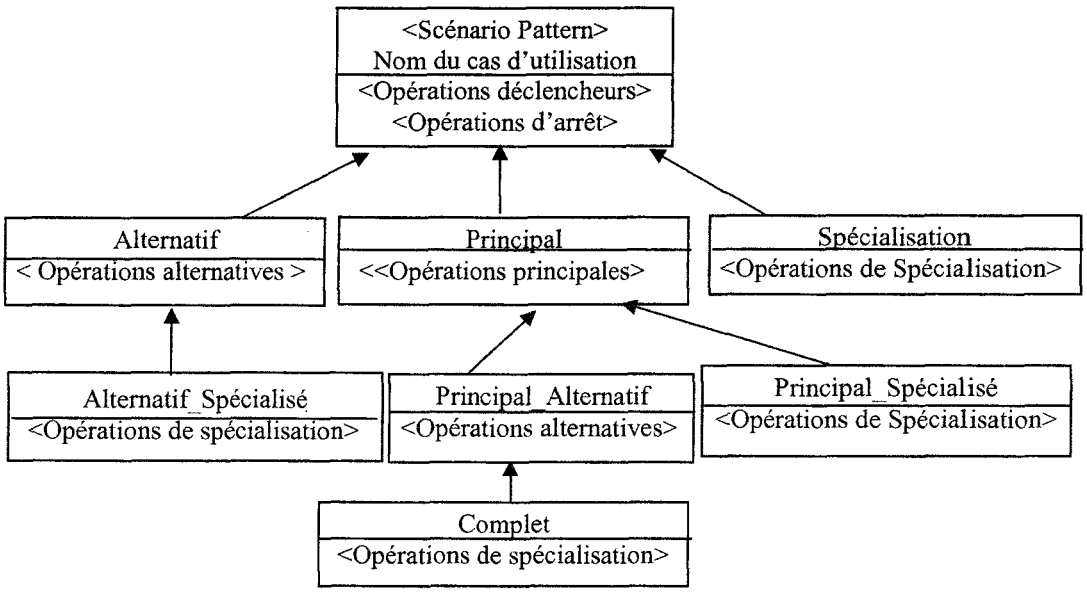


Figure 3.2 : Classes de cas d'utilisation des scénarios patterns

3.2 Langage de description des scénarios patterns

Ledang a proposé une approche pour développer une spécification B à partir d'un modèle de cas d'utilisation d'un système à construire. L'approche de Ledang fournit un cadre pour analyser formellement la cohérence du modèle des besoins produit dans un processus de développement par objets et elle permet d'intégrer la méthode B dès la phase d'élicitation des besoins dans un processus de développement de logiciels [Ledang 01].
L'approche de Ledang a ignoré dans la spécification B les objets concernés par les opérations. Les cas d'utilisation sont vus comme des chaînes d'opérations sans donner aucune importance aux objets réels du domaine. Ceci nous semble faible du point de vue modélisation du monde réel.

Pour éviter le manque de l'approche de Ledang, nous avons adopté dans la description des scénarios patterns les objets du domaine d'application. Alors, le modèle de scénarios patterns est celui du Ledang plus les objets du domaine. Le modèle constitué par les opérations et les objets semble plus proche des modèles utilisés par l'approche objet. La construction du modèle de scénarios patterns est effectuée à l'aide d'un langage spécial appelé langage de description des scénarios patterns (LSP). Dans les sections suivantes nous présentons la grammaire et la syntaxe du langage LSP.

3.2.1 La grammaire du langage LSP

a) Alphabet du langage

L'alphabet du langage LSP est construit par les lexèmes suivants :

- Identificateurs
- Mots réservés

- Chaînes
- Nombres
- Lettres
- Caractères spéciaux

Les identificateurs sont composés de lettres ou une combinaison de lettres et de chiffres. Les identificateurs commencent obligatoirement par une lettre.
<Identificateur> :: <Lettre>| <Lettre><Nombre>| <Lettre> <Identificateur>
<Lettre> :: A|B|.....|Z|a|b|.....|z
<Nombre> :: <Chiffre>| <Chiffre><Nombre>
<Chiffre> :: 0|1|2|3|4|5|6|7|8|9

Les chaînes, dans le langage LSP, représentent les noms des objets, des opérations, des cas d'utilisation, leurs types, et des acteurs. Elles sont constituées par au moins trois lettres.
<Chaîne> :: <Lettre><Lettre><Lettre>| <Lettre><Chaîne>

Les mots réservés sont limités aux mots suivants :
<Mots réservés>:: OpD, OpP, OpS, OpA, OpR, boolean, SP, BEGIN et END.

Les caractères spéciaux utilisés dans le langage LSP sont: « ; » , « , » , « / »

b) La syntaxe du langage

<SP> :: BEGIN <Nom SP> ; <Type SP> ; <Acteurs> ; /SP <Lignes> SP/ END

<Nom SP> :: <Chaîne>

<Type SP> :: <Chaîne>

<Acteurs> :: <Chaîne> | <Chaîne> ; <Acteurs>

<Lignes> :: <ligne>|<Ligne><Lignes>

<Ligne>::<Nom objet>,<Type Opération>,<Niveau>,<Nom Opération>, <Ids> ;<Ids>

<Nom objet> :: <Chaîne>

<Type Opération> :: OpD| OpP| OpS| OpA| OpR

<Niveau> :: <Nombre>

<Nom opération> :: <Chaîne>

<Ids> :: <Identificateur> | <Identificateur> , <Ids>| boolean

Un scénario pattern est délimité par les deux mots réservés BEGIN et END. Le mot réservé BEGIN est suivi par le nom du SP, son type et les éventuels acteurs.
Les commentaires dans un SP commencent par le symbole // et ils sont ignorés par le compilateur du langage.

N.B : Dans "Ligne", le premier composant « Ids » représente des identificateurs de type entrée et le deuxième composant « Ids » ceux de type de sortie. L'un des composants « Ids » peut être vide. Le SP utilise un nom d'objet particulier appelé « interface » pour gérer les entrées et les sorties dans "Ligne".

Exemple de la syntaxe semi formelle pour la rédaction d'un scénario pattern :

BEGIN
//Commentaire//
<Nom du SP> ;
<Type du SP> ;
<Acteurs du SP>;
/SP
// Plusieurs lignes d'opérations agissant sur un ou plusieurs objets//
.....
.....
.....
.....
<Nom objet>,<Type opération>,<niveau>,<nom opération>,<Ids> ;<Ids>
<Nom objet>,<Type opération>,<niveau>,<nom opération>,<Ids> ;<Ids>
.....
.....
.....
.....
SP/
END

Un exemple d'utilisation du langage LSP, traitant les scénarios utilisés par les machines GAB de la banque AUDI et la banque SGBL au LIBAN, est détaillé en annexe D.

Scénario pattern principal:

BEGIN
Retrait d'argent ;
Principal ;
Client ;
/SP

GAB, OpD, 1, insérer ; boolean
Carte, OpD, 2, valider, numéroCarte, dateExpiration ; boolean
interface, OpD, 3, saisir ; motDePasse
Carte, OpD, 4, vérifier, motDePasse ; balance, boolean
interface, OpP, 1, sélectionner ; option
interface, OpP, 2, sélectionner ; monnaie
interface, OpP, 3, sélectionner ; montant
Carte, OpP, 4, autoriser, numéroCarte, solde ; boolean
GAB, OpP, 5, rejeter ; boolean

GAB, OpP, 6, rejeter ; boolean
GAB, OpR, 1, récupérer, boolean
GAB, OpR, 2, récupérer, boolean
GAB, OpR, 3, réinitialiser, boolean
SP/
END

c) La sémantique des scénarios patterns

La standardisation des scénarios patterns veut dire que les connaissances métier utilisent le même vocabulaire (c'est-à-dire les mêmes noms des objets, des opérations, etc.). Dans le langage LSP, ce vocabulaire définit la sémantique du langage. Le LSP possèdera un dictionnaire de connaissances métier qui doit être largement suffisant pour valider et vérifier les objets et les opérations utilisés par un scénario pattern.

La standardisation des connaissances métier à l'aide d'un dictionnaire vise à réduire le taux d'erreurs sémantiques lors de la rédaction des scénarios patterns. Lorsque les experts métier rédigent leurs scénarios patterns il est probable qu'il y aura des fautes d'orthographe ou de vocabulaire dues au problème de saisi. Pour pallier ce problème, la vérification sémantique devient un support de contrôle puissant pour les scénarios patterns.

Le dictionnaire de connaissances métier peut être enrichi par des nouvelles connaissances que l'expert métier décrit au moment de la rédaction des scénarios patterns pour évoluer le standard des connaissances métier. Les éléments de base du dictionnaire de connaissances métier sont : le domaine du métier, le sous domaine du métier, l'activité dans le métier, ses acteurs, ses noms d'opérations et ses noms d'objets. Chaque élément du dictionnaire possède une description sémantique dans son contexte. Exemple :

Nom du domaine	Bancaire	
Nom du sous domaine	Machine GAB	
Nom de L'activité	Retrait d'argent	
Nom acteur	Client	
Nom de l'opération	insérer carte	Insérer carte dans le lecteur de la machine GAB
Nom abstrait de l'opération	insérer	
Nom de l'objet	GAB	Guichet automatique de billets

3.2.2 Spécification semi formelle et méta-modèle des scénarios patterns

A l'aide du langage LSP, les besoins informels de l'expert métier sont traduits dans une spécification semi formelle basée sur la notion des scénarios patterns. Le LSP permet, lors d'une étape de spécification, d'éliminer les ambiguïtés du langage naturel et

les mauvaises interprétations qui en découlent. Les méthodes et langages de la spécification semi formelle possèdent tous trois composantes :

1. Une syntaxe qui précise la notation utilisée pour procéder à la spécification,
2. Une sémantique qui donne une définition non ambiguë de cette syntaxe,
3. Un ensemble de relations qui définissent les règles qui donnent les propriétés des objets satisfaisant la spécification.

Notre compilateur du langage LSP est constitué de trois composants : l'analyseur lexical, l'analyseur syntaxique et l'analyseur sémantique (figure 3.3).

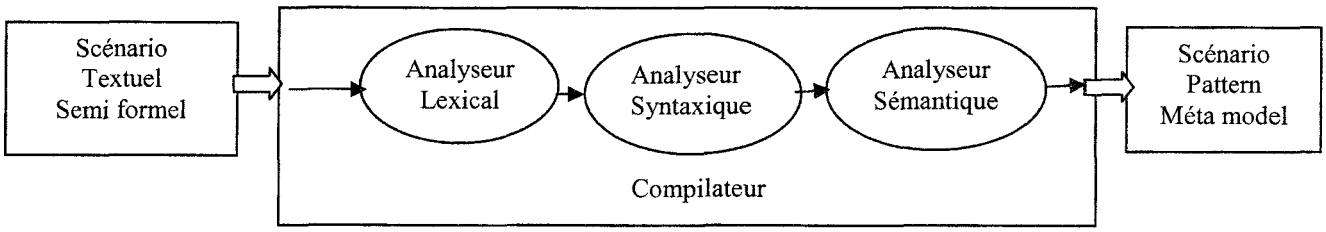


Figure 3.3: Noyau du langage LSP

Le LSP se présente comme un langage de communication entre l'expert métier, l'ingénieur système et la machine. Le scénario métier tel qu'il est saisi et compilé par l'expert métier, et qui est devenu un scénario pattern, est sauvegardé dans une base de scénarios patterns. Cette base est structurée de manière à permettre, facilement, à l'ingénieur système de sélectionner et de consulter les scénarios patterns relatifs au contexte du métier. La base de scénarios patterns est la structure physique de stockage des scénarios. Pour rendre simple l'accès à cette base, les scénarios patterns sont classifiés selon le domaine du métier, le sous domaine et le type du scénario (principal, alternatif, spécialisé, etc.....).

Un scénario pattern est alors identifié par le quadruplet (nom du domaine, nom du sous domaine, un cas d'utilisation, un type du scénario et un séquentiel), le séquentiel est une donnée de distinction lorsque plusieurs scénarios de même type appartiennent à un même cas d'utilisation.

Méta-modèle du scénario pattern

Le modèle d'un scénario pattern est une instance du méta-modèle qui décrit le concept de base des scénarios patterns. Le méta-modèle du scénario pattern est représenté en formalisme UML. Un scénario pattern est un ensemble des lignes scénaristes dans lesquelles nous présentons les opérations et leurs objets (figure 3.4).

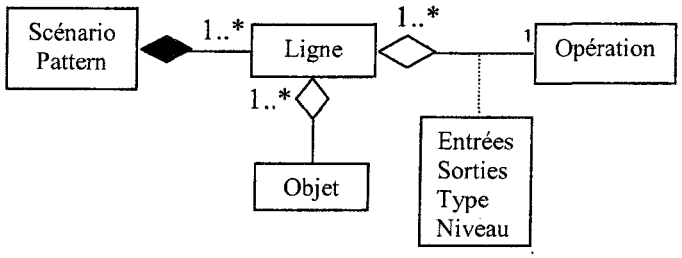


Figure 3.4 : Méta-modèle du scénario pattern

Avec le méta-modèle, le concept des scénarios patterns sera plus rigide et peut se développer facilement. Les techniques de métamodélisation fournissent des mécanismes permettant de manipuler des modèles. Elles conduisent à disposer d'un format concret de représentation pour les modèles afin que ceux-ci soient manipulables par des outils informatiques [Kadima 05]. Dans notre approche, le méta-modèle permet de valider la transformation du modèle des besoins en un modèle d'analyse.

Base de connaissances métier vers la base de scénarios patterns

L'un des problèmes majeurs des connaissances métier relevées de l'expérience est l'absence d'une méthode ou outil simple permettant aux experts métier de sauvegarder leurs connaissances métier pour une réutilisation ultérieur.
Les méthodes de capitalisation des connaissances métier telles que nous avons décrit au chapitre 2 ne fournissent pas un moyen opérationnel pour réutiliser les connaissances métier dans un contexte informatique. Le manque d'un langage de représentation des connaissances métier qui soit opérationnel à la fois par l'expert métier et la machine reste en question. C'est à ce point là ou notre approche vise à combler ce fossé entre l'expert et la machine. La capitalisation des connaissances métier, basée sur la notion des scénarios patterns et leur représentation semi formelle à l'aide du langage LSP, permet de les rendre réutilisable par les systèmes informatiques. Les experts métier, à l'aide du langage LSP, transforment leurs connaissances métier en un ensemble de scénarios patterns classifiés selon leurs domaines métier et raffinés en sous domaines et ensuite en cas d'utilisation.

Les experts métier préparent la base de connaissances métier sous forme de base de scénarios patterns. Par le biais du LSP, les experts métier peuvent éditer et modifier des anciens scénarios patterns en vu de les corriger sémantiquement ou de les faire évoluer.
La base de scénarios patterns (BSP) est représentée en formalisme UML (figure 3.5).

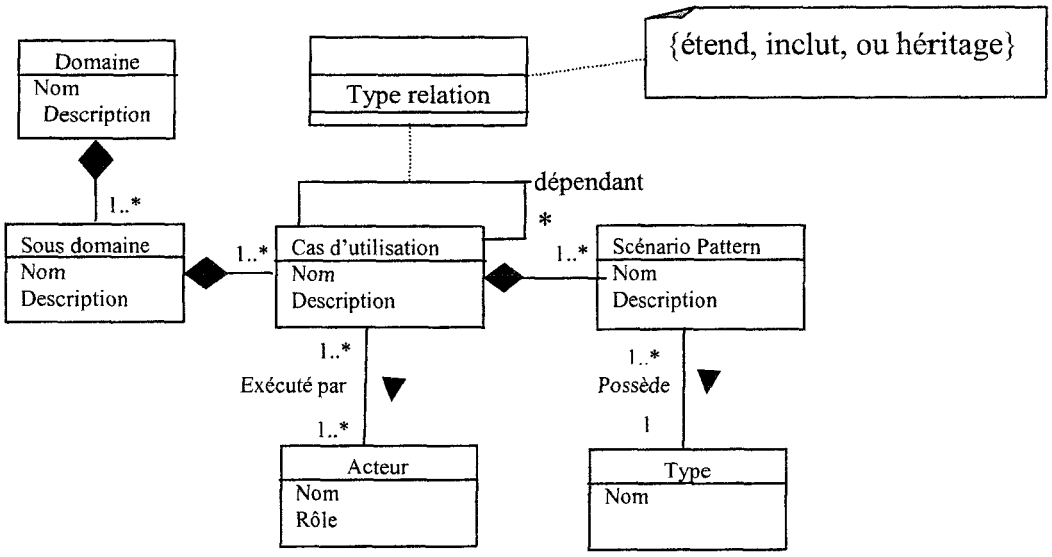


Figure 3.5: Base de scénarios patterns

3.2.3 Modèles objets des scénarios patterns

Selon le méta-modèle de la figure 3.4, les objets manipulés par les scénarios patterns sont décrits par un ensemble d'attributs et d'opérations relevés à partir des lignes du scénario pattern. Le scénario pattern principal de l'exemple présenté dans la section 3.2.1 sera modélisé de la façon suivante :

Scénario pattern principal:

BEGIN
Retrait d'argent ;
Principal ;
Client ;
/SP
GAB, OpD, 1, insérer ; boolean
Carte, OpD, 2, valider, numéroCarte, dateExpiration ; boolean
interface, OpD, 3, saisir ; motDePasse
Carte, OpD, 4, vérifier, motDePasse ; solde, boolean
interface, OpP, 1, sélectionner ; option
interface, OpP, 2, sélectionner ; monnaie
interface, OpP, 3, sélectionner ; montant
Carte, OpP, 4, autoriser, numéroCarte, montant ; boolean
GAB, OpP, 5, rejeter ; boolean
GAB, OpP, 6, rejeter ; boolean
GAB, OpR, 1, récupérer, boolean
GAB, OpR, 2, récupérer, boolean
GAB, OpR, 3, réinitialiser, boolean
SP/
END

Une illustration des instances du méta-modèle du SP est donnée en annexe E. Un exemple d'instance de scénarios patterns décrivant la première ligne du scénario pattern principal du cas d'utilisation « Retrait d'argent d'une machine GAB » (figure 3.6).

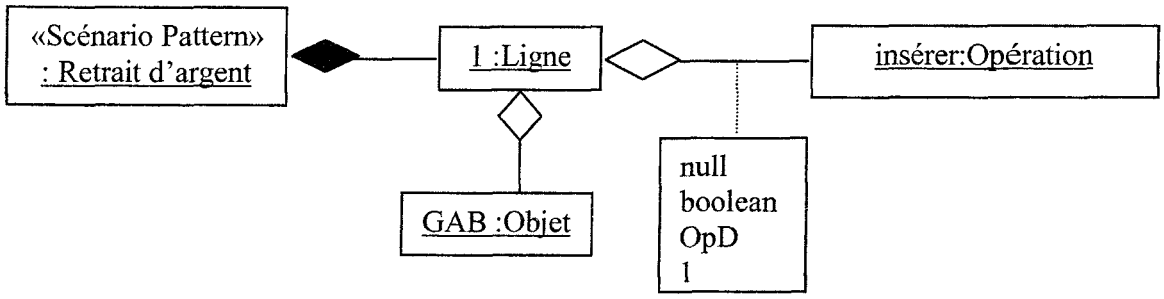


Figure 3.6: Instance du Méta-modèle du scénario pattern « Retrait d'argent »

Les objets retirés du scénario pattern principal sont: GAB, interface et Carte. Les objets sont caractérisés par un ensemble d'attributs et d'opérations. La table 3.1 présente l'ensemble de ces caractéristiques extraites du scénario pattern principal.

Objet	Attributs	Opérations
GAB	état option monnaie montant	insérer rejeter //carte rejeter //monnaie récupérer // carte récupérer // monnaie réinitialiser
Interface	motdepasse option monnaie montant	saisir sélectionner//option sélectionner // monnaie sélectionner // montant
Carte	numéroCarte dateExpiration motDePasse solde	valider vérifier autoriser

Table 3.1: Recueil des caractéristiques des objets du SP principal

3.3 Introduction aux systèmes RàPC

Le Raisonnement à Partir de cas (RàPC) est un paradigme de raisonnement qui s'appuie sur cinq opérations de base ; l'élaboration, la remémoration, l'adaptation, la révision et l'apprentissage gravitant autour d'une base de connaissances (appelée base de cas) du domaine d'application (figure 3.7).

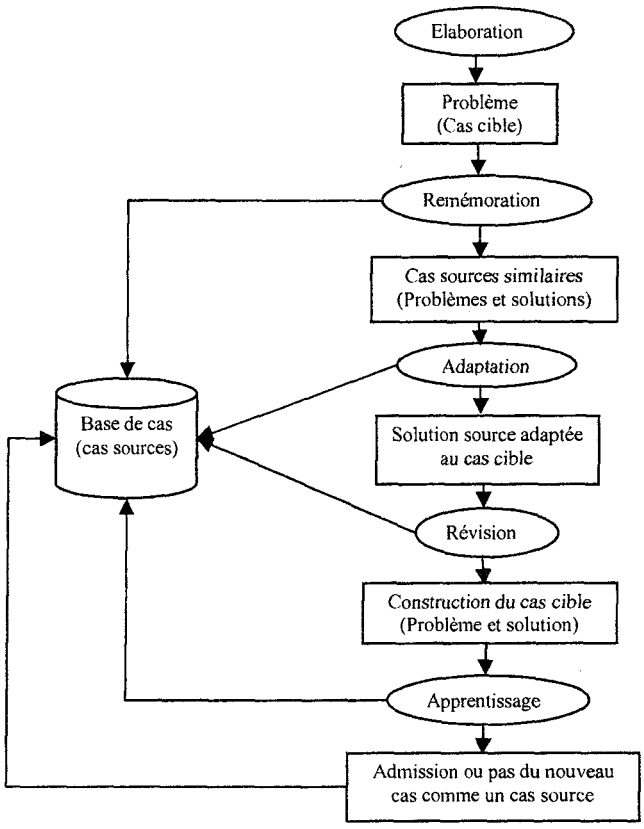


Figure 3.7: Opérations de base d'un système RàPC

La remémoration de problèmes passés résolus, appelés les cas sources, pour résoudre de nouveaux problèmes, appelés les problèmes cibles [Aamodt et Plaza 94]. Le RàPC est utilisé dans des nombreux systèmes industriels pour résoudre des problèmes dans des domaines similaires [Renaud et *al.* 08].

Selon [Cordier et Fuchs 05], la première étape du RàPC est de construire la spécification du problème à résoudre. Élaborer un cas pertinent, c'est-à-dire qui à la fois, reflète le problème courant et soit exprimé en termes cohérents par rapport à la base de connaissances utilisée est une vraie question d'ingénierie des connaissances.

Le problème cible élaboré est ensuite utilisé dans l'étape de remémoration pour retrouver un cas similaire dans la base de cas. Déterminer la similarité entre deux cas est loin d'être une opération triviale. Elle implique l'utilisation de mesures et de connaissances de similarités fortement liées au domaine d'application. Une étude détaillée qui s'intéresse à la similarité a été proposée dans [Rifqi 96].

Durant l'étape d'adaptation, la solution du cas source est adaptée pour obtenir une solution au problème cible. L'adaptation est un des processus les plus délicats du RàPC. Dans [Fuchs *et al.* 00], les auteurs ont proposé une approche de l'adaptation s'appuyant sur la notion d'influence du problème sur la solution qui, combinée avec les appariements effectués au moment de la remémoration, permet d'adapter la solution du cas cible. Longtemps, les connaissances utilisées pour la remémoration et pour l'adaptation ont été considérées comme distinctes. Smyth et Keane [Smyth et Keane 93] ont proposé d'utiliser les connaissances d'adaptation au moment de la remémoration pour privilégier la remémoration des cas minimisant l'effort d'adaptation. Si la remémoration est faite en anticipant correctement sur l'adaptation, cette dernière en sera d'autant plus facilitée et l'on sera sûr d'avoir un cas adaptable. Les connaissances de similarité et d'adaptation apparaissent alors comme étant duales voire même confondues.

C'est lors de la phase de révision que la solution proposée peut être corrigée, acceptée ou refusée par l'utilisateur. Cette étape permet d'évaluer l'adaptation. Elle permet également de préparer l'apprentissage puisqu'elle fait émerger de nouvelles connaissances : une nouvelle solution (acceptée ou refusée) mais aussi des connaissances d'adaptation si certaines corrections ont été apportées par l'utilisateur [Aamodt 91], ou des connaissances de remémoration si celle-ci n'est pas jugée satisfaisante [Fox et Leake 94]. Le résultat de l'évaluation de la solution met en évidence une insuffisance de l'adaptation à produire une solution satisfaisante ou de la remémoration à sélectionner le cas adéquat. L'étape de révision permet également d'évaluer l'utilité du cas nouvellement résolu et d'élaborer une stratégie de rétention ou d'oubli des cas selon leur contribution à la compétence du système [Smyth et Keane 95].

La phase d'apprentissage soulève également un certain nombre de problématiques de recherche. Les questions qui se posent sont avant tout de savoir quelles sont les connaissances qui doivent être apprises et comment les apprendre. La plupart des recherches portent sur l'apprentissage des cas passés résolus, des méthodes d'indexation et de l'organisation de la base de cas, mais plus rares sont les recherches qui comme [Aamodt 91] s'intéressent à l'apprentissage de connaissances implicites telles que les connaissances de similarité ou d'adaptation.

3.3.1 Interaction entre RàPC et EMI

Selon le paradigme du RàPC, notre approche vise à intégrer le concept des systèmes RàPC dans l'Environnement de Modélisation Intelligent (EMI). Le cas source du RàPC, qui contient le problème et sa solution, est représenté par le scénario pattern et son pattern de conception dans l'EMI. Le problème source est décrit par le scénario pattern et la solution source est décrite par le pattern de conception.

Dans l'EMI, La base de cas est représentée par une base de patterns (figure 3.8) et elle est manipulée par l'ingénieur système comme étant des composants réutilisables mais adaptables. L'ingénieur système après avoir sélectionner la solution (pattern de conception), il peut l'adapter en introduisant certaines modifications pour qu'elle soit plus pertinente aux besoins du client.

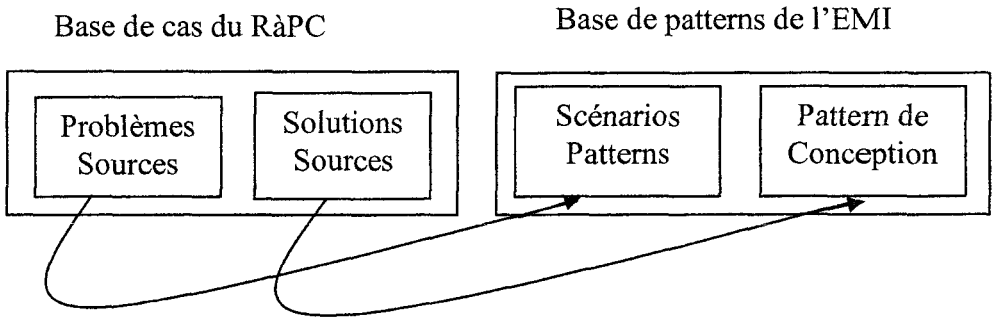


Figure 3.8: Base de patterns de l'EMI par rapport au RàPC

3.3.2 Le RàPC au noyau de L'EMI

Le RàPC comme nous l'avons décrit dans les paragraphes précédents est un paradigme de résolution des problèmes basé sur des cas passés. Le concept de base du RàPC s'appuie sur une base de cas où le cas présente un problème avec sa solution. L'EMI profite de ce concept pour réutiliser des solutions de conception passées (appelées patterns de conception) et les faire adapter à des cas cibles dans des domaines similaires.

3.3.2.1 Base de cas et base de scénarios

Les cas, dans une base de cas, d'un système RàPC s'appellent des cas sources. Chaque cas source est formé du problème et de sa solution (figure 3.9).

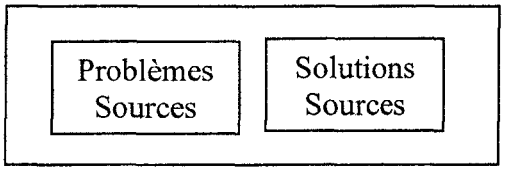


Figure 3.9: Base de cas sources d'un RàPC

Dans l'EMI, les problèmes sources sont représentés par les connaissances métier qui sont capitalisées à l'aide des scénarios patterns. Les solutions sources sont des schémas de conception construits à l'aide du langage UML tels que le digramme de classes. Dans notre approche nous utilisons le terme « *pattern de conception* » pour designer des schémas de conception (figure 3.10).

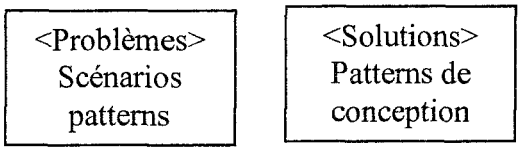


Figure 3.10: Base de cas sources de l'EMI

3.3.2.2 L'expert métier et l'expert de conception

Nous avons déjà montré le rôle de l'expert métier dans la rédaction des scénarios patterns. L'expert métier est un acteur important dans la philosophie de notre nouvelle approche. Il joue le rôle d'un examinateur des connaissances métier dont le but est de préparer les connaissances sous forme de scénarios patterns réutilisables par le modèle des besoins futurs du client. Les scénarios patterns sont ensuite sauvegardés dans une bibliothèque et classifiés en domaine, sous domaine et cas d'utilisation.

L'expert de conception, dans l'EMI, complète la bibliothèque des scénarios patterns par des patterns de solution de conception afin d'associer à chaque scénario pattern un pattern de conception convenable.

3.3.2.3 Les patterns de conception

Malgré les différentes approches proposées pour parvenir à la meilleure adéquation entre les besoins exprimés et la réalisation proposée, le problème d'inadéquation reste toujours en question. Les patterns de conception (traduits par schémas ou patrons) proposent des familles d'architecture de conception génériques fréquemment rencontrées pendant la conception et la programmation objet.

Les patterns de conception peuvent donc être vus comme un moyen élaboré de réutiliser des connaissances acquises par l'expérience. Les problèmes frames [Jackson

01] (que nous traduirons par schémas de problèmes) sont proposés pour structurer les problèmes de manière globale.

Les styles d'architecture [Garlan et Shaw 93] [Bass et *al.* 98] offrent des structures de granularité plus fine qui sont souvent utilisables au niveau de la conception.
La conception d'un logiciel orienté objet suppose de nombreux choix aux nombreuses conséquences. Dans cette tâche délicate, des modèles de conception réutilisables fournissent une aide précieuse : les design patterns font bénéficier leur utilisateur de l'expérience de ceux qui, avant lui, ont réfléchi à la meilleure exploitation possible des avantages du concept orienté objet.

Un élément central de cette conception est le diagramme de classes, modèle représentant la structure statique du système à concevoir. Par ailleurs, ce diagramme peut être enrichi au moyen de design patterns ; dans cet objectif, le concepteur peut avoir recours à des outils d'aide à la conception : ces derniers disposent d'un catalogue de diagrammes de classes correspondant aux design patterns et peuvent en enrichir un diagramme particulier.
En 1977, Christopher Alexander a inventé l'idée de base des patrons, il considère que « Chaque patron décrit un problème qui se manifeste constamment dans notre environnement, et donc décrit le cœur de la solution à ce problème, d'une façon telle que l'on puisse réutiliser cette solution des millions de fois, sans jamais le faire deux fois de la même manière ».

Les travaux les plus célèbres des designs patterns sont ceux introduits par quatre informaticiens et qui sont reconnus mondialement sous le nom de Gang of Four. Ils mirent au point et documentèrent vingt-trois modèles permettant de résoudre vingt-trois problèmes classiques de conception.
Bien d'autres design patterns furent créés depuis, mais ces vingt-trois-là restent les plus connus et les plus utilisés (parfois inconsciemment) par les concepteurs de programmes orientés objet [GoF 94].

Une description non formelle mais détaillée des designs patterns est présentée dans le livre de [GoF 99]. Plus précisément, un design pattern est composé de 4 éléments : le nom de modèle, le problème, la solution, les conséquences.

1. Le nom de modèle est un moyen de décrire en un ou deux mots un problème de conception, ses solutions, et leurs conséquences. Donner un nom à un modèle accroît immédiatement le vocabulaire de conception. Cela permet de travailler à un degré d'abstraction plus élevé.

2. Le problème décrit les situations où le modèle s'applique. Il expose le sujet à traiter et son contexte. Il peut s'agir de problèmes spécifiques de conception comme, par exemple, la représentation d'algorithmes comme des objets. Il peut décrire des structures de classes ou d'objets, typiques d'une conception immuable. Le problème

comportera parfois une liste de conditions à satisfaire pour que le modèle s'applique valablement.

3. La solution décrit les éléments qui constituent la conception, les relations entre eux, leur part dans la solution, leur coopération. La solution ne décrit pas un modèle précis, ni une implémentation, puisqu'un modèle est une sorte de patron qui s'applique dans diverses situations. Le modèle fournit plutôt la description générique d'un problème de conception, et indique comment un agencement d'éléments (dans notre cas, des classes et des objets) peut le résoudre.

4. Les conséquences sont les effets résultant de la mise en œuvre du modèle et les variantes de compromis que celle-ci entraîne. Bien que ces conséquences soient rarement évoquées lors de la description des choix de conception, elles sont déterminantes pour l'évaluation des alternatives de conception et pour l'appréciation des avantages et des inconvénients de l'application du modèle.

Etant donné que la réutilisation est souvent le facteur déterminant d'une conception orientée objet, les conséquences d'un modèle incluent son impact sur la flexibilité d'un système, son extensibilité, ou sa portabilité. Faire la liste explicite des conséquences permet de les comprendre et de les évaluer" [GoF 99].

Un modèle de conception est un terme assez générique et pourrait couvrir bien des cas. Dans notre contexte, nous nous attacherons à des modèles de conception qui "font la description d'objets coopératifs et de classes que l'on a spécialisés pour résoudre un problème général de conception dans un contexte particulier" [GoF 99].

Dans notre approche, les patterns de conception sont des éléments importants de la base de cas de l'EMI. Les patterns de conception représentent les solutions de conception des scénarios patterns. Le scénario pattern et son pattern de conception forment le cas source de la base de cas (base de patterns). Cette base de patterns est similaire à celle utilisée dans les systèmes RàPC. Un pattern de conception, dans l'approche EMI, représente un ou plusieurs scénarios patterns (figure 3.11). Une instance de la classe pattern de conception est une instance du méta-modèle du pattern de conception (figure 3.12).

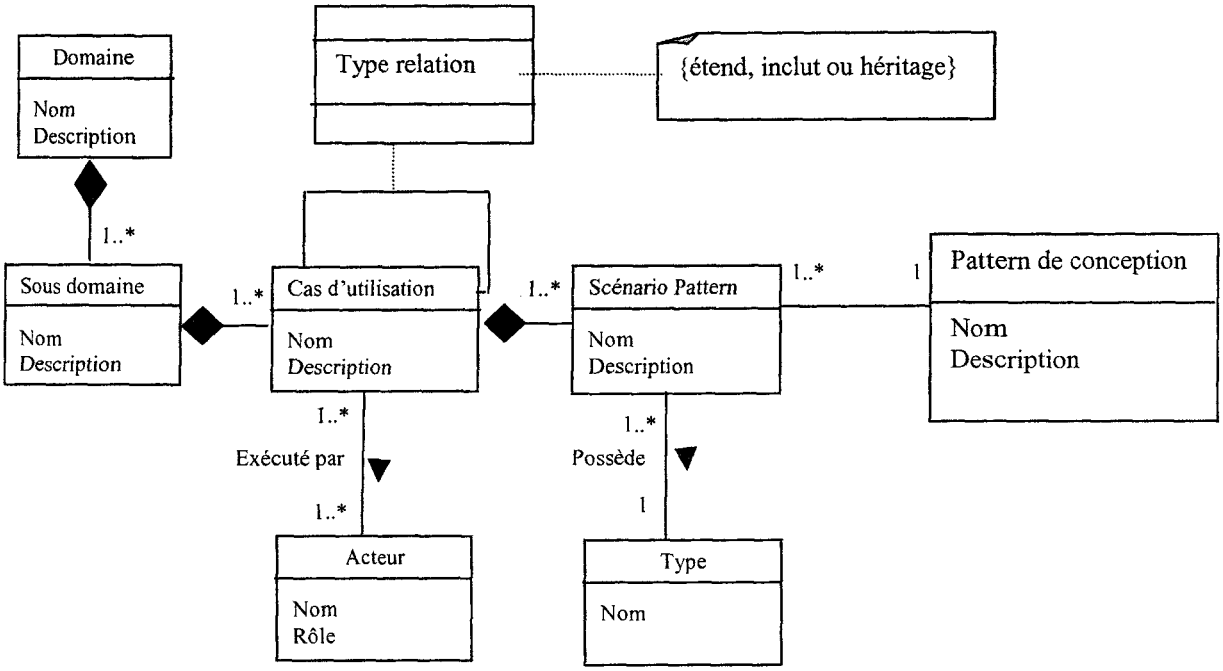


Figure 3.11: Scénario pattern et pattern de conception

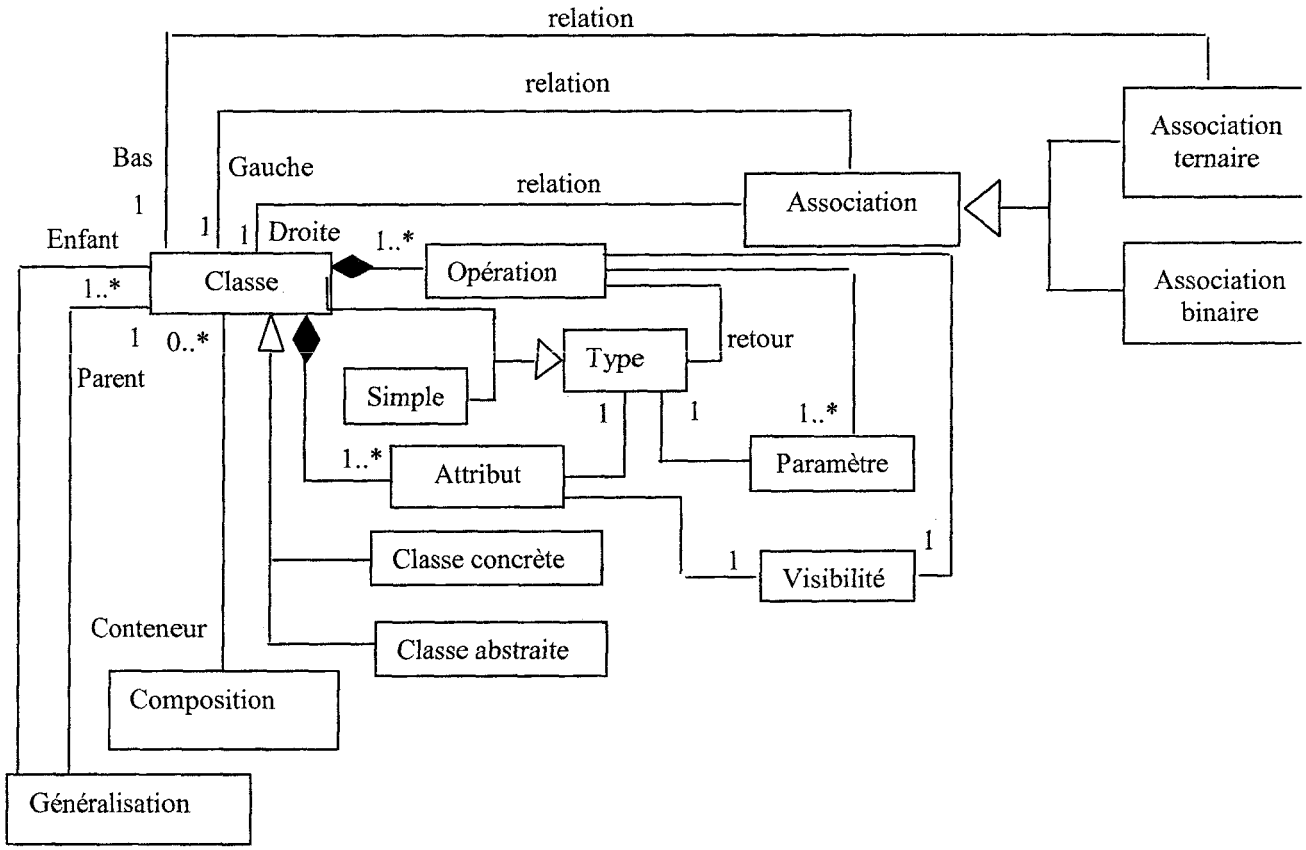


Figure 3.12: Méta-modèle du pattern de conception

Le méta-modèle du pattern de conception

Un pattern de conception est un modèle de diagramme de classes relatif à un contexte particulier qui est une instance du méta-modèle (figure 3.12).
Le méta-modèle d'un pattern de conception contient les éléments suivants : les classes, les associations, les compositions, les généralisations, les types de classes (abstrait ou concret), les opérations et les multiplicités. Le méta-modèle de la figure 3.12 décrit tous les concepts fondamentaux des éléments d'un diagramme de classes. Une instance du méta-modèle est un modèle de diagramme de classes qui est associé à un scénario pattern particulier ainsi qu'à un cas d'utilisation d'un sous domaine.

Exemple : Le scénario pattern principal du cas d'utilisation « retrait d'argent »

/Scénario Pattern « Retrait d'argent »
BEGIN
Retrait d'argent ;
Principal ;
Client ;
/SP
GAB, OpD, 1, insérer ; boolean
Carte, OpD, 2, valider, numéroCarte, dateExpiration ; boolean
interface, OpD, 3, saisir ; motDePasse
Carte, OpD, 4, vérifier, motDePasse ; balance, boolean
interface, OpP, 1, sélectionner ; option
interface, OpP, 2, sélectionner ; monnaie
interface, OpP, 3, sélectionner ; montant
Carte, OpP, 4, autoriser, numéroCarte, solde ; boolean
GAB, OpP, 5, rejeter ; boolean
GAB, OpP, 6, rejeter ; boolean
GAB, OpR, 1, récupérer, boolean
GAB, OpR, 2, récupérer, boolean
GAB, OpR, 3, réinitialiser, boolean
SP/
END

La table 3.1 montre les objets et leurs caractéristiques statiques (attributs) et de comportement (Opérations). L'objet carte de crédit est une instance de la classe « Carte de Crédit », l'objet « GAB » est une instance de la classe GAB (figure 3.13). L'objet particulier « interface » est une instance de la classe de contrôle et il ne fait pas partie du pattern de conception.

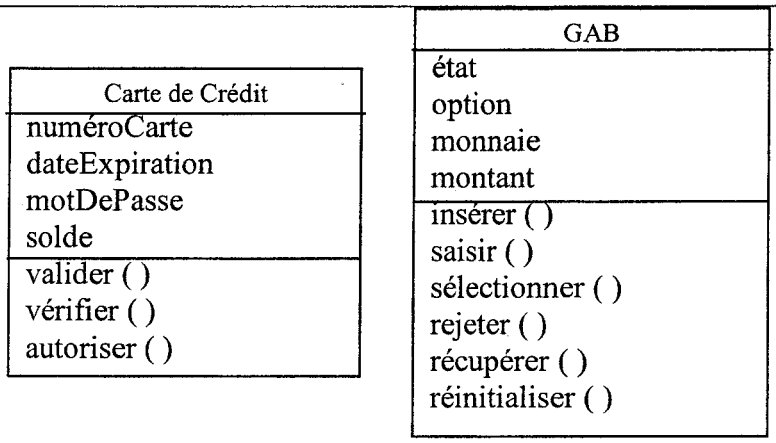


Figure 3.13: Classes instanciées du méta-modèle

Exemple de pattern de conception en forme de diagramme de classes (figure 3.14) :

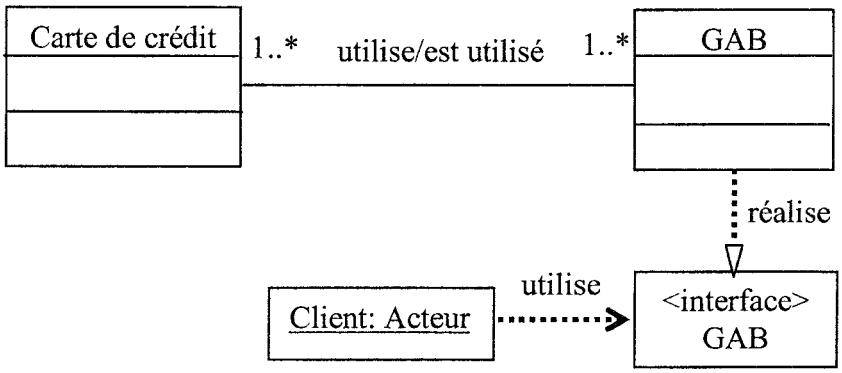


Figure 3.14: Pattern de conception

Avec les patterns de conception la base de cas de L'EMI est complétée. Cette base de cas est le noyau de notre approche qui simule le fonctionnement d'un système RÀPC. Dans les paragraphes suivants nous allons présenter les techniques utilisées pour exploiter les scénarios patterns et les patterns de conception dans le développement du modèle d'analyse d'une nouvelle application informatique.

3.3.3 Processus de développement des logiciels dans l'EMI

La démarche du processus de développement des logiciels dans l'EMI démarre par la description du modèle des besoins du client. Dans cette étape il faut décrire correctement et sans aucune ambiguïté le problème à résoudre afin de permettre l'identification d'un cas similaire pertinent dans la base de cas.

Il existe de nombreux travaux sur la représentation des cas [Sehaba et Estraillier 05] [Elorriaga et Fernandez-castro 00] [Fox et Leake 94] [Kolodner 93], mais la représentation la plus communément reprise est la représentation structurée en liste de descripteurs qui peuvent être des objets complexes. La représentation orientée objet permet de manipuler les connaissances à caractère complexe à l'aide de différentes formes de mise en relation des classes représentant les concepts du monde [Ruet 02].

Cette représentation permet d'élaborer aisément une structure hiérarchique de descripteurs que l'on pourra exploiter dans la phase d'adaptation. La plupart des applications industrielles du RàPC choisissent de décrire les cas comme des formulaires remplis. Les informations sont saisies quand un problème se pose et sont complétées et corrigées au fur et à mesure que le problème est résolu.

Dans l'EMI, la description du problème à résoudre fournit un modèle des besoins du client. Ce modèle est représenté par un diagramme de cas d'utilisation. Le diagramme de cas d'utilisation est décrit à l'aide des formulaires structurés contenant plusieurs descripteurs suffisants pour construire le modèle des besoins du client.
Les descripteurs du modèle des besoins cible tels qu'ils sont définis dans les deux formulaires sont les suivants :

Nom du descripteur	Observation
Domaine	Le domaine de l'application dans lequel le système futur sera établi.
Sous domaine	Un sous domaine relatif au domaine de l'application.
Cas d'utilisation	Activité ou service demandé par le client relatif au sous domaine de l'application proposée.
Pré-condition	Une condition de démarrage du cas d'utilisation.
Post-condition	Une condition d'arrêt du cas d'utilisation pour montrer que le cas à atteindre son objectif.
Acteurs	Personnes utilisateurs ou objets physiques qui échangent de messages avec le cas d'utilisation.
Cas d'utilisation en relation	Les cas d'utilisation qui dépendent des autres cas.
Type	Nature de dépendances entre les cas d'utilisation.
Contraintes	Contraintes sur les dépendances entre les cas d'utilisation.

3.3.3.1 Le modèle des besoins du client

Le modèle des besoins, par la définition d'un diagramme de cas d'utilisation, est un moyen élaboré pour décrire les besoins futurs d'un système et les services à fournir après sa réalisation. Le diagramme de cas d'utilisation définit les limites du système futur selon les désirs du client.

Dans notre approche, le client doit remplir des formulaires de recueil des besoins fonctionnels pour construire son pré-modèle des besoins au moyen d'un diagramme de cas d'utilisation. Les formulaires se présentent comme des fiches détaillées informelles des besoins demandés par le client et ses attentes du système à réaliser (voir table 3.2 ; table 3.3) [Hussein et *al.* 08]. Afin de remplir correctement les formulaires, le client doit connaître suffisamment le domaine du travail, posséder une bonne méthode de communication, capable de transmettre l'information et être conscient de l'avantage que la technologie informatique peut porter sur les systèmes d'information.

Roques et Vallée [Roques et Vallée 03] utilisent plusieurs fiches pour décrire les cas d'utilisation d'un système. Pour eux, chaque cas est documenté via une fiche de documentation de cas d'utilisation qui indique les acteurs qui interagissent avec le cas étudié. Une synthèse des cas apparaîtra dans la fiche de synthèse des cas d'utilisation. Ces cas d'utilisation sont organisés via la fiche de structuration des cas d'utilisation. Chacun des packages, qui regroupent les cas d'utilisation par thèmes, devra être illustré par un diagramme de cas d'utilisation. Sur cette fiche apparaîtra le nom du cas, son but, son résumé et les acteurs qui interagissent avec ce cas, les pré-conditions, enchaînements nominaux et alternatifs et exceptions. Contrairement aux fiches utilisées par Roques et Vallée qui détaillent les cas d'utilisation pour construire le modèle des besoins, notre approche évite cette description détaillée textuelle et propose une description au niveau abstrait ; seulement le nom du cas avec ses caractéristiques sera représenté.

Formulaire descriptif général

Domaine d'application					
Sous domaine	Cas d'utilisation	Pré-condition	Post-condition	Acteurs	
Sous domaine	Cas d'utilisation	Pré-condition	Post-condition	Acteurs	

Table 3.2: Formulaire descriptif général

Le domaine d'application est le contexte métier demandé par le client. Par exemple, le domaine bancaire, industriel, médical, importation et exportation, commerce électronique, etc.

Le domaine d'application, selon notre approche, est divisé en un ensemble de sous domaines chacun représentant un ensemble des activités. Par exemple, les sous domaines transactions bancaires, les machines GAB et les bons de trésors appartiennent au domaine bancaire.

Les cas d'utilisation représentent les objectifs majeurs du système à modéliser. Il ne s'agit en aucun cas de tâches (granularité trop basse) mais bien de services rendus par le système, voire un comportement attendu par le système. Ces cas d'utilisation doivent refléter des sémantiques de connaissances métier du domaine équivalent.

Les pré-conditions et les post-conditions représentent un contrat de vérification des cas d'utilisation. Ce contrat permet de vérifier si le cas d'utilisation bien fourni les résultats demandés. En général, les pré-conditions et les post-conditions pendant la description textuelle du cas d'utilisation peuvent être informelles ou formelles. Par exemple, le cas d'utilisation « Retrait d'argent » possède la pré-condition « le client possède un compte », « montant<=solde » et les post-conditions « le guichetier ferme le compte » et « le client récupère l'argent ».
Dans cet exemple, la pré-condition et les post-conditions sont toutes les deux informelles. Un autre exemple, le cas d'utilisation « acheter un produit en ligne », la pré-condition est « l'abonné est identifié » et les post-conditions sont « le système à enregistrer les informations, concernant la transaction d'achat, dans la base de données » et «le système à enregistrer l'autorisation de payement en ligne » [Charroux, Osmani et Thierry-mieg 05].

En effet, les pré-conditions et les post-conditions représentent des opérations précises de déclenchement et d'arrêt du cas d'utilisation. Si ces opérations renvoient des valeurs booléennes vraies, ceci signifie que le cas d'utilisation est bien réussi.
Dans notre approche, lorsque l'ingénieur système saisit les formulaires du client, les pré-conditions et les post-conditions seront transformées en des opérations précises avec leur description informelle telle qu'elle est décrite par le client.
Par exemple, la pré-condition « l'abonné est identifié » représente l'opération déclencheur identifier () avec un type de retour booléen. La post-condition « le système à enregistrer les informations, concernant la transaction d'achat, dans la base de données » est transformée en l'opération enregistrer ( ) avec un type de retour booléen.
Les opérations qui caractérisent les pré-conditions et les post-conditions seront obligatoirement testées par l'alternatif « Si » pour s'assurer si le cas d'utilisation a atteint son objectif ou non.

Les acteurs représentent les différentes personnes qui interagissent avec le système en cours de modélisation. A une même personne peuvent être associés plusieurs acteurs selon les rôles fonctionnels qu'ils jouent sur le système. Un acteur peut cependant être

une machine extérieure au système mais qui interagit avec ce dernier. De la même manière que pour une personne, à une entité peuvent être associés plusieurs acteurs d'un système.

Dans notre approche, et lors du remplissage des formulaires, il faut prendre en considération les rôles joués par les acteurs. Ainsi, les acteurs sont forcément extérieurs au système.

Formulaire des contraintes entre les cas d'utilisation

Sous domaine	Cas d'utilisation	Cas d'utilisation en relation	Type	Contrainte

Table 3.3: Formulaire des contraintes entre les cas d'utilisation

Dans la table 3.3, les dépendances entre les cas d'utilisation sont retirées du langage UML. Il existe trois types de relations entre les cas d'utilisation « inclut », « étend » et « héritage ». La relation « inclut » signifie qu'un cas d'utilisation A peut inclure un cas d'utilisation B. La relation « étend » signifie qu'un cas d'utilisation A peut avoir ou non un cas d'utilisation d'extension (dans certains cas, l'extension est conditionnelle, il faut valider la contrainte avant d'étendre le cas d'utilisation). La relation d'héritage signifie qu'un cas d'utilisation B peut hériter les caractéristiques d'un cas d'utilisation parent A (éventuellement, des scénarios) (figure 3.15).

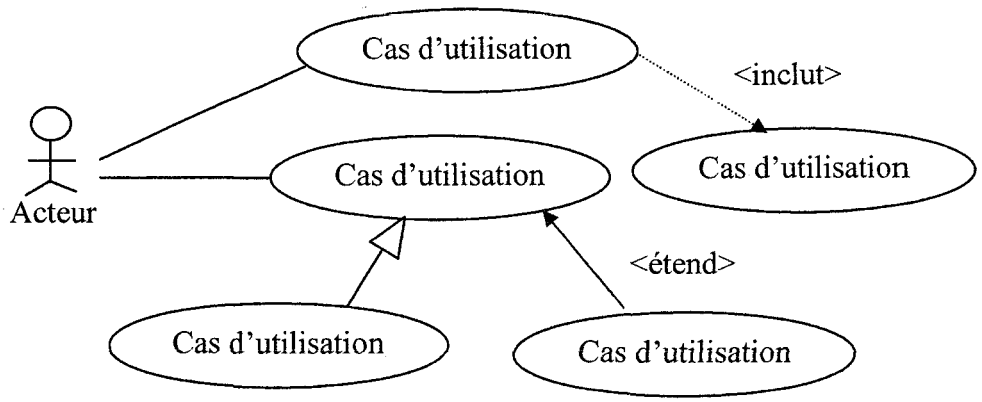


Figure 3.15: Diagramme de cas d'utilisation avec les 3 types de relations possibles

Exemple descriptif du pré-modèle des besoins:

Domaine de L'application	Bancaire				
Sous domaine	**Cas d'utilisation**	**Pré-condition**	**Post-condition**	**Acteurs**	
Machine GAB	Retrait d'argent	Client identifié	Le client récupère l'argent. Le système enregistre la transaction.	Client	
	Consulter solde	Client identifié	Néant	Client	
	Initialiser GAB	Néant	GAB est dans l'état prêt	Administrateur	
	Arrêter GAB	Pas d'opérations en cours	Message d'arrêt bien affiché	Administrateur	
	Identification	Possède une carte de crédit	Client identifié	Client	
Sous domaine	**Cas d'utilisation**			**Acteurs**	
Transactions bancaires	Dépôt d'argent	Opérateur identifié	L'opérateur prend l'argent. Le système enregistre la transaction.	Opérateur	
	Retrait d'argent	Opérateur identifié	L'opérateur donne l'argent au client. Le système enregistre la transaction.	Opérateur	
	Consulter compte	Opérateur identifié	Néant	Opérateur	
	Consulter compte en ligne	Client distant identifié	Néant	Client distant	
	Change de monnaie	Opérateur identifié	Le système enregistre la transaction de change.	Opérateur	
	Transfert de compte	Opérateur identifié	Le système enregistre la transaction de transfert.	Opérateur	
	Identification	Néant	Opérateur identifié	Opérateur	

Table 3.4: Formulaire des besoins

Sous domaine	Cas d'utilisation	Cas d'utilisation en relation	Type	Contrainte
Machine GAB	Retrait d'argent	Identification	Inclut	
	Retrait d'argent	Identification	Inclut	
	Initialiser GAB	Identification	Inclut	
	Arrêter GAB	Identification	Inclut	
Transactions bancaires	Dépôt d'argent	Identification	Inclut	
	Retrait d'argent	Identification	Inclut	
	Consulter compte	Dépôt argent	Etend	
	Consulter compte	Retrait d'argent	Etend	Si compte est solvable
	Consulter compte	Identification	Inclut	
	Consulter compte en ligne	Consulter compte	Héritage	
	Change de monnaie	Identification	Inclut	
	Transfert de compte	Identification	Inclut	

Table 3.5: Formulaire des contraintes

L'ingénieur système révise les formulaires, avant de les saisir dans l'EMI, et il peut introduire des modifications pour avoir des formulaires plus lisibles et ne contenant pas d'informations contradictoires.

Une fois les formulaires sont préparés, l'ingénieur système commence à les saisir dans l'environnement EMI pour en construire un modèle des besoins sous forme de diagramme de cas d'utilisation. Nous rappelons que les pré-conditions et les post-conditions sont transformées en des opérations significatives et abstraites. Dans l'exemple précédent, les opérations qui représentent les pré-conditions et les post-conditions sont : identifier, état, insérer, récupérer, enregistrer, et afficher. Après la fin de la saisie des formulaires, l'ingénieur système sauvegarde le modèle de besoins du client dans une structure de base intégrée dans l'EMI (figure 3.16).

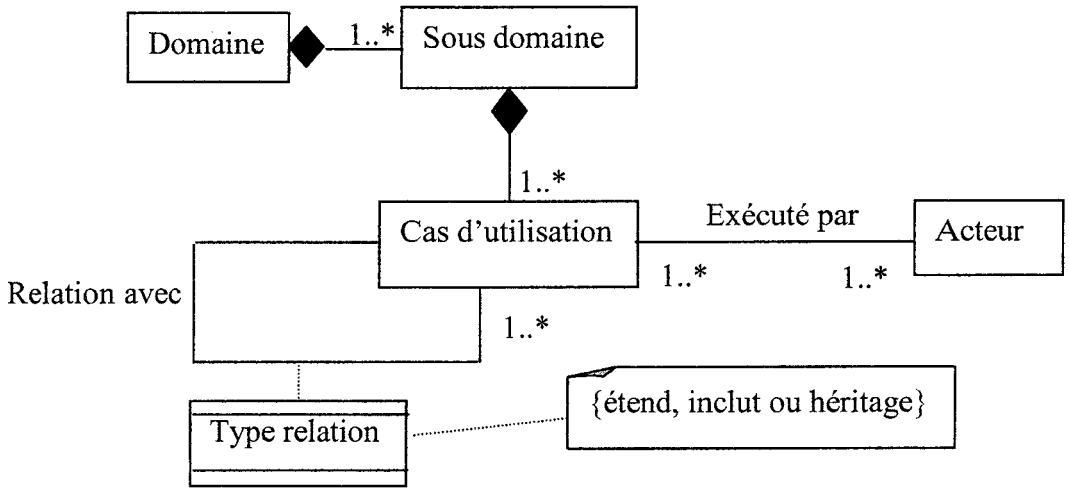


Figure 3.16: Structure de base du modèle des besoins

Description de cette structure :

Classe	Attributs	Opérations
Domaine	String nom, String description, (Sous domaine) Vector sousdomaines	Domaine(), String Lirenom(), String Liredescription() Sous domaine [] liresousdomaine()
Sous domaine	String nom, String description, (Domaine) domaine	SousDomaine(), String Lirenom(), String Liredescription(), Domaine liredomaine()
Acteur	String nom String description (Cas d'utilisation) Vector usecase;	Acteur(), String Lirenom(), String Liredescription(), Cas d'utilisation [] lireusecase
Type	String nom, String description	Type() Lirenom() Liredescription()
Cas d'utilisation	String nom, String description, SousDomaine sousdomaine, String [][] precondition, String [][] postcondition, Cas d'utilisation [] caslié, String [ ] typerelation, (Acteur) Vector acteurs	Cas d'utilisation(), String Lirenom(), String Liredescription(), SousDomaine liresousdomaine(), String lireprecondition(), String lirepostcondition(), Cas d'utilisation lirecaslié(), String liretyperelation(), Acteur [ ] lireacteurs()

Les formulaires représentent des structures abstraites du modèle des besoins. L'ingénieur système et le client doivent compléter le modèle en détaillant les cas d'utilisation par un ensemble de scénarios réutilisables (scénarios patterns).
Pour mener à bien cet objectif, l'ingénieur système présente au client un ensemble de scénarios patterns extraits de la base de cas source de l'EMI et relatif au modèle de cas d'utilisation demandé par le client; le client assiste au choix des scénarios et il peut demander des modifications selon ses désirs pour en construire le modèle des besoins final et complet.

L'extraction des scénarios patterns depuis la base de l'EMI est l'étape la plus délicate, c'est dans cette étape où il faut chercher les cas d'utilisation sources similaires aux cas d'utilisation cibles. Dans les systèmes RàPC, cette étape s'appelle l'étape de remémoration : il s'agit de retrouver un cas similaire dans la base de cas source à partir d'un cas cible.

3.3.3.2 Principe de test de similarité

La remémoration des cas sources similaire au cas cible est une étape qui intègre deux processus, appariement et recherche. Le processus d'appariement utilise une fonction qui calcule le degré de similarité entre les cas. Le processus de recherche consiste à élaborer des méthodes d'investigation optimales dans la base de cas qui tiennent compte de sa structure et de ses propriétés.

L'appariement est défini dans [Kolodner 93] par "Matching is the process of comparing two cases to each other and determining their degree of match". La méthode d'appariement que nous avons utilisée est celle du « plus proche voisin » « (k-nearestneighbors) ».
Cette méthode, utilisée dans [Elorriaga et Fernandez-castro 00] [VoB 94] [Leake 96], possède l'avantage d'être "très simple à implémenter" [Bisson 00] et le fait qu'elle utilise directement la notion de similarité pour mesurer la correspondance entre chaque cas source et cas cible. La similarité est mesurée en termes d'attributs appropriés des descripteurs de cas (cible ou source). Rappelons, que dans notre approche, les descripteurs d'un cas concernent l'ensemble des descripteurs du modèle des besoins du client construit à partir des formulaires de recueil des besoins. Ces descripteurs sont partagés entre les différents cas d'utilisation du modèle des besoins du client. Donc, chaque cas d'utilisation du modèle des besoins du client représente un cas cible dont nous allons extraire les cas sources similaires.

La méthode du « plus proche voisin » distingue deux filtres dans le processus d'appariement : le premier filtre sélectionne les cas sources dont la similarité avec le cas cible est supérieure à un seuil déterminé ; le deuxième filtre sélectionne les cas sources qui minimisent l'effort d'adaptation. Dans notre approche, ce seuil est une distance acceptable entre le cas cible et le cas source.
Le premier filtre utilise la fonction numérique ϕ qui calcule le degré de similarité entre un cas cible C et un cas source S. La fonction ϕ est définie comme étant la moyenne pondérée des valeurs de similarité sur chacun des descripteurs de C et S.
- D est l'ensemble des descripteurs du cas cible C.
- W_d est le coefficient d'importance du descripteur *d*. Les descripteurs les plus importants sont pris en considération dans le calcul de la fonction de similarité que d'autres descripteurs moins importants.
- φ_d (C,S) est la similarité entre deux valeurs du descripteur *d* associées à C et S ; c'est une fonction logique qui renvoie 1 si le descripteur de C est similaire au descripteur du S et renvoie 0 dans le cas contraire.

$$\phi(C, S) = \frac{\sum_{d \in D} W_d * \varphi_d(C,S)}{\sum_{d \in D} W_d}$$

Le deuxième filtre minimise l'effort d'adaptation des cas sources à la situation actuelle du cas cible. L'effort d'adaptation est calculé en fonction des buts et des descripteurs du cas cible non assurés par le cas source. Plus ces données sont importantes, plus l'effort d'adaptation est important, et moins la sélection du cas source est retenue.

Principe de détermination du seuil

Dans notre approche, pour chaque cas d'utilisation, le seuil est calculé dynamiquement en fonction du nombre de descripteurs et du degré d'importance donné à chaque descripteur (appelé poids).
Le nom du cas d'utilisation ou son synonyme possède le degré d'importance le plus élevé. Ensuite, nous donnons aux descripteurs « post-conditions », « pré-conditions », « noms d'acteurs », « noms des cas d'utilisation dépendants et leurs types de relation », respectivement des poids décroissants (Table 3.6) [Hussein et *al.* 08].

Descripteur	Poids
Nom du cas d'utilisation	m
Post-conditions	m-1
Pré-conditions	m-2
Noms acteurs	m-3
Noms cas dépendants	m-4
Types cas dépendants	m-5
Contrainte	m-6
Nom du sous domaine	m-7
Nom du domaine	m-8

Table 3.6: Tableau de poids des descripteurs

La variable m est un entier strictement positif avec m>=k où k est le nombre total des descripteurs. Alors, le seuil, pour un cas d'utilisation, est déterminé en fonction de descripteurs et leurs poids.

- S_c le seuil d'un cas d'utilisation cible (C),
- d_i est un descripteur,
- poids est le poids du descripteur,
- n est le nombre de descripteurs évalués,
- k est le nombre total des descripteurs dans le contexte de développement.

$$S_c = \sum_{i=1}^{n} \text{poids}(d_i) / \sum_{i=1}^{k} \text{poids}(d_i)$$

Pour n=k, cas où tous les descripteurs sont évalués, le seuil est maximal: $S_c = 1$.
Le seuil est dit minimal (S_c = m/ (m+m-1+m-2+….+m-k+1)) lorsque le seul descripteur évalué est le nom du cas d'utilisation. L'intervalle accepté de seuils est compris entre m/(m+m-1+m-2+….+m-k+1) et 1. Chaque cas source, dont la fonction ϕ appartient à cet intervalle, sera retenu.
Chaque cas d'utilisation de l'exemple précédent (Table 3.4 et Table 3.5) possède un ensemble de descripteurs. Dans la table 3.8, nous montrons les descripteurs évalués par

chaque cas d'utilisation, et dans la table 3.9 les valeurs récapitulatives des seuils dynamiques calculés en fonction du nombre de descripteurs [Hussein et *al.* 08].

Descripteurs	Change monnaie	Transfert compte	Identification	Retrait	Dépôt	Consulter compte	Consulter compte en ligne
Nom du cas d'utilisation	X	X	X	X	X	X	X
Post-conditions	X	X	X	X	X	-	-
Pré-conditions	X	X	-	X	X	X	X
Noms acteurs	X	X	X	X	X	X	X
Noms cas dépendants	X	X	-	X	X	X	X
Types cas dépendants	X	X	-	X	X	X	X
Contrainte	-	-	-	X	-	-	-
Nom du sous domaine	X	X	X	X	X	X	X
Nom du domaine	X	X	X	X	X	X	X

Table 3.7: Tableau de descripteurs des cas d'utilisation

Soit m=9, les seuils calculés seront :

Descripteurs	Change monnaie	Transfert compte	Identification	Retrait	Dépôt	Consulter compte	Consulter compte en ligne
Nom du cas d'utilisation	9	9	9	9	9	9	9
Post-conditions	8	8	8	8	8	-	-
Pré-conditions	7	7	-	7	7	7	7
Noms acteurs	6	6	6	6	6	6	6
Noms cas dépendants	5	5	-	5	5	5	5
Types cas dépendants	4	4	-	4	4	4	4
Contrainte	-	-	-	3	-	-	-
Nom du sous domaine	2	2	2	2	2	2	2
Nom du domaine	1	1	1	1	1	1	1
Seuil	0.94	0.94	0.58	1	0.94	0.76	0.76

Table 3.8: Seuils dynamiques calculés en fonction du nombre de descripteurs

Distance acceptable à un seuil d'un cas source

Pour chaque cas cible, nous procédons à la recherche des cas sources les plus proches afin de construire un modèle détaillé des besoins client, c'est-à-dire un diagramme de cas d'utilisation enrichi par un ensemble de scénarios patterns qui modélise le système futur désiré par le client. Pour un cas cible nous calculons la fonction de similarité à un cas source ; cette fonction est une distance entre les deux cas. Soit D cette distance, D est acceptable lorsqu'elle est plus grande au minimum du seuil dynamique. Notons que le seuil dynamique, pour chaque cas, est calculé en fonction du nombre de descripteurs du cas. Notre fonction $\phi(C, S)$ est calculée en fonction du nombre de descripteurs évalués. La fonction φ_d (C,S) est égale à 1 si le deux descripteurs (cible et source) sont similaires et elle est égale à 0 dans le cas contraire.

Par exemple, le seuil dynamique du cas d'utilisation « Retrait » est égal à 1, c'est un seuil maximal car il est calculé en fonction de tous ses descripteurs. Ainsi, le seuil minimal du même cas d'utilisation est 0.2, c'est le cas où le seul descripteur évalué est le nom du cas d'utilisation.

La distance D entre deux cas d'utilisation (source et cible) est représentée par la formule : D = S_c (cas cible) - $\phi(C, S)$

La distance D est dite acceptable, c'est-à-dire le cas d'utilisation source est retenue comme un cas d'utilisation similaire, lorsque D >=0 et D<=S_c-Min(S_c) (cas cible). Si on calcule la fonction $\phi(C, S)$ pour le cas d'utilisation « Retrait » et on considère que le cas d'utilisation source possède les mêmes descripteurs que le cas d'utilisation cible alors la fonction $\phi(C, S)$ sera égale à 1. En effet, la valeur de la fonction $\phi(C, S)$ dépend de descripteurs du cas source tel qu'il est décrit dans la base de cas par l'expert de conception.

Un deuxième exemple présente un digramme de cas d'utilisation modélisant un site Web « Free Song » qui permet à des visiteurs de s'inscrire afin de télécharger, gratuitement, des chansons (figure 3.17). Le visiteur peut s'inscrire par le biais du cas d'utilisation « Inscription ». L'abonné est un visiteur déjà inscrit, il peut télécharger des chansons à l'aide du cas d'utilisation « Téléchargement » après une identification par le biais du cas d'utilisation « Identification ». Chaque cas d'utilisation du diagramme possède un ensemble de descripteurs (Table 3.9).

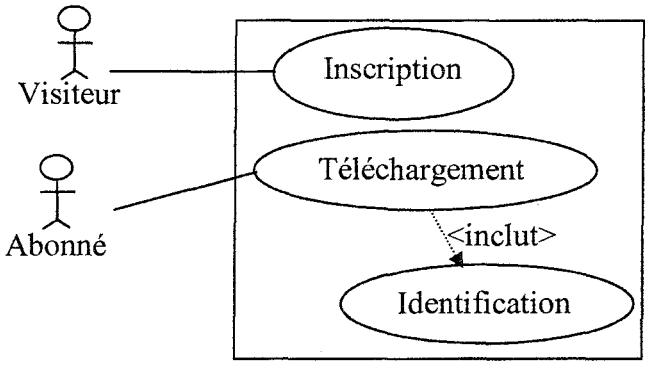


Figure 3.17: diagramme de cas d'utilisation du site Web "Free Song"

Descripteurs	Inscription	Téléchargement	Identification
Nom du cas d'utilisation	Inscription	Téléchargement	Identification
Post-conditions	Visiteur est inscrit dans le web	La chanson est téléchargée	Abonné est identifié
Pré-conditions	Néant	Abonné est identifié	Néant
Noms acteurs	Visiteur	Abonné	Abonné
Noms cas dépendants	Néant	Identification	Néant
Types cas dépendants	Néant	Inclut	Néant
Contrainte	Néant	Néant	Néant
Nom du sous domaine	téléchargement des chansons gratuites	téléchargement des chansons gratuites	téléchargement des chansons gratuites
Nom du domaine	Application Web	Application Web	Application Web

Table 3.9: Tableau de descripteurs des cas d'utilisation

Le choix des seuils dynamiques, pour chaque cas d'utilisation en fonction du nombre de descripteurs, est donné dans la table 3.10 avec m=k=9.

Descripteurs	Inscription	Téléchargement	Identification
Nom du cas d'utilisation	9	9	9
Post-conditions	8	8	8
Pré-conditions	-	7	-
Noms acteurs	6	6	6
Noms cas dépendants	-	5	-
Types cas dépendants	-	4	-
Contrainte	-	-	-
Nom du sous domaine	2	2	2
Nom du domaine	1	1	1
Seuil	**0.58**	**0.94**	**0.58**

Table 3.10: Seuils dynamiques calculés en fonction du nombre des descripteurs

Principe de la recherche des cas similaires

La recherche dans la base de cas dépend fortement de sa structure. La structure que nous avons adoptée dans notre approche c'est celle vue dans la figure 3.11. Les cas ayant des similarités propres sont organisés sous la forme d'une structure plus générale qui regroupe les données communes afin de faciliter l'indexation et la recherche. Cette organisation est semblable à celle utilisée par le modèle de Schank [Schank 82].

L'idée fondamentale du modèle est d'organiser les différents cas sous d'épisodes généralisés ou EG. Un EG de Schanck contient trois types d'objets :

• Norme (Norm) : elle représente les données communes à tous les cas indexés sous l'EG. Elle est représentée par une liste de couples de la forme «attribut, valeur ».

• Cas (Cases) : ils représentent des expériences individuelles.

• Index (Indices) : ils représentent les différences entre les cas d'un même EG. Chaque index est lié à un attribut concret d'un descripteur et contient une liste de paires « valeur, nœud ». Chaque paire d'index lie son EG avec un autre nœud (cas ou EG) correspondant.

Ainsi se forme un graphe hiérarchique (voir l'exemple de la figure 3.18) dont les nœuds sont soit des EGs soit des cas. Les arcs représentent les liens entre les index et les nœuds.

Dans cette structure hiérarchique, la recherche est guidée par les attributs partagés de chaque EG (normes) et les index. Ce problème de recherche présente quelques propriétés spéciales. D'abord, la structure hiérarchique de la base de cas permet une

recherche heuristique guidée par la fonction de similarité. Cette dernière est limitée aux attributs des descripteurs du cas cible inclus dans la norme de l'EG. Une autre caractéristique de la recherche réside dans l'absence d'un critère d'arrêt : un cas partageant tous les attributs du cas cible est rarement trouvé. Par conséquent, le critère de sélection est basé sur une comparaison entre la valeur de la fonction de similarité et le seuil. Chaque cas source dont la valeur de la fonction de similarité avec le cas cible dépasse le seuil sera retenu comme cas candidat pour la deuxième étape (l'étape d'adaptation) [Sehaba et Estraillier 05]. Ces propriétés nous permettent d'établir des techniques de sélection basées sur des méthodes classiques de recherche. Pour mener à bien le principe de test de similarité nous avons adopté une recherche précédée par un filtrage des cas sources basé sur les normes domaine et sous domaine ; ceci limite le nombre des cas sources à tester et évite par conséquent l'absence d'un critère d'arrêt. La sémantique du modèle des besoins du client, qui est constitué par un ensemble de cas d'utilisation, est prélevée du vocabulaire du métier.

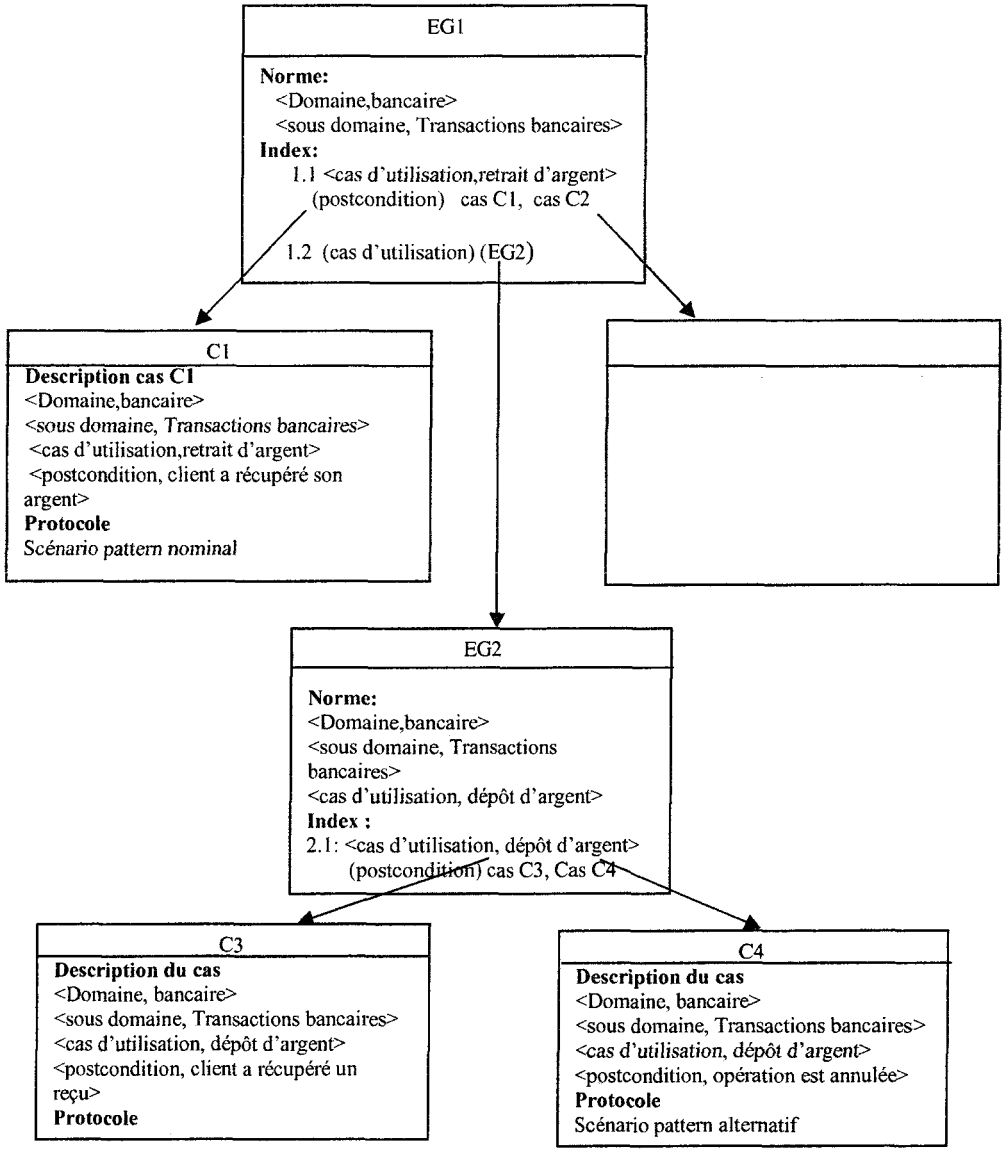


Figure 3.18: Graphe hiérarchique de la structure des cas sources

Un cas d'utilisation possède un nom dans le contexte du métier. Ce nom, qui est un des descripteurs, n'est pas arbitraire, c'est une abstraction sémantique permettant de fournir au cas d'utilisation un moyen lisible de communication entre les gens qui pratiquent le même métier. Le nom du cas d'utilisation peut ne pas être le même dans le même domaine mais il doit posséder la même sémantique. Donc, le dictionnaire de connaissances métier doit contenir le vocabulaire des différentes écoles du métier pour y permettre une transition facile. En effet, dans le même domaine du métier, deux écoles qui utilisent des vocabulaires différents possèdent le même sémantique métier.
Plus le dictionnaire est riche en vocabulaire métier plus l'EMI pourra facilement répondre aux exigences du client et de l'ingénieur système.

Voici un exemple retiré de la banque « Audi » et la banque « SGBL » au Liban. Le cas d'utilisation relatif au processus de retrait d'argent de la machine GAB chez la banque « Audi » est intitulé « Retrait espèce », alors que le même cas est intitulé dans la machine GAB de la banque SGBL « Retrait rapide ». Il est clair que les deux cas ont le même objectif (i.e. la même sémantique), c'est celui du retrait d'argent, mais ils possèdent deux noms différents.

En effet, le but de la phase de remémoration est d'extraire, de la base de cas source, tous les cas similaires au pré-modèle des besoins du client construit à l'étape précédente. Cette phase doit renvoyer aussi l'ensemble de scénarios patterns relatifs aux cas sources retenus afin de construire, pour le client, un modèle des besoins détaillé.
Cette phase est déclenchée par l'appel de l'algorithme de filtrage qui renvoie, en premier lieu, un ensemble de cas correspondants aux normes de la structure EG, dont le but est de diminuer l'ensemble visé de cas sources (voir annexe H). Après la fin de l'exécution de l'algorithme de filtrage, l'algorithme « Renvoyer_cas_similaires » est appelé (voir annexe H). Le but de cet algorithme est de tester la similarité des cas filtrés et de renvoyer ceux qui sont similaires au cas cible (c'est-à-dire au modèle des besoins du client). Le résultat de l'algorithme « Renvoyer_cas_similaires » est une liste des cas d'utilisation similaires au cas cible. Cette liste sera traitée par un troisième algorithme appelé « Renvoyer_cas_plus_similaires » (voir annexe H). Le but de cet algorithme est de renvoyer une liste plus courte contenant seulement les cas les plus similaires au cas cible selon un seuil précisé par l'ingénieur système. Le principe de cet algorithme est de filtrer les cas sources, sélectionnés pour un cas cible, en fonction de leurs fonctions de similarité ; ceux où la fonction de similarité est plus proche du seuil maximum seront retenus.
Lorsque la liste, des cas les plus similaires, est renvoyée par l'algorithme précédent nous procédons à la recherche des scénarios patterns des cas retenus, en appliquant l'algorithme sélectionner_scénarios_patterns (voir annexe H).

3.3.3.3 Sélection des patterns de conception

Après la remémoration, les solutions des cas sources sélectionnés seront prêtes; dans notre approche, ces solutions sont des patterns de conception. Les patterns de conception sont relatifs à des cas d'utilisation cibles. Ainsi, l'EMI doit construire à partir de ces fragments des patterns de conception un pattern de conception global qui représente le

schéma de conception statique du diagramme de cas d'utilisation (modèle des besoins du client). Ce schéma de conception est un diagramme de classes en formalisme UML.

3.3.3.4 Elaboration d'une pré-solution orientée objet

La construction du pattern de conception global (diagramme de classes), relatif à un diagramme de cas d'utilisation, est une composition des fragments de patterns de conception des cas du diagramme de cas d'utilisation. La procédure de composition est itérative, elle permet après chaque itération de fournir un schéma de conception plus riche en classes et associations. Les classes et les associations, elles aussi, seront enrichies par des nouveaux attributs et opérations.

L'exemple ci-dessous présente la façon de construire un pattern de conception par composition itérative. L'idée de cet exemple est tirée de l'article de [Ledang 01] qui traite un système de contrôle d'accès [Ledru, Padiou et Jaray 00].

Système de contrôle d'accès

L'exemple s'agit d'un système de contrôle d'accès à un bâtiment à l'aide des cartes d'accès. Le système doit vérifier la validité de la carte avant de permettre le passage. Nous proposons de traiter l'exemple dans le cas où le système fonctionne en mode normal et non avec le cas où l'incendie peut se produire à n'importe quel moment.

Modèle des besoins

Le modèle des besoins, du système de contrôle d'accès, est représenté à l'aide d'un diagramme de cas d'utilisation qui spécifie la fonctionnalité complète du système [Rumbaugh, Jacobson et Booch 98]. Le comportement d'un cas d'utilisation est souvent décrit par des textes structurés. La forme textuelle est largement admise grâce à sa capacité d'expression et la facilité de la structuration des cas d'utilisation [Cockburn 97] [Coo 99] [Cokburn 00].

Le langage LSP tel qu'il est conçu dans notre approche permet de transformer ces formes textuelles en des scénarios patterns (voir 3.2.2). Dans l'exemple du système de contrôle d'accès, il y a un seul acteur « Utilisateur »; il s'agit de la personne qui souhaite entrer ou sortir d'un bâtiment. Il y a aussi un cas d'utilisation unique « gérer passage » (figure 3.19).

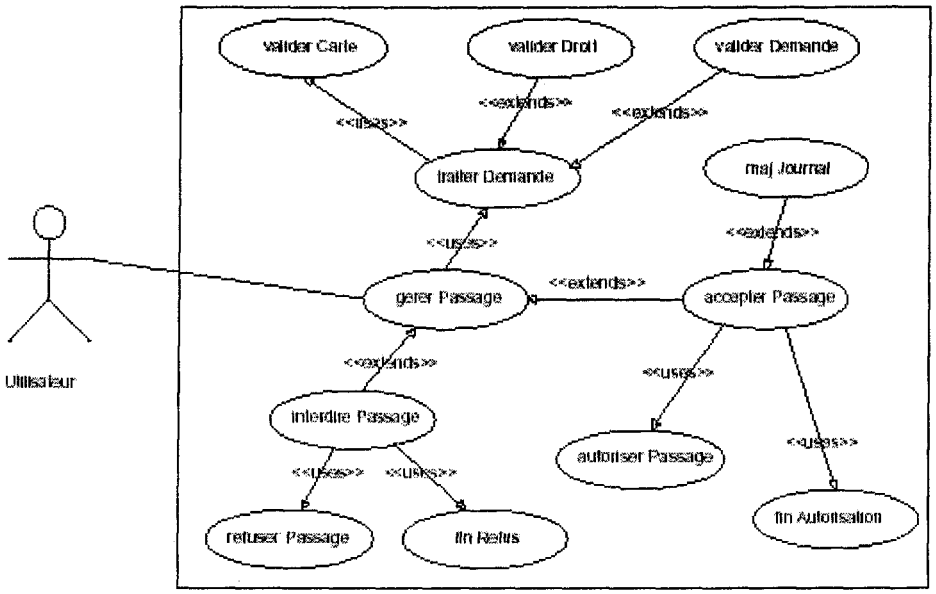


Figure 3.19 – Structuration du cas d'utilisation "gérer passage"

(Gp) correspondant au protocole de gestion de passage. La description de ce cas d'utilisation est une réécriture du protocole de passage décrit dans le cahier des charges du système.

Nom du cas: gérer Passage
Paramètre : Un utilisateur se présente à un lecteur, muni d'une carte d'accès.

Pré-condition :
Le lecteur est libre; ça signifie que le voyant rouge et le voyant vert du lecteur sont éteints; la porte correspondante est dans l'état bloqué.

Le scénario de base :

1. Le système vérifie si l'accès est autorisé.
2. Si l'accès est autorisé, alors
3. L'utilisateur retire sa carte du lecteur.
4. Le système allume le voyant vert.
5. Le système débloque la porte.
6. Si la porte est franchie (l'utilisateur passe par la porte), alors
7. Le système met à jour le journal de passage.
8. Le système bloque la porte.
9. Le système éteint le voyant vert.

Post-condition
Le lecteur en question redevient libre.

Le scénario alternatif 1: l'accès est refusé
Recopier l'étape 1 du scénario de base.
Condition d'occurrence : l'accès est refusé
Action :

1. Le système allume le voyant rouge pendant 2 secondes.
2. L'utilisateur retire sa carte.

Le scénario alternatif 2: la porte n'est pas franchie
Recopier les étapes 1-5 du scénario de base.
Condition d'occurrence : timeout
Action :

1. Le système bloque la porte.
2. Le système éteint le voyant vert.

Description des scénarios textuels par des scénarios patterns

Le nom du cas d'utilisation est « Gérer passage »,
Le scénario de base est de type principal selon LSP, les autres sont alternatifs.
L'acteur est l'utilisateur.

//Scénario pattern principal

BEGIN
gérer passage ;
principal ;
utilisateur ;
/SP
lecteur, OpD, 1, insérer, boolean
carte, OpD , 2, valider, numérocarte, dateexpiration ; boolean
lecteur, OpD, 3, retirer , boolean
lampe vert, OpP, 1, allumer, état ;boolean
porte , OpP,2, débloquer, boolean
porte, OpP, 3, franchir, boolean
journal, OpP, 4, miseAjour, numérocarte, dateexpiration, dateentree, tempsentree ;boolean
porte , OpR,1, bloquer, boolean
lampe vert, OpR, 2, éteindre, état ; boolean
SP/
END

//Scénario pattern alternatif 1

BEGIN
gérer passage;
alternatif;
utilisateur;
/SP
lecteur, OpD, 1, insérer, boolean
carte, OpD , 2, valider, numérocarte, dateexpiration ; boolean
lampe rouge, OpP, 1, allumer, état ; boolean
lampe rouge, OpR, 1, éteindre, état ; boolean
lecteur, OpR, 2, retirer , boolean
SP/
END

//Scénario alternatif 2

BEGIN
gérer passage ;
alternatif;
utilisateur ;
/SP
lecteur, OpD, 1, insérer, boolean
carte, OpD , 2, valider, numéroCarte, dateExpiration ; boolean
lecteur, OpD, 3, retirerCarte , état ; boolean
lampe vert, OpP, 1, allumer, état ; boolean
porte , OpP,2, débloquer, boolean
porte, OpP, 3, franchir, boolean
porte , OpR,1, bloquer, boolean
lampe vert, OpR, 2, éteindre, état ; boolean
SP/
END

Pour chaque scénario pattern, un fragment de pattern de conception relatif est créé. Le pattern de conception du scénario pattern principal est composé des objets lecteur, carte, lampe vert et porte.
Selon ce scénario, chaque objet est décrit par un ensemble d'attributs et d'opérations. De la première ligne du scénario pattern principal nous tirons l'objet lecteur, son opération insérer sans paramètres d'entrée mais avec un paramètre de sortie boolean. De la deuxième ligne nous tirons l'objet carte, son opération valider avec paramètres d'entrée le numéroCarte et la dateExpiration et un paramètre de sortie boolean. De la troisième ligne nous tirons l'objet lecteur, son opération retirer sans paramètres d'entrée et un paramètre de sortie boolean. De la quatrième ligne nous tirons l'objet lampe verte, son opération allumer avec paramètres d'entrée état et un paramètre de sortie boolean. De la

cinquième et la sixième ligne nous tirons l'objet porte, ses deux opérations débloquer et franchir sans paramètres d'entrée mais avec un paramètre de sortie boolean. De la ligne sept, c'est l'objet journal à tirer avec l'opération "miseAjour" et les paramètres d'entrée "numérocarte", "dateexpiration", "dateentree" et "tempsentree" et un paramètre de sortie "boolean". De la ligne huit nous avons l'objet "porte" de nouveau avec l'opération "bloquer" sans paramètre d'entrée et un seul paramètre de sortie "boolean". De la dernière ligne nous tirons l'objet "lampe verte" et son opération "éteindre" avec un paramètre d'entrée "état" et un paramètre de sortie "boolean".
Ceci aboutit à concevoir les classes suivantes (figure 3.20):

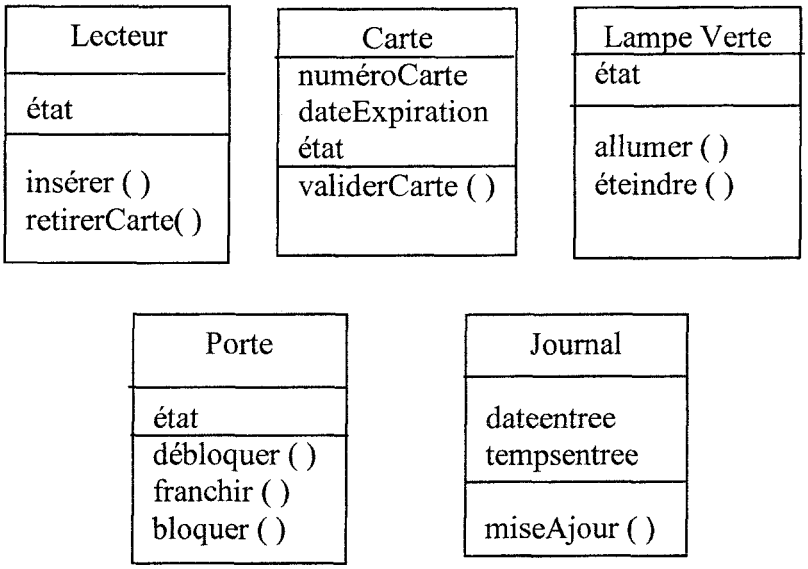


Figure 3.20: classes extraites du scénario pattern principal

Ainsi, nous pouvons conclure des relations possibles entre les classes (figure 3.21). La classe Journal est en association avec la classe Carte, cette association est présente par le fait que la classe Journal utilise des attributs descripteurs de la classe Carte (voir la ligne 7 du scénario principal).

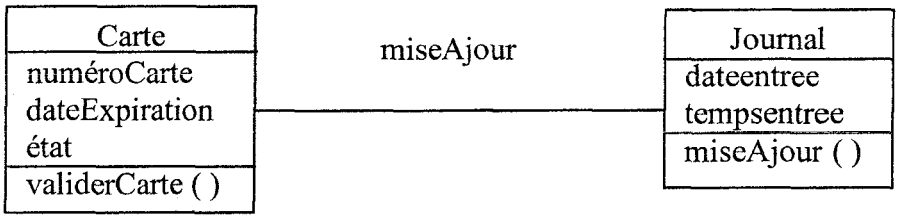


Figure 3.21: association extraite du scénario pattern principal

Le schéma de conception du scénario alternatif 1 est donné dans la figure (figure 3.22):

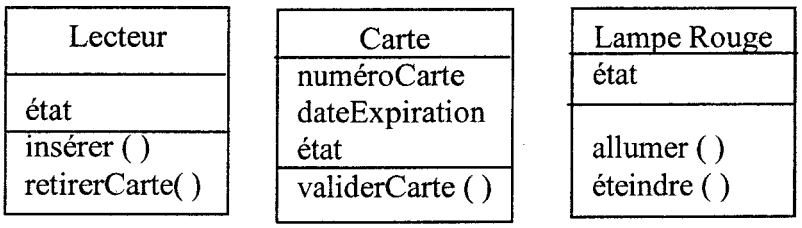


Figure 3.22: classes extraites du scénario pattern alternatif 1

Le schéma de conception du scénario alternatif 2 est donné dans la figure (figure 3.23):

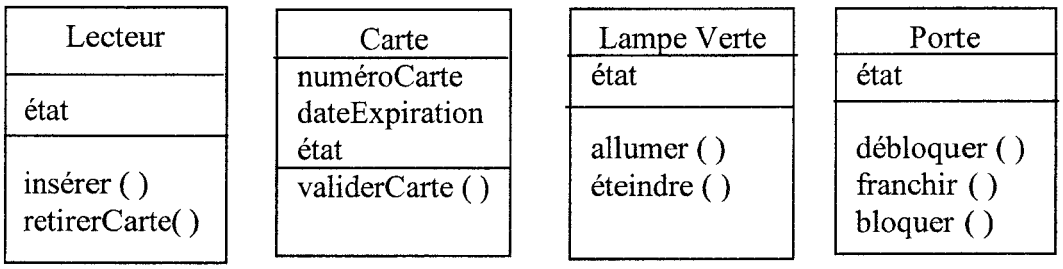


Figure3.23: classes extraites du scénario alternatif 2

La construction du pattern de conception global du cas d'utilisation « gérer passage » est réalisée par la composition itérative de tous les fragments de schémas de conception cités précédemment. Nous pouvons ajouter lors de cette construction des améliorations au pattern de conception global dont le but est de mieux structurer les classes. Les classes d'objets qui possèdent les mêmes caractéristiques peuvent être généralisées. Dans notre exemple les deux lampes verte et rouge peuvent être généralisées (figure 3.24).

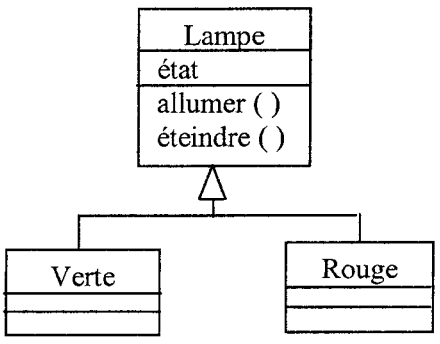


Figure 3.24: généralisation des classes

Le pattern de conception global du cas d'utilisation « gérer passage » devient (figure 3.25):

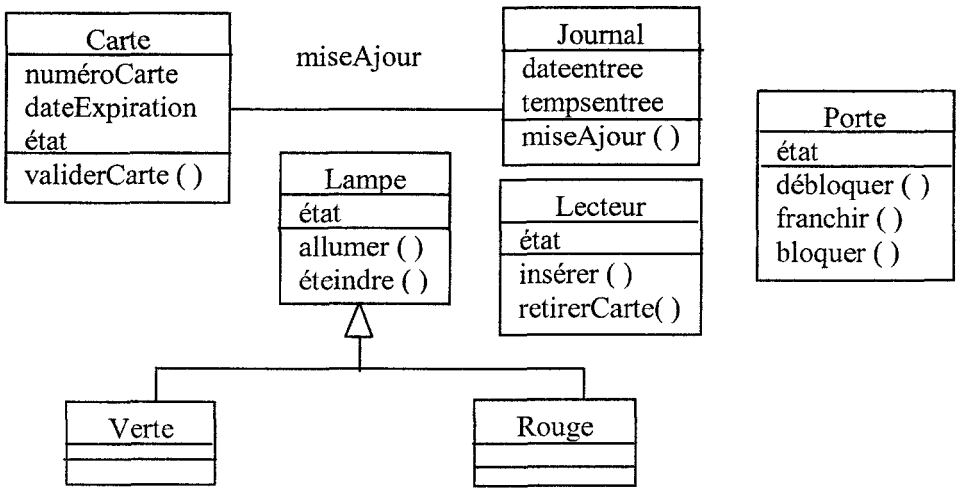


Figure 3.25: Pattern de conception du cas d'utilisation « gérer passage »

3.3.4 Principe d'adaptation

Durant l'étape d'adaptation, la solution du cas source (pré-solution) est adaptée pour obtenir une solution finale au problème cible. L'adaptation est un des processus les plus délicats du RàPC. Dans cette étape, la pré-solution proposée peut être corrigée, acceptée ou refusée par l'ingénieur système. Dans notre approche l'adaptation concerne deux niveaux ; le premier consiste à modifier ou non les scénarios patterns renvoyés par l'étape de remémoration et le deuxième consiste à introduire des modifications aux patterns de conception à l'aide des éditeurs graphiques.

Mise en œuvre de l'EMI
Chapitre 4

Après avoir présenté les concepts clefs de notre approche, et avoir établi une nouvelle vision de développement des logiciels basée sur la réutilisabilité des connaissances métier, nous présentons dans ce chapitre la description de l'Environnement de Modélisation intelligent (EMI), son architecture, sa plate-forme, ses principaux composants, ses interactions avec l'extérieur et les différents outils de base que l'environnement proposera pour l'aide au développement des logiciels.

4.1 Champs d'application de l'EMI

L'EMI est un environnement de développement de tout genre de logiciels. Cet environnement est orienté sujet, il est fonction de la base de cas qu'il comporte. Ainsi, pour développer un logiciel de gestion bancaire, la base de cas utilisée doit contenir le dictionnaire de connaissances métier et les scénarios patterns relatifs au domaine de l'application.

Tout d'abord, et avant tout développement il faut construire des EMI orientés sujet. Ainsi, la base de cas est le point de départ de tout EMI. Le RàPC est le noyau de l'EMI, ses domaines d'application sont très nombreux et variés. Le RàPC a fait l'objet de développement de beaucoup de logiciels tels que Kaidara, CBR Express, ReCall, Esteem, CBR-Works, ReMind, Case Advisor. À leur tour, ces logiciels ont permis d'élaborer des systèmes de RàPC spécifiques à certaines applications industrielles. En voici quelques exemples: CaseBank, CaseLine, Kaidara, CBR Express, Patdex, Creek, PROTEUS, NEMOSYS, etc... Les détails de ces logiciels sont présentés dans l'annexe B.

Donc, le champ d'application de l'EMI n'est pas limité, il dépend de la nature de sa base de cas. Pour mener à bien notre nouvelle vision de développement des logiciels, nous proposons en premier lieu de construire les bases de cas de l'EMI.

Les expert métier et de conception sont invites à assister à la construction des bases de cas orientes sujet. Notre environnement met à la disposition des experts tous les outils qui leur permettent de capitaliser leurs connaissances et expériences, afin de les rendre facilement réutilisables par d'autres utilisateurs (comme les développeurs).

L'expert métier est responsable de transformer le contexte du métier, à l'aide de l'outil LSP de l'EMI, en deux grands composants : Le dictionnaire du métier et la base des scénarios patterns.
Le contexte métier est une illustration de [Frey, Gomes et Sagot 07]:

- La culture métier: qui traduit l'histoire cognitive de l'organisation étudiée,
- Les processus métier: qui représentent les activités, ressources et délivrables requis (activités séquentielles, parallèles, terminales, etc.),
- L'expertise métier: qui traduit les règles expertes sous forme de conseils, contraintes, choix métier (cahier des règles),
- Le vocabulaire métier: qualifié aussi de glossaire métier, regroupant les différents termes métier et des liens entres eux (réseau sémantique), avec à la clé, des exemples, des synonymes, etc.
- L'expérience métier : regroupant des recueils de cas décrivant des cas de référence, des bêtises ou des exclusions métier.

L'expert de conception, celui qui complète la base des scénarios patterns par des schémas de conception (patterns de conception) sous forme de diagramme de classes.
L'expert de conception peut profiter des patterns de conception (design patterns) du [GoF] qui offrent une réutilisabilité informelle. Plus loin, l'expert de conception peut profiter des bibliothèques de patterns de conception à des problèmes récurrents tels que l'outil de [Rapicault 99].

4.2 Architecture et plate-forme de l'EMI

L'EMI est un environnement de développement des applications logicielles à base d'objets. L'environnement de l'EMI est l'espace où évoluent les principaux artifacts qui concourent au processus de développement. L'EMI fournit aux utilisateurs un espace de réutilisation des connaissances métier pour la construction des modèles des besoins, de génération automatique des modèles d'analyse (par exemple le diagramme de classes en UML) et d'édition du modèle d'analyse pour faire les modifications nécessaires afin d'adapter ce modèle aux besoins réels du client. L'EMI fournit aussi un autre espace pour la représentation des connaissances métier sous forme de scénarios patterns, du dictionnaire du vocabulaire métier et de construction de base de cas similaire à ceux utilisés dans les systèmes RàPC.

Plusieurs acteurs contribuent à la réalisation de cet environnement et ils sont mentionnés dans la table 4.1 avec une indication fonctionnelle pour chaque acteur. Les acteurs de notre environnement sont classifiés en deux groupes : le premier groupe comporte l'expert métier et l'expert de conception qui manipulent la partie hors ligne de l'environnement (partie administration de l'environnement) ; le deuxième groupe comporte l'ingénieur système et le client qui manipulent la partie en ligne de l'environnement (partie exploitation de l'environnement).

La vue hors ligne concerne la création de la base de cas de l'EMI. La vue en ligne a pour objectif de réutiliser de façon différente les connaissances de la base de cas dans des buts fonctionnels différents.

Acteur	Vue	Fonction
Expert métier	Hors ligne	Saisie et revue des connaissances métier
Expert métier	Hors ligne	Saisie du dictionnaire du vocabulaire métier
Expert métier	Hors ligne	Compilation des scénarios patterns
Expert de conception	Hors ligne	Construction de la base de cas
Ingénieur système	En ligne	Saisie des besoins client
Ingénieur système	En ligne	Construction du modèle des besoins
Ingénieur système	En ligne	Génération du modèle d'analyse
Ingénieur système	En ligne	Edition et adaptation du modèle d'analyse

Table 4.1 : Fonctions des acteurs dans les deux vues de l'EMI

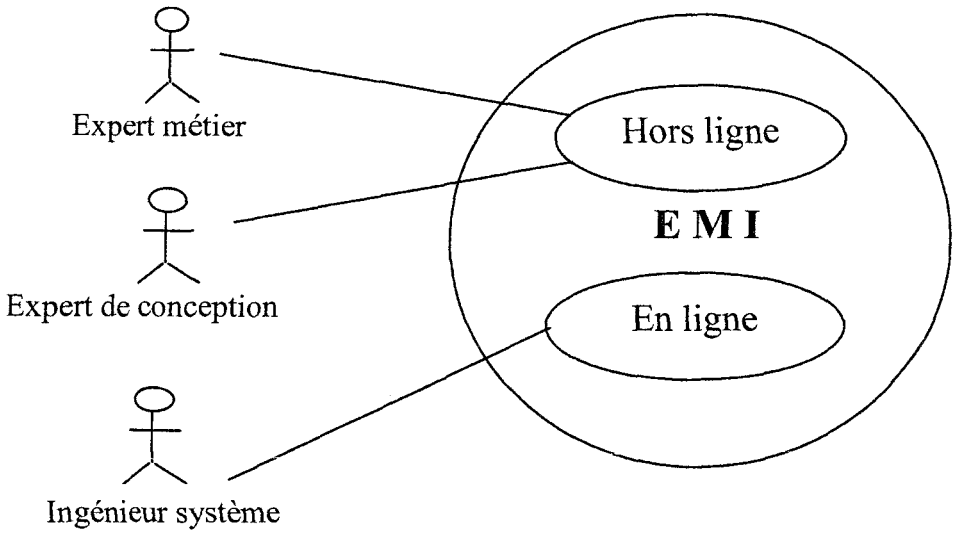


Figure 4.1: Les acteurs des vues de l'EMI

4.2.1 Architecture de base de l'EMI

L'architecture de base de l'EMI s'articule autour de deux composants principaux l'un est hors ligne, l'autre est en ligne. Le composant en ligne évolue en fonction du composant hors ligne qui lui fournit les connaissances nécessaires pour son évolution (figure 4.2).

Le composant hors ligne est constitué d'une base de cas et d'un dictionnaire de vocabulaires métier. Il est manipulé par trois modules, l'éditeur du dictionnaire, le langage LSP et l'éditeur des patterns de conception et des cas. Le composant en ligne évolue dans l'EMI pour fournir un modèle des besoins sous forme d'un diagramme de

cas d'utilisation et un modèle d'analyse sous forme de diagramme de classes. Les modules nécessaires intégrés dans le composant en ligne sont le générateur du diagramme de cas d'utilisation, le générateur du diagramme de classes et leurs éditeurs. L'EMI permet aussi, à ces diagrammes, l'exportation à des éditeurs externes.

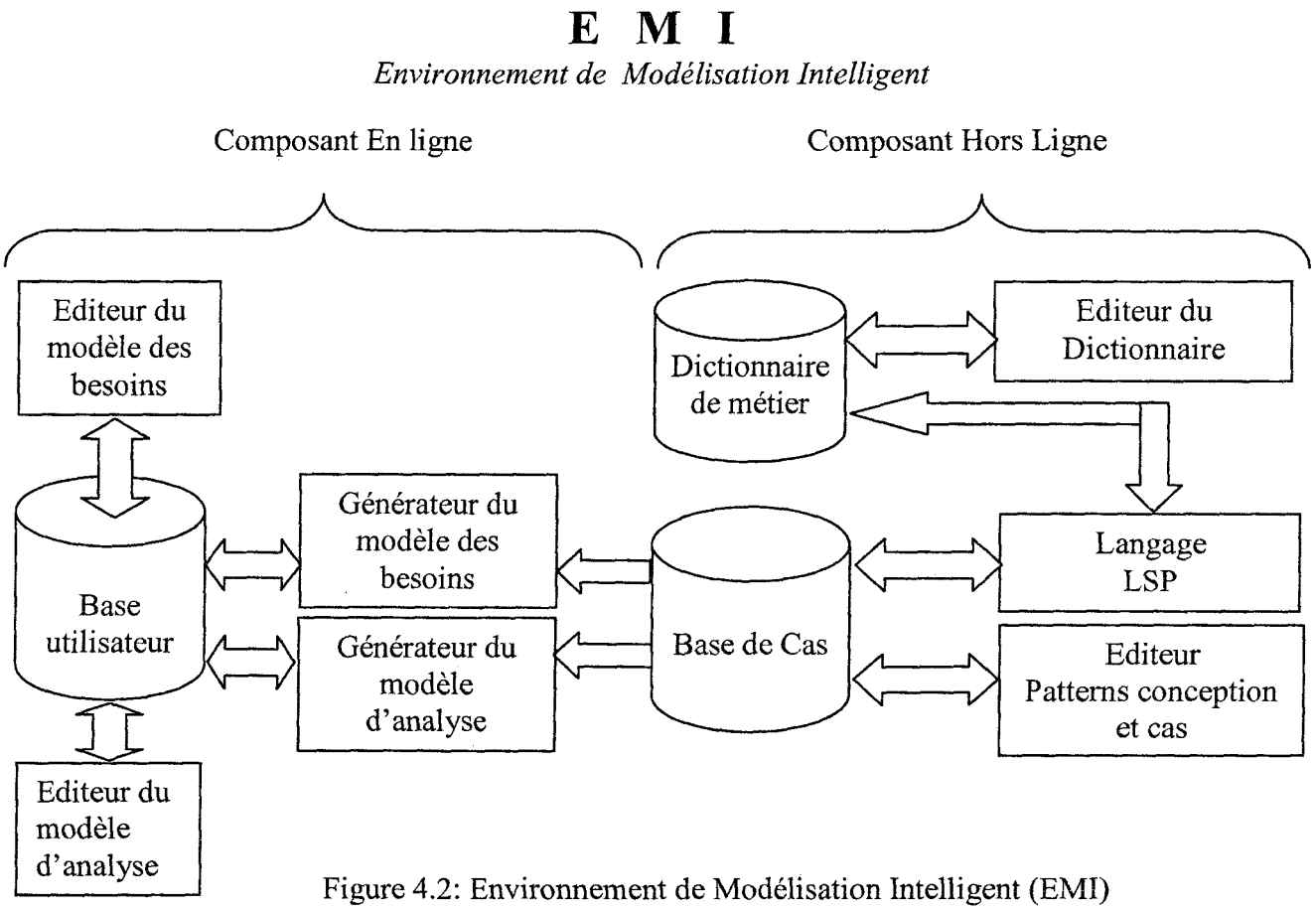


Figure 4.2: Environnement de Modélisation Intelligent (EMI)

4.2.1.1 Composant hors ligne

a- Dictionnaire de métier

Le dictionnaire de métier est l'un des sous composants du composant hors ligne. Il est manipulé par l'expert du métier pour y ajouter des nouveaux mots ou termes avec leurs synonymes possibles. La structure du dictionnaire métier telle qu'elle est présentée dans le chapitre 3 est organisée en domaines, sous domaines et activités. Les mots ou les termes utilisés dans l'une des activités d'un sous domaine sont des objets, des opérations, des acteurs ou des attributs. Chaque mot ou terme peut posséder des synonymes métiers. Dans la table 4.2 et la table 4.3, nous présentons quelques mots utilisés pendant les activités de dépôt et de retrait d'argent dans un compte client. Ces termes sont issus du glossaire des opérations bancaires courantes élaboré dans le cadre d'une concertation entre les banques, les pouvoirs publics et les associations de consommateurs sous l'égide du CCSF (Comité Consultatif du Secteur Financier) et sont issus aussi du glossaire de la

Société Générale de France. Il est destiné à aider les fonctionnaires du secteur bancaire de comprendre les termes employés dans les relations bancaires au quotidien [Glossaire SG 08]. Le détail du dictionnaire métier est représente dans l'annexe F. Voici un exemple :

Domaine	Bancaire
Sous Domaine	Gestion des comptes client
Activité	Dépôt d'argent

Mot	Nature	Forme abstraite	Explication	Synonyme
Commission	Attribut	Commission	Somme perçue par une banque en rémunération d'un service fourni à son client.	Rémunération Remise Intérêt
Crédit	Opération	Créditer	Opération comptable qui augmente le solde du compte, par exemple à la suite d'un virement reçu, d'un dépôt d'espèce, ou d'une remise de chèque.	Déposer Dépôt
Opérateur	Acteur	Opérateur	Employé de la banque pour servir les clients de leurs opérations comptables.	Guichetier Agent Employé Utilisateur

Table 4.2 : Quelques termes et mots de l'activité dépôt d'argent

Domaine:	Bancaire
Sous Domaine:	Gestion des comptes client
Activité:	Retrait d'argent

Mot	Nature	Forme abstraite	Explication	Synonyme
Agence	Acteur	-	Lieu d'accueil de la clientèle d'une banque.	Branche
Débit	Opération	Débiter	Opération comptable qui diminue le solde du compte, par exemple à la suite de l'émission d'un chèque, d'un prélèvement ou d'un retrait d'espèces à un DAB.	Retirer Prélever
Débiteur	Acteur	Débiteur	Personne physique ou morale tenue de remplir une obligation. Le plus souvent, il s'agit de payer une somme d'argent à un créancier. Un compte de dépôt est dit débiteur lorsque son solde est négatif.	Client

Table 4.2 : Quelques termes et mots de l'activité retrait d'argent

Le dictionnaire des métiers fournit un moyen efficace pour la rédaction des scénarios métiers et surtout pendant la construction du modèle des besoins du client. L'éditeur du dictionnaire permet de mettre à jour les mots et les termes. Il permet aussi d'ajouter des synonymes aux termes et aux mots ayant les mêmes sémantiques dans le même domaine des métiers.

La nature d'un mot ou d'un terme dans le dictionnaire exprime la fonctionnalité du mot dans le contexte métier : elle peut être un acteur, un objet, une opération ou un attribut. La forme abstraite d'un mot ou d'un terme est l'écriture que l'expert métier utilise lors de la rédaction des scénarios patterns. L'explication est un champ descriptif détaillé pour les mots et les termes du dictionnaire tels qu'ils sont utilisés dans le domaine bancaire. Le synonyme contient tous les mots ou les termes équivalents aux mots sources et permet au langage LSP d'identifier les mots à travers leurs synonymes.

b- La base de cas

La base de cas est la structure la plus importante du composant hors ligne. C'est dans cette base où les scénarios patterns et les patterns de conception sont enregistrés.
La base de cas ou base de scénarios est construite par deux étapes :

- création des scénarios patterns,
- création des scénarios de conception.

La création des scénarios patterns est effectuée par l'expert métier qui utilise le langage LSP pour rédiger des scénarios patterns. A la fin de rédaction de chaque scénario pattern, l'expert métier compile le scénario pattern pour éviter toute source d'erreur et pour que le scénario pattern soit compatible avec la grammaire du langage. Après la compilation, le scénario pattern, sera enregistré dans la base de cas comme un problème source.

La création des patterns de conception est un travail réservé à l'expert de conception qui, pour chaque scénario pattern, doit élaborer un pattern de conception en formalisme UML afin de compléter le cas source par une solution source. Dans notre approche nous avons utilisé le diagramme de classes comme un pattern de conception.
L'EMI met à la disposition de l'expert de conception un éditeur de scénario de conception qui permet de construire, de modifier et de gérer des patterns de conception.
Après la deuxième étape, la base de cas est prête, l'ingénieur système peut la manipuler pour construire un modèle d'objets.

4.2.1.2 Composant en ligne

Ce composant est en ligne car c'est dans ce composant qu'évolue le développemen d'une application informatique. Ce composant est formé d'une base utilisateur dans laquelle se trouvent le modèle des besoins et le modèle d'analyse du client. Cette base est manipulée par l'ingénieur système en utilisant les quatre outils de base suivants:

- générateur du modèle des besoins,
- générateur du modèle d'analyse,
- éditeur du modèle d'analyse,
- éditeur du modèle des besoins.

Le générateur du modèle des besoins est l'outil qui, à partir d'un écran de saisie des formulaires descriptifs des besoins client, génère automatiquement un modèle des besoins sous forme d'un diagramme de cas d'utilisation en formalisme UML. Ce diagramme de cas d'utilisation peut être exporté vers des éditeurs graphiques (exemple VISIO). Le générateur du modèle d'analyse permet la construction d'un diagramme de classes en fonction du modèle des besoins. Le principe adopté par le générateur est de construire par étape le diagramme de classes. Pour chaque cas d'utilisation du modèle des besoins le générateur identifie, dans la base de cas, le pattern de conception correspondant et puis il compose tous ces patterns dans un diagramme de classes global. L'identification du pattern de conception utilise le principe de test de similarité détaillé dans le chapitre 3.

Les éditeurs des modèles des besoins et d'analyse sont utilisés par l'ingénieur système dans la phase d'adaptation. L'ingénieur système peut introduire aux modèles résultants des modifications pour les adapter au maximum aux exigences du client.

4.2.2 Plate-forme de l'EMI

La plate-forme de l'EMI est développée à l'aide du « NetBeans IDE 6.0 », elle est constituée de deux modules de programmes contenant chacun un groupe d'interfaces. Chaque module est dédié à des utilisateurs différents. Le premier module appelé « LSP & CBR Case Building V.1.0 » est indépendant du deuxième de point de vue manipulation. Il est dédié aux experts métier et de conception qui contribuent à la création de la base de cas de l'EMI. L'expert métier est responsable de l'édition des scénarios patterns et du dictionnaire des connaissances métier. L'expert de conception, lui aussi est responsable de la rédaction des patterns de conception pour compléter la base de cas.

Le module « LSP & CBR Case Building V.1.0 » comprend un groupe d'interfaces formé de six interfaces principales. Quatre interfaces manipulées par l'expert métier et deux interfaces manipulées par l'expert de conception (voir annexe G). Le deuxième Module appelé « Project Developer V.1.0 » comprend un groupe d'interfaces formé de cinq interfaces principales. L'ingénieur système est l'acteur principal de ces cinq interfaces (voir annexe G). En plus de ces interfaces, « Project Developer V.1.0 » utilise deux éditeurs graphiques et d'adaptation. Le premier est destiné à manipuler le diagramme de classes fourni par le générateur du diagramme de classes et le deuxième pour la manipulation du diagramme de cas d'utilisation.

4.3 Interaction Homme-Machine à travers l'EMI

Le diagramme de séquences de l'UML présente un moyen efficace pour la description de l'échange de messages entre les acteurs et le système. Le diagramme de séquences est une représentation intuitive lorsque l'on souhaite concrétiser des interactions entre deux entités (deux sous-systèmes ou deux classes d'un futur logiciel). Ils permettent à l'architecte de créer au fur et à mesure sa solution. Cette représentation intuitive est également un excellent vecteur de communication dans une équipe d'ingénierie pour discuter cette solution.

Les diagrammes de séquence peuvent également servir à la problématique de test. Les traces d'exécution d'un test peuvent en effet être représentées sous cette forme et servir de comparaison avec les diagrammes réalisés lors des phases d'ingénierie [Cian 03].

4.3.1 Interface Expert métier / Système

L'échange de messages entre l'expert métier (acteur principal) et le système pendant la rédaction des scénarios patterns est représenté en figure 4.3. Ce diagramme de séquences est nominal, il représente les interactions entre l'acteur et le système dans les bonnes conditions, c'est-à-dire l'acteur édite un scénario, le compile sans erreurs et puis le sauvegarde.

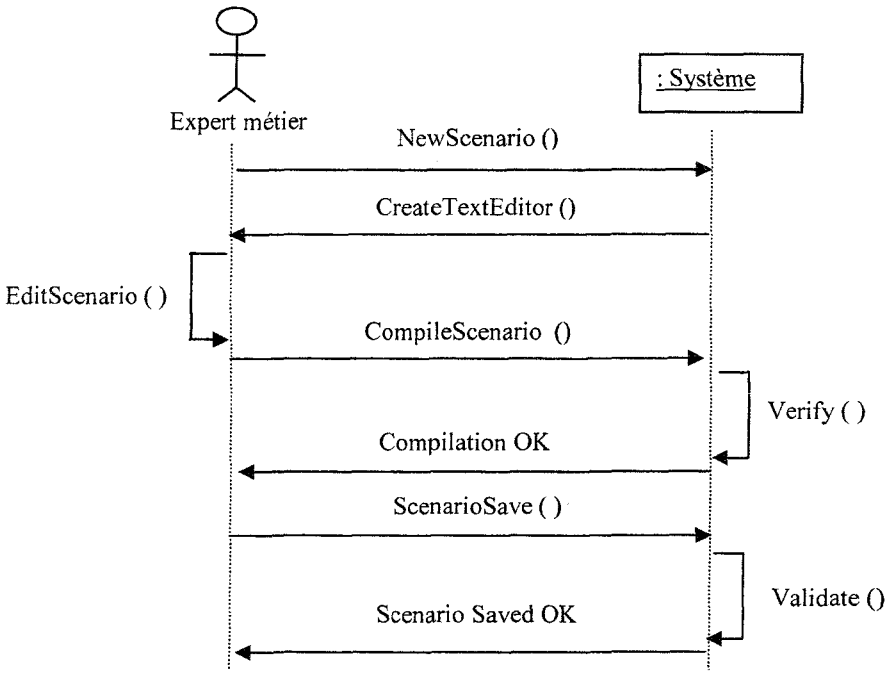


Figure 4.3: Diagramme de séquences pour la rédaction des scénarios patterns

L'interface du langage LSP ne permet pas de sauvegarder un scénario pattern si la compilation contient des erreurs. Un scénario alternatif possible est présenté dans le diagramme de séquences suivant (figure 4.4).

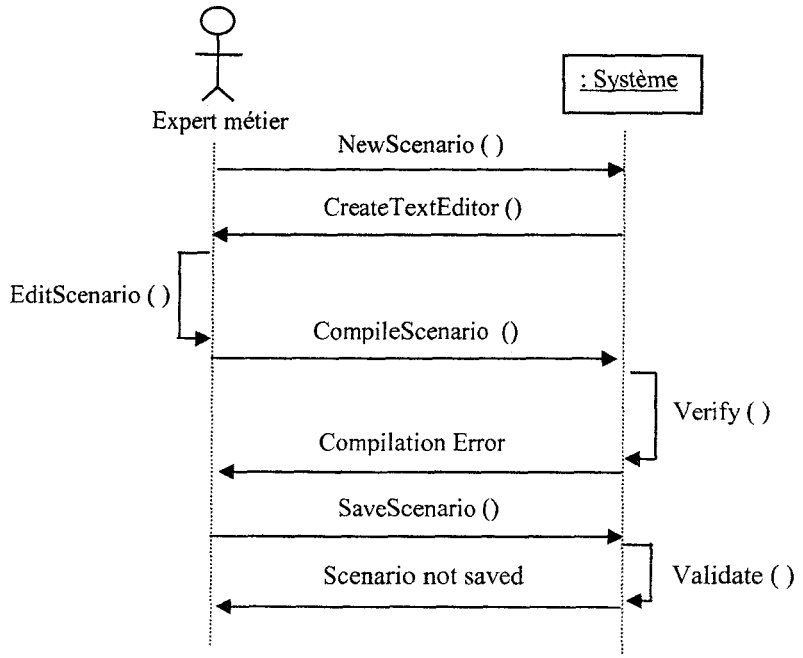


Figure 4.4: Diagramme de séquences pour un scénario alternatif

L'interface du langage LSP permet aussi de manipuler le dictionnaire des connaissances métier. L'expert métier peut consulter, ajouter et modifier des mots ou des termes dans le dictionnaire. Le diagramme de séquences (figure 4.5) représente l'échange de messages entre l'expert métier et le système lors de la recherche d'un mot dans le dictionnaire des connaissances métier.

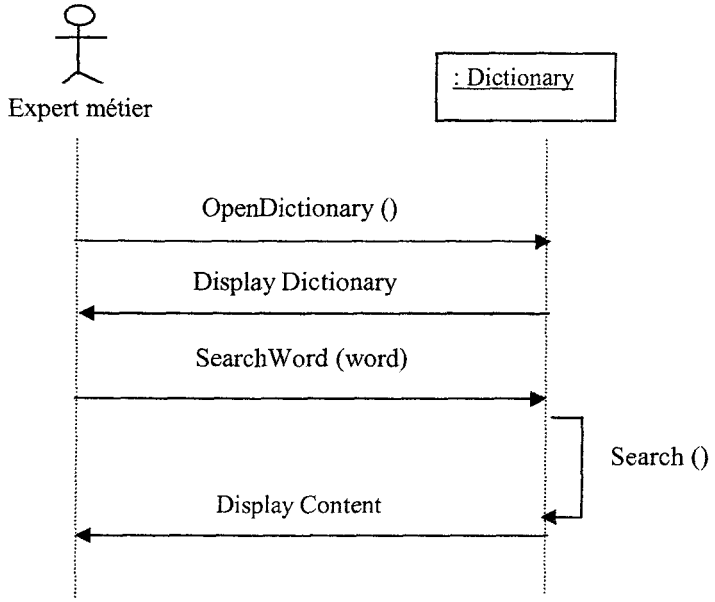


Figure 4.5: diagramme de séquences pour chercher un mot dans le dictionnaire

4.3.2 Interface Expert de conception / Système

Après avoir terminé la création des scénarios patterns, le rôle de l'expert de conception commence. Ce rôle consiste à compléter la base de cas par des patterns de conception. L'expert de conception construit pour chaque scénario pattern un pattern de conception convenable. Ce pattern de conception est une description structurée d'un diagramme de

classes. Le diagramme de séquences (figure 4.6) représente l'échange de messages entre l'expert de conception et le système pendant la création des patterns de conception.

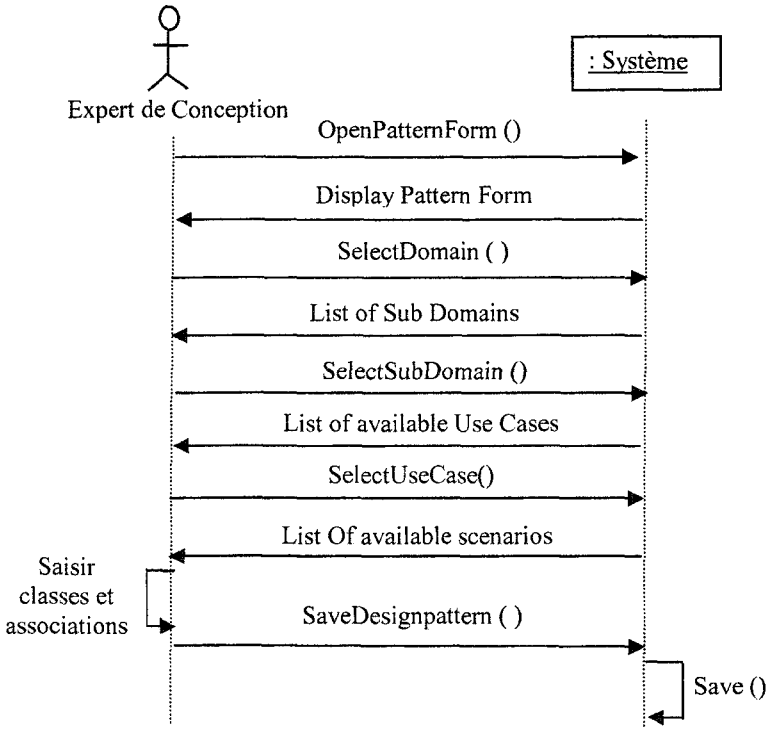


Figure 4.6: Diagramme de séquences pour créer un pattern de conception

4.4 Description des outils de base

4.4.1 Editeur et compilateur des scénarios patterns

L'expert métier introduit les scénarios patterns conformément à la grammaire et la syntaxe du langage LSP. L'éditeur met à la disposition de l'expert métier des options dans le menu général pour créer, modifier et sauvegarder des scénarios patterns. Lorsqu'un scénario pattern est compilé sans erreurs, il pourra être sauvegardé. La compilation se fait à partir de l'option « Compile » dans l'interface de rédaction d'un scénario pattern (voir annexe G). L'expert métier peut accéder au dictionnaire des connaissances métier au moment de la saisie des scénarios patterns. L'expert métier pourra alors enrichir le dictionnaire par des nouveaux mots et termes. L'accès se fait depuis l'option « Dictionary » (voir annexe G) .

4.4.2 Développeur des projets (Project Developer 1.0)

Le logiciel « Project Developer 1.0 » sera le fruit de l'approche méthodologique décrite dans cette thèse et présente l'Environnement de Modélisation intelligent (EMI). Il intègre les outils principaux suivants: UCDG 1.0 (Use Case Diagram Generator), CDG 1.0 (Class diagram Generator), UCDE 1.0 (Use Case Diagram Editor), CDE 1.0 (Class diagram Editor) et BKD 1.0 (Business Knowledge Dictionary) (voir Annexe G) .

4.4.2.1 L'outil UCDG 1.0

Cet outil est responsable de traduire les besoins informels du client en un modèle des besoins semi-formel et graphique (diagramme de cas d'utilisation). Chaque cas

d'utilisation du diagramme est constitué en un ensemble de scénarios patterns sélectionnés à partir de la base de cas (figure 4.7).

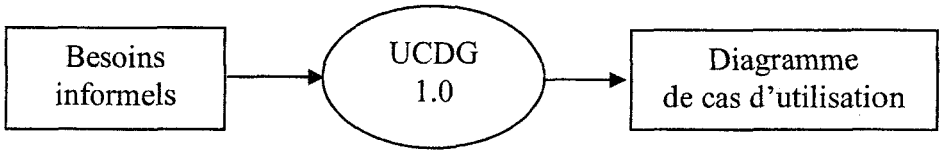


Figure 4.7: Outil UCG 1.0 pour la génération du diagramme de cas d'utilisation

L'outil UCDG 1.0 intègre les algorithmes « filtrage », « Renvoyer_cas_similaires » « Renvoyer_cas_plus_similaires » et « sélectionner_scénarios_patterns » qui sont présentés dans le chapitre 3. Cet outil applique les techniques de test de similarité entre le modèle initial des besoins informels du client et les scénarios patterns de la base de cas afin de sélectionner ceux qui sont les plus similaires aux exigences du client.
Le résultat fourni par cet outil est un diagramme de cas d'utilisation décrit en formalisme UML.

4.4.2.2 L'outil UCDE 1.0

L'outil UCDE 1.0 est un éditeur graphique et textuel du diagramme de cas d'utilisation. Il permet de modifier et d'adapter le diagramme de cas d'utilisation résultant aux besoins réels du client. Tout d'abord, l'outil est un éditeur graphique, Il possède les moyens pour manipuler des éléments graphiques à l'intérieur d'un espace de dessin. Ces éléments graphiques sont nécessaires à la construction d'un diagramme de cas d'utilisation comme les cercles, les lignes, les flèches, les rectangles, etc.....
L'outil UCDE 1.0 possède aussi un éditeur de texte pour la manipulation des scénarios patterns renvoyés par l'UCDG 1.0. Cet ensemble de scénarios patterns est analysé par l'ingénieur système et le client. Cette analyse a pour rôle de modifier, d'annuler ou d'accepter certains scénarios afin de construire un système le plus proche des besoins spécifiés.

4.4.2.3 L'outil CDG 1.0

L'outil CDG 1.0 est un générateur de diagramme de classes. Pour chaque cas d'utilisation du modèle des besoins un ensemble de patterns de conception est sélectionné. L'outil CDG 1.0 compose le diagramme de classes par itération. Dans chaque itération, il traite un cas d'utilisation et construit une partie du diagramme de classes. Le nombre d'itérations dépend du nombre de cas d'utilisation dans le diagramme de cas d'utilisation.

4.4.2.4 L'outil CDE 1.0

L'outil CDE 1.0 est un éditeur graphique du diagramme de classes. Il permet de modifier et d'adapter le diagramme de classes aux besoins réels du client. Cet éditeur ressemble à tous les éditeurs graphiques de UML, comme le logiciel « Rational Rose ». Cet outil permet aussi de consulter le diagramme de classes sous forme d'un texte

structuré. L'outil possède aussi une interface d'exportation du diagramme de classes vers les autres éditeurs graphiques.

4.4.2.5 L'outil BKD 1.0

L'outil BKD 1.0 permet de consulter le dictionnaire de connaissances métier. L'ingénieur système, durant le développement de son logiciel, il a besoin toujours d'un support de connaissances métier pour lui fournir la sémantique nécessaire pour son travail. Il réutilise des scénarios patterns, donc il est toujours face à des nouvelles connaissances dont il a besoin de comprendre. Cette nécessité parait clairement pendant la phase d'adaptation où l'ingénieur système introduit des modifications aux modèles générés. Afin de confirmer les modifications, l'ingénieur système peut faire une comparaison avec les connaissances du dictionnaire de métier.

Evaluation expérimentale et perspectives
Chapitre 5

Ce chapitre présente un prototype de l'environnement EMI dédié au développement des modèles d'analyse pour les applications informatiques. Ce prototype permet de valider les propositions de cette thèse.

A la fin de ce chapitre nous présentons une application que nous avons traitée avec les outils de l'environnement EMI. Cette application servira, comme nous le verrons, à ouvrir des perspectives pour l'utilisation de l'EMI dans le domaine des systèmes d'information.

5.1 Choix du cas d'application

Un cas d'application appartient souvent à une famille d'applications ayant le même contexte de développement. Les informaticiens font recours à utiliser le modèle de prototypage pour développer des logiciels pour des applications similaires. Bien que le modèle de prototypage a réussi dans plusieurs cas mais il présente toujours plusieurs problèmes :

- L'adaptation est une phase trop chère en fonction de la taille de l'application.
- Les prototypes ne représentent que des façades des systèmes complets
- Les prototypes sont souvent construits d'une façon rapide et pas chère, manquent de la qualité logicielle et sont difficiles à maintenir.

L'EMI est une approche différente, le développeur profite de la réutilisabilité des besoins, et de schémas de solutions de conception et des générateurs de modèles. Le développeur peut générer un modèle d'analyse autant de fois qu'il y a de changement dans le modèle des besoins.

5.2 Application « Gestion bancaire »

Le contexte bancaire actuel de forte compétitivité nous amène de plus en plus à utiliser le retour d'expérience. C'est la raison pour laquelle nous avons proposé l'exemple de la « Gestion bancaire » comme une application directe de notre approche.

L'exemple « Gestion bancaire » nous montre toutes les étapes à suivre pour développer un modèle d'analyse dans l'Environnement de Modélisation intelligent (EMI).
Nous allons considérer que l'EMI contient la base de cas et le dictionnaire de métier orientés « Gestion bancaire».

L'exemple suivant montre quelques informations de l'EMI qui sont enregistrées par l'expert métier et l'expert de conception dans la base de cas.
On rappelle que la base de cas est formée d'un ensemble de couples scénario pattern et son pattern de conception.

Dictionnaire

Le dictionnaire de connaissances métier du domaine bancaire est définit dans le chapitre 4. Il est extrait du glossaire général de banque société générale de France (Voir annexe A).

Base de cas

La base de cas est formée des couples (scénario pattern, pattern de conception). Les scénarios patterns sont construits par l'expert de métier et les patterns de conception sont construits par l'expert de conception.

Scénarios patterns et patterns de conception

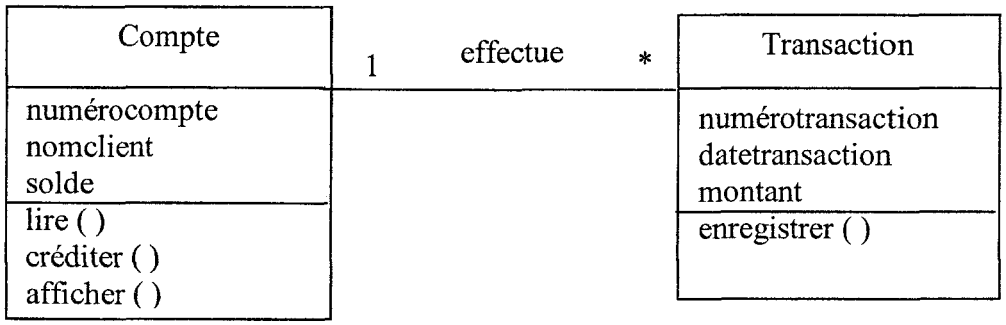
Domaine	Gestion bancaire
Sous domaine	Transactions bancaires
Cas d'utilisation	Dépôt d'argent
Acteur	Opérateur
Type du Scénario	Principal

Scénario Pattern textuel
- L'opérateur saisit le numéro du compte du client. - Le système lit le compte et affiche le nom du client. - L'opérateur saisit la date de la transaction, le montant à déposer et clique sur enregistrer. - Le système génère un numéro de transaction et enregistre la transaction. - Le système met à jour le solde du compte du client.

Scénario Pattern semi formel en langage LSP
BEGIN Dépôt ; Principal ; Opérateur ; /SP

interface, OpD, 1, saisir ; numérocompte
compte, OpD, 2, lire, numérocompte ; boolean
compte, OpD, 3, afficher, nomclient
interface, OpP, 1, saisir ; datetransaction, montant
transaction, OpR, 1, enregistrer, numérotransaction, datetransaction, montant
compte, OpR, 2, créditer, solde
SP/
END

Pattern de conception relatif au scénario pattern principal



Domaine	Gestion bancaire
Sous domaine	Transactions bancaires
Cas d'utilisation	Retrait d'argent
Acteur	Opérateur
Type du Scénario	Principal

Scénario Pattern textuel

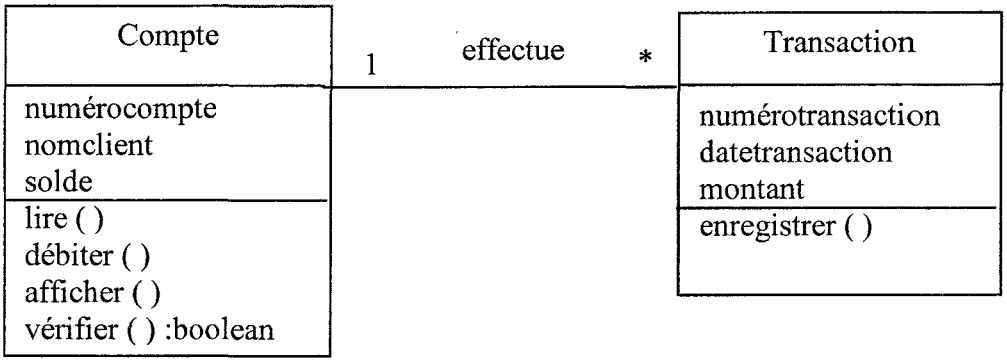
- L'opérateur saisit le numéro du compte.
- Le système lit le compte et affiche le nom du client.
- L'opérateur saisit la date de la transaction, le montant à retirer et clique sur enregistrer.
- Le système vérifie la solvabilité du compte et renvoie « compte solvable».
- Le système génère un numéro de transaction et enregistre la transaction.
- Le système met à jour le solde du compte du client.

Scénario Pattern semi formel en langage LSP

BEGIN
Retrait ;
Principal ;
Opérateur ;
/SP
interface, OpD, 1, saisir ; numérocompte
compte, OpD, 2, lire, numérocompte ; boolean
compte, OpD, 3, afficher, nomclient
interface, OpP, 1, saisir ; datetransaction, montant

compte, OpP, 2, vérifier, solde ; boolean
interface, OpP, 3, afficher, message
transaction, OpR, 1, enregistrer, numérotransaction, datetransaction, montant
compte, OpR, 2, débiter, solde
SP/
END

Pattern de conception relatif au scénario pattern principal



Domaine	Gestion bancaire
Sous domaine	Transactions bancaires
Cas d'utilisation	Retrait d'argent
Acteur	Opérateur
Type du Scénario	Alternatif

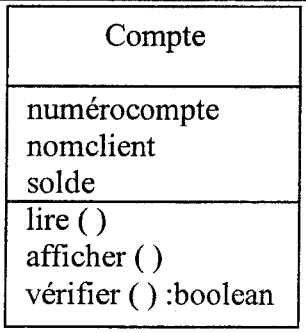
Scénario Pattern textuel

- L'opérateur saisit le numéro du compte.
- Le système lit le compte et affiche le nom du client.
- L'opérateur saisit la date de la transaction et le montant à retirer.
- Le système vérifie la solvabilité du compte et renvoie « compte insolvable».

Scénario Pattern semi formel en langage LSP

BEGIN
Retrait ;
Alternatif ;
Opérateur ;
/SP
interface, OpD, 1, saisir ; numérocompte
compte, OpD, 2, lire, numérocompte ; boolean
compte, OpD, 3, afficher, nomclient
interface, OpP, 1, saisir ; datetransaction, montant
compte, OpP, 2, vérifier, solde ; boolean
interface, OpR, 1, afficher, message
SP/
END

Pattern de conception relatif au scénario pattern alternatif



Domaine	Gestion bancaire
Sous domaine	Transactions bancaires
Cas d'utilisation	Transfert compte
Acteur	Opérateur
Type du Scénario	Principal

Scénario Pattern textuel

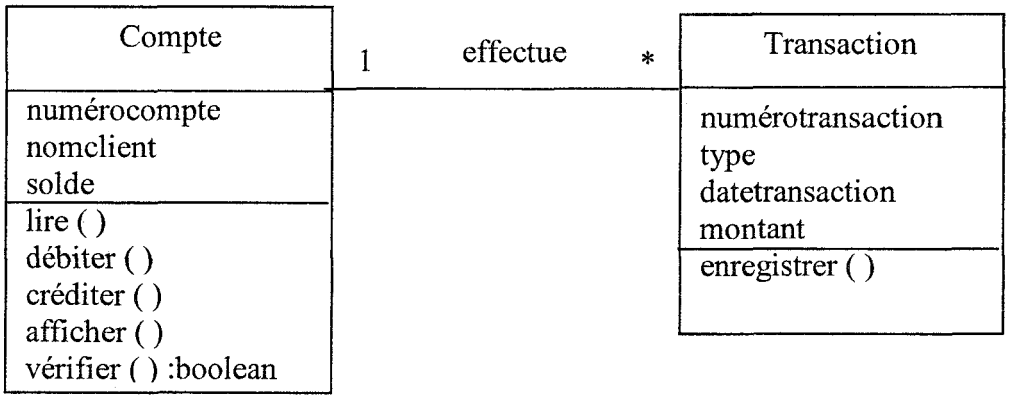
- L'opérateur saisit le numéro du compte à débiter.
- Le système lit le compte et affiche le nom du client.
- L'opérateur saisit le numéro du compte à créditer.
- Le système lit le compte et affiche le nom du client.
- L'opérateur saisit la date de la transaction, le montant à débiter et clique sur enregistrer.
- Le système vérifie la solvabilité du compte débit et renvoie « compte solvable».
- Le système génère un numéro de transaction, indique son type et enregistre la transaction sur les deux comptes.
- Le système met à jour le solde du compte débiteur et le solde du compte créditeur.

Scénario Pattern semi formel en langage LSP

BEGIN
Retrait ;
Principal ;
Opérateur ;
/SP
interface, OpD, 1, saisir ; numérocompte
compte, OpD, 2, lire, numérocompte ; boolean
compte, OpD, 3, afficher, nomclient
interface, OpD, 4, saisir ; numérocompte
compte, OpD, 5, lire, numérocompte ; boolean
compte, OpD, 6, afficher, nomclient
interface, OpP, 1, saisir ; datetransaction, montant
compte, OpP, 2, vérifier, solde ; boolean
interface, OpP, 3, afficher, message

transaction, OpR, 1, enregistrer, numérotransaction, typetransaction, datetransaction, montant
transaction, OpR, 2, enregistrer, numérotransaction, typetransaction, datetransaction, montant
compte, OpR, 3, débiter, solde
compte, OpR, 4, créditer, solde
SP/
END

Pattern de conception relatif au scénario pattern principal



Domaine	Gestion bancaire
Sous domaine	Transactions bancaires
Cas d'utilisation	Consulter compte
Acteur	Opérateur
Type du Scénario	Principal

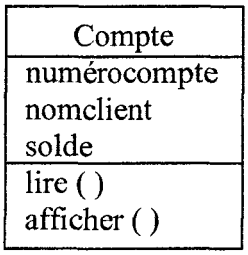
Scénario Pattern textuel

- L'opérateur saisit le numéro du compte.
- Le système lit le compte et affiche le nom du client et le solde du compte.

Scénario Pattern semi formel en langage LSP

BEGIN
Consulter ;
Principal ;
Opérateur ;
/SP
interface, OpD, 1, saisir ; numérocompte
compte, OpD, 2, lire, numérocompte ; boolean
compte, OpP, 1, afficher, nomclient, solde
SP/
END

Pattern de conception relatif au scénario pattern principal



Domaine	Gestion bancaire
Sous domaine	Transactions bancaires
Cas d'utilisation	Consulter compte en ligne
Acteur	Internaute
Type du Scénario	Principal

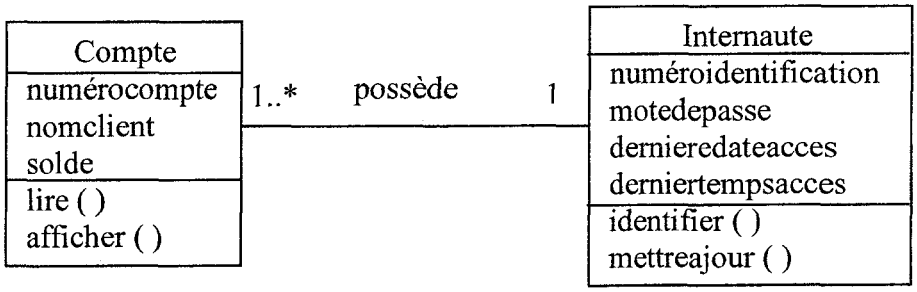
Scénario Pattern textuel

- L'internaute saisit le numéro d'identification et le mot de passe.
- Le système identifie l'internaute et affiche le nom de l'internaute.
- L'internaute saisit le numéro du compte.
- Le système lit le compte et affiche le solde du compte.
- Le système enregistre la dernière date et temps d'accès de l'internaute.

Scénario Pattern semi formel en langage LSP

BEGIN
Consulter ;
Principal ;
Opérateur ;
/SP
interface, OpD, 1, saisir ; numéroidentification, motdepasse
internaute, OpD, 2, vérifier, numéroidentification, motdepasse; boolean
interface, OpD, 1, saisir ; numérocompte
compte, OpD, 2, lire, numérocompte ; boolean
compte, OpD, 3, afficher, nomclient, solde
SP/
END

Pattern de conception relatif au scénario pattern principal

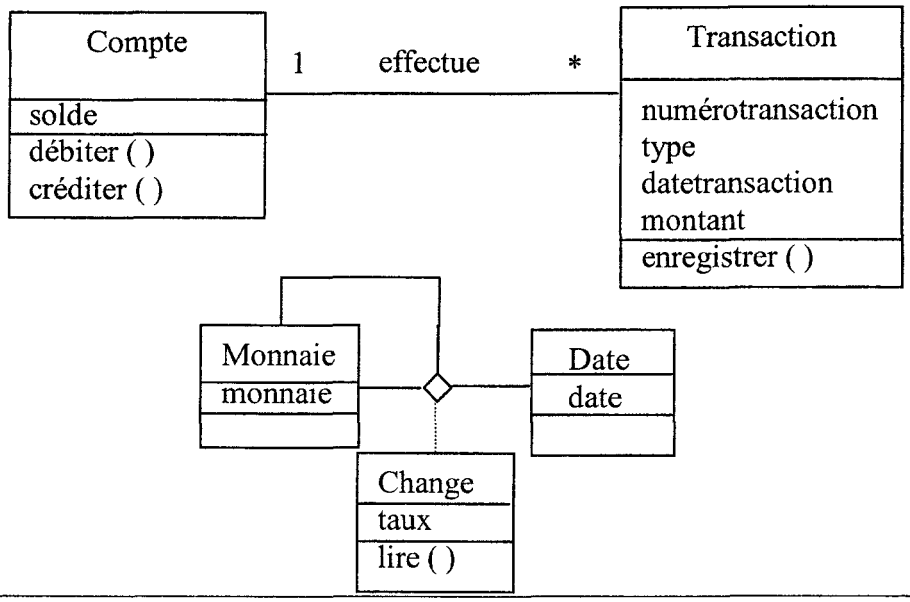


Domaine	Gestion bancaire
Sous domaine	Transactions bancaires
Cas d'utilisation	Change monnaie
Acteur	Opérateur
Type du Scénario	Principal

Scénario Pattern textuel
- L'opérateur saisit la date de la transaction, le montant, la monnaie d'origine et la monnaie du change. - Le système lit le taux de change, fait la conversion, affiche le montant en monnaie demandée, indique le type de la transaction, enregistre la transaction sur les comptes de change et imprime un reçu. - Le système met à jour le solde du compte caisse débiteur et le solde du compte caisse créditeur.

Scénario Pattern semi formel en langage LSP
BEGIN Change ; Principal ; Opérateur ; /SP interface, OpD, 1, saisir ; datetransaction, montant1, monnaie1, monnaie2 change, OpP, 1, lire, taux interface, OpP, 2, convertir; montant2 interface, OpP, 3, afficher; montant2 transaction, OpR, 1, enregistrer, numérotransaction, typetransaction, datetransaction, montant1 transaction, OpR, 2, enregistrer, numérotransaction, typetransaction, datetransaction, montant2 compte, OpR, 3, débiter, solde compte, OpR, 4, créditer, solde SP/ END

Pattern de conception relatif au scénario pattern principal



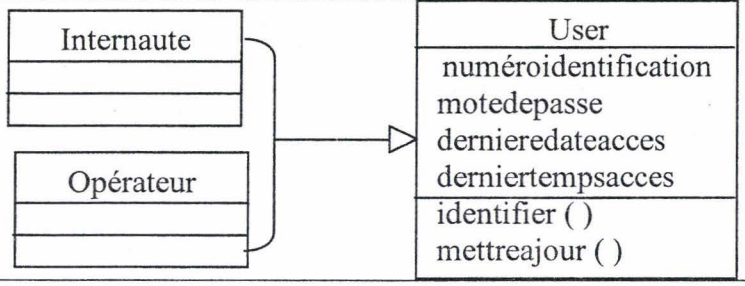
Domaine	Gestion bancaire
Sous domaine	Transactions bancaires
Cas d'utilisation	Identification
Acteur	Opérateur
Type du Scénario	Principal

Scénario Pattern textuel

- L'opérateur saisit le numéro d'identification et le mot de passe.
- Le système identifie l'opérateur, affiche son nom et affiche « Accès autorisé ».

Scénario Pattern semi formel en langage LSP

BEGIN
Identification;
Principal ;
Opérateur, Internaute ;
/SP
interface, OpD, 1, saisir ; numéroidentification, motdepasse
Opérateur, OpD, 2, vérifier, numéroidentification, motdepasse; boolean
SP/
END

Pattern de conception relatif au scénario pattern principal


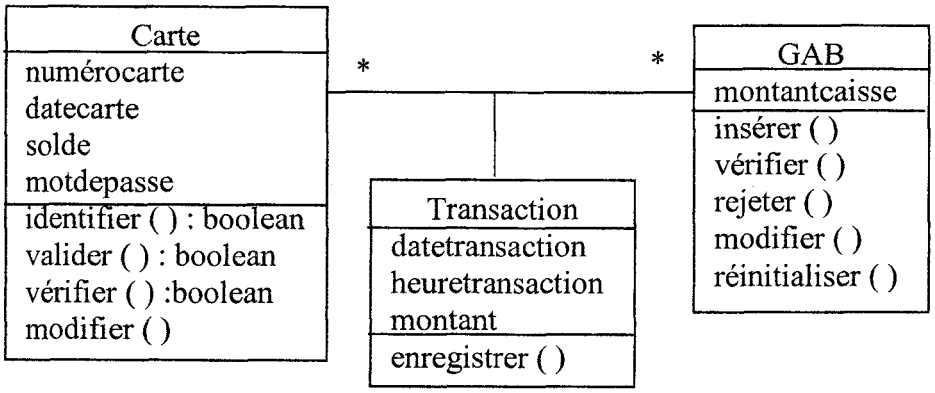
Domaine	Gestion bancaire
Sous domaine	Machine GAB
Cas d'utilisation	Retrait d'argent
Acteur	Client
Type du Scénario	Principal

Scénario Pattern textuel
- Le client insère la carte. - Le GAB valide la carte et affiche « entrer mot de passe ». - Le client saisit le mot de passe. - Le GAB identifie le client et affiche « Accès autorisé ». - Le client saisit le montant à retirer. - Le GAB vérifie la solvabilité de la caisse. - Le GAB vérifie la solvabilité du compte. - Le GAB rejette l'argent et la carte. - Le client récupère l'argent et la carte. - Le GAB met à jour le montant de la caisse. - Le GAB met à jour le solde du compte du client. - Le GAB enregistre la date, l'heure et le montant de la transaction - Le GAB se réinitialise.

Scénario Pattern semi formel en langage LSP
BEGIN Retrait ; Principal ; Client ; /SP GAB, OpD, 1, insérer ; numérocarte, datecarte carte, OpD, 2, valider, numérocarte, datecarte ; boolean GAB, OpD, 3, saisir ; motdepasse carte, OpD, 4, identifier, motdepasse GAB, OpP, 1, saisir ; montant GAB, OpP,2 , vérifier, montantcaisse ; boolean

carte, OpP, 3, vérifier, solde ; boolean
GAB, OpR, 1, rejeter ; boolean
GAB, OpP, 2, rejeter ; boolean
GAB, OpR, 3, modifier, montantcaisse
Carte, OpR, 4, modifier, solde
Transaction, OpR, 5, enregistrer, datetransaction, heuretransaction, montant
GAB, OpR, 6, Réinitialiser ; boolean
SP/
END

Pattern de conception relatif au scénario pattern principal



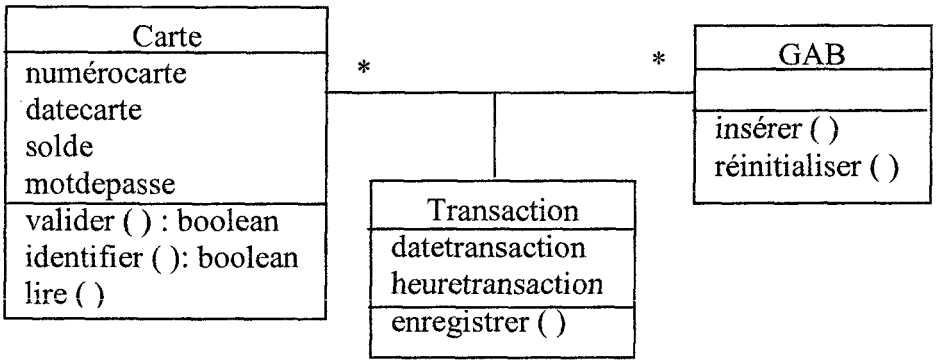
Domaine	Gestion bancaire
Sous domaine	Machine GAB
Cas d'utilisation	Consulter solde
Acteur	Client
Type du Scénario	Principal

Scénario Pattern textuel

- Le client insère la carte.
- Le GAB valide la carte et affiche « entrer mot de passe ».
- Le client saisit le mot de passe.
- Le GAB identifie le client et affiche « Accès autorisé ».
- Le client demande la consultation du solde.
- Le GAB affiche le solde.
- Le GAB rejette la carte.
- Le client récupère la carte.
- Le GAB enregistre la date et l'heure de la transaction
- Le GAB se réinitialise.

Scénario Pattern semi formel en langage LSP
BEGIN Consulter; Principal ; Client ; /SP GAB, OpD, 1, insérer ; numérocarte, datecarte carte, OpD, 2, valider, numérocarte, datecarte ; boolean GAB, OpD, 3, saisir ; motdepasse carte, OpD, 4, vérifier, motdepasse GAB, OpP, 1, saisir ; optionconsulter carte, OpP, 2, lire, solde GAB, OpP, 3, afficher; message GAB, OpP, 1, rejeter ; boolean Transaction, OpR, 2, enregistrer, datetransaction, heuretransaction GAB, OpR, 3, Réinitialiser ; boolean SP/ END

Pattern de conception relatif au scénario pattern principal

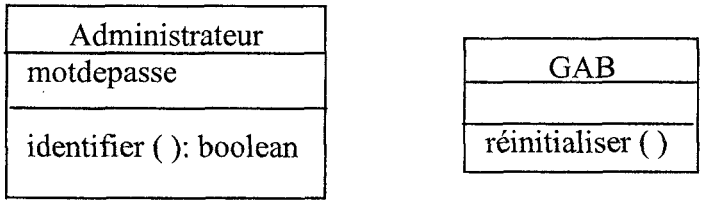


Domaine	Gestion bancaire
Sous domaine	Machine GAB
Cas d'utilisation	Initialiser GAB
Acteur	Administrateur
Type du Scénario	Principal

Scénario Pattern textuel
- L'administrateur saisit le mot de passe. - Le GAB identifie l'administrateur. - L'administrateur réinitialise le GAB.

Scénario Pattern semi formel en langage LSP
BEGIN Initialiser ; Principal ; Client ; /SP GAB, OpD, 1, saisir ; motdepasse administrateur, OpD, 2, identifier, mordepasse; boolean GAB, OpP, 1, Réinitialiser ; boolean SP/ END

Pattern de conception relatif au scénario pattern principal

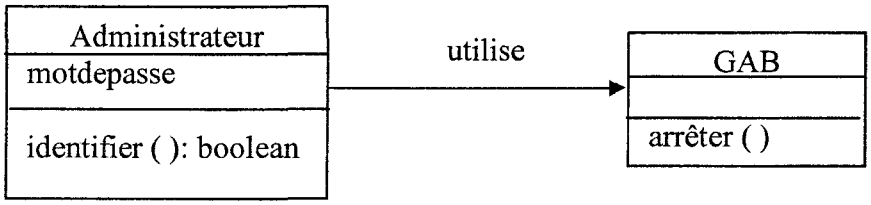


Domaine	Gestion bancaire
Sous domaine	Machine GAB
Cas d'utilisation	Arrêter GAB
Acteur	Administrateur
Type du Scénario	Principal

Scénario Pattern textuel
- L'administrateur saisit le mot de passe. - Le GAB identifie l'administrateur. - L'administrateur arrête le GAB.

Scénario Pattern semi formel en langage LSP
BEGIN Initialiser ; Principal ; Client ; /SP GAB, OpD, 1, saisir ; motdepasse administrateur, OpD, 2, identifier, mordepasse; boolean GAB, OpP, 1, arrêter; boolean SP/ END

Pattern de conception relatif au scénario pattern principal



5.3 Phases de développement du modèle d'analyse

5.3.1 Elaboration des besoins client

La première étape du développement dans l'EMI est la construction d'un pré modèle des besoins du client. Le développeur doit saisir deux formulaires qui décrivent les besoins du client afin de générer, à l'aide de l'outil UCDG 1.0, un modèle des besoins sous forme de diagramme de cas d'utilisation.

L'exemple ci-dessous présente deux digrammes de cas d'utilisation (figures 5.1 et 5.2) extraits des formulaires (tables 5.1 et 5.2). Ils modélisent les besoins du client pour gérer un système bancaire.

Domaine de L'application	Bancaire				
Sous domaine	**Cas d'utilisation**	**Pré-condition**	**Post-condition**	**Acteurs**	
Machine GAB	Retrait d'argent	Client identifié	Le client récupère l'argent. Le système enregistre la transaction.	Client	
	Consulter solde	Client identifié	Néant	Client	
	Initialiser GAB	Néant	GAB est dans l'état prêt	Administrateur	
	Arrêter GAB	Pas d'opérations en cours	Message d'arrêt bien affiché	Administrateur	
	Identification	Possède une carte de crédit	Client identifié	Client	

Sous domaine	Cas d'utilisation			Acteurs	
Transactions bancaires	Créditer compte	Guichetier identifié	Le guichetier prend l'argent. Le système enregistre la transaction.	Guichetier	
	Retrait d'argent	Guichetier identifié	Le guichetier donne l'argent au client. Le système enregistre la transaction.	Guichetier	
	Consulter compte	Guichetier identifié	Néant	Guichetier	
	Consulter compte en ligne	Client distant identifié	Néant	Client distant	
	Change de monnaie	Guichetier identifié	Le système enregistre la transaction de change.	Guichetier	
	Transfert de compte	Guichetier identifié	Le système enregistre la transaction de transfert.	Guichetier	
	Identification	Néant	Guichetier identifié	Guichetier	

Table 5.1 : Premier formulaire descriptif des besoins du client

Sous domaine	Cas d'utilisation	Cas d'utilisation en relation	Type	Contrainte
Machine GAB	Retrait d'argent	Identification	Inclut	
	Retrait d'argent	Identification	Inclut	
	Initialiser GAB	Identification	Inclut	
	Arrêter GAB	Identification	Inclut	
Transactions bancaires	Créditer compte	Identification	Inclut	
	Retrait d'argent	Identification	Inclut	
	Consulter compte	Créditer compte	Etend	
	Consulter compte	Retrait d'argent	Etend	Si compte est solvable
	Consulter compte	Identification	Inclut	
	Consulter compte en ligne	Consulter compte	Héritage	
	Change de monnaie	Identification	Inclut	
	Transfert de compte	Identification	Inclut	

Table 5.2: Deuxième formulaire descriptif des besoins client

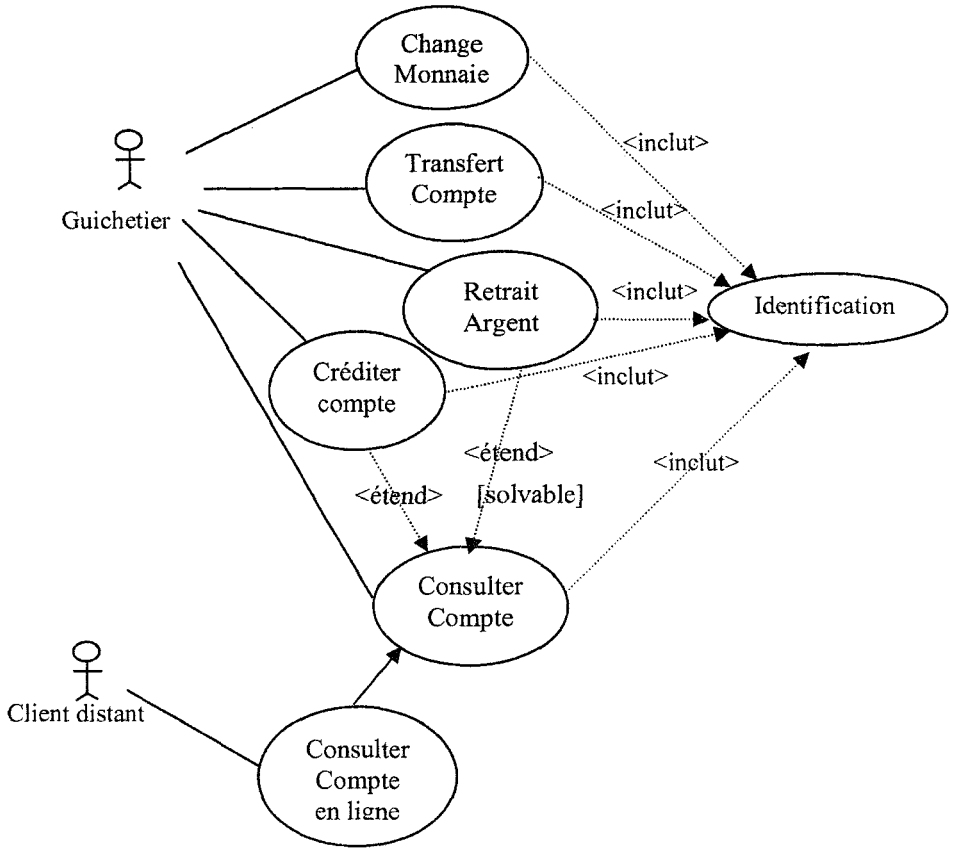


Figure 5.1: Diagramme de cas d'utilisation du sous domaine transactions bancaires

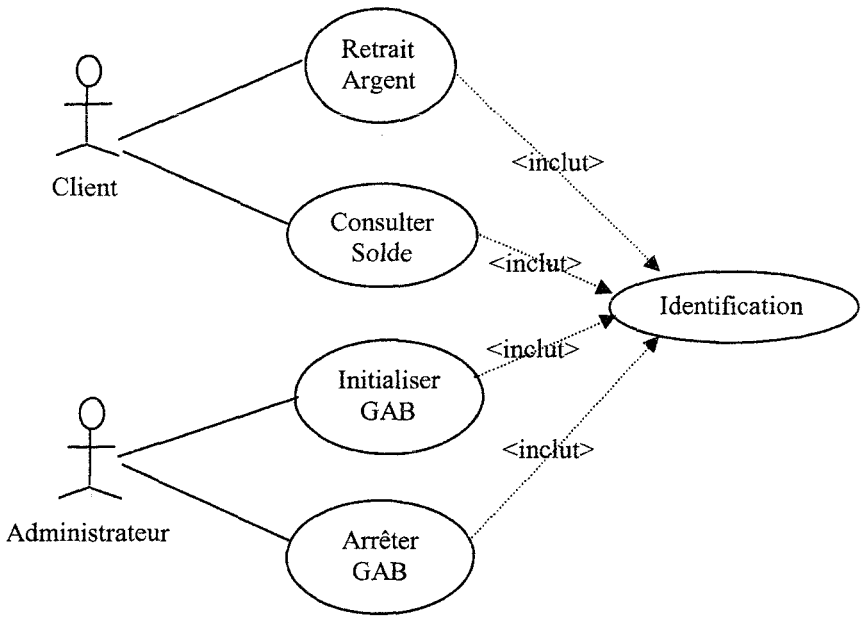


Figure 5.2: Diagramme de cas d'utilisation du sous domaine machine GAB

Chaque cas d'utilisation du diagramme possède un ensemble de descripteurs. Dans les deux tables 5.3 et 5.4, nous montrons les descripteurs évalués par chaque cas d'utilisation, et dans les tables 5.5 et 5.6 les valeurs récapitulatives des seuils dynamiques calculés en fonction du nombre de descripteurs.

Descripteurs	Change monnaie	Transfert compte	Identification	Retrait	créditer	Consulter compte	Consulter compte en ligne
Nom du cas d'utilisation	X	X	X	X	X	X	X
Post-conditions	X	X	X	X	X	-	-
Pré-conditions	X	X	-	X	X	X	X
Noms acteurs	X	X	X	X	X	X	X
Noms cas dépendants	X	X	-	X	X	X	X
Types cas dépendants	X	X	-	X	X	X	X
Contrainte	-	-	-	X	-	-	-
Nom du sous domaine	X	X	X	X	X	X	X
Nom du domaine	X	X	X	X	X	X	X

Table 5.3: Tableau de descripteurs des cas d'utilisation pour le sous domaine transactions

Descripteurs	Identification	Retrait	Consulter compte	Initialiser GAB	Arrêter GAB
Nom du cas d'utilisation	X	X	X	X	X
Post-conditions	X	X	-	-	-
Pré-conditions	-	X	X	X	X
Noms acteurs	X	X	X	X	X
Noms cas dépendants	-	X	X	X	X
Types cas dépendants	-	X	X	X	X
Contrainte	-	-	-	-	-
Nom du sous domaine	X	X	X	X	X
Nom du domaine	X	X	X	X	X

Table 5.4 : Tableau de descripteurs des cas d'utilisation pour le sous domaine machine GAB

Descripteurs	Change monnaie	Transfert compte	Identification	Retrait	Créditer	Consulter compte	Consulter compte en ligne
Nom du cas d'utilisation	9	9	9	9	9	9	9
Post-conditions	8	8	8	8	8	-	-
Pré-conditions	7	7	-	7	7	7	7
Noms acteurs	6	6	6	6	6	6	6
Noms cas dépendants	5	5	-	5	5	5	5
Types cas dépendants	4	4	-	4	4	4	4
Contrainte	-	-	-	3	-	-	-
Nom du sous domaine	2	2	2	2	2	2	2
Nom du domaine	1	1	1	1	1	1	1
Seuil	0.94	0.94	0.58	1	0.94	0.76	0.76

Table 5.5: Seuils dynamiques calculés en fonction du nombre des descripteurs

Descripteurs	Identification	Retrait	Consulter compte	Initialiser GAB	Arrêter GAB
Nom du cas d'utilisation	9	9	9	9	9
Post-conditions	8	8	-	-	-
Pré-conditions	-	7	7	7	7
Noms acteurs	6	6	6	6	6
Noms cas dépendants	-	5	5	5	5
Types cas dépendants	-	4	4	4	4
Contrainte	-	-	-	-	-
Nom du sous domaine	2	2	2	2	2
Nom du domaine	1	1	1	1	1
Seuil	0.58	0.94	0.76	0.76	0.76

Table 5.6: Seuils dynamiques calculés en fonction du nombre des descripteurs

5.3.2 Génération du modèle des besoins à l'aide de l'outil UCDG 1.0

La deuxième étape de développent est la génération du modèle des besoins détaillé enrichi par un ensemble de scénarios patterns (en appliquant les algorithmes vus dans le chapitre 3). Le UCDG 1.0, utilise les techniques de test de similarité pour en trouver les cas d'utilisation les plus proches que ceux demandés par le client et décrits par son pré modèle des besoins. Le résultat fournit est un nouveau diagramme de cas d'utilisation enrichi par des scénarios patterns. Nous rappelons que l'outil UCDG 1.0 utilise le dictionnaire de connaissances métier lorsque les mots ou les termes utilises par le développeur ne sont pas les mêmes que ceux trouvés dans la base de cas. Donc, il doit trouver leurs synonymes.

Dans le présent exemple, le développeur a utilisé deux termes différents que ceux trouvés dans la base de cas. La consultation du dictionnaire nous a mené à identifier les synonymes suivants :

- synonyme du « Guichetier » est « opérateur »
- synonyme du « Créditer» est « dépôt »

L'ensemble de scénarios patterns, renvoyé par L'UCDG 1.0, peut être édité et consulté par le développeur afin de faire d'éventuelles modifications.

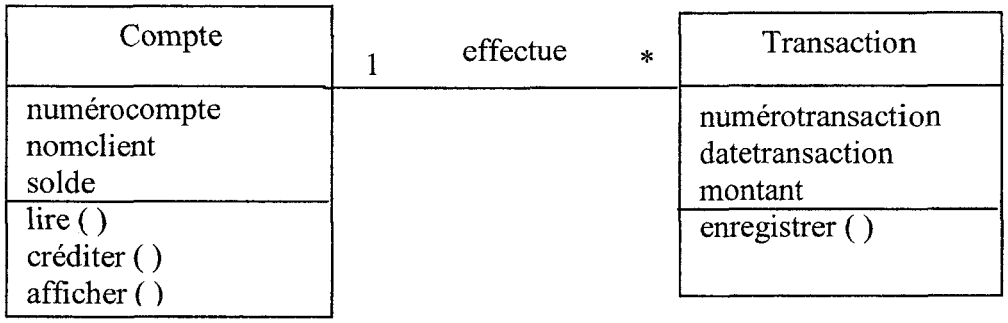
5.3.3 Génération du modèle d'analyse à l'aide de l'outil CDG 1.0

Une fois le modèle des besoins fourni par l'EMI sera prêt, un deuxième modèle (diagramme de classes), fondé sur le premier, est généré. Le diagramme de classes représente la solution de conception du modèle des besoins généré à l'étape précédente.

L'outil CDG 1.0 est responsable de cette opération, il suit une démarche de composition itérative pour construire le diagramme de classes. A chaque scénario pattern de la base de cas est associé un pattern de conception (fragment de diagramme de classes). A partir de l'ensemble de ces fragments, le CDG 1.0 construit par composition itérative le diagramme de classes complet.

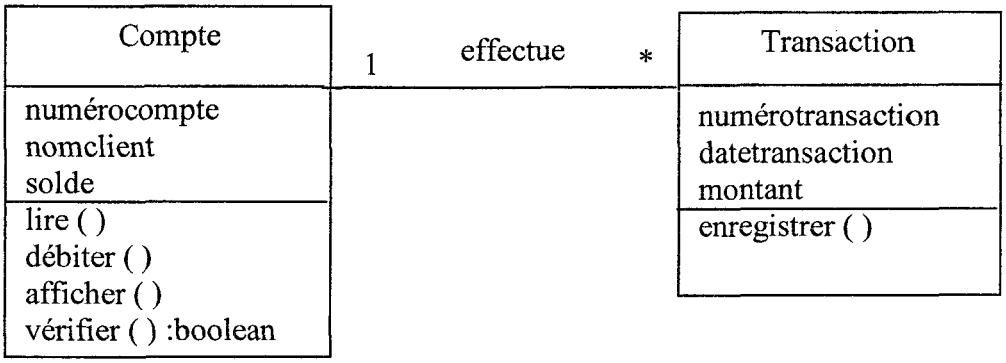
Cas d'utilisation : Dépôt d'argent
Itération 1: scénario pattern principal / pattern de conception

Pattern de conception relatif au scénario pattern principal



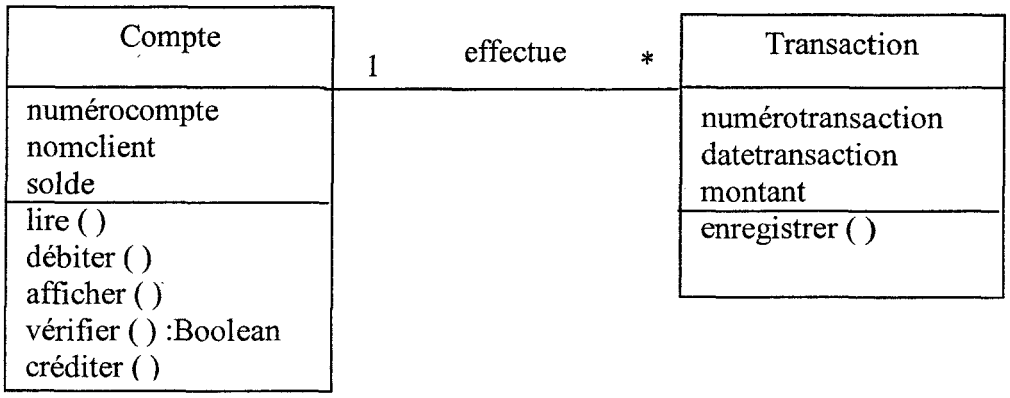
Cas d'utilisation : Retrait d'argent
Itération 2: scénario pattern principal / pattern de conception

Pattern de conception relatif au scénario pattern principal



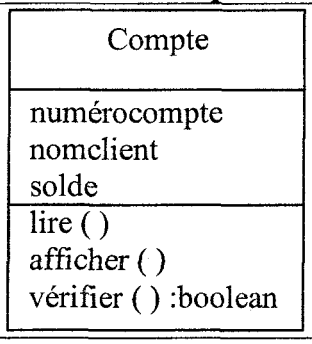
Composition itération 1 et itération 2

Pattern de conception Pattern de conception résultant



Cas d'utilisation : Retrait d'argent
Itération 3: scénario pattern alternatif / pattern de conception

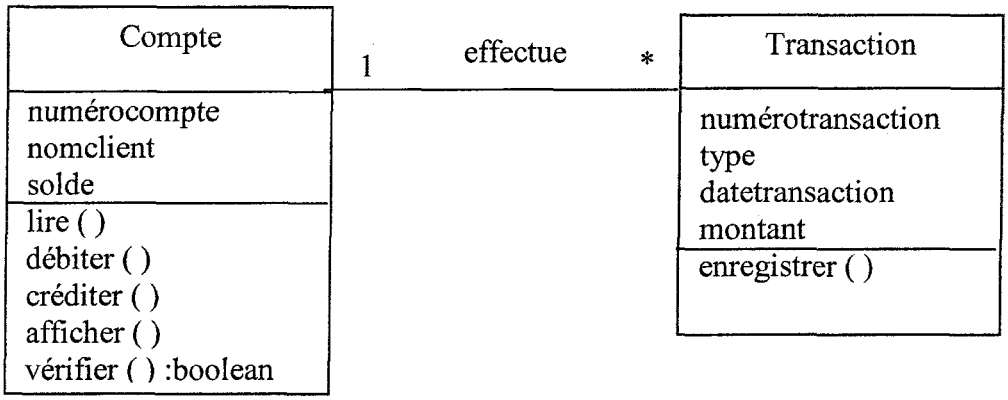
Pattern de conception relatif au scénario pattern alternatif



Dans cette itération pas de modification au pattern de conception précèdent

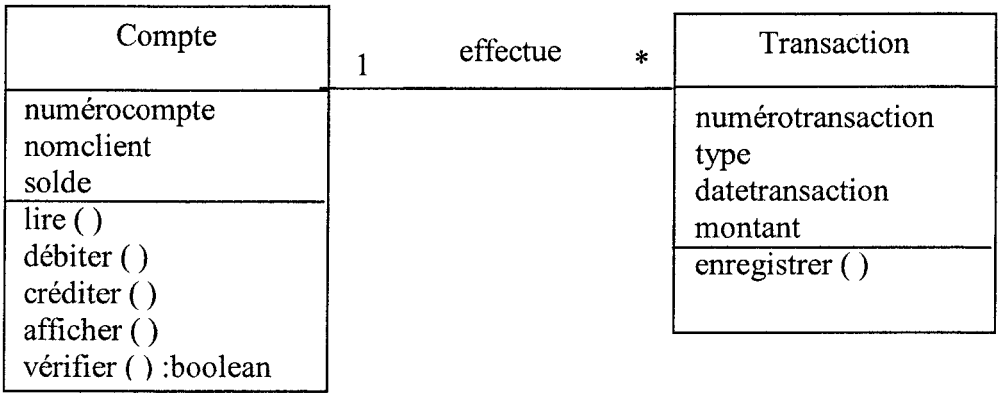
Cas d'utilisation : Transfert compte
Itération 4: scénario pattern principal / pattern de conception

Pattern de conception relatif au scénario pattern principal



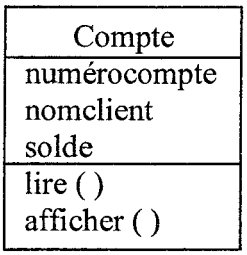
Composition itération 4 avec le pattern de conception résultant de l'étape précédente.

Pattern de conception résultant



Cas d'utilisation : Consulter compte
Itération 5: scénario pattern principal / pattern de conception

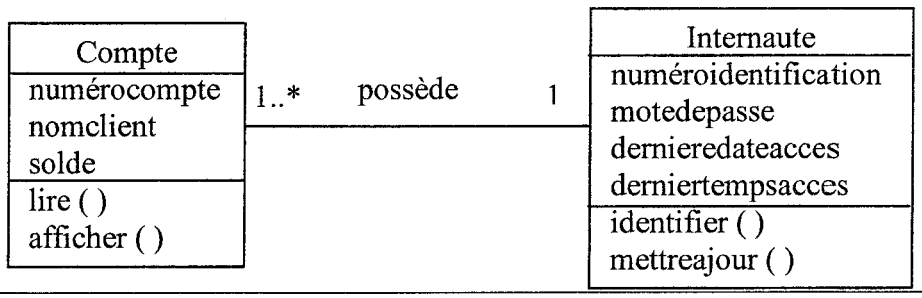
Pattern de conception relatif au scénario pattern principal



Dans cette itération pas de modification au pattern de conception précèdent

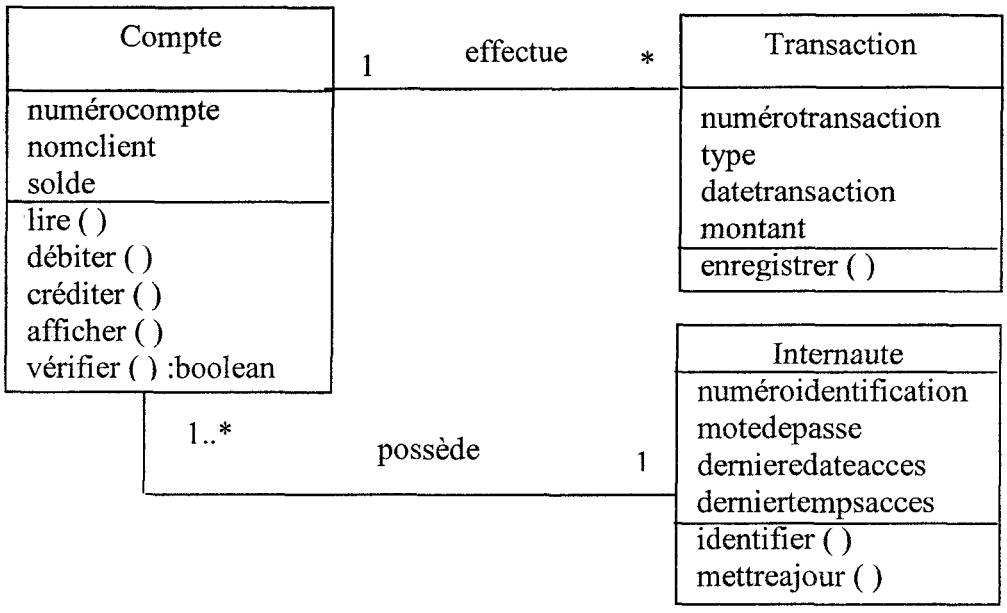
Cas d'utilisation : Consulter compte en ligne
Itération 6: scénario pattern principal / pattern de conception

Pattern de conception relatif au scénario pattern principal



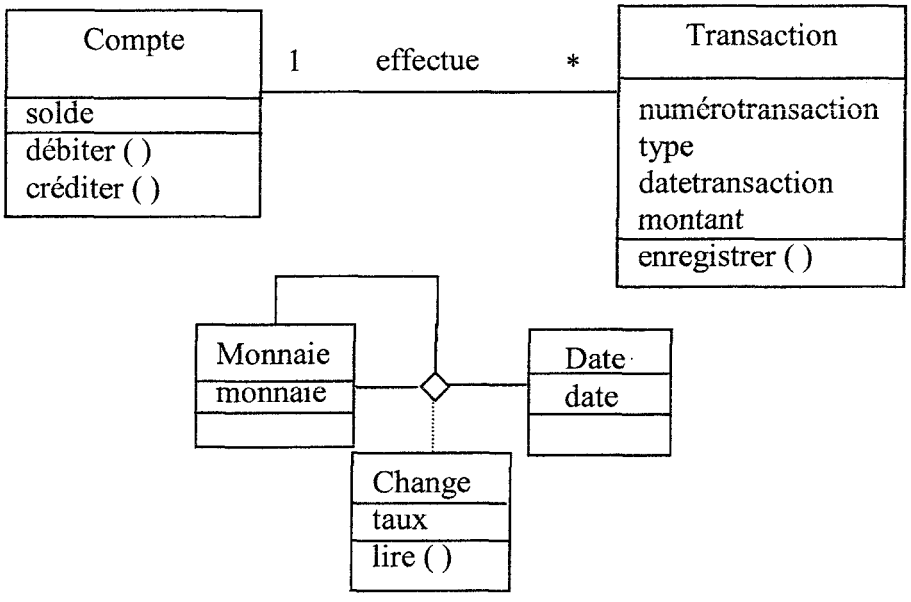
Composition itération 6 avec le pattern de conception résultant de l'étape précédente.

Pattern de conception résultant



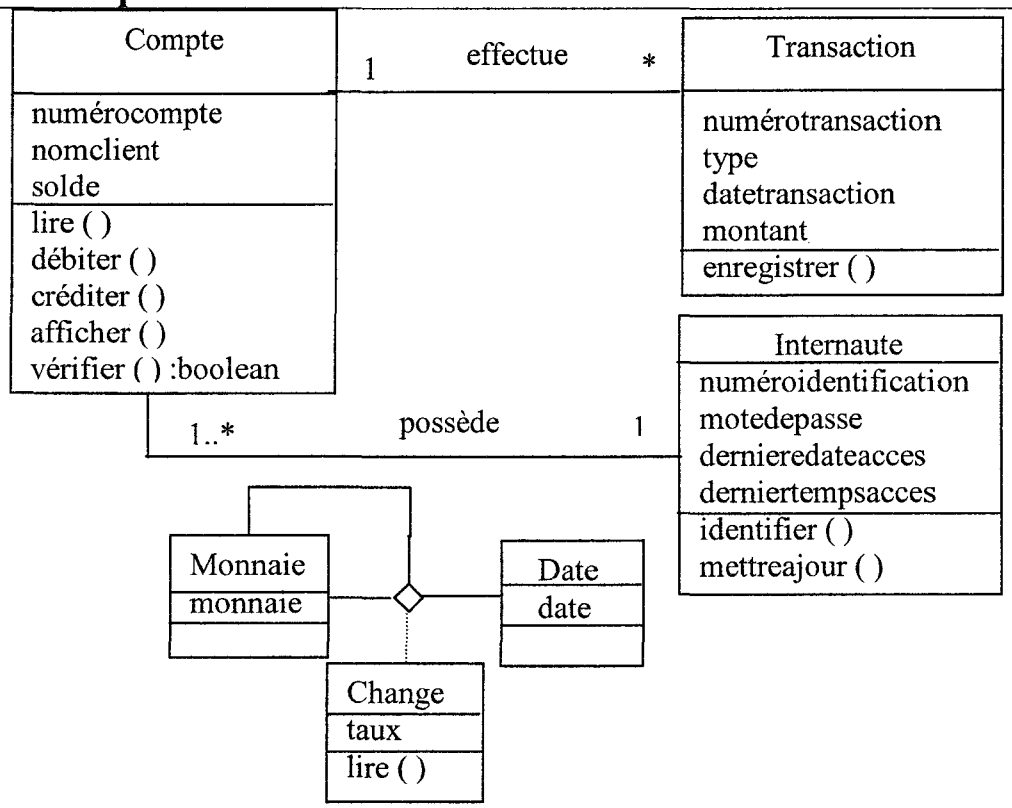
Cas d'utilisation : Change monnaie
Itération 7: scénario pattern principal / pattern de conception

Pattern de conception relatif au scénario pattern principal



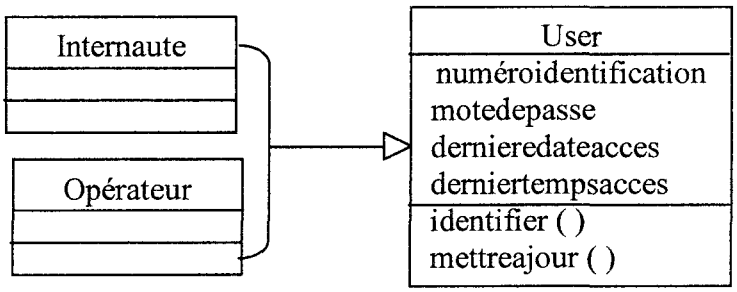
Composition itération 7 avec le pattern de conception résultant de l'étape précédente.

Pattern de conception résultant



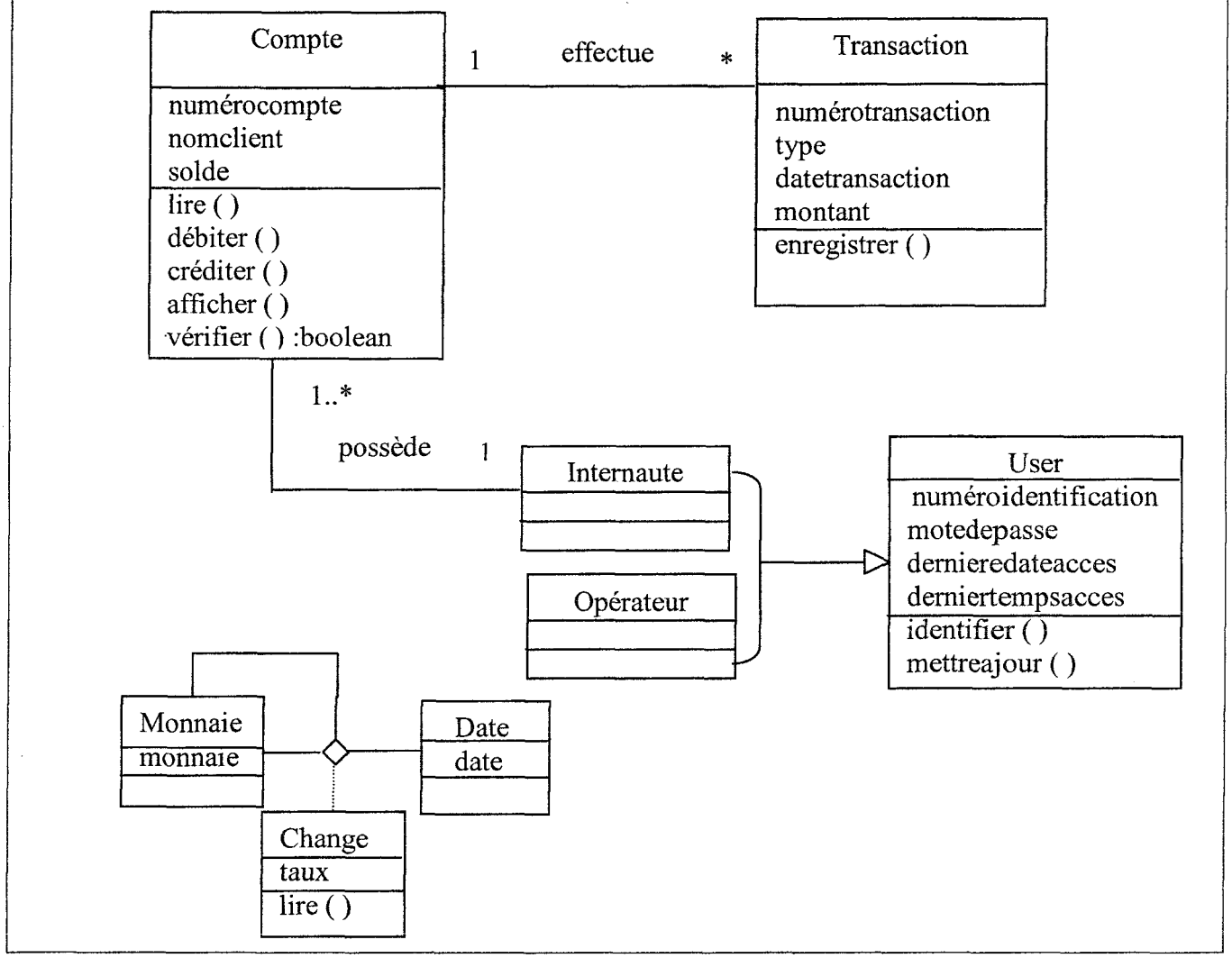
Cas d'utilisation : Identification
Itération 8: scénario pattern principal / pattern de conception

Pattern de conception relatif au scénario pattern principal



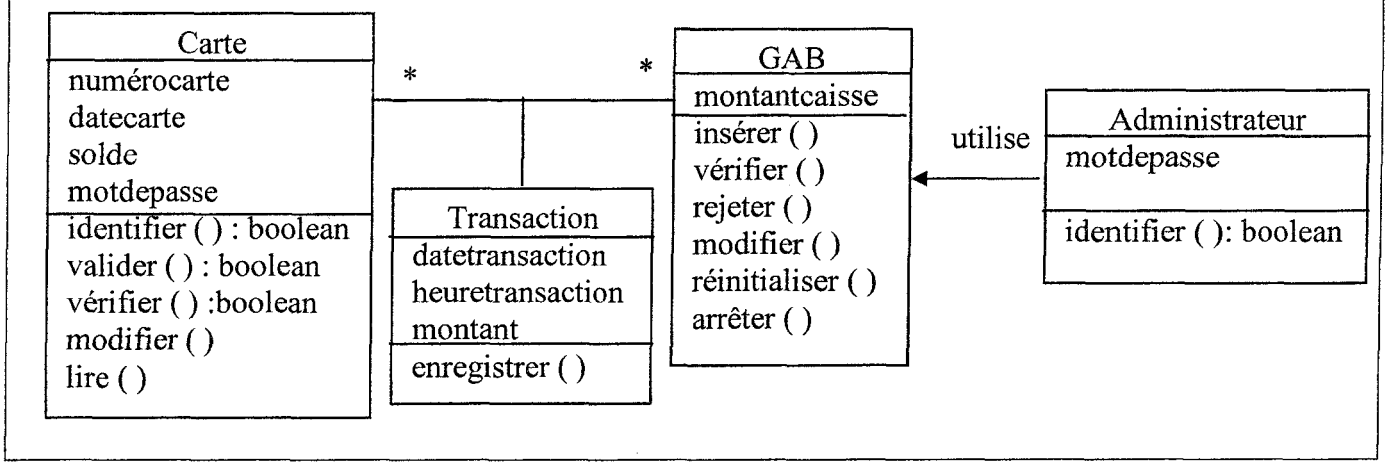
Composition itération 8 avec le pattern de conception résultant de l'étape précédente.

Pattern de conception résultant



Nous reprenons la même chose pour le sous domaine machine GAB, l'opération de composition des patterns de conception, après toutes les itérations, nous donne le pattern de conception suivant :

Pattern de conception résultant



5.3.4 Phase d'adaptation du modèle d'analyse

La troisième étape du développement du modèle d'analyse concerne la phase d'adaptation. Le développeur peut introduire des modifications au diagramme de classes à l'aide de l'outil CDE 1.0. Cette phase est complètement sous le contrôle du développeur. Il peut, à l'aide de l'éditeur, modifier le diagramme de classes où l'exporter vers d'autres éditeurs.

5.4 Apports de la méthodologie

La principale originalité de la méthodologie présentée est de faire contribuer le RàPC et les scénarios patterns au développement des modèles d'analyse pour les applications informatiques. La démarche, par son caractère ascendant depuis la construction des bases de cas orientés sujet jusqu'au le développement du modèle d'analyse place résolument l'expert métier, l'expert de conception, le développeur et le client au centre des préoccupations.

Le développement dans l'EMI débute par le recueil des données de l'expert métier. Ceci met à contribuer des méthodes de recueil et d'analyse issues de travaux en psychologie cognitive. Le cycle de développement est finalisé par le modèle de conception d'analyse. Il repose sur deux principes de base, d'une part, la réutilisabilité des connaissances métier (scénarios patterns) et la réutilisabilité des patterns de conception et, d'autre part, l'intégration des techniques de raisonnement à partir de cas (RàPC). Ces principes améliorent le cycle de développement en fonction de la qualité, du coût et du temps.

Le modèle d'analyse final permet une validation intermédiaire, par l'expert du domaine d'application, des connaissances qui seront représentées au sein du système de résolution de problèmes à partir de cas. Il constitue un guide pour sa conception en répondant aux questions : « qu'est-ce qu'un cas ? », « quels sont ses descripteurs ? » et « comment sont-ils exploités ? ». Enfin, le modèle d'analyse forme un riche capital de connaissances dont l'intérêt dépasse les spécifications cognitives du futur système.

Pour sa part, le RàPC apporte une réponse concrète et efficace au problème de la maîtrise du cycle de retour d'expérience, si crucial pour les entreprises.

En conclusion, l'approche va dans le sens de l'évolution actuelle des travaux en ingénierie des connaissances. On acquiert, certes pour construire des bases de cas comme des systèmes de résolution de problèmes, mais avant tout, et de plus en plus, pour capitaliser et valoriser les connaissances d'une entreprise.

5.5 Limites de la méthodologie

Notre approche méthodologique compte beaucoup sur la construction des bases de cas orientés sujet. Pour mener à bien cet objectif il faudra avoir un recours important vers la capitalisation des connaissances métiers.

A ce jour, plusieurs méthodes offrant la possibilité de capitaliser les connaissances ont été élaborées tels que les mémoires projet [Djaiz, 2006], le modèle KnoVA-Sigma [Serrafero, 2004] ou la méthode GAMETH [Grundstein, 2002]. Ces outils et méthodes sont très utiles, mais sans doute encore trop peu nombreuses pour pouvoir prétendre à une reconception basée sur les connaissances expertes. Il existe d'autres méthodes non encore exploitées en « knowledge based engineering », mais qui sont susceptibles d'encapsuler de nombreuses connaissances expertes. Il s'agit, par exemple, des méthodes AMDEC (Analyse des Modes de Défaillance, de leur Effets et de leurs criticité) ou QFD (Quality Function Deployment) [Chan et Wu 02], qui contribuent, soit à fiabiliser le produit ou le processus étudié, soit à conserver une traçabilité des choix de conception en lien avec les besoins du client. Ces méthodes sont régies par une ou plusieurs connaissances expertes non formalisées dans les outils actuels.

Malgré le fait que les méthodes et les outils d'aujourd'hui ne supportent pas la capitalisation des connaissances métier dans tous les domaines, la gestion de l'expérience reste une nécessité accrue et stratégique pour les entreprises [Delange et Vogin 94].

La question qui se pose au niveau des connaissances concerne le degré de couverture du domaine d'application par les cas archivés. Au-delà des tests d'évaluation, il est difficile de garantir avec certitude l'entière représentativité des connaissances présentes en mémoire, surtout dans un domaine d'application aussi évolutif. Il en résulte que la maintenance et l'enrichissement de la base de cas sont vitaux pour assurer la meilleure adéquation de l'expertise.

Comme pour tout système de type RàPC, l'EMI se heurte à la difficulté de prendre en compte le temps dans la représentation et la gestion des informations et dans la formalisation du raisonnement. Cet aspect temporel constitue l'une de nos principales préoccupations pour l'évolution future de l'EMI.

5.6 Perspective d'intégration et d'extension de l'EMI

Nous abordons dans cette dernière section les perspectives de recherche et de développement qui s'ouvrent à la lumière des premiers résultats de l'EMI. Celles-ci

s'inscrivent dans trois directions. La première perspective concerne les améliorations techniques à apporter à l'EMI de façon à en accroître les performances. La seconde aborde le problème d'un fonctionnement en temps réel de l'EMI. Enfin, la dernière perspective s'intéresse à l'extension de la méthodologie à d'autres domaines d'applications.

5.6.1 Quelques améliorations de l'EMI

Plusieurs améliorations peuvent contribuer à accroître les performances de l'EMI. Ces améliorations doivent se porter sur les points essentiels suivants :

- La première amélioration à apporter au système, qui tient plus de l'organisation que du développement, est l'enrichissement des bases de cas orientés sujet. Ceci peut se faire à l'aide de la réutilisation des scénarios patterns et des patterns de conception dans des divers domaines tels que bancaire, médical, industriel, enseignement, etc.

- La seconde amélioration à apporter aussi au système est l'implantation des algorithmes qui accélèrent, pendant la phase de remémoration, la recherche dans la structure de l'arbre d'index.

- La troisième amélioration est d'enrichir le modèle d'analyse par d'autres diagrammes d' UML. Jusqu'à maintenant l'EMI génère un diagramme de classes à partir des patterns de conception qui se trouvent dans la base de cas. Les diagrammes ciblés seront dynamiques tels que les diagrammes de séquence, d'états, d'activités, etc.

Nous recommandons aussi des améliorations aux éditeurs utilisés par l'EMI. Ces éditeurs doivent cependant permettre d'exporter et d'importer des diagrammes UML entre les différents éditeurs.

5.6.2 Extension de la méthodologie à d'autres domaines

La méthodologie présentée dans ce mémoire repose sur la construction de la base de cas par l'approche de scénario pattern et de pattern de conception et sur l'utilisation du RàPC afin de réaliser un système de génération automatique de modèle d'analyse pour les applications informatiques.

La méthodologie des exemples que nous avons traités dans ce mémoire relève plus du domaine de gestion que d'autres domaines d'application. En réalité, tous les domaines sont visés par cette méthodologie et surtout les domaines qui nécessitent une réutilisation des expériences passées. Les domaines d'application qui sont,

habituellement, développés à l'aide des modèles de prototypage sont des exemples pratiques pour notre approche.

Ainsi, tous les domaines d'application qui peuvent être décrits à l'aide des scénarios métier sont des exemples opérationnels et pratiques pour notre méthodologie. Les applications web sont parmi ces domaines d'application et elles sont des exemples pour tester la robustesse de notre méthodologie.

Conclusion générale

Bilan et apports du travail

Le travail de recherche présenté dans ce mémoire concerne la contribution des scénarios patterns et du RàPC au développement du modèle d'analyse pour les applications informatiques. L'objectif de ce travail est d'optimiser la tâche amont de modélisation conceptuelle du développeur en lui offrant un outil d'aide à la génération automatique du modèle d'analyse statique de l'application au format UML. Dans ce but, un Environnement de Modélisation Intelligent (EMI), fondé sur la réutilisation de connaissances métier (« *scénarios patterns* ») et de leurs patterns d'analyse (« *patterns de conception* »), a été conçu, réalisé et validé. Ces deux éléments forment ensemble la structure de la base de cas d'un système RàPC. L'exemple applicatif du travail s'appuie sur la réutilisabilité des scénarios patterns et des patterns de conception du problème de gestion bancaire. Il se concrétise par la construction de la base de cas (problèmes et solutions), des outils de génération automatique des modèles des besoins et d'analyse (diagramme de cas d'utilisation et diagramme de classes) mettant en œuvre les mécanismes du raisonnement à partir de cas (RàPC). Ce travail se veut prospectif et, en ce sens, les solutions avancées s'étendent au-delà du domaine d'application particulier.

La réutilisabilité des connaissances métier au niveau d'analyse des besoins est l'un des principaux résultats dans ce mémoire. Cette réutilisabilité permet de réduire le rôle de l'ingénierie des besoins dans le cycle de développement des applications logicielles. L'introduction de la notion des scénarios patterns est aussi un apport d'innovation dans le processus d'ingénierie des besoins. Les connaissances métier sont modélisées à l'aide des scénarios patterns formalisés par un langage spécifique appelé le langage de description des scénarios patterns (LSP).

Un deuxième principal résultat dans ce mémoire est l'intégration de l'approche des systèmes RàPC dans le cycle de modélisation conceptuelle. Le concepteur utilise la base de cas, comportant les « scénarios patterns » et les « patterns de conception », pour produire son modèle de conception relatif aux besoins du client. Dans ce sens, les techniques utilisées par les systèmes RàPC sont exploitées par notre approche de modélisation.

Perspectives de recherche

Au terme de ce travail, trois axes de recherche particulièrement intéressants sont proposés à l'exploration.

Dans une première direction, il semble intéressant d'approfondir la façon dont le RàPC peut être combiné à d'autres formalismes de représentation des connaissances et à d'autres modes de raisonnement pour accroître l'efficacité et les performances du système global. L'intégration du RàPC au noyau de développement des modèles d'analyse par réutilisabilité des connaissances de l'expertise met en œuvre le développement des techniques de description, de remémoration et d'adaptation spécialisés à résoudre ce genre de problèmes.

Un autre axe de recherche concerne la définition d'une bibliothèque de modèles « génériques » de résolution de problèmes relatifs aux modèles d'analyse conceptuels et leurs divers diagrammes UML tels que diagramme de séquence, d'états, d'activités, etc.... En effet, de par les principes d'organisation des connaissances adoptées par le modèle conceptuel de l'expertise et dans les limites de leur niveau de généricité, les modèles de résolution de problèmes obtenus au cours de ce travail peuvent constituer les premiers éléments de cette bibliothèque. Toujours en relation avec l'acquisition structurée des connaissances métier et de façon à améliorer la méthodologie de développement de systèmes RàPC dans l'EMI, une réflexion sur les méthodes de recueil des besoins et les formalismes de modélisation des connaissances métier à l'aide des scénarios patterns nous parait également devoir être évoluée.

Enfin, une dernière voie dans laquelle nous souhaitons nous engager à l'avenir concerne le développement des patterns de conception qui décrivent une application informatique dans les vues dynamiques. En effet, les bases de cas sources ne contiennent pas seulement des patterns de conception comme des fragments de diagrammes de classes mais aussi des fragments d'autres diagrammes dynamiques d'UML. Nous envisageons aussi de permettre à notre EMI d'exporter ses modèles fournis à d'autres systèmes pour profiter des outils de transformation de modèles existants sur d'autres plates-formes tel que le MDA (Model Driven Architecture).

ANNEXE A

Glossaire de la Banque Société Générale de France

Dictionnaire des connaissances métier

Mot ou terme	Explication
Abattement	Réduction de la base d'imposition, la taxation s'opérant au-delà du montant de l'abattement.
Abonnement	Convention passée avec une banque à un prix déterminé et périodique pour la mise à disposition régulière ou pour l'usage habituel de services.
Abonnement à des services de banque à distance	Frais perçus par la banque pour un abonnement à son offre de services de banque à distance (Internet, téléphone fixe, téléphone mobile, SMS...).
Actif net	Ensemble des actifs d'une société (ou d'un OPCVM) après déduction de l'intégralité de ses dettes.
Actif successoral	Ensemble des biens matériels et immatériels appartenant au défunt et entrant, lors de l'ouverture de la succession de ce dernier, dans la masse des biens à partager.
Action	Titre représentant la propriété d'une fraction du capital d'une société, le plus souvent négociable en Bourse.
Actionnaire	Détenteur d'une action. L'actionnaire est copropriétaire de l'entreprise, ce qui lui confère généralement les droits et devoirs d'un associé : droit à l'information, droit de vote, droit aux dividendes...
Agence	Lieu d'accueil de la clientèle d'une banque.
Agios	Intérêts débiteurs perçus par la banque, généralement à l'occasion d'un découvert en compte, calculés en fonction de la somme, de la durée et du taux d'intérêt du découvert et auxquels s'ajoutent les frais et commissions.
Aller retour	Terme de bourse désignant l'achat puis la revente rapprochée d'un titre (dans la perspective de réaliser un bénéfice).
Allocation d'actifs	Répartition d'un investissement entre les différentes classes d'actifs, c'est-à-dire entre les différents marchés.
Amortissement	Remboursement prévu en une ou plusieurs fois d'un emprunt. Selon le type d'emprunt, il peut être étalé dans le temps ou effectué en une seule fois en fin de contrat.

Arbitrage	Opération consistant à vendre une valeur mobilière, un produit financier, une devise... pour en acheter un(e) autre (dans la perspective d'obtenir un meilleur rendement ou de réaliser des plus-values).
Assuré	Personne physique sur la tête de qui repose le risque (décès, survie, maladie...).
Autorisation de découvert	Accord donné par la banque permettant de bénéficier d'un découvert d'un montant maximum déterminé et remboursable selon des modalités convenues d'avance, notamment dans la convention de compte de dépôt ou dans un contrat.
Autorisation de prélèvement	Autorisation donnée par le client à sa banque de payer les prélèvements qui seront présentés par une société ou des créanciers désignés par l'autorisation.
Banque	Établissement autorisé par la loi à assurer des opérations de banque c'est-à-dire la réception de fonds du public, les opérations de crédit, ainsi que la mise à la disposition de la clientèle ou la gestion de moyens de paiement. Le terme « établissement de crédit » ou « caisse » est également utilisé.
Bénéficiaire	Toute personne de votre choix (conjoint, enfant, parent, ami...) que vous aurez désignée pour bénéficier des avantages d'un contrat (notamment dans le cadre de l'assurance vie).
Bourse	Marché public organisé où les organismes qui émettent des titres (entreprises, administrations, Etat...) peuvent se procurer des fonds, et où les investisseurs (particuliers, institutionnels, entreprises...) peuvent acheter ou vendre ces titres.
Capitalisation	Mécanisme financier consistant à ajouter les intérêts au capital afin de les faire eux-mêmes fructifier. Les intérêts du capital placé ne sont pas distribués au bénéficiaire mais ajoutés à ce capital, venant l'augmenter d'autant. Le nouveau capital ainsi constitué produit alors des intérêts au même taux que le capital initial. Concernant les OPCVM : Choix opéré par les OPCVM de ne pas distribuer les revenus encaissés de leur portefeuille. Les revenus capitalisés ne sont pas imposables à l'Impôt sur le Revenu (IR) et contribuent à l'augmentation de la valeur liquidative des actions de SICAV et des parts de FCP qui restent soumises à la taxation sur les plus-values en cas de cessions supérieures au seuil : toutefois, les avoirs fiscaux attachés aux dividendes capitalisés ne sont pas récupérés. *S'oppose à distribution.*
Carte bancaire	Moyen de paiement prenant la forme d'une carte émise par un établissement de crédit et permettant à son titulaire, conformément au contrat passé avec sa banque, d'effectuer des paiements et /ou des retraits. Des services connexes peuvent y être associés (assurance, assistance).
Carte de crédit	Carte de paiement permettant à son titulaire de régler des achats et/ou d'effectuer des retraits au moyen d'un crédit préalablement et contractuellement défini.
Carte de retrait	Carte délivrée par la banque permettant d'effectuer exclusivement des retraits de billets dans des automates bancaires (DAB/GAB). Son utilisation peut être limitée ou non à un seul guichet bancaire, à une seule banque ou à une seule agence.
Change manuel	Opération qui consiste à convertir des billets de banque d'une monnaie dans une autre monnaie. Cette opération donne généralement lieu à la perception d'une commission de change.
Chèque	Moyen de paiement normalisé avec lequel le titulaire (tireur) d'un compte donne l'ordre à son banquier (tiré) de payer au bénéficiaire du chèque la somme inscrite sur celui-ci. La provision doit être disponible lors de l'émission du chèque et maintenue jusqu'à sa présentation.
Chèque de banque	Chèque émis par une banque à la demande du client, et dont le montant, immédiatement débité du compte de dépôt du client, est ainsi garanti. Les chèques de banque sont souvent exigés pour le règlement d'achats importants.
Chèque sans	Chèque émis sur un compte de dépôt dont le solde disponible ou le découvert

provision	autorisé est insuffisant pour régler le montant du chèque. L'émetteur se voit interdit d'émettre des chèques jusqu'à ce qu'il régularise sa situation. L'émission d'un chèque sans provision entraîne des frais bancaires et éventuellement des pénalités à payer au Trésor Public (si la régularisation n'a pas lieu dans un délai de 2 mois après l'interdiction). La provision doit être disponible dès l'émission du chèque et maintenue jusqu'à sa présentation.
Chéquier	Carnet comportant généralement 25 à 30 formules de chèques (ou « vignettes »). Certaines banques donnent le choix du format du carnet.
Commission	Somme perçue par une banque en rémunération d'un service fourni à son client.
Compte de dépôt	Compte bancaire ordinaire (ou compte courant) utilisé pour gérer quotidiennement son argent. C'est sur ce compte qu'un client dispose en général d'une carte bancaire et d'un chéquier. Le compte doit être créditeur, sauf accord avec la banque.
Compte joint	Compte de dépôt ouvert au nom de deux (ou plusieurs) personnes qui ont chacune le droit de disposer seule de l'avoir du compte. Dans le cas d'un compte joint entre époux, le compte n'est pas bloqué en cas de décès d'un des co-titulaires.
Créancier	Personne physique ou morale à qui est due une somme d'argent.
Crédit (écriture de crédit)	Opération comptable qui augmente le solde du compte, par exemple à la suite d'un virement reçu, d'un dépôt d'espèce, ou d'une remise de chèque.
Crédit (opération de crédit)	Opération par laquelle un établissement de crédit met ou promet de mettre à disposition d'un client une somme d'argent moyennant intérêts et frais, pour une durée déterminée ou indéterminée (lorsque le crédit est dit gratuit, les frais et les intérêts sont nuls).
Crédit affecté	Crédit par lequel le contrat de prêt définit précisément l'objet financé. Le déblocage des fonds est étroitement lié à la réalisation de l'opération envisagée.
Crédit d'impôt	Avantage fiscal obtenu dans certains cas et venant en déduction de l'Impôt sur le Revenu. En cas d'imposition insuffisante, l'excédent de crédit d'impôt est reversé au contribuable.
Crédit renouvelable	Opération par laquelle un établissement de crédit met ou promet de mettre à la disposition d'un client une somme d'argent moyennant intérêts et frais sur la partie utilisée. Cette somme est réutilisable au fur et à mesure des remboursements en capital. Elle peut être remboursée à tout moment, en totalité ou en partie.
DAB	Distributeur Automatique de Billets. Appareil qui permet de retirer une somme d'argent du solde du compte bancaire à l'aide d'une carte bancaire et d'un code confidentiel, dans des limites fixées à l'avance contractuellement.
Date d'effet	Date d'entrée en vigueur d'un contrat, d'un avenant ou d'une garantie pouvant correspondre à la date du paiement de la première cotisation ou du premier versement.
Date d'opération	Date à laquelle l'opération est effectuée par le client.
Date comptable	Date à laquelle la banque enregistre comptablement l'opération sur le compte du client.
Date de valeur	Date de référence qui sert au calcul des intérêts créditeurs ou débiteurs. La Société Générale n'applique pas de dates de valeur, à l'exception des remises de chèques pour lesquelles une date de valeur de 2 jours ouvrés est prise en compte pour le calcul des intérêts en raison des délais techniques d'encaissements.
Débit (écriture de débit)	Opération comptable qui diminue le solde du compte, par exemple à la suite de l'émission d'un chèque, d'un prélèvement ou d'un retrait d'espèces à un DAB.
Débiteur (nom)	Personne physique ou morale tenue de remplir une obligation. Le plus souvent, il s'agit de payer une somme d'argent à un créancier. Un compte de dépôt est dit débiteur lorsque son solde est négatif.
Débiteur	Position d'un compte de dépôt dont le solde est négatif, ou adjectif qualifiant des

(adjectif)	intérêts (intérêts débiteurs).
Découvert du compte	Position d'un compte de dépôt lorsque son solde est négatif. Cette situation peut avoir été contractualisée (autorisation de découvert) préalablement ou non par le banquier.
Dépassement	Fait d'excéder le montant d'un seuil (plafond autorisé) ; par exemple découvert du compte ou seuil de retrait d'espèces autorisé par carte bancaire.
Dépassement	Fait d'excéder le montant d'un seuil (plafond autorisé) ; par exemple découvert du compte ou seuil de retrait d'espèces autorisé par carte bancaire.
Détachement	Situation d'une personne envoyée à l'étranger pour une mission professionnelle inférieure à 6 mois et qui, de ce fait, reste fiscalement domiciliée en France.
Donateur	Auteur d'une donation.
Donation	Acte juridique par lequel un donateur se dépouille immédiatement et irrévocablement de la chose donnée au profit du donataire qui l'accepte.
Droit au compte	Droit qui vous permet, si vous n'avez pas ou plus de compte de dépôt et si une banque refuse de vous en ouvrir un, de demander à la Banque de France de désigner une banque où vous pourrez bénéficier d'un compte et des services bancaires gratuits associés au droit au compte.
Echéance	Date à laquelle un engagement doit être exécuté ou date qui marque la fin d'un contrat.
Emission de chèque	Signature du chèque par son titulaire et remise ou envoi à son bénéficiaire.
Facilité de caisse	Type d'autorisation de découvert accordé au client, et utilisable jusqu'à 15 jours, consécutifs ou non, par mois calendaire. Des agios sont facturés pour l'utilisation de cette facilité.
GAB	Guichet Automatique de banque. Appareil qui permet à l'aide d'une carte bancaire et de son code confidentiel d'effectuer un certain nombre d'opérations sur un compte (retrait d'argent, consultation du compte, commande de chéquier,...), contrairement au DAB qui ne permet que des retraits de billets.
Indice	Chiffre une évolution en prenant pour référence une valeur définie (on prend généralement le niveau 100).
Intérêts créditeurs	Somme due au client au titre de ses comptes rémunérés ou de ses placements. Le calcul de cette somme tient compte des dates de valeur.
Intérêts débiteurs	Somme due à la banque lorsqu'un compte présente un solde négatif pendant un ou plusieurs jours. Le calcul de cette somme tient compte des dates de valeur.
Obligation	Valeur mobilière représentative d'un droit de créance à moyen ou long terme émise par une entreprise, une collectivité ou l'Etat, le plus souvent négociable en Bourse et remboursable dans des conditions fixées dans le contrat d'émission.
Opposition carte par la banque	Opération par laquelle la banque refuse toute transaction en cas d'utilisation abusive d'une carte par le titulaire de la carte (client-porteur).
Opposition sur prélèvement	Opération par laquelle le titulaire d'un compte donne l'ordre à sa banque, par courrier, Internet ou téléphone confirmé par courrier, de refuser à l'organisme émetteur la demande de(s) paiement(s) qu'il a présentée et pour laquelle une autorisation préalable de prélèvement avait été donnée.
Prélèvement	Opération qui permet à la banque, conformément à l'autorisation de prélèvement donnée par le client, de payer un créancier en débitant son compte de dépôt.
Prélèvement impayé	Rejet d'un prélèvement par la banque quand le solde disponible du compte est insuffisant pour le régler.
Prêt	Opération par laquelle la banque met à la disposition d'un client une somme d'argent. En contrepartie, celui-ci verse à la banque des intérêts et divers frais et lui rembourse le capital selon des modalités déterminées dans un contrat. Les prêts

	sont de diverses formes, selon leur objet : prêt immobilier, prêt à la consommation (prêt personnel, prêt affecté...).
Recherche de documents	Prestation généralement payante de recherche et d'édition par la banque, à la demande du client, de documents concernant son compte (historique de compte, duplicata de relevés de compte, documents juridiques...).
Rejet de chèque	Refus de paiement, par la banque de l'émetteur, d'un chèque remis à l'encaissement par le bénéficiaire. Le refus est le plus souvent dû à un défaut ou à une insuffisance de provision.
Rejet de prélèvement	Refus du paiement d'un prélèvement du fait d'une insuffisance de provision ou d'une opposition demandée par le client.
Relevé de compte	Document récapitulant les opérations enregistrées sur le compte d'un client pendant une période déterminée, généralement mensuelle. Il est conseillé de le conserver pendant 10 ans.
Remboursement périodique de prêt	Paiement à la banque, à l'échéance contractuelle convenue, d'une partie du capital et des intérêts auxquels s'ajoutent des frais d'assurance éventuels.
Remise de chèque(s)	Dépôt de chèque(s) par le client auprès de sa banque pour encaissement. Elle nécessite la signature du bénéficiaire au dos du chèque (endos) ainsi que l'indication du numéro de compte à créditer.
Réserve (successorale)	Part de la succession devant être attribuée aux héritiers réservataires et ne pouvant pas faire l'objet d'une donation à un tiers.
Retrait	Opération par laquelle un client retire de son compte, au distributeur de billets ou au guichet, une certaine somme en espèces dont le montant est porté au débit de son compte. Les conditions de facturation ne sont pas les mêmes suivant que le retrait est fait ou non auprès d'une autre banque que la sienne, et à l'intérieur ou hors de l'Union Européenne.
Saisie-attribution	Procédure juridique permettant à un créancier de se faire payer le montant de sa créance. Le créancier doit nécessairement disposer d'un titre exécutoire (jugement). Il existe une somme insaisissable sur le compte (solde bancaire insaisissable) dans la mesure où ce compte est créditeur.
Solde bancaire insaisissable	Somme forfaitaire qui ne peut être saisie. Elle est destinée aux besoins alimentaires immédiats lorsque le compte est saisi. Toute personne, dont le compte est saisi peut, sur simple demande auprès de sa banque dans les 15 jours suivant la saisie, disposer de cette somme insaisissable égale au RMI « pour une personne seule », dans la limite du solde créditeur du compte. Le solde bancaire insaisissable n'est possible que sur un seul compte de dépôt même si le client en dispose de plusieurs.
Solde bancaire insaisissable	Somme forfaitaire qui ne peut être saisie. Elle est destinée aux besoins alimentaires immédiats lorsque le compte est saisi. Toute personne, dont le compte est saisi peut, sur simple demande auprès de sa banque dans les 15 jours suivant la saisie, disposer de cette somme insaisissable égale au RMI « pour une personne seule », dans la limite du solde créditeur du compte. Le solde bancaire insaisissable n'est possible que sur un seul compte de dépôt même si le client en dispose de plusieurs.
Solde du compte	Différence entre la somme des opérations au débit et au crédit d'un compte. Le solde est dit créditeur (positif) lorsque le total de ses crédits excède celui de ses débits (solde positif), et débiteur (négatif) dans le cas contraire.
Taux de change	Valeur d'échange d'une monnaie dans une autre.
Taux d'intérêt	Pourcentage permettant de calculer la rémunération d'une somme d'argent pour une période donnée (jour, mois, année).
Télé-règlement	Moyen de paiement permettant de régler à distance une dette. Il est à l'initiative du débiteur.
Testateur	Auteur d'un testament.

Tireur	Personne qui crée un chèque.
Virement	Opération par laquelle un client donne l'ordre à sa banque de débiter son compte pour en créditer un autre. Il peut être occasionnel ou permanent.
Virement occasionnel	automatisable dans l'UE Virement occasionnel, d'un montant inférieur ou égal à 12 500 EUR, émis dans l'UE (y compris la France), d'un compte vers un autre compte en fournissant l'identité correcte du bénéficiaire. En France, cette identité est fournie par le RIB et dans l'UE par l'IBAN, et le BIC.
Virement occasionnel non automatisable	émis sans RIB, BIC ou IBAN (Europe), ou émis hors d'Europe Virement soit ne remplissant pas les conditions d'automatisation dans l'UE (cf. déf. virement occasionnel automatisable dans l'UE), soit pour lequel le compte du bénéficiaire est situé en dehors de l'Union Européenne.
Virement permanent	Virement dont le renouvellement est automatique, à une date et selon une procédure convenues entre la banque, le client et le bénéficiaire.
Warrant	Instrument financier conférant à son détenteur le droit d'acheter ou de vendre un actif financier dans des conditions de prix et de durée définies à l'avance.

ANNEXE B

Domaines d'application du RàPC

L'approche de RàPC peut être utilisée dans différents contextes applicatifs touchant deux catégories de situation :
- la résolution des problèmes : conception, planification, design, diagnostic, aide à la décision
- l'interprétation: domaine juridique, interprétation, explication, apprentissage.

Systèmes de développements et applications industrielles associées

Le RÀPC a fait l'objet de développement de logiciels industriels tels que Kaidara, CBR Express, ReCall, Esteem, CBR-Works, ReMind, Case Advisor... À leur tour, ces logiciels ont permis d'élaborer des systèmes de RàPC spécifiques à certaines applications industrielles. En voici quelques exemples :

CaseBank a permis la mise au point des systèmes interactifs d'aide au diagnostic de pannes permettant de guider les opérateurs de maintenance dans le dépannage — ceci étant dédié à l'industrie aéronautique, pour le diagnostic des moteurs d'avions. CaseBank a travaillé pour HONEYWELL, AIR CANADA, BRITISH AIRWAYS, GENERAL ELECTRIC.

CaseLine est un démonstrateur utilisé par BRITISH AIRWAYS pour le diagnostic des pannes et la réparation des moteurs de Boeing 747-400. Ce système a été implémenté à l'aide de l'outil de développement ReMind.

Kaidara a travaillé pour PSA PEUGEOT CITROËN sur un système de diagnostic et de réparation des véhicules sous garantie malgré la complexité induite par les nouveaux modèles intégrant des calculateurs en réseau.

CBR Express : le système CheckMate, mis au point à partir de CBR Express, est dédié à l'assistance technique et au diagnostic d'imprimantes industrielles sur différents types de matériaux. Ses fonctions principales sont la documentation et la distribution d'informations et d'expertise technique.

Actuellement plus d'une trentaine d'outils de développement existent sur le marché international, mais ils ne développent pas toutes les phases du raisonnement. Par exemple les RàPC conversationnels, la plupart du temps, n'ont pas de phase d'adaptation. De plus, il faut souvent agencer un ensemble de modules achetés indépendamment pour réaliser les différentes phases du RàPC : module de création d'une base de cas, de sa mise à jour ainsi que de sa maintenance, module destiné à la recherche des cas similaires, et enfin module pour l'adaptation de cas s'il en existe.

Systèmes de développements universitaires et prototypes

Des travaux de recherches ont créé des logiciels académiques qui vont plus loin dans certaines phases du cycle de RàPC :

- le système **Patdex** pour le diagnostic technique des machines complexes a été développé par Richter à l'université Kaiserslautern.

- **Creek** est un système de diagnostic des pannes de voiture, basé sur un modèle intégrant résolution de problèmes et apprentissage.

- **Pad'im** aide à la conception des systèmes de supervision dans le milieu industriel. Il propose une aide à l'opérateur dans le choix d'un nouvel environnement (différents tableaux de bord affichés sur les écrans de contrôle) après un changement de situation.

Méthodes de Spécification

La spécification est la phase la plus importante et délicate du cycle de vie du logiciel. Plusieurs méthodes d'analyse et de conception ont été utilisées pour construire des spécifications pour les systèmes à modéliser. En réalité, la plupart des méthodes peuvent être classées dans l'une des catégories suivantes [Sommerville 04]:

- conception orientée par les données,
- conception structurée descendante,
- conception systémique,
- conception orientée objets,
- conception orientée composants,
- conception orientée modèles.

Un bon exemple de **conception orientée par les données** (devenu obsolète) se trouve dans les premiers travaux de Jackson [Jackson 75], [Jackson 83] et les méthodes d'Orr [Orr 71]. Dans cette méthode, la structure d'un logiciel est obtenue en associant les entrées du système à ses sorties. La conception orientée par les données a été appliquée avec succès à un certain nombre de domaines complexes, particulièrement en informatique de gestion, qui impliquent des relations directes entre les entrées et les sorties du système, mais ne nécessitent pas un soin particulier envers la chronologie des événements.

La **conception structurée descendante** repose sur le concept qu'un système est conçu d'un point de vue fonctionnel, en commençant au niveau le plus général et en descendant progressivement vers la conception détaillée. Un bon exemple de conception structurée descendante est fourni par le travail de Yourdon et Constantine [Yourdon et Constantine 79], Myers [Myers 78] et Page-Jones [Page-Jones 88]. Les fondements de cette méthode proviennent du travail de Wirth [Wirth 83], [Wirth 86] et Dahl, Dijkstra et Hoare [Dahl et *al.* 72]; une importante variante de la conception structurée se trouve dans la méthode de conception de Mills, Linger et Hevner [Mills et *al.* 86]. Chacune de ces variations applique la décomposition algorithmique. Il est probable que plus de logiciels ont été écrits en utilisant ces méthodes de conception qu'avec n'importe quelle autre. Parmi les exemples, nous citons les méthodes SADT, SD, SA.

La **conception structurée** de Yourdon et Constantine [Yourdon et Constantine 79] (SD: Structured Design) reprend les principes de décomposition fonctionnelle de l'analyse structurée, en précisant les liens (simples, itératifs, alternatifs) ainsi que les passages de paramètres, entre les différents modules. La notation utilisée est celle des diagrammes de structure. Un modèle d'information, similaire au modèle entité-association, complète souvent cette méthode. L'analyse structurée de De Marco [De Macro 79] (SA: *Structured Analysis*) est une méthode descendante par raffinements successifs des traitements ou processus (voir la figure c.1).

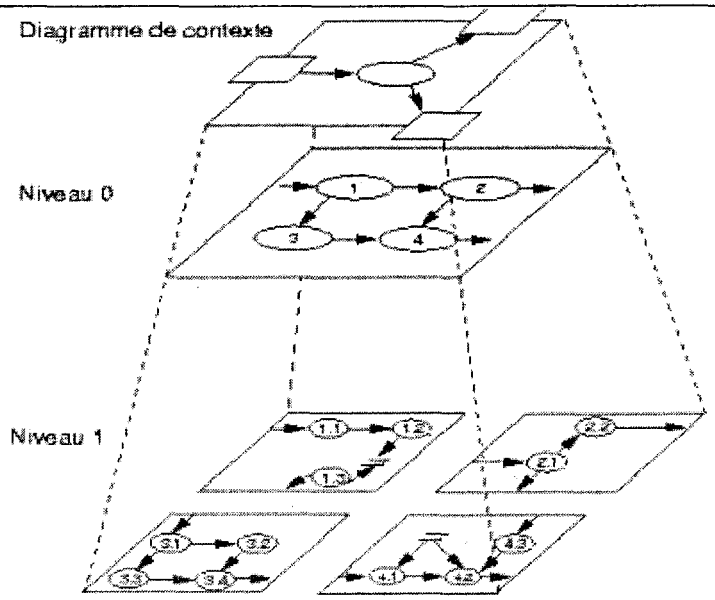


Figure c.1: Raffinements successifs des traitements

La **conception systémique** est une approche inspirée d'une vision systémique et vise à modéliser les deux aspects d'un système structurel et comportemental. La méthode MERISE est l'une des méthodes de conception systémique, elle est conçue pour le développement des applications des systèmes complexes. MERISE constitue un standard de fait en conception de système d'information et elle est basée sur les concepts suivants [Nanci et Espinasse 96]:

- Extension du formalisme entité-relation avec notamment l'explication de types et sous-types, de contraintes d'intégrité, etc....
- Clarification de la modélisation des traitements à l'aide du formalisme issu des réseaux de Pétri, à différents niveaux de préoccupation.
- Extension des niveaux d'abstraction et de modèles, avec l'émergence du modèle logique de traitements (MLT) et du modèle organisationnel de données (MOD).
- Couplage avec des méthodes de conduite de projet.
- Développement d'ateliers de génie logiciel (AGL) de conception intégrant de façon plus ou moins complète la méthode MERISE (AMC, MEGA, TRAMIS, SILVERRUN, WIN'DESIGN...).
- Ouverture vers les autres méthodes de génie logiciel (MERISE et YOURDON [Phan 89]...), de génie cognitif (MERISE et KADS [Brunet et *al.* 90] ....),....

Cette ouverture s'est particulièrement manifestée vers :

- L'approche objet qui, à partir des langages de programmation, ambitionne de couvrir progressivement l'ensemble du champ du génie informatique.
- Le développement rapide, associe à la diffusion d'outils de productivité, qui a remis en cause la démarche classique adoptée par Merise.
- Le client-serveur qui a installé définitivement le micro-ordinateur comme poste de travail.
- Le Réingénierie du processus d'affaires « Business Process Reengineering » qui ravive l'intérêt sur les problèmes d'organisation et les systèmes d'information.

Le concept de base de la **conception orientée objets** est que l'on doit modéliser les logiciels comme des ensembles d'objets coopérants, en traitant les objets individuels comme des instances d'une classe à l'intérieur d'une hiérarchie de classes.
La conception orientée objets est une méthode de conception incorporant le processus de décomposition orientée objets et une notation permettant de dépeindre à la fois les modèles logiques et physiques, aussi bien que statiques et dynamiques du système à concevoir [Booch 94]. Cette définition comporte deux parties importantes : la conception orientée objets (1) conduit à une décomposition orientée objets et (2) utilise des notations différentes pour exprimer des modèles différents de la conception logique (structure de classes et d'objets) et physique (architecture des modules et des processus) d'un système.
Le support de la décomposition orientée objets est ce qui la différencie de la conception structurée ; la première utilise des abstractions de classes et d'objets pour structurer logiquement les systèmes, alors que la seconde utilise des abstractions algorithmiques.
Le modèle à objets est la base d'une conception orientée objet. Il y a quatre éléments majeurs dans ce modèle [Booch 94]:

- l'abstraction,
- l'encapsulation,
- la modularité,
- la hiérarchie.

Par majeurs, nous voulons dire que si un modèle ne possède pas l'un de ces éléments, il n'est pas orienté objets.
Il y a trois éléments mineurs dans le modèle à objets [Booch 94]:

- le typage,
- la simultanéité,
- la persistance.

Par mineurs, nous voulons dire que chacun de ces éléments est une partie utile, mais non essentielle, du modèle à objets.

La conception orientée composant est un cadre de modélisation dont le but est de développer des applications par assemblage des composants standards indépendants basé sur un modèle de composant.
Councill et Heineman [Councill et Heineman 01] définissent un composant comme :

> *« A software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard".*

Cette définition est trop abstraite et ne donne pas une image claire de ce qu'il peut supporter un composant. Une importante façon de présenter un composant est celui d'un fournisseur de services autonomes (« standalone service provider ») [Sommerville 04]. Deux caractéristiques importantes sont à signaler lorsque le composant est vu comme un fournisseur de services autonomes :

1. Le composant est une entité indépendante et exécutable. Le code source n'est pas disponible, donc il n'est pas nécessaire de compiler le composant pour qu'il soit utilisé avec un autre système de composants.

2. Les services offerts par un composant sont construits pour être disponibles à travers une interface. L'interface du composant est exprimée par des opérations paramétrisées et son état interne n'est jamais exposé (figure c.2).

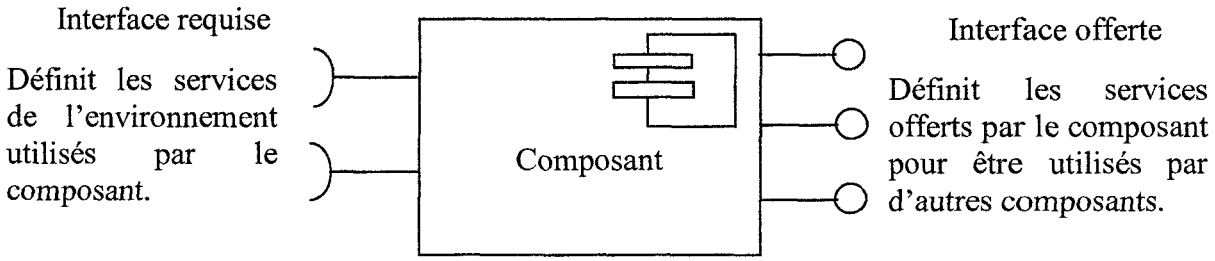


Figure c.2 : Interfaces de composant

Interface offerte : définit les services fournis par le composant. Ces services sont implémentés par des méthodes argumentées et qui seront appelées par l'usager de ce composant.
Interface requise : spécifie les services qui sont fournis par d'autres composants dans le système. Si ces services ne sont pas disponibles le composant ne peut pas fonctionner.

Les composants possèdent des méthodes qui sont similaires à celles utilisées par les classes d'objets. Les composants sont construits par des langages orientés objets comme avec les classes d'objets. Mais quelles distinctions existent-elles entre les composants et les classes d'objets ?

- Les composants sont des entités déployables et qui ne seront pas compilés pour être utilisés dans un programme d'application mais ils sont installés directement dans une plate-forme exécutable.
- Les composants ne définissent pas de types comme avec les classes qui sont des types abstraits et les objets seront des instances de ces types. Le composant est une instance et non pas une structure (template) pour définir des instances.
- L'implémentation des composants est cachée de l'usager des composants. Souvent, les composants sont délivrés en forme binaire et les utilisateurs n'ont pas la possibilité d'accéder à cette implémentation.
- Les composants sont construits par des langages orientés objets mais ils peuvent être utilisés par des langages non orientés objets.
- Les composants sont standardisés et ne sont pas comme les classes où on peut les implémenter de plusieurs façons différentes. Les composants doivent être conformes au modèle de composant pour être validés.

Le modèle de composant

Un modèle de composant est une définition de standards, d'implémentation, documentation et de déploiement. Ces standards permettent l'interopérabilité des composants.
Plusieurs modèles de composant sont proposés tels que Entreprise Java Beans [Sun 01], CORBA [Wang et Schmidt 01], Microsoft COM$^+$ [Ewald 01]. Les éléments de base d'un modèle de composant sont ceux proposés par Weinreich et Sametinger [Weinreich et Sametinger 01] (figure c.3).

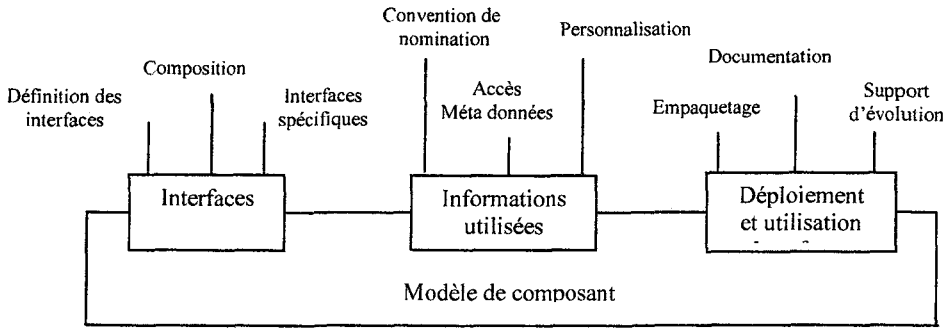


Figure c.3 : Eléments de base d'un modèle de composant

La définition des interfaces décrit les noms des opérations, les paramètres et les exceptions qui doivent être inclus dans l'interface. Le modèle doit encore spécifier le langage utilisé pour définir les interfaces (IDL : *Interface Definition Language*). Quelques modèles de composant nécessitent des interfaces spécifiques qu'il faut définir par un composant. Ces interfaces spécifiques permettent l'intégration des composants avec l'infrastructure du modèle de composant qui fournit des services standardisés tels que la sécurité et la gestion des transactions. Afin d'être distribués et accédés à distant, les composants devront avoir un nom unique. COM+ fournit un identificateur unique sur 128 bits. COBRA et EJB fournissent une hiérarchie de nom basée sur une racine correspondant au nom du domaine de l'internet.

La méta-donnée du composant est un ensemble de données concernant le composant lui-même comme les interfaces et les attributs. La méta-donnée est importante car elle permet aux utilisateurs de trouver les services fournis et demandés par le composant.
Les composants sont des entités génériques, quand ils sont déployés ils doivent être personnalisés dans le nouvel environnement de l'application. Pour cela, le modèle de composant doit spécifier comment les composants binaires seront-ils configurés pour les intégrer dans un environnement de déploiement. Ceci signifie que le modèle de composant doit définir la façon de paqueter les composants car ils sont des entités indépendantes. Ainsi le modèle de composant doit inclure les règles guidant le remplacement des composants par des autres.

Finalement, le modèle de composant doit définir la documentation du composant à produire. Les modèles de composant ne sont pas uniquement des standards mais aussi les bases des systèmes « middleware » qui fournissent un support pour l'exécution des composants. L'implémentation d'un modèle de composant fournit des services partagés par les composants ; ces services sont classés en deux catégories : services plateformes et services horizontaux (figure c.4).

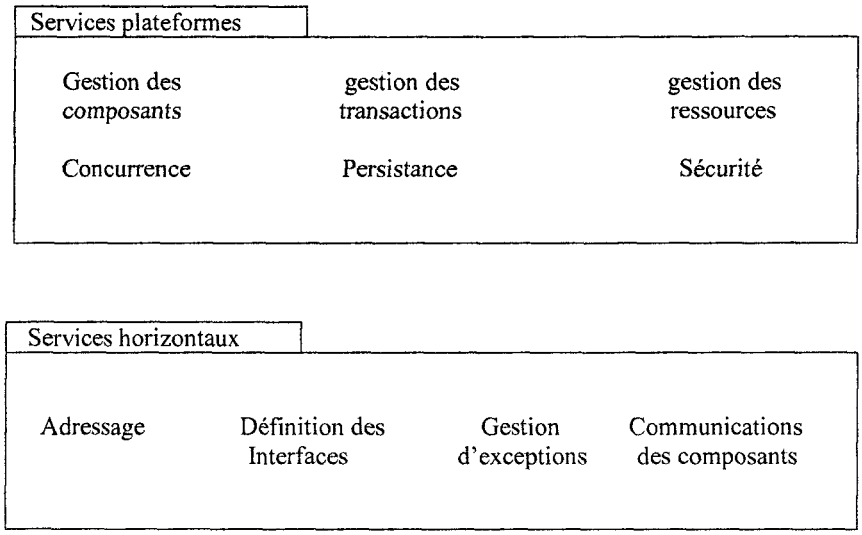


Figure c.4 : Services offerts par un modèle de composant

Les services plateformes sont fondamentaux qui permettent aux composants de se communiquer avec des autres composants (exemple CORBA *plateform*). Les services horizontaux sont utilisés pour réduire le coût de développement des composants et pour éviter leur incompatibilité.

La conception orientée modèles ou dirigée par les modèles permet de s'abstraire complètement des préoccupations technologiques comme les plateformes et les langages de programmation. Elle permet aussi de capitaliser les architectures métier et applicatives à même d'être implantées sur différentes plateformes technologiques [Kadima 05].
La méthode la plus célèbre qui utilise cette conception est le MDA (Model Driven Architecture). MDA définit un paradigme de développement et d'intégration de systèmes d'informations qui permet l'élaboration des modèles métier indépendamment de toute plate-forme technique et assure la transformation de ces modèles en modèles techniques dépendant de plates-formes.

La transformation de modèles comporte essentiellement deux étapes qui consistent respectivement à spécifier les règles de transformation exprimant la correspondance entre les concepts du métamodèle source (c'est-à-dire le métamodèle qui décrit le modèle source) et les concepts du métamodèle cible et ensuite à appliquer ces règles au modèle source pour produire le modèle cible (figure 2.5).
Les règles de transformation expriment la correspondance entre les concepts au niveau métamodèle. Elles sont indépendantes de modèles instances spécifiques et peuvent ainsi être appliquées à tous les modèles instances du métamodèle source. Le moteur de transformation applique les règles de transformation au modèle source pour produire le modèle cible.

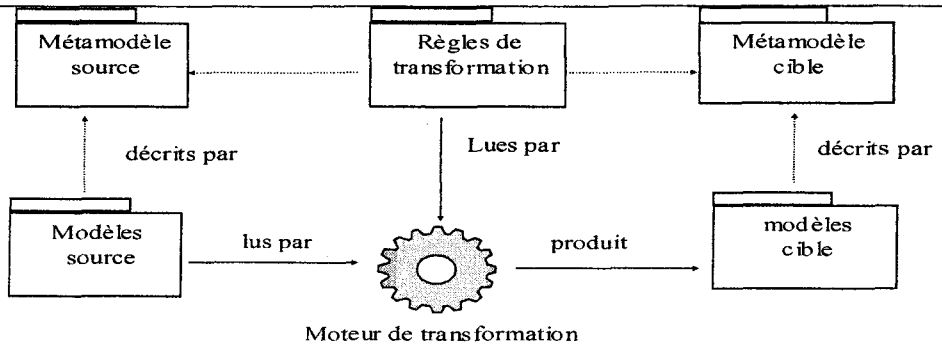


Figure c.5 : Processus de transformation de modèles pilotés par les méta-modèles

Cette indépendance met le MDA dans un défi de réaliser les objectifs de portabilité, d'interopérabilité et de réutilisabilité à travers une séparation architecturelle des modèles [Miller et Mukerji 03].
Le modèle d'architecture MDA prend en charge l'ensemble du cycle de conception, de déploiement, d'intégration et d'administration des applications et des données dans une infrastructure distribuée, ouverte et normalisée.

Annexe D

Exemple d'utilisation du langage LSP

L'exemple ci-dessous présente un Guichet Automatique des Billets (GAB) avec plusieurs scénarios possibles. L'exemple est retiré des machines GAB de la banque AUDI et la banque SGBL au LIBAN. Malgré que l'exemple semble trop pris par beaucoup des exemples en UML mais il se présente comme un cas explicatif et clair pour tous les concepts, les règles et les contraintes de notre approche.

Les scénarios présentés dans la table d.1 forment un groupe de scénarios possibles des machines DAB des banques déjà citées. Ces scénarios sont souvent similaires dans toutes les banques avec quelques modifications superficielles non profondes.

Type de SP	Acteur	Opération	Type
principal	**Client**	- L'utilisateur insère la carte de crédit.	**OpD**
		- Le GAB valide la carte et demande d'insérer le mot de passe.	**OpD**
		- L'utilisateur saisit le mot de passe.	**OpD**
		- Le GAB vérifie le mot de passe et demande de choisir une option.	**OpD**
		- L'utilisateur choisit l'option « retrait en liquide ».	**OpP**
		- Le GAB affiche deux possibilités de monnaies: Livre libanaise et dollar américain.	**OpP**
		- L'utilisateur choisit la monnaie dollar.	**OpP**
		- Le GAB affiche les montants : 50, 100, 200, 300, 400, 600.	**OpP**
		- L'utilisateur choisit un montant.	**OpP**
		- Le GAB valide le montant et rejette la carte et l'argent.	**OpP**
		- L'utilisateur récupère la carte et l'argent.	**OpR**
		- Le GAB se réinitialise et demande à l'utilisateur d'insérer sa carte de crédit.	**OpR**
Principal spécialisé **« Certains GAB impriment des reçus après opérations »**	**Client**	- L'utilisateur insère la carte de crédit.	**OpD**
		- Le GAB valide la carte et demande d'insérer le mot de passe.	**OpD**
		- L'utilisateur saisit le mot de passe.	**OpD**
		- Le GAB vérifie le mot de passe et demande de choisir une option.	**OpD**
		- L'utilisateur choisit l'option « retrait en liquide ».	**OpP**
		- Le GAB affiche deux possibilités de monnaies: Livre libanaise et dollar américain.	**OpP**
		- L'utilisateur choisit la monnaie dollar.	**OpP**
		- Le GAB affiche les montants : 50, 100, 200, 300, 400, 600.	**OpP**
		- L'utilisateur choisit un montant.	**OpP**
		- Le GAB valide le montant.	**OpP**
		- Le GAB imprime un reçu.	**OpS**
		- Le GAB rejette la carte et l'argent	**OpP**

		- L'utilisateur récupère la carte et l'argent. - Le GAB se réinitialise et demande à l'utilisateur d'insérer sa carte de crédit.	**OpR** **OpR**
alternatif	**Client**	- L'utilisateur insère la carte de crédit. - Le GAB valide la carte et demande d'insérer le mot de passe. - L'utilisateur saisit le mot de passe. - Le GAB vérifie le mot de passe. - Le GAB affiche « mot de passe incorrecte » et rejette la carte. - L'utilisateur récupère la carte et l'argent. - Le GAB se réinitialise et demande à l'utilisateur d'insérer sa carte de crédit.	**OpD** **OpD** **OpD** **OpD** **OpA** **OpR** **OpR**
Principal alternatif	**Client**	- L'utilisateur insère la carte de crédit. - Le GAB valide la carte et demande d'insérer le mot de passe. - L'utilisateur saisit le mot de passe. - Le GAB vérifie le mot de passe et demande de choisir une option. - L'utilisateur choisit l'option « retrait en liquide». - Le GAB affiche deux possibilités de monnaies: Livre libanaise et dollar américain. - L'utilisateur choisit la monnaie dollar. - Le GAB affiche les montants : 50, 100, 200, 300, 400, 600,. - L'utilisateur choisit un montant. - Le GAB valide le montant. - Le GAB affiche « solde pour retrait insuffisant » et rejette la carte. - L'utilisateur récupère la carte. - Le GAB se réinitialise et demande à l'utilisateur d'insérer sa carte de crédit.	**OpD** **OpD** **OpD** **OpD** **OpP** **OpP** **OpP** **OpP** **OpP** **OpP** **OpA** **OpR** **OpR**
réduit	**Client**	- L'utilisateur insère la carte de crédit. - Le GAB valide la carte et affiche « Carte de crédit n'est pas valide ». - Le GAB rejette la carte. - L'utilisateur récupère la carte. - Le GAB se réinitialise et demande à l'utilisateur d'insérer sa carte de crédit.	**OpD** **OpD** **OpR** **OpR** **OpR**

Table d.1: Un groupe de scénarios possibles pour une machine GAB

Les types d'opérations figurées dans la table précédente sont des symboles réservés pour le langage LSP, leur description est la suivante :

OpD: Opération déclencheur
OpP: Opération principale
OpS: Opération spécialisée
OpA: Opération Alternative
OpR: Opération d'arrêt

Les scénarios de la table 3.1 sont traduits par l'éditeur et la syntaxe du langage LSP en des scénarios patterns réutilisables. Voici un exemple pour quelques scénarios.

Scénario pattern principal:

BEGIN
Retrait d'argent ;
Principal ;
Client ;
/SP
GAB, OpD, 1, insérer ; boolean
Carte, OpD, 2, valider, numéroCarte, dateExpiration ; boolean
interface, OpD, 3, saisir ; motDePasse
Carte, OpD, 4, vérifier, motDePasse ; balance, boolean
interface, OpP, 1, sélectionner ; option
interface, OpP, 2, sélectionner ; monnaie
interface, OpP, 3, sélectionner ; montant
Carte, OpP, 4, autoriser, numéroCarte, solde ; boolean
GAB, OpP, 5, rejeter ; boolean
GAB, OpP, 6, rejeter ; boolean
GAB, OpR, 1, récupérer, boolean
GAB, OpR, 2, récupérer, boolean
GAB, OpR, 3, réinitialiser, boolean
SP/
END

Scénario pattern principal spécialisé:

BEGIN
Retrait d'argent ;
Principal spécialisé ;
Client ;
/SP
GAB, OpD, 1, insérer ; boolean
Carte, OpD, 2, valider, numéroCarte, dateExpiration ; boolean
interface, OpD, 3, saisir; motDePasse
Carte, OpD, 4, vérifier, motDePasse; balance, boolean
interface, OpP, 1, sélectionner; option
interface, OpP, 2, sélectionner; monnaie
interface, OpP, 3, sélectionner; montant
Carte, OpP, 4, autoriser, numérocarte, montant; boolean
GAB, OpS, 1, imprimer, numérocarte, montant
GAB, OpP, 1, rejeter ; boolean
GAB, OpP, 2, rejeter ; boolean
GAB, OpR, 3, récupérer ; boolean
GAB, OpR, 4, récupérer ; boolean
GAB, OpR, 5, réinitialiser ; boolean
SP/
END

Scénario pattern alternatif:

BEGIN
Retrait d'argent ;
Alternatif ;
Client ;
/SP
GAB, OpD, 1, insérer, boolean
Carte, OpD, 2, valider, numéroCarte, dateExpiration ; boolean
interface, OpD, 3, saisir ; motDePasse
Carte, OpD, 4, vérifier, motDePasse; balance, boolean
GAB, OpP, 1, rejeter ; boolean
GAB, OpR, 1, récupérer ; boolean
GAB, OpR, 2, réinitialiser ; boolean
SP/

Scénario pattern réduit:

BEGIN
Retrait d'argent ;
Réduit ;
Client ;
/SP
GAB, OpD, 1, insérer, boolean
Carte, OpD, 2, valider, numéroCarte, dateExpiration ; boolean
GAB, OpP, 1, rejeter ; boolean
GAB, OpR, 1, récupérer ; boolean
GAB, OpR, 2, réinitialiser ; boolean
SP/

Annexe E

Instances de Scénarios Patterns

Exemples d'instances de scénarios patterns décrivant le scénario pattern principal du cas d'utilisation « Retrait d'argent d'une machine GAB » (figures e).

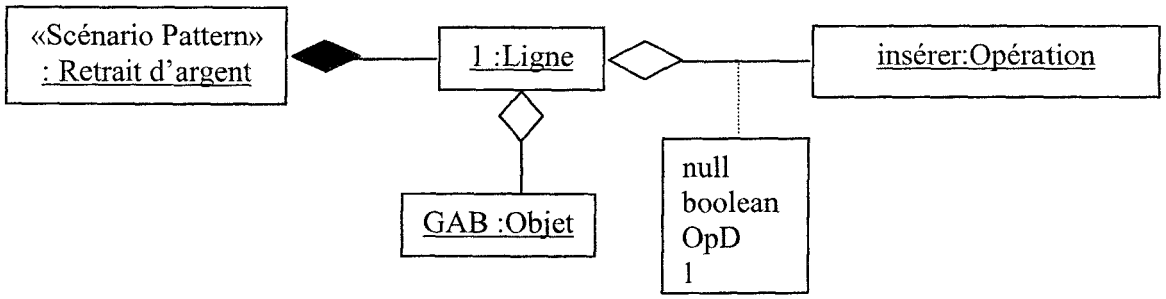


Figure e.1: Instance du Méta-modèle du scénario pattern « Retrait d'argent »

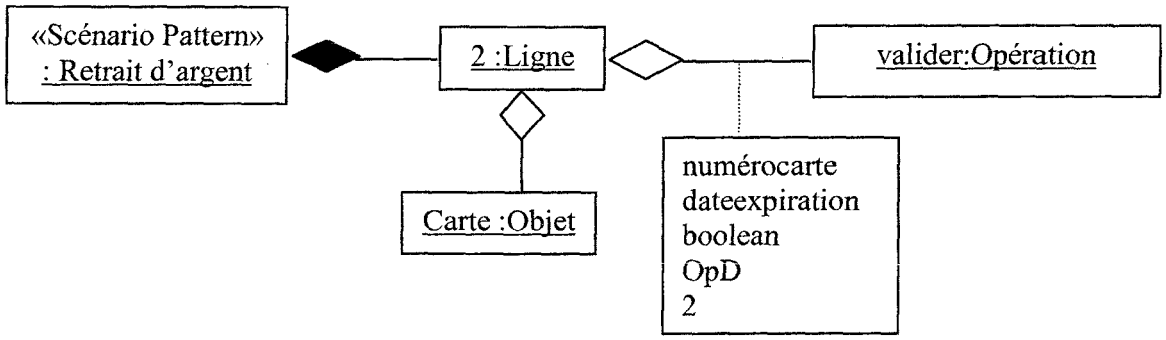


Figure e.2: Instance du Méta-modèle du scénario pattern « Retrait d'argent »

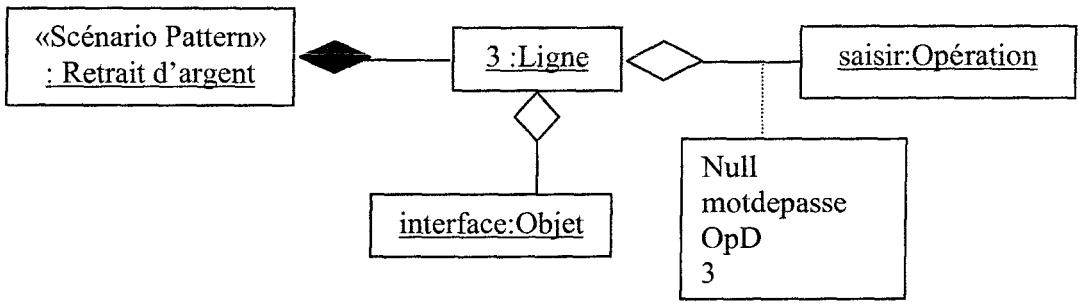


Figure e.3: Instance du Méta-modèle du scénario pattern « Retrait d'argent »

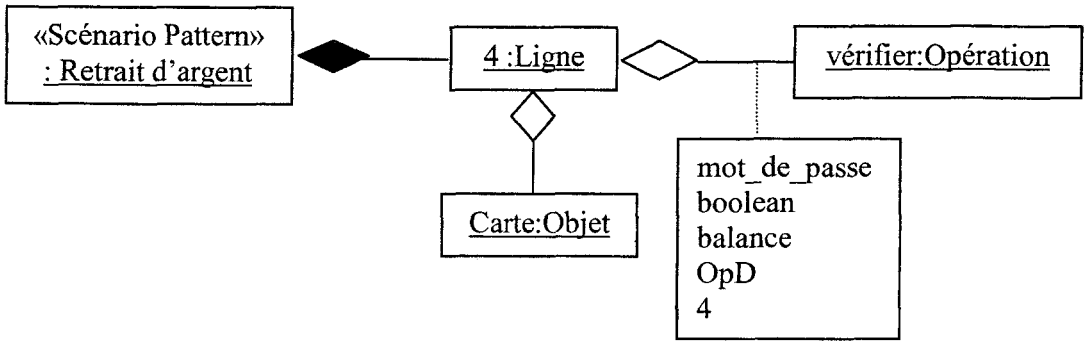


Figure e.4: Instance du Méta-modèle du scénario pattern « Retrait d'argent »

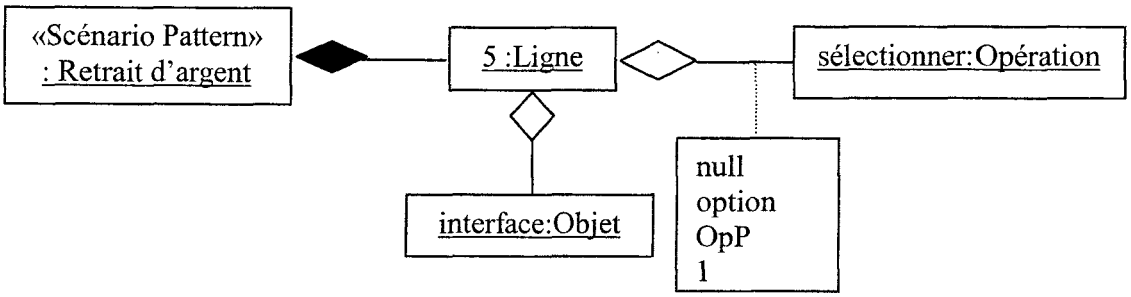


Figure e.5: Instance du Méta-modèle du scénario pattern « Retrait d'argent »

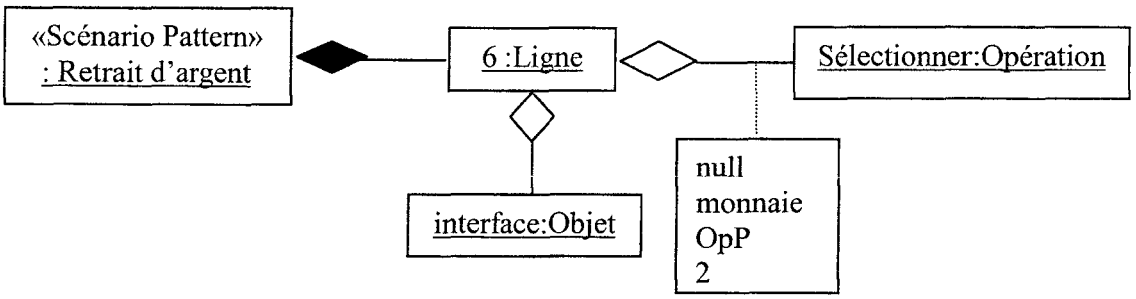


Figure e.6: Instance du Méta-modèle du scénario pattern « Retrait d'argent »

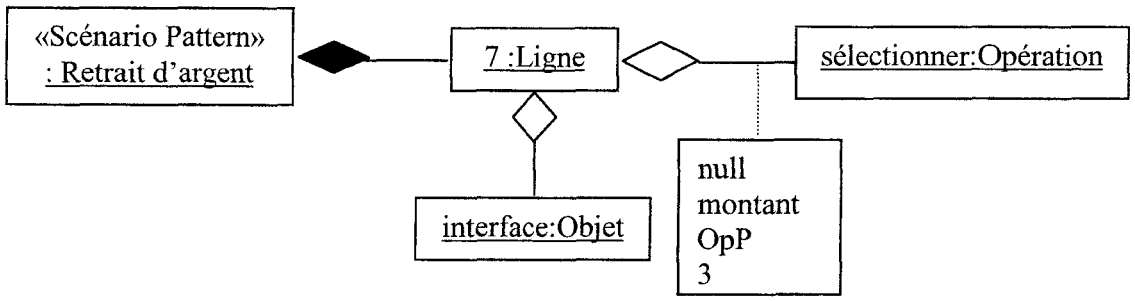


Figure e.7: Instance du Méta-modèle du scénario pattern « Retrait d'argent »

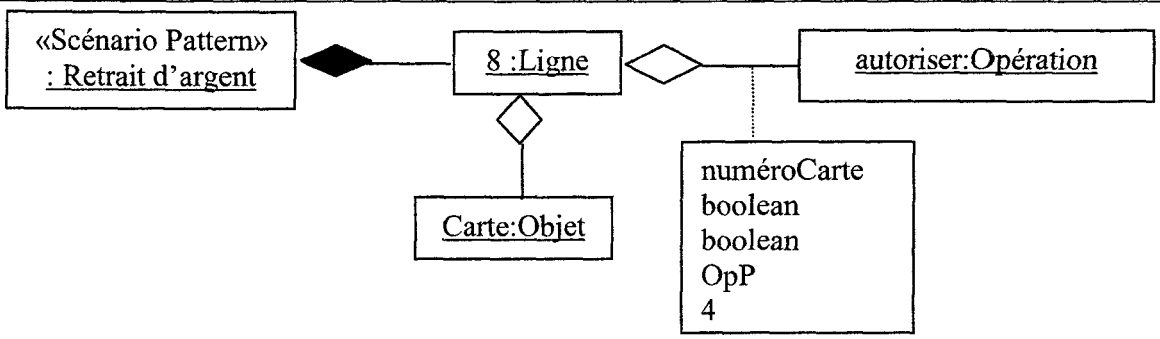


Figure e.8: Instance du Méta-modèle du scénario pattern " Retrait d'argent"

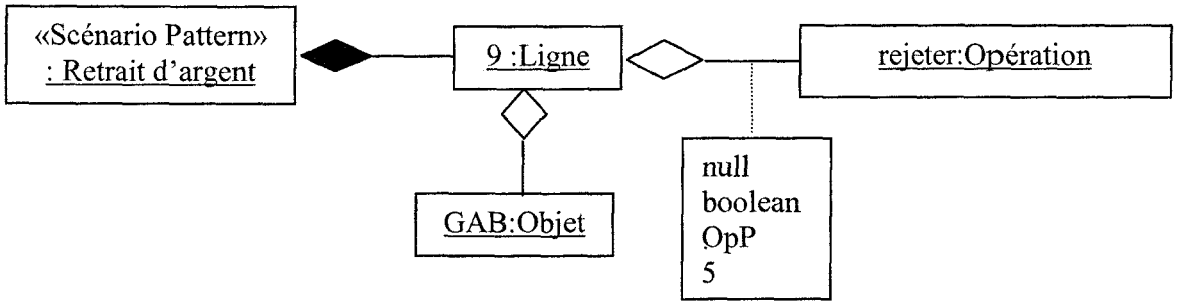


Figure e.9: Instance du Méta-modèle du scénario pattern " Retrait d'argent"

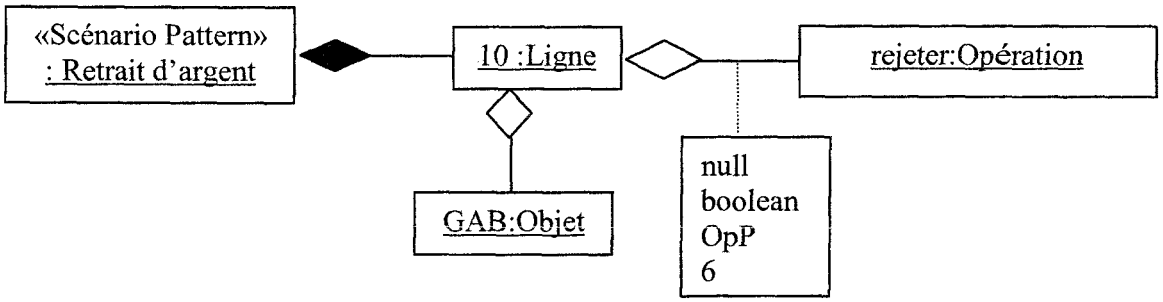


Figure e.10: Instance du Méta-modèle du scénario pattern " Retrait d'argent"

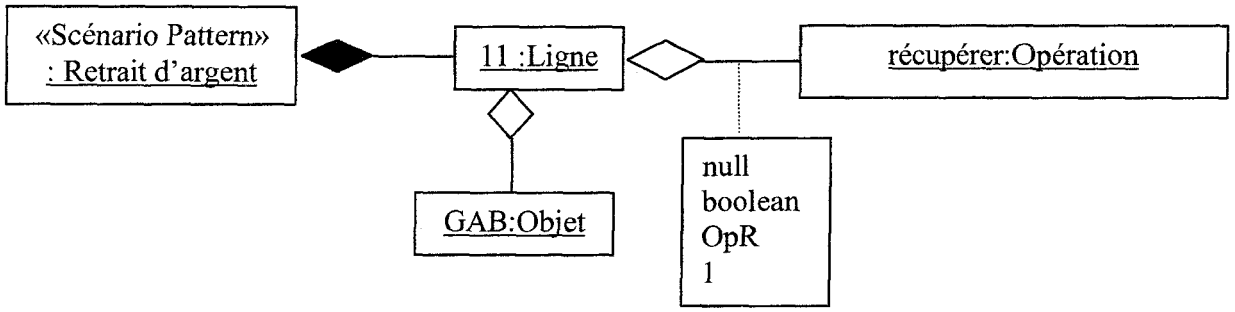


Figure e.11: Instance du Méta-modèle du scénario pattern " Retrait d'argent"

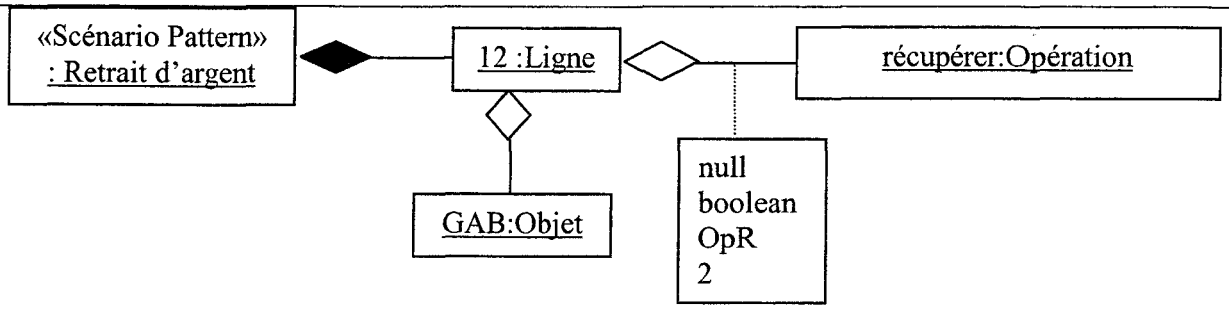


Figure e.12: Instance du Méta-modèle du scénario pattern " Retrait d'argent"

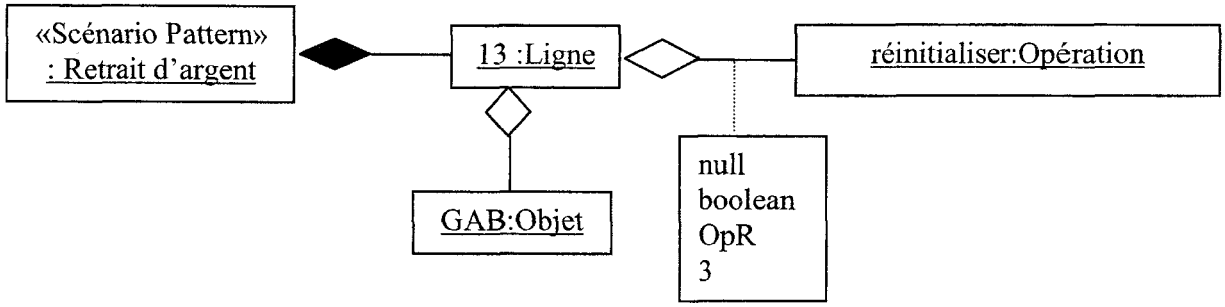


Figure e.13: Instance du Méta-modèle du scénario pattern " Retrait d'argent"

Annexe F

Dictionnaire des Connaissances Métier

Domaine	Bancaire
Sous Domaine	Gestion des comptes client
Activité	Dépôt d'argent

Mot	Nature	Forme abstraite	Explication	Synonyme
Agence	Acteur	Agence	Lieu d'accueil de la clientèle d'une banque.	Branche
Banque	Acteur	Banque	Établissement autorisé par la loi à assurer des opérations de banque c'est-à-dire la réception de fonds du public, les opérations de crédit, ainsi que la mise à la disposition de la clientèle ou la gestion de moyens de paiement. Le terme « établissement de crédit » ou « caisse » est également utilisé.	Néant
Chèque	Objet	Chèque	Moyen de paiement normalisé avec lequel le titulaire (tireur) d'un compte donne l'ordre à son banquier (tiré) de payer au bénéficiaire du chèque la somme inscrite sur celui-ci. La provision doit être disponible lors de l'émission du chèque et maintenue jusqu'à sa présentation.	Néant
Chèque de banque	Objet	Chèque	Chèque émis par une banque à la demande du client, et dont le montant, immédiatement débité du compte de dépôt du client, est ainsi garanti. Les chèques de banque sont souvent exigés pour le règlement d'achats importants.	Néant
Chèque sans provision	Objet	Chèque	Chèque émis sur un compte de dépôt dont le solde disponible ou le découvert autorisé est insuffisant pour régler le montant du chèque. L'émetteur se voit interdit d'émettre des chèques jusqu'à ce qu'il régularise sa situation. L'émission d'un chèque sans provision entraîne des frais bancaires et éventuellement des pénalités à payer au Trésor Public (si la régularisation n'a pas lieu dans un délai de 2 mois après l'interdiction). La provision doit être	Néant

			disponible dès l'émission du chèque et maintenue jusqu'à sa présentation.	
Chéquier	Objet	Chéquier	Carnet comportant généralement 25 à 30 formules de chèques (ou « vignettes »). Certaines banques donnent le choix du format du carnet.	Carnet
Client	Acteur	Client	Propriétaire d'un ou plusieurs comptes	Client distant
Client distant	Acteur	Client	Personne autorisée d'accéder à ses comptes depuis l'Internet.	Client
Commission	Attribut	Commission	Somme perçue par une banque en rémunération d'un service fourni à son client.	Rémunération Remise Intérêt
Compte de dépôt	Objet	Compte	Compte bancaire ordinaire (ou compte courant) utilisé pour gérer quotidiennement son argent. C'est sur ce compte qu'un client dispose en général d'une carte bancaire et d'un chéquier. Le compte doit être créditeur, sauf accord avec la banque.	Néant
Compte joint	Objet	Compte	Compte de dépôt ouvert au nom de deux (ou plusieurs) personnes qui ont chacune le droit de disposer seule de l'avoir du compte. Dans le cas d'un compte joint entre époux, le compte n'est pas bloqué en cas de décès d'un des co-titulaires.	Néant
Crédit	Opération	Créditer	Opération comptable qui augmente le solde du compte, par exemple à la suite d'un virement reçu, d'un dépôt d'espèce, ou d'une remise de chèque.	Déposer Dépôt
Crédit	Attribut	Crédit	Opération par laquelle un établissement de crédit met ou promet de mettre à disposition d'un client une somme d'argent moyennant intérêts et frais, pour une durée déterminée ou indéterminée (lorsque le crédit est dit gratuit, les frais et les intérêts sont nuls).	Somme
Créditeur	Acteur	Créditeur	Personne physique ou morale tenue de remplir une obligation. Le plus souvent, il s'agit de déposer une somme d'argent dans son compte. Un compte de dépôt est dit créditeur lorsque son solde est positif.	Client
Date d'opération	Attribut	Date	Date à laquelle l'opération est effectuée par le client.	Néant
Date comptable	Attribut	Date	Date à laquelle la banque enregistre comptablement l'opération sur le compte du client.	Néant
Date de valeur	Attribut	Date	Date de référence qui sert au calcul des intérêts créditeurs ou débiteurs. La Société Générale n'applique pas de dates de valeur, à l'exception des remises de chèques pour lesquelles une date de valeur de 2 jours ouvrés est prise en compte pour le calcul des intérêts en raison des délais techniques	Néant

			d'encaissements.	
Intérêts créditeurs	Attribut	Intérêt	Somme due au client au titre de ses comptes rémunérés ou de ses placements. Le calcul de cette somme tient compte des dates de valeur.	Néant
Opérateur	Acteur	Opérateur	Employé de la banque pour servir les clients de leurs opérations comptables.	Guichetier Agent Employé Utilisateur
Relevé de compte	Opération	Relever	Document récapitulant les opérations enregistrées sur le compte d'un client pendant une période déterminée, généralement mensuelle. Il est conseillé de le conserver pendant 10 ans.	Néant
Remise de chèque(s)	Opération	Remettre	Dépôt de chèque(s) par le client auprès de sa banque pour encaissement. Elle nécessite la signature du bénéficiaire au dos du chèque (endos) ainsi que l'indication du numéro de compte à créditer.	Dépôt Déposer
Solde du compte	Attribut	Solde	Différence entre la somme des opérations au débit et au crédit d'un compte. Le solde est dit créditeur (positif) lorsque le total de ses crédits excède celui de ses débits (solde positif), et débiteur (négatif) dans le cas contraire.	Néant
Taux d'intérêt	Attribut	Taux	Pourcentage permettant de calculer la rémunération d'une somme d'argent pour une période donnée (jour, mois, année).	Pourcentage

Domaine:	Bancaire
Sous Domaine:	Gestion des comptes client
Activité:	Retrait d'argent

Mot	Nature	Forme abstraite	Explication	Synonyme
Agence	Acteur	-	Lieu d'accueil de la clientèle d'une banque.	Branche
Autorisation de découvert	Opération	Autoriser	Accord donné par la banque permettant de bénéficier d'un découvert d'un montant maximum déterminé et remboursable selon des modalités convenues d'avance, notamment dans la convention de compte de dépôt ou dans un contrat.	Accorder
Autorisation de prélèvement	Opération	Autoriser	Autorisation donnée par le client à sa banque de payer les prélèvements qui seront présentés par une société ou des créanciers désignés par l'autorisation.	Accorder
Banque	Acteur	-	Établissement autorisé par la loi à assurer des opérations de banque c'est-	Néant

			à-dire la réception de fonds du public, les opérations de crédit, ainsi que la mise à la disposition de la clientèle ou la gestion de moyens de paiement. Le terme « établissement de crédit » ou « caisse » est également utilisé.	
Carte bancaire	Objet	Carte	Moyen de paiement prenant la forme d'une carte émise par un établissement de crédit et permettant à son titulaire, conformément au contrat passé avec sa banque, d'effectuer des paiements et /ou des retraits. Des services connexes peuvent y être associés (assurance, assistance).	Néant
Carte de crédit	Objet	Carte	Carte de paiement permettant à son titulaire de régler des achats et/ou d'effectuer des retraits au moyen d'un crédit préalablement et contractuellement défini.	Néant
Carte de retrait	Objet	Carte	Carte délivrée par la banque permettant d'effectuer exclusivement des retraits de billets dans des automates bancaires (DAB/GAB). Son utilisation peut être limitée ou non à un seul guichet bancaire, à une seule banque ou à une seule agence.	Néant
Chèque	Objet	Chèque	Moyen de paiement normalisé avec lequel le titulaire (tireur) d'un compte donne l'ordre à son banquier (tiré) de payer au bénéficiaire du chèque la somme inscrite sur celui-ci. La provision doit être disponible lors de l'émission du chèque et maintenue jusqu'à sa présentation.	Néant
Chèque de banque	Objet	Chèque	Chèque émis par une banque à la demande du client, et dont le montant, immédiatement débité du compte de dépôt du client, est ainsi garanti. Les chèques de banque sont souvent exigés pour le règlement d'achats importants.	Néant
Chèque sans provision	Objet	Chèque	Chèque émis sur un compte de dépôt dont le solde disponible ou le découvert autorisé est insuffisant pour régler le montant du chèque. L'émetteur se voit interdit d'émettre des chèques jusqu'à ce qu'il régularise sa situation. L'émission d'un chèque sans provision entraîne des frais bancaires et éventuellement des pénalités à payer au Trésor Public (si la régularisation n'a pas lieu dans un délai de 2 mois après l'interdiction).	Néant

			La provision doit être disponible dès l'émission du chèque et maintenue jusqu'à sa présentation.	
Chéquier	Objet	Chéquier	Carnet comportant généralement 25 à 30 formules de chèques (ou « vignettes »). Certaines banques donnent le choix du format du carnet.	Carnet
Client	Acteur	Client	Propriétaire d'un ou plusieurs comptes	Client distant
Client distant	Acteur	Client	Personne autorisée d'accéder à ses comptes depuis l'Internet.	Client
Commission	Attribut	Commission	Somme perçue par une banque en rémunération d'un service fourni à son client.	Rémunération Remise Intérêt
Compte de dépôt	Objet	Compte	Compte bancaire ordinaire (ou compte courant) utilisé pour gérer quotidiennement son argent. C'est sur ce compte qu'un client dispose en général d'une carte bancaire et d'un chéquier. Le compte doit être créditeur, sauf accord avec la banque.	Néant
Compte joint	Objet	Compte	Compte de dépôt ouvert au nom de deux (ou plusieurs) personnes qui ont chacune le droit de disposer seule de l'avoir du compte. Dans le cas d'un compte joint entre époux, le compte n'est pas bloqué en cas de décès d'un des co-titulaires.	Néant
DAB	Objet	DAB	Distributeur Automatique de Billets. Appareil qui permet de retirer une somme d'argent du solde du compte bancaire à l'aide d'une carte bancaire et d'un code confidentiel, dans des limites fixées à l'avance contractuellement.	GAB
Date d'effet	Attribut	Date	Date d'entrée en vigueur d'un contrat, d'un avenant ou d'une garantie pouvant correspondre à la date du paiement de la première cotisation ou du premier versement.	Néant
Date d'opération	Attribut	Date	Date à laquelle l'opération est effectuée par le client.	Néant
Date comptable	Attribut	Date	Date à laquelle la banque enregistre comptablement l'opération sur le compte du client.	Néant
Date de valeur	Attribut	Date	Date de référence qui sert au calcul des intérêts créditeurs ou débiteurs. La Société Générale n'applique pas de dates de valeur, à l'exception des remises de chèques pour lesquelles une date de valeur de 2 jours ouvrés est prise en compte pour le calcul des intérêts en raison des délais techniques d'encaissements.	Néant
Débit	Opération	Débiter	Opération comptable qui diminue le	Retirer

			solde du compte, par exemple à la suite de l'émission d'un chèque, d'un prélèvement ou d'un retrait d'espèces à un DAB.	Prélever
Débiteur	Acteur	Débiteur	Personne physique ou morale tenue de remplir une obligation. Le plus souvent, il s'agit de payer une somme d'argent à un créancier. Un compte de dépôt est dit débiteur lorsque son solde est négatif.	Client
Débiteur (adjectif)	Objet	Débiteur	Position d'un compte de dépôt dont le solde est négatif, ou adjectif qualifiant des intérêts (intérêts débiteurs).	Néant
Découvert du compte	Attribut	Solde	Position d'un compte de dépôt lorsque son solde est négatif. Cette situation peut avoir été contractualisée (autorisation de découvert) préalablement ou non par le banquier.	Néant
Dépassement	Opération	Dépasser	Fait d'excéder le montant d'un seuil (plafond autorisé) ; par exemple découvert du compte ou seuil de retrait d'espèces autorisé par carte bancaire.	Néant
Echéance	Attribut	Echéant	Date à laquelle un engagement doit être exécuté ou date qui marque la fin d'un contrat.	Date
Emission de chèque	Opération	Emettre	Signature du chèque par son titulaire et remise ou envoi à son bénéficiaire.	Signer
GAB	Objet	GAB	Guichet Automatique de banque. Appareil qui permet à l'aide d'une carte bancaire et de son code confidentiel d'effectuer un certain nombre d'opérations sur un compte (retrait d'argent, consultation du compte, commande de chéquier,...), contrairement au DAB qui ne permet que des retraits de billets.	DAB
Intérêts débiteurs	Attribut	Intérêt	Somme due à la banque lorsqu'un compte présente un solde négatif pendant un ou plusieurs jours. Le calcul de cette somme tient compte des dates de valeur.	Pourcentage
Opérateur	Acteur	Opérateur	Employé de la banque pour servir les clients de leurs opérations comptables.	Guichetier Agent Employé Utilisateur
Opposition carte par la banque	Opération	Opposer	Opération par laquelle la banque refuse toute transaction en cas d'utilisation abusive d'une carte par le titulaire de la carte (client-porteur).	Refuser Rejeter
Opposition sur prélèvement	Opération	Opposer	Opération par laquelle le titulaire d'un compte donne l'ordre à sa banque, par courrier, Internet ou téléphone confirmé par courrier, de refuser à	Refuser Rejeter

			l'organisme émetteur la demande de(s) paiement(s) qu'il a présentée et pour laquelle une autorisation préalable de prélèvement avait été donnée.	
Prélèvement	Opération	Prélever	Opération qui permet à la banque, conformément à l'autorisation de prélèvement donnée par le client, de payer un créancier en débitant son compte de dépôt.	Retirer Débiter
Prélèvement impayé	Opération	Prélever	Rejet d'un prélèvement par la banque quand le solde disponible du compte est insuffisant pour le régler.	Retirer
Rejet de chèque	Opération	Rejeter	Refus de paiement, par la banque de l'émetteur, d'un chèque remis à l'encaissement par le bénéficiaire. Le refus est le plus souvent dû à un défaut ou à une insuffisance de provision.	Refuser Opposer
Rejet de prélèvement	Opération	Rejeter	Refus du paiement d'un prélèvement du fait d'une insuffisance de provision ou d'une opposition demandée par le client.	Refuser Opposer
Relevé de compte	Objet	Relevé	Document récapitulant les opérations enregistrées sur le compte d'un client pendant une période déterminée, généralement mensuelle. Il est conseillé de le conserver pendant 10 ans.	Néant
Remise de chèque(s)	Opération	Remettre	Dépôt de chèque(s) par le client auprès de sa banque pour encaissement. Elle nécessite la signature du bénéficiaire au dos du chèque (endos) ainsi que l'indication du numéro de compte à créditer.	Déposer
Retrait	Opération	Retirer	Opération par laquelle un client retire de son compte, au distributeur de billets ou au guichet, une certaine somme en espèces dont le montant est porté au débit de son compte.	Prélever
Solde du compte	Attribut	solde	Différence entre la somme des opérations au débit et au crédit d'un compte.	Néant
Taux d'intérêt	Attribut	Taux	Pourcentage permettant de calculer la rémunération d'une somme d'argent pour une période donnée.	Pourcentage

Annexe G

Interfaces Homme-Machine

Interfaces expert métier/système

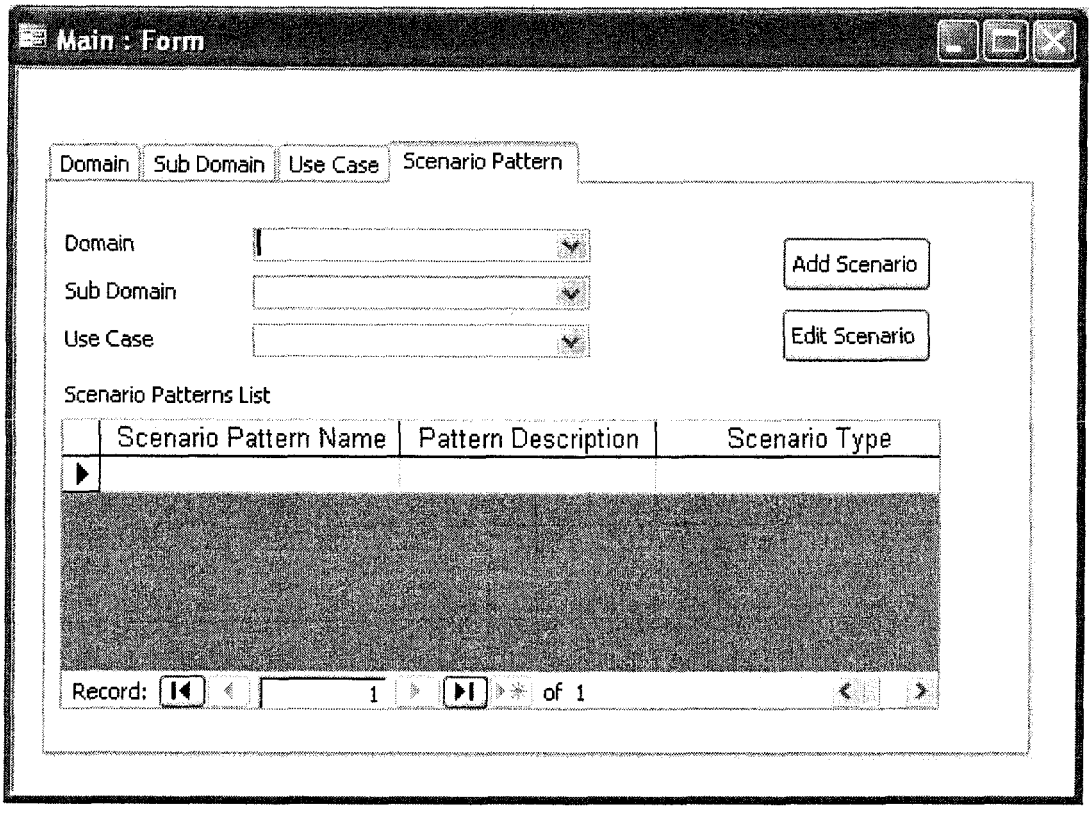


Figure g.1: Interface de création de l'ensemble des scénarios patterns

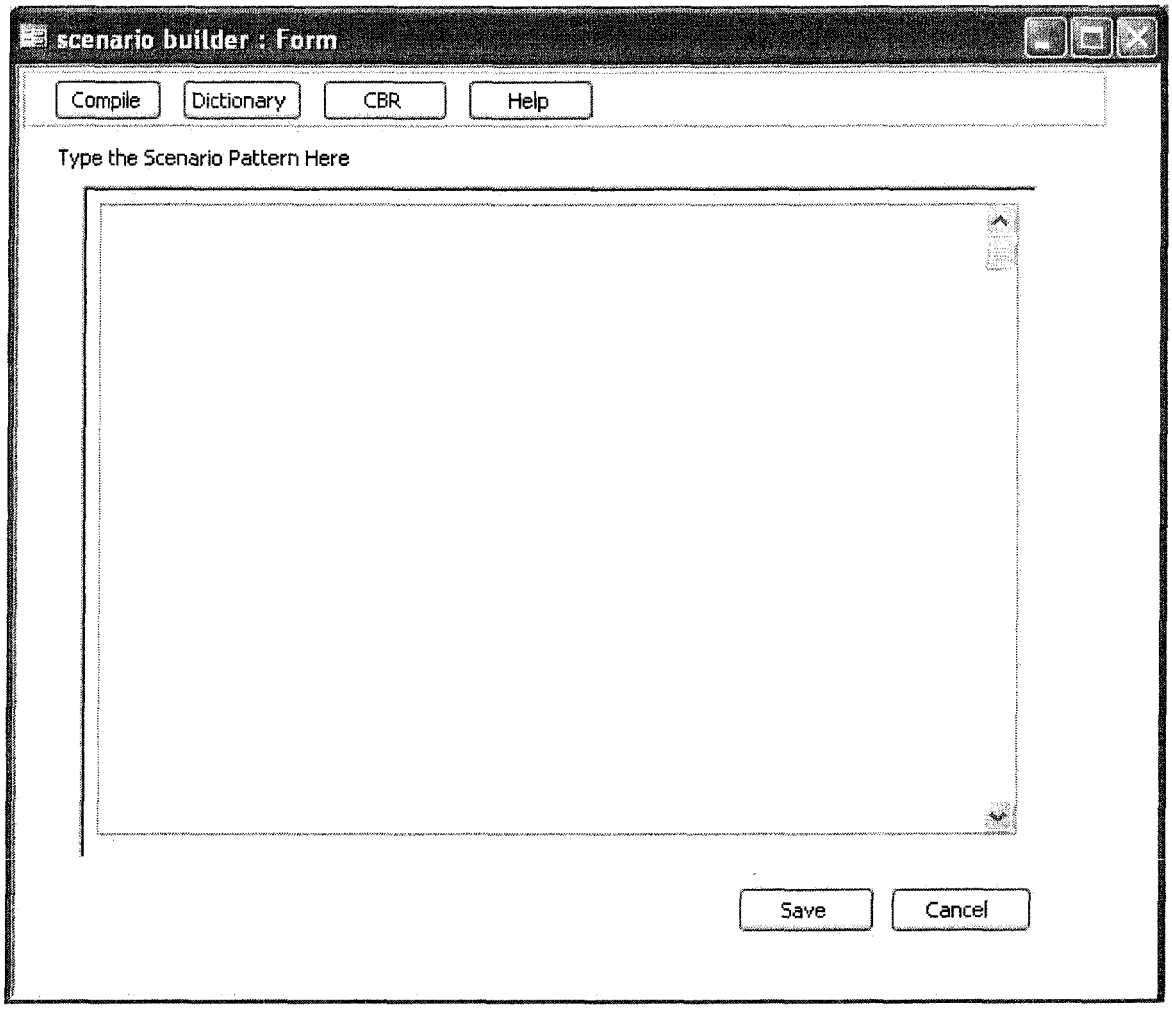


Figure g.2: Interface de rédaction d'un scénario pattern en langage LSP

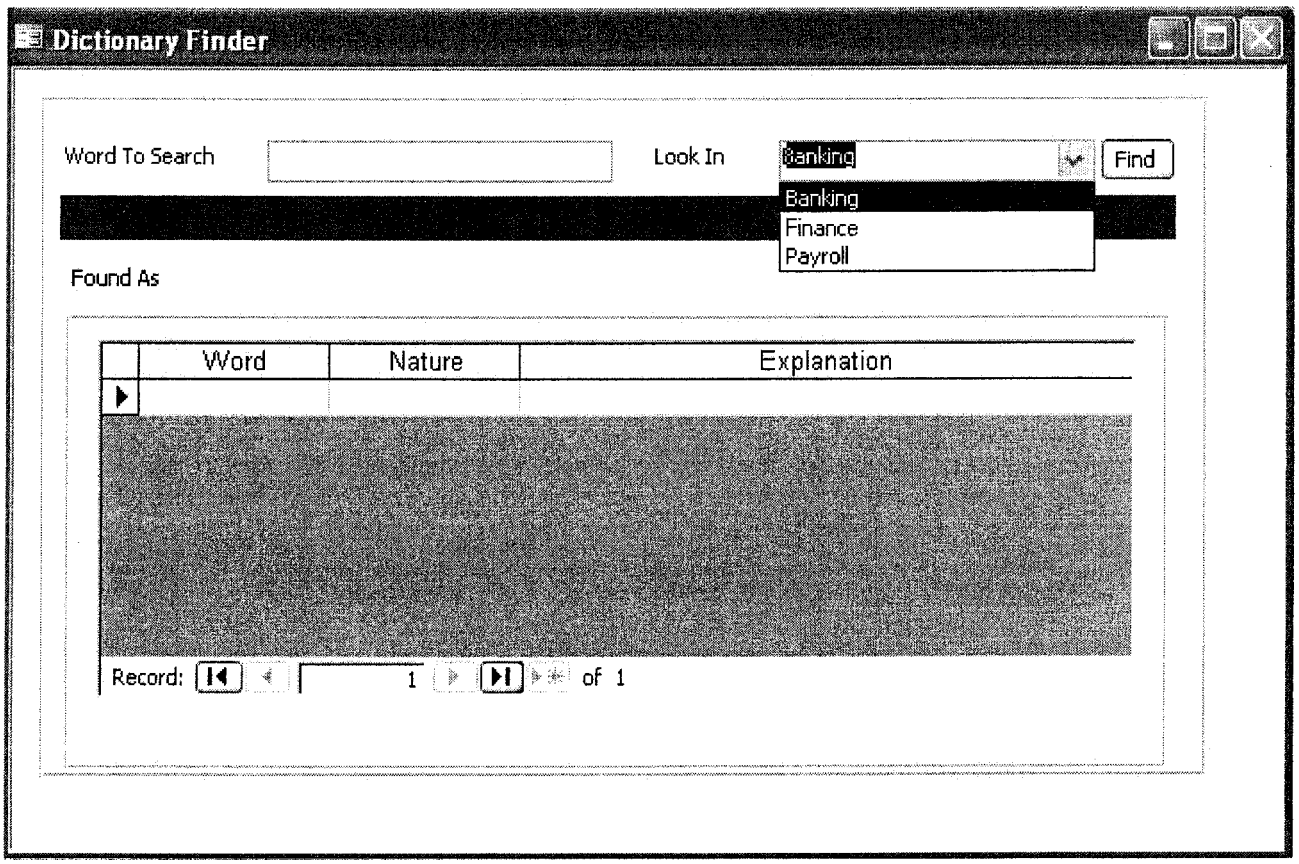


Figure g.3: Interface de consultation du dictionnaire métier

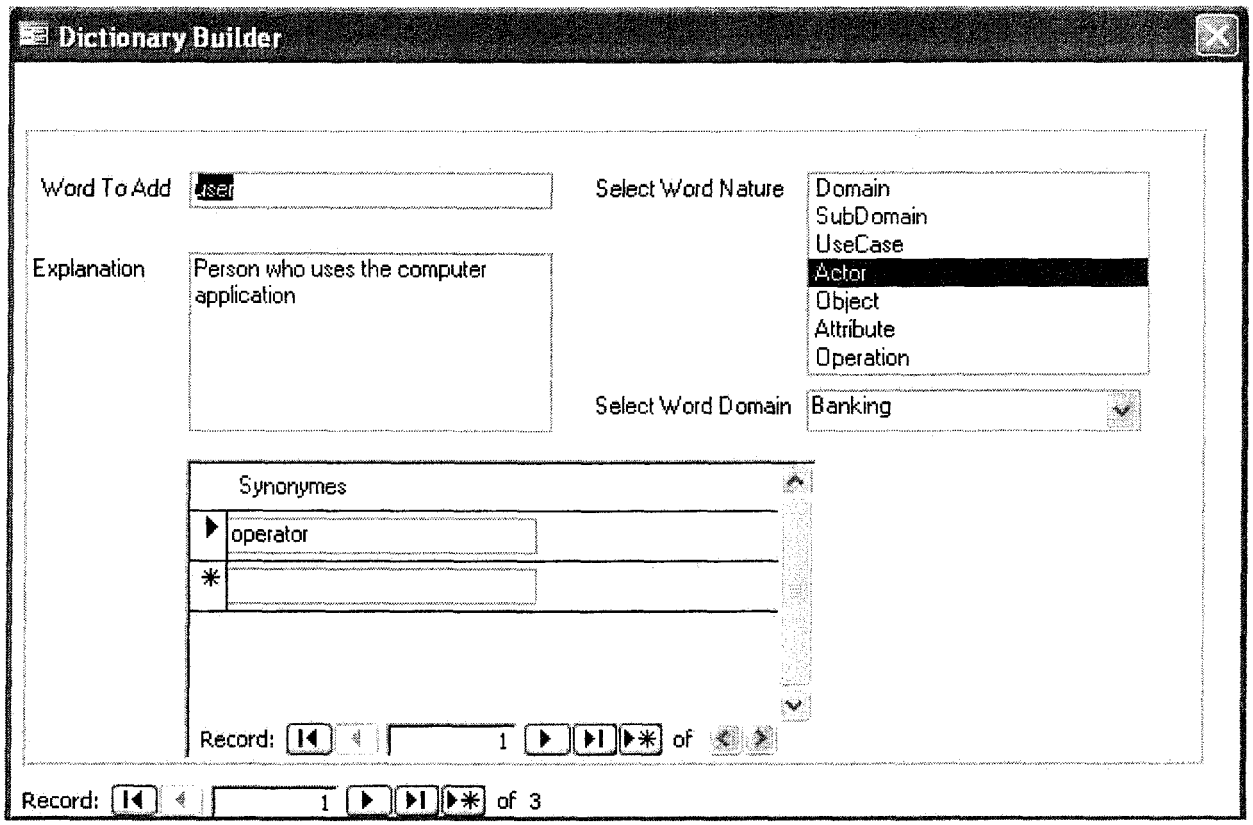


Figure g.4: Interface de remplissage du dictionnaire métier

Interface expert de conception/système

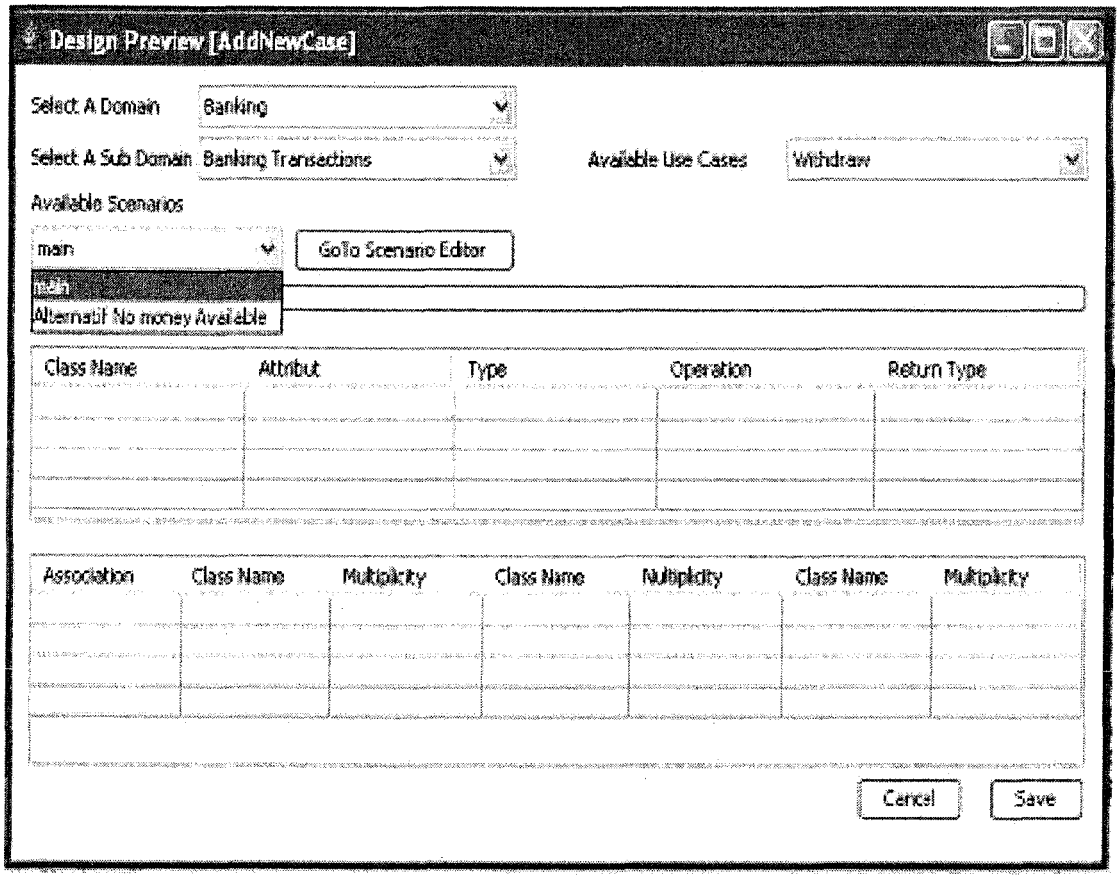


Figure g.5: Interface d'édition des patterns de conception

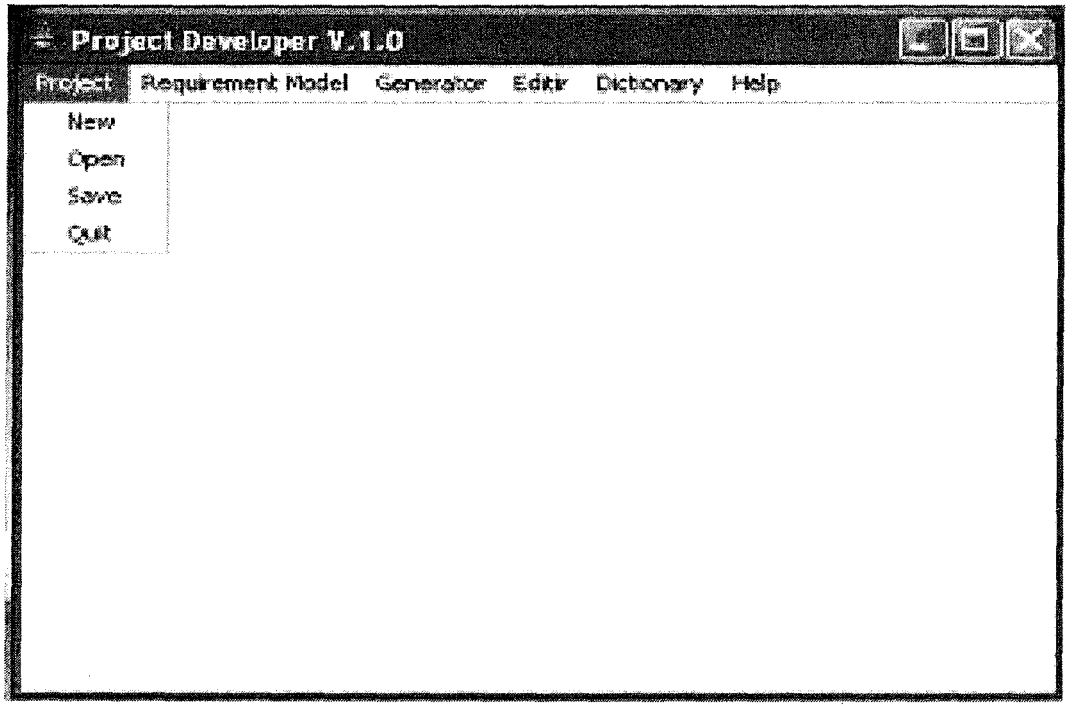


Figure g.6 : Interface principale du « Project Developer V.1.0 »

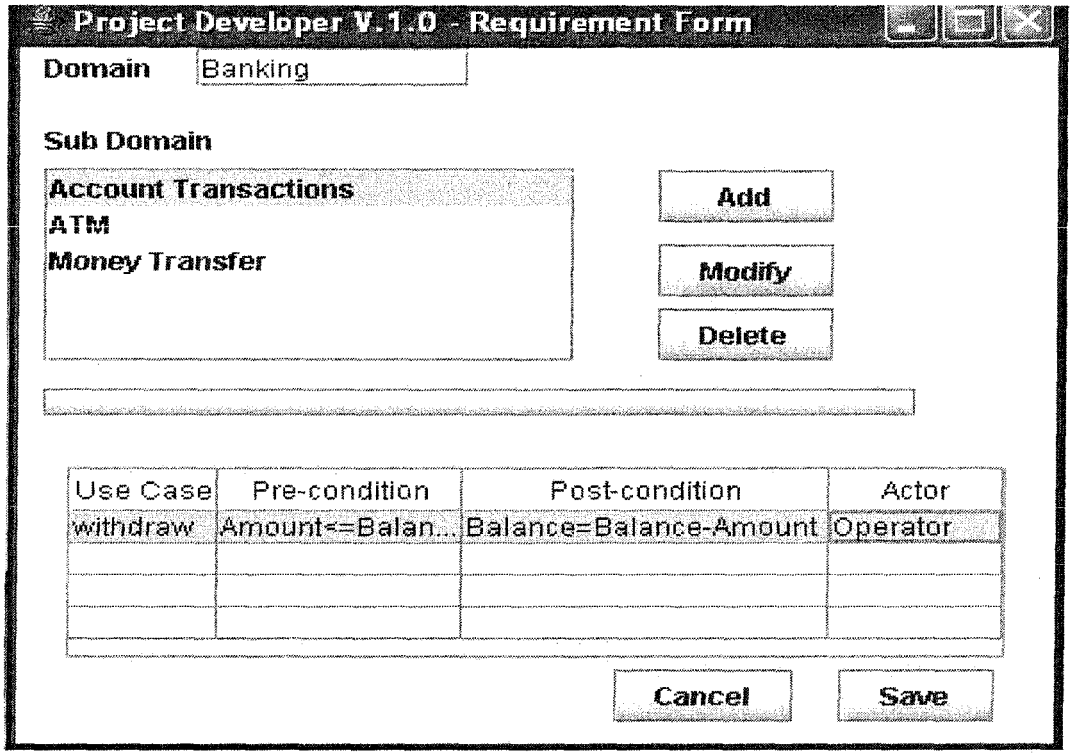


Figure g.7: Ecran de saisie du modèle des besoins

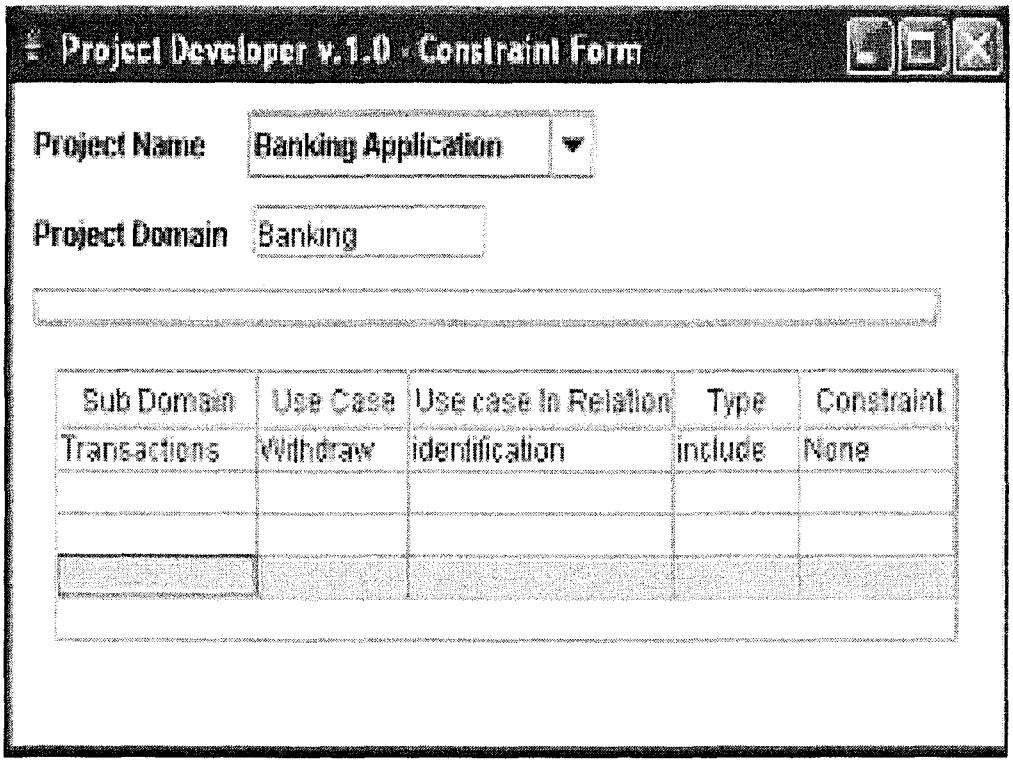


Figure g.8: Ecran de saisie des contraintes des besoins

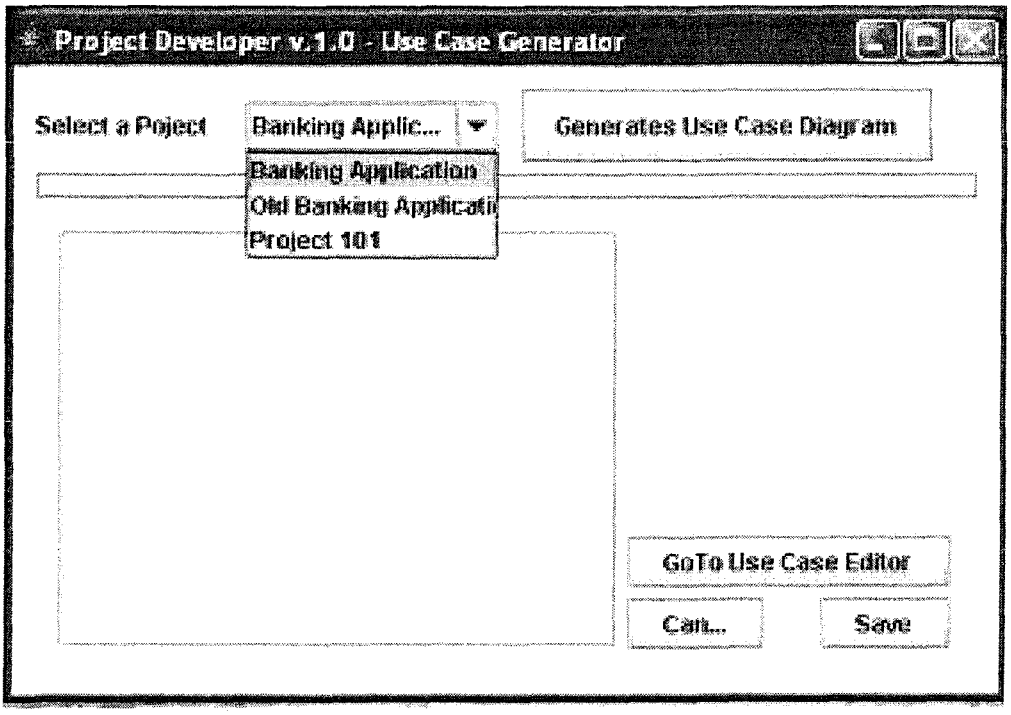


Figure g.9: Interface du générateur du diagramme de cas d'utilisation

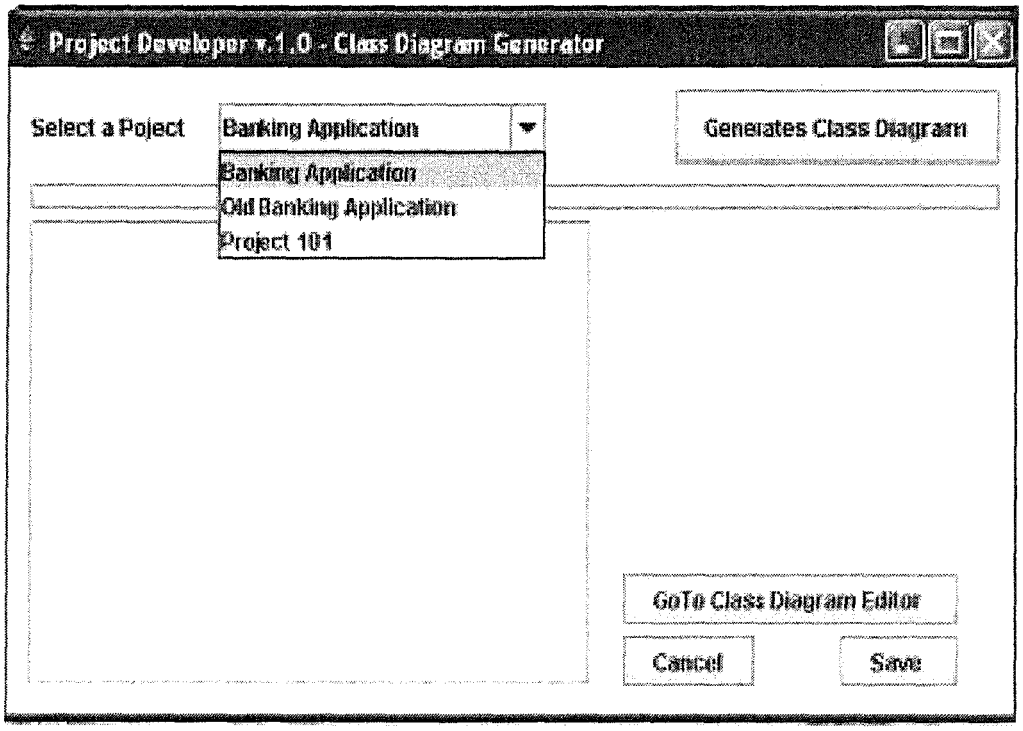


Figure g.10: Interface du générateur du diagramme de classes

Annexe H

Algorithmes de recherche et de test de similarité

Procédure Filtrage (domaine_c, sous domaine_c)
Début
Object [ ] [7] Liste_cas_utilisation_source;
J=0 ;
Tant que Index (EG) <> Null
Lire (EG)
Soit « noms » le nom du cas d'utilisation source;
Soit acteurs_s la référence de la liste des acteurs du cas source;
Soit cas_dépendants_s la référence de la liste des cas dépendants du cas source;
Soit type_dépendants_s la référence de la liste des types des cas dépendants du cas source;
Soit contrainte_dépendants_s la référence de la liste des contraintes des cas dépendants du cas source;
Soit preconditions_s la référence de la liste des pré-conditions du cas source;
Soit postcondition_s la référence de la liste des pré-conditions du cas source ;
Liste_cas_utilisation_source [j][0]=noms;
Liste_cas_utilisation_source [j][1]=acteurs_s;
Liste_cas_utilisation_source [j][2]=Cas_dépendants_s;
Liste_cas_utilisation_source [j][3]=type_dépendants_s;
Liste_cas_utilisation_source [j][4]=Contrainte_dépendants_s;
Liste_cas_utilisation_source [j][5]=preconditions_s;
Liste_cas_utilisation_source [j][6]=postconditions_s;
J++ ;
FinTantque ;
Fin.

Procédure Renvoyer_cas_similaires (Liste_cas_utilisation_source, Liste_cas_similaires)
Début
Soit φ1,φ2,φ3,φ4,φ5, φ6 φ7, les similarités, entre deux valeurs du descripteur *d,* associées au cas cible et au cas source ;
Soit Wi le coefficient d'importance du descripteur *d.* Les descripteurs les plus importants doivent être pris en considération, dans le calcul de la similarité, avec des valeurs plus élevées que d'autres descripteurs moins importants ;
Soit liste_cas_similaires, la liste des cas similaires et retenus ;
Pour tout nom de cas d'utilisation cible
Faire
Soit « nomc » le nom du cas d'utilisation dans le diagramme de cas d'utilisation cible ;

Soit « acteurs_c », la liste des acteurs du cas d'utilisation cible ;
Soit « cas_dépendants_c », la liste des cas d'utilisation en dépendance avec nomc ;
Soit « Type_cas_c », la liste du type de relation des cas dépendants du cas cible;
Soit « contrainte_cas_c », la liste des contraintes des cas dépendants du cas cible;
Soit « précondition_c », liste des pré-conditions du cas d'utilisation ;
Soit « postcondition_c », liste des post-conditions du cas d'utilisation ;
Lire (nomc) ;
Si nomc existe dans liste_cas_utilisation_source
Alors
  Evaluer_descripteurs (nomc, acteurs_c, cas_dependants_c, type_cas_c,
                        contrainte_cas_c, précondition_c, postcondition_c) ;
Sinon
    Chercher dans le dictionnaire les synonymes du « nomc » ;
    Soit « noms » la liste des synonymes;
    Pour tout nom de cas d'utilisation source « noms »
    Faire
        Evaluer_descripteurs (nomc, acteurs_c, cas_dependants_c, type_cas_c,
                              contrainte_cas_c, précondition_c, postcondition_c) ;
    FinFairePour ;
Fin.

Procédure évaluer_descripteurs (nomc, acteurs_c, cas_dependants_c, type_cas_c, contrainte_cas_c, precondition_c, postcondition_c)

Début
    $\varphi 1$=m ;
    Soit acteur_s, la liste des acteurs du cas source ;
    Lire(acteurs_s) ;
    $\varphi 2$=φ_acteurs(acteur_c, acteurs_s) ;
    $\varphi 3$=φ_cas_dependants (cas_dépendants_c, cas_dépendants_s) ;
    $\varphi 4$=φ_ type (type_cas_c, type_cas_s) ;
    $\varphi 5$=φ_contrainte (contrainte_cas_c, contrainte_cas_s);
    $\varphi 6$=φ_précondition (précondition_c, précondition_s)
    $\varphi 7$=φ_postcondition (postcondition_c, postcondition_s)
    ϕ = (W1*φ1 + W2*φ2 + W3*φ3 + W4*φ4 + W5*φ5 +W6*φ6+W7*φ7)/ (W1+ W2+ W3+ W4+W5+W6+W7)
    Calculer le seuil dynamique du cas cible en fonction du nombre de descripteurs;
    Soit S ce seuil,
    Calculer la distance entre le seuil et la fonction ϕ,
    Soit D cette distance, D=S - ϕ,
        Si D>=0 et D<=S-Min (S)
        alors
        Ajouter le cas d'utilisation source à la liste_cas_similaires;
        Sinon
        afficher «Cas d'utilisation cible n'est pas identifié par l'EMI » ;
        finsi ;
Fin.

Procédure Renvoyer_cas_plus_similaires (Liste_cas_similaires, $D_{acceptable}$)

```
Début
    D_acceptable, cette variable est introduite par l'ingénieur système,
    Soit Liste_cas_plus_similaires, la liste des cas les plus similaires ;
    Tant que list_cas_similaires non vide
    Si D >= D_acceptable
    Alors
        Ajouter cas similaire à la Liste_cas_plus_similaires ;
    Finsi ;
    FinTantque ;
Fin.
```

Procédure sélectionner_scénarios_patterns (Liste_cas_plus_similaires, List_SP)

```
Début
    Soit List_SP, la liste des scénarios patterns relatifs aux cas retenus,
    Pour chaque cas_plus_similaire Faire
        Sélectionner scénarios patterns
        Ajouter les scénarios sélectionnés à la List_SP,
    FinPour;
Fin.
```

Fonction φ_acteurs (chaîne [ ] acteurs_c, acteurs_s) : réel ;

```
Début
    acteurs=0 ;
    valeur=0 ;
    Pour i allant de 0 à [longueur (acteurs_c) -1]
    Faire
        J=0 ;
        Tant que j<=[longueur (acteurs_s) -1] et non trouvé
        Faire
            Si acteurs_c[i]=acteurs_s[j] ou synonyme(acteurs_c[i])=acteurs_s[j]
            Alors
                trouvé=vrai ;
                acteurs=acteurs+1 ;
            Finsi ;
        FinTantque ;
    FinFairePour ;
    Si i<>0 et acteurs=i Alors valeur=1  FinSi;
    Retourner valeur;
Fin.
```

```
Fonction φ_cas_dependants (chaîne [ ] cas_dépendants_c, cas_dépendants_s) : réel;
Début
        dependants=0 ;
        valeur=0 ;
        Pour i allant de 0 à [longueur (cas_dépendants_c) -1]
        Faire
                J=0 ;
                Tant que  j<=[longueur (cas_dépendants_s) -1] et non trouvé
                Faire
                        Si (cas_dépendants_c[i]=cas_dépendants_s[j] OU
                                synonyme(cas_dépendants_c[i])=cas_dépendants_s[j])
                        Alors
                                trouvé=vrai ;
                                dependants=dependants+1;
                        Finsi ;
                FinTantque ;
        FinFairePour ;
        Si i<>0 et dependants=i Alors valeur=1 FinSi;
        Retourner valeur;
Fin.
```

```
Fonction φ_type (chaîne [ ] type_cas_c, type_cas_s) : réel;
Début
        type=0;
        valeur=0 ;
        Pour i allant de 0 à [longueur (type_cas_c) -1]
        Faire
                J=0 ;
                Tant que  j<=[longueur (type_cas_s) -1] et non trouvé
                Faire
                Si (type_cas_c[i]=type_cas_s[j] OU
                                        synonyme (type_cas_c[i])=type_cas_s[j])
                Alors
                        trouvé=vrai ;
                        type=type+1;
                Finsi ;
                FinTantque ;
        FinFairePour ;
        Si i<>0 et type=i Alors valeur=1 FinSi;
        Retourner valeur;
Fin.
```

```
Fonction φ_contrainte (chaîne [ ] contrainte_cas_c, contrainte_cas_s) : réel;
Début
        Contrainte=0;
        Pour i allant de 0 à [longueur (contrainte_cas_c) -1]
        Faire
                J=0 ;
                Tant que  j<=[longueur (contrainte_cas_s) -1] et non trouvé
                Faire
                        Si (contrainte_cas_c[i]=contrainte_cas_s[j]
                                OU synonyme(contrainte_cas_c[i])=contrainte_cas_s[j])
                        Alors
                                trouvé=vrai ;
                                contrainte=contrainte+1;
                        Finsi ;
                FinTantque ;
        FinFairePour ;
        Si i<>0 et contrainte=i Alors valeur=1 FinSi;
        Retourner valeur;
Fin.

Fonction φ_précondition (chaîne [ ] précondition_c, précondition_s): réel;
Début
        pre=0 ;
        Pour i allant de 0 à [longueur (précondition_c) -1]
        Faire
                J=0 ;
                Tant que  j<=[longueur (précondition_s) -1] et non trouvé
                Faire
                        Si précondition_c[i]= précondition_s [j]
                                ou synonyme(précondition_c[i])= précondition_s[j]
                        Alors
                                trouvé=vrai ;
                                pre=pre+1 ;
                        Finsi ;
                FinTantque ;
        FinFairePour ;
                Si i<>0 et pre=i Alors valeur=1  FinSi;
        Retourner valeur;
Fin.


Fonction φ_postcondition(chaîne [ ] postcondition_c, postcondition_s): entier ;
Début
        post=0 ;
        Pour i allant de 0 à [longueur (postcondition_c) -1]
        Faire
```

```
            J=0 ;
            Tant que  j<=[longueur (postcondition_s) -1] et non trouvé
            Faire
                  Si postcondition_c[i]= postcondition_s [j]
                        ou synonyme(postcondition_c[i])= postcondition_s[j]
                  Alors
                        trouvé=vrai ;
                        post=post+1 ;
                  Finsi ;
            FinTantque ;
      FinFairePour ;
      Si i<>0 et post=i Alors valeur=1  FinSi;
      Retourner valeur ;
Fin.
```

Références bibliographiques

[Aamodt et Plaza 94] A. Aamodt, E. Plaza,
« Case-based reasoning: Foundational issues, methodological variations, and system approaches ». *AICOM*, **7**, pp. 39–59, 1994.

[Aamodt 91] A. Aamodt,
« A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning », PhD thesis, University of Trondheim, Norwegian Institute of Technology, Department of Computer Science, 1991.

[Amyot et al. 97] D. Amyot, L. Logrippo, R.J.A. Buhr
« Spécification et conception de systèmes communicants: une approche rigoureuse basée sur des scénarios d'usage ». CFIP 97, Ingénierie des protocoles, 1997.

[Arsajani et Alpigini 01] A. Arsanjani, J. Alpigini,
"Using Grammar-oriented Object Design to Seamlessly Map Business Models to Component-based Software Architectures", In Procs of the International Symposium of Modelling and Simulation, Pittsburgh, PA, USA, pp 186-191, May 16-18, 2001

[Bisson 00] G. Bisson,
« La similarité : une notion symbolique/numérique. Apprentissage symbolique-numérique (tome 2) ». In CEPADUES Ed. pp. 169–201, 2000.

[Booch 94] G.Booch,
« Analyse et conception orientées objet », Addison Wesly – 2me édition, 1994.

[Booch 94] G. Booch
« Analyse et Conception Orientées objets », S.A France, Addison-Wesley, octobre 1994.

[Brunet et *al.* 90] E. Brunet, G. Dorbes,
« KADS et Merise : vers une unification du génie cognitif et du génie logiciel et systèmes experts », no 19, publication EC2, 1990.

[Bass et *al.* 98] L. Bass, P. Clements, R. Kazman,
« Software Architecture in Practice », Addison-Wesley, 1998.

[Chan et Wu 02] L-K. Chan, M-L. Wu
« Quality function deployment » : A literature review. European of Operational Research, Volume 143, Issue 3, pp. 463-497, 2002.

[Charroux, Osmani et Thierry-mieg 05] B.Charroux, A Osmani, Y. Thierry-mieg,
« UML 2 », Collection Synthex, Pearson Education, France, 2005.

[Christel et Kang 92] M.G. Christel, K.C. Kang,
"Issues in Requirements Elicitation", Technical report. CMU/SEI-92-TR-12, September 1992.

[Cian 03] Cian,
« Diagramme de séquence nouveautés UML 2.0 : Diagramme de séquence », 31/12/2003, UML 2.0, www.developpez.com, 2003.

[Coad et Yourdon 91] P. Coad, E. Yourdon,
« Object-Oriented Analysis », 2nd edition Yourdon Press, Englewood Cliffs, N.J., 1991.

[Cockburn 97] A. Cockburn,
« Structuring Use Cases with Goals », members.aol.com/acockburn/papers/usecases.htm, 1997.

[Cokbun 00] A. Cockburn,
« Writing Effective Use Cases », Addison-Wesley, ISBN 0201702258, 2000.

[Coo 99] Cooperative Requirements Engineering With Scenarios,
« Reports series 1996, 1997, 1998, 1999 ». http://SunSITE.Informatik.RWTH-Aachen.DE/CREWS, 1999.

[Cordier et Fuchs 05] A. Cordier, B. Fuchs,
« Un assistant pour la conception et le développement des systèmes de RàPC » , 13ème Atelier RàPC, 2005.

[Councill et Heineman 01] W.T Councill, G.T. Heineman,
« Definition of a Software Component and its elements », In component-Based Software Engineering", Boston: Addison-Wesley 5-20 (ch. 19), 2001.

[Curtis et *al.* 88] B. Curtis, H. Krasner, N. Iscoe
« A Field Study of the Software Design Process for Large Systems », Comm. ACM 31 (11), pp. 1268-1287, 1988.

[Dahl et *al.* 72] O. Dahl, E. Dijkstra, C. Hoare,
« Structured Programming », Londres, Angleterre : Academic Press, 1972.

[Delange et Vogin 94] L. Delang, R. Vogin,
« La croissance de sûreté de fonctionnement par le retour d'expérience dans le domaine technique et industriel ». Performances Humaines et Techniques, 69, pp. 14-20, Mars-Avril 1994.

[De Macro 79] T. De Macro
« Structured Analysis and System Specification », Englewood Cliffs, Prentice Hall, 1979.

[Djaiz et Matta 06] C. Djaiz, N. Matta
« Stratégies de regroupement des connaissances dans les mémoires projet », Colloque CITE, 2006.

[Dieng et *al.*, 01] Dieng-Kuntz R., Corby O., Gandon F., Giboin A., Golebiowska J., Matta N., Ribière M.
« Méthodes et outils pour la gestion des connaissances ; une approche pluridisciplinaire du Knowledge Management », 2e édition, Dunod, Paris, 2001.

[Elorriaga et Fernandez-castro 00] J. Elorriaga, I. fernandez-castro,
« Using case-based reasoning in instructional planning : towards a hybrid self-improving instructional planner ». In International Journal of Artificial Intelligence in Education, p. 416–449, 2000.

[Ewald 01] T. Ewald
« Overview of COM+. In component-Based Software Engineering », (W.T Councill, G.T. Heineman) Boston: Addison-Wesley, pp. 573-588 (ch. 19), 2001.

[Fox et Leake 94] S. Fox, D.B Leake,
« Using introspective reasoning to guide index refinement in case-based reasoning », In Proceedings of the sixteenth annual Conference of the Cognitive Science Society, 1994.

[Frey, Gomes et Sagot 07] E. FREY, S. GOMES, J. SAGOT
« Application de la méthode QFD comme outil d'extraction des connaissances métier en conception intégrée » projet CoDeKF, mai 2007.

[Fuchs et *al.* 00] B. Fuchs, J. Lieber, A. Mille. A. Napoli
« An Algorithm for Adaptation in Case-Based Reasoning », In Proceedings of the 14th European Conference on Artificial Intelligence, ECAI 2000, pp. 45–49, Berlin: IOS Press, 2000.

[Garlan et Shaw 93] D. Garlan , M. Shaw,
« An introduction to software architecture » . Dans V. Ambriola et G. Tortora, editeurs, Advances in Software Engineering and Knowledge Engineering, volume 1, World Scienti_c Publishing Company, 1993.

[Glossaire SG 08] Glossaire de la banque Société Générale de France, http://particuliers.societegenerale.fr/, 2008

[GoF 94] E. Gamma, R. Helm, R. Johnson, J. Vlissides,
« Design Patterns – Elements of Reusable Object-Oriented Software », Addison Wesley, 1994.

[GoF 99] E. Gamma, R. Helm, R. Johnson, J. Vlissides,
« Design Patterns - Catalogue de modèles de conception réutilisables », Vuibert, Paris, 1999.

[GoF *et al.* 95] E. Gamma, R. Helm, R. Johnson, J. Vlissides,
« Design Patterns, Elements of Reusable Object Oriented Software », Addison-Wesley, 1995.

[Grosz 94] G. Grosz,
« MENTOR : A Step Forward in Guidance for Information System Development », in the Proceedings of the Fifth Workshop on the Next Generation of Case Tools, B.Theodoulidis (ed), Utrech, the Netherlands, 6-7 June, 1994.

[Grosz et *al.* 96] G. Grosz, S. Si-Said, C. Rolland,
« MENTOR : Un Environnement Pour L'Ingénierie des Méthodes et des Besoins »,
14ème congrès INFORSID, Bordeaux, France, Juin 1996.

[Grundstein et *al.* 03] M. Grundstein, C. Rosenthal-Sabroux, A. Pachulski,
« Reinforcing decision aid by capitalizing on company's knowledge: future prospect », European Journal of Operational Research, 145, pp. 256-272, 2003.

[Harker et *al.* 93] S.D.P.Karker, K.D. Eason, J.E. Dobson,
« The Change and Evolution of Requirements as a Challenge to the Practice of Software Engineering », IEEE Symposium on Requirements Engineering, RE'93, San Diego, CA, Jan. 4-6, pp. 266-272, 1993.

[Hussein et *al.* 08] B. Hussein, A. Mhanna, P. Caulier, Y. Rabih,
« RÀPC au cœur du developpement intelligent des logiciels », 16ème atelier de Raisonnement à Partir de Cas, Nancy, France, 1-2 avril, 2008.

[Hussein et *al.* 06 (a)] B. Hussein, P. Caulier, Y. Monsef, P. Millot et Y. Rabih,
« Intelligent environment for rapid process design », EAM'06 European Annual Conference on Human Decision-Making and Manual Control, Sep 27-29, Valenciennes, France, 2006.

[Hussein et *al.* 06 (b)] B. Hussein, P. Caulier, Y. Moncef , P. Millot, Y. Rabih,
« Vers un environnement intelligent d'aide à la modélisation conceptuelle », 14^{e} Atelier de Raisonnement à Partir de Cas, Besançon, France, 30-31 mars, 2006.

[ITU 96] ITU,
« Recommendation Z ». 120: Message Sequence Chart (MSC), Geneva, 1996.

[Jacobson 93] I. Jacobson
« Le génie logiciel orienté objet ». Une approche basée sur les cas d'utilisation. Addison-Wesley, France, S.A, 1993.

[Jackson 01] M. Jackson,
« Problem Frames. Analyzing and structuring software development problems ». Addison-Wesley, 2001.

[Jackson 75] M. Jackson,
« Principles of Program Design », Orlando, FL: Academic Press, 1975.

[Jackson 83] M. Jackson,
« System Development », Englewood Cliffs, NJ: Prentice-Hall, 1983.

[Jacobson et *al.* 93] I. Jacobson, M. Christerson, P. Jonsson et G. Övergaard,
« Object-Oriented Software Engineering , A Use Case Driven Approach », Addison-Wesley, ACM Press, 1993.

[Jarke et *al.* 94] M. Jarke, K. Pohl, C. Rolland, J. R. Schmitt,
« Experienced-Based Method Evaluation and Improvement: A Process Modeling Approach », Int. IFIP WG8. 1 Conf. in CRIS series: Method and associated Tools for the Information Systems Life Cycle, North Holland (Pub.), 1994.

[Jarzombek et Stanley 99] Jarzombek, J. Stanley
« The 5th Annual Joint Aerospace Weapons Systems Support, Sensors, and Simulation Symposium » (JAWS S3), Proceedings, 1999.

[Jérôme Costanzo 00] J. Costanzo,
« Méthodes pour l'ingénierie du besoin complexe », mémoire DEA, 2000.

[Kadima 05] H. Kadima,
« MDA conception orientée objet guidée par les modèles », Dunod, Paris, 2005.

[Kolodner 93] J. Kolodner,
« Case-based reasoning ». In Morgan Kaufmann Pub., 1993.

[Leake 96] D.Leake,
« Cbr in context : The present and the future ». In Case-Based Reasoning: Experiences, Lessons and Future Directions, 1996.

[Ledang 01] H. Ledang
« Des cas d'utilisation à une spécification B », Proceedings of AFADL'2001, LORIA, Nancy, France, 11-13 June, 2001.

[Ledru, Padiou et Jaray 00] Y. Ledru, G. Padiou, and J. Jaray.
« Étude de cas: Système de contrôle d'accès », //www.lsr. imag.fr/afadl2000/EtudeDeCas/, 2000.

[Leishman et Cook 02] R. Leishman, A. Cook,
«Requirements Risks Can Drown Software Projects », Journal of Defense Software Engineering, Apr 2002.

[Lubars et *al.* 93] M. Lubars, C. Potts, C. Richer,
«A review of the state of the practice in requirements modeling », Proc. IEEE Symp. Requirements Engineering, San Diego, 1993.

[Mallet 05] Mallet,
« La construction de cas d'utilisation textuels », Centre d'Innovation par les Technologies de Réf.: USE_CASES_SPINOV Version: 2.0 , 2005.

[McGraw et Harbison 97] K. Mc Graw, K. Harbison,
« User Centered Requirements, The Scénario-Based Engineering Process », Lawrence Erlbaum Associates Publishers, 1997.

[Miller et Mukerji 03] J. Miller et J. Mukerji.
« MDA Guide Version 1.0.1 », Copyright © 2003 OMG, Document Number: omg/2003-06-01, 12 Juin, 2003

[Mills et *al.* 86] H. Mills, R. Linger, A. Hevner
« Principles of Information System Design and Analysis », Orlando, Fl: Academic Press, 1986.

[Myers 78] G. Myers,
« Composite/Structured design », New York, NY: Van Nostrand Reinhold, 1978.

[Nanci et Espinasse 96] D. Nanci, B. Espinasse,
« Ingénierie des systèmes d'information : Merise deuxième génération », Amsterdam, SYBEX, 1996.

[Nancy et Espinasse 96] D. Nancy, B. Espinasse,
« Ingénierie des systèmes d'informations », MERISE - Cybex troisième édition, 1996.

[Orr 71] K. Orr,
« Structured Systems Development », New York, NY: Yourdon Press, 1971.

[Page-Jones 88] M. Page-Jones,
« The practical guide to Structured Systems Design », Englewood Cliffs, NJ: Yourdon Press, 1988.

[Pham 89] T.Q. Pham,
« Merise et Yourdon, génie logiciel et systèmes experts », no 15, publication EC2, 1989.

[Plihon 96] V. Plihon,
« Un environnement pour l'ingénierie des méthodes », Thèse de doctorat, Janvier 1996.

[Prat 99] N. Prat,
« Réutilisation de la trace par apprentissage dans un environnement pour l'ingénierie des processus », Thèse de doctorat, février 1999.

[Ralyté et Ben Achour 97] J. Ralyté, C. Ben Achour,
« Scenario Integration into Requirements Engineering Methods », CREWS Report Series 97-08, Proceedings of the Workshop on the Many Facets of Process Engineering (MFPE'97), Gammarth, Tunis, September 22-23, 1997.

[Ramadour et Cauvet 02] P. Ramadour, C. Cauvet,
« Approche et modèle pour la spécification de composants-métier », Revue ISI "Ingénierie des Systèmes d'Information", numéro spécial "Connaissances métier dans l'Ingénierie des Systèmes d'Information", volume 7, numéro 4, Hermès, 2002.

[Rapicault 99] P. Rapicault,
« Vers une meilleure intégration des designs patterns à la conception », Journée GRACQ, LIP6, 7 juin, 1999.

[Renaud et *al.* 08] J. Renaud, B. Morello, B. Fuchs, J. Lieber,
« Raisonnment à partir de cas 1 : conception et configuration de produits », Traité IC2, série Informatique et systèmes d'information, Lavoisier, 2008.

[Rifqi 96] M. Rifqi,
« Mesures de comparaison, typicalité et classification d'Objets flous : théorie et pratique », Thèse d'université, Université Pierre et Marie Curie, Paris VI, Paris, 1996.

[Rolland et *al.* 98] C. Rolland, C. Souveyet, C. Ben Achour,
"Guiding goal modelling using scenarios", IEEE Transactions on Software Engineering, Special Issue on Scenario Management, Vol. 24, No. 12, December 1998.

[Rolland et *al.* 99a] C. Rolland, G. Grosz, V. Plihon,
« Process Modelling, in The NATURE of Requirements Engineering », Shaker Verlag, Aachen, pp. 175-200, 1999.

[Rolland et *al.* 99b] C. Rolland, V. Plihon, S. Si-Said,
« Engineering Process in The NATURE of Requirements Engineering », Shaker Verlag, Aachen, p. 201-230, 1999.

[Rolland et Ben Achour 97] C. Rolland, C. Ben Achour,
"Guiding the construction of textual use case specifications", Data & Knowledge Engineering Journal Vol. 25 N° 1, pp. 125-160, (ed. P. Chen, R.P. van de Riet) North Holland, Elsevier Science Publishers, March 1997.

[Rolland et Prakash 96] C. Rolland, N. Prakash,
« A proposal for context-specific method engineering »; Proc. IFIP WG8.1 Int. Conf. on "Method Engineering", Chapman&Hall (Pub.), Atlanta, USA, August 1996.

[Rolland et Schwer 99] C. Rolland, S. Schwer,
« Theoritical Formalisation of the Meta-Modelling Approach », in "The NATURE of Requirements Engineering", Shaker Verlag, Aachen, pp. 361-376, 1999.

[Roques et Vallée 03] P. Roques, F. Vallée,
« UML en action », Eyrolles, 2003.

[Ruet 02] M. Ruet,
« Capitalisation et réutilisation d'expériences dans un contexte multi acteurs », Thèse au Laboratoire Génie de Production de l'Ecole Nationale d'Ingénieurs de Tarbes, 2002.

[Rumbaugh, Jacobson et Booch 98] J. Rumbaugh, I. Jacobson, G. Booch,
« The Unified Modeling Language Reference Manual ». Addison-Wesley, ISBN 0-201-30998-X, 1998.

[Sánchez-Alonso, Murillo et Hernández 04] M. Sánchez-Alonso, J. Murillo, J. Hernández,
« COFRE: Environment for Specifying Coordination Requirements using Formal and Graphical Techniques1 », Journal of Research and Practice in Information Technology, Vol. 36, No. 4, November 2004.

[Schank 82] R. Schank
« Dynamic memory ; a theory of reminding and learning in computers and people ». In Cambridge University Press, 1982.

[Sehaba et Estraillier 05] K. Sehaba, P. Estraillier,
« Exécution adaptative par observation et analyse de comportement dans un contexte de jeu éducatif », Plate-forme AFIA / Nice, ATELIER : Raisonnement à Partir de Cas, 30 mai au 3 juin, 2005.

[Serrafero 02] P. Serrafero,
« Vers la mesure de quantité de connaissance et de compétence industrielle : le modèle KnoVA », Conférence invitée, 1er colloque du Groupe de Gestion des Compétences et des Connaissances en Génie Industriel, Nantes, 2002.

[SiSaid et *al.* 96] S. Si-Said, C. Rolland, G. Grosz,
« MENTOR : A Computer Aided Requirements Engineering Environment », in the Proceedings of the 8th CAISE Conference on Challenges In Modern Information Systems, Heraklion, Crete, Greece, May 1996.

[Si-Said96 et *al.* 96] S. Si Said, C. Rolland, G. Grosz,
« MENTOR : A Computer Aided Requirements Engineering Environment »; Proc. of the Int. Conference CAISE'96, Springer Verlag (pub), Heraklion, Greece, 1996.

[Smyth et Keane 93] B. Smyth, M.T Keane,
« Retrieving adaptable cases. the role of adaptation knowledge in case retrieval ». In M. M. RICHTER, S. WESS, K.-D. ALTHOFF & F. MAURER, Eds., *First European Workshop on Case-Based Reasoning - EWCBR-93*, pp. 209–220, Kaiserslautern, Germany: LNAI, vol. 837, Springer, Berlin, 1993.

[Smyth et Keane 95] B. Smyth, M.T Keane,
« Remembering to forget: A competence-preserving deletion policy for CBR systems », In 14th Joint Conference on Artificial Intelligence, IJCAI-95, pp. 377–382, Montréal, Canada, 1995.

[Sommerville 85] I. Sommerville
« Software engineering », Second Edition. Workingham, Angleterre: Addison-Wesley, p. 68, 1985.

[Sommerville et Sawyer 97] I. Sommerville, P. Sawyer,
« Requirements Engineering: A good Practice Guide », Chichester: John Wiley & Sons, 1997.

[Sommerville 04] I. Sommerville,
« Software Engineering », Seventh edition, Pearson - Addison-wesley, 2004.

[Standish 95] The Standish Group, *Chaos*.
"Standish Group Internal Report", http://www.standishgroup.com/chaos.html, 1995.

[Standish 94] The Standish Group International, Inc.
« The *CHAOS* report », 1994.

[Sun 01] Sun Microsystems,
« Enterprise Java Beans Specification »,Version 2.0, http://java.sun.com/products/ejb/docs.html, 2001.

[Tawbi et *al.* 00] M. Tawbi, F. Velez, C. Souveyet, C. Ben Achour,
« Evaluating the CREWS-L'Ecritoire requirements elicitation process », CREWS Internal Report, 2000.

[Tawbi et *al.* 98] M. Tawbi, C. Souveyet, C. Rolland,
« *L'ECRITOIRE a tool to support a goal-scenario based approach to requirements engineering* », Information and Software Technology journal, Editor : Martin Shepperd, Puplisshers : Elsevier Science B.V, 1998

[Tawbi et Souveyet 99] M. Tawbi, C. Souveyet,
« Guiding Requirement Engineering with a Process Map », Proceedings of MFPE'99 : 2nd International Workshop on The Many Facets of Process Engineering, Gammarth, Tunisia, 12-14 May, 1999.

[UML 03] OMG Group
« Revision Task Force », OMG UML Specification. http://www.uml.org.

[Vallée et Roque 02] F. Vallée, P. Roque,
« UML en Action », Eyrolles, 2002.

[VOB 94] ED. VOB ,
« Similarity Concepts and Retreival Methods ». FABEL project Technical Report number 13. Gesselschaft für Mathematik und Datenverarbeitung (GMB). Sankt Augustin, 1994.

[Wang et Schmidt 01] N. Wang, D. Schmidt,
« Overview of the CORBA component model », In component-Based Software Engineering, (W.T Councill, G.T. Heineman), Boston: Addison-Wesley, 557-572 (ch, 19), 2001.

[Weinreich et Sametinger 01] R.Weinreich, J. Sametinger,
« Component models and component services: concepts and principles. In component-Based Software Engineering », (W.T Councill, G.T. Heineman) , Boston: Addison-Wesley, pp. 33-48 (ch. 19), 2001.

[Wirth 83] N. Wirth
« Program development by stepwise Refinement », Communications of the ACM vol. 26 (1), 1983.

[Wirth 86] N. Wirth
« Algorithms and Data Structures », Englewood Cliffs, NJ: Prentice-Hall, 1986.

[Yourdon et Constantine 79] E. Yourdon, L. Constantine,
« Structured design », Englewood Cliffs, NJ : Prentice-Hall, 1979.

[Zave 95] P. Zave,
« Classification of Research Efforts in Requirements Engineering », In Proceedings of RE'95, 1995.