CREWS-L'Ecritoire: Une approche Guidant l'Ingénierie des Besoins

Mustapha Tawbi

E-mail : tawbi@univ-paris1.fr
Centre de Recherche en Informatique (CRI), Université de Paris1 Panthéon Sorbonne,
90 rue de Tolbiac 75013 Paris, France, tel + 33 1 44 07 86 34.

Résumé

L'Ingénierie des Besoins(IB) est une phase importante dans un projet de développement de système. Sous-estimer cette importance peut mener à l'inachèvement des projets, au dépassement de leurs budgets ou à la défaillance des systèmes développés. La plupart des approches d'IB se base sur deux concepts essentiels : Les buts et les scénarios. Si ces deux concepts sont bien définis en tant que produits, toutes les approches n'indiquent pas quoi faire, quand faire ou même comment faire pour les obtenir, les démarches proposées par les approches d'IB sont souvent informelles, imprécises, trop générales et mal adaptées aux problèmes particuliers rencontrés. Par conséquent, les outils associés à ces démarches souffrent du même problème ; ils aident à extraire, stocker, manipuler et documenter le produit mais contribuent peu au processus intellectuel d'obtention du produit.
Ce papier essaie de donner des solutions à ces problèmes en proposant une approche qui guide l'IB en se basant sur un modèle de processus : le MAP. Ce processus guidé est implémenté par un logiciel : 'L'Ecritoire'.

1. Introduction

La grande majorité des approches d'ingénierie des besoins se base sur un des deux concepts : *but* ou *scénario*. Les approches dirigées "par les buts" offrent beaucoup de diversité : identification des dépendances entre agents ou acteurs[Yu98], opérationalisation des buts par des formules de la logique temporelle [VanL98], modélisation des buts par décomposition sous forme d'arbres et/ou [BUBE94]. Les approches d'IB "par les scénarios" se sont avérées efficaces dans la découverte des besoins pour des situations prévues à l'avance [POTT97], d'élucider les cas exceptionnels [JACO99], de dériver les modèles conceptuels à partir des scénarios [DANO1997], et de raisonner à propos des choix de conception [CARO1995].

L'application d'une approche *"par les buts"* dans un cadre de développement industriel soulève de nombreuses difficultés : il n'existe pas de définition unifiée du concept de but [Yu98]. Les approches existantes n'indiquent pas comment un but doit être formulé [ANTO96], ce qu'il doit exprimer, ni si un but est adapté aux besoins [POTT97]. Les scénarios sont plus faciles à utiliser que les buts. Les buts peuvent être explicites seulement une fois qu'une profonde compréhension du système est atteinte. De plus, les scénarios décrivent des comportements concrets. Par conséquent, ils permettent de capturer des besoins concrets et réels. Cependant, étant donné qu'ils utilisent des exemples et des illustrations concrètes, les scénarios fournissent des descriptions des besoins qui sont restreintes et nécessitent d'être généralisées afin d'obtenir des besoins complets.

Pour trouver des solutions aux difficultés et aux limites des approches "*par les buts*" et à celles "par les scénarios", de nouvelles approches basées sur un couplage de ces deux concepts ont été proposées. Dans [DANO1997][JACO99][LEIT997], les buts sont considérés comme des propriétés contextuelles des cas d'utilisation. Alors que dans[COCK00], ils sont utilisés comme un moyen de structurer les cas d'utilisation. Dans [BENA99] les scénarios sont utilisés pour décrire différentes manières possibles d'atteindre le même but.

Cependant, les praticiens rapportent plusieurs problèmes que ces approches ne résolvent pas. Nous désignons trois problèmes liés aux approches d'extraction des besoins sous :

La relation unidirectionnelle (but, scénario) : les approches couplant les scénarios et les buts suggèrent une relation unidirectionnelle des buts vers les scénarios (les buts sont opérationnalisés par les scénarios, les scénarios sont associés aux buts). Ces approches traitent des buts étant déjà identifiés et ne proposent aucun mécanisme pour faciliter leur identification. Elles adressent le problème de réalisation des buts et ne disent pas comment procéder pour les identifier auparavant.

Le guidage méthodologique, la modélisation et la formalisation des démarches d'IB : si les scénarios et les buts sont bien définis en tant que produits obtenus à partir d'un processus d'IB, toutes les approches n'indiquent pas quoi faire, quand faire ou même comment faire pour les obtenir. Il est largement admis qu'un tel guidage devrait être fourni sous la forme d'un modèle de processus. Cependant, le plus souvent, le guidage est plutôt basé sur des heuristiques manuelles et informelles que sur des règles systématiques. Les démarches proposées sont souvent informelles, imprécises, trop générales et mal adaptées aux problèmes particuliers rencontrés.

a) *Le support outillé :* les outils d'ingénierie des besoins (SOMATIC, SAVRE, REQUISTPRO, DOORS.etc) offrent une multitude d'éditeurs graphiques, textuels, vérificateurs syntaxiques de spécifications mais ces outils sont essentiellement passifs. Ils aident à extraire, stocker, manipuler et documenter les besoins mais contribuent peu au processus intellectuel d'obtention du besoin. Le guidage offert par ces outils porte sur le produit et non pas sur le processus. Le guidage portant sur le processus lui-même est imprécis et vague et même parfois inexistant.

Dans ce papier, nous proposons :

a) de coupler les deux concepts de *but* et de *scénario* selon une relation *bidirectionnelle*. Le processus de construction associé à l'approche que nous avons nommé CREWS-L'Ecritoire[1] est un mouvement itératif, systématique et bidirectionnelle entre les deux concepts du couple. Lorsqu'il y a un but à atteindre, un scénario est écrit pour décrire une manière de le réaliser. Celui-ci, une fois écrit et couplé au but, est analysé pour découvrir de nouveaux buts. Ce mouvement bidirectionnel est répété jusqu'à l'extraction de tous les besoins du futur système.

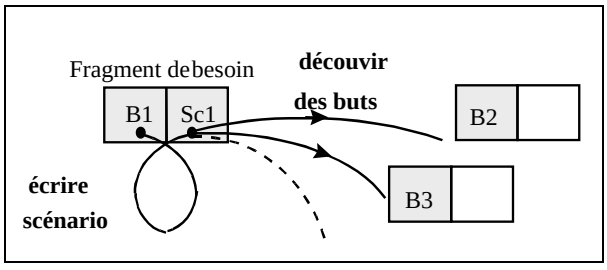

Figure 1: Le processus de l'approche CREWS-L'Ecritoire

[1] Cette approche a été proposé dans le cadre du projet européen CREWS (Cooperative Requirements Engineering With Scenarios) 21.903. , d'où le nom CREWS-L'Ecritoire.

b) La démarche de l'approche est modélisée par un 'MAP'. Le MAP est un modèle de processus vu comme un panel de processus à partir duquel, en se basant sur des directives de guidage et par la sélection dynamique, la prescription particulière qui est la mieux adaptée à la situation rencontrée est sélectionnée.

c) Le MAP de l'approche est automatisé par un logiciel *: 'L'Ecritoire'* dont l'architecture est adaptée aux notions du MAP et basée essentiellement sur le concept du *composant logiciel*. 'L'Ecritoire' permet de guider le processus de l'approche CREWS-L'Ecritoire.

Le papier est organisé de la façon suivante. En section 2 nous présentons le concept et la structure linguistique d'un but. La section 3 présente le concept, et la structure linguistique de scénario. La section 4 présente la notion et la hiérarchie des fragments des besoins. En section 5, nous présentons le modèle du processus 'MAP' ainsi que le processus de construction de l'approche. La section 8 détaille l'implémentation de l'approche en utilisant le concept du composant logiciel, la relation entre celui-ci et les concepts du MAP ainsi que l'architecture d'un logiciel implantant un processus construit par un MAP. La section 9 présente le logiciel L'Ecritoire qui implémente le processus CREWS-L'Ecritoire. Nous concluons en section 10.

2. Le concept du But

Nous définissons un but comme '*un objectif à réaliser en utilisant le futur système'* [PLIH98]. Un but est une proposition en langage naturel formalisée selon une structure inspirée de [PRAT99] (Figure 2). Selon cette structure, un but est composé d'un verbe suivi d'un ensemble des paramètres (une fonction sémantique est associée à chaque paramètre, la même fonction ne pouvant pas être associée à deux paramètres différents). Il existe les quatre types de paramètres suivants : *Cible, Direction, Voie et Bénéficiaire.*

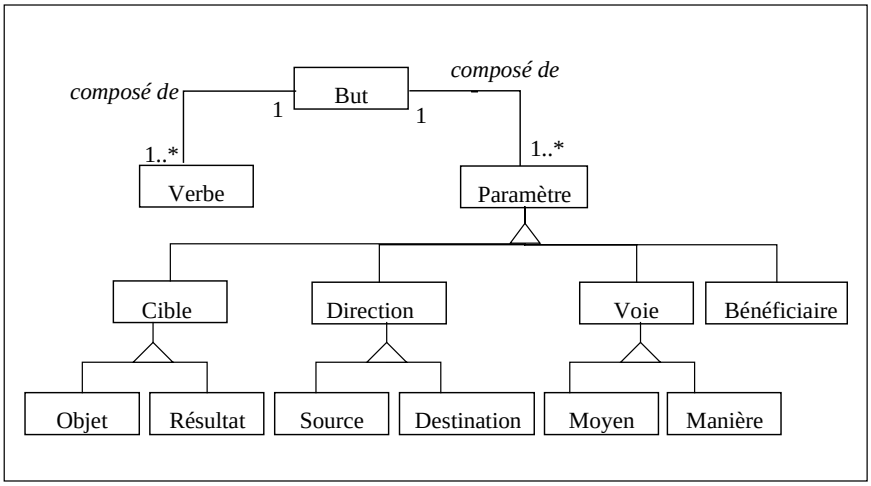

Figure 2: la structure du but selon une notation UML

Cible : La *cible* (Cib) concerne les entités affectées par le but. Il y a deux types de cibles : L'*objet* et le *résultat.* L'objet existe avant la réalisation du but et peut éventuellement être modifié ou supprimé par celui-ci, alors que le résultat est l'entité qui résulte de la réalisation du but désigné par le verbe.

Direction: Les deux types de direction sont appelés la *source* et la *destination* et identifient respectivement l'endroit initial et final de l'objet. La source est le point du départ du but (source d'information ou lieu physique), et la direction est son point d'arrivée.

Voie: Une voie est spécialisée par les deux paramètres *manière* et *moyen*. La manière spécifie la façon d'atteindre le but et le moyen est l'entité ou l'outil, par lequel le but est atteint.

Bénéficiaire: La personne ou le groupe en faveur de qui le but doit être atteint.

Les deux exemples suivants montrent deux buts formalisés selon cette structure. Le deuxième exemple montre que cette structure peut être appliquée d'une manière récursive sur le paramètre manière, ces deux exemples sont utilisés pour le processus d'extraction des besoins d'un système du guichet automatique bancaire GAB :

'$(Fournir)_{verbe}$ $(des\ services\ de\ retrait\ bancaires)_{Rés}$ $à$ $(nos\ clients)_{Bén}$ $au\ (moyen\ d'un\ GAB)_{\ Moy}$,

'$(Fidéliser)_{verbe}$ $(nos\ clients)_{Bén}$ en $((fournissant)_{verbe}$ $(des\ services\ de\ retrait\ d'argent)_{Rés}$ $au\ (moyen\ d'un\ GAB)_{Moy})_{Man}$''

3. Le concept du Scénario

Un scénario est '*un comportement possible limité à un ensemble d'interactions entre plusieurs agents'* [BENA99]. Le scénario est écrit en langage naturel comprenant une ou plusieurs actions. Une combinaison des actions dans un scénario décrit un chemin unique menant d'un état initial à un état final des agents du système. La Figure 3 présente la structure d'un scénario. Dans les paragraphes ci-dessous, nous détaillons les concepts présentés à la Figure 3.

Un scénario est caractérisé par deux états: *Initial* et *final*. Un état initial attaché à un scénario définit les pré-conditions nécessaires au déclenchement de celui-ci alors q'un état final définit les états atteints à la fin du scénario. Par exemple, le scénario *' retirer de l'argent à partir du GAB dans le cas normal'* ne peut se déclencher que si les deux états *'l'utilisateur a une carte'* et *'le GAB est prêt'* sont vrais. Le déclenchement de ce scénario se termine donc avec les trois états finaux *' l'utilisateur a une carte', 'l'utilisateur a de l'argent'* et *'le GAB est prêt '*.

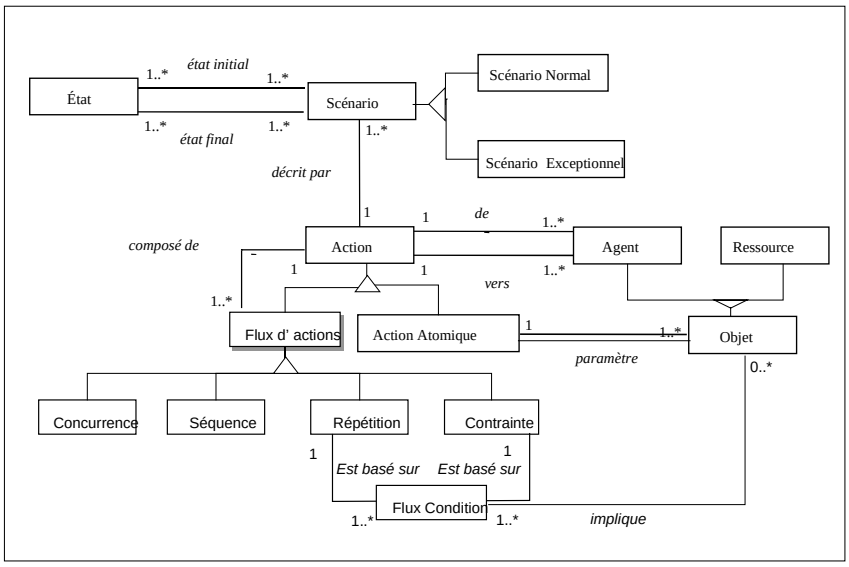

Figure 3: La structure d'un scénario

On distingue deux types des scénarios: Les scénarios *normaux* et les scénarios *exceptionnels*. Un scénario normal permet d'atteindre le but, ce qui n'est pas le cas avec un scénario exceptionnel. Par exemple, le scénario associé au but ‘*retirer de l'argent à partir du GAB en saisissant trois fois un code erroné '* est du type exceptionnel avec les états finaux *' l'utilisateur n'a pas une carte', 'l'utilisateur n'a pas de l'argent'* et *'le GAB est prêt.*

Les actions sont de deux types: *Atomique* et *flux*. Une action atomique est une interaction entre deux agents affectant un objet paramètre. Un agent et un objet peuvent figurer dans plusieurs interactions différentes. La clause '*l'utilisateur insère une carte dans le GAB'* est un exemple d'action atomique, le paramètre de cette action étant *'une carte*, et c'est une action de communication impliquant les deux agents *l'utilisateur* ' et *'le GAB'*.

Un flux d'actions est composé de plusieurs actions. La phrase *'si la carte est valide, le GAB affiche un message demandant le code de la carte est affiché à l'utilisateur*' en est un exemple. Le flux d'actions a l'une des sémantiques suivantes : *séquence, contrainte, répétition ou concurrence.*

Les deux types de flux (*contrainte et répétition)* sont associés à des *conditions du flux* caractérisant la progression des actions dans un scénario. Dans la proposition *' si le code est valide, un message demandant le montant est affiché par le GAB à l'utilisateur',* la condition *'si le code est valide'* identifie un cas unique d'utilisation du GAB décrit par un scénario.

Les patrons sémantiques

Les patrons sémantiques ont été proposés pour la première fois par [BENA99]. Ces patrons permettent de vérifier la correspondance de scénario à la structure. Les patrons sémantiques sont de deux types: L*es patrons sémantiques de clause* et *les patrons sémantiques de séquence*. Les premiers fournissent la sémantique des actions atomiques et les derniers celle des flux d'actions.

Un verbe est le concept principal d'un patron sémantique de clause. Un verbe est associé à un ensemble de paramètres. Les verbes sont classifiés en deux types: Verbes d'*action* et verbes de *communication*. Les verbes d'action sont ceux utilisés dans des actions simples telles que ‘*valider, vérifier, etc.’* Ils nécessitent un paramètre *objet*, qui est l'objet du verbe et un paramètre *agent*, qui est son sujet. Ainsi l'action atomique *‘le GAB vérifie la validité de la carte’* est représentée par le patron sémantique suivant :

Action(vérifier)[Agent : le GAB; Objet: 'la validité de la carte'].

Les verbes de communication sont ceux utilisés dans des actions concernant un échange d’objets entre plusieurs agents. Un objet échangé peut être de nature physique, *'une carte'* par exemple ou de nature informationnelle, *'message*' par exemple. Dans l'action *'l'utilisateur insère une carte dans le GAB'*, ‘insérer’ est un verbe de communication et ‘une carte' est un objet physique. Un verbe de communication nécessite aussi deux paramètres: Une *source ‘l'utilisateur'* et une *destination ‘le GAB'*. Ainsi l'action *précédente* est représentée par le patron sémantique suivant :

Communication (insérer) [Agent: 'l'utilisateur'; Objet : une carte ; Source : l'utilisateur; destination : 'le GAB'].

L'objectif de patron sémantique de séquence est de modéliser les flux d'actions. Nous distinguons ainsi quatre types de patrons de séquences: *Séquence, contraintes, itération,* et *concurrence. (*Pour avoir plus de détailles sur les patrons sémantiques nous vous

conseillons [BENA99])

4. La notion et la hiérarchie des Fragments de besoins

Pour documenter les besoins du système extraits à l'aide des buts et des scénarios, nous utilisons le concept du *fragment de besoin (FB)*. Un fragment de besoin est un couple <but-scénario>. Etant donné qu'un but est intentionnel et un scénario opérationnel, nous définissons un FB comme '*une manière de réaliser un but au moyen d'un scénario'*.

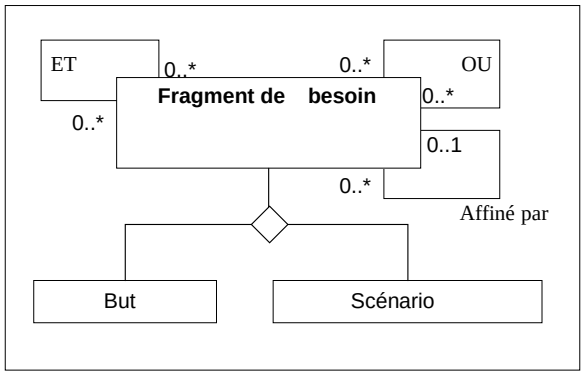

Figure 4: La structure d'un fragment de besoin

Les FBs sont assemblés selon trois types de relations nommés c*omposition, alternatifs* et *affinement*. Les deux premières relations sont représentées par une structure horizontale reliant les FBs par des liens appelés *'ET'* et *'OU'*. La troisième relation est représentée par un lien appelé 'A*ffiné par'* pour une structure verticale comprenant trois niveaux d'abstraction: *Comportemental, fonctionnel*, et *physique.*

L'objectif du niveau *comportemental* est d'identifier les services que le système doit fournir à l'entreprise. Un *FB* comportemental associe un *but de gestion* et un *scénario de service*. Le but de gestion correspond à un objectif de l'organisation à l'égard de ses clients. Le scénario de service décrit le flux de services entre les agents (l'un d'entre eux est le système lui-même), qui est nécessaire pour satisfaire le but de gestion. Une action atomique dans le scénario de service est un service.

Le niveau *fonctionnel* détaille chaque service du niveau *Comportemental* comme un ensemble d'interactions entre agents (le système lui-même et ses utilisateurs). Un FB fonctionnel définit le flux d'actions pour assurer un service du système. Il couple un *but de service* et un *scénario d'interaction*. Le but de service exprime une manière d'assurer le service. Le scénario d'interaction associé décrit un flux d'interactions entre le système et ses utilisateurs pour atteindre le but de service.

Le niveau *physique* détaille chaque interaction du niveau fonctionnel par un ensemble d'actions internes au système. Chaque action peut faire référence à des objets du système et à des objets externes comme d'autres systèmes. Un FB *physique* est un couple comprenant un *but de système* et un *scénario interne*. Un but de système exprime une manière possible pour exécuter une interaction identifiée dans le scénario d'interaction. Un scénario physique décrit le flux d'actions internes au système pour satisfaire le but.

La Figure 5 montre l'exemple d'une partie de la hiérarchie des fragments de besoins.

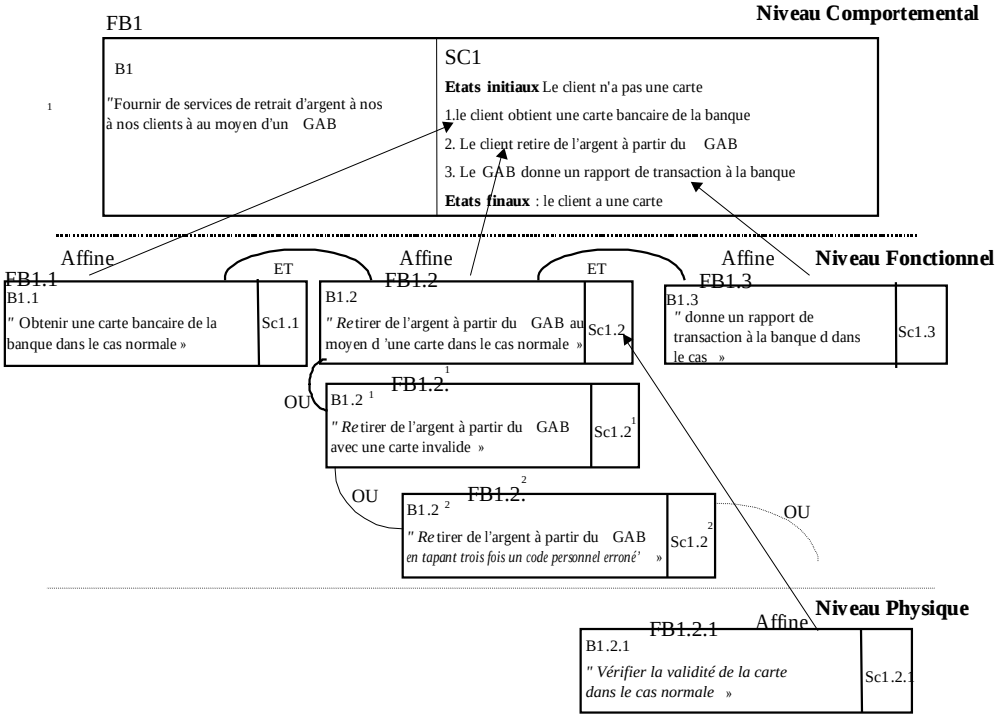

Figure 5: La hiérarchie des fragments de besoins

5. Le processus de l'approche CREWS-L'Ecritoire

CREWS-L'Ecritoire est une approche semi-automatique de découverte et de documentation des besoins. Pour ce faire, le processus de l'approche exploite la relation entre les deux concepts du but et de scénario selon un mouvement bidirectionnel consistant en deux activités principales: L'écriture des scénarios et la découverte des buts. Les deux sous-sections suivantes font le survol des ces deux tâches.

5.1 L'écriture des scénarios

Les scénarios sont écrits en langage naturel, l'écriture d'un scénario est une activité complexe où l'on peut distinguer trois sous-activités: Ecrire le scénario, vérifier et compléter les actions du scénario à l'aide de patrons sémantiques, et conceptualiser le scénario.

5.1.1 Ecrire le scénario

Cette activité commence par l'écriture d'un texte en langage naturel décrivant une manière possible d'atteindre un but. Pour réaliser cette activité l'utilisateur est guidé par deux types de directives: Les directives du *contenu* et les directives du *style*. Les directives du style guident la formulation du texte du scénario, tandis que les directives du contenu se focalisent sur le contenu du scénario. Par exemple, étant donné le but '*retirer de l'argent à partir du GAB dans le cas normal '*, l'écriture du scénario permet d'obtenir la description suivante:

États initiaux: *'l'utilisateur a une carte', 'le GAB est prêt '.*
L'utilisateur insère une carte. Le GAB vérifie la validité de la carte, si elle est valide, un message demandant le code est affiché. L'utilisateur tape son code sur le clavier du GAB. si celui-ci est valide alors, un message demandant le montant est affiché à l'utilisateur.
L'utilisateur introduit le montant dans le GAB. Si celui-ci est disponible, alors le GAB éjecte la carte à l'utilisateur. Si un reçu est demandé le GAB imprime un reçu et il délivre les billets à l'utilisateur.
États finaux: *'l'utilisateur a une carte', 'le GAB est prêt ', 'l'utilisateur a de l'argent'* .

5.1.2 vérifier et compléter les actions du scénario à l'aide des patrons sémantiques

Les patrons sémantiques permettent d'identifier et de compléter les paramètres manquants dans une action. Un paramètre manquant est exprimé par un point d'interrogation. Prenons l'exemple de l'interaction suivante:

'L'utilisateur insère une carte'.

Le verbe '*insére*r' est un verbe de communication et l'interaction est donc exprimée par le patron intancié suivant:

Communication ('insérer') [Agent : 'l'utilisateur'; Objet : 'une carte' ; Source: 'l'utilisateur'; Destination : '?'].

Le paramètre 'Destination' est inconnu et donc exprimé par un point d'interrogation '?'. Ceci permet de détecter l'absence du paramètre et de compléter l'interaction par l'ajout de celui-ci ('le GAB). L'interaction est complétée ainsi pour obtenir la description suivante:

*'L'utilisateur insère une carte **dans le GAB**'.*

Ce mécanisme de vérification est automatique dans l'approche. Par exemple la vérification des actions du scénario permet d'obtenir le texte suivant (où les modifications sont indiquées en gras):

États initiaux: *'l'utilisateur a une carte' , 'le GAB est prêt '.*
L'utilisateur insère une carte **dans le GAB**. Le GAB vérifie la validité de la carte, si **la carte** est valide, un message demandant le code est affiché **à l'utilisateur.** L'utilisateur tape son code sur le clavier du GAB. si **le code** est valide alors, un message demandant le montant est affiché **par le GAB** à l'utilisateur. L'utilisateur introduit le montant dans le GAB. Si **le montant** est disponible, alors le GAB éjecte la carte à l'utilisateur. Si un reçu est demandé par **l'utilisateur au GAB**. le GAB imprime un reçu **à l'utilisateur** et **le GAB** délivre les billets à l'utilisateur.
États finaux: *'l'utilisateur a une carte' , 'le GAB est prêt ', 'l'utilisateur a de l'argent'* .

5.1.3 Conceptualiser le scénario:

La conceptualisation d'un scénario permet de transformer un texte en langage naturel en un texte semi-structuré, ceci facilite la lisibilité des scénarios et rendre possible l'application de la deuxième activité du processus qui est la découverte des buts. Cette transformation est automatique dans l'approche et est appliquée aux scénarios vérifiés et complétés. L'exemple suivant montre le scénario précédent conceptualisé et couplé au but dans un fragment de besoin.

Retirer de l'argent du GAB au moyen d'une carte dans le cas normal	États initiaux: *'l'utilisateur a une carte' , 'le GAB est prêt '.* 1. l'utilisateur insère une carte dans le GAB 2. le GAB vérifie la validité de la carte 3. si la carte est valide, alors 4. un message demandant le code est affiché par le GAB à l'utilisateur 5. l'utilisateur tape son code sur le clavier du GAB 6. *si le code est valide*,alors 7. un message demandant le montant est affiché par le GAB à l'utilisateur 8. l'utilisateur introduit le montant dans le GAB 9. si le montant est disponible, alors 10. le GAB éjecte la carte à l'utilisateur 11. si un reçu est demandé par l'utilisateur au GAB, alors 12.le GAB imprime un reçu à l'utilisateur 13. le GAB délivre les billets à l'utilisateur États finaux: *'l'utilisateur a une carte' , 'le GAB est prêt ', 'l'utilisateur a de l'argent' .*

5.2 Découvrir des buts

Une fois le scénario est conceptualisé, l'approche propose d'exploiter les actions de celui-ci afin de découvrir de nouveaux buts. L'approche propose trois types de découverte des buts: *Alternatifs, complémentaires* ou *affinant* le fragment de besoin en cours. Chacun de ces types s'appuie sur un ensemble des règles. Une règle est composée d'un *but de règle* et d'un *corps de règle*. Le but décrit l'objectif de la règle, et le corps fournit la description et les directives permettant de guider l'ingénieur des besoins pour mener à bien l'application de la règle. L'exemple suivant montre l'application de la règle C1 étant une des cinq règles guidant la découverte des buts complémentaires.

Règle C1

But : Découvrir (à partir de FB <B, Sc>)$_{So}$ (les buts complémentaires du but B)$_{Res}$ (en analysant les interactions du scénario Sc)$_{Man}$

Corps :

Etape 1 : Dans le scénario Sc identifier les objets qui correspondent à des ressources physiques

Etape 2 : Pour chaque ressource construire les paires d'interactions (Consommation, Production)

Etape 3 : Pour chaque paire incomplète (soit avec l'interaction de consommation manquante ou avec l'interaction de production manquante) suggérer de nouveaux buts

Etape 4 : Sélectionner les buts et les nommer

Etape 5 : Le (s) but (s) sélectionné (s) est (sont) complémentaire (s) à FB.

Figure 6: La règle C1

Tout d'abord, les interactions contenant des objets qui correspondent à des ressources physiques du système doivent être identifiées (Etape 1). La règle applique ensuite le principe de production/consommation pour chaque ressource. C'est à dire qu'elle cherche les paires d'interactions où l'une consomme la ressource qui est produite par l'autre interaction de la paire (Etape 2). Chaque paire incomplète est à l'origine de nouveaux buts (Etape 3). L'ingénieur des besoins sélectionne les buts pertinents et les décrit selon la structure prédéfinie (Etape 4).

Les couples production/consommation extraits du scénario sont les suivants:

(*1. le GAB vérifie la validité de la carte, 10. le GAB éjecte la carte à l'utilisateur)*

*(12. le GAB imprime un reçu à l'utilisateur, **?**)*

*(13. le GAB délivre les billets à l'utilisateur,**?**)*

Les couples ‘b’et ‘c’ représentent des pairs incomplets où aux interactions de production ne correspondent pas des interactions de consommations équivalentes. Ainsi pour chaque pair incomplet, il faut identifier un (ou plusieurs) but(s) dont les FBs permettant d'établir l'équilibre entre la production et la consommation des ressources. Par conséquent, deux buts sont identifiés:

'Alimenter le GAB avec de l'argent' et,

'Alimenter le GAB avec de papier de reçu'

Ces deux buts sont liés par un lien 'ET' au fragment de besoin 'retirer de l'argent du GAB dans le cas normal’.

Le processus de l'approche CREWS-L'Ecritoire est représenté par un modèle de processus :'MAP'. La section suivante présente la notion du 'MAP' et met l'accent sur le MAP CREWS-L'Ecritoire.

6. Le MAP

Le MAP est un modèle fournissant un support dans la sélection d'alternatives en proposant des aides méthodologiques. En effet, le MAP est représenté comme un graphe étiqueté et dirigé qui utilise deux notions fondamentales *intention* et *stratégie* et comporte un ensemble des sections correspondant chacune à un triplet < *Intention source, Intention cible, Stratégie*>.

Une intention (représentée dans le MAP par un nœud (Figure 8)) capture la notion de tâche que l'utilisateur projette d'accomplir à un moment donné du processus. Une intention est une expression en langage naturelle qui commence par un verbe auquel on associe plusieurs paramètres. Le paramètre clé dans l'expression de l'intention c'est la cible. Par exemple, dans les expressions d'intention suivantes, 'un scénario' et 'un but' sont les cibles respectives à 'Conceptualiser' et ' découvrir ':

$(Conceptualiser)_{Ver}$ *(un scénario)*$_{Cible}$

$(Découvrir)_{Ver}$ *(un but)*$_{Cible}$

Une stratégie (représentée dans le MAP par une flèche (Figure 8)) est une manière par laquelle une intention est réalisée. Une stratégie dans un triplet < Intention source, Intention cible, Stratégie > caractérise le flux de l'intention source (déjà réalisée) à l'intention cible (à réaliser) et la façon d'accomplir l'intention cible.

Une *section* s'applique sur des instances de produit. A chaque instance de produit et à chaque intention on associe une *signatur*e. Une signature associée à une instance de produit indique l'état actuel de cette instance et par conséquent, en se référant au MAP, d'identifier les prochaines sections à appliquer sur cette instance. Une signature associée à une intention correspond à l'état de produit que la réalisation de cette intention permet d'atteindre. L'application d'une section sur une instance de produit permet de modifier cette instance et, par la suite, de changer sa signature. Par exemple, la figure suivante montre la section:

<Découvrir but, Ecrire Scénario, stratégie par prose libre>

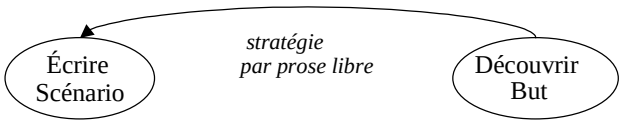

Figure 7: Exemple d'une section

Cette section s'applique sur les instances de produit ayant la signature *'But découvert'* et consiste à écrire un scénario que l'on associe au but en cours. Ainsi la signature de l'instance sur laquelle cette section a été appliquée passe de *'But découvert'* à *'Scénario écrit'*.

Le MAP est vu comme un panel de processus à partir duquel, par la sélection dynamique, la prescription particulière qui est la mieux adaptée à la situation du produit rencontrée est sélectionnée. Il est dynamique dans le sens où aucun enchaînement de sections inclus dans la MAP n'est recommandé " a priori " mais l'approche suggère une construction dynamique du chemin réel en naviguant dans le MAP. Il est évolutif dans le sens où chaque modification sur le processus global se traduit par un ajout, une suppression ou une mise à jour de la section correspondante du MAP sans affecter les autres sections non concernées par cette modification.

7. Utilisation du MAP

7.1 Règles d'utilisation

L'utilisation d'un MAP peut se résumer en trois étapes : le démarrage, la navigation, et l'arrêt de l'exécution.

7.2 Démarrage

L'utilisateur commence l'exécution de du MAP par une section dont l'intention source est l'intention '*Démarrer'* (voir Figure 8). Il choisit ensuite une intention cible parmi toutes les sections possibles (dont l'intention source est *Démarrer*). Selon la situation de la méthode d'origine, l'utilisateur choisit la stratégie la plus adaptée qui réalisera l'intention cible désirée.

7.3 Navigation dans le MAP

Chaque exécution d'une section permet à l'utilisateur de réaliser une intention générique du MAP sur une instance de produit. L'instance de produit prend ainsi la signature associée à l'intention cible. Cette intention devient donc une intention source pour cette instance de produit à partir duquel on doit naviguer pour atteindre une autre intention (cette fois-ci une intention cible). De la même manière, l'utilisateur doit tout d'abord choisir une intention cible parmi celles qui lui sont proposées par le MAP. Ensuite, il choisit une stratégie adaptée à la situation. Une fois cette intention atteinte, la navigation reprend sur le même principe.

7.4 Arrêt

Une fois l'utilisateur à atteint son objectif initial et qu'il souhaite arrêter l'exécution du MAP, il choisit l'intention cible *Arrêter*

La Figure 8 montre le MAP associé au processus de l'approche CREWS-L'Ecritoire, ce MAP est composé de 5 intentions, 12 stratégies et de 12 sections suivantes

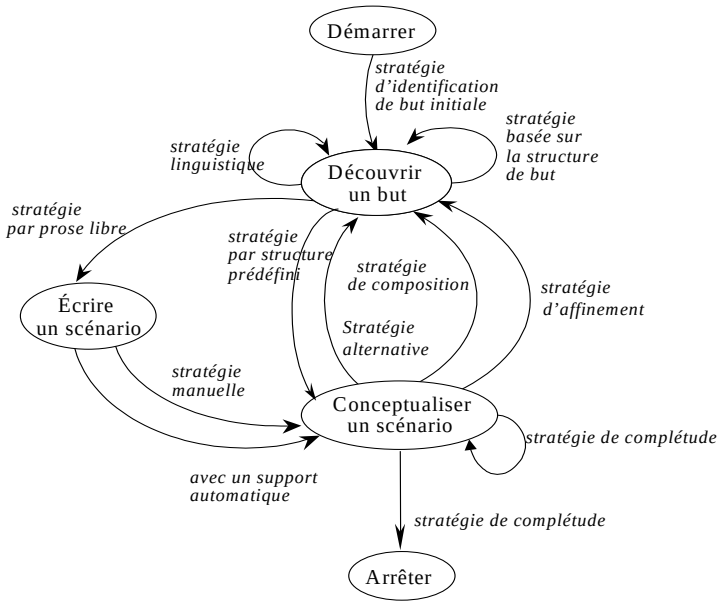

Figure 8 : Le Map CREWS-L'Ecritoite

1. *< Démarrer, Découvrir un but, stratégie d'identification du but initial>*
2. *< Découvrir un but, Découvrir un but, stratégie linguistique>*
3. *< Découvrir un but, Découvrir un but, stratégie basée sur la structure de but>*
4. *< Découvrir un but, Ecrire un scénario, stratégie par prose libre>*
5. *< Ecrire un scénario, Conceptualiser un scénario, avec un support automatique >*
6. *< Ecrire un scénario, Conceptualiser un scénario, stratégie manuelle >*
7. *< Découvrir un but, Conceptualiser un scénario, stratégie basée sur la structure prédéfinie >*
8. *< Conceptualiser un scénario, Conceptualiser un scénario, stratégie de complétude >*
9. *< Conceptualiser un scénario, Découvrir un but, stratégie d'affinement >*
10. *< Conceptualiser un scénario, Découvrir un but, stratégie de composition>*
11. *< Conceptualiser un scénario, Découvrir un but, stratégie d'alternatives >*
12. *< Conceptualiser un scénario, Arrêter, stratégie de complétude >*

Le MAP capture toutes les activités associées à la démarche de l'approche CREWS-L'Ecritoire telles que la formalisation des buts, l'écriture, la conceptualisation et la complétude des scénarios, la découverte des buts et la documentation des fragments des besoins. Ainsi par exemple, la section 10 comporte cinq règles de découverte des buts complémentaires. La section 11 comporte trois règles de découverte des buts alternatifs, etc [TAWB99b].

8. Implémentation de l'approche

Aucun MAP n'a été auparavant implémenté par un logiciel. Un tel logiciel se doit d'hériter de toutes les caractéristiques du MAP tel que le dynamisme, la modularité et l'évolutivité. Pour ce faire, nous proposons une architecture d'implémentation des MAPs basée sur la

notion du *composant logiciel*. Un composant logiciel est une classe indépendante et réutilisable. Les composants logiciels sont stockés dans une bibliothèque. Une section dans un MAP est implémentée par un composant logiciel et un MAP de processus est ainsi implémentée à l'aide d'un ensemble de composants logiciels. La section suivante illustre la notion du composant logiciel.

8.1 La notion du composant logiciel.

La Figure 9 illustre la structure d'un composant logiciel. Un composant logiciel est composé de deux parties: *Interface* et *Corps*;

L'interface contient les directives d'utilisation du composant et elle est composée de deux sous-interfaces, *l'interface utilisateur* qui fournit à l'utilisateur les directives permettant d'exécuter le composant, et *l'interface ingénieur* contenant les directives permettant de guider la réutilisation de celui-ci.

La partie corps concerne la partie exécutable d'un composant. Un composant est exécuté sur un produit en deux modes: Le mode *'Do'* qui permet de changer l'état courant d'un produit (exprimé sous forme d'une signature) à l'état suivant de son cycle d'évolution et le mode *'Undo*' permettant de changer le produit de son état courant à l'état précédent du cycle d'évolution.

Chaque composant logiciel agit sur un ensemble d'instances des produits stockées dans des tables de données ou dans des fichiers textuels.

La partie 'Input' indique les données à fournir au composant afin de l'exécuter et la partie 'Output' indique le produit sortant à partir de l'exécution du composant. 'Sérialise Input' et 'Sérialise Output' sont les deux parties à implémenter pour permettre la réutilisation d'un composant dans un environnement différent de son environnement d'origine. Il s'agit des adaptateurs permettant de convertir les données de l'environnement où on désire réutiliser le composant en données conformes à la nature de celui-ci (Sérialise Input), et de la même façon de récupérer les données sortant en les convertissant en données conformes à la nature de cet environnement (Sérialise Ouptut).

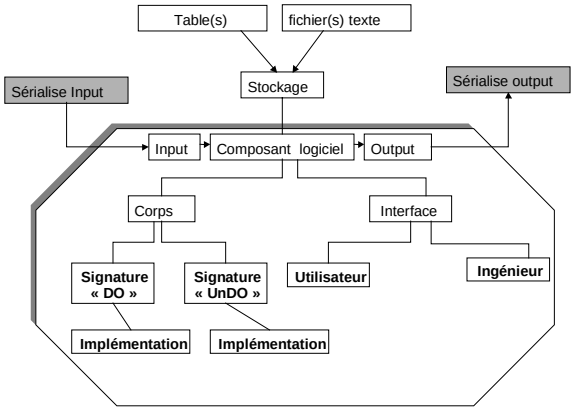

Figure 9: La structure d'un composant logiciel

La définition de la notion du composant logiciel nous permet de présenter L'architecture d'un outil implémentant un MAP.

8.2 L'architecture d'un outil implémentant un MAP

Un MAP est lui-même implémenté comme un composant logiciel. La Figure 10 présente les liens entre les concepts du MAP (niveau conceptuel) et celui du composant logiciel (niveau logique) selon une notation UML. Un MAP est composé d'un ensemble des sections auxquelles nous associons des directives d'application, celles-ci sont spécialisées en deux types: Directive d'*action* et directive de *MAP*.

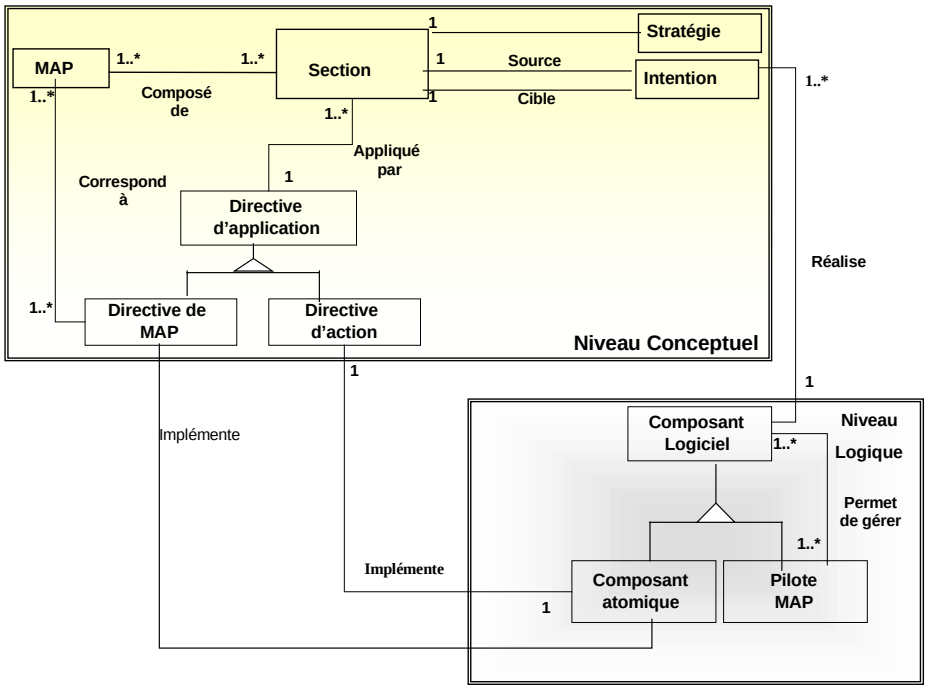

Figure 10: Les liens entre les concepts du MAP et leur implémentation sous forme des composants logiciels

Une directive d'action permet de guider l'exécution d'une section, ainsi par exemple, la directive d'action associée à la section < *Découvrir un but*, *Ecrire un scénario*, *stratégie par prose libre*>, va guider l'utilisateur à écrire un scénario correspondant au but découvert en respectant le modèle linguistique du scénario.

Une directive de MAP guide la navigation entre les différentes sections du MAP. une directive de MAP serait par exemple:

Après avoir conceptualiser un scénario et l'attacher à un but dans un FB , vous pouvez appliquer dessus le 5 sections suivante:

1. *<Conceptualiser un scénario, Conceptualiser un scénario, stratégie de complétude >*
2. *< Conceptualiser un scénario, Découvrir un but, stratégie d'affinement >*
3. *< Conceptualiser un scénario, Découvrir un but, stratégie de composition>*
4. *< Conceptualiser un scénario, Découvrir un but, stratégie d'alternatives >*
5. *< Conceptualiser un scénario, Arrêter, stratégie de complétude >*

Si vous voulez détailler les action de votre FB en des buts au niveau d'abstraction strictement inférieur, vous devriez appliquer section 2.

Si vous voulez chercher des FBs complémentaires à votre FB, vous devriez appliquer la section 3.etc,

Au niveau logique, un composant logiciel permet de réaliser une intention du MAP et il est spécialisé en deux types: Composant *atomique* et composant *pilote MAP*. Le premier implémente une directive d'action et le deuxième implémente un ensemble de directives du MAP associes à un MAP spécifique.

Finalement, un composant pilote MAP permet de gérer la navigation entre les différents composants logiciels selon les chemins dynamiques définis par le MAP.

L'architecture d'un outil implémentant un MAP est présentée à la Figure 11. Cette architecture est constituée de deux couches : la couche *Pilotage* et la couche *Exécution*.

Le composant 'Pilote MAP' se situe dans la couche Pilotage et il guide l'exécution des différents composants logiciels. Le couche pilotage gère deux bases de données, la base de '*signatures des produits'* et la base des '*traces des produits'* et le fichier '*traces de processus'*. La première base stocke les signatures courantes de chaque instance manipulée du produit et permet par conséquent au MAP d'indiquer les prochaines sections à exécuter sur celles-ci. La deuxième base stocke les anciennes signatures de chaque instance manipulée du produit. Elle permet par conséquent au MAP d'indiquer les sections qui ont été exécutées sur une instance du produit pour permettre un éventuel retour en arrière. Le fichier 'traces de processus' permet de tracer le processus global en indiquant à quel moment et comment une section a été exécutée ainsi que deux versions d'instance modifiée (une version d'avant et une autre d'après l'exécution de la section).

La base contenant les instances de produits se trouve dans la couche d'exécution. Par conséquent, l'environnement du MAP est indépendant de l'environnement des composants logiciels, ce qui permet la réutilisation des composants logiciels indépendamment d'un MAP spécifique. Pour exécuter un MAP, l'utilisateur interagit avec le pilote qui lui propose les sections à exécuter et les instances des produits dont les signatures sont attachées à ces sections, ensuite le pilote se charge d'appeler le composant logiciel correspondant au choix d'utilisateur. Une fois le composant exécuté, un message est envoyé au pilote qui se charge de mettre à jour les bases '*signatures et traces des produits'* et le fichier *'traces de processus'*.

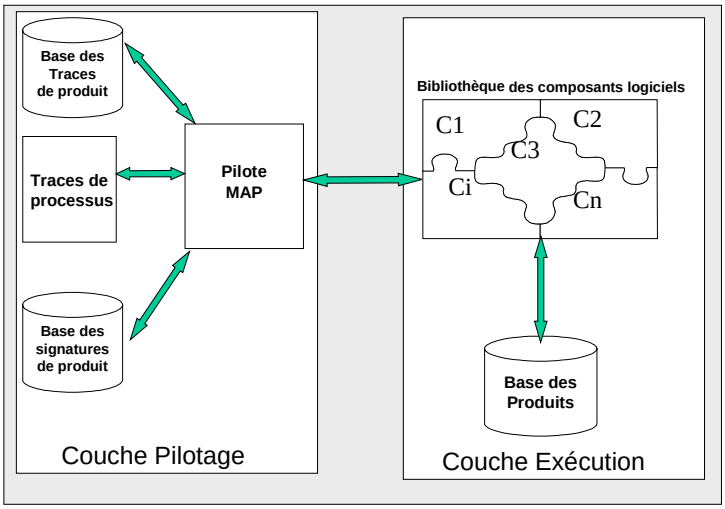

Figure 11: l'architecture d'un outil à base de MAP

Un logiciel ayant cette architecture comme base possède les caractéristiques suivante:

- *Flexible et Guide le processus :* à chaque étape de processus, des directives expliquant comment exécuter cette étape, quelles sont les étapes suivantes et quelles étaient les étapes précédentes pour supporter un éventuel retour en arrière, sont fournies. De plus, chaque composant est fourni avec une interface appelée (interface utilisateur) qui comporte une animation (crée par le logiciel DemoShiled) de l'utilisation de ce composant durant le processus.
- *Gère la traçabilité* : le fichier *'traces de processus'* permet de garder la trace sur le processus global, ainsi que les traces sur l'évolution des instances des produits.
- *Facile à faire évoluer* : chaque composant implémente une section. Ainsi, chaque modification (ajout, suppression ou modification) d'une section du MAP est traduite par une modification du composant correspondant. Cette modification n'affecte pas les autres composants de la bibliothèque.

9. L'Ecritoire

L'Ecritoire est le logiciel qui automatise le processus de l'approche CREWS-L'Ecritoire. Il implémente le MAP de l'approche suivant une architecture à base des composants. La bibliothèque de composants de L'Ecritoire est constituée de 13 composants logiciels. Ce logiciel est implémenté dans un environnement Visual BASIC comprenant 10000 lignes de code environ. Les différentes bases des données sont créées dans environnement Microsoft ACCES. Le MAP CREWS-L'Ecritoire est implémenté lui aussi comme un composant logiciel. Le moteur gérant la structure linguistique et les patrons sémantiques des scénarios est implémenté en Prolog. Chaque composant logiciel est un ActiveX et se trouve dans la bibliothèque sous forme d'un fichier DLL.

La Figure 12 suivante montre l'interface graphique du Pilote du MAP. L'utilisateur choisit une section et *appuie* sur le bouton ‘Do’, le moteur lui indique les instances de produit sur lesquelles on peut appliquer cette section. Dans L'exemple présenté à la Figure 12, la

section '<*Discover Goal, Full Prose Strategy, Write Scenario*>[2]' est sectionnée avec l'option 'Do' et le moteur indique à l'utilisateur que ceci est possible sur une seule instance du produit: Le but *'Withdraw cash from the ATM in a normal way'*.

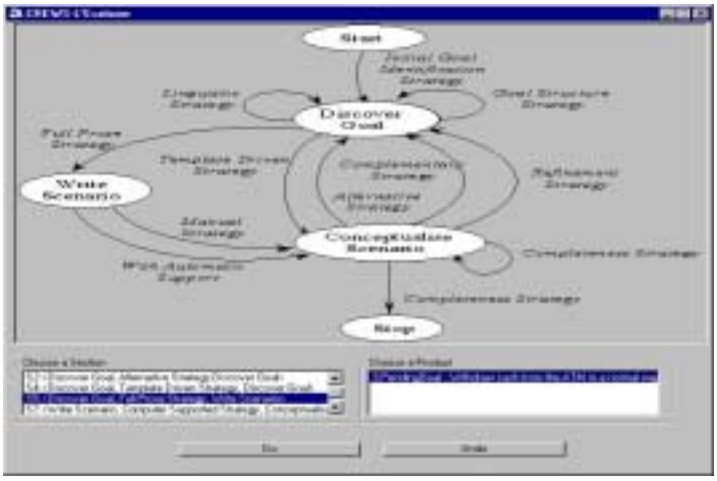

Figure 12: L'interface graphique de Pilote du MAP

Double-cliquer sur le but indiqué dans la zone de liste '*Choose a product'* permet d'appeler le composant logiciel *"Write a scenario in Full Prose*".

La Figure 13 montre l'exécution de ce composant logiciel sur le but *'Withdraw cash from the ATM in a normal way'*. La figure montre aussi les deux fenêtres permettant de guider l'écriture du scénario en fournissant à l'utilisateur deux types des directives, les directives du style et les directives du contenu.

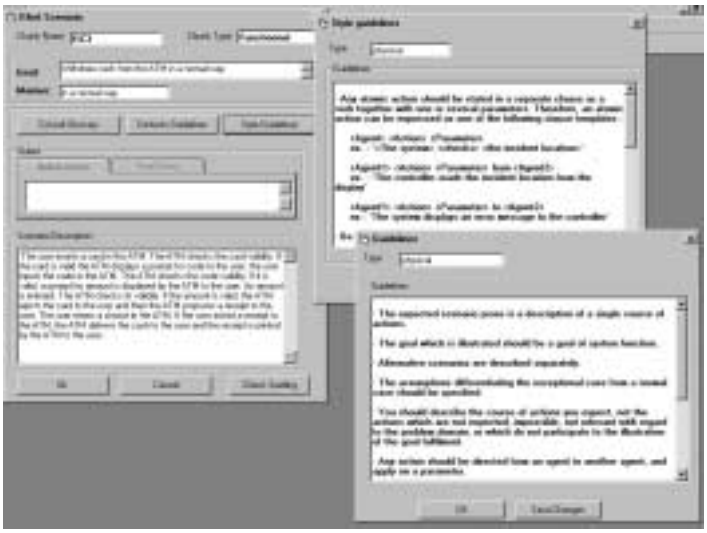

Figure 13: L'interface du composant 'Write scenario'

10. Conclusion

Nous avons proposé dans ce papier une nouvelle approche d'ingénierie des besoins basée

[2] Les interfaces de L'Ecritoire sont écrites en anglais étant donné que ce logiciel a été développé dans le cadre d'un projet comprenant des partenaires européens.

sur le concept du fragment du besoin. La démarche de l'approche a été modélisée par un modèle permettant de guider le processus d'ingénierie des besoins : le 'MAP'. Et a été automatisée par un logiciel construit selon une architecture à base de MAP et basé sur le concept modulaire de composant logiciel.

L'approche a été évaluée à travers 4 expériences de nature différente : Ateliers de travail [ROLL98], Études empiriques [TAWB99a, TAWB00], Étude de cas [ROLL99] et Tutoriaux [CREWSR]. Les résultats obtenus par ces expériences ont été très encourageants et nous ont permis d'un côté de valider l'applicabilité et l'efficacité de notre approche et de l'autre côté d'ajouter des nouvelles stratégies sur le MAP CREWS-L'Ecritoire, ainsi que d'améliorer certains des stratégies existantes.

Ce travail est détaillé plus en profondeur dans une thèse en cours de rédaction.

11. Références

[ANTO96] : A.I. Anton, Jhon dempster, Devon F.siege "*Goal based requirements analysis"*. Proceedings of the 2nd International Conference on Requirements Engineering ICRE'96, pp. 136-144, 1996.

[BENA99]: Camille Ben Achour, 'Extraction des Besoins par Analyse des Scénarios Textuels', Thèse du doctorat à l'Université de Paris6. Janvier1999.

[BUBE94]: J. Bubenko Jr., Marite Kirikova. 'Worlds' in Requirements Acquisition Modelling ", 4th European - Japanese Seminar on Information Modelling and Knowledge Bases, Kista, Sweden, Kangassalo and Wangler (Eds.), IOSpub),1994.

[CARO95] : J. M. Caroll, "*The Scenario Perspective on System Development*", in Scenario-Based Design: Envisioning Work and Technology in System Development, Ed J.M. Carroll, 1995.

[COCK00]: A. Cockburn, Writing Effective Use Cases (The Crystal Collection for Software Professionals), Addison-Wesley Longman, Incorporated(2000).

[CREWSR]: http://www.univ-paris1.fr/CRINFO/users/benachour/ESEM/index.html.

[DANO97] B. Dano, H. Briand, F. Barbier, *A use case driven requirements engineering process*. Third IEEE International Symposium On Requirements Engineering RE'97, Antapolis, Maryland, IEEE Computer Society Press, 1997.

[JACO99] : I. Jacobson, G. Booch, J. rumbaugh**,** **'**The Unified Process A Software Engineering Process Using the Unified Modelling Language', Addison-Wesley Longman, Incorporated(1999)

[LEIT97] J.C.S. do Prado Leite, G. Rossi, F. Balaguer, A. Maiorana, G. Kaplan, G. Hadad and A. Oliveros, *Enhancing a requirements baseline with scenarios*. In Third IEEE International Symposium On Requirements Engineering RE'97, Antapolis,

Maryland, IEEE Computer Society Press, pp. 44-53, 1997.

[PRAT97]: N. Prat, *Goal formalisation and classification for requirements engineering.* Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, pp. 145-156, June 1997.

[PRAT99]: N. Prat, *"Réutilisation de la trace par apprentissage dans un environnement pour l'ingénierie des processus"*, thèse présenté à l'université de paris1, février

1999.

[PLIH98]: V. Plihon, J. Ralyté, A. Benjamen, N.A.M. Maiden, A. Sutcliffe, E. Dubois, P. Heymans, *A reuse-oriented approach for the construction of scenario based methods*. Proceedings of the International Software Process Association's 5th International Conference on Software Process (ICSP'98), Chicago, Illinois, USA, 14-17 June 1998.

[POTT97] : C. Potts, *Fitness for use : the system quality that matters most.* Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, pp. 15-28, June 1997.

[ROLL98]: 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, 1055- 1071, Decembre 1998.

[ROLL99]: Colette Rolland, Georges Grosz, Régis Kla, " Experience With Goal-Scenario Coupling In Requirements Engineering, Fourth IEEE International Symposiumon Requirements Engineering (RE'99), University of Limerick, Ireland 7-11, June 1999.

[TAWB99a] : Mustapha Tawbi, Camille Ben Achour, Fernando Velez, " Guiding the

Process of Requirement Elicitation through Scenario Analysis : Results of an Empirical Study", REP'99 (The first international Workshop on the Requirements Engineering Process"Florence, Italy, September 99.

[TAWB99 b]: 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.

[TAWB00]: Mustapha Tawbi, Fernando Velez, Camille Ben Achour, Carine. Souveyet ' *Scenario Based RE with CREWS-L'Ecritoire* :*Experimenting the approach "*, RESQ'2000, 'Sixth International Workshop on Requirements Engineering: Foundation for Software Quality , June 5-6 2000, Stockholm, Sweden

[VanL98] : Van Lamweerde A., Willemet L., 'Inferring Declarative Requirements Specification from Operational Scenarios', IEEE Transactions on Software Engineering, Special Issue on Scenario Management, Vol. 24, No. 12, 1089-

1114, Dec. 1998

[WEID98] : K. Weidenhaupt, K. Pohl, M. Jarke, P. Haumer, *Scenario usage in system development : a report on current practice*. IEEE Software, March 1998.

[Yu98] : E.Yu, J. Mylopoulos , " Why goal oriented Requirement Engineering ", proceedings of the 4th International Workshop on Requirements Engineering Foundation for Software Quality. E. Dubois, A. Opdahl, K. Pohl (Eds). June, Pisa, Italy 1998.