CHAPITRE 4

ONTO-RUP : UNE DÉMARCHE DE DÉVELOPPEMENT DES SYSTÈMES ONTOGÉNÉTIQUES

Bien que la méthodologie RUP (Rational Unified Process) ne soit pas adaptée à la production des systèmes ontogénétiques, elle présente de nombreux atouts pratiques et jouit d'une notoriété mondiale. De ce fait, son adaptation pour les systèmes ontogénétiques est une voie prometteuse. Nous proposons ONTO-RUP, une démarche pour le développement des systèmes ontogénétiques, basée sur une extension du RUP. Pour cela, l'approche proposée prévoit un nouveau cycle de vie pour le RUP, en proposant l'adjonction d'autres disciplines, des éléments de modélisation et des directives pour mener la modélisation de l'ontogenèse. Pour cela, l'approche préserve tous les outils et atouts du RUP.

4.1 Approches Existantes et Ontogenèse

Plusieurs travaux de recherche ont montré que plus de 80% des coûts du cycle de vie du logiciel sont engagés après la livraison [Pim 11], [Sal 04], [Aur 05]. Par conséquent, il existe des raisons économiques importantes pour obtenir une meilleure compréhension du **pourquoi** et du **comment** les systèmes évoluent. La vision **comment** porte sur l'évolution du système logiciel comme une discipline d'ingénierie. Elle étudie les aspects plus pragmatiques qui aident le développeur du logiciel ou le chef de projet dans leurs tâches au jour le jour. Cependant, l'évolution des logiciels étudie également le processus de changement lui-même, l'analyse des débris du logiciel pour extraire les tendances, faire des

prédictions ou comprendre la nature même du phénomène de l'évolution du logiciel même (c'est à dire, il explore le **quoi** et **pourquoi**) [Mens 08].

4.1.1 Travaux Existants et Modélisation de l'Evolution

Les méthodes et les mécanismes standards actuels couvrent seulement une partie du processus de logiciel, et souvent ne supportent pas la phase de maintenance et d'évolution. La standardisation est rarement appliquée et les changements sont fréquemment implémentés manuellement. Ils n'existent pas actuellement des démarches de développement appropriées qui redonnent à l'évolution le statut de concept central.

La communauté du génie logiciel admet que les enjeux des nouvelles technologies se situent plutôt dans le contexte des méthodes de développement [Ben 12, Joh 05, Lim 11, Cha 01, Mes 08]. Ces dernières doivent faire face aux problèmes de l'évolution des systèmes logiciels mais aussi à leur complexité qui s'accroît à un rythme sans rapport avec nos capacités actuelles à les appréhender.

Sur le plan des démarches, il n'en existe pas encore, à notre connaissance, qui soient dédiées à l'approche ontogénétique qui mettent l'accent sur l'évolution et font que l'évolution devienne la préoccupation primaire de l'implémentation d'un système logiciel. La vision ontogénétique est radicale et sépare les changements dans deux catégories: Les changements anticipés et changements non anticipés.

Nous citons en premier les travaux de Pimentel and al. [Pim 11] qui représente notre source d'inspiration pour l'élicitation des évolutions anticipées des besoins. Les auteurs proposent une approche pour effectuer des changements sur les exigences exprimées par les modèles de but en se basant sur une représentation du futur fournie par la méthode « futures wheel ». Cependant, le travail se concentre principalement sur la façon dont les méthodes prospectives peuvent être utilisées pour l'élicitation des besoins futurs, et discute les impacts d'étudier le futur sur l'activité d'ingénierie des besoins. Les auteurs résument 17 méthodes prospectives où certaines de ces méthodes sont encore utilisées pour l'élicitation des besoins, mais pas pour étudier le futur; comme, par exemple, le cas des méthodes des scénarios et les méthodes participatives.

Une approche qui s'intéresse à la problématique de l'évolution sous l`angle des besoins est proposée dans [Eti 04] et [Sal 04] pour l'adaptation des systèmes d'information. Dans cette approche le processus d'évolution des besoins est modélisé en utilisant un méta-modèle et une typologie générique d'opérateurs exprimant les différents types des évolutions et ceci en utilisant le concept de l'écart (Gap en anglais).

D’autre travaux ont été proposés notamment, pour les systèmes d’information et les systèmes logiciels aux données-intensives où il est essential d’avoir une description claire et précise de schémas de bases de données. Hainaut and al. explorent en détail comment faire évoluer et migrer un schéma de base de données [Hai 08].

Nous pouvons citer pour l’évolution des systèmes d’information le travail basé sur de Hainaut et al. [Est 12]. Les auteurs proposent un Framework qui aide à identifier et contrôler les effets provoqués par l’évolution d’un système d’information et leurs conséquences.

Finalement, l’ingénierie de ligne de produits est une approche largement utilisée pour le développement efficace des systèmes [Bac 05]. Pour tenir compte des différences entre les produits logiciels, certaines adaptations des actifs de base sont habituellement exigées. Avec la sélection et la modification, un atout essentiel est choisi dans la base d'actifs et modifié ou configuré d'une façon planifiée. L'actif de produit résultant est encore reconnaissable comme une variante de l'actif de base d'origine.

Les changements non anticipés sont des évolutions qui ne se produisent qu’après la livraison du produit et qui ne sont pas prévues durant le développement du système. Bien qu’il soit possible de raisonner sur le futur [Gle 72, Pim 11, Lim 11, Kav 06, Gle 09, Eck 96, Zow 96, Fis 98], il n’est pas possible de prévoir toutes les possibilités de changement. Il est bien clair qu’il n’existe aucun modèle qui puisse éliciter quelque chose qui est totalement inattendu ou inconnu. Seulement, nous pouvons parler dans ce contexte des paradigmes comme le développement continu, amélioration continu de systèmes logiciels, concevoir des produits à faire évoluer dans le futur, etc. [Cis 14, Joh 05, Kru 01].

Une autre facette très importante et particulière de l’évolution est la mise à jour dynamique [Dow 01, Hic 01]. Elle permet à plusieurs applications importantes de s’exécuter continuellement et sans interruption. C’est le cas des systèmes critiques, tels que la commutation téléphonique, les transactions financières, la réservation des places d’avion et le contrôle du trafic aérien ou ferroviaire.

La littérature sur la maintenance du logiciel est beaucoup moins abondante que celle sur le développement du logiciel. Jusqu’aux années 1990, le thème de la maintenance était peu abordé dans les cursus universitaires d’informatique et de génie logiciel, et c’était en travaillant dans les organisations elles-mêmes que les ingénieurs logiciels s’initiaient aux spécificités de la maintenance et y développaient une expertise spécifique. La prise de conscience de l'importance de la maintenance durant le développement de systèmes logiciels a augmenté au cours des dernières décennies. De plus en plus, le logiciel n'est plus développé à partir de rien, mais constitue une poursuite du développement de logiciels existants [Phi 00].

Actuellement, beaucoup des travaux ont été proposés pour améliorer le processus d’évolution et de maintenance des systèmes logiciels, tel que la retro-ingénierie [Chi 90], la

compréhension de programmes [O'B 03], [Ber 07] et la réingénierie [Chi 90, Arn 93, Som 01].

Malgré l'existence de plusieurs méthodes de développements de logiciels, Ils en n'existent pas actuellement qui peuvent produire des systèmes ontogénétiques, c'est-à-dire celles qui prennent en charge les changements conformément à l'approche ontogénétique.

On peut distinguer deux catégories de méthodes de développement qui existent actuellement. Les méthodes traditionnelles axées-plan qui se basent sur un ensemble d'activités séquentielles. La deuxième catégorie est celle dite agile [Mes 08].

Ni l'approche traditionnelle ni l'approche agile ne peuvent produire des systèmes ontogénétiques. Nous proposons dans les sections suivantes notre approche Onto-RUP, une démarche pour le développement des systèmes ontogénétique basée sur une extension du Rational Unified Process.

4.1.2 Approches de Développement Existantes et Ontogenèse

Les systèmes ontogénétiques évoluent tout en restant opérationnels et en réponse à des stimuli internes. Les changements/évolutions anticipés sont analysés et codifiés durant les phases de développement du système, sous forme de patchs, qui s'exécutent si certaines conditions sont vérifiées. Comme signalé précédemment, ils n'existent pas dans la littérature des méthodes de développement qui prennent en charge la conception et l'implémentation des évolutions anticipées qui se déclenche automatiquement pendant le fonctionnement du système.

Pour faire évoluer les systèmes ontogénétiques pendant leur exécution tout en préservant leur disponibilité, c'est-à-dire leur continuité de service, les évolutions/changements non anticipés sont pris en charge dés leur arrivée et le système ontogénétique reste opérationnel et mis à jour dynamiquement, cette caractéristique existe dans certains systèmes logiciels mais elle est liée beaucoup plus à la plateforme du codage du logiciel qu'à la méthode de son développement et production.

Toutes les formes d'évolution sont traitées de la même façon. Ce point est important car les systèmes ontogénétiques présentent l'avantage de considérer les deux formes de changements (anticipés ou non) de manière identique. Cette vision est absente dans toutes les méthodes de développement existantes qu'elles soient traditionnelles ou agiles. Le cycle de maintenance du système logiciel est totalement indépendant du cycle de son développement (Figure 4.1).

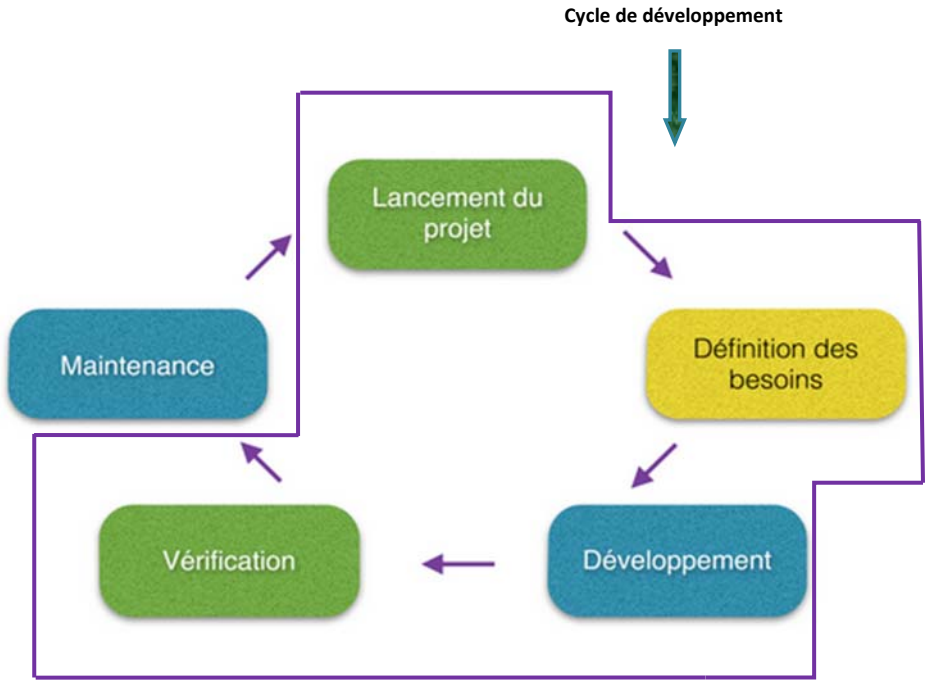

Figure 4.1 Cycle de développement et de maintenance de logiciel

Dans le développement des systèmes ontogénétique, nous proposons le paradigme de ***Développement Continu du Système Ontogénétique***.

4.2 Le Développement des Systèmes Ontogénétique : Aperçu

L'approche ontogénétique reporte la prise en compte des changements anticipés jusqu'à ce qu'ils deviennent nécessaires (i.e. certaines conditions se réalisent), suite à cela le code est altéré dynamiquement pour incorporer ces changements. Cette approche a l'avantage de maintenir le code performant, simple et reflétant les besoins réels actuels. Concrètement le code est débarrassé des structures alternatives au profit d'une évolution dynamique des comportements des objets en fonction des besoins réels actuels. Bien entendu cela suppose que le code du logiciel ait une structuration qui permet d'incorporer les changements de manière adéquate et que la plateforme d'accueil soit dotée de composants qui peuvent altérer le code du logiciel dynamiquement en fonction de la description du changement anticipé.

Les changements non anticipés sont des changements qui apparaissent une fois le logiciel livré. L'approche ontogénétique traite les deux types de changement de la même façon au niveau opérationnel. Cependant au niveau abstrait, le traitement diffère. La démarche

utilisée doit tenir compte d'un existant qu'il faut comprendre, dont il faut respecter certaines propriétés, etc.

Dans l'approche ontogénétique, le code du logiciel se compose de deux parties : Un code exécutable qui réalise les fonctionnalités du système, il est appelé Phénotype comme pour les organismes vivants, et un code nommé Génotype dont l'exécution façonne continuellement et dynamiquement le phénotype.

Autour du code ainsi formé, se trouve une suite de composants nécessaires représentant la plateforme d'exécution. Le schéma de la figure 4.2 décrit l'approche ontogénétique.

Les phases de développement du système ontogénétique concernent d'ingénierie et l'analyse aussi bien les besoins actuels des parties prenantes que ceux du futur anticipé. En effet, tous ses besoins seront des entées aux phases d'implémentation. Le Phénotype est opérationnel dès la livraison du produit, le génotype contient une structure spécifique contenant un ensemble de structures et contenant des versions anticipées du système au futur. L'arrivée d'un changement/évolution non anticipé représente une entrée au processus ***Analyse État actuel et futur du système***. Ce dernier effectue une analyse des changements, et les fournit à l'équipe de développement continu pour le codage, le test, et mise à jour du ***Composant Chargé de l'Altération du Génotype selon les Changements non Anticipés***.

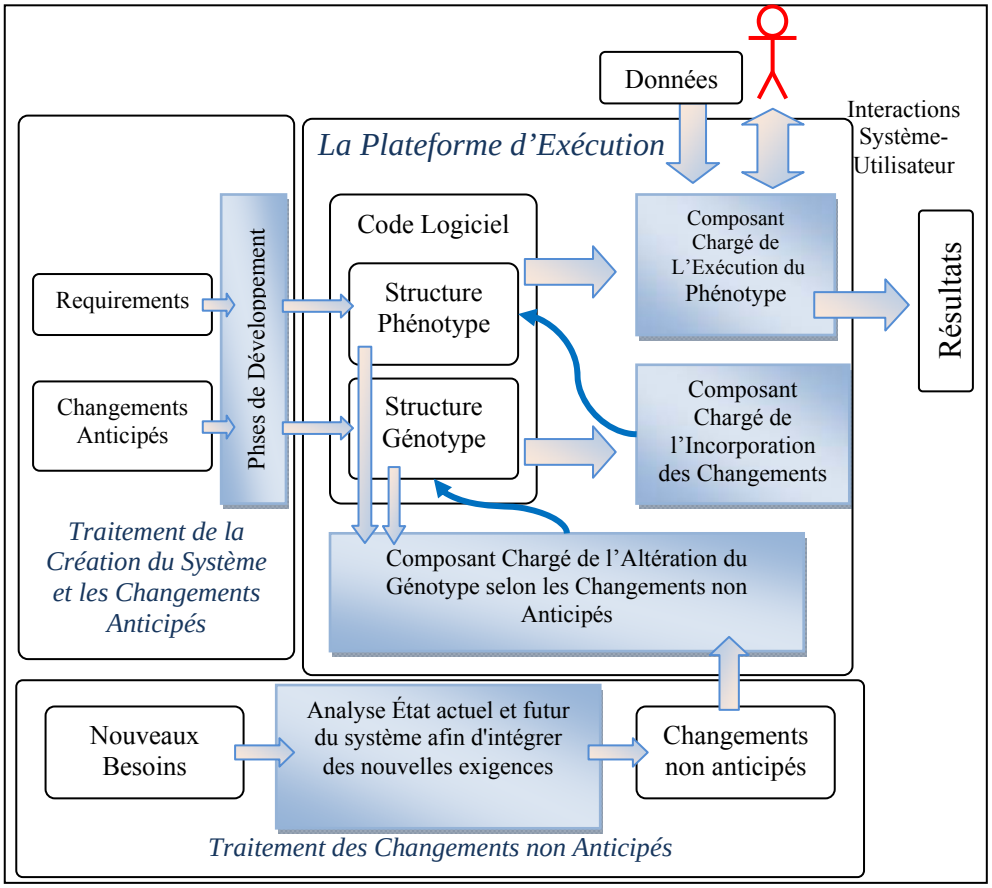

Figure 4.2 La vision ontogénétique d'un système logiciel

4.3 Onto-RUP : Une Extension du RUP

Le Rational Unified Process comme toutes les méthodes agiles a montré ses avantages et apports quant au développement des systèmes logiciels. C'est un cadre de processus fondé sur des pratiques avérées (voir Chapitre 2).

Le RUP et toutes les méthodologies de développement ne prennent pas en charge les évolutions futures des besoins durant le processus de développement. Nous proposons une extension du RUP pour l'adapter aux systèmes ontogénétiques. L'extension concerne l'adjonction de deux nouvelles disciplines et d'autres éléments de modélisation qui redonne à l'évolution un aspect fondamental dès le démarrage du processus du développement du système logiciel [Khe 14].

En raison des insuffisances des méthodes de développement classiques ou modernes pour conduire le développement des systèmes ontogénétiques, nous proposons l'extension du RUP. L'approche préserve tous les outils et atouts du processus RUP, et propose une extension pour l'adapter à notre contexte. L'extension porte sur deux nouvelles disciplines appelées respectivement Ant-Ev et Unant-EV et d'autres éléments de modélisation.

4.4 Onto-RUP : Un Nouveau Cycle de Vie pour le RUP

Nous proposons un nouveau cycle de vie pour le RUP. Le nouveau cycle concerne l'adjonction de nouvelles disciplines pour la modélisation des changements. Ceci va entrainer bien sûr une révision des objectifs des différentes phases du cycle de développement du RUP : Inception, Elaboration, Construction et Transition.

4.4.1 Axe Horizontal : Révision de la phase Inception

L'axe vertical de la structure représente l'aspect statique d'Onto-RUP. Donc quatre phases distinctes : Inception, Elaboration, construction et Transition. La phase d'inception du RUP comprend un ensemble d'activités riches et de conseils, ses objectifs sont :

- Comprendre le système à construire;
- Identifier les fonctionnalités clé du système;
- Déterminer au moins une solution possible;
- Comprendre les coûts, le calendrier et les risques associés au projet ;

- Décider sur le processus à suivre et les outils à utiliser.

La prise en compte le l'ingénierie des changements/évolution anticipés durant cette phase nous a conduit à l'adjonction des nouveaux objectifs suivants :

- Etude des tendances et évènements futurs
- Identifier les changements/évolutions anticipés;
- Comprendre les coûts, le calendrier et les risques des tendances futures

4.4.2 Axe Vertical : Nouvelles Disciplines

L'axe vertical de la structure d'Onto-RUP, représente l'aspect statique de la structure. Cet axe représente toutes les disciplines du RUP en plus les deux disciplines proposées dans cette approche et qui sont :

- la discipline AntRE et
- la discipline UnantRE.

4.5 La Discipline Ant-Ev d'Onto-RUP

La discipline ***Requirements*** du RUP utilise le modèle des cas d'utilisation comme des artéfacts englobant les besoins fonctionnels du système logiciel à produire. Le RUP est un cadre de processus extensible [Kro 03]. Nous proposons une discipline additionnelle Ant-RE, représentant une extension des disciplines RUP pour mener l'anticipation des besoins de changement/évolution.

Au début nous nous somme intéressés à la réponse de la question qui s'imposait fortement: « Existe-t-il un modèle qui représente une prospection du futur des cas d'utilisation? », dont la réponse était « Oui, il existe un modèle dit le ***modèle des cas de changement***, proposé par E. Ecklud [Eck 96] et qui représente une altération du modèle des cas de d'utilisation. Cependant, nous avons proposé dans [Khe 09a, Khe 09b] une démarche d'extraction de changement/évolution anticipé à partir du modèle des cas d'utilisation en utilisant une combinaison d'approches classiques d'élicitation des besoins tels que : Brainstorming, Interview, Questionnaires, et Groupwork. Néanmoins, ces approches ne sont pas appropriées pour l'étude des changements/évolutions des évènements et tendances futures du système logiciel à produire. Le processus d'élicitation des exigences de changement/évolution est un processus qui doit se baser sur un raisonnement intentionnel, à commencer par des objectifs de haut niveau et d'affiner ces objectifs jusqu'à atteindre ceux qui sont opérationnalisable. L'approche GORE se base sur la définition des exigences comme étant des buts qui peuvent être divisés et raffinés [Lam 09]. Cependant,

une possibilité de modéliser des évolutions/changements est l'utilisation des modèles orientés-but (cf. chapitre 3). La modélisation des besoins orientés buts a été proposée durant l'élicitation des besoins pour décrire le comportement organisationnel actuel. Également les techniques d'analyse orientées buts ont été utilisées dans le contexte de la négociation des besoins pour aider le raisonnement sur la nécessité de changement organisationnel, et de fournir le contexte dans lequel se produit la délibération durant l'ingénierie des besoins [Kav 06, Pim 11, Nur 02]. En effet, i* [Yu 95, Yu 97] fournit un mécanisme convenable pour représenter les comportements alternatives d'un système à travers les liens « moyen-finalité». Cette caractéristique intègre facilement les besoins futurs avec ceux actuels.

Cependant, nous proposons l'élaboration d'un modèle orienté but au cours des activités de la discipline proposée. L'altération/adaptation du modèle orienté but du système sert à représenter une altération du modèle des cas d'utilisation, d'où la création du modèle des cas de changement.

4.5.1 Objectifs de la discipline Ant-Ev

Avant de citer les objectifs de la discipline d'ingénierie des besoins de changement, nous rappelons ceux de la discipline RE du RUP.

L'élicitation des besoins pour un système logiciel est une activité conceptuelle complexe, nécessitant des efforts importants de mise en œuvre. C'est est un processus d'une grande importance permettant de mieux spécifier le système logiciel à produire. Cette activité comprend les étapes suivantes :

1. Comprendre le domaine de l'application
2. Identifier les sources des exigences
3. Analyser les besoins des parties prenantes
4. Choix des techniques, approches et outils les plus appropriés.

Les objectifs de la discipline de la gestion des exigences du RUP se résument donc dans les points suivants :

- Établir un accord entre les intervenants sur les objectifs du système à développer.
- Fournir aux développeurs du système une meilleure compréhension des exigences logicielles.
- Définir les limites du système à développer.
- Fournir un plan initial des itérations à réaliser.

- Fournir des estimations initiales des coûts, des échéanciers pour développer le système.
- Définir les interfaces graphiques en se basant sur les besoins des utilisateurs.

Le résultat de la combinaison de ces étapes est la spécification des besoins actuels du système en cours de développement.

Nous ajoutons aux objectifs cités précédemment les objectifs spécifiques à la discipline Ant-Re :

- ⇨ La construction du modèle orienté but associé à l'analyse des Extraction des évènements et tendances futures
- ⇨ La construction du modèle orienté but altéré ou adapté.
- ⇨ L'analyse du mapping i*/cas utilisation
- ⇨ La construction du modèle des cas de changement

4.5.2 Activités de la Discipline Ant-RE

La discipline Ant-RE propose, d'autres activités pour l'anticipation des évolutions futures. Ces activités permettent d'étudier les possibilités de prévoir les besoins futurs du système.

L'ingénierie des besoins doit poser la question POURQUOI développer un système et aider les parties prenantes du projet à y répondre [Rol 03]. Le rôle de l'ingénierie des besoins est de déterminer ensuite les fonctionnalités que le système doit mettre en œuvre pour aider à la satisfaction de ces buts et identifier les contraintes qui restreignent la mise en œuvre de ces fonctions. Ces buts, fonctions et contraintes, constituent les 'besoins' qui doivent ultérieurement être convertis en une spécification précise permettant le développement du système. Dans ce sens nous utiliserons le modèle des cas d'utilisation de la discipline RE du RUP comme entrée aux différentes activités de la discipline proposée. La discipline Ant-RE propose, d'autres activités pour l'anticipation des évolutions futures. Ces activités permettent d'étudier les possibilités de prévoir les besoins futurs du système.

Supposons qu'un modèle orienté-but et son modèle des cas d'utilisation correspondant sont déjà établit lors du processus d'analyse et spécification des besoins. Ces deux modèles représentent des entrées aux activités de la discipline proposée. Le développement du système logiciel se produit dans un contexte où les processus organisationnels sont bien établis [Bub 94]. La modélisation des besoins orientée but a été proposée durant l'élicitation des besoins pour décrire le comportement organisationnel actuel du système [Kav 06, Lam 01]. Dans ce contexte, nous utilisons le modèle i*, décrit en 3.2.2, pour exprimer l'ensemble des besoins actuels du système ontogénétique. Le choix d'utilisation

des modèle orienté-but et son adaptation a été inspiré des travaux de J. Pimentel et al. [Pim 11].

Également les techniques d'analyse orientées but ont été utilisées dans le contexte de la négociation des besoins pour aider le raisonnement sur la nécessité de changement organisationnel, et de fournir le contexte dans lequel se produit la délibération durant (Figure 4.3).

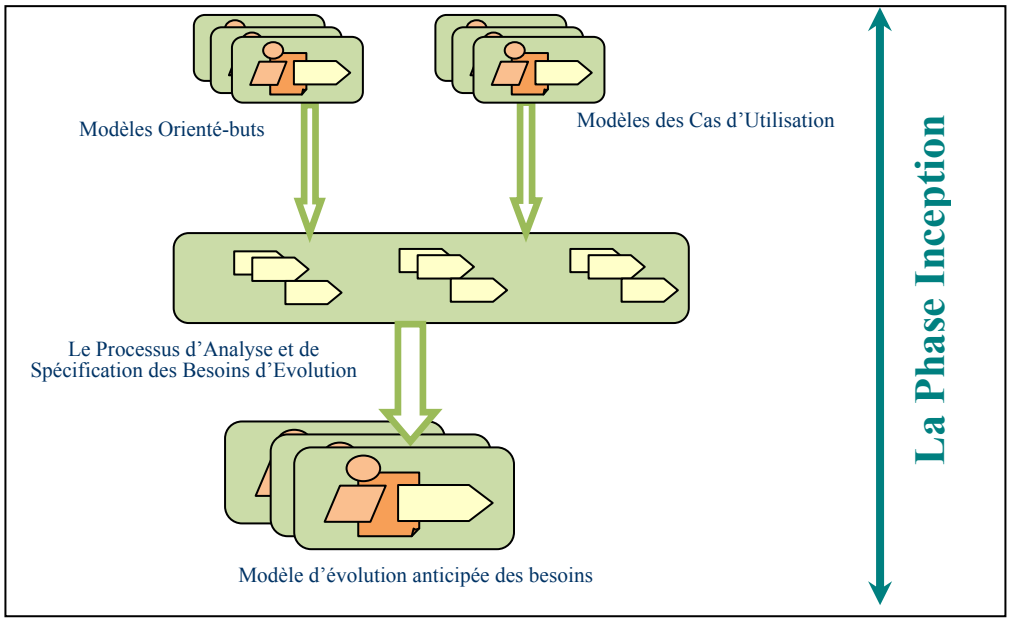

Figure 4.3 Les entées du processus d'analyse de l'évolution anticipée des besoins

4.5.2.1 Activité : Construction du modèle des buts

L'ingénierie des besoins doit d'abord s'intéresser aux buts du contexte organisationnel du système à développer parce qu'ils permettent de comprendre les raisons justifiant son développement. Le but de l'élaboration du modèle des buts est de permettre le raisonnement sur la nécessité de changement ultérieurement, c'est-à-dire juste après l'extraction des tendances et évènements futurs. L'intention de Changer est un but qui doit être satisfait avant d'être convertis ultérieurement en une exigence de changement anticipé dans le système à produire. Le modèle des buts est conforme à la notation i* [Yu 95, Yu 97] décrite en 3.2.2.

Exemple

Le but principal du « Movies For Me » est d'informer les personnes intéressées sur les horaires des téléfilms, qui réalisé par les tâches "Découvrir les horaires des téléfilms" et "Afficher les horaires téléfilms". La tâche "Découvrir les horaires des téléfilms" est achevée avec les deux tâches "Découvrir films Chaîne A" et "Découvrir films chaîne B". Chacune

d'elle est décomposée respectivement, en " Analyse site web de la chaîne", "Obtenir description films", "Obtenir date et horaires films" et "Obtenir images films".

Le but "Afficher calendrier téléfilms" est achevé avec les tâches "Liste des films sur site web" et les films peuvent être regroupés par jour d'exposition ou par chaîne, Figure 4.4.

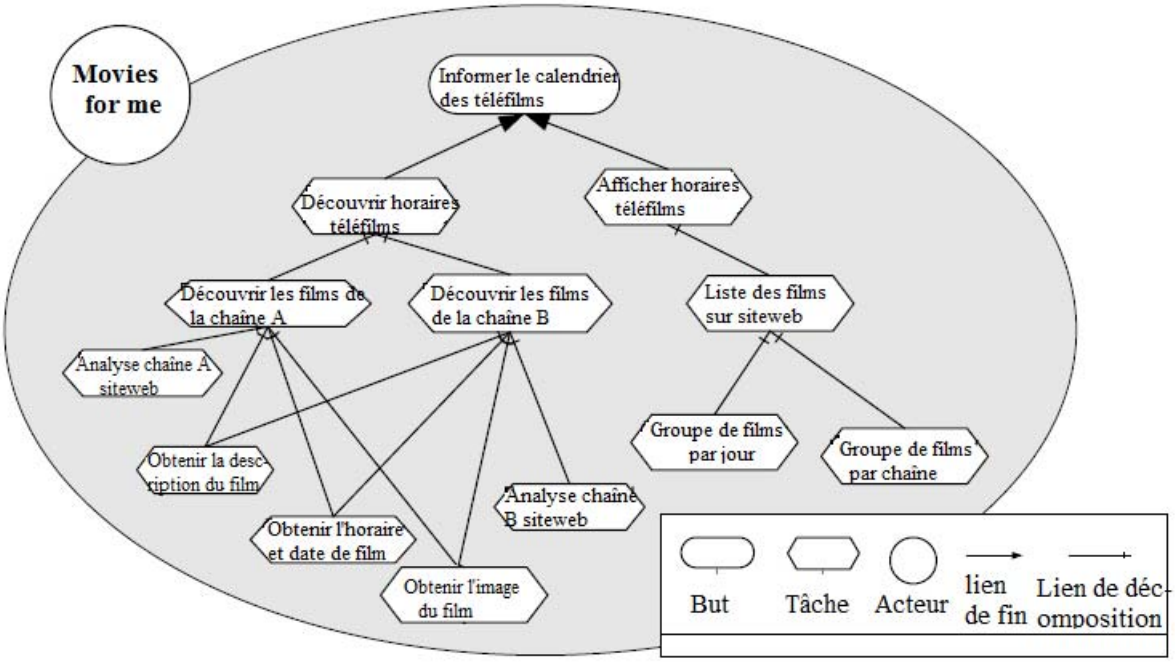

Figure 4.4 Le Modèle orienté-but du système « Movies for Me » [Pim 11]

4.5.2.2 Activité : Élicitation des Besoins Anticipés: Analyse Futures wheel

La méthode Futures wheel [Gle 72] semble plus appropriée pour l'élicitation des besoins futurs, car elle offre un processus qui :

1. fournit une image claire des événements futurs qui pourraient avoir une incidence sur le système,
2. est facile à comprendre et à utiliser par les parties prenantes,
3. nécessite moins d'effort que les autres approches [Gor 04], et est plus approprié pour l'ingénierie des besoins [Pim 11] et par conséquent, évite de compromettre le calendrier du projet.

Cette dernière caractéristique est très importante pour préserver l'un des principes du processus RUP : "la réduction de la complexité" du système à construire.

Comme décrit en 3.4.1, le processus Futures wheel comprend deux étapes, l'élicitation des évènements et leurs conséquences. A la fin du processus, on aura autant de diagrammes

que des évènements futurs. Un modèle d'un seul évènement se représente sous la forme présentée en figure 4.5.

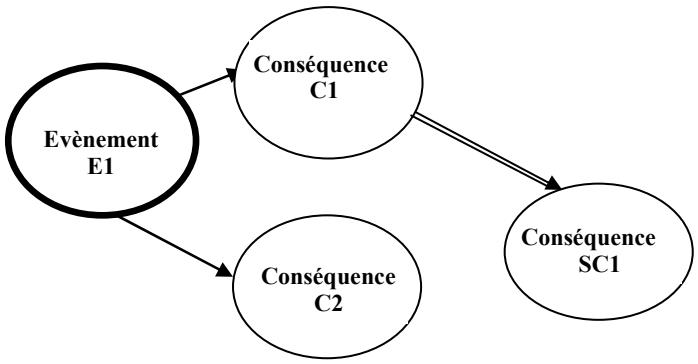

Figure 4.5 Exemple de notation Futures wheel

On attribue pour chaque évènement, un modèle Futures wheel. Un graphique dans lequel on peut voir toutes les conséquences possibles sortantes du cercle de l'évènement. L'événement est représenté par un cercle avec une bordure épaisse. Les conséquences sont représentées par un cercle avec une bordure normale. L'événement principal est lié aux conséquences primaires par une flèche sur une seule ligne; les principales conséquences sont liées aux conséquences secondaires par une ligne doublée et fléchée, et ainsi de suite.

Dans la figure 4.6 l'évènement Adoption réussie de la télévision numérique terrestre a comme conséquences directes :

- Plus de chaînes TV disponibles
- La disponibilité des chaînes TV dans les appareils mobiles
- La diffusion EPG (Description électronique du contenu diffusé)
- Et la haute définition

Par exemple une sous-conséquence de la conséquence « Plus de chaînes TV disponibles » sera : Le système aura besoin de rassembler les données sur des chaînes qui n'existent pas encore.

A son tour « Haute Définition » a une autre conséquence : Les gens sont plus susceptibles de regarder des films à la télévision.

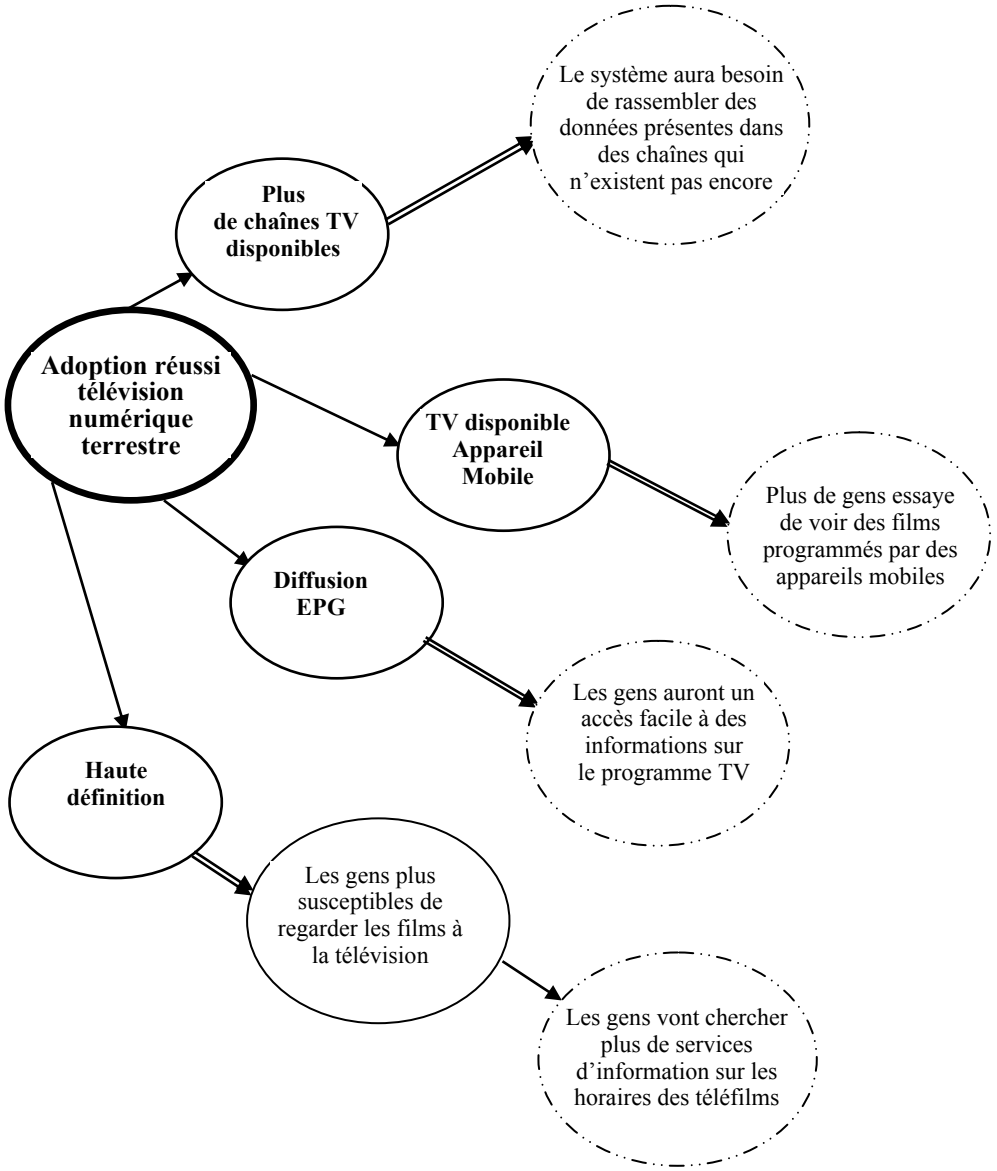

Figure 4.6 Le modèle Futures wheel pour l'évènement ″ adoption réussie de la TV terrestre numérique″

Un autre évènement analysé est « Croissance économique ». Comme montré dans figure 4.7, les conséquences considérées et relatives au domaine sont :

- Augmentation de demande de TV payante et,
- Augmentation d'accès à l'internet.

« Plus de personnes accédants à Internet » est une conséquence secondaire de cet évènement. Encore une fois, pour chaque conséquence feuille, des conséquences directes sont identifiées.

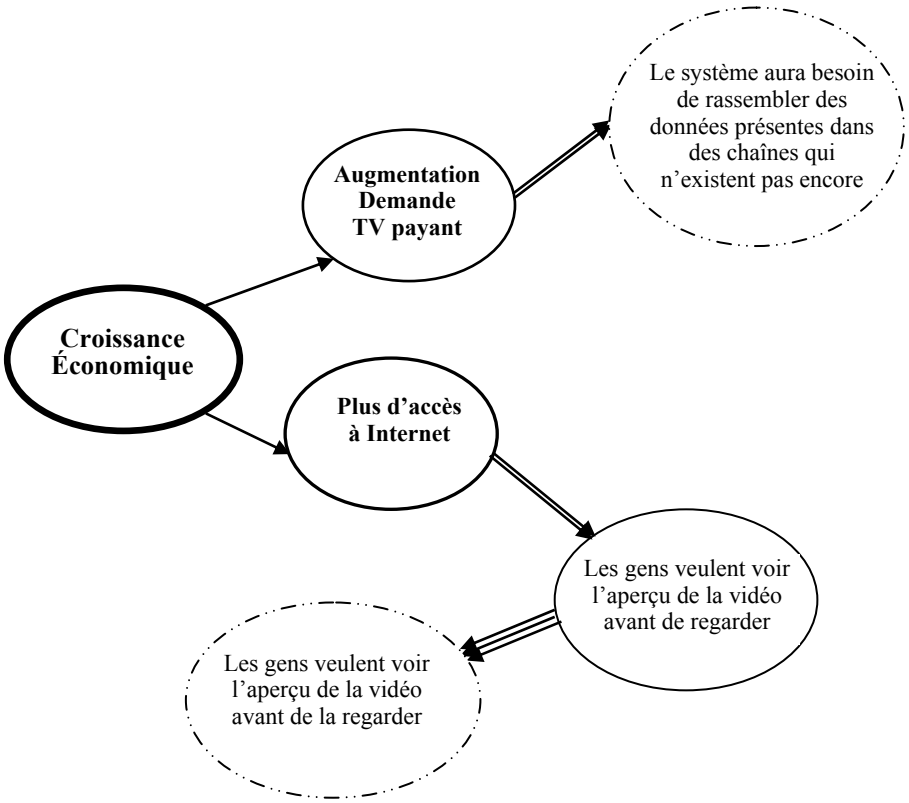

Figure 4.7 Le Modèle futures wheel de l'évènement "croissance économique" [Pim 11]

4.5.2.3 Activité : Adaptation du Modèle des Buts

Le résultat du processus d'élicitation des besoins futurs forme l'ensemble des Futures wheel, en d'autres termes le résultat de ce processus est en ensemble d'évènements associés à leurs conséquences. La modélisation des évolutions anticipée est un processus qui utilise le modèle des Futures wheel comme entrée. Cette modélisation s'effectue en deux étapes.

Étape 1 : Extension du Modèle Futures wheel

Le résultat du processus d'élicitation des évolutions anticipées est la construction du modèle Futures wheel qui contient l'ensemble des évènements ainsi que leurs conséquences associées. À ce stade, il y a encore un grand écart entre les conséquences et la configuration système requise. Ainsi, pour chaque conséquence feuille, c'est-à-dire les conséquences qui n'ont pas d'autres conséquences, on se demande comment cette conséquence affecte-t-elle le système ? On qualifie ces conséquences de directes, car elles sont directement liées au système. Pour expliciter quelles sont les conséquences directes, il est préférable de les représenter comme des cercles avec une bordure en pointillée (Figure 4.8).

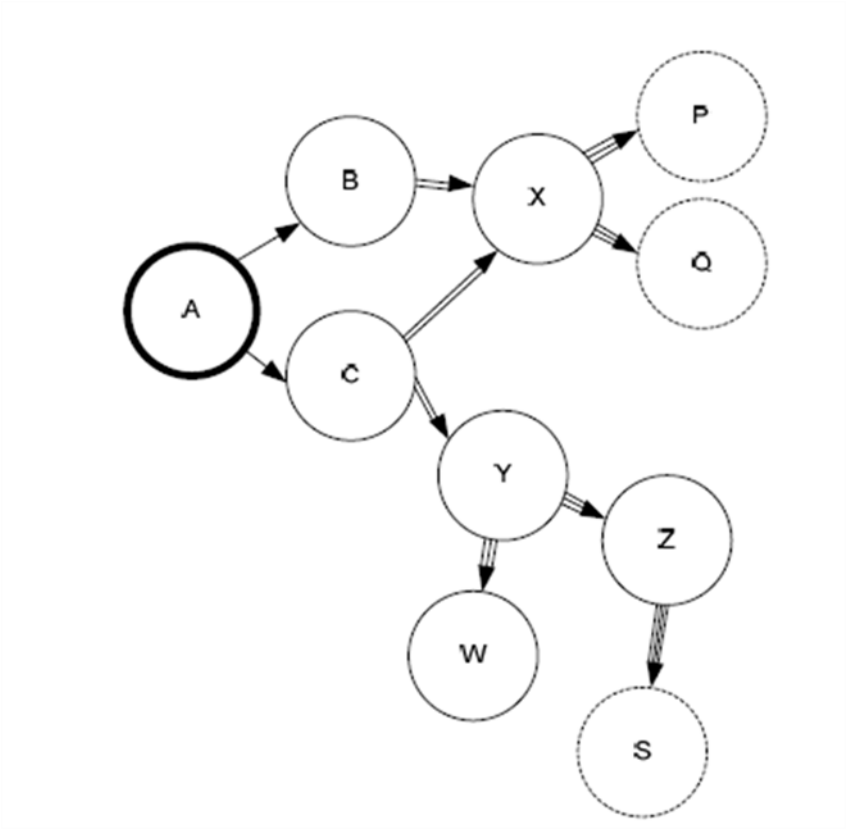

Figure 4.8 Exemple de notation du modèle Futures wheel étendu

Dans l'exemple étudié, les deux figures (Figure 4.6) et (Figure 4.7) représentent respectivement les modèles Futures wheel étendu des deux évènements adoption réussie de la TV terrestre numérique et "croissance économique".

Étape 2 : Création du Modèle Orienté-But Adapté

La deuxième étape dans le processus proposé est l'altération du modèle orienté-but existant. Ce modèle ne remplace pas le modèle initial, mais utilise le modèle en entrée pour la création d'un nouveau modèle dit ***le modèle orienté-but altéré***. L'altération du modèle des besoins orienté-but s'effectue après l'analyse du modèle Futures wheel. Cette analyse permet de voir, vérifier, et comparer pour répondre à la question « comment ces conséquences pourront-elles altérer le modèle orienté-but existant? ».

Les analystes doivent par la suite répondre à cette question en effectuant les changements appropriés dans le modèle initial conformément aux conséquences directes collectées lors de l'analyse Futures wheel.

Le tableau 4.2 montre les changements qui ont été apportés au modèle des buts, pour chaque conséquence directe. Pour résoudre les conséquences A et F, la tâche "Ajouter

ajout un support pour une nouvelle chaîne " doit être ajouté. De cette manière, le système devra être capable d'obtenir des informations à partir d'une nouvelle chaîne informé par son utilisateur. Le softgoal "Soyez moteur de recherche convivial" a été ajouté pour aborder la conséquence B, de sorte que lorsque les gens recherchent le calendrier des films TV, ils atteignent le site web "Movies For Me ". Il était également ajouté un softgoal et une tâche qui contribue positivement à ce softgoal.

L'intention d'aborder la conséquence C est le softgoal "Portabilité". En outre, afin de satisfaire le softgoal "Portabilité", la tâche "Liste des films sur un site web" est décomposée en trois tâches supplémentaires, de telle sorte que chaque type d'appareil dispose d'un site web spécifique. La conséquence E est abordée par l'ajout de la tâche "Obtenir Teaser/Trailer", de sorte que le système peut fournir un aperçu de la vidéo des films pour ses utilisateurs.

Conséquences Directes	Impact Spécifique
(A) Le système aura besoin de recueillir des données à partir des chaînes qui n'existent pas	Ajouter la tâche "ajout un support pour une nouvelle chaîne "
(B) Les gens vont chercher plus de services qui leur informent sur le calendrier des chaînes TV.	Ajouter le softgoal " Etre moteur de recherche convivial"; ajout le softgoal "Utiliser de bon mots clé"; ajouter la tâche "Utiliser liens sponsorisés".
(C) Plus les gens vont essayer de voir le calendrier des films à travers des dispositifs mobiles.	Ajout le softgoal "Portabilité" ; Ajout la tâche "site web pour PC et PC portable" ; Ajout la tâche " site web Spécifique pour des dispositifs mobiles" ; Ajout la tâche "Site web Spécifique pour TV".
(D) Les gens auront un accès facile aux informations sur le calendrier de la télé	Aucun
(E) Les gens voudront voir un aperçu vidéo du film avant de le regarder réellement.	Ajouter la tâche "ajout un support pour une nouvelle chaîne "
(F) Le système devra recueillir des données à partir de chaînes de télévisions payantes.	

Table 4.2 Information de traçabilité des conséquences directes et leur impact sur le modèle des buts

Toutes les conséquences directes résultats de l'analyse ou la modélisation Futures wheel sont prises en compte pour la création du modèle orienté-but adapté.

Le modèle orienté-but altéré du système « Movies for Me » est donné dans la figure 4.10.

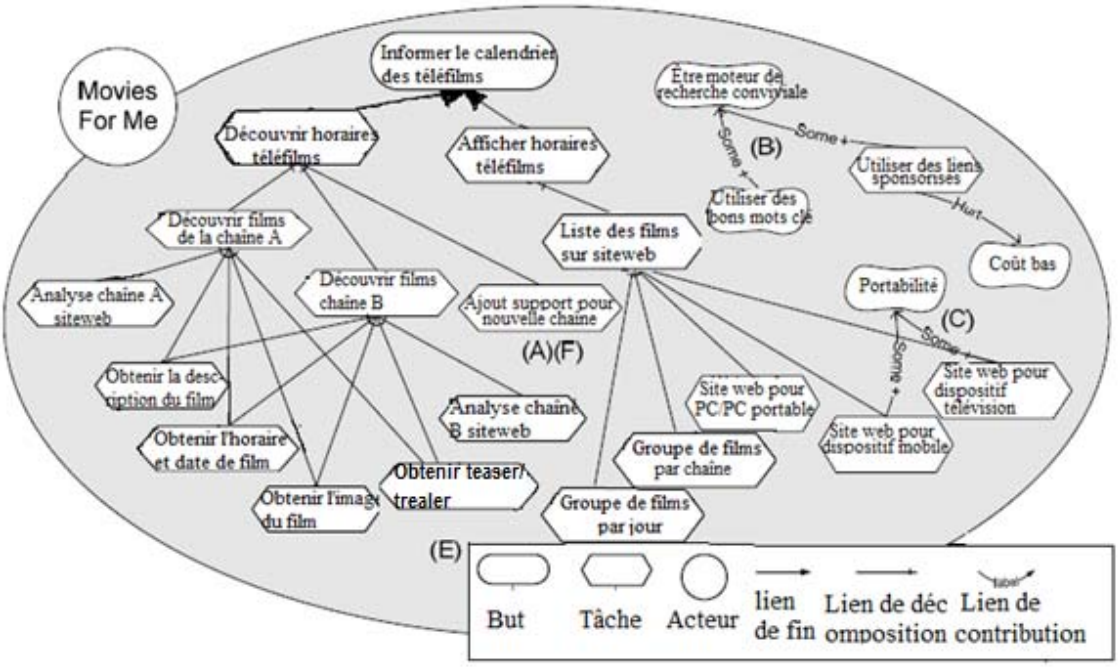

Figure 4.10 Le Modèle orienté-but altéré du système « Movies for Me » [Pim 11]

4.5.2.4 La Construction du Modèle des Cas de Changement

La discipline RE du RUP utilise le modèle de cas d'utilisation comme étant la structuration et la spécification des besoins fonctionnels du produit en cours de production. Donc, notre approche préserve l'utilisation des cas d'utilisation comme étant le noyau de la discipline d'ingénierie des exigences. A ce stade, une question se pose : quelle sera la relation entre le modèle des buts, le modèle des buts adapté et le modèle des cas d'utilisation ?

Nous avons proposé dans [Khe 9a], [Khe 9b] une approche basée sur les cas de changements [Eck 96] pour la modélisation des changements à partir du modèle des cas d'utilisation et une extraction d'un ensemble d'opérateurs appelés opérateurs de changement. Néanmoins, comme signalé précédemment, il est plus adéquat pour le raisonnement sur la nécessité de changement d'utiliser l'approche GORE. Wautelet et al. ont proposé dans [Wau 13a, Wau 13b] un mapping pour intégrer le modèle i* dans le RUP/UML.

Donc, il est possible d'extraire un modèle de cas d'utilisation RUP/UML à partir d'un modèle orienté-but i*.

Le modèle de cas de changement constitue le différentiel qui existe entre le modèle de cas d'utilisation mappé à partir du modèle des buts initial et le modèle de cas d'utilisation mappé à partir du modèle des buts adaptés suite à l'évolution.