THESE

présentée en vue de l'obtention du

DOCTORAT DE L'UNIVERSITE TOULOUSE I

Spécialité : **INFORMATIQUE**

par

Marcelo Soares PIMENTA

TAREFA: Une Approche pour l'Ingénierie des Besoins des Systèmes Interactifs

Soutenue le 23 Octobre 1997 à l'Université Toulouse I, devant le jury composé de :

Mme M.F. BARTHET	Professeur à l'Université Toulouse I	Directeur
M. G. BOY	Directeur de Recherche à EURISCO à Toulouse	Rapporteur
M. C. JOHNSON	Professeur à l'University of Glasgow	Rapporteur
M. P. PALANQUE	Maître de Conférence à l'Université Toulouse I	Examinateur
M. D.L. SCAPIN	Directeur de recherche à l'INRIA, Rocquencourt	Examinateur

---

Laboratoire d'accueil :

L.I.S. - Université Toulouse I, Place Anatole-France, F-31042 Toulouse Cedex

L'université n'entend ni approuver
ni désapprouver les opinions particulières du candidat

A minha esposa Ângela e ao meu filho Pedro
com todo o meu amor

Remerciements

Je tiens à remercier Marie-France Barthet pour avoir accepté ce défi qui est le sujet de thèse qui j'avais choisi auparavant, pour m'avoir accueilli dans son laboratoire et pour m'avoir accordé sa confiance au long de ce travail.

J'exprime également mes remerciements à Philippe Palanque pour sa disponibilité à travailler, pour son enthousiasme à m'aider et pour la qualité des commentaires résultants de ses relectures.

Mes remerciements vont aussi à Chris Johnson et Guy Boy, pour l'intérêt qu'ils ont manifesté envers ce travail, pour leur lecture minutieuse de cette thèse et pour leurs remarques pertinentes à ce travail. Je remercie également à Dominique Scapin pour avoir accepté de participer au jury et pour ses encouragements.

Merci aux membres du laboratoire pour leur accueil sympathique, et leur soutien moral dans les plusieurs situations de stress. Merci à Christelle Farenc pour ses diverses lectures. Merci en particulier, à 'Herr' Alassane Cissé, Jaime 'El Cabron' Munoz, Ndiaye Souleimane, Le Duc Hoa et Narjes Bellamine pour leur disponibilité et pour leur lutte quotidienne pour créer un cadre de travail stimulant.

Meus mais profundos agradecimentos sao sem duvida para minha esposa Ângela Foletto e meu filho Pedro. É uma grande alegria que voces estejam ao meu lado, enfeitando minha vida. Esta tese é dedicada a vocês dois com muito carinho. Agradeço tambem às nossas familias e aos nossos amigos pelo encorajamento durante todo este periodo em que estivemos distantes.

Le travail de recherche de cette thèse a été financé par une bourse CAPES du gouvernement du Brésil, avec le soutien du Departamento de Informatica e Estatistica (INE) da Universidade Federal de Santa Catarina (UFSC) à Florianopolis, Brésil.

Sommaire

Partie II : La Proposition de TAREFA

Chapitre IV - TAREFA: Fondements et Vue d'Ensemble

Chapitre V - TAREFA : La macroactivité d'Analyse

Introduction Générale

Interagir avec des systèmes informatiques fait partie aujourd'hui de notre vie quotidienne. Ces systèmes soutiennent une gamme étendue des activités humaines, des plus simples comme la préparation des additions aux restaurants, aux plus complexes comme le contrôle du trafic aérien. Il est essentiel que ces systèmes soient utiles et utilisables, ce qui signifie qu'il faut **bien** les concevoir. Concernant la définition de ce 'bien', de nombreuses questions se posent dans le domaine de l'Interaction Homme-Machine et les réponses influeront largement et inévitablement sur la manière dont les besoins du système seront spécifiés. L'objectif est ici de fournir aux utilisateurs plus de satisfaction et d'efficacité dans la réalisation de leurs tâches avec le système, tout en restant proche des problèmes de construction de ces systèmes là, dans une optique du génie logiciel.

En effet, au début des années 80, il a été largement reconnu que les principaux problèmes pour améliorer la qualité du logiciel et la productivité de son développement étaient concentrés dans les premières parties du cycle de développement du logiciel. C'est bien sûr une des raisons lesquelles, l'intérêt pour le processus d'**Ingénierie des Besoins** est croissant.

Le processus d'Ingénierie des Besoins' (*Requirements Engineering* en anglais) consiste à comprendre le domaine du problème et à utiliser cette compréhension pour identifier les besoins (buts de l'utilisateur, fonctions du système et contraintes imposées à son développement) et pour spécifier un modèle du système qui les prenne en compte.

Présentation de la problématique de recherche

L'Ingénierie des Besoins a toujours été un processus complexe mais cela parait encore plus difficile pour les systèmes interactifs. En fait, pour ces derniers, la notion d'utilisabilité est aussi importante que la notion d'utilité, qui a longtemps été la notion primordiale pour les systèmes informatiques traditionnels. L'utilité est la valeur du rapport qu'il existe entre les fonctions offertes par le système et celles demandées dans les tâches de l'utilisateur. L'utilisabilité est la valeur du rapport entre la manière pour un utilisateur donné de réaliser une tâche interactive et les caracteristiques - cognitives, physiques et niveau (novice, expert) - de cet utilisateur [Farenc 95]. L'utilisabilité est un facteur critique pour la satisfaction et la productivité des utilisateurs et elle n'est perçue qu'à travers l'interface utilisateur du système.

La conception de logiciels interactifs peut être vue comme étant composée de deux dimensions de base: la dimension interne , qui correspond à la construction du composant sémantique (dénommé également application); et la dimension externe, qui correspond à la construction de l'interface homme-machine. Bien sûr, concevoir des systèmes informatiques interactifs nécessite l'intégration de concepts et techniques spécifiques à l'Interaction Homme-Machine (IHM) et de concepts et méthodes de développement de systèmes - traditionnellement considérés comme faisant partie du Génie Logiciel (GL).

Dans les domaines du Génie Logiciel (GL) et de l'Interaction Homme-Machine (IHM) la phase d'Analyse est interprétée de manière différente, avec différents types d'informations recueillies et différentes techniques de recueil de ces informations. En

fait, les besoins déterminés par l'Analyse du GL sont insuffisants pour concevoir les aspects de l'interaction, et inversement l'IHM se concentre en général sur l'interaction sans une considération adéquate des autres besoins du logiciel qui doivent aussi être pris en compte, comme par exemple la fiabilité (*reliability*), la maintenabilité, la portabilité et la réutilisabilité des spécification ou du code. Ce problème d'intégration interdisciplinaire, se manifeste déjà au début de l'analyse, continue pendant tout le cycle de vie du logiciel interactif [Barthet 88], et est en partie dû à une mauvaise compréhension et mise en oeuvre des modèles d'architecture tels que Seeheim [Green 86], qui propose de décomposer un système interactif en trois composants distincts : le noyau fonctionnel , le dialogue et la présentation. Ce découpage systématique et artificiel résulte en un **fractionnement des besoins** car chaque domaine ne considère que quelques aspects différents et souvent disjoints du système sans aucune correspondance explicite et systématique entre eux. Sans cette correspondance, il n'y a pas interaction possible entre les experts de chacun des domaines, qui deviennent de fait incapables d'analyser les impacts des besoins déterminés par les autres experts sur ceux déterminés par eux-mêmes.

De plus, l'évolution des méthodes du GL s'est effectuée moins rapidement dans les vingt dernières années que les systèmes eux-mêmes et la manière dont on les utilise. Ainsi, les approches traditionnelles de développement de logiciels se montrent souvent inadéquates et inopérantes pour développer des systèmes interactifs, ne couvrant pas ou couvrant à peine les deux dimensions. Cela finit par générer des systèmes inefficaces à la fois en termes d'utilité et d'utilisabilité. Le principal reproche que l'on peut faire à ces méthodes de conception traditionnelles est qu'elles n'incitent pas les informaticiens à compléter leur conception du logiciel selon le point de vue de l'utilisateur, et les mènent à déduire la conception du composant de dialogue à partir des fonctions de l'application [Barthet 1993]. Il est encore commun de trouver des logiciels conçus avec peu ou aucune attention portée à l'activité humaine qu'ils sont censés soutenir. Ceci s'inscrit dans une tendance générale à traiter le développement des systèmes interactifs comme un 'cas spécial' du développement de logiciel.

Nous pensons au contraire qu'il faut considérer les systèmes interactifs comme un sujet en lui-même. En effet, les travaux de recherche récents convergent sur l'idée centrale que pour développer un logiciel interactif qui soit à la fois utile et utilisable nous avons besoin d'une perspective **multidisciplinaire**, qui intègre dans un cadre fédérateur une variété de méthodes, théories et personnes de différentes disciplines, découlant notamment du GL et de l'IHM. Notre travail correspond à un effort d'intégration des concepts du Génie Logiciel et de l'Interaction Homme-Machine, plus particulièrement pour **l'ingénierie des besoins des systèmes interactifs** et il suit la tendance actuelle de changement d'attitude du génie logiciel vis à vis des facteurs humains.

En fait, depuis le début des années 80 l'Informatique subit un changement de paradigme qui fait que l'objet d'étude contient non seulement les composants informatiques (logiciel et matériel) mais aussi les personnes qui interagissent avec (utilisent) ces composants pour réaliser leurs tâches [Denning 95]. Même les principes théoriques fondamentaux (comme par exemple la machine de Turing) commencent à être remis en question à cause de leur incapacité à prendre en compte la diversité des possibilités originaires de l'interaction entre les êtres humains et les ordinateurs [Wegner 97]. En effet, la conception d'un système interactif demande la spécification d'un comportement beaucoup plus riche que la conception d'un système non interactif. Cela exige un changement de point de vue de la part de ceux qui développent les logiciels et des scientifiques de l'Informatique: **dévier le centre d'attention du système**

d'Informatique vers l'utilisateur et **du produit final (logiciel) vers son processus de développement.** Ce changement d'attitude se confirme dans les travaux récents, tels ceux de [Sommerville & Monk 94], qui expliquent la difficulté de formalisation de ce processus; et [C. Johnson & Jones 97], qui regroupent des contributions dans le domaine de la conception centrée utilisateur spécifiquement pour l'ingénierie des besoins. L'intérêt croissant des travaux comme les méthodes et ateliers de développement basée sur modèles de tâches (*task-based approaches* et *model-based approaches*) et la conception participative (*participatory design*) est aussi un exemple de ce changement d'attitude. L'IB doit donc considérer non seulement le point de vue de l'utilisateur mais aussi comment la conception va correspondre à ses besoins dans la situation d'utilisation, c'est à dire, les processus de travail et de communication des utilisateurs et la manière par laquelle ils sont affectés par le système.

Cependant, dans le domaine du GL, l'utilisation véritable du point de vue de l'utilisateur dans les méthodes de développement reste encore une question ouverte, ce qui rejoint le constat déjà ancien, mais toujours d'actualité, de [Barthet 88]. Par exemple, l'analyse et la modélisation des tâches ne sont pas représentés explicitement dans les principaux modèles de développement logiciel (cascade, 'V', spirale). En particulier, comme bien le fait remarquer [Hix & Hartson 93], pratiquement aucune des approches du GL ne fournit d'aide ou de démarche pour l'intégration des modèles de tâche au processus de l'ingénierie des besoins. Toutefois, à l'heure actuelle, l'ingénierie des besoins des systèmes interactifs reste encore non systématisée et fondée en général sur l'expérience et la subjectivité de l'analyste.

Bien que la prose en compte du point de vue de l'utilisateur soit considérée comme fondamentale dans le domaine de l'IHM [Myers 94], [Shneiderman 87], [Laurel 91], [Scapin 93] cette considération se concentre surtout sur la dimension externe du logiciel et non sur l'ensemble logiciel. Les méthodes de développement d'interface peuvent ainsi être critiquées comme superficielles pour l'ingénierie des besoins parce qu'elles ont une préoccupation principale centrée sur l'utilisabilité de l'interface au détriment de la fonctionnalité, sans préoccupation pour d'autres critères de qualité du logiciel dans sa totalité et sans connexion avec les méthodes de développement de systèmes.

La nécessité d'intégration des techniques et modèles dans un cadre multidisciplinaire est essentiel pour le développement des systèmes d'information interactifs [Andriole 93]. En particulier, les contributions des domaines de GL et IHM peuvent être utilisés de façon complémentaire. Cette constatation est à l'origine de plusieurs approches d'intégration de méthodes de conception de logiciel et de techniques de conception d'interface homme-machine, notamment les approches DIANE, TRIDENT, ADEPT et MUSE. Ces approches ont proposées différentes façons de concilier les deux domaines pendant la conception du logiciel, particulièrement en identifiant comment des techniques spécifiques peuvent se compléter l'une à l'autre pour réussir à arriver aux objectifs de la conception d'un système interactif. Bien que les résultats obtenus par toutes ces approches soient très satisfaisants, elles ont encore des lacunes importantes, notamment le fait qu'aucune n'ait été conçue spécifiquement pour l'Ingénierie des Besoins ce qui a pour effet l'insuffisance de la prise en compte du contexte pour la détermination des besoins. En outre, elles manquent toutes de liaison à une méthode informatique orientée objet, ce qui est très important aujourd'hui dans la mesure où la plupart des implémentations des systèmes interactifs suivent une approche orientée objet.

Hypothèse Générale

L'ensemble de considérations de la problématique nos permet d'énoncer l'hypothèse générale sur laquelle se base notre travail :

La prise en compte du contexte est indispensable pour la conception de systèmes qui soient plus utiles et plus utilisables.

Le terme 'contexte' fait ici référence à la fois aux aspects relativement stables pendant la conception et utilisation du système comme l'organisation, le langage du domaine, les objets et les tâches et aux aspects relativement instables comme les différentes situations d'utilisation possibles; Le contexte détermine les objectifs, les ressources et les contraintes d'un système.

Nous pensons que les méthodes actuelles de conception de logiciel ne permettent pas la conception de systèmes à la fois utiles et utilisables parce que le contexte n'est pas prise en compte de façon adéquate.

Nous pensons aussi qu'il est préférable que la prise en compte du contexte soit faite le plus tôt possible dans le cycle de vie, ce qui justifie notre choix pour l'étape de l'ingénierie des besoins. En effet, la prise en compte du contexte risque de remettre en cause la conception et nous savons que tout ce qui risque de le faire, doit être fait le plus tôt possible dans le cycle de vie. En plus, l'information sur le contexte est très informelle et donc elle n'est intégrable que dans le phases amont du cycle de vie qui sont les phases les plus informelles.

Structure de la proposition

Le but de cette thèse est de montrer que l'hypothèse est vraie et de proposer une approche pour la prise du contexte dans l'ingénierie des besoins des systèmes interactifs. Ainsi nous proposons **TAREFA** (Task Analysis based Requirements Engineering Framework and Approach), une approche pour déterminer et modéliser les besoins d'un système interactif, dont le résultat conduit à une spécification orientée objet de ce système. TAREFA a été conçue spécifiquement pour systématiser l'Ingénierie des Besoins des systèmes interactifs et est basée sur l'intégration des concepts, modèles et techniques du GL et de l'IHM.

TAREFA constitue un mode d'emploi opérationnel, dans la mesure où elle propose des modèles pour représenter l'information , des modes d'articulations entre les différents éléments de ces modèles et les démarches précises pour leur construction. TAREFA soutient les deux macroactivités fondamentales et complémentaires de l'Ingénierie des Besoins:

- la macroactivité d'Analyse, qui a pour but de comprendre l'univers du domaine du problème et d'identifier les besoins initiaux des utilisateurs. Cette compréhension est basée sur l'integration de différents modèles décrivant le contexte et les différents aspects de l'espace problème ;

- la macroactivité de **Synthèse**, qui a pour but de déterminer et d'exprimer les besoins du système pour atteindre les besoins des utilisateurs, de préférence en représentant l'historique des décisions et les justificatives des décisions qui ont été retenues.

Ces deux macroactivités sont réalisés itérativement et incrementalement : parfois il faut reprendre la macroactivité d'analyse avant de poursuivre la synthèse. En plus, chacune de ces macroactivités est composée par l'entrelacement d'activités systématiques de l'analyste et d'activités coopératives entre les acteurs du processus de l'IB. TAREFA propose que ces deux types d'activités - systématiques et émergentes - sont complémentaires et doivent être prises en compte. Les activités systématiques

correspondent aux activités de l'IB qui suivent la structure de contrôle d'une démarche. Dans l'IB, les activités émergentes correspondent aux activités non-systematiques de la coopération entre les acteurs pour intégrer leur compétences multidisciplinaires et pour la négociation, la prise de décision et la résolution de problèmes de manière collective. Alors, une démarche pour les activités systématiques et un modèle de processus émergent pour les activités émergentes (coopératives) sont proposés.

Contexte de travail

TAREFA a été conçue dans le but d'être appliqué dans un domaine précis : systèmes d'information interactifs 'maison' dont l'interaction est réalisé par l'intermédiaire des interfaces monoutilisateur du type "WIMP[1]". Un exemple typique de système dans ce domaine est le logiciel d'un distributeur de billets ATM utilisé dans cette thèse pour illustrer la présentation de TAREFA. Par contre, TAREFA intervient très tôt dans les phases du cycle de vie et donc serai facilement adaptable et en grand partie réutilisable pour les autres types d'application.

Le domaine de systèmes 'maison' a été choisi parce que dans ce type de systèmes les informations sur le contexte et les intervenants qui sont concernés par ces informations sont clairement identifiés. Les interfaces par manipulation directe et multiutilisateur se situent au-delà du champs d'application de ce travail : les premiers parce que ils sont influencés par des considérations principalement techniques (comme dispositif d'entrée et rendu graphique) que par le contexte et les deuxièmes parce que ils sont un champs de recherche en soi et sa prise en considération biaiserai le travail que nous voulions présenter. Toutefois nous sommes conscientes que les systèmes (et leurs interfaces) multiutilisateurs sont des systèmes très importantes. C'est pour cela que nous avons accepté un défi et nous avons appliqué TAREFA à un étude de cas d'un système d'information interactive 'maison' qui est multiutilisateur, présenté au chapitre 8.

Structure de la thèse

Cette thèse est structurée en trois parties distinctes.

La *première partie* est consacré à la présentation des travaux existants dans le domaine de systèmes interactifs , de l'ingénierie des besoins, puis plus spécifiquement l'ingénierie des besoins des systèmes interactifs.

- Nous débutons cette partie au *chapitre I* par une revue des concepts des systèmes interactifs qui vont intervenir dans la compréhension de notre travail ;

- Le *chapitre II* fait un tour d'horizon de l'ingénierie des besoins (définitions basiques, techniques, activités et principalement les problèmes typiques) en suivant un point de vue du Génie Logiciel. Ensuite, nous examinons pourquoi et comment ces concepts doivent être modifiés pour faire face aux spécificités des systèmes interactifs ;

- Le *chapitre III* de ce mémoire présente les approches les plus significatives utilisées pour l'ingénierie des besoins à la fois dans le domaines du GL et celui de l'IHM, ainsi que les tentatives d'intégration de ces deux domaines dédiées à l'ingénierie des besoins des systèmes interactifs. Nous avons choisi de comparer et de mettre en évidence la nature, les avantages et les inconvénients de ces intégrations, ce qui nous a permis de positionner nos travaux par rapport à l'ensemble de ces travaux existants.

1 WIMP (Windows, Icons, Menus et Pointers) par opposition au type d'interface utilisateur par manipulation directe.

La ***deuxième partie*** de ce mémoire contient la description de l'approche TAREFA (TAsk based Requirements Engineering FrAmework) que nous avons proposé.

- Le *chapitre IV* présente les fondements et une vue de l'ensemble de TAREFA. En particulier, sont évoqués les prémisses de la proposition de TAREFA, son concept des besoins, son organisation en deux macroactivités d'Analyse et Synthèse en prenant en compte l'entrelacement des activités systématiques et des activités coopératives.

- Le *chapitre V* présente la démarche et les modèles associés à la macroactivité d'Analyse ;

- Le *chapitre VI* présente la démarche et les modèles associés à la macroactivité de Synthèse.

Pour illustrer l'utilisation de TAREFA, dans chacun de ces 3 chapitres, les modèles devant être construits sont exemplifiés en prenant des extraits de l'étude de cas d'un distributeur des billets (ATM). Bien qu'il soit simple, nous pensons que l'exemple est suffisamment complexe pour illustrer à la fois le processus et les modèles de TAREFA.

- Le *chapitre VII* présente le modèle de processus basé sur le concept de l'émergence, proposé pour les activités coopératives de TAREFA.

La troisième partie de ce mémoire contient le *chapitre VIII*, qui présente une étude de cas, l'ingénierie des besoins pour un Outil d'Aide à la Planification Cooperative des Opérations du Satellite Multi-Instrument SOHO.

Partie I : L’Etat de l’Art

Chapitre I - Systèmes Interactifs : un tour d'horizon

Les systèmes interactifs, comme l'expression indique, sont des systèmes qui soutiennent la communication bidirectionnelle entre l'utilisateur et l'ordinateur pour la réalisation d'une activité humaine. Pour cette communication, il est utilisé aussi le terme 'Interaction homme-machine'[2].

L'objectif de cette thèse est l'ingénierie des besoins des systèmes interactifs donc nous allons présenter un état de l'art sur les systèmes interactifs. L'intention n'est pas être exhaustif et complet (si cela est possible !) mais surtout de souligner progressivement les concepts et caractéristiques des systèmes interactifs et de leur développement qui vont intervenir dans la compréhension du travail.

1.Contexte de travail

La caractéristique la plus fondamentale de tous les systèmes interactifs est ***le soutien à l'activité humaine*** que nous considérons ici comme des activités de travail et pas de loisir.

Un système interactif peut nous aider à réaliser nos activités de façon plus facile, plus précise, avec moins d'erreurs et de façon plus agréable ou satisfaisante. Ces arguments servent souvent à justifier l'argent investi dans son développement. En effet, un système doit être transparent dans le contexte d'usage et permettre à l'utilisateur de travailler directement sur le domaine du problème et de réaliser ses tâches [Weiser 94] .

En contrepartie, un système inadapté à l'activité interfère comme un obstacle à sa réalisation et il devient une source de problèmes. Par conséquent, l'utilisateur a besoin de plus d'effort pour réaliser l'activité et pour contourner ces obstacles. D. Scapin établit une typologie des comportements les plus courants que l'utilisateur développe selon le degré d'inadaptation du système qui vont du développement d'activités compensatoires qui modifie la tâche initiale, au rejet du système en passant par des degrés intermédiaires comme par exemple l'utilisation partielle du logiciel ou la conservation de la procédure manuelle en plus de la procédure informatisée [Scapin 86]. En effet, un système-obstacle est condamné à être sous-utilisé ou à être remplacé par un système plus adéquat.

En fait, la mesure du niveau de soutien à l'activité est une manière d'évaluer le succès du système [Denning 94] , [Newman 95] . Si le système permet à l'utilisateur de mieux réaliser son activité, à un degré qui justifie son coût, le système est un succès.

Pour parvenir à cet objectif, il faut répondre à deux questions :

- Comment est-il possible d'identifier les activités dont le soutien présente une utilité réelle pour l'utilisateur ? - cette question est associée à la fonctionnalité du système ;

- Quelle est la signification de l'expression 'réaliser ***mieux*** son activité' ? - cette question est associée à l'**utilisabilité** du système.

Dans la littérature est trouvé de plus en plus évidente la complémentarité de la

2 L'expression "homme-machine" est ici utilisée par abus de langage en rapport aux expressions plus précises "homme-ordinateur" ou "personne-système".

fonctionnalité et de l'utilisabilité du système. Les informaticiens ont souvent une fausse impression : plus il y a de fonctions fournies, plus le système sera accepté par l'utilisateur. Bien au contraire, l'acceptation d'un système peut dépendre plutôt de la qualité de réalisation de quelques fonctions que de la quantité de fonctions disponibles [Nickerson 81] . Cependant, le développement d'un système ne se réduit pas à chercher l'utilisabilité et la fonctionnalité. Par exemple, les systèmes doivent être aussi robustes et faciles à maintenir. En même temps, le développement de systèmes doit respecter des contraintes comme le coût et le temps de développement. Un système qui ne respecte pas ces facteurs n'est pas de bonne qualité, surtout dans le sens courant du Génie Logiciel.

Après délimiter l'ensemble des systèmes interactifs en rapport aux types d'applications et au contexte de développement, nous nous centrons dans cette thèse à des systèmes d'information interactifs 'maison' (SIIM).

1.1Delimitation du type d'application

[Ghezzi 91] a identifié quatre grandes types d'application des systèmes informatiques, sans pourtant prétendre en faire une liste exhaustive :

- Systèmes basées sur des informations ou simplement des systèmes d'information (*information systems*) ;
- systèmes temps-réel (*real-time systems*) ;
- systèmes distribués (*distributed systems*) et
- systèmes embarqués (*embedded systems*)

Nous nous sommes intéressés dans cette thèse au type systèmes d'information, appelés ainsi parce que le but premier de ces systèmes est de gérer l'information, c'est à dire, créer, enregistrer, récupérer, actualiser ou exporter des informations. Les systèmes d'information sont de plus en plus importants du fait de la valeur croissante de l'information comme ressource des entreprises. Les systèmes d'information sont des systèmes typiquement orientés données et peuvent être caractérisés par l'intégrité des données manipulées et par la fonctionnalité qu'ils offrent. L'intégrité inclut la correction des informations enregistrées (intégrité logique), la prévention en rapport à un mauvais fonctionnement du système ou du matériel (intégrité physique) et la sécurité contre des accès non-autorisés. La fonctionnalité représente la quantité et la variété des fonctions que l'on peut réaliser à travers le système. [Cox 93] classifie trois principaux types de systèmes d'information en rapport à la fonctionnalité:

- SI pour le traitement des transactions ('*transaction processing*'), comme l'exemple ATM présenté dans cette thése (voir les chapitres IV, V et VI);
- SI pour le soutien à la décision ('*decision-support* '), comme l'étude de cas SOHO présenté dans cette thèse (voir chapitre VIII);
- SI pour la récupération d'informations ('*information retrieval*') non considéré dans cette thèse.

1.2Delimitation du type de construction : ‘systèmes maison’

On peut classifier les systèmes d’information interactifs en trois types basiques, selon le contexte de construction du système [Grudin 91] :

- 1) **les systèmes définis par contrat**, où il y a un client et plusieurs fournisseurs possibles du système (le cas de systèmes développés pour l’état - connus aussi comme ‘appel d’offre’);

- 2) **les systèmes 'package'**, où il y a un fournisseur qui développe un produit utilisable par plusieurs clients (le cas de la majorité des logiciels commerciaux);

- 3) **les systèmes ‘maison’** (ou *'in-house*'), où le client et le fournisseur font partie de la même entreprise.

Nous allons nous concentrer sur ce dernier type , à savoir, les ‘systèmes maison’, parce que :

- dans ce type de systèmes les informations sur le contexte et les intervenants qui sont concernés par ces informations sont clairement identifiés; ces aspects sont particulièrement mis en valeurs dans notre approche ;

- il permet la reutilisabilité du développement dans un domaine plus que les autres types de systèmes.

Les caractéristiques principales de ce type de système sont:

1) l'utilisateur dans un poste de travail est membre d'un groupe et travaille dans une organisation et

2) les tâches d'un poste de travail font partie des activités et des processus organisationnels.

Pour les systèmes ‘maison’, une caractéristique importante est que la plupart des systèmes développés dans une entreprise soutiennent les activités dans un même domaine du problème. Cela permet la reusabilité du développement entre eux, ou tout au moins, la reutilisabilité d’une grande partie de l’information representant le contexte.

1.3La structure, le comportement et le développement d’un système interactif

Cette section est dédiée à la structure, au comportement et au développement des systèmes interactifs. D’abord les modèles et principes architecturaux principaux pour la structure de ces systèmes sont résumés. Ensuite, un modèle de comportement des systèmes interactifs est décrit. Finalement, quelques caractéristiques générales du processus de développement et les principales approches de conception sont présentées.

1.3.1La structure d’un système interactif

Il y a plusieurs manières de décomposer un système interactif et d’établir des liaisons entre ces composants. Ces manières s’expriment sous la forme de modèles architecturaux qui respectent des principes, présentés respectivement dans les sections suivantes.

1.3.1.1Modèles architecturaux de haut niveau

Comme [Carneiro et al. 93] , nous faisons la distinction entre modèles architecturaux de haut niveau (*high order architecture models*), dont la sémantique des composants est indépendante des détails de l'implémentation ; et les modèles architecturaux d'implémentation, dont la dépendance à ces détails est très forte. Plusieurs modèles architecturaux de haut niveau ont été proposés, par exemple Seeheim [Green 86] , PAC [Coutaz 90] , ADV [Carneiro et al. 93] et le modèle Client-Serveur adopté par MICO [Palanque et al. 96] . La figure 1.1 montre ces modèles comparativement à une architecture monolithique. La figure 1.2 illustre l'architecture du modèle de Seeheim.

Figure 1.1 - Modèles Architecturaux de haut niveau (adapté de [Carneiro et al. 93])

1.3.1.2Principes structuraux de systèmes interactifs

Même si l'on a un ensemble de modèles distincts, il y a à l'heure actuelle un consensus sur un ensemble de lignes directrices concernant la structure d'un système interactif. Les deux principes structuraux fondamentaux présentés ci-dessous correspondent à ces caractéristiques consensuelles et suivent l'application du principe de la modularité du GL [Ghezzi 91] à l'architecture d'un système interactif.

1.3.1.3Principe 1 : indépendance de dialogue

Traditionnellement, un système interactif est formé de deux composants permettant d'assurer la communication entre l'utilisateur et le système : le composant sémantique (aussi appelé noyau fonctionnel) et l'interface utilisateur.

Le **composant sémantique** contient les concepts du système et les fonctions que ce système est capable d'effectuer sur eux. Concepts et opérations constituent la sémantique de l'application et sont donc indépendants du médium d'interaction.

Le terme Interface homme-machine est défini comme le moyen - y compris ses parties matériel (dispositifs périphériques d'entrée/sortie) et logiciel - pour l'interaction

homme-machine. Dans cette thèse, le terme **interface utilisateur** fait référence à la partie logiciel.

La distinction fonctionnelle entre le composant sémantique et l'interface utilisateur est conventionnellement appelée **indépendance de dialogue** et, en pratique, permet que les décisions de conception pour le composant de dialogue n'affectent pas la structure du composant sémantique [Hartson & Hix 89] .

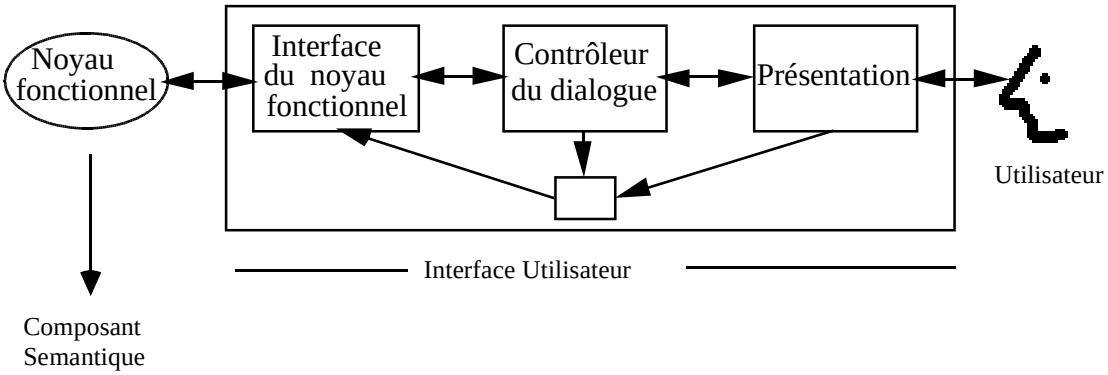


Figure 1.2 : Modèle de Seeheim

1.3.1.4Principe 2 : les composants de l'interface

Le second principe admis est que le composant du dialogue comprend au moins deux parties qui permettent l'interaction entre le noyau fonctionnel et l'utilisateur, à savoir :

La **présentation** qui effectue la présentation physique de l'interface en termes d'objets concrets (Eléments graphiques, sons, etc.), ainsi que de la réception des interactions de l'utilisateur sur les périphériques d'entrée (clavier, souris, etc), en utilisant en général, un serveur graphique comme X Window, NeWS ou MS-Windows.

Le **dialogue** qui est un médiateur entre le noyau fonctionnel et l'utilisateur, plus précisément entre l'interface du noyau fonctionnel et la présentation. Il analyse et établie une éventuelle séquence des échanges entre ces deux modules, en faisant :

- la transformation des informations (commandes et données) à partir de l'interprétation des interactions de l'utilisateur et l'envoi au noyau fonctionnel; et inversement

- la maintenance de la cohérence entre les fonctions et les données du noyau fonctionnel et les commandes et les informations présentées à l'utilisateur.

1.3.2Le comportement d'un système interactif

Don Norman a proposé un modèle qui organise le comportement d'un système interactif comme une boucle de contrôle (*control loop*), conformément à la figure 1.3 [Norman 86]. Cette boucle permet de comprendre comment les actions d'un utilisateur sont en relation avec ses buts et avec le système qu'il utilise, bien qu'en pratique les interactions ne sont pas toujours ainsi : parfois il y a des étapes qui ne sont pas faites ou qui sont répétées.

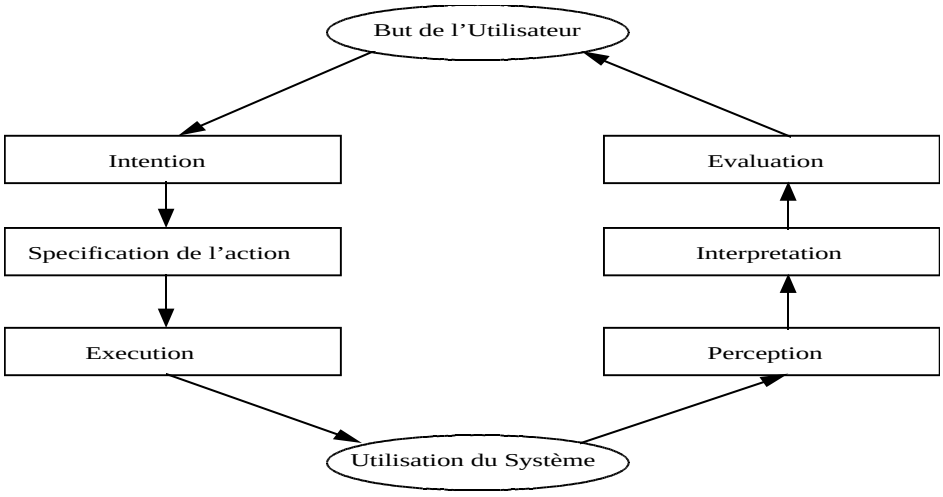


Figure 1.3 - Modèle de l'interaction, d'après /Norman 86/

Dans ce modèle d'interaction, l'utilisateur et le système sont des *agents*, qui réalisent des tâches pour arriver à des buts. L'interaction n'est pas une partie du système mais une **relation entre les utilisateurs et le système**. En fait, un système interactif est conçu pour soutenir le travail de l'utilisateur. Ce dernier est généralement exprimé en termes de buts. Un but peut être décrit comme un prédicat sur un état, qui peut être atteint par l'exécution de tâches. L'exécution des tâches interactives implique la coordination non seulement des actions de l'utilisateur et des procédures du système mais aussi des interactions entre les deux agents pour arriver au but désiré. Les stimulus qui émergent quand ces agents communiquent par l'intermédiaire des actions sont appelées *événements*. Un événement arrive quand les deux agents réalisent des actions qui provoquent un échange d'informations. Si l'un des deux n'est pas engagé, l'événement n'arrive pas. Par exemple, si le système actualise un écran et l'utilisateur l'observe et interprète cette actualisation, un événement est arrivé. Si l'utilisateur ne s'en est pas aperçu, il n'y a pas d'événement.

Du point de vue d'un agent, soit l'utilisateur soit le système, une interaction est une séquence d'événements **expressifs** (événements déclenchés par lui-même pour stimuler l'autre agent) et événements **réceptifs** (déclenchés par le stimulus de l'autre agent).

Du point de vue de l'utilisateur, une interaction est une entrelacement de *formulations* et d'*évaluations*. Une **formulation** intègre : établir une intention, spécifier une action correspondante et de la réaliser avec le système; tout cela pour arriver (même partiellement) au but. Une évaluation intègre la perception de l'état résultant du système, une interprétation de cet état perçu et l'évaluation par rapport au but et à l'intention.

Du point de vue du système, l'interaction est l'entrelacement de *fonctions réceptives* et *expressives***.** Les **fonctions réceptives** acceptent les formulations et les exécutent; les **fonctions expressives** reflètent l'état du système résultant des actions.

Comme [Blanford et al. 93], nous pensons aux interactions en termes d'événements. Plus spécifiquement, une interaction est une séquence d'événements expressifs et réceptifs. Du point de vue du système (voir aussi la Théorie Général des Systèmes), le système reçoit un stimulus de l'utilisateur (formulation) et réagi en déclenchant des services/opérations internes. Ces opérations changent l'état interne du système. La partie perceptible de cet état (aussi appelé *rendering* par [Blandford et al. 93]) doit

refléter son changement, usuellement par un changement des éléments de l'écran mais parfois par l'intermédiaire d'autres canaux de communication comme le sons ou le mouvement. Ce changement du *rendering* permet à l'utilisateur de percevoir, d'interpréter et d'évaluer les résultats de sa formulation au système (évaluation). La figure 1.4 illustre cette description.

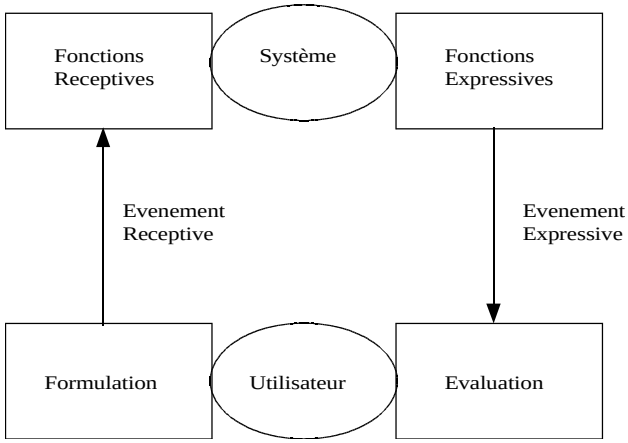


Figure 1.4 - L'interaction comme un entrelacement d'événements réceptifs et expressifs.

Selon [Blandford et al. 93] la formulation d'une tâche interactive en termes événements est appelée **performance**. En pratique, une performance est un ensemble ordonné d'événements (un *event trace* [Hoare 85]), typiquement les événements réceptifs intercalés par les événements expressifs. Il y a plusieurs performances possibles pour chaque tâche interactive et c'est un des buts du développement des systèmes interactifs, en particulière de ces premières activités comme l'ingénierie des Besoins, que de déterminer quelles performances supportera le système, le pourquoi de cette choix et comment seront-elles réalisées en termes de formulations des utilisateurs et de fonctions du système.

1.3.3Le développement d'un système interactif

Il n'y a pas une méthode ou un cycle exact pour la construction de systèmes interactifs mais les chercheurs ont défini leurs propositions [Gram & Cockton 95] , [Hartson 84] , [Barthet 88] , [Newman 95] , [Macaulay 95] , [Lim & Long 94] . On ne veut pas consolider ou intégrer ces différentes vues ici, parce que ce n'est pas le but du travail. On veut seulement mettre en valeur un ensemble de concepts et d'activités basiques qui représentent l'axe de la conception de systèmes interactifs. En vérité, il est pas prouvé qu'une méthode qui s'applique bien à un problème va s'appliquer pour un autre. Cependant, il est reconnu que dans le processus de conception de systèmes interactifs il y a constamment un changement entre activités d'analyse et de synthèse [Newman 95] , [Curtis et al. 88] , [Macaulay 95] .

1.3.3.1L'Analyse et la Synthèse

L'Analyse comprend l'étude et l'évaluation des utilisateurs actuels et futurs, de leur travail dans une organisation. La Synthèse comprend la spécification, conception et réalisation d'un système, y compris son composant sémantique et son interface utilisateur.

Analyse et Synthèse forment une boucle : l'analyse aide à comprendre les activités

actuelles et les aspects à changer ; la synthèse aide à concevoir et à structurer les alternatives de changement ; l'analyse aide à évaluer les alternatives ou leurs résultats ; l'évaluation indique où la synthèse doit être modifiée. Enfin, cette boucle peut être répétée plusieurs fois pendant la construction. On peut alors effectuer l'Analyse **avant** la Synthèse pour recueillir des informations sur l'existant ou **après** la synthèse pour évaluer les améliorations des nouvelles tâches interactives avec le système synthétisé par rapport à des tâches anciennes. Ces deux activités sont réalisées de différentes manières par plusieurs approches et chaque approche utilise un sous-ensemble déterminé des connaissances - explicite ou implicitement - de façon à construire des systèmes utiles et utilisables.

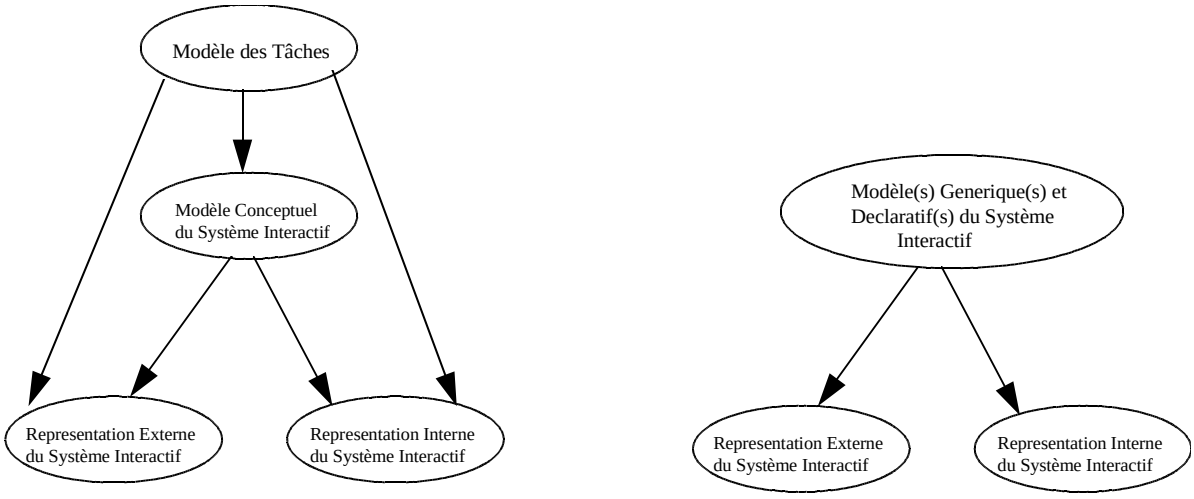
Aujourd'hui l'explicitation des connaissances s'est faite de plus en plus par l'intermédiaire de **modèles** - représentations abstraites de ces connaissances. En fait, les modèles permettent la structuration des informations en différents niveaux d'abstraction - pour aider à la gestion de la complexité et augmenter la communication entre les personnes impliquées - et le raisonnement sur leurs caractéristiques - pour aider à la prise de décision pendant la construction.

1.3.3.2Approches de développement des systèmes interactifs

Il existe deux principales approches visant à l'utilisation de modèles pour la construction de systèmes interactifs : **l'approche basée sur les modèles** (Model-Based approach) et **l'approche basée sur les tâches** (Task Based approach), voir figure 1.5.

Les deux approches ont des aspects en commun : premièrement, l'utilisation de modèles pour représenter les diverses types d'information utilisés dans la construction [Wilson & P. Johnson 96] . Différents types de modèles sont cités, par exemple, par [Puerta 96] , [Wilson 96] , [Schlungbaum 96] [Szekely 96] . Deuxièmement, les deux approches proposent l'utilisation de ces modèles pour les activités de construction, souvent en utilisant un modèle pour en créer/dériver un autre.

La distinction entre elles est surtout due à l'usage de ces modèles. Les approches basées sur les modèles (comme UIDE [Foley et al. 91] , HUMANOID [Szekely et al. 92] et MECANO [Puerta 96]) , dans le but de fournir de l'aide à la construction de l'interface utilisateur, proposent un ensemble de transformations sur les modèles à partir de descriptions de haut niveau (la plupart déclaratives) du système. Ces descriptions, même abstraites, sont déjà des modèles de solution et non un soutien à la création de solutions. En général les transformations sont guidées par des règles de style et/ou des heuristiques de construction et consistent basiquement en transitions des descriptions abstraites vers des descriptions concrètes, voir exécutables, du système (ou de son interface utilisateur). Ainsi, usuellement ces approches sont intensément supportées par des outils, voire outils de génération automatique de code ou d'interprétation des modèles lors de l'exécution [Szekely 96] .



Development Basé sur Tâches
(Task-Based Development)

Development Basé sur Modèles
(Model-Based Development)

Figure 1.5 - Approches Basées sur les Tâches et basées sur les modèles (adapté de [Vanderdonckt 96]

Les approches basées sur les tâches (comme ADEPT [P. Johnson & H. Johnson 93] , TRIDENT [Bodart et al. 94] , MUSE [Lim & Long 94] et MASTERMIND [Szekely et al. 96]), par contre, sont surtout orientés vers la création de systèmes à partir des tâches des utilisateurs, dans le but de construire des systèmes permettant de soutenir leurs activités de travail. Avec l'aide des heuristiques de construction, le modèle de tâches est utilisé principalement pour conduire le processus de modélisation du système interactif. Les autres modèles ne forment qu'un cadre contextuel pour cette modélisation [Vanderdonckt 96] . Ce processus n'est pas algorithmique comme les transformations de l'approche précédente et donc il n'est pas si bien supporté par des outils, qui se résument typiquement à des éditeurs pour modéliser les tâches ou des vérificateurs de propriétés de modèles. En fait, il n'y a pas beaucoup de travaux sur la formulation de principes pour le processus de conception basés sur les tâches. L'expérience collective accumulée par la communauté d'IHM dans les dernières années sur ce processus est exprimé par une séquence de choix et de décisions basés sur un ensemble (parfois structuré) de 'DOs' et 'DON'Ts', dérivé de l'examen des tentatives qui se sont bien ou mal succédées pendant la construction des systèmes. Cet ensemble (typiquement présenté sous la forme de principes, 'guidelines' ou recommandations) est très générique, il n'y a pas de méthodes associées et il n'est pas toujours utilisable directement par les concepteurs [Farenc et al. 95] . Cependant, il y a déjà quelques initiatives importantes comme la construction orientée scénarios [Rosson 95] et l'ensemble extensif de règles de guidage proposé par [Wilson & P. Johnson 96] .

1.4Perspective d'Outil

Les rôles joués respectivement par l'utilisateur et le système influencent à la fois la structuration d'un système interactif et le processus de son développement. [Kammersgaard 88] a proposé quatre perspectives différentes pour cela :

1) la perspective de système, où les utilisateurs sont considérés comme composants "data-entry" des systèmes;

2) la perspective de “partenaire” de dialogue, où les utilisateurs et les systèmes sont envisagés comme des parties équivalentes dans la conversation, les deux sont capables d’agir comme émetteur et récepteur dans le dialogue, c’est à dire, l’ordinateur est considéré comme un partenaire humain dans le dialogue;

3) la perspective d’outil, où les systèmes sont regardés comme des outils (instruments) manipulés par les utilisateurs pour réaliser une tâche déterminée;

4) la perspective de “media”, où les systèmes sont considérés comme moyen de communication (échange de messages) entre des êtres humains.

Traditionnellement, la **perspective de système** est la plus connue et utilisée pour analyser l’IHM. En fait, elle est intéressante pour aider à comprendre le rapport structurel entre les différents composants d’un système informatique mais elle est extrêmement faible pour aider à comprendre l’utilisation de ces systèmes. La **perspective de partenaire** est bien exploitée surtout dans le domaine de l’intelligence Artificielle pour les système orientés langage naturelle [Sabah 88]. Ces deux perspectives s’occupent plus des aspects de l’interaction pour elle même, de manière indépendante de sa signification, et considèrent les humains et les ordinateurs comparables : dans la première (perspective de système), les humains sont vus comme identiques aux autres composants (automatiques) du système; dans la deuxième (perspective de “partenaire”), les ordinateurs sont vus comme capables d’avoir un comportement (au moins au niveau de la communication) humain

Les **perspectives d’outil** et “**media**” permettent de se concentrer sur le sens du processus d’interaction (la sémantique respectivement du processus de travail avec un outil et du processus de communication avec un “media”) comme les sources primaires des principes de conception, en respectant les différences entre des humains et des ordinateurs. En plus, elles ont un aspect en commun : dans les deux cas, il y a quelqu’un qui développe un système (comme un outil ou comme un message exécutable) pour qu’un autre l’utilise/le décode selon sa nécessité. Alors, il a dû y avoir une préoccupation et une compréhension du contexte d’utilisation pour cette personne de façon à faire un meilleur outil ou une meilleur codification du message. C’est pour cela que nous nous concentrons sur les perspectives d’outil et de media et une description succincte de ces perspectives d’outil et de media sera ensuite présentée.

Perspective d’Outil

Dans la perspective d’outil, le système (qui est l’intégration des composants A, B et C dans la figure 1.6) sous contrôle de l’utilisateur, est utilisé pour la réalisation d’une tâche. L’utilisateur n’est pas intéressé par la distinction entre l’interface et l’application. Son attention principale est l’exécution de la tâche et pas l’utilisation du système, de la même façon qu’il utilise un marteau pour enfoncer un clou. Alors, le point focal de conscience est l’interface entre le système et la tâche (C) et non l’interface entre l’utilisateur et le système (A). Cela veut dire que, pour la perspective d’outil, un bon système a l’interface A invisible et l’interface C comme un vrai point d’attention.

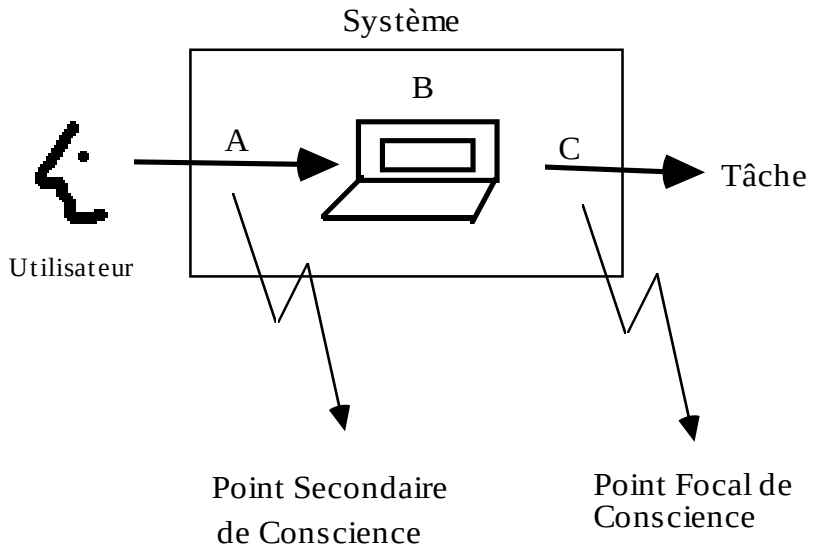


Fig. 1.6 - Perspective Outil de systèmes interactifs

Un outil est à la fois une ressource et une contrainte à une activité parce que les outils changent la manière d'interagir avec le contexte. En même temps, un outil est toujours influencé par le contexte social et culturel de son utilisation. Donc, dans la perspective d'outil, les systèmes interactifs ne peuvent pas être conçus sans considérer l'ensemble des activités à soutenir, les buts précis à atteindre et le contexte dans lequel son utilisation sera réalisée. Pour respecter les mêmes principes, l'approche TAREFA se situe dans cette perspective.

Perspective de media

La perspective de "media" envisage l'ordinateur comme un moyen de communication au travers duquel les personnes communiquent entre elles; quelque chose de comparable aux journaux, livres, films, télévision, téléphone, vidéo, etc. Une compréhension intuitive du processus de communication inclut la notion de messages envoyées par un (plusieurs) émetteur(s) pour un(plusieurs) récepteur(s). Les messages sont l'expression d'un signifié, codés par l'émetteur, et qui peuvent être décodés et interprétés par le récepteur. Avec la perspective "media" deux types de communication sont intéressants :

1) Toute la communication entre (groupes de) les utilisateurs qui est faite à travers une application informatique. Les exemples typiques sont les systèmes de courrier électronique ("*mail systems*"), les systèmes de conférence et la majorité des applications de "Computer Mediated Communication", qui sont analysés avec plus de détails en [Bannon 85] et [Salber 95] ;

2) La communication différée unilatérale du concepteur à l'utilisateur qu'il y a quand une application conçue par le premier est utilisée par le deuxième. Cette communication est unilatérale parce que l'utilisateur ne peut pas envoyer une réponse au concepteur (sauf peut être en situation d'évaluation ou de maintenance) et elle devient différée de façon identique à d'autres types de communication comme celle entre auteur-lecteur et entre compositeur-auditeur, où les deux agents (émetteur et récepteur) ne partagent pas la même dimension

espace-temps;

Ce deuxième type de communication perçois le système comme un message très particulier. D'abord, les systèmes informatiques sont des messages dynamiques (par opposition les livres et les journaux sont des messages statiques) et exécutables. En plus, l'exécution des systèmes est elle-même un acte communicatif, dans lequel le message joue le rôle de l'émetteur et du récepteur d'autres messages. Alors, ces systèmes peuvent être considérés comme des **artefacts de métacommunication** [Souza 1993] .

2.Concepts pertinents pour l'ingénierie des besoins

Cette section regroupe certains concepts spécialement pertinents pour l'ingénierie des besoins.

2.1Les acteurs d'un système interactif

Les rôles humains traditionnellement associés au logiciel interactif sont :

- l'utilisateur ;
- l'analyste , qui spécifie le système
- le developpeur, qui construit le système spécifié par le concepteur.

Les approches plus récentes reconnaissent qu'il faut aussi la participation d'autres personnes qui sont touchés par le système parfois même sans l'utiliser, y compris les représentants des équipes de marketing, formation (*training*), installation et maintenance. Le terme '**acteur**'[3] est utilisé en IB pour designer toutes les personnes impliqués par le développement ou l'utilisation du système. [Newmann 95] divise les acteurs en deux groupes principaux appelés de façon générique **l'utilisateur** et **l'analyste**, qui représentent respectivement le point de vue collectif de ceux qui utilisent et de ceux qui conçoivent et développent. Dans cette thèse nous allons adopter souvent ces noms génériques pour faire référence à ces groupes.

L'analyste et le **developpeur** font partie de l'équipe de développement du logiciel. Dans une thèse d'Informatique, il est inutile de décrire leur complexité, mais il est important de remarquer que ces rôles peuvent être joués par la même personne. En particulier, le nombre d'analystes peut varier selon le type d'univers étudié, selon l'ampleur du projet, et les ressources humaines disponibles pour cette activité. Lorsqu'il est assez grand, un tel groupe peut se diviser en sous-équipes de composition et de taille variées, mais il peut éventuellement se réduire à un analyste.

Connaître **l'utilisateur** est une condition requise et considérée comme cruciale pour la spécification d'IHM. La connaissance de l'utilisateur est associé, en général, à l'adoption d'un modèle mental d'utilisateur interne au système. [Carrol 91] a défini un modèle mental comme "les structures et processus que l'on impute à une personne afin de justifier le comportement et la pratique de cette personne" et la représentation de ce modèle est un sujet de travail de la Psychologie Cognitive.

En général, **l'utilisateur** d'un système n'existe pas; il y a **plusieurs** types d'utilisateur avec des besoins, des talents, des buts, des connaissances et des préferences

3 Notre traduction de l'expression anglais '**stakeholder**' [Macaulay 95].

qui changent avec l'expérience. Pourtant, une simple interface ne peut pas être adaptée en même temps ni à tous ses utilisateurs possibles, ni au même utilisateur pendant longtemps.

Pour représenter cette variabilité, les 'modèles de l'utilisateur' utilisés en pratique sont des descriptions de profils stéréotypés d'utilisateur selon quelques critères. Il y a plusieurs critères décrits dans la littérature:

1) **fréquence d'usage** : fréquent, occasionnel ;

2) **expérience** : grand public, novice, débutant avancé, compétent, efficace, expert [Chin 88] ;

3) **catégorie d'usage** :

   - utilisateur final (*end user*) , qui veut simplement utiliser l'application pour résoudre ses problèmes et atteindre ses objectifs. Le logiciel doit être 'transparent' dans son contexte d'usage et lui permettre de travailler directement sur le domaine du problème et de réaliser ses tâches.

   - le client (*customer*) qui a commandé le système ;

Un profil d'utilisateur est typiquement la combinaison de ces critères (par exemple utilisateur novice et occasionnel, expert et fréquent, etc). Un utilisateur peut évidemment changer de profil au cours du temps.

Sauf quand c'est mentionné explicitement, le terme 'utilisateur' est employé dans cette thèse dans le sens d'utilisateur final.

2.2L'activité

Comme nous l'avons vu à la section 1 ci-dessus, un système interactif est fondamentalement un soutien aux activités humaines. La Théorie de l'Activité [Nardi 1996] fournit un cadre conceptuel pour comprendre comment les activités peuvent être comprises dans un contexte de l'interaction entre les être humains et leur environnement. Selon la Théorie de l'activité, les **activités** sont les unités de base de notre vie et leur réalisation est toujours orientée buts et médiatisée par l'utilisation des outils. Une activité est constitué d'**actions** (combinées ou simples), exécutées consciemment par les personnes. Une action est aussi orientée buts, néanmoins subordonnés aux buts de l'activité. Si une activité peut être réalisée en utilisant différentes actions, en fonction de la situation, une même action peut être associée à différentes activités. Des actions sont aussi perçues comme une combinaison d'**opérations**, qui sont des routines bien définies utilisés par l'individu de manière sous-consciente (mécaniquement) comme réponses à des conditions envisagées pendant la réalisation de l'action. Au passage du niveau d'action au niveau de l'opération, on traverse la frontière entre les processus conscients et automatiques. On a alors une hiérarchie en trois niveaux : **activités** réalisées par l'intermédiaire de quelques **actions** qui respectivement sont réalisées par l'intermédiaire de quelques **opérations** [Kaptelinin 95] .

Les activités sont distinguées en activités internes et externes. La notion traditionnelle de processus mentaux correspond aux activités internes. Les activités externes sont des manipulations des objets réels. Ces deux types d'activités ne peuvent pas être analysées séparément parce qu'une peut être transformée en une autre, suivant

le contexte de l'activité, par l'intermédiaire de internalisations et externalisations.

L'internalisation est la transformation d'activités externes en activités internes et fournit la possibilité pour les êtres humains de simuler les interactions potentielles sans manipulations réelles sur les objets. L'externalisation est la transformation d'activités internes en externes, souvent utilisé pour la réparation d'actions internalisées ou quand la collaboration entre plusieurs agents a besoin d'être coordonnée.

Selon le principe d'externalisation/internalisation, modéliser une activité externe peut résulter en la modélisation d'une activité interne. Cela explique pourquoi l'utilisation d'un outil comme médiateur est fondamentale pour la théorie de l'activité.

Une importante contribution de la théorie de l'activité est l'asymétrie entre les personnes et les choses, due aux notions de **motif** et **conscience** - attributs typiquement humains. Les personnes ne sont pas réduits à '*nodes*' ou agents dans un système; le traitement d'information n'est pas modélisé de façon similaire pour des personnes et des machines. Dans la théorie de l'activité, les artefacts sont des médiateurs entre la pensée et le comportement mais ils n'occupent pas le même espace ontologique. Cette théorie propose donc une vue plus humaine du rapport entre les personnes et les artefacts : ces derniers sont des outils pour soutenir les activités de travail des premiers.

2.3Les informations pour le développement des systèmes interactifs

Le processus de développement de Systèmes Interactifs, qu'il soit assisté ou partiellement automatisé par des outils ou qu'il soit le résultat de l'application manuelle d'une méthode, a besoin d'au moins trois domaines de connaissances: le domaine du travail actuel de l'utilisateur, le domaine des options technologiques - considérées à la fois comme ressources et contraintes - disponibles, et le domaine de travail avec le nouveau système (aussi appelé système futur) [Kensing 93] .

Cette division montre que les connaissances des deux premiers domaines constitue l'"entrées" pour le processus de développement et que sa sortie inclut le nouveau système proposé, cf. Figure 1.7. Cependant, même si cette division est vraie d'un point de vue générique, la connaissance du système futur est en fait nécessaire **avant** qu'il ne soit conçu. Cette apparente contradiction est une des raisons de la complexité du développement, en particulier pour l'étape de l'ingénierie des besoins, d'un système interactif.

Bien qu'il y ait plusieurs façons de représenter les domaines de travail (actuel et futur), la plus utilisée est réalisée par l'intermédiaire de modèles d'activités [CACM 95].

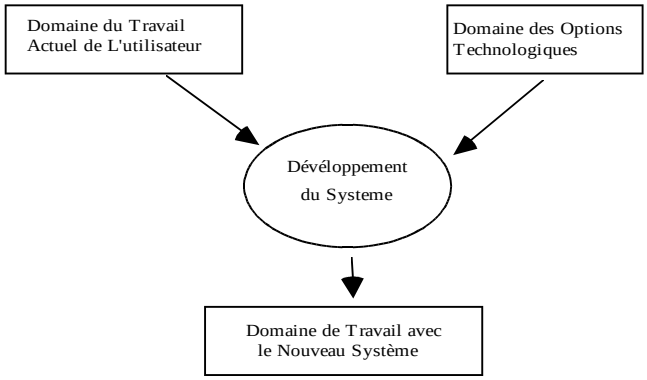


Figure 1.7 - Les connaissances pour la construction des systèmes interactifs

Les connaissances intervenant dans le développement du système peuvent être explicitement représentées en modèles qui contribuent de diverses façons pendant ce développement. Les exemples de modèles sont (voir [Schlungbaum 96] pour plus de détails) :

1) Modèle des tâches : pour décrire les tâches que l'utilisateur veut réaliser;

2) Modèle de l'application : pour spécifier les services fournis par le noyau fonctionnel du système ;

3) Modèle de Dialogue : pour décrire le dialogue utilisateur-système ;

4) Modèle de Présentation : pour spécifier les composants des unités de présentation (*displays*) en composants et l'apparence de chaque composant ;

5) Modèle de Comportement : pour spécifier le comportement des composants de la présentation ; en général le modèle de présentation et de comportement sont encapsulés et pre-definis pour un ensemble d'objets interactifs type WIMP ;

6) Modèle de Plateforme : pour décrire les caractéristiques du matériel ou logiciel disponibles pour le développement ;

7) Modèle de l'Utilisateur : pour spécifier les caractéristiques des utilisateurs finaux du système ;

8) Modèle de l'environnement : pour décrire les caractéristiques (culturelles, politiques, physiques, etc) de l'environnement où les tâches de l'utilisateur se déroulent.

Evidemment, ces modèles ne sont pas présents dans toutes les approches. Dans une évaluation présentée à CADUI 96, Schlungbaum identifie les 5 premiers modèles d'entre les 8 modèles cités ci-dessus comme les plus utilisés [Schlungbaum 96] . ADEPT est la seule approche qui utilise un modèle de l'utilisateur explicite. Il est étonnant de constater qu'aucune approche évaluée n'utilise un modèle de plate-forme ou de l'environnement.

Les modèles adoptent des notations formelles et/ou semi-formelles pour la représentation des informations et plusieurs relations peuvent être établies entre eux. [Cockton et al. 96] a identifié 5 classes principales de notations:

1) Notations contextuelles qui représentent les activités humaines et leur environnement;

2) Notations projectives, pour envisager les possibilités potentielles d'utilisation du système ;

3) Notations de '*design rationale*' , qui capturent les raisons associées aux décisions de conception;

4) Notations comportementales , pour représenter la vue externe du Système, c'est à dire, le système du point de vue des utilisateurs ;

5) Notations contructionnelles , pour représenter la vue interne du système, c'est à

dire, le système du point de vue des developeurs.

L'utilité et l'utilisabilité de ces notations peuvent s'évaluer selon certains critères comme Les Principes de Namur, établis par un groupe de travail lors de l'Eurographics DSV-IS'96 [C. Johnson 96] .

Dans les sections suivantes nous abordons plus précisément les informations de tâches et de contexte.

2.3.1Les tâches

Les travaux récents des communautés du GL et d'IHM convergent vers une idée centrale : pour concevoir des systèmes plus fonctionnels et plus utilisables il faut bien connaître les tâches que les personnes réalisent et bien appliquer cette connaissance au processus de conception [Barthet 88, Benyon 92, Bodart et al. 94, Carrol 92, Curtis & Hefley 94, Dowell &Long 89, Moran 81, Diaper 89, P. Johnson 92] . L'analyse des tâches utilisateurs dans leur contexte de travail est appelée Analyse de la Tâche et son résultat est un modèle conceptuel appelé Modèle de Tâche. Les modèles de tâches sont aussi considérés comme un type de modèle du domaine [Benyon 96] .

L'Analyse de la Tâche a émergée de l'Ergonomie comme une aide fondamentale pour les approches de conception des systèmes interactifs parce qu'elle est une méthode empirique pour comprendre comment les personnes réalisent leurs tâches de travail. Cette compréhension peut servir à la fois à plusieurs niveaux comme par exemple aider à délimiter un problème ; aider à recueillir les informations sur un problème de manière systématique ; organiser l'information recueillie ; modéliser les processus d'activités pour identifier les sources de problèmes et générer des hypothèses pour résoudre les problèmes. Cette diversité d'intentions est une raison de la prolifération des méthodes et techniques d'analyse de la tâche. Le dénominateur commun est la description des tâches en termes d'unités d'activité identifiables. Le lecteur intéressé peut trouver des synthèses sur l'analyse de la tâche dans [Shepherd 95] , [Diaper 89] et [Phillips 88] .

Avant de présenter l'analyse de la tâche, il faut considérer la signification du terme 'tâche'. En fait, 'tâche' est un concept ambigu que plusieurs auteurs utilisent avec une signification différente. Il y a plusieurs définitions de tâche (voir par exemple [Draper 93], [Benyon 92], [Storrs 95], [Diaper & Addison 1992], [Phillips 88] , [Bodart et al. 94]).

Nous retenons la définition de [Storrs 95] , principalement parce qu'elle explicite l'importante relation entre actions, buts et contextes mais aussi parce qu'elle a été formulée pour unifier la terminologie du sujet :

> 'Une tâche est un but et les ensembles ordonnés des actions qui peuvent le satisfaire dans les contextes appropriés.'[4]

Ces définitions de tâche sont à l'origine des différentes méthodes pour l'analyse de la tâche et des différents modèles de tâche. En fait, il y a plusieurs modèles de tâches dans la littérature, chacun faisant attention à un sous-ensemble des aspects associés à la manière dont les personnes font leurs tâches. Par exemple, MAD [Scapin & Pierret-Golbreich 89] , TKS [P. Johnson et al. 88] et ATOM [Walsh 89] font attention respectivement à la décomposition des tâches, à la connaissance nécessaire pour réaliser la tâche et aux relations acteurs-objet-actions associées aux tâches.

Parmi les nombreuses classifications, nous avons retenu la taxonomie des

4 Traduction livre de l'auteur de la définition en anglais : '*A task is a goal together with the ordered sets of tasks and actions that would satisfaire it in the appropriate contexts.*'

formalismes destinés à la spécification des systèmes interactifs, proposée par P. Brun et M. Beaudoin-Lafon [Brun 95] , et qui s'appuie sur l'origine des formalismes. Les auteurs ont fait aussi une évaluation très subjective des formalismes en prenant en compte douze critères, dont un est la description des actions et tâches utilisateur. Bien qu'elle soit assez étendue il y a des formalismes qui ne sont pas pris en compte, comme celui proposé dans Mastermind - non présent probablement parce qu'il est très récent ; comme TKS [H. Johnson & P. Johnson 91] ; et encore comme les Blocs de Connaissances proposés par G. Boy (voir [Boy 95]), rebaptisés récemment Blocs d'Interaction, qui si situent entre les modèles de performance et les modèles de dialogue.

Selon la classification de [Brun 95] - voir figure 1.8 - les modèles pour l'analyse de la tâche sont MAD et CLG dont l'origine sont les Sciences Cognitives. Les autres formalismes souvent utilisés pour la modélisation des tâches ont la même origine mais ils sont classés soit comme **modèles de performance** (y compris GOMS et Keystroke) soit comme **modèles de dialogue** (y compris UAN et XUAN) .

Les **modèles de performance** concernent l'évaluation de la performance de l'utilisateur, en incorporant des techniques pour prédire les temps d'exécution des tâches et les taux d'erreurs. GOMS [Card 83] et TAG [Payne & Green 86] sont des exemples de techniques evaluatives qui modélisent respectivement la performance et la compétence de l'utilisateur.

Les **modèles de dialogue** sont une manière de représenter le dialogue entre l'utilisateur et le système à un bas niveau d'abstraction. UAN (User Action Notation) est un modèle semi-formel développé originellement par Hartson, Siochi et Hix [Hartson et al 90] . UAN permet de décrire les action de l'utilisateur et le comportement (*feedback*) de l'interface homme-machine, en remplaçant avantageusement les descriptions informelles sous forme de texte et copies d'écran qui constituaient la plupart des spécifications externes des interfaces. UAN structure la description en un ensemble de tableaux. Chaque tableau représente une tâche qu'il est possible d'effectuer avec le système. Chaque tableau se compose de trois colonnes : les actions de l'utilisateur (typiquement physiques comme enfoncer le bouton de la souris), la réponse de l'interface et l'état de l'interface. La notation propose un ensemble de symboles représentant des actions utilisateur ou des réponses de l'interface, comme par exemple **Mv** pour l'action d'enfoncer le bouton de la souris.

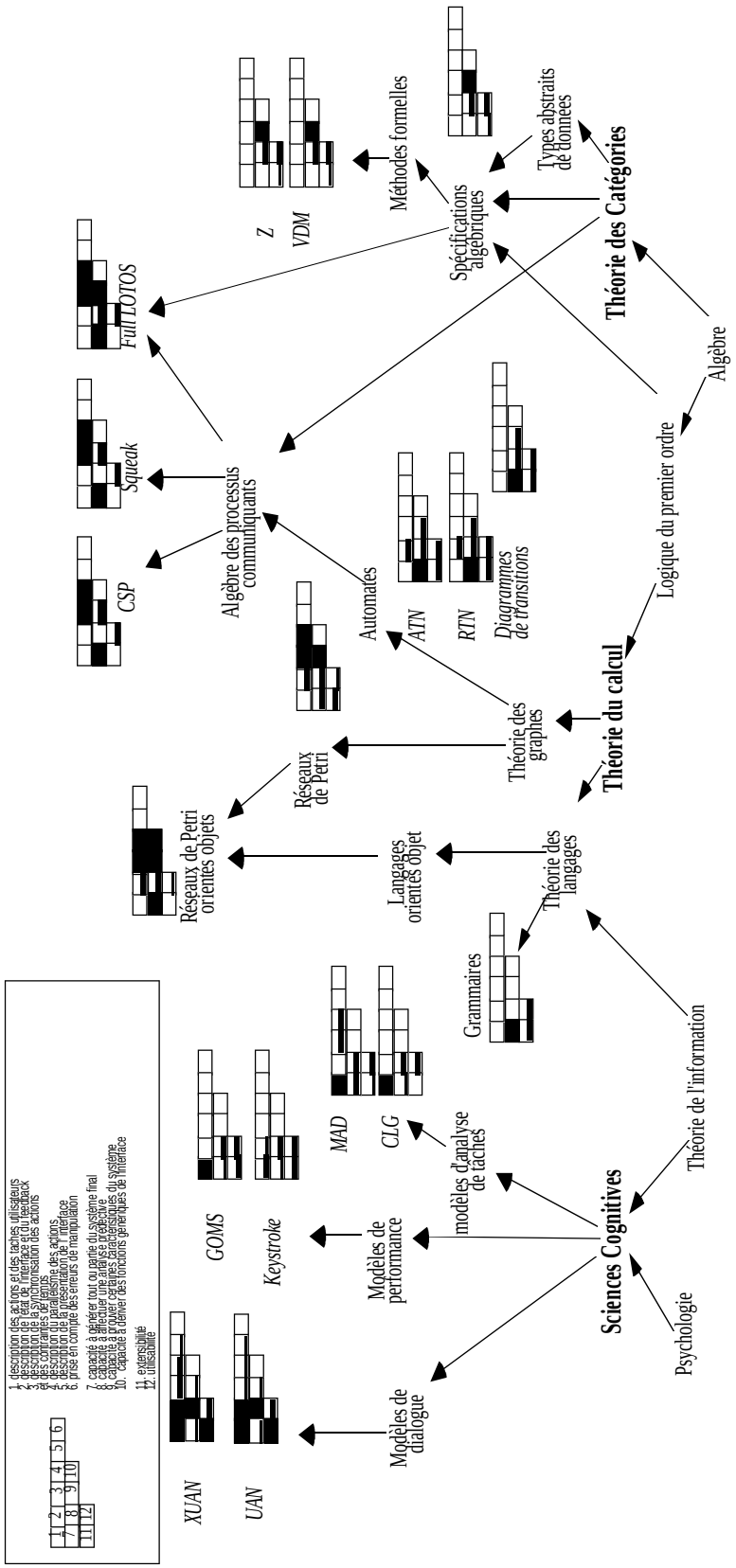


Figure 1.8 - Classification et évaluation des formalismes pour systèmes interactifs [Brun et al. 95] . Le remplissage du carré est proportionnel à couverture du critère evalué. Traduction et schéma des auteurs, gentilment fournis par F. Jambon.

UAN offre également un ensemble d'opérateurs permettant d'exprimer les relations d'ordonnancement entre tâches : séquence, parallélisme, interruptions, entrelacement, et autres cf. Figure 1.9. UAN a été étendue en XUAN afin de tenir compte des contraintes temporelles [Gray et al. 94] .

Les **modèles de tâches** sont basés sur les concepts naturels du travail qui peuvent être trouvés dans le domaine du problème, et ainsi les utilisateurs peuvent facilement les comprendre, les valider et participer activement au processus de l'analyse. Donc, ils sont une manière adéquate de considérer le **point de vue de l'utilisateur** pour la conception. Certains travaux ne font pas la différence entre 'tâches existantes' et 'tâches envisagées', c'est à dire, entre des tâches que l'utilisateur réalise aujourd'hui et des tâches potentielles futures réalisées en interaction avec le (nouveau) système. Clairement, les premières font partie de l'espace problème et les deuxièmes font partie de l'espace solution du processus de développement. Dans ce travail, le terme 'tâches de l'utilisateur' signifie '**tâches existantes de l'utilisateur**'. Ensuite, nous présentons une description succincte des modèles de tâches MAD et CLG.

MAD

MAD[5] (une Méthode Analytique de Description de tâches) [Scapin & Golbreich 89] vise la description de tâches humaines dans un but de meilleure prise en compte de l'ergonomie dans la conception d'interfaces utilisateurs. MAD s'appuie sur le concept de tâche et sur une décomposition hiérarchique des tâches à l'aide de constructeurs, qui décrivent l'agencement des différentes tâches impliquées.

Le concept de tâche est représenté par un objet générique appelé object-tâche et composé d'un état initial, d'un état final, des pré-conditions et des post-conditions.
Les constructeurs définis sont :

- SEQ(séquence), qui exprime l'enchaînement séquentiel de plusieurs tâches;
- ALT (alternative), qui traduit la possibilité de choix entre plusieurs tâches;
- PAR (parallélisme), qui désigne l'exécution entrelacée de plusieurs tâches par un même agent;
- SIM (simultanéité), qui désigne l'exécution parallèle de plusieurs tâches par des agents distincts
- @ (boucle), qui désigne l'exécution itérative d'une tâche.

CLG

CLG (Command Language Grammar), proposé par [Moran 81] , est un modèle de l'interaction homme-machine organisé en quatre niveaux d'abstraction : tâche, sémantique, syntaxique, interaction. A chaque niveau, CLG introduit une symétrie objets/actions : il faut décrire les entités concernées par les tâches et les actions que l'on va exercer dessus. CLG adopte un style de conception descendante (top-down) : chaque niveau raffine le niveau précèdent et les liens entre les niveaux sont explicites mais le passage d'un niveau à un autre n'est pas évident.

Le **niveau des tâches** caractérise les actions à accomplir de manière indépendante du

5 Nous avons choisi de décrire la version de MAD basée sur le document [Scapin & Golbreich 89]. Cependant, récemment MAD a été étendue à nouveau, sous le nom de MAD*, par [Hammouche 95].

système de support. Dans une planification hiérarchique, ce niveau correspond aux tâches plus abstraites. Entre les caractéristiques il y a le traitement des échecs, la fréquence des tâches, leur importance et aussi la définition des paramètres à chacune des tâches.

Le **niveau sémantique** décrit la fonctionnalité du système, par l'intermédiaire des entités sémantiques et des opérations (qui peuvent être regroupées pour former des procédures).

Le **niveau syntaxique** décrit la façon dont la fonctionnalité peut être mise en oeuvre à l'aide des **commandes**.

Le **niveau interaction** décrit la syntaxe des commandes, c'est à dire, l'ordre du passage des arguments et les interactions physiques pour les communiquer. Ce niveau rend compte aussi des attentes de la machine, des validations utilisateurs et des traitements automatiques même si l'interaction ne concerne qu'une partie de la commande.

Pour décrire une IHM, CLG fait l'enchaînement des activités réalisées par l'homme et les traitements automatiques de la machine pour aboutir au but souhaité.

CLG peut être considéré selon au moins deux points de vue :

- le point de vue du concepteur de l'IHM, qui peut organiser la conception de façon systématique avec les décisions adaptés à chacun des niveaux ;

- le point de vue de l'utilisateur de l'IHM, puisque chacun des quatre niveaux présente le système vu par l'utilisateur ;

Chaque niveau est décrit par une grammaire, qui est le formalisme nécessaire pour faire une description CLG.

Relations Temporelles	**Modèle TKS** **[H. Johnson 91, Wilson 94]**	**Modèle MAD** **[Scapin 89]**	**Modèle UAN** **[Gray, Hix 92]**
1.Sequence	sequence (A >> B)	SEQuentiel	(A B)
2.Choix (Alternative)	choice (A [ ] B)	ALTernatif	(A | B)
3.Parallele (concurrency)	parallel (A || B)		A | | B
4.Repetition	repetition	@ Iteratif	$(A)^{+}$, $(A)^{n}$, $(A)^{*}$
5.Interruption	A disable B		$(B \rightarrow A)$
6.Intercalation (multi-threading)	Interleaved (A ||| B)	PARallèle	$(A \Leftrightarrow B)$
7.Simultaneité		SIMultanée	
8.Repeating Choice	Combination de 4 et 2	Combination de 4 et 2	$(A \mid B)^{n}$, $(A \mid B)^{*}$
9 Ordre Indépendant			(A & B)
10. Attente d'un temps ***n***			A (t > ***n*** seconds) B

Figure 1.9 - Constructeurs temporelles d'ordennancement de modèles de tâches

2.3.2Le contexte

Le développement des systèmes interactifs doit être placé dans un contexte réel d'activité humaine. Un bon système implique la compréhension du contexte dans lequel l'artefact développé sera introduit - l'environnement extérieur (*outer environment*) identifié par Herbert Simon [Simon 81] .

En pratique, la notion de ***contexte*** est très difficile à définir et comme le dit [Cockton et al. 95] la plupart des définitions d'usage général son inadéquates. Nous retenons la définition suivante de contexte, adaptée de [Storrs 95] :

‘Un contexte d’une tâche est la partie de l’environnement de travail dont l’état est pertinent pour cette tâche et qui sert à la fois comme ressource et comme contrainte à sa réalisation’

[Boy 95] fait la distinction entre deux types de contexte à prendre en compte dans la réalisation d’une tâche : **le contexte permanent** (ou stationnaire), qui est associé à l’environnement social et organisationnel où la tâche se déroule ; et le **contexte éphémère**, qui est associé aux situations émergentes de la dynamique de la réalisation de la tâche.

La tradition de la perspective socio-technique considère le contexte comme un autre système dont les composants sont des sous systèmes techniques (y compris les systèmes informatiques) et sociaux (y compris les rôles des utilisateurs). Modéliser le contexte consiste de modéliser l’information sur ces deux composants et sur les relations entre eux. Dans la plupart des cas, l’information sur le contexte obtenue pendant l’analyse est non-structurée, partiellement inconsistante ou incomplète. Elle est perdue à cause de la difficulté de l’intégrer à l’information structurée utilisée par les notations plus formelles des méthodes couramment utilisées. Cependant, sa modélisation et la définition explicite de ses relations avec d’autres modèles du système la rendent utile pendant le développement.

2.3.2.1L’information contextuelle et le développement des systèmes

Le fossé (‘*gap*’) entre l’analyse et la conception des systèmes interactifs peut être exprimé, selon [Cockton et al. 96] , comme le fossé entre l’information contextuelle et les modèles du système interactif, voir figure 1.10. L’**Information contextuelle** est l’information recueillie et organisée sur le contexte dans lequel le système sera developpé et utilisé. Il n’y a pas une définition consensuelle pour la notion de ‘contexte’ (voir par exemple [Cockton 95] ou encore l’édition spéciale de la revue Human Computer Interaction avec 24 commentaires sur un seul article [Seely-Brown et al. 94]). Nous avons adopté comme point de vue que l’information contextuelle est constituée typiquement par : l’information sur les utilisateurs, leurs tâches et leur domaine d’activité. **Les modèles du système interactif** correspondent aux modèles utilisés pour représenter la structure , l’apparence et le comportement (interne et externe) du système. Evidemment, comme il a été présenté aux sections 1.3 et 2.2, il y a plusieurs points de vue et niveaux d’abstractions possibles pour modéliser un système interactif. Chaque modèle a un but précis et une utilisation prévue selon la démarche de développement adoptée. **L’information** opérationnelle, présente dans la région **du fossé,** représente les connaissances des liaisons contexte-système. Il y plusieurs types de liaisons qui peuvent être établies entre eux, comme la contrainte, la transformation, etc. Plus on détermine les liaisons entre l’information contextuelle et le système, plus le fossé est réduit. L’information opérationnelle est créé pendant le processus itératif de IB comme un résultat partiel qui sert à alimenter ce processus.

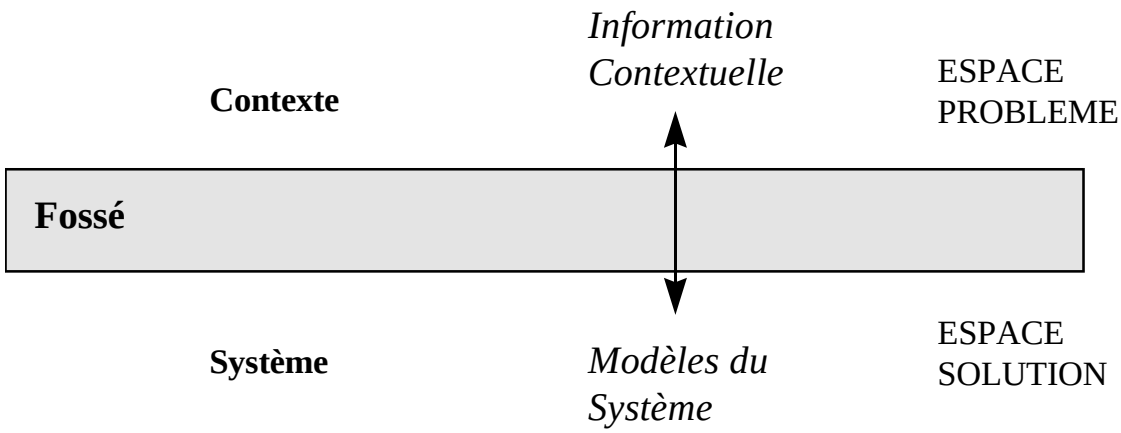


Figure 1.10 - Fossé entre l'information contextuelle et les modèles du système interactif

Nous associons les 5 classes principales de notations d'IHM identifiées par [Cockton 96] aux régions d'information de la figure 1.10:

- Les notations contextuelles représentent l'information du Contexte placée dans l'espace problème ;

- Les notations projectives et les notations de '*design rationale*' représentent des informations considérées pendant le développement, qui sont pour cela placées dans la région du Fossé;

- Les notations comportementales et les notations contructionnelles représentent respectivement la vue externe du Système et la vue interne du Système, qui sont placées dans la region de l'espace solution.

Il faut alors comprendre le problème et son contexte organisationnel **avant** de pouvoir proposer une solution. Cette compréhension est nécessaire pour faire l'identification des besoins et la spécification d'un modèle du système qui puisse les atteindre. L'obtention de ces connaissances est une fonction du processus d**'Ingénierie des Besoins,** dont un tour d'horizon est présenté au chapitre 2.

2.3.2.2Reutilisabilité de l'Information Contextuelle dans un domaine

Le développement traditionnel des systèmes suit un cycle de vie du début du projet jusqu'à sa livraison, son utilisation et son éventuel abandon. Cependant, nous nous basons sur notre expérience personnelle pour affirmer que le plus grand effort du cycle de vie du logiciel n'est pas la génération de nouveaux systèmes/programmes mais surtout la maintenance, l'intégration et la modification de systèmes existants. Cette constatation, bien que sans le support des travaux quantitatifs pour la valider, est aussi faite par Ivar Jacobson dans son livre (page 21) 'Object Oriented Software Engineering' [Jacobson 92]):

' All systems change during their life cycles. This must be borne in mind when developing systems expected to last longer than the first version (that is, pratically all systems). Most development methods today focus on new development, treating revision work only briefly, even though it is known that changes constitute the main part of the total life-cycle cost of most systems....'.

En vérité, le développement des systèmes est vraiment un processus de changement

progressif [Lehmann 80]. Le nouveau développement n'est qu'un cas spécial - un changement à partir de 'rien' jusqu'à la première version du système.

La notion de développement du logiciel comme une construction d'une théorie, proposé par [Naur 85] pour la programmation, aide à comprendre pourquoi ce nouveau développement est si difficile. Selon [Naur 85], un programme reflète une théorie[6] construite par les concepteurs à travers une compréhension croissante et évolutive du domaine de l'application. Le système (et sa documentation) n'explicite que quelques aspects de la théorie sous-jacente. Le développement d'un nouveau système est traditionnellement perçu comme une activité où les idées informelles sur le domaine du problème sont formalisées graduellement par l'analyste (concepteur) à la recherche d'une solution. Ainsi, toute la connaissance avant la formalisation tend à être perdue : théoriquement, du fait de la difficulté d'intégration des informations informelles dans la spécification et pratiquement, du fait de la dispersion de cette connaissance en des documentations volumineuses, inutilisables et non-maintenables [Jarke et all. 93] .

Un exemple typique de théorie construite est l'information contextuelle. Même s'il est clair qu'il n'y pas de grandes difficultés à recueillir un grand volume d'information contextuelle [Cockton 96], cette tâche a un coût élevé pour un seul développement. Cependant, comme ces informations sont relativement stables, elles peuvent être reutilisées pour le développement d'autres systèmes dans le même domaine et organisation. Le coût est alors distribué par les diverses applications.

Un nouveau concept prometteur pour les projets où il y a plus d'un produit final (par exemple, plusieurs programmes dans un grand système ou même plusieurs systèmes similaires) ou pour quand le temps de vie utile du système est estimé grand: l'ingénierie du domaine, basé sur la reutilisabilité des modèles du domaine [Prieto-Diaz 87] .

L'approche d'ingénierie du domaine a pour but d'analyser et de modéliser less aspects d'un domaine du problème dans un modèle du domaine qui sera utilisé et reutilisé par les systèmes qui le partagent. Cela implique, entre autres choses, l'existence d'un double cycle de vie : un pour créer et maintenir le modèle du domaine et l'autre pour développer et maintenir les différentes systèmes dans ce domaine [Gronquist et al. 92] .

Nous pensons que l'information contextuelle recueillie est un modèle du domaine et peut être bien (re)utilisé si :

- elle est représentée et organisée ;
- plusieurs systèmes la partagent total ou partiellement;
- les modèles ont une utilisation prévue dans le cycle de vie de développement .

2.4Qualité d'un système interactif

Qualité peut être définit comme 'la totalité de propriétés et caractéristiques d'un produit ou service qui contribuent à sa capacité de satisfaire des besoins' déterminés [IEEE 83] . L'obtention de la qualité est toujours un compromis : la satisfaction d'une propriété peut nuire à la satisfaction d'une autre. Néanmoins, une propriété n'est pas bonne ni mauvaise de manière absolue : elle sert simplement pour définir un espace de possibilités pour la conception du produit ou service [Abowd et al. 92] . R. Freund a

6 Le mot "théorie" est utilisé par Naur dans le sens du philosophe anglais Gilbert Ryle, et dénote une "compréhension qui nous permet donner des réponses qualifiés aux questions du domaine d'intérêt, à agir intelligemment en situations relevantes et à prendre des décisions en rapport à ces actions".

bien observé que la qualité est un concept relatif, dont l'importance des différentes propriétés changent d'une personne à l'autre et changent au cours du temps. Il est très important que l'approche de définition des propriétés désirables reconnaisse ces différences et permette différents poids ou priorités de jugement selon le contexte ou la situation considérée [Freund 85] .

Même reconnue comme essentielle, la recherche de la qualité implique des choix. En effet, dans le rapport [REAW 91] un groupe de travail a mis l'obtention de la qualité en contreposition de deux autres objectifs : date de livraison et budget. Pour eux, deux de ces trois objectifs ne peuvent être atteints qu'au détriment du troisième. Comme la qualité est l'objectif le plus difficile à mesurer, il est en général le plus affecté pour respecter le temps et le coût.

Les sections suivantes discutent trois aspects de la qualité des systèmes interactifs : la section 2.4.1 présente le point de vue des facteurs de qualité établis avec les besoins du système ; la section 2.4.2 discute l'utilisabilité comme un composant critique de qualité des systèmes interactifs. ; la section 2.4.3 présente la définition des facteurs de utilisabilité.

2.4.1Facteurs de Qualité

La préoccupation de la qualité est à l'origine de plusieurs initiatives.

Auparavant, la qualité était évaluée **à la fin** du processus de développement, dans une phase de Test, en espérant que les résultats soient positifs, mais sans possibilité de refaire tout le processus.

A cause de la croissante complexité des systèmes interactifs d'aujourd'hui, une attention continue à sa qualité **pendant** le processus de développement est nécessaire pour l'obtention de systèmes de qualité. L'intention est de chercher la qualité par construction.

Cependant, les approches 'Qualité Totale' (TQM - *Total Quality Management*) suggèrent qu'il est plus rapide et moins cher de concentrer l'effort **au début** du développement, c'est à dire, détecter et corriger les fautes aussi tôt que possible dans le cycle de vie [Denning 86] . Cela signifie que plus d'efforts doivent être réalisés durant les étapes d'analyse/ingénierie des besoins [Zultner 92a]. En effet, des recherches empiriques montrent que plus de 50% des erreurs qui surviennent pendant le développement des systèmes sont le résultat de besoins définis de manière imprécise [Boehm 81] . Augmenter la qualité des définitions des besoins est donc fondamental pour satisfaire les clients/utilisateurs et pour améliorer la productivité du processus de développement.

Nous suivons cette ligne d'action où les critères de qualité doivent être considérés conjointement avec les **besoins** du système, plus que comme seuls facteurs à mesurer à la fin du développement.

2.4.2Qualité des Systèmes Interactifs et Utilisabilité

Pour les systèmes interactifs, la définition des facteurs de qualité nécessite plus que de remplir/satisfaire les besoins fonctionnels : la notion d'utilisabilité du système est aussi importante que la notion de son utilité, qui a longtemps été la plus importante pour les systèmes informatiques traditionnels. L'utilisabilité est devenu un facteur critique pour la satisfaction et la productivité des utilisateurs.

Pour le GL, l'utilisabilité est seulement un type de besoins non-fonctionnel [Abowd et al. 92] , de même dégrée d'importance que d'autres facteurs tels que la portabilité, la

maintenabilité, etc. Les méthodes traditionnelles du GL ne permettent pas (ou s'ils permettent cela est inadéquat) le développement de l'interface ni la détermination des besoins associés à l'utilisabilité du système. Dans le cycle de développement associé à une méthode traditionnelle, il y a une prépondérance des besoins du noyau fonctionnel du système par rapport aux besoins de l'interface. De plus, bien que les besoins non-fonctionnels soient fréquemment enregistrés dans un catalogue des besoins pour consultation par les concepteurs, aucune aide n'est disponible pour l'analyste lors de la spécification des besoins non-fonctionnels (rédaction de ce catalogue) ni pour le concepteur lors de la vérification que les besoins non-fonctionnels sont bien pris en compte (lecture de ce catalogue).

En effet, on ne peut traiter des besoins de systèmes interactifs sans aborder ses besoins d'utilisabilité. L'utilisabilité est un concept qui date du début des années 80 et il est historiquement lié à la convivialité (*user friendliness*) de l'interface homme-machine. Cependant, aujourd'hui, l'utilisabilité se définit de façon beaucoup plus large qu'auparavant et désigne les aspects d'adéquation entre les caractéristiques des utilisateurs et de leur interaction avec le système pour réaliser leurs tâches. Ces aspects font référence, entre autres, aux attributs comme la facilité d'apprentissage, la facilité d'utilisation, la facilité de mémorisation, l'utilisation sans erreurs, l'efficacité d'utilisation et la satisfaction. Comme [Bastien 96] la remarqué, ces attributs ne sont pas forcement indépendants les uns des autres et ils reflètent différents points de vue sur la mesure de l'utilisabilité :

- le point de vue 'performance des utilisateurs', qui met l'accent surtout sur les temps d'apprentissage et d'utilisation et sur les erreurs commises ;

- le point de vue 'utilisateur', qui se base sur la mesure des effets de l'interaction sur l'utilisateur (effort mental, charge cognitive, etc) ;

- le point de vue 'produit', qui permet de le mesurer en prenant en compte des recommandations, principes et guides de style pour les caractéristiques du logiciel.

L'intégration de ces différents points de vue correspond à ce que Bevan appelle la **Qualité d'Utilisation** (*Quality of Use*) [Bevan 95] et à ce que Bastien appelle la **Qualité Ergonomique des Systèmes Interactifs** [Bastien 96] , c'est à dire, tous les attributs permettant de déterminer si un logiciel est adapté aux utilisateurs visés et aux tâches pour lesquelles il a été développé. En fait, utilisabilité ne doit plus être liée qu'aux mesures de performance humaine, typiquement en termes de vitesse et coefficients d'erreur, comme c'était le cas auparavant. En pratique, des mesures d'utilisabilité sont des mesures du soutien que le système offre aux tâches que l'utilisateur réalise pour arriver à ses buts. On adopte ici la définition de [Farenc 95] : par définition, l'**utilité** est la mesure du rapport entre les fonctions du système et les tâches de l'utilisateur alors que l'**utilisabilité** est la mesure du rapport entre la manière à travers laquelle une tâche interactive est réalisée par un utilisateur particulier et le profil cognitif de cet utilisateur. Une compréhension de ces tâches est une condition nécessaire pour la définition des facteurs d'utilisabilité, dont l'importance des approches de modélisation des tâches vues à la section 2.3.1.

2.4.3Facteurs de l'Utilisabilité

Devant la multiplicité et l'hétérogénéité des éléments qui concourent à l'utilisabilité, plusieurs classifications des facteurs d'utilisabilité sont possibles. Evidemment, les

classifications ne sont pas consensuelles. Plusieurs travaux de recherche sont actuellement concentrés sur la définition et la mesure de l'utilisabilité d'un logiciel [Abowd et al. 92]; [Farenc et al.95]; [Shackel 91]; [Nielsen 93]; [Jordan 94]; [Bevan 95]; [ISO 94]. Citons quelques définitions de facteurs d'utilisabilité.

2.4.3.1Les facteurs de qualité selon [Abowd et al. 92]

Cette définition est une extension des critères de qualité du logiciel proposés par [McCall 77] . L'utilisabilité est enlevé de la categorie de qualité des opérations du logiciel pour devenir une catégorie en elle-même. Les facteurs de utilisabilité de la typologie de [Abowd et al. 92] sont classés en trois groupes :

1) facilité d'apprentissage;

2) flexibilité d'interaction ;

3) robustesse d'interaction.

Une explication plus précise et des exemples de ces facteurs peuvent être trouvés dans [Abowd et al. 92] .

Facilité d'apprentissage (*learnability*)

Ce concept regroupe les critères du logiciel qui permettent à l'utilisateur débutant de comprendre initialement comment l'utiliser et ensuite arriver par l'expérience à un niveau maximal de performance. Les critères qui contribuent directement à la facilité d'apprentissage sont :

- Prévisibilité (*predictability*), qui définit le support permettant à la connaissance de l'historique des interactions passées d'être suffisante pour déterminer le résultat de l'interaction future;

- Synthèse (*synthesis*), qui définit le support pour l'utilisateur permettant d'identifier quelles opérations passées ont eu une influence sur l'état courant;

- Familiarité (*familiarity*), qui définit la mesure de la corrélation entre la connaissance possédée par l'utilisateur et la connaissance requise pour une interaction efficace;

- Généralisabilité (*generalizability*), qui définit le support pour permettre à l'utilisateur d'étendre la connaissance du comportement d'une interaction spécifique à d'autres situations identiques ou similaires mais inconnues;

Flexibilité d'interaction

Ce concept regroupe les critères qui permettent une multiplicité des manières d'échanger des informations entre l'utilisateur et le système. Les critères qui contribuent à la flexibilité d'interaction sont :

- Initiative de dialogue, qui définit les types de dialogues en conformité avec le degré de liberté (latitude décisionnelle) pour effectuer une action avant de poursuivre une tâche courante interrompue :

  - ♦ non-préemption, si l'utilisateur a toujours l'initiative du dialogue ;
  - ♦ préemption locale, qui ne bloque que le fil de dialogue interrompu et laisse l'utilisateur continuer les autres fils de dialogue ;

- ♦ préemption globale, qui interdit à l'utilisateur d'effectuer toute autre action que celle requise par le système; ce critère est typique des messages d'erreur ;

- Dialogue à fils multiples (*multithreading*), qui définit la capacité du système à permettre la réalisation de plusieurs tâches, en général de manière entrelacée ou parallèle;

- Migration des tâches (*task migrability*), qui définit la capacité du système à permettre de changer le contrôle de l'exécution des tâches entre l'utilisateur et le système;

- Substituabilité (*substitutivity*), qui définit la capacité du système à permettre de remplacer des valeurs d'entrée et sortie par d'autres valeurs équivalentes;

- Multimodalité (*multimodality*), qui définit la capacité du système à permettre d'utiliser les multiples canaux de communication (modalités) humains, comme vision, audition, etc. La multimodalité peut être *exclusive*, si un canal est utilisé en entrée (ou sortie) à chaque fois, même si les autres sont disponibles, ou *synergique*, quand les entrées (ou sorties) peuvent être exprimées par l'intermédiaire d'une combinaison de modalités ;

- Configurabilité (*customizability*), qui définit la capacité du système à permettre des modifications de l'interface par l'utilisateur du système ou par le système lui-même (modification automatique). La configurabilité peut se rencontrer à différents niveaux et sous différentes formes : il est possible de modifier certains aspects de la présentation ou du comportement du système. Les divers noms présents dans la littérature sont un reflet de la diversité des définitions : *adaptivity, adaptability, tailorability, parameterazibility.*

Robustesse d'interaction

Ce concept regroupe les critères de l'interaction qui supportent la réalisation et l'évaluation des buts. Les critères qui contribuent à la robustesse de l'interaction sont :

- Observabilité (o*bservability*) ; la capacité du système à rendre perceptibles à l'utilisateur les variables d'état internes pertinentes pour la tâche en cours. Cette propriété est affinée par la propriété de l'honnêteté (*honesty*) qui en plus établie que la forme du rendu doit conduire l'utilisateur à l'interpréter correctement ;

- Récuperabilité (*recoverability*), la capacité qu'a l'utilisateur de réaliser une action corrective lorsqu'une erreur est reconnue. Cette propriété recouvre plusieurs aspects : le plus courant est *l'annulabilité*, qui permet de revenir à l'état antérieur en annulant l'effet de la dernière commande, mais il y a d'autres aspects comme par exemple la réparation (*repair*) et la récuperabilité avant et arrière (*forward and backward recovery*)- qui permet de revenir à un état qui satisfait une condition déterminée .

- Conformité à la Tâche (*task conformance*), qui définit le degré de support du système aux tâches que l'utilisateur veut réaliser; la conformité exprime à la fois si les tâches qui intéressent l'utilisateur sont supportées et si l'utilisateur trouve ce support adéquat.

2.4.3.2Les facteurs d'utilisabilité selon les normes ISO

Les facteurs d'utilisabilité de la spécification ISO 94 [ISO 94] sont des mesures de la performance - voir l'efficacité et l'efficience (en anglais '*effectiveness*' and '*efficiency*') - et de la satisfaction pendant l'utilisation d'un produit dans un contexte de travail particulier. Ces mesures sont associées à des attributs, dont l'importance change suivant les buts considérés. Par exemple, si l'usage n'est pas fréquent, les attributs d'apprentissage et de re-apprentissage ont plus d'importance. Exemples d'attributs d'utilisabilité sont donnés au figure 1.11. Les carrés vides indiquent l'absence d'attributs.

Objectif d'Utilisabilité	**Attributs de Efficacité**	**Attributs de Efficience**	**Attributs de Satisfaction**
Utilisabilité totale	pourcentage de buts accomplis ; pourcentage de utilisateurs qui réussissent à compléter ses tâches ; précision moyenne de tâches complètes	temps pour compléter une tâche ; tâches complétées par unité de temps ; coût monétaire d'exécuter la tâche ;	échelle d'indice de satisfaction ; taux d'usage dans le temps ; fréquence de réclamations ;
Utilisabilité des utilisateurs entraînés	numéro de tâches relevantes réalisées ; pourcentage de fonctions relevantes utilisées	efficience relative comparée avec un utilisateur expert ;	échelle d'indice de satisfaction avec les tâches relevantes
Utilisabilité de grand public ('walk up and use')	pourcentage de tâches complétées avec succès dans le premier essai	temps pris pour le premier essai ; efficience relative au premier essai	taux d'utilisation volontaire
Utilisabilité d'usage non fréquente ou intermittente	-	temps pris pour re-apprendre les fonctions ; numéro d'erreurs persistants	fréquence de re-utilisation
Minimisation des besoins de support	numéro de références à documentation ; numéro d'appels au support ; numéro d'accès à l'aide *on-line*	temps productif ; temps pour apprendre le critère	
Apprentissage	numéro de fonctions apprises ; pourcentage d'utilisateurs capables d'apprendre le critère	temps pour apprendre le critère ; temps pour re-apprendre le critère ; efficience relative pendant l'apprentissage	échelle d'indice pour facilité d'apprentissage
Tolérance aux erreurs	pourcentage d'erreurs corrigés ou identifiés par le système ; numéro de erreurs d'utilisateur tolérés	temps pris pour la correction d'erreurs	échelle d'indice de manipulation d'erreurs
Lisibilité	pourcentage de mots vus correctement à une distance normal		

Figure 1.11 - Attributs mesurables d'utilisabilité (d'après ISO DIS 9241-11)

2.4.3.3Les facteurs d'utilisabilité selon [Macaulay 95]

Pour la définition présentée par [Macaulay 95] , les facteurs sont appelés objectifs d'utilisabilité, et doivent être associés à des critères de succès mesurables. Exemples de critères mesurables sont :

- nombre de touches ;
- nombre de commandes utilisées ;
- nombre d'actions que l'utilisateur réalise pour exécuter la tâche ;
- temps pour l'exécution des actions et des commandes ;
- temps pour apprendre un groupe de commandes ;

- nombre de commandes mémorisées ;
- connaissance de l'usage du système ;
- attitudes et opinions ;
- disponibilité du support matériel;
- type d'erreur dans l'exécution des actions ;
- temps pour corriger une erreur.

2.4.3.4Utilisabilité et Satisfaction

La satisfaction est une notion très difficile à définir. Les psychologues généralement sont d'accord pour affirmer que la satisfaction dans une situation donnée est l'union des sensations, attitudes et réactions (positives et négatives) de quelqu'un en rapport à un ensemble de facteurs qui affectent cette situation [Schwab & Cummings 73, Cross 73] . Dans le contexte socio-technique de l'utilisation d'un système, il est clair que les facteurs techniques et humains interagissent entre eux pour déterminer ces réactions. Bien que la satisfaction puisse être reliée à la performance, il est difficile d'indiquer le sens et l'intensité de cette relation [Srivasta 77] . Didier Mottay [Mottay 96] , par exemple, associe la satisfaction à 5 caractéristiques du travail proposées par [Hackman et Oldham 80] :

1. variété des compétences,
2. identité de la tâche
3. valeur de la tâche
4. autonomie
5. '*feedback*'.

Succinctement, ces caractéristiques influencent un ensemble d'états psychologiques intermédiaires qui, à leur tour, influencent la satisfaction au travail et la motivation interne. Par exemple, les trois premières caractéristiques influencent l'intérêt du travail ; l'autonomie influence la responsabilité vis à vis des résultats et le feedback influence la connaissance des résultats. Les changements techniques dans le travail d'une personne, comme l'adoption d'un système informatique, peuvent influencer ces caractéristiques et en conséquence la satisfaction. Malgré son importance, la satisfaction ne sera pas considéré directement dans cette thèse parce qu'elle est encore un sujet ouvert de recherche au delà de l'envergure de notre travail.

3.Synthèse

Dans ce chapitre, nous avons examiné les concepts fondamentaux des systèmes interactifs et de leur développement qui vont intervenir dans la compréhension de notre travail. Lors de la réalisation des activités en vue de déterminer les besoins d'un système interactif utile, utilisable et capable de soutenir efficacement les activités de

travail, il faut entre autres considérer les point suivants :

1) **Systèmes interactives sont des outils pour soutenir des activités humaines ;** En considérant un système interactif comme un soutien à une activité, nous adoptons la perspective d'outil d'un système interactif, parce qu'elle permet de se concentrer sur la correspondance entre le sens du processus d'interaction et la sémantique du processus de travail avec un outil et ainsi d'utiliser cette correspondance comme la source primaire des principes de conception ;

2) **Le développement du système interactif doit prendre en compte** plusieurs types de connaissances**,** notamment sur **les tâches des utilisateurs** et **leur contexte** . La **modélisation de la tâche** doit être à la base du développement parce qu'elle est une manière adéquate de considérer le **point de vue de l'utilisateur ;** Cependant, elle doit être située dans un cadre fourni par l'utilisation complémentaire **des informations contextuelles** , qui permettent de considérer les besoins de ces tâches dans un contexte large d'utilisation et qui, en plus, peuvent être reutilisées pour d'autres développements futurs dans le même domaine ;

3) **il faut faire des projections sur le comportement du système interactif** dans une situation de travail parce que la connaissance du système futur est nécessaire avant qu'il ne soit conçu ;.

Il faut établir au préalablement les facteurs d'utilisabilité à considérer pendant le développement ; S'agissant de permettre le rapport entre le GL et l'IHM, nous adoptons la typologie des facteurs d'utilisabilité proposée par [Abowd et al. 92] , qui est fondée sur les critères de qualité du logiciel proposés par [McCall 77] . Ces facteurs ne sont pas liés à un système particulier mais à une classe de systèmes. Leur caractère général les rends réutilisables pour différents systèmes .Chapitre II - Ingénierie des Besoins : des systèmes classiques aux systèmes interactifs

---

L'Ingénierie des Besoins (on utilise aussi l'abréviation **IB**) est, ces dernières années, un important et fondamental sujet de recherche. L'IB est le terme générique établi dans le Génie Logiciel pour la combinaison de théories, modèles, techniques et outils pour la détermination des objectifs, fonctions et contraintes d'un système [Zave 95].

L'Ingénierie des Besoins est un problème clé dans le développement de systèmes informatiques surtout parce qu'elle permet d'identifier les objectifs du système (le problème fondamental qui consiste à savoir **ce qui** doit être développé) [Brooks 87]:

4)

The hardest single part of building a software system is deciding what to build. (...) No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.

En étant le premier pas dans le processus de développement de logiciels, il est clair que si l'IB n'est pas efficacement et adéquatement faite, un impact significatif dans la qualité du système en résulte : des besoins mal établis sont grandement responsables du dépassement du budget et des chronogrammes, très fréquents dans les systèmes actuellement développés. Plusieurs besoins incorrects arrivent non détectés aux phases postérieures du cycle de vie et corriger ces erreurs pendant ou après l'implémentation est une activité extrêmement coûteuse: selon [USDD 91] c'est deux fois plus cher que de corriger les erreurs avant la conception et l'implémentation.

Dans ce chapitre, pour mieux comprendre l'IB, nous verrons la définition des besoins, et le changement de point de vue entre l'Analyse et l'IB. Après cela, la façon de réaliser l'IB pour les systèmes dits classiques est présentée, typiquement par l'intermédiaire de concepts basiques de l'IB, comme par exemple, quel est le point de vue sur les besoins et les facteurs de qualité, quelles sont les activités réalisées, quelles sont les techniques de recueil utilisées, et quels sont les résultats souhaités.

Le terme 'système classique' est utilisé ici en opposition aux systèmes interactifs. Le

développement de systèmes classiques était commun quand la technologie de l'information était localisée dans quelques îles d'automatisation, les types des services fournis par les systèmes étaient très limités - en général des systèmes orientés transactions basées en ordinateur central (*mainframe*) - et l'importance de l'utilisabilité d'un logiciel n'était pas encore cruciale pour son acceptation. Dans les dernières décennies, cela a beaucoup changé et les systèmes interactifs sont à la base de nouvelles utilisations de l'ordinateur.

Ensuite, nous examinerons pourquoi et comment ces concepts de l'IB pour des systèmes classiques doivent être modifiés pour faire face aux spécificités des systèmes interactifs.

4.Qu'est-ce que c'est un Besoin ?

Aujourd'hui, une définition cohérente et consensuelle du terme 'besoins' dans le domaine du GL n'existe pas encore. Plusieurs textes dans la littérature définissent un besoin comme une fonction ou une caractéristique d'un système qui est nécessaire à (aux) l'utilisateur (s). Par exemple, un 'besoin' est défini dans [IEEE 90] comme :

5)

'...(1) a condition or capability needed by a user to solve a problem or achieve an objective; (2) a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents; (3) a documented representation of a condition or capability as in (1) or (2).'

Cependant, les besoins ne sont pas seulement des fonctions. C'est, selon [Christel & Kang 92], une méprise introduite en grande partie par la diffusion des techniques d'analyse structurée, largement utilisées dans les années 70.

Selon l'analogie de Bob Balzer présentée dans [Arango 88], cf. Figure 2.1, un besoin est un prédicat sur les buts à atteindre par le système, alors qu'une spécification est un plan pour une solution, instanciée par un programme. Cette notion est partagé par Julio Leite pour qui un besoin est "une condition nécessaire pour remplir un certain objectif" [Leite 90].

Besoin	➔	Prédicat sur les buts ("*goal predicate*")
Spécification	➔	Plan pour une solution ("*plan for a solution*")
Programme	➔	Solution

Figure 2.1 - Analogie : Besoin *versus* Spécification *versus* Programme

Cette notion de prédicat permet de comprendre l'aspect déclaratif des besoins pour exprimer plusieurs propriétés du système et aussi pourquoi les classification actuellement disponibles sont aussi variées que nombreuses.

Ashworth classifie les besoins en 5 catÈgories [Ashworth 89]:

1. Fonctions,

2. Données,

3. Besoins non-fonctionnels,

4. Buts (dÈfinis pour guider le développeur du système vers une implémentation des besoins en accord avec l'utilisateur)

5. Contraintes de la conception et de l'implémentation

[Mullery 96] a identifié 3 types de besoins non-ortogonaux:

1. Domaine (ou Opérationnel), pour exprimer les aspects du domaine que le système doit prendre en compte ;

2. Technique (ou du Système) pour spécifier ce que le système doit fournir pour soutenir les besoins du Domaine ;

3. Contractuel, pour exprimer les restrictions de temps et de ressources pour le développement et les critères pour l'acceptation.

Cependant, la plupart des classifications proposent une distinction entre besoins fonctionnels et besoins non-fonctionnels (comme [Southwell 87] et [Davis 93]).

Couramment, les besoins fonctionnels définissent **ce que** le système doit faire, ce qui est traduit par la spécification en termes de fonctions spécifiées pour le système [Davis 93]. Les Besoins fonctionnels sont à l'origine de la **fonctionnalité** du système.

Les besoins non-fonctionnels incluent une grande variété de types de besoins. Dans la classification de [Southwell 87], par exemple, les Besoins non-fonctionnels sont :

- besoins de performance ou de fiabilité,

- besoins d'interfaces et

- des contraintes pour la conception.

En fait, la tendance de l'IB est d'augmenter l'intérêt sur les besoins non fonctionnels [Boehm 96].

Evidemment, certains besoins non-fonctionnels sont autant essentiels que les besoins fonctionnels pour le succès d'un projet de développement de logiciel. Néanmoins, cela ne doit pas suggérer que tous les types de besoins sont toujours significatifs pour toutes les applications et tous les domaines. Quelques types de besoins peuvent varier de " essentiel " à " inapplicable " selon la nature du logiciel, le contexte dont le système fait parti et les décisions des *acteurs*.

5.De l'Analyse à l'Ingénierie des Besoins

Dans le début des années 80, il a été largement reconnu que les principaux problèmes pour améliorer la qualité du logiciel et la productivité de son développement étaient concentrés dans les premières parties du cycle de développement du logiciel (pour une démonstration empirique voir [Curtis et al. 88]). C'est bien sûr la raison de l'intérêt croissant pour l'Analyse des Systèmes et plus récemment pour le processus de l'Ingenierie des Besoins.

Ces deux termes reflètent différents points de vue sur le processus de détermination des besoins du système. Ces différences, plus que pour montrer l'évolution chronologique du domaine de l'Analyse à l'IB, sont importantes pour faire remarquer le changement de la problématique et par conséquence le changement des solutions proposées. Les sections suivantes montrent les caractéristiques génériques des processus d'analyse et d'IB.

5.1L'Analyse

Le processus traditionnellement connu comme Analyse (ou Analyse des Systèmes) consiste à comprendre le domaine du problème (l'environnement, les utilisateurs et leurs caractéristiques) et utiliser cette compréhension pour **l'Analyse des Besoins** et la Conception de Haut Niveau, c'est à dire, pour faire respectivement l'identification des besoins et la spécification conceptuelle d'un système qui puisse les atteindre. Puisque appliquée d'abord aux systèmes d'information, l'Analyse de Systèmes est orientée surtout vers la spécification d'une application informatique. **Plusieurs suppositions sont implicites à l'Analyse**, en particulier à l'Analyse des Besoins [Jarke 94]:

i) Il y a un **problème bien défini** qui peut être clairement délimité et décrit (et ensuite résolu); une fois que le système est terminé, il reste stable ;

ii) La spécification des besoins forme la base d'un **contrat** entre le concepteur et le demandeur, qui n'est pas normalement le responsable de l'utilisation subséquente du système; ce contrat est supposé rester stable pendant tout le développement du système ;

iii) Les **utilisateurs** ne connaissent pas le domaine informatique mais sont spécialistes dans le domaine d'application et ils savent bien ce qu'ils veulent, il suffit donc seulement de leur poser les bonnes questions de manière adéquate;

iv) Les méthodes d'analyse sont des **généralisations** ou des dérivations des **méthodes informatiques** utilisées dans les autres phases du développement du logiciel. Les méthodes structurées dans les années 70 et 80, par exemple, ont été crées chronologiquement pour la programmation structurée, la conception structurée et finalement pour l'analyse structurée. On constate le même schéma chronologique pour l'approche orientée objet.

v) L'Analyse de Besoins est un **processus linéaire basé sur les abstractions** : il est possible d'atteindre sans la participation de l'utilisateur une spécification des besoins complète, cohérente et non ambiguë avant le début des autres étapes du développement.

5.2L'Ingénierie des Besoins

Le processus de l'Ingénierie des Besoins (*Requirements Engineering* en anglais) cherche l'incorporation d'une orientation d'ingénierie à l'analyse de systèmes. Le terme 'Ingénierie des Besoins' a été introduit par E. Dubois et J. Hagelstein [Dubois & Hagelstein 89] pour faire référence à la partie du cycle de vie du logiciel qui consiste à l'investigation des besoins (buts de l'utilisateur, fonctions faisant partie du système et contraintes imposées à son développement) et la spécification d'un modèle du système qui permette de les atteindre. Cependant 'Ingénierie' ici n'a pas le sens d'une approche purement positiviste et pratique sans considérations contextuelles mais surtout le sens d'une approche professionnellement systématique et responsable pour l'identification et la maintenance des besoins [Loucopoulos & Potts 96].

En fait, l'Ingénierie n'est pas seulement une extension de l'analyse mais surtout un changement de point de vue pour être plus réaliste [Mullery 96]. Il a été reconnu que :

i) le problème que le système doit soutenir n'est pas toujours bien défini ;

ii) les besoins changent dans le temps, même pendant le développement et ils ne peuvent pas être considérés comme fixes ;

iii) les utilisateurs ne savent pas nécessairement bien ce qu'ils veulent du système et en général leurs exigences émergent ou se modifient à partir des interactions avec l'analyste et les versions préliminaires (exécutables ou non) du système ;

iv) il n'est pas possible d'arriver d'un coup à des spécifications correctes et complètes - parfois il faut plusieurs essais pour définir des besoins, ou il faut manipuler des spécifications avec différents degrés d'incomplétude, ou il faut attribuer des priorités différentes aux besoins.

Cependant, l'IB reste encore **orientée système** : l'analyse du domaine du problème continue à être faite typiquement en termes de concepts des modèles informatiques et non en termes de concepts des utilisateurs.

L'Ingénierie des Besoins est typiquement ***multidisciplinaire***, en utilisant des paradigmes et modèles de plusieurs disciplines comme les systèmes d'information, le génie logiciel, la sociologie et les IHM [Andriole 93] . L'IB reconnaît les besoins non plus comme contrats mais comme ***liaisons entre le système et le contexte de travail*** qu'il souhaite soutenir. Une fois que la réalité est dynamique, le contexte change. Au fur et à mesure que le contexte change, les besoins du système changent, ce qui implique un changement du système. Un changement du système modifie le contexte (ou même la compréhension établie sur lui) dans un mouvement cyclique de changements. Due à ce cycle, l'IB est surtout orientée vers la paire (contexte, système) et son but est de fournir un soutien conceptuel à l'identification et la maintenance de ces liaisons [Jarke 94]. Ce soutien est **itératif** et en double sens, pour permettre de gérer les changements et leur propagation ainsi que de garder la trace des informations recueillies.

5.2.1Les Trois Dimensions de l'Ingenierie des Besoins

Le processus d'IB peut être vu comme un ensemble d'activités qui part du domaine du problème et qui arrive à un modèle des besoins du système.

Au début de l'IB, les connaissances sur le problème (et par conséquence sur le système) sont incomplètes. Il y a quelques caractéristiques connues mais il y a une **compréhension vague** du système avec beaucoup de définitions confuses et incomplètes. Dès que plusieurs personnes (différents utilisateurs, différents concepteurs) sont impliquées dans cette activité, il y a **plusieurs points de vue personnels** et il n'y a pas un format commun pour les représenter. En général, on utilise des **représentations informelles** comme, par exemple, le langage naturel ou des notations graphiques sans une sémantique précise.

A la fin de l'IB, il doit exister une spécification du système à construire (un modèle conceptuel), qui sert de point de départ pour la macro-activité de Synthèse. On doit avoir, alors, une **compréhension plus complète** du système. Cette compréhension doit être le résultat d'un **consensus** (agréement commun) entre les personnes impliquées. Pour éviter différentes interprétations de cette compréhension, on doit utiliser une **représentation plus formelle** ou rigoureuse pour l'enregistrer.

Alors, on peut considérer trois grands objectifs dans le processus de l'IB, associés respectivement aux dimensions de **Spécification**, **Représentation** et **Accord** [Pohl 93] :

- i) transformer une compréhension vague en une compréhension complète, c'est à dire, augmenter la compréhension du problème et du système;

- ii) transformer la connaissance des représentations informelles en une représentation formelle, c'est à dire, formaliser la connaissance du système;

- iii) obtenir un consensus à partir des différents points de vue personnels.

5.2.2Les problèmes typiques de l'IB

Selon [Christel & Kang 92] les problèmes typiques de l'IB peuvent être groupés en 3 catégories :

1. problèmes d'envergure (*scope*), conséquence de l'excès ou de l'insuffisance d'informations ;

2. problèmes de compréhension, conséquence d'une communication déficiente entre les acteurs impliqués ;

3. problèmes de volatilité, conséquence de la nature dynamique des besoins.

Cette classification est très générique et très attachée aux conséquences et non aux causes des problèmes, ce qui rend difficile la proposition de solutions.

Les problèmes de l'IB sont organisés par [Zave 95] en trois grandes classes :

1. problèmes pour l'identification des buts, fonctions et contraintes du système, y compris le problème de communication utilisateur-analyste, le problème des objectifs vagues, le problème d'allocation de fonctions entre le système et les utilisateurs, le problème de priorités de satisfaction, le problème d'estimation de coût et délai, et le problème d'assurance de complétude ;

2. problèmes de spécification du comportement du système, y compris le problème d'intégrer plusieurs points de vue et représentations, le problème d'évaluer les différentes stratégies pour satisfaire les besoins, le problème de choisir les besoins à satisfaire, le problème d'obtenir des spécifications complètes et cohérentes, le problème de validation des besoins, le problème de créer des spécifications réalisables (dont l'implémentation est possible) ;

3. problèmes de gestion de l'évolution de systèmes et familles de systèmes ;

Cette classification présente aussi des inconvénients : elle est aussi générique que la première et ses catégories ne sont pas nécessairement orthogonales. Par exemple, la modification d'une fonction peut être considérée soit comme une tâche d'identification des fonctions (classe 1) soit comme une tâche d'obtention de cohérence de spécification du système (classe 2).

Malgré ces inconvénients, nous retenons cette classification parce qu'elle est surtout attaché aux causes des problèmes pour lesquels une approche de l'IB doit proposer des solutions.

Nous faisons deux modifications à cette classification:

1. nous ajoutons deux autres problèmes que nous trouvons aussi essentiels, à savoir :

    - Problèmes de manque de connaissance sur l'univers (contexte) de l'application ;

    - Problèmes de 'gestion de changement' ;

2. nous séparons les problèmes de communication entre utilisateurs-analystes de la classe 1 et les promouvons au niveau d'une classe (la classe 4), à cause de son extrême importance.

Ces problèmes sont résumés dans les sections suivantes.

5.2.2.1La manque de connaissance sur l'univers (contexte) de l'application

Le processus d'IB peut être vu comme un sous-processus du processus de développement des systèmes dont le produit, le logiciel, est un sous-système d'un système majeur. Ce système majeur, dans lequel le logiciel va fonctionner, défini ce qu'on appelle l'**Univers du Système** ou encore le **Contexte du Système.**

L'Univers du Système, c'est le contexte général dans lequel le logiciel devra être développé et il inclut toutes les sources d'information et toutes les personnes en relation avec le domaine du problème (on les appelle **les acteurs** de l'univers). Cependant, avoir l'Univers du Système délimité n'est pas équivalent à avoir les informations nécessaires disponibles parce que l'Univers délimite seulement les sources pour obtenir les informations. Il faut l'étudier et identifier à la fois ses propriétés et ses caractéristiques.

Le développement avec succès de systèmes grands et complexes nécessite une connaissance approfondie et spécifique de l'univers du système. En fait, le manque de connaissance - y compris sur le contexte du problème - est considéré comme le problème le plus significatif dans le développement industriel du logiciel, comme l'étude conduite par Bill Curtis [Curtis et al. 88] le montre.

L'insuffisance d'informations recueillies pendant l'étape de détermination est lié évidemment à l'insuffisance des modèles (et des formalismes associés) utilisés pour la décrire. En fait, MFBarthet établit un lien à double sens entre l'insuffisance de l'analyse de l'existant dans les méthodes informatiques et l'insuffisance du modèle et des formalismes associés pour décrire cet existant [Barthet 88]. Le pouvoir limité d'expression des modèles et formalismes informatiques incite inconsciemment l'informaticien à ne recueillir que ce qu'il peut représenter. Traditionnellement, les informations collectées par les modèles informatiques concernent des fonctions (comme en SSA, SADT, Analyse Structurée Moderne, par exemple), des données (comme en ER - *Entity-Relationship* -, JSD, par exemple), des objets (comme en OOA, OMT, Booch par exemple) ou des séquences d'états (comme en *statecharts*, Réseau de Petri pas exemple) qui sont surtout orientés à la dimension interne du système à être construit. Ces méthodes informatiques traditionnelles suivent, parfois implicitement, un paradigme traditionnel pour le développement de logiciel, appelé paradigme orienté produit [Floyd 88], dont la caractéristique principale est l'orientation système.

L'utilisateur constitue le principal rôle humain attaché au système interactif. Mais malheureusement, la vision du monde qui détermine les termes dans lesquels vont être determinée les caractéristiques et propriétés de l'Univers est celle du concepteur du système. Une information plus précise sur le point de vue de l'utilisateur doit représenter les connaissances sur le contexte d'usage du système, l'utilisateur lui-même ainsi que les tâches qu'il effectue.

Dans le cadre de systèmes interactifs, l'intérêt d'une bonne compréhension de l'univers, où le système va être développé et utilisé, s'affirme progressivement. Les diverses noms présents dans la littérature pour cette compréhension sont un reflet de la diversité de approches pour le faire. Nous l'appelons Analyse Contextuelle (voir chapitres 4 et 5) parce que l'univers est en fait le contexte de travail à soutenir. L'ensemble des informations recueillies et représentées sur l'univers est appelée Information Contextuelle. Bien que plusieurs auteurs [Cockton 95], [Dowell & Long

89], [Wilson & P. Johnson 96] aient montré l'importance cruciale de cette information, peu de modèles et approches pour l'analyse contextuelle sont disponibles. La diversité de types d'information considérés nécessaires, la complexité de leur modélisation et même la confusion sur la signification de 'contexte' sont bien sur à la source de ce manque de modèles.

5.2.2.2Problemes de communication entre utilisateurs-analyste

Il est nécessaire d'établir la correspondance entre le point de vue des concepteurs (y compris celui des ergonomes et des informaticiens) et le point de vue des utilisateurs afin de pouvoir intégrer leurs connaissances lors de la conception de logiciels interactifs. Cette correspondance est faite par la communication analyste-utilisateur, toujours considéré comme difficile par ces rôles. Cette difficulté est due au fait que chacun a :

- une vision très particuliere du problème, du système existant (s'il y en a) et du système informatique futur; cette vision est due aux différentes expériences (*background*) et points de vue de chacun ;

- un vocabulaire particulier, où un même mot peut avoir différentes significations ou un même concept peut avoir différents noms pour différentes personnes.

Plusieurs solutions à ce problème cherchent à éliminer cette communication directe entre analyste-concepteur. Par exemple, l'approche de l'analyse du discours - pour capturer les informations en analysant les expressions écrites des besoins faites par les utilisateurs en langage naturel, à travers les techniques de l'intelligence artificielle [Iris 92] [Biébow 92] [Dankel 92]. Cette approche pourtant n'est plus utilisée hors de domaines très spécifiques ou des cas d'école.

Une autre approche dans cette ligne est la communication à travers un traducteur, qui est "quelqu'un qui parle le langage de l'utilisateur et le langage de l'analyste et qui sait comment leurs responsabilités de travail sont affectées par le projet" [Williams 93]. Le traducteur n'est ni un utilisateur ni un concepteur et sa tâche n'est pas réduite à une traduction terminologique mais inclut aussi l'interprétation des activités de chacun, de la culture de travail et la proposition d'une politique d'aide à la communication et à la négociation. Le problème de cette approche, centrée dans le rôle du traducteur, est justement de trouver quelqu'un qui ait les compétences pour ce rôle.

La plupart des approches existantes, qui utilisent des traducteurs (humains ou automatiques) comme intermédiaires entre utilisateurs et analystes, se sont révélés inefficaces probablement à cause de l'impossibilité d'étudier le langage de travail séparément de son contexte non-symbolique ce qui a été démontré dans [Andersen 90].

Ces approches, comme la plupart des approches traditionnelles existantes pour l'Analyse des Besoins, considèrent les utilisateurs comme des 'sources' passives d'information et leur participation seulement comme une manière de recueillir cette information.

Cependant, il y a déjà une acceptation d'une perspective qui permet une participation plus active et effective des utilisateurs dans le développement des SI [Grudin 91]. En fait, au lieu d'éliminer la communication directe, le but est de soutenir cette communication. Une façon de réduire les problèmes de communication directe est l'utilisation du langage naturel. Cependant, cela peut créer des difficultés pour l'utilisation des outils et introduire d'autres problèmes comme l'ambiguïté [Christel 92]. Pour mieux comprendre le vocabulaire très particulier du domaine, une alternative est

d'établir une **unification de terminologie**. Deux exemples d'unification de terminologie sont les expressions reformulées des tâches de l'utilisateur proposées par [Aubin et al. 94] - voir dans le Figure 2.2 un extrait de taxonomie des tâches - et l'ontologie proposée pour les entreprises virtuelles [O'Leary et al. 97].

Opérateur de Tâche	Définition
Discriminate	*Examine two or more objects in order to discover differences*
Detect	*Discover presence of an object or a property*
Calculate	*Perform a mathematical operation on two or more objects*
Select	*Choose from among a number of objects*
Integrate	*Synthesyse several objects into a functioning whole*

Figure 2.2 - Opérateurs de tâches extraits de la taxonomie de [Aubin et al.94]

5.2.2.3 Problèmes de gestion de changement (ou de l'évolution des besoins)

Dans le cycle de vie en cascade, un changement des besoins fait partie de la maintenance s'il arrive après la livraison du système. Cependant, les besoins peuvent changer même **durant** le développement et cela est en fait très fréquent, typiquement due à deux raisons : a) les besoins de l'utilisateur ont changé depuis sa documentation et b) une révision des besoins est faite à partir des réactions aux premières versions des besoins.

En effet, les besoins évoluent au cours du cycle de vie du système. Ils changent typiquement :

- de nature - du général au spécifique ;
- d'envergure - par l'incorporation ou la suppression de caractéristiques ;
- de contenu et de forme, pour devenir plus consistants, précis et clairs.

Les besoins ont des relations entre eux et des connections avec d'autres artefacts du développement (spécification, code, documentation). Ces relations et connections évoluent aussi dans le temps. Alors la définition des besoins doit aussi être capable d'évoluer itérativement durant le changement de manière à ce que les rapports entre les besoins, la spécification et le code soient maintenus [REAW 91]. Ainsi les modifications du code, de la spécification ou d'autres besoins sont réfléchies dans les besoins. 'Tracer' ces connections à partir des besoins (*forward tracing*) ou à partir des artefacts (*backward tracing*) est une propriété très utile.

Cette propriété est appelé traçabilité des besoins (*requirements traceability*) et elle est fondamentale pour la gestion des changements des besoins [Gotel et Filkenstein94a]. Par définition, la traçabilité des besoins est définie comme :

'...the ability to describe and follow the life of a requirement in both a forwards and backwards direction (i.e. from its origins, through its développement and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases.' [Gotel & Filkenstein 94b]

Bien que l'intérêt souvent donné comme le plus important pour la traçabilité soit le

soutien à l'activité de gestion des changements [Pinheiro 96], la traçabilité permet aussi la vérification des caractéristiques du système en rapport aux besoins, l'identification de sources d'erreurs et la documentation des contributions de chaque individu (ou groupe) pour l'établissement des besoins - appelée aussi traçabilité pré-spécification (*pre-requirements-specification traceability*) [Gotel & Filkenstein 94b].

5.2.2.4Les Illusions de l'Ingénierie des Besoins

Comme nous l'avons vu, les efforts de recherche en IB incluent plusieurs vrais problèmes 'réels'. En addition à eux, un rapport d'un groupe de travail du 4th IEEE Workshop on Software Specification and Design [Arango 88] suggèrent deux illusions associés à l'IB :

- '**Illusion de la complétude**': comment savoir si la spécification est complète?
- '**Illusion de la page blanche**': le début du processus commence-t-il à partir de zéro?

Ces illusions servent à reconnaître deux faits :

- la nécessité d'avoir des analyses incrémentales et pouvoir manipuler des connaissances partielles pendant le processus de l'IB ;
- il faut réutiliser les informations (contextuelles, besoins, architectures, etc) déjà organisées.

6.L'Ingénierie des Besoins des Systèmes Classiques

Dans cette section, un ensemble d'aspects qui caractérisent l'ingénierie des besoins des systèmes classiques est présenté : la définition de besoins, les activités de l'IB, les techniques de recueil existantes, et les résultats souhaités. Ces concepts sont fortement basés sur la littérature du GL et ont été établis comme valables généralement pour la plupart des classes de systèmes classiques.

6.1Les besoins et les facteurs de qualité

Pour les systèmes classiques, l'IB est traditionnellement conduite par les besoins fonctionnels.

Malgré l'importance reconnue des besoins non fonctionnels et des facteurs de qualité (voir [Boehm 78] et [Lindstrom 93]), les concepteurs n'ont pas une manière systématique de les prendre en compte pendant le développement du système. En pratique, ces aspects sont considérés à la fin du développement ou considérés en parallèle mais séparés de la conception de la fonctionnalité - et en général, les décisions de conception finissent pour prioriser les besoins fonctionnels.

Une raison pour cela est le manque de recommandations ou heuristiques pour bien appliquer ces concepts de qualité. Les définitions de 'sécurité', 'précision', 'performance' sont très génériques (voir par exemple les définitions de [McCall 77] ou [Boehm 78]) et difficiles à exprimer comme un but pour un produit spécifique. Une deuxième raison est le fait que ces concepts impliquent la recherche d'un compromis : chaque décision pour l'adoption d'un facteur peut affecter les autres facteurs. Cette décision est en général prise par le concepteur, qui ne possède pas une aide

systématique pour identifier les interactions entre les besoins fonctionnels et les facteurs de qualité et pour évaluer les alternatives de compromis.

En effet, un début de systématisation apparaît déjà dans quelques travaux qui visent à utiliser les besoins et les facteurs de qualité ensemble durant la conception, comme les approches décrites dans [Chung 94] et [Zultner 92b].

6.2Les activités

Dans le Génie Logiciel, l'IB était traditionnellement perçue comme une activité vague où les idées informelles de l'utilisateur sont formalisées par l'analyste sans aucune méthode ou systématisation. Ainsi, toute la connaissance avant la formalisation tendait à être perdue : théoriquement, due à l'incapacité d'intégration des informations informelles dans la spécification formelle, et pratiquement, parce que cette connaissance était dispersée dans des documentations volumineuses, inutilisables et non-maintenables [Jarke et all. 93].

Aujourd'hui, plusieurs auteurs pensent que l'IB a son propre cycle de vie et quelques phases sont toujours identifiées même avec des noms différents. Selon Herb Krasner, les phases sont au nombre de 5 : identification des besoins (*needs*) et analyse du problème ; détermination des exigences (*requirements*) ; spécification des exigences ; 'remplissage' des exigences ; et gestion des changement de ces exigences [Krasner 88]. Selon Willian Rzepka, les phases sont au nombre de 3 : '*elicitation*', vérification de consistance et validation [Rzepka 89].

Ces cycles adoptent en général une idée séquentielle des activités. En réalité, pourtant, le processus d'IB est inévitablement itératif et ces activités sont re-visitées plusieurs fois [Southwell 87], [Siddiqi 96]. En fait, comme le dit Joseph Goguen [Goguen 92] :

6)

'...the belief that the steps of a life cycle should be executed sequentially is a crude form ot the myth that there is more or less a unique best system to be built.'

7)

Des exemples de cycles de vie itératifs sont le modèle de '*Inquiry Cycle*' [Potts 94], qui est basé sur l'itération de 3 activités - documentation, discussion et évolution ; et le modèle de Julio Leite, qui propose l'entrelacement de l'*elicitation* et la modélisation [Leite 87]. Le modèle itératif classique de l'IB cependant est le cycle de détermination, expression et validation proposé par Mathias Jarke et Klaus Pohl [Jarke 93], que nous détaillons ci-dessous.

La **Détermination** est le processus itératif d'obtention et d'élucidation des informations, c'est à dire, qu'elle concerne l'identification des sources d'information dans l'univers du discours, la collecte d'informations et le raffinement et l'intégration des informations relevantes. Selon les différentes sources, il peut y avoir différents points de vue sur le même ensemble d'informations. Pendant l'activité de Détermination, toutes les informations sont représentées de manière informelle.

L'**Expression** (aussi appelée Modélisation) concerne la construction incrémentale d'un modèle conceptuel plus formel à partir des informations obtenues dans la Détermination, en utilisant, par exemple, des diagrammes DFD (*Data Flow Diagrams*), ERA (*Entity Relationship Attribut*), les formalismes de Merise ou quelques autres inhérents à la méthode de développement utilisée dans l'entreprise. Le produit de l'IB est, alors, un modèle, a partir du quel un document des besoins du système (Cahier des Charges) est produit. Les deux (le modèle des besoins et le Cahier des Charges) sont les bases sur lesquels le système informatique est développé.

La **Validation** est le processus d'analyse de l'information recueillie et modélisée pour vérifier si elle est un reflet précis de la réalité. La notion de validation de logiciel du Génie Logiciel - c'est à dire, la confirmation que le produit est celui souhaité par l'utilisateur - arrive normalement à la fin du cycle de vie dans une phase appelée Test. En Ergonomie, la notion d'évaluation d'un système consiste à estimer la conformité des performances effectives avec les performances désirées du système [Dowell & Long 89]. Dans le cas de l'IB, la validation doit être faite bien sûr avant la réalisation du produit et avant même la spécification du système.

Il y plusieurs techniques de validation des besoins [Leite 88], comme par exemple :

1. Evaluation Informelle ;

2. Prototypage ;

3. Vérification Formelle ;

4. Réutilisation de domaines

Toutes ces techniques sont capables d'identifier une différence (Delta **Δ**) entre les information recueillies et l'univers d'informations. Les différences entre ces techniques sont déterminées par le type d'information et les procédures utilisées pour l'identification des Deltas.

1. L'évaluation informelle (appelé aussi *Walkthrough*).est, de manière simplifiée, une tâche de lecture des descriptions des besoins et l'utilisation d'une '*checklist*' pour détecter les problèmes d'expression des besoins. Le résultat est alors une liste de problèmes entre l'information recueillie et les faits :

Δ = liste de problèmes (information, faits)

2. Le prototypage est basé sur l'évaluation par l'utilisateur du comportement d'un prototype pour valider les besoins en rapport à ses expectatives. Le résultat est alors une liste de différences de comportements :

Δ = comportement des faits (expectatives de l'utilisateur, faits)

3. La Vérification Formelle exige une description formelle des besoins pour l'identification (souvent automatique ou assistée) de incohérences comme ambiguïtés, contradictions et incomplétude de définitions. Le résultat est exprimé par l'ensemble de incohérences :

Δ = Incohérences (faits, information**)**

4. La réutilisation du domaine exige une description préalable des faits sur un domaine et la validation est basée sur la comparaison entre les besoins recueillis. Le résultat est l'union de l'ensemble de faits incorrects et des faits manquants :

Δ = faits incorrects (faits du domaine, faits) ∪
faits manquants (faits du domaine, faits)

Les trois sous-procédés ci-dessus (Détermination, Expression et Validation) ne forment pas une séquence de pas successifs, c'est à dire, un pas ne doit pas être terminé pour commencer l'autre. En vérité, on fait les trois de manière intégrée et conjointe. L'information recueillie est plus facilement organisée et comprise à travers les modèles la représentant. Un modèle peut servir de guide pour structurer les informations à recueillir. Un modèle est aussi une aide fondamentale à la validation : un modèle clair et complet permet une communication plus facile et l'obtention plus facile d'un consensus entre les acteurs. La validation peut exiger une augmentation de la compréhension sur le domaine du problème ce qui se traduit par un recueil plus important ou par une modélisation plus adéquate.

Une autre définition du processus est proposée par [Davis 93], basée sur **le type d'information** que l'on veut manipuler. Il fait alors la distinction entre deux types d'activités : **analyse du problème** et **description du produit**. Pendant l'analyse du problème, l'objectif est d'obtenir le plus de connaissances possible sur le problème et son contexte, y compris l'identification des besoins de l'utilisateur. Il est nécessaire de collecter et de modéliser plus d'informations sur les utilisateurs et leur contexte, comme, par exemple, leurs buts, la façon de les réaliser, leur rapport avec les activités, les acteurs et les processus de travail de l'organisation, les objets qui représentent l'information du domaine, la plateforme disponible (matériel et logiciel) et le vocabulaire du domaine. Dans cette thèse, cette information est appelée **Information Contextuelle** et elle est fondamentale pour la vraie compréhension du domaine du problème. Pendant la description du produit, l'objectif est la construction de la spécification du système, qui inclut la spécification des besoins fonctionnels et non fonctionnels. Plusieurs auteurs décrivent le produit au niveau de l'IB par un ensemble de modèles comme le modèle architectural, le modèle des objets de l'application, le modèle de dialogue et le modèle de présentation.

En fait, ces deux types d'information - l'information du contexte et la description du produit - correspondent aux notions d'informations contextuelles et d'informations du système présentés dans le chapitre I. Ces deux types d'information sont aussi présentes sous différentes formes par d'autres travaux d'IB. Par exemple, Michael Jackson utilise respectivement les notions de 'l'information de l'espace problème' et 'l'information de l'espace solution' [Jackson 95] lorsque [REAW 91] utilise les termes 'contexte du problème' et 'caractéristiques du système'.

Malgré l'intérêt croissant pour l'IB, on observe que la plupart des travaux et des méthodes de la littérature dirigent leur attention vers le problème de l'expression, créant plusieurs modèles différents conformément aux informations du domaine du problème et du système qu'on veut souligner. Traditionnellement, les informations collectées concernent des fonctions, des données, objets ou des séquences d'états qui sont importants pour le contrôle du problème et qui doivent être modélisées d'après l'application (dimension interne) du système à construire.

Cependant, la recherche de modèles reflétant cette dimension interne oublie souvent l'aspect social du modèle. Un modèle sert à capturer l'essence de l'information de façon à ce que les personnes (analystes, utilisateurs) puissent comprendre ce qu'il représente : il est aussi bien un véhicule de communication qu'une base pour la conception [Olsen 95].

6.3Les techniques de recueil

Le recueil est le début du processus graduel de construction de la connaissance pour développer et utiliser un système. En effet, selon [Naur 85], un programme reflète une théorie construite par les développeurs à travers une croissante et évolutive compréhension du domaine du problème.

Dans la plupart des cas, le recueil est traditionnellement vu comme une activité vague où les idées informelles des utilisateurs doivent être comprises et organisées par l'analyste. Dans ce cas là, l'expérience de l'analyste dans le domaine du problème devient fondamentale pour cette facilité de compréhension. Pour l'aider à obtenir les informations, il peut disposer de quelques techniques de recueil, qui peuvent être classées en 3 catégories principales :

1. **Techniques Basées sur la Communication (TBC)**, sous-divisées en :

- **Communication Indirecte** ou écrite, qui inclut l'utilisation de questionnaires fournis par l'analyste et remplis par l'utilisateur ou encore l'utilisation de textes de besoins informels rédigés par l'utilisateur;
- **Communication Directe** ou orale, qui inclut entre autres des dialogues à diverses degrés de formalité, des interviews (dialogue formel avec sujet, date et durée prédéterminées) jusqu'à des conversations informelles (en général, dialogues en dehors du lieu et des horaires de travail);

2. **Techniques Basées sur l'Etude (TBE)**, qui inclut le recueil d'informations à partir de documents ou formulaires utilisés dans l'environnement de travail ou à partir de la littérature sur le sujet ou encore à partir de l'analyse de systèmes informatiques existants (Ingénierie Reverse - en anglais *Reverse Engineering*) soit dans l'entreprise elle-même soit dans autres organisations;

3. **Techniques Basées sur l'Observation (TBO)**, qui inclut entre autres:

   - l'immersion (réalisation de tâches réelles de travail par l'analyste),
   - l'observation directe (par une personne ou au moyen d'un enregistrement vidéo) et ses variations (comme l'observation assistée et Ethnographie) ;
   - l'observation verbalisée (aussi appelée protocole concurrent), où l'utilisateur réalise ses tâches en présence d'un observateur et en expliquant ce qu'il est en train de faire),
   - observation suivie de dialogue (aussi appelée protocole rétrospectif), où l'utilisateur est invité à générer a posteriori un rapport verbal sur la réalisation des activités de travail.

La plupart des techniques de recueil mentionnées sont des techniques standards utilisées par les informaticiens ou les ergonomes. Le lecteur intéressé peut trouver plus de détails sur ces techniques dans [P. Johnson 94], [Barthe 95] ou [Welbank 83].

Dans le domaine du GL, la gamme des techniques utilisées est en pratique très pauvre, se limitant à une ou deux techniques. Les techniques le plus utilisées sont surtout les techniques basées sur la communication (TBC), mais aussi les techniques basées sur l'étude (TBE). En fait, la formation de base des informaticiens est insuffisante pour maîtriser les techniques d'observation. La mise en oeuvre des TBC et TBE est généralement entachée de vices méthodologiques, comme par exemple

- la liste des questions préparée pour les interviews ne remplace pas une grille d'analyse bien structurée ; et

- l'échantillon sélectionné (en général, les cadres supérieurs et intermédiaires) est rarement représentatif de la population concernée parce que l'opérateur de base est souvent négligé.

Cependant le problème le plus important et spécifique à la pratique du GL est que **la distinction entre le travail prescrit et le travail réel n'est pas repérée** : soit parce que l'opérateur n'est pas une source d'information totalement fiable dû à la déformation qu'il se fait de son propre travail par rapport à la réalité soit parce que le travail réel laisse moins de traces que les supports formels (documents, etc) du travail prescrit.

Dans le domaine de l'Ergonomie, les TBO sont les techniques les plus utilisées dans les activités de recueil. Bien que les TBO soient coûteuses en temps de dépouillement des données, elles sont nécessaires pour obtenir une description fiable de comment l'utilisateur organise son travail, quelles opérations il fait et leur ordre, comment il adapte l'exécution prévue malgré les difficultés qui surgissent et quels sont les

automatismes développés. En fait, les procédures effectives du travail réel sont souvent recueillies par les TBO alors que les TBC et TBE sont adéquats pour recueillir les procédures prévues du travail prescrit [Barthet 88]. Le travail prescrit est la manière officielle de faire les choses alors que le travail effectif est ce qui se fait réellement.

6.4Les résultats attendus

Les résultats typiquement souhaités de l'IB sont :

1. un ensemble de modèles pour représenter les informations sur le contexte et le système envisagé ;

2. une spécification des besoins sous la forme d'un ensemble de :

    - définitions fonctionnelles du système, y compris l'ensemble d'informations (sa partie statique) et l'ensemble des services (sa partie dynamique) ;

    - définitions non-fonctionnelles du système, y compris les facteurs de qualité qui servent comme ressources et comme contraintes à considérer pour le développement du système.

3. la documentation des processus de :

    - détermination des besoins;

    - intégration entre les besoins et les modèles du système;

    - décision des alternatives de solution ('*design rationale*').

6.5La participation de l'utilisateur

Une hypothèse de travail des chercheurs de l'IB est que la participation[7] de l'utilisateur est nécessaire pour le succès du processus. [Mumford 84]. Le succès de l'IB est identifié comme un concept multidimensionnel où les deux plus importantes dimensions sont la qualité du service de l'IB et la qualité des produits de l'IB [El Emam 95]. En fait, la participation de l'utilisateur peut réduire l'incertitude dans le processus d'IB et par conséquence augmenter la qualité des besoins résultants et la probabilité d'acceptation du système [El Emam 96].

Toutes les approches de l'IB reconnaissent que la participation de l'utilisateur est très importante, qu'elle soit prise en compte de façon explicite ou implicite. Cependant, le dégré de cette participation est très varié. [Macaulay 95], par exemple, défini trois types de participation de l'utilisateur (et de l'analyste):

- participation passive, typique des approches centrées analyste (*design from user*) ;

7 Le terme 'participation' est préféré délibérément au contraire de '*involvement*', conformément aux définitions de [El Emam 96]. La participation est définie comme les comportements et activités que l'utilisateur réalise pendant la conception. *Involvement* est défini comme l'état psychologique et subjectif des utilisateurs face à un système particulier.

- participation active en coopération avec l'analyste ;

- participation active en coopération dans un groupe multidisciplinaire.

L'idée de participation active des utilisateurs est chaque fois plus présente dans la communauté IHM, qui a une préférence pour le terme Conception Participative (*Participatory Design*), qui consiste en l'exécution de quelques activités conjointement entre concepteurs et utilisateurs.

Avec cette approche, les utilisateurs et les concepteurs ont des compétences complémentaires mais d'importance équivalente (sans hiérarchie) pour comprendre l'Univers d'informations, pour identifier les réels besoins des utilisateurs et pour prendre des décisions sur la conception du système. La participation conjointe pendant le processus de développement du système est une condition nécessaire pour créer de nouvelles connaissances requises (on peut aussi dire "construire la théorie du système", selon [Naur 85]) et pour une expérience partagée de la conception. Soutenir la coopération entre les concepteurs et les utilisateurs signifie faciliter le processus de communication entre eux, en visant un apprentissage mutuel (le terme anglais utilisé est "*mutual learning*").

Nous ajoutons encore à ces types deux autres formes de participation :

- la définition des besoins réalisée totalement par l'utilisateur, typique des systèmes définis par contrat (voir section 3.4.2 au chapitre I) où l'organisation de l'utilisateur spécifie les besoins devant être respectés par l'organisation responsable par le développement ; des exemples typiques sont les logiciels commandés par le gouvernement des Etats Unis [Grudin 91] ;

- la participation hybride, résultant de la combinaison des autres formes de participation décrites précédemment;

7.Ingenierie des Besoins des Systèmes Interactifs

L'Ingénierie des Besoins des Systèmes Interactifs demande l'utilisation conjointe de concepts et techniques spécifiques à l'interface homme-machine et de concepts et méthodes de développement de systèmes - traditionnellement considérés comme faisant partie du Génie Logiciel.

C'est un fait que les besoins déterminés par des mÈthodologies du GL, orientées surtout vers les besoins fonctionnels, sont insuffisants pour concevoir les aspects de l'interaction homme-machine. En outre, l'IHM se concentre en général sur l'interaction sans une considération adéquate des autres besoins du logiciel observés par le GL. Cette distinction est à la base d'un problème d'intégration interdisciplinaire, qui continue tout au long du cycle de vie du logiciel interactif. La conséquence la plus imminente de ce problème est le *fractionnement des besoins***:** parce que chaque domaine ne considère que quelques aspects différents et souvent disjoints du système, les besoins du GL sont décrits en documents ou sections différents des besoins de l'IHM, sans une manière systématique pour faire une correspondance explicite entre eux.

Dans les domaines du GL et de l'IHM, il y a encore une tendance à traiter le développement des systèmes interactifs comme soit un 'cas spécial' du développement de logiciel soit l'addition de plus de fonctionnalités à une interface utilisateur. Cependant, nous sommes d'accord avec plusieurs auteurs qui pensent qu'il faut considérer les systèmes interactifs comme un sujet en soi. Ce point de vue est à la base des approches d'intégration IHM-GL pour la conception de systèmes interactifs,

présentées au chapitre 3.

Il est naturel alors que l'IB des systèmes interactifs ait ses propres concepts. Les concepts que nous avons vus avant étaient liés aux systèmes classiques. Maintenant, nous les considérons sur l'optique spécifique des systèmes interactifs. Donc dans cette section nous présentons :

- une définition des besoins et facteurs de qualité pour les systèmes interactifs ;
- des techniques de recueil pour les systèmes interactifs ;
- les activités de l'IB des systèmes interactifs;
- la participation de l'utilisateur ;
- les résultats souhaités de l'IB des systèmes interactifs.

7.1Les besoins et les facteurs de Qualité

Dans le domaine du GL, les facteurs de qualité sont associés surtout à la qualité du produit et à la qualité de son processus de développement. L'utilisabilité n'est qu'un des facteurs de qualité du système. Dans le domaine de l'IHM les facteurs de qualité sont traditionnellement divisés en facteurs associés à l'utilité du produit et facteurs associés à l'utilisabilité du produit.

Comme nous l'avons vu au chapitre I, pour les systèmes interactifs, la notion d'utilisabilité est aussi importante que la notion d'utilité, qui a longtemps été la plus importante pour les systèmes informatiques traditionnels. L'utilité est la mesure du rapport entre les fonctions du système et les tâches de l'utilisateur. En pratique, l'utilité est exprimé par le bon fonctionnement (*correctness*), la fiabilité, l'efficacité et l'intégrité du système. L'utilisabilité est la mesure du rapport entre la manière à travers laquelle une tâche interactive est réalisée par un utilisateur particulier et le profil cognitif de cet utilisateur [Farenc 95]. L'utilisabilité est un facteur critique pour la satisfaction et la productivité des utilisateurs et elle est perçue à travers l'interface utilisateur du système.

Pour mieux réfléchir à l'importance de l'utilisabilité pour les systèmes interactifs, il faut que les facteurs de qualité d'utilisabilité soient considérés aussi pour la spécification, cf. Figure 2.3.

Facteurs de qualité	Besoins	Contribution de (discipline)
Qualité du système comme un produit (utilité)	Besoins Fonctionnels du système	Génie Logiciel (GL)
Qualité du processus de développement du système	Besoins non fonctionnels du système	Génie Logiciel (GL)
Qualité de l'interaction utilisateur système (utilisabilité)	Besoins Comportementaux	Interaction Homme-Machine (IHM)
But : Développement d'un système à la fois utile et utilisable	But : Intégration des besoins comportementales, fonctionnels et non fonctionnels	But : Intégration de théories, modèles, méthodes et outils du GL et de l'IHM

Figure 2.3 - Facteurs de qualité et Besoins de systèmes interactifs

Les catégories classiques de besoins fonctionnels et non-fonctionnels typiques du GL (voir section 3 de ce chapitre) ont une caractéristique en commun : la priorité est préférentiellement d'identifier d'abord les besoins fonctionnels puis identifier après les

besoins non-fonctionnels. En fait, cela suit une tradition en Ingénierie qui énumère les besoins du produit distingués des besoins du processus de son développement. Cette attention sur les besoins fonctionnels rend cette classification inadéquate pour les systèmes interactifs et encourage les concepteurs à ne pas accorder l'importance nécessaire aux besoins comportementaux de l'interaction avec les utilisateurs. Le guide IEEE/ANSI [IEEE 84] pour la spécification de besoins de logiciel en est un exemple.

[ONeill 95],en démontrant l'intérêt avec les besoins comportementaux, a proposé que les besoins de l'interaction soient considérés comme des besoins du système, voir figure 2.4.

8)

Besoins du système	**Besoins du dévelopment**	**Besoins commerciaux**
Besoins de la Tâche	Besoins de Portabilité	Besoins Contractuels
Besoins de l'Interaction	Besoins de Modifiabilité	Besoins de Marketing
Besoins de l'Interface	Besoins de Extensibilité	
Besoins du Contexte	Besoins de Comprehensibilité`	
Besoins de Performance	Besoins de Conception	
	Besoins de l'Implementation	

Figure 2.4 - Classification des besoins d'après [ONeill 95]

Il y a d'autres classifications qui considèrent les besoins comportementaux comme besoins du système. Nous pensons que cela n'est pas adéquat pour les systèmes interactifs. Comme nous avons montré dans le chapitre I, l'interaction n'est pas une partie du système mais une **relation entre les utilisateurs et le système**. Donc il faut considérer les besoins comportementaux **séparément** des autres besoins (fonctionnels et non fonctionnels) du système, cf. Figure 2.3. Cependant, ces besoins sont bien sûr très fortement liés aux composants (sémantiques et de dialogue) du système qui les réalisent. La séparation proposé est donc physique et non logique.

Nous avons proposé pour TAREFA une nouvelle taxonomie pour les besoins des systèmes interactifs, présenté au chapitre 4.

7.2Les activités

Aujourd'hui, il est reconnu que, pour concevoir un système interactif on doit considérer non seulement le contexte du problème de l'utilisateur et la fonctionnalité du système, mais aussi comment la conception de ce système va correspondre aux besoins de l'utilisateur dans la situation d'utilisation. Ainsi il faut considérer le contexte d'utilisation du système, c'est à dire, les processus de travail et de communication des utilisateurs et la manière avec laquelle ils sont affectés par le système pour qu'on puisse les prendre en compte le mieux possible avec l'interface utilisateur (dimension externe) du système. Pour ce faire, il faut une **idée envisagée de l'utilisation du système**, qui nous appelons **projection d'usage**.

En effet, le cycle determination-expression-validation de Jarke et Pohl (voir section 3.2 de ce chapitre) suit l'idée de définir l'IB comme un processus d'établissement des vues en contexte (*visions in context*) [Jarke & Pohl 93]. Le contexte est divisé en trois mondes : sujet, usage et système. Le **sujet** représente la partie du monde externe au **système** - ce système étant représenté par une description structurelle - dans laquelle le système existe pour servir à un **usage** déterminé par des buts d'individus ou d'organisations.

A niveau de l'IB chaque type d'information est considéré d'une façon particulière :

- le sujet est représenté par l'information contextuelle obtenue par l'analyse de l'existant et du problème ;

- l'usage est représenté par la projection d'usage futur;
- le système est représenté par la description des besoins du produit.

	---------- Sous-procédés ----------		
Types d'information	Détermination	Expression	Validation
Information Contextuelle (sujet)	techniques de recueil d'information sur le contexte	représentation de l'information sur le contexte	Validation de l'information sur le contexte
Projection de l'Usage du Système (usage)	techniques de projection	représentation de projection	validation des situations envisagées
Besoins du Système (système)	techniques de détermination des besoins du système	représentation des besoins du système	validation des besoins du système

Figure 2.5 - Activités de l'Ingénierie des Besoins

Nous adoptons ici l'IB des systèmes interactifs comme l'ensemble d'activités qui sont au croisement des types d'information de contexte (les mondes sujet, usage et système) et les trois sous-procédés classiques : donc pour chaque type d'information, il y a le cycle de 3 phases (détermination, expression, validation), conformément Figure 2.5.

7.3Les techniques de recueil

La section 5 a présenté une classification des techniques de recueil utilisées par les informaticiens et les ergonomes. Evidemment, chaque technique de recueil dépend du type de population cible, du produit, de la nature du problème à résoudre et également des ressources disponibles.

En effet, pour les systèmes interactifs, le recueil d'information sur l'univers doit intégrer les techniques utilisées par les informaticiens et les ergonomes. Ces techniques doivent être utilisés de manière complémentaire - au minimum un ensemble avec une technique de chaque catégorie TBC, TBE, TBO - pour permettre une richesse et une meilleure vérification des informations recueillies.

-	**Etude de formulaires et manuels (TBE)** **Etude de Systèmes Existants (TBE)** **Interview Structurée (TBC)** **Questionnaire (TBC)** **Observation Directe (TBO)** **Observation Verbalisée (TBO)** **Observation Suivi de Dialogue (TBO)**
+	**Immersion (TBO)**

Figure 2.6 - Echelle Croissante de Cout d'utilisation des techniques de recueil

Pour le choix des techniques, on adopte une heuristique proposé par P. Johnson : utiliser autant que possible les techniques moins coûteuses, selon l'échelle croissante de cout [P. Johnson et al. 88], montrée dans la Figure 2.6.

Le coût ici fait référence à la fois au temps nécessaire pour la réalisation du recueil (le temps de l'analyste et bien sur le temps de la population de l'échantillon) et aux équipements nécessaires (vidéo, salle spéciale, etc). Alors, l'ensemble préférentiel au niveau du coût serait l'Etude de Formulaires et manuels (TBE), les interviews

structurées (TBC) et l'Observation Directe (TBO) qui sont les trois qui couvrent les trois catégories et qui sont les moins coûteuses.

7.4Les résultats attendus

Les résultats souhaités pour l'IB des systèmes interactifs d'une certaine manière sont similaires aux résultats de l'IB pour les systèmes traditionnels. A la fin du processus il y a:

1. un ensemble de modèles pour représenter les informations recueillies sur le contexte,

2. une spécification des besoins dans la forme d'un ensemble de :

    - définitions fonctionnelles du système, y compris l'ensemble d'informations (sa partie statique) et l'ensemble des services (sa partie dynamique) ;

    - définitions non-fonctionnelles du système, y compris les facteurs de qualité qui servent comme ressources et comme contraintes à considérer pour le développement du système ; et

    - **définitions comportementales du système**, y compris la projection d'usage et les besoins des interactions entre le système et ses utilisateurs ;

3. la documentation des processus de:

    - détermination des besoins;

    - intégration entre les besoins et les modèles du système;

    - décision des alternatives de solution ('*design rationale*')

La **différence principale** au niveau des résultats est qu'il y a en plus la spécification de besoins de comportement et la prise en compte des facteurs d'utilisabilité comme facteurs de qualité.

7.5La participation de l'utilisateur

La participation de l'utilisateur dans l'IB des systèmes interactifs est typiquement hybride. Plus spécifiquement la participation de l'utilisateur peut appartenir, en fonction du type de l'activité réalisée, à un des 3 premiers types cités à la section 8, à savoir:

- passive (centrée analyste) quand il faut réaliser les activités méthodologiques de l'IB;

- active (coopération avec l'analyste), quand il faut réaliser des activités centrés utilisateur ou activités centrés utilisabilité, comme par exemple analyse de la tâche;

- active (coopération entre tous les *acteurs*) pour les activités émergentes de l'IB où

la négociation entre les participants est nécessaire.

Chapitre III: Approches pour l'Ingénierie des Besoins de Logiciel Interactif

---

L'objectif de ce chapitre est de présenter certains approches significatifs pour l'IB dans les domaines de GL et IHM, de manière séparée et les tentatives d'intégrations des deux domaines. Cette présentation est une tentative de comprendre les idiosyncrasies de chaque domaine et les avantages et inconvénients de chaque approche en particulier.

Comme nous avons vue au chapitre 2, l'Ingénierie des Besoins des Systèmes Interactifs demande l'utilisation conjointe de concepts et techniques spécifiques à l'Interaction Homme-Machine (IHM) et de concepts et méthodes du Génie Logiciel (GL). En fait, l'IB est une activité clé pour la construction de systèmes et elle est présente en tous les modèles de cycle de vie dans la littérature du GL - comme par exemple 'Waterfall' [Boehm 81] et Spiral [Boehm 88] - et de l'IHM, comme par exemple le cycle de l'Ingénierie d'Utilisabilité (*Usability Engineering*) [Nielsen 93] et le cycle étoile ('*star*') [Hartson & Hix 89].

Cependant, dans ces deux domaines, l'Ingénierie des Besoins est interprétée de manière différente, avec différents types d'information considérées, différentes techniques de recueil de ces informations, differents modèles pour les représenter. En fait, les approches orientées système - préoccupés surtout avec les aspects architecturaux et procéduraux pour la spécification et la réalisation du système - sont surtout originaires du domaine de GL [Brown 97] alors que les approches orientés utilisateur - préoccupées avec une compréhension des utilisateurs et leurs activités de travail pour guider la conception du système - ont principalement pour origine le domaine de l'IHM.

C'est un fait que les besoins déterminés par des méthodologies du GL, orientées surtout aux besoins fonctionnels, sont insuffisants pour prend en compte les aspects de l'interaction. En outre, l'IHM se concentre en général sur l'interaction et l'utilisabilité sans une adéquate considération des autres besoins du logiciel observés par le GL. Cependant, le développement d'un système ne peut pas se résumer en une recherche de l'utilisabilité et de l'utilité. Les systèmes doivent par exemple être robustes et faciles à maintenir. En même temps, le développement de systèmes doit respecter des contraintes comme l'intégration dans un cycle de vie, le coût (budget) et le temps de développement. Un système qui ne respecte pas ces aspects n'a pas de qualité, surtout dans le sens courant du GL.

Dans les domaines du GL et de l'IHM, il y a encore une tendance à traiter le développement des systèmes interactifs comme respectivement un 'cas spécial' du développement de logiciel (cas GL) ou l'addition de plus de fonctionnalité à une interface utilisateur (cas IHM). Cette distinction est à la base d'un problème d'intégration interdisciplinaire, qui continue tout au long du cycle de vie du logiciel interactif. La conséquence la plus imminente de ce problème est le *fractionnement des besoins***,** parce que chaque domaine ne considère que des aspects différents et souvent disjoints du système (les besoins du GL sont décrits en documents ou sections de documents différents des besoins de l'IHM) sans une manière systématique pour faire une correspondance explicite entre eux. Nous sommes d'accord avec plusieurs auteurs qui pensent qu'il faut considérer les systèmes interactifs comme un sujet en soi. Ce

point de vue est à la base des approches d'intégration IHM-GL pour la conception de systèmes interactifs.

La section 1 de ce chapitre présente des approches d'IB du GL, en particulier les approches d'Analyse Orientée Objet. D'abord les caractéristiques générales des approches du GL sont soulignées. Puis, comme un but de TAREFA est l'intégration à une méthode de conception orientée objet, trois approches d'Analyse Orientée Objet, à savoir OOA, OMT et Objectory sont décrites.

La section 2 présente une révision des approches les plus communes de l'IHM pour la construction de systèmes interactifs. La plupart d'entre eux ne possède pas une étape ou phase explicitement appelée Ingénierie des Besoins. En général, elle est diluée dans le processus de conception. Cette section a pour but d'essayer d'identifier comment l'IB est faite selon le point de vue de chaque approche.

La section 3 présente un ensemble d'approches qui visent à l'intégration de ces deux perspectives - en particulier l'intégration des facteurs humains typiques des approches d'IHM à des méthodologies de conception de logiciel typiques du GL. Dans le cadre de ce mémoire nous nous intéressons principalement aux avantages et inconvénients de ces approches d'intégration. Pour cela, nous avons choisi de présenter leur comparaison selon un ensemble de critères pertinents, définis dans la section 3.1. Cette comparaison va nous permettre plus tard (au chapitre IV) de positionner nos travaux par rapport à l'ensemble des travaux existants.

8.Approches du GL

La plupart des approches du GL font partie de méthodologies de développement de systèmes, sans une préoccupation spécifique avec des systèmes interactifs. En fait, les systèmes interactifs sont très difficiles à construire et demandent des ressources et des connaissances supplémentaires, qui en général ne font pas partie de la formation des informaticiens.

Il faut reconnaître que les méthodes du GL ont changé moins rapidement dans les vingt dernières années que les systèmes eux-mêmes et la manière de les utiliser. En fait, étant donné leur origine, beaucoup d'entre eux ignorent ou donnent une importance secondaire aux aspects de l'IHM ou plus généralement d'utilisabilité .

Par compte, les approches du GL ont démontré leur efficacité pour :

- structurer et contrôler le développement du logiciel, difficile surtout pour le travail en équipe ;
- représenter les aspects de données, architecture et fonctionnalité du système ;
- aider à atteindre les buts de qualité du produit.

8.1Une vue générale des approches

La démarche classique des approches du GL consiste à recueillir l'information sur le domaine du problème et à utiliser cette information pour la spécification d'une application informatique. La première activité concerne typiquement la construction d'un modèle conceptuel à partir des informations obtenues. Traditionnellement, les informations collectées concernent plutôt des fonctions, des données, des séquences d'états ou objets qui doivent être modélisées d'après l'application (dimension interne) du système à construire et non selon l'importance pour la compréhension du problème.

Malgré l'intérêt croissant porté à l'Analyse des Besoins, on observe que la plupart des travaux et des méthodes de la littérature dirigent leur attention vers le problème de la modélisation, créant plusieurs modèles différents, mais modélisant tous nécessairement les mêmes types d'information orientés système. Le choix des méthodes et des outils pour cette modélisation est déterminé alors en fonction de ces informations :

- SSA, SADT, Analyse Structurée Moderne, par exemple - dans le cas d'une analyse orientée fonctions;

- ER ("Entity-Relationship"), Merise, JSD, par exemple - dans le cas d'une analyse orientée données;

- "statecharts", Réseau de Petri - dans le cas d'une analyse orientée séquence d'états, utilisée surtout pour les systèmes temps réel ;

- OMT, OOA, OOSA, OMT, OBA et Objectory- dans le cas d'une analyse orientée objet.

Ces approches sont très connues des informaticiens et font partie intégrante du curriculum typique de courses d'Informatique. La plupart d'entre eux font partie de méthodologies de développement de systèmes, sans une préoccupation spécifique avec des systèmes interactifs.

Ce point de vue est typique d'un développement de logiciel orienté produit en opposition à un développement orienté processus [Floyd 88].

Pour le **développement orienté produit**, le logiciel est considéré comme un produit (composé d'un ensemble de programmes et de sa documentation associée), en séparant ses besoins du contexte social de l'application dans lequel le logiciel est inclut. Cela veut dire que ce contexte d'usage est considéré fixe et bien compris et par conséquence les besoins du logiciel peuvent être déterminés complètement à priori, en restant stables pendant le développement. A partir de ces besoins, une spécification complète (et sa réalisation) peut être dérivée à partir de ces besoins à travers des manipulations formelles, ce qui n'est vraiment pas le cas pour les systèmes d'information interactifs [Fischer 89].

Bien sur, ce développement orienté produit ne sert pas comme bonne base méthodologique pour les systèmes interactifs parce qu'il oublie :

- a) le soutien aux processus de communication et d'apprentissage mutuel entre des utilisateurs et des concepteurs;

- b) la perspective d'utilisation de l'ordinateur comme un outil de travail pour la résolution des problèmes par les utilisateurs;

- c) le contexte du problème et sa nature dynamique.

Selon [Floyd] le type de développement plus adéquat serait le **développement orienté processus**. Ce type de développement ne structure pas le développement du logiciel en phases comme conception, implémentation ou maintenance mais il le regarde comme une séquence de cycles de développement dont les concepteurs et les utilisateurs coopèrent. Le but de chaque cycle est de produire une version du système qui peut être évaluée. La transition d'un cycle au suivant est commencé par une révision, basée dans les résultats de l'évaluation, de la version courante.

Un résumé des caractéristiques plus significatives de chaque type de développement est montré dans le figure 3.1.

9)

Caractéristique	Développement Orienté Produit	Développement Orienté Processus
Programmes (définition)	Sont des objets formels mathématiques	Sont des outils ou environnements de travail pour les personnes
Construction de programmes	Sont dérivés d'une spécification abstraite à travers de procédures formalisés	Sont conçus en procès d'apprentissage et de communication de manière à remplir les besoins humaines
Système de référence (l'univers où s'insère le système informatique)	Est considéré comme statique, avec 2 états : avant et après le système informatique être introduit	Est considéré comme dynamique et en évolution constante : le système informatique change l'environnement et doit changer pour atteindre les conséquentes nouveaux besoins
Le concepteur du système	Est en dehors du système réfèrent et le développement et l'utilisation sont considérés séparément	Devient une partie du système réfèrent et les procès de développement et d'utilisation s'influencent mutuellement
L'interaction entre le logiciel et l'environnement	Est predefinie comme partie du développement	Est définie pendant le processus de travail de manière à atteindre les réels nécessités des utilisateurs
Critères de Qualité de Logiciel	Associés à des caractéristiques du produit	Associés aux procès d'utilisation du produit
Détermination de Qualité	Validation (Combinaison de test et preuves)	Évaluation (tentative d'utilisation, argumentation et critiques)
Développement du Logiciel	La production est divisée en phases, en séquence linéaire, et la maintenance est une phase séparée à la fin de la séquence	La production est cyclique et il n'y a pas une séquence prédéterminé et comme le développement est continu, il n'y a pas de séparation entre développement et maintenance
Compétence d'utilisateur	Signifie "savoir opérer le programme" et les utilisateurs sont catégorisés selon cela; les erreurs sont l'expression d'incompétence	Signifie "exécuter des tâches de travail avec l'aide du programme" et change continuellement comme tout processus d'apprentissage; les erreurs sont inhérents a ce processus et servent pour acquérir la compétence
Modèle d'utilisateur	Est maintenu par le programme et sert à moniteur d'utilisateur	Il n'y a pas un modèle rigide
Compréhension des programmes	Doit être possible à partir des documents seulement	Les tentatives d'utilisation et les discussion entre les personnes sont indispensables et les documents facilitent ces activités
Utilisation des formalismes	Fondamental pour la description du modèle de spécification	Quand c'est nécessaire mais la langage naturel est la partie essentielle pour la communication
Documentation	Pour décrire les résultats des processus de conception	Doit décrire aussi comment les résultats ont été obtenus : les décisions, ses raisons et les choses apprises
Méthodes et leur utilisation	Les méthodes sont des produits génériques; l'utilisation propre des méthodes arrivera aux résultats uniformes et deviendra le développement moins dépendant des personnes; les méthodes servent à spécifier des aspects du produit	Il n'y a pas des méthodes fixées mais des procès de développement ou d'adaptation et d'utilisation des méthodes à un type d'application spécifique; les méthodes servent comme soutien à la communication entre les personnes, qui coopérativement découvrent graduellement le produit

Figure 3.1 - Comparaison entre les deux types de développement de logiciel

Dans la section suivante nous nous concentrons particulièrement sur les caractéristiques de méthodes d'Analyse Orientées Objet.

8.2L'Analyse Orientée Objet

Le paradigme objet est aujourd'hui le paradigme le plus répandu parce qu'il est reconnu par les ingénieurs logiciel comme le paradigme qui permet le développement (et la reutilisabilité de ce développement) et la maintenance de systèmes de bonne qualité, en accord avec les critères de qualité soit internes soit externes [Meyer 88], [Jacobson 92].

Même sans un consensus sur ce qu'est exactement un objet, sur comment identifier un objet et le représenter en utilisant une approche de modélisation orientée objet, il y a

beaucoup de méthodologies et méthodes pour la conception des systèmes selon le paradigme orienté objet comme, par exemple, Object-Oriented Design (**OOD**) [Booch 91], Object-Oriented Systems Analysis (**OOSA**) [Shlaer 88], Object Modeling Technique (**OMT**) [Rumbaugh 91], Object-Oriented Analysis (**OOA**) [Coad 91], Object Behaviour Analysis (**OBA**) [Rubin 92] et Objectory[8] [Jacobson 92]. Le problème central, pour toutes ces méthodes citées ci-dessus, est toujours de trouver une structure d'objets approprié au système. En fait, chacune de ces méthodes suggère des heuristiques différentes pour cette identification. Par exemple, '(...) les noms et les verbes utilisés dans les descriptions informelles du domaine correspondent respectivement aux classes et opérations du modèle objet (...)' [Booch 94].

La proposition d'une Analyse Orienté Objet, comme les autres modes d'analyse, est d'obtenir une compréhension de l'application, en analysant de manière intégrée et itérative le comportement et l'information du système. Parmi les méthodes ci-dessus, trois affirment soutenir les activités de l'Analyse des Besoins : OOA, OMT et Objectory, c'est pourquoi nous nous intéressons à elles. Un résumé des méthodes OOA et OMT est présenté dans la section suivante. La section 1.2.1.2 souligne quelques critiques à ces deux méthodes. Puis la méthode Objectory - qui représente un changement de point de vue pour l'analyse orienté objet - sera présentée dans la section 1.2.1.3.

8.2.1.1L'Analyse Orientée Objet : les méthodes OOA et OMT

La méthode **OOA**, Object Oriented Analysis, a pour but de développer un modèle d'analyse qui décrit la fonctionnalité du système. Pour cela, elle utilise une notation graphique aujourd'hui presque reconnue comme un standard. La méthode OOA consiste en cinq pas successifs, dans l'ordre suivant :

1- Trouver les Classes et Objets;
2- Identifier les Structures (Généralisation,Agrégation, etc);
3- Définir les Sujets, une approximation du concept de vue ou sous-système;
4- Définir les Attributs;
5- Définir les Services.

La méthode **OMT**, Object Modelling Technique, possède plus de concepts que les autres parce qu'elle soutient l'analyse, la conception et l'implémentation. La méthode consiste en quatre phases : analyse, conception du système, conception des objets et implémentation. Trois modèles du système sont d'abord développés puis raffinés durant toutes les phases. Spécifiquement pour la phase d'Analyse :

- le modèle d'objets décrit la structure statique du système (classes et objets du domaine, ses relations et attributs);
- le modèle dynamique capture les aspects temporels du modèle d'objets avec des séquences ("traces") d'événements et diagrammes d'état pour les classes;
- le modèle fonctionnel décrit comment les valeurs d'entrée sont transformées en valeurs de sortie en termes d'opérations sur les objets ;

8.2.1.2Quelques critiques à OOA et OMT

La grande déficience de ces deux méthodes est, en fait, que les techniques pour

8 Le nom de la méthode Objectory est parfois confondu avec le nom du livre dans lequel elle a été proposée : Object Oriented Software Engineering (**OOSE**).

trouver les objets, les classes, les attributs et les opérations sont très heuristiques. On commence par une découverte des classes (et objets) et des opérations qui forment le vocabulaire du domaine du problème, à partir respectivement des noms et des verbes utilisés. Après, quelques abstractions et relations entre classes et objets (généralisation, spécialisation, instanciation, agrégation, décomposition, association, etc.) aident à organiser le squelette de la structure d'objets.

Cette stratégie, même si elle est appropriée pour quelques systèmes de petite taille, ne sert pas pour la majorité des grands systèmes interactifs parce que :

- il y a une grande difficulté pour définir exactement quels sont les objets (et aussi ses attributs) essentiels pour un système particulier sans savoir comment le système sera utilisé; un objet peut être parfaitement adéquat à un système dans un domaine d'application mais totalement inadéquat à un autre système dans le même domaine : cela signifie qu'un objet doit être considéré **dans un contexte d'utilisation;**

- les objets identifiés, typiquement des objets qui correspondent à des entités réelles, reflètent les composants **statiques** do domaine du problème: il n'y a pas de soutien pour capturer systématiquement les rôles **dynamiques** des objets, soit des rôles **fonctionnels** (les opérations qu'ils peuvent exécuter, les opérations qu'on peut exécuter avec eux, les messages échangés et tous les autres types de collaboration avec d'autres objets), soit les rôles **comportementaux** (les interactions que l'utilisateur peut faire avec eux);

En plus, l'utilisation de la notion d'objet, bien que primordiale dans les phases de conception et implémentation, a l'inconvénient de privilégier **la logique du système** au détriment de **la logique du poste de travail**. Malgré le consensus entre les informaticiens que l'orientation objet modélise naturellement les concepts du domaine du problème, les ergonomes reconnaissent que l'utilisateur comprend mieux des notions comme tâches, buts, procédures et rôles [Sebillote 88]. Dans les approches orientées objet, ces notions ne sont pas présents de manière explicite, mais de manière diluée dans les description des classes/objets [Barthet & Tarby 92].

Comme les ergonomes admettent que les utilisateurs utilisent la notion de tâches pour décrire leur travaux et leur activités plutôt que la notion d'objet [Sebillote 91], il faut que la modélisation conceptuelle prenne en compte le point de vue des tâches à la modélisation orienté objet. Cependant, l'utilisation véritable des modèles de tâche dans les méthodologies orientées objet - et plus généralement dans les méthodologies de développement de logiciel - reste encore une question ouverte. En particulier, les méthodologies du GL ne fournissent aucune aide pour l'intégration des modèles de tâche au processus d'analyse des besoins comme il est dit par [Davis 93], [Hix 93], [P. Johnson 94].

8.2.1.3Un point de vue différent : la méthode Objectory

La méthode Objectory [Jacobson 92] supprime ces déficiences par l'intermédiaire d'une définition des scénarios d'utilisation futur du système, appelés **cas d'utilisation** (*use cases* en anglais). Les cas d'utilisation fournissent un mécanisme pour diviser un domaine en des petits scénarios qui reflètent comment le logiciel sera utilisé par des agents extérieurs (typiquement les utilisateurs ou d'autres systèmes ou encore d'autres organisations).

La méthode **Objectory** soutien le développement du logiciel à travers le modèle des

besoins, le modèle d'analyse, le modèle de conception, le modèle d'implémentation et le modèle de test associés respectivement aux processus d'analyse, de construction et de test, cf. Figure 3.2. Une différence importante entre les deux premiers modèles c'est que dans le modèle des besoins on spécifie les besoins fonctionnels du système à travers une approche orientée cas d'utilisation ("**use case**") et dans le modèle d'analyse on spécifie la structure du système de manière indépendante de l'environnement d'implémentation pour avoir une plus grande stabilité aux changements. Selon [Jacobson 92], un modèle qui utilise uniquement des objets qui correspondent à des entités réelles ne génèrent pas des systèmes stables, parce qu'il n'y a pas une préoccupation de localité des fonctions et de la maintenance. Alors, on doit modifier les objets du modèle des besoins pour construire le modèle d'analyse. Le modèle d'analyse n'a pas de correspondance dans les autres méthodes orientées objets. On va se concentrer un peu plus sur le modèle des besoins puis sur le concept de cas d'utilisation.

10)

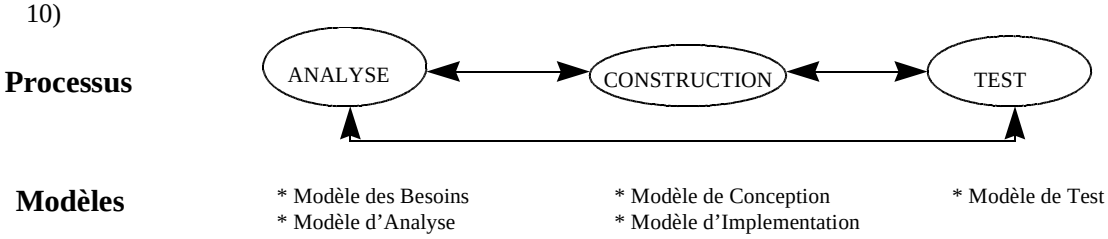


Figure 3.2 - Les processus et modèles de la méthode Objectory

Un **modèle des besoins** complet, selon la méthode Objectory, consiste de 3 modèles décrits ensuite :

1. un modèle des cas d'utilisation;

2. des descriptions des interfaces utilisateurs ;

3. un modèle du domaine du problème, qui inclut des objets qui ont une correspondance directe avec les entités réelles du domaine du problème.

A. Modèle de Cas d'Utilisation

Les Cas d'Utilisation (aussi appelés par l'anglicisme ***use cases***) ont été introduits par Ivar Jacobson pour sa méthode Objectory [Jacobson 92]. Cas d'utilisation reçoivent maintenant beaucoup d'attention de la part de la communauté 'orientée objet', comme OMT [Rumbaugh 91], Booch [Booch 94], ROOM [Selic et al. 94], OBA [Rubin and Goldberg 92] et même la nouvelle méthode unifiée [Booch & Rumbaugh 95]. Le figure 3.3 résume l'utilisation de cas d'utilisation pour les méthodes orientées objet qui les prennent le plus en compte.

11)

	OMT	**Booch**	**Objectory**
Concepts associés	use case, acteur, scénario, pré et post condition, exception, *adds*	use case, scénario, initiateur, pre et post condition	use case, acteur, exception, *extends, uses*
Notation utilisée	langage naturelle avec quelques recommandations pour la structurer	langage naturelle	langage naturelle pour décrire les use cases ;diagrammes pour décrire les relations entre les use cases
Rôle dans le processus de développement	Identification des objets	Identification des objets ; Conception ; Planning de version	Fils conducteur du processus (Analyse, Conception, Implémentation et Test)
Création des cas d'utilisation	Liste d'actions pas-à-pas ; scénarios sont combinés/généralises	Quelques recommandations	Quelques heuristiques pour identifier les use cases et construire l'architecture du système

	en use cases		

Figure 3.3 - Vue d'ensemble de l'utilisation des cas d'utilisation (Use Cases) par les méthodes orientées objet

Pour l'IB, les cas d'utilisation sont très interessants pour la détermination, l'analyse et la documentation des besoins, comme demontré par les travaux de [Potts 94, Potts 95, Hsia et al. 94, Leite 97, Sutcliffe 97]. Les cas d'utilisation permettent aussi la traçabilité des besoins à partir de la conception, de l'implémentation et de l'évaluation (validation ou vérification).

Un objectif principal de l'Analyse Orienté par Cas d'utilisation (*Use-Case Driven Analysis*) de la méthode Objectory est de modéliser les besoins fonctionnels du système par l'intermédiaire de la description de différents scénarios d'utilisation du système. L'ensemble des descriptions est appelé Modèle des Cas d'Utilisation (*Use Case Model*). Les concepts basiques de ces descriptions sont les acteurs et les cas d'utilisation. Un acteur possède un rôle spécifique joué par l'utilisateur du système ; il représente une catégorie d'utilisateurs réels qui montrent un comportement similaire pendant l'utilisation du système. La manière avec laquelle un acteur utilise le système est décrite par un cas d'utilisation. Une avantage de l'analyse orienté cas d'utilisation est l'aide à la gestion de la complexité, parce que il n'y a qu'une situation d'utilisation intéressante à chaque instant. Une autre avantage est qu'elle fournit les moyens pour une participation plus active de l'utilisateur pendant l'IB, parce que les cas d'utilisation sont exprimés en termes qui lui sont familiers.

En fait, pour I. Jacobson, un cas d'utilisation est une '*....behaviourally related sequence of transactions in a dialogue between the user and the system*' [Jacobson 93]. En OBA, des scripts '*...record the use of the (proposed) system*' [Rubin & Goldberg 92]. Les deux notions sont d'accord sur la caractéristique plus importante des cas d'utilisation : décrire le système du point de vue de son utilisation, c'est à dire, du point de vue de l'utilisateur.

Un **cas d'utilisation** d'un système est une manière spécifique pour un acteur d'utiliser une partie de la fonctionnalité du système. Cela représente un dialogue entre un acteur et le système. Un acteur c'est un rôle joué par une entité externe au système (personne, système, institution, etc) qui échange des informations avec le système. Trouver les acteurs d'un système c'est identifier les personnes que le système est supposé aider. Trouver les cas d'utilisation d'un système c'est identifier les événements initiés par les acteurs et manipulés par le système. L'ensemble de tous les cas d'utilisation représente la fonctionnalité complète d'un système, du point de vue de l'utilisateur, et c'est le point de départ pour son développement ou sa maintenance. Ainsi, on utilise les cas d'utilisation comme le concept principal non seulement pour délimiter le système et définir sa fonctionnalité - c'est à dire, construire le modèle des besoins du système - mais aussi pour conduire le processus de construction ou de modification des autres modèles du système.

B. Descriptions de l'Interface

Pour Objectory, les descriptions de l'interface sont faites en général, à travers des maquettes ou des prototypes. En fait, comme la plupart des méthodes informatiques Objectory ne présente rien de bien approfondi sur ce sujet .

12)

C. Modèle du Domaine du Problème

Le modèle du domaine du problème est le principal point d'attention de plusieurs méthodes (y compris les deux premières méthodes décrites, OOA et OMT) qui l'utilisent comme base pour la conception, en faisant la correspondance directe entre les

classes de ce modèle et les classes d'implémentation.

8.2.1.4Objectory : discussion des avantages et inconvénients

Objectory utilise les cas d'utilisation comme le concept principal non seulement pour délimiter le système et définir sa fonctionnalité - c'est à dire, construire le modèle des besoins fonctionnels du système - mais aussi pour conduire le processus de conception, d'implémentation et detest du système. Une fois que le système est déterminé par des besoins fonctionnels des utilisateurs dans une situation d'utilisation, Objectory est une méthode orientée manipulation d'événements, dont les événements sont déclenchés par les tâches d'utilisation du système. Alors, Objectory est une méthode informatique qui est un très bon candidat pour l'intégration d'une méthode d'analyse de la tâche à une méthodologie orienté objet. En fait, ses descriptions orientés utilisation offrent un contexte réel et riche pour l'analyse des objets du domaine du problème et permettent une meilleure compréhension de problèmes d'utilisabilité.

Malgré la reconnaissance de ses contributions (voir par exemple le Workhop sur '*Use Cases*' en OOPSLA 95), quelques aspects de Objetory ont besoin encore de raffinement et amélioration. Des travaux récents incluent par exemple la formalisation du concept de cas d'utilisation ou la combinaison de Objectory et des techniques de vérification/validation [Wohlin 94]. Sans doute on peut ajouter que le concept de cas d'utilisation a besoin de une définition plus claire parce que [Regnell 95] :

- le taille et les niveaux d'abstraction de chaque cas d'utilisation sont arbitrairement choisis par l'analyste ;

- les cas d'utilisation peuvent se chevaucher totalement ou partiellement, se produire simultanément ou s'influencer l'un à l'autre ; donc ils ne sont pas indépendants ;

- la notation des cas d'utilisation est le langage naturelle, non adéquate à la conception assistée ou automatique ;

- l'information du contexte dans lequel le cas d'utilisation se produit n'est pas considéré.

Alors pour une utilisation effective de Objectory, il faut faire attention à ces aspects. On va voir dans les chapitres 4 et 6 comment le concept de cas d'utilisation est utilisé par TAREFA pour l'ingénierie des besoins des systèmes interactifs.

9.Approches de l'IHM

La plupart des approches de l'IHM pour la construction de systèmes interactifs ne possèdent pas une étape ou phase explicitement appelée Ingénierie des Besoins. En général, elle est diluée dans le processus de conception.

En général, ce processus de conception des approches IHM est surtout expérimental et bien moins formel que les méthodes du GL. En fait, les concepteurs d'IHM résistent à définir un processus, peut être parce qu'ils ont besoin de flexibilité pour sa pratique de travail ou même parce que la discipline d'IHM n'est pas encore mûre pour le définir.

Cependant, il peut être considéré à deux niveaux [Sutcliffe 95] :

- IHM haut niveau ('HCI in the large'), par definition '.... specification and design of system functionality with particular attention to designing interactive software

to cooperate with users in achieving a job of work'.

- IHM bas niveau ('*HCI in the small*'), par définition '*...design of detail of interaction, i.e., the user-system dialogue and user interface displays*'.

Bien sur, les deux niveaux sont interconnectés. Les besoins de l'utilisateur doivent être définis pour découvrir les buts du système avant de commencer la définition des détails de l'interaction. En contrepartie, les décisions faites au niveau de détail peuvent avoir des conséquences au niveau de la tâche. Le processus de conception d'IHM implique à la fois des aspects techniques (dispositifs d'entrée/sortie, multimodalité, etc) et sociaux (description de travail, coopération, etc). Son résultat n'est pas forcement un système informatique mais probablement des nouvelles procédures de travail (job design), y compris des changements d'environnement, des tâches et des parties qui aident l'utilisateur - comme manuels d'utilisation et programmes de formation (voir [Harker & Eason 89]) . Nous nous concentrons dans cette thèse et plus particulièrement dans cette section sur les approches d'IHM haut niveau.

Une des questions principales pour les approches d'IHM est l'identification et la définition des propriétés des systèmes interactifs de haute qualité, dans la perspective des utilisateurs. Ces propriétés, comme vu dans le chapitre I, peuvent être considérées comme besoins du développement.

Il y a deux grandes classes d'approches d'IHM pour développer les interfaces qui respectent ces besoins :

1. Les **approches prescriptives**, qui utilisent les principes, les théories ou les formalismes pour la construction d'une spécification de l'interface ; y compris respectivement les approches basées sur heuristiques, les approches basées sur théories cognitives et les approches formelles.

2. Les **approches expérimentales**, qui construisent plusieurs versions exécutables de l'interface du système en développement pour évaluation par l'utilisateur ; y compris la Conception Participative, la Conception Centrée Utilisateur et les approches basées sur Scénarios.

Bien sur, comme il est très difficile de prévoir l'adéquation et l'efficacité d'une interface par rapport aux utilisateurs, il est impératif d'obtenir l'opinion de l'utilisateur : même pour les approches prescriptives, le système doit être testé par les utilisateurs potentiels ou leurs représentants. En fait, cela est encore nécessaire parce que aucune fondement théorique de la cognition et du comportement humain ne sont suffisamment complèts et mûrs pour servir de base unique à la construction.

L'attention sur l'utilisateur est considérée comme fondamentale par les concepteurs d'interface utilisateur, un fait insisté par plusieurs auteurs comme [Coutaz 90] [Myers 94], [Shneiderman 87], [Laurel 91], [Scapin 93],[Barthet 88]. En fait, il y a un consensus dans la communauté IHM sur le fait que l'expérimentation avec la participation de l'utilisateur est la principale technique pour réduire les risques dans la conception de logiciels interactifs. L'expérimentation est planifié pour guider le recueil des informations et les résultats des expériences sont testés pour valider, réviser ou rejeter les idées de conception. Le cycle de l'expérimentation et l'évaluation est à la base du principe de développement itératif des IHM. Un cas spécial de ce développement itératif est le prototypage, c'est à dire, la construction rapide d'une portion (horizontale ou verticale) du système avec une fonctionnalité limitée. Un prototype peut être construit manuellement ou généré automatiquement à partir d'une spécification. Ces deux manières sont aussi caractéristiques respectivement des

approches expérimentales et prescriptives de l'IHM décrites dans cette section.

Dans les paragraphes qui suivent, nous présentons une description succincte de certaines approches très representatives des approches prescriptives et experimentales les plus significatives, à savoir,

- 2 approches basées sur théories cognitives - approches basées sur GOMS et l'approche des fonctions cognitives - comme representantes des approches prescriptives ; et

- les approches de Conception Participative, de la Conception Centrée Utilisateur et les approches basées sur Scénarios comme representantes des approches experimentales.

9.1Approches Cognitives

L'application des concepts et techniques des Sciences Cognitives - notamment basés sur les théories originaires de la Psychologie, de l'IA et de la Théorie de Information - pour l'IHM a été originairement organisée par [Card, Moran & Newell 83]. Ces auteurs ont proposé deux niveaux d'application : le plus bas niveau inclut le Modèle de Processeur Humain et le plus haut inclut le modèle GOMS. Le modèle du processeur humain rassemble les concepts perceptifs, cognitifs et moteurs dans une forme qui permet de les appliquer pour l'analyse quantitative et prédictive des temps d'exécution des tâches.

Le modèle GOMS (acronyme pour Goals Operators Methods and Selection rules) permet de décrire la connaissance procedurale impliquée dans la réalisation d'une tâche et il est donc utilisé surtout pour prédire la performance d'un utilisateur. En fait les approches basées sur GOMS sont les approches cognitives les plus couramment utilisées en IHM. Cependant, récemment G. Boy a proposé une nouvelle approche : la méthodologie "Analyse des fonctions cognitives" (CFA). Nous les avons choisies parce qu'ils representent bien deux lignes différentes de l'application des Sciences Cognitives à l'IHM : la ligne américaine (GOMS) et la ligne européene (CFA). Un résumé de ces deux types d'approches est présenté ci-après.

9.1.1Approches basées sur GOMS

Le modèle GOMS [Card 83] décrit l'activité de l'utilisateur à l'aide de buts (états recherchés), d'opérateurs (actions élémentaires réalisés), de méthodes (séquences d'opérateurs pour atteindre un but) et de règles de sélection (choix d'une méthode). Il permet d'évaluer et de prédire le comportement et le temps que met l'utilisateur pour atteindre un but. GOMS peut être à la fois utilisé en conception comme modèle de tâche et en évaluation pour prédire les performances de l'utilisateur. Selon la proposition originale de [Card 83], le nouveau utilisateur d'un système utilisera plusieurs stratégies d'apprentissage et de résolution de problèmes pour réaliser ses tâches. Avec l'expérience, ces stratégies deviennent des procédures utilisés de façon routinière. Selon [Kieras 88], le concepteur peut décider les opérateurs et procédures du système et écrire des méthodes de l'interface. Les propriétés de ces procédures vont diriger la facilité d 'apprentissage et la facilité d'usage du système. Si souhaité, le modèle GOMS peut être plus élaboré jusqu'au niveau plus détaillé (*keystroke level*) pour produire des prédictions d'utilisabilité.

Diverses approches ont été développés et validés empiriquement pour créer et utiliser

des modèles GOMS en IHM, en addition à la proposition original de [Card 83]. Un bon résumé est présenté par [Olson & Olson 90] et [John & Kieras 94].

9.1.2Analyse des Fonctions Cognitives [Boy 97a]

L'idée principale de cette approche est le transfert de fonctions cognitives de l'homme vers la machine. Une fonction cognitive est un processus cognitif, a l'origine humain, qui a un rôle dans un contexte limité, en utilisant un ensemble de ressources. Par définition, une **fonction cognitive** permet à son utilisateur de transformer une tâche de haut niveau en une opération bas niveau. Par exemple, l'identification d'une situation, la coordination d'actions, la prise de décision et la planification sont des fonctions cognitives de haut niveau. Elles peuvent être décomposées en fonctions cognitives de plus bas niveau.

Une question importante est que les contraintes doivent être suffisamment explicites afin de guider les décisions pendant le processus de conception. La méthodologie d'Analyse des Fonctions Cognitives (en anglais, *Cognitive Function Analysis* ou simplement CFA) permet à une équipe de conception de mieux comprendre le bon équilibre entre les fonctions cognitives qui doivent être affectées à (aux) l'utilisateur(s) et les fonctions cognitives qui doivent être affectées a la (aux) machine(s). Elle est basée sur trois aspects :

- les documents actifs qui servent de support au transfert des fonctions cognitives et à stocker la logique de conception;
- la définition de critères d'utilisabilité pour une conception centrée sur l'homme;
- la gestion d'une mémoire externe de documents actifs.

9.2Conception Participative

La participation active des utilisateurs comme des acteurs centraux dans les activités de développement est designée dans la communauté IHM par le terme de Conception Participative (Participatory Design ou simplement PD). En fait, Conception Participative est un terme attribué à une famille d'approches proposées d'abord en Scandinavie [Kyng 93] [Gronbaek 93] [Mogensen 92] [Bodker 92] et qui consistent à l'exécution d'activités conjointement par les concepteurs et les utilisateurs. Deux mot clés sont associés aux approches de Conception Participative : apprentissage mutuel et 'concevoir en faisant' (*design by doing*).

Dans ces approches, les utilisateurs et les concepteurs ont des compétences complémentaires mais d'importance équivalente (sans hiérarchie) pour comprendre l'univers d'informations, pour identifier les réels besoins des utilisateurs et pour prendre des décisions sur la conception du système. La participation conjointe pendant le processus de développement du système est une condition nécessaire pour créer de nouvelles connaissances requises (on peut aussi dire "construire la théorie du système", selon [Naur 85]) et pour une expérience partagée de la conception. Soutenir la coopération entre les concepteurs et les utilisateurs signifie faciliter le processus de communication entre eux, en visant un apprentissage mutuel (le terme anglais diffusé est "*mutual learning*"). A travers l'apprentissage mutuel, le concepteur apprend les pratiques de travail de l'utilisateur et l'utilisateur apprend à comprendre les différentes possibilités techniques envisageables pour le système.

Le deuxième mot clé '*design by doing*' signifie que la conception est faite à travers une expérimentation interactive, avec l'utilisation soit des outils '*low tech*' - comme des notes Post-It, un tableau noir, du papier et des stylos - soit des outils informatiques - comme des prototypes et/ou des maquettes. En fait, la pratique courante de Conception Participative est un défi à la flexibilité des outils informatiques actuelles et plusieurs auteurs préfèrent les outils '*low tech*'.

Malgré ces mots clés, il y a peu de disposition de la communauté Conception Participative en spécifier et systématiser techniques et pour définir une démarche standard de sa réalisation.

9.3Conception Centrée Utilisateur

L'idée d'avoir plus de connaissances sur les utilisateurs pour la conception de systèmes interactifs est chaque fois plus présente dans la communauté IHM, qui a une préférence pour le terme Conception Centrée Utilisateur (*User Centred Design* ou simplement UCD). En fait, UCD est un terme sans une définition consensuelle et qui peut être attribué à un ensemble d'approches [Karat 96]. Ensuite on présente les idées basiques des approches UCD.

9.3.1Caractéristiques générales

Les idées basiques de UCD correspondent aux principes d'une bonne conception (*good design*) présentés auparavant - sans utiliser le terme UCD - par [Gould 88] :

- attention très tôt et continue aux utilisateurs (y compris la suggestion de participation de l'utilisateur au processus de conception) ;
- évaluation aussi très tôt et continue des résultats (même partiels) ;
- conception itérative ;
- conception intégrée du système comme un tout (composant sémantique et de dialogue) .

L'UCD est alors un processus itératif qui utilise les utilisateurs ou les informations des utilisateurs comme sources des idées de conception ou comme critères d'évaluation pour la construction des systèmes utilisables. Evidemment l'idée de 'centre' peut être critiqué parce que, comme suggéré par [Atwood 95] la perspective de l'utilisateur n'est pas la seule devant être prise en compte.

L'exemple typique d'un approche de Conception Centré Utilisateur est l'Ingénierie d'Utilisabilité ('*Usability Engineering*') [Nielsen 93] que nous résumons ensuite.

9.3.2Ingénierie d'Utilisabilité

Cette approche a été proposé par Jakob Nielsen [Nielsen 93] comme une méthodologie pour le développement de systèmes interactifs, surtout dans un cadre industriel. Le mot clé pour la décrire est 'pratique' et c'est peut être pour cela que c'est une approche très connue et très appliquée. Elle est composé d'un cycle de 11 pas - voir figure 3.4 - dont l'objectif est de connaître l'utilisateur et les tâches qu'il réalise pour proposer des versions successives de prototypes du système qui seront testés par l'utilisateur.

Quelques pas de l'approche sont dérivés d'autres disciplines comme la Psychologie Cognitive, l'Ergonomie ou même le Marketing. Ces pas sont la combinaison de techniques qualitatives et quantitatives, et sont en général bien moins formels que les techniques du GL. En particulier, les pas relatifs à l'Ingénierie des Besoins - à savoir 1,2, 3, 8, 9 et 10 - ne montrent pas de manière claire la relation entre le composant du dialogue et le composant sémantique.

13)

1. **Connaître l'Utilisateur :**
    - **Caractéristiques Personnelles**
    - **Tâche Actuelle**
    - **Analyse Fonctionnelle**
    - **Evolution de l'Utilisateur**
2. **Analyse Compétitive**
3. **Etablir des Objectifs pour l'Utilisabilité**
4. **Conception Parallèle**
5. **Conception Participative**
6. **Conception Coordonnée de l'ensemble de l'interface**
7. **Lignes Directrices (*guidelines*) et Analyse d'Heuristiques**
8. **Prototypage**
9. **Tests Empiriques**
10. **Capturer la *Design Rationale***
11. **Retour sur l'utilisation du système sur le terrain**

Figure 3.4 - Les pas de l'Ingénierie d'Utilisabilité [Nielsen 93]

9.4Approches Basées sur Scénarios

Les scénarios sont des descriptions narratives des interactions entre des utilisateurs et le système. Un scénario est décrit du point de vue de l'utilisateur et il peut décrire une situation courante d'utilisation ou une utilisation potentielle d'un système à développer.

D'une certaine manière, les scénarios ne sont pas une nouveauté dans l'activité de conception. C'est extrêmement commun pendant la conception qu'on imagine des situations 'et si...', ou qu'on fasse une narration d'une interaction pour quelqu'un ou un groupe afin de réfléchir sur une situation spécifique.

Dans son livre '*Scenario-Based Design - Envisioning work and technology in System Development* [Carroll 95], John Carroll affirme que les scénarios sont un médium pertinent pour représenter, analyser et planifier comment un système informatique va changer les activités et les expériences de l'utilisateur.

Parce que ces descriptions peuvent inclure des interactions concrètes, les scénarios sont utiles pour aider les utilisateurs et les analystes à développer une compréhension partagée et consensuelle de la fonctionnalité et de l'utilisabilité d'un système, qu'il soit futur ou existant.

Les scénarios sont utilisés pendant la conception pour structurer l'interaction entre l'utilisateur et le concepteur afin de développer rapidement un ensemble de besoins initiaux. [Hoolbrook 90] suggère que les scénarios sont appropriés pour communiquer les caractéristiques spécifiques du système en situations de grande incertitude parce qu'ils ont un petit coût et une expressivité limitée.

En bref, les aspects principaux des scénarios sont :

- Au contraire d'une spécification générique, qui décrit des comportements 'normaux' en général dissociés d'une situation réelle d'usage, les scénarios sont un moyen efficace pour faciliter la communication utilisateur-analyste parce que les scénarios permettent d'exemplifier des comportements et de réfléchir sur leur

adéquation par l'intermédiaire de situations concrètes d'usage du système ;

- Les scénarios sont intéressants pour comparer différentes alternatives pour les séquences d'action, les quelles peuvent différer par exemple en fonction du degré d'automatisation ou de la métaphore d'interaction ;

- Les scénarios peuvent devenir des artefacts utiles pour tout le développement (comme en Objectory [Jacobson 92]), qui peuvent être augmentés et réarrangés au fur et à mesure que le développement avance.

- Un historique des scénarios décrits pendant le développement peut être considérée comme une manière élémentaire de capturer son '*design rationale*', c'est à dire, l'historique des décisions et arguments associés à chaque modification des scénarios.

14)

Perspective des Scénarios	Perspective du Génie Logiciel
descriptions concrètes	descriptions abstraites
attention à instances particulières	attention à types génériques
orientée travail	orientée technologie
fragmentaire, incrementale	complète, exhaustive
informelle	formelle, rigoureuse
solutions envisagées	solutions spécifiées

Figure 3.5 - La perspective des scénarios *versus* la perspective du GL (d'après [Carroll 95])

Une petite comparaison, extraite de [Carroll 95], peut illustrer les principales caractéristiques des scénarios par rapport à la perspective adoptée par le GL, voir figure 3.5. Les caractéristiques des scénarios montrent que ils sont l'antithèse d'une spécification : un scénario exemplifie un comportement particulier alors qu'un spécification décrit un comportement générique. Cette généralité est une raison de la difficulté de la compréhension des spécifications par les utilisateurs, comme [Potts 95] l'a remarqué.

10.Approches d'integration GL-IHM

La nécessité d'intégration des techniques et modèles dans un cadre multidisciplinaire est essentiel pour le développement des systèmes d'information interactifs [Andriole 93]. En fait, selon [Barnard 93] , il n'y a pas un seul modèle qui capture toute l'information significative aux projets d'échelles et de durées différentes ni des moyens d'intégrer verticalement et horizontalement les différentes formes d'information multidisciplinaire (systèmes, cognition d'utilisateur et concepteurs) pour la conception des systèmes interactifs. Il faut, alors, identifier comment des techniques spécifiques peuvent se compléter l'une à l'autre pour réussir à arriver aux objectifs de la conception. Cela veut dire qu'il faut trouver une architecture pour l'intégration de ces concepts et utiliser des processus dans lesquels l'interaction entre les deux domaines (IHM et GL) soit explicite.

John Long a discuté différentes manières d'intégrer les deux domaines, en fonction du type de connaissance des facteurs humaines impliqué - connaissance substantive (comme par exemple les standards et *guidelines*) ou connaissance méthodologique (comme les méthodes d'analyse de la tâche) [Long 95]. En effet, il est possible de constater diverses tentatives d'intégration entre les approches qui vont être présentées dans cette section :

- intégration de paramètres ergonomiques et de l'Analyse du travail à une méthode de conception de logiciel, comme DIANE+ [Tarby & Barthet 96]

- intégration d'un méthode d'organisation des connaissances des facteurs humains à une méthode de conception de logiciel, comme MUSE [Lim & Long 94] ;

- intégration de modélisation des tâches à un cycle de prototypage, comme ADEPT [P. Johnson & H. Johnson 93] ;

- intégration de *guidelines*, recommandations, standards et modèles de tâche (de l'IHM), et de modèles d'architecture et des objets du domaine (du GL) pour la génération d'un prototype du système, comme TRIDENT [Bodart et al 94];

- intégration d'un modèle de tâche à la spécification formelle et exécutable d'une application interactive comme ICO [Bastide & Palanque 95].

La section 3.1 présente une description succincte de ces travaux. Ensuite la section 3.2 défini les critères choisis pour leur comparaison, qui est présentée dans la section 3.3. Puis la discussion (section 3.4) aide à comprendre la motivation de développer une approche spécifique pour l'IB dans le domaine des systèmes interactifs.

10.1Description des approches

Comme nous nous sommes intéressés spécifiquement à l'IB, notre description n'examinera que les concepts et caractéristiques associés à cette activité. En plus, les avantages et inconvénients de chaque approche en rapport à l'IB sont aussi presentés.

10.1.1DIANE+

La méthode DIANE [Barthet 88] intègre des facteurs humaines (en particulier de l'Ergonomie et des Sciences Cognitives) à la méthode informatique Merise. cf. Figure 3.6.

15)

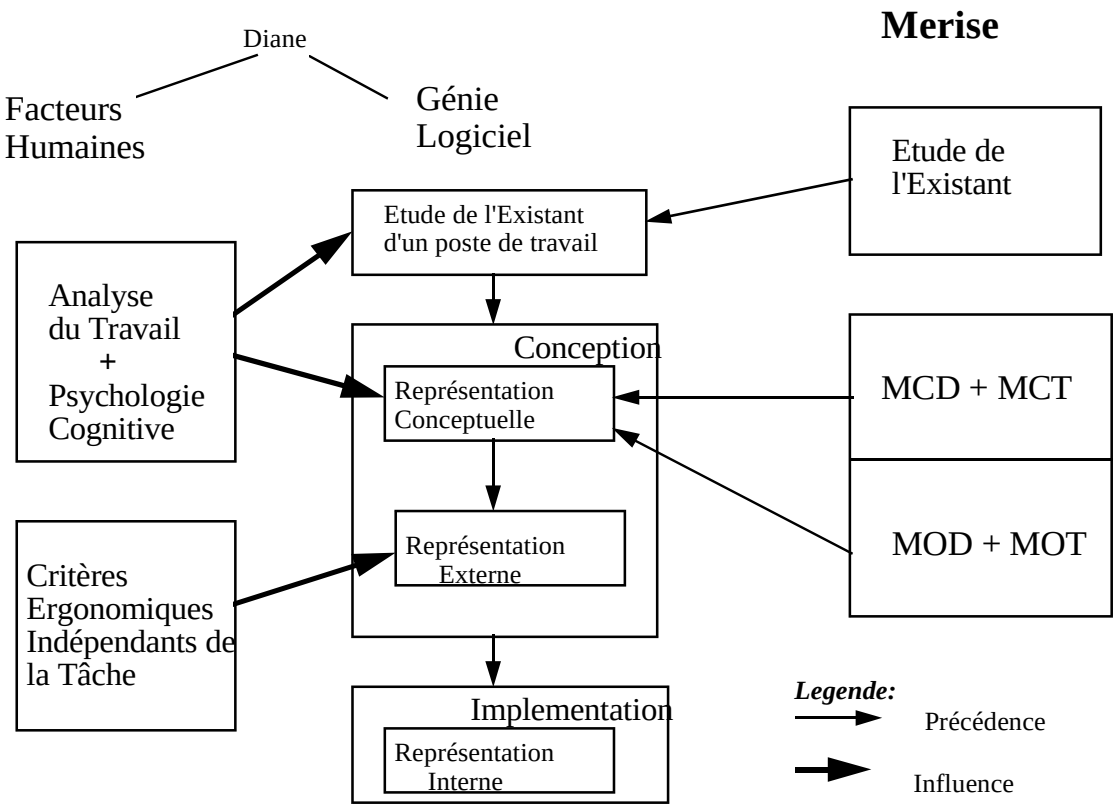


Figure 3.6 - La méthode DIANE et son integration avec Merise

DIANE utilise les résultats de l'Analyse du travail (y compris la logique d'utilisation et les connaissances sur les tâches, les utilisateurs, leurs buts, leur latitude décisionnelle, leur expérience) et propose un modèle et un formalisme pour la description du contrôle du dialogue et son interface avec le composant sémantique du système. Les notions de procédure effective, prévue et minimale représentent les différentes modes opératoires pour la réalisation d'une tâche avec un système : respectivement le mode standard, les modes réels et le mode qui indique les contraintes que l'opérateur doit impérativement respecter dans tous les cas. Cette variabilité permet d'adapter le niveau de contrôle exercé par le système sur le travail des opérateurs..

DIANE+ [Tarby 93] est une extension de la méthode DIANE - voir figure 3.7 - faite pour permettre la génération automatique de l'interface utilisateur. Cette extension porte d'une part sur le modèle du contrôle du dialogue d'autre part sur l'intégration d'un modèle objet OPAC, qui est une extension du modèle PAC [Coutaz 90]. Une application interactive sera donc d'abord décrite en termes de tâches spécifiant la répartition du contrôle entre l'homme et la machine auxquelles viendront s'ajouter les données OPAC manipulées. DIANE+ est une méthode de conception qui couvre tout le cycle de vie d'une application en incorporant des phases de génération de code (interface utilisateur, noyau fonctionnel, données) et de gestion automatique de l'application (dynamique de l'application, aide contextuelle). Etant donné qu'elle est restreinte aux applications de type préplanifiées [Rasmussen 83] par opposition aux applications de type expertes ou créatives, DIANE+ supporte les tâches associées à un poste de travail. Une application est définie en terme de buts associés à des tâches, elles-mêmes associées à des postes de travail supportant des utilisateurs de différents niveaux. Cette description étant réalisée, il est possible de générer un squelette d'application suivant des paramètres ergonomiques et en incluant l'interface utilisateur complète, ainsi qu'une partie du code suffisante pour la gestion automatique de

l'application (interface, noyau fonctionnel et aide contextuelle).

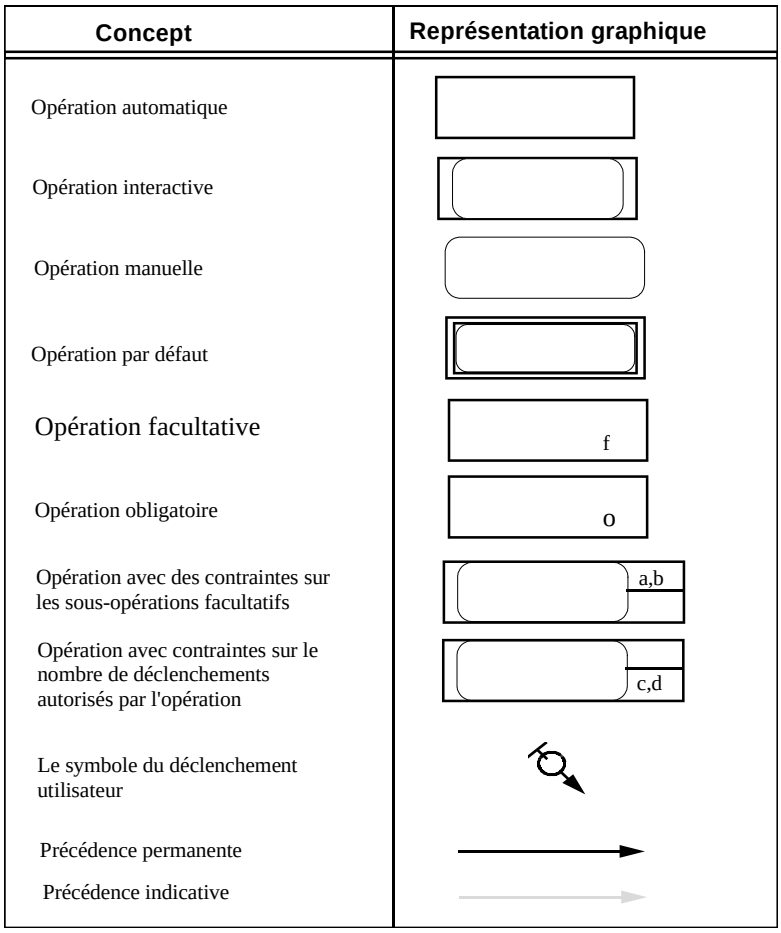


Figure 3.7 - Le formalisme graphique DIANE+

Les outils associées à DIANE+ sont un éditeur Diane+ et un contrôleur de dialogue implémenté sous la forme d'un moteur d'inférence chargé de la gestion automatique de la dynamique et de l'aide.

Avantages principaux de DIANE+ :

- Le concept de procédure minimale permet d'identifier les contraintes impératives et facultatives du dialogue utilisateur-système ;
- Les résultats de l'Analyse du travail sont intégrés très tôt aux processus de conception Merise ;
- La participation de l'utilisateur pendant le processus est possible mais son intervention n'est pas prévue jusqu'à utilisation du système ;
- La notation DIANE+ est plus adaptée au niveau de comprehension de l'utilisateur que les notations informatiques orientés système (SADT, DFD, etc) ;

Inconvénients principales de DIANE+:

- Le processus de passage de l'Analyse de l'existant à la spécification conceptuelle n'est pas explicite ni guidé par des étapes de la méthode. La réussite est fonction de l'expérience et de la qualité du travail de la personne responsable ;

- DIANE+ utilise la notion des objets OPAC mais sans utiliser aucune méthode orienté objet pour les identifier ;

- DIANE+ ne fournit pas de soutien direct à la spécification de la présentation de l'interface mais surtout à la spécification du dialogue.

10.1.2TRIDENT

TRIDENT (***T**ools fo**R** an **I**nteractive **D**evelopment **E**nvironme**NT***) est une méthodologie et un atelier pour le développement d'applications de gestion interactives. Le développement en utilisant TRIDENT part de l'analyse de la tâche de l'application future et arrive à la génération d'une maquette (prototype sans l'intégration du noyau sémantique). Les différents processus (parfois manuels, parfois assistés ordinateur, parfois automatisés), produits (résultants des processus), et outils de TRIDENT sont basés principalement sur les modèles ou éléments suivants, listés en ordre d'utilisation :

- un **modèle de tâche**, résultant d'une Analyse de Tâche hiérarchique qui utilise la méthode TKS et qui inclut, à partir des interviews et observations, une caractérisation des utilisateurs (existants et potentiels) et leur environnement physique pour mieux comprendre leurs tâches ;

- un **modèle entité-association** (*Entity Relationship* ou ER [Chen 76]) **étendu** pour la modélisation de la partie information des besoins fonctionnels, qui décrit les entités présentes dans le modèle de tâche, leurs attributs et leurs associations (agrégation, généralisation, etc) de manière récursive (un attribut peut lui-même constituer un objet) ;

- **un modèle de décomposition hiérarchique** pour la modélisation de la partie comportement des besoins fonctionnels, qui décrit les fonctions qui seront les méthodes associés aux objets du noyau sémantique de l'application. Ces fonctions doivent être dérivées ou actualisées à partir des tâches interactives modélisées;

- un **graphe d'enchaînement des fonctions**, qui exprime **a)** le flux d'informations entre les fonctions du domaine de l'application qui s'enchaînent (en accord avec les liens OU, ET et OU-exclusif) en vue de réaliser le(s) but(s) de la tâche interactive et **b)** à chaque fonction, les informations en entrée ou en sortie associées, qui correspondent aux attributs des objets (ou des associations) concernés par cette fonction;

En bref, la méthode TRIDENT part de l'analyse de tâche de l'application future et arrive à la génération d'une maquette, en utilisant un modèle entité association et un graphe d'enchaînement des fonctions pour connecter les tâches de l'interface utilisateur avec les fonctions et les informations de l'application.

L'atelier TRIDENT représente une tentative ambitieuse de générer un prototype réel à partir d'un modèle de tâche. L'état actuel de l'atelier est centré sur la génération de la présentation.

Dans la figure 3.8, les boites blanches dénotent les activités qui ont un rapport directe avec l'IB et les boites grises, les autres.

16)

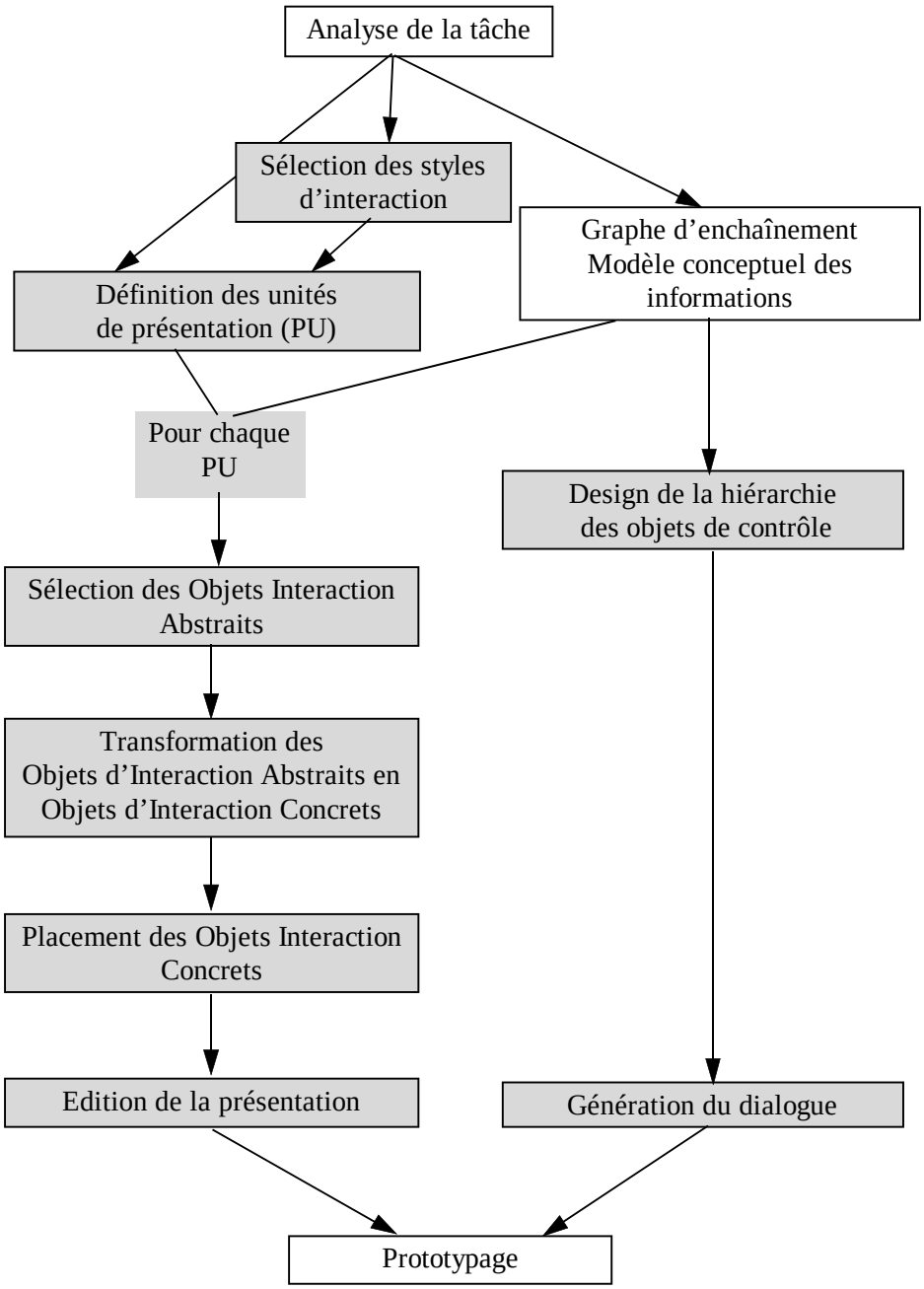


Figure 3.8 - Le processus de développement TRIDENT (d'après [Bodart et al 95])

Avantages principaux de TRIDENT:

- Le modèle de tâche est le point de départ pour la construction de l'architecture et la définition des services de l'application et aussi pour la génération de la présentation de l'interface utilisateur ;

- La structuration des connaissances ergonomiques en une base (très étendue !) de règles et d'heuristiques applicables à génération ;

Inconvénients principaux de TRIDENT :

- La classe d'applications interactives est réduite : systèmes de gestion ;

- La participation de l'utilisateur dans le processus de décision n'est pas incitée; l'utilisateur peut évaluer le prototype généré et faire des suggestions pour modifier le prototype ;

- Les tâches modélisées sont des tâches interactives prévues et non des tâches existantes, donc cette approche ne sert pas à l'IB mais à la conception du système : les besoins doivent être établis avant son utilisation.

10.1.3MUSE

MUSE (*Method for USability Engineering*) [Long [propose un processus de conception des facteurs humaines lié à des méthodes informatiques structurées. MUSE est la seule méthode décrite dans cette section dont l'utilisateur est un ergonome et pas un informaticien.

La version décrite ici est la plus connue MUSE/JSD, liée à la méthode informatique JSD, mais une version pour la méthode MERISE est actuellement en train de développement .

MUSE/JSD [Lim & Long 94] est une méthode pour l'intégration de l'approche facteurs humains à la méthode structurée classique d'analyse et de spécification des applications informatiques JSD (**J**ackson **S**ystem **D**evelopment) [Jackson 83]. MUSE propose un développement en parallèle par des équipes différentes, à savoir informaticiens et ergonomes, mais elles utilisent toutes les deux la structure de phases et la notation de modélisation de JSD pour exprimer leurs produits, ce qui facilite leur intégration dans les points de liaison prévus - voir figure 3.9.

La première phase, **Synthèse de la Conception,** comprend l'analyse et la spécification conceptuelle du système cible, à partir **a)** de l'explicitation des besoins de l'utilisateur (fonctions, nécessités futures et caractéristiques désirables) recueillies dans les documents (contrats ou documents informels) et dans les interviews avec l'utilisateur et **b)** des informations (objectifs, tâches courantes, sous-tâches, critiques et problèmes) sur les tâches générales du système existant et de systèmes analogues, obtenus explicitement dans la phase de **recueil et d'analyse d'informations** par étude des manuels, des protocoles verbaux ou observation et implicitement par l'expérience d'autres conceptions antérieures. Dans cette phase, on utilise de manière intégrée les **exigences** demandés explicitement par l'utilisateur ("*user requirements*") et les **besoins** de l'utilisateur ("*user needs*"), recueillies par l'analyse de la tâche. Pour la spécification du système cible, MUSE propose des points d'ancrage sur la méthode JSD. En pratique, MUSE est une méthode pour l'organisation et le recueil des connaissances en facteurs humains (Ergonomie et Psychologie Cognitive) lors de la conception d'un logiciel interactif.

17)

18)

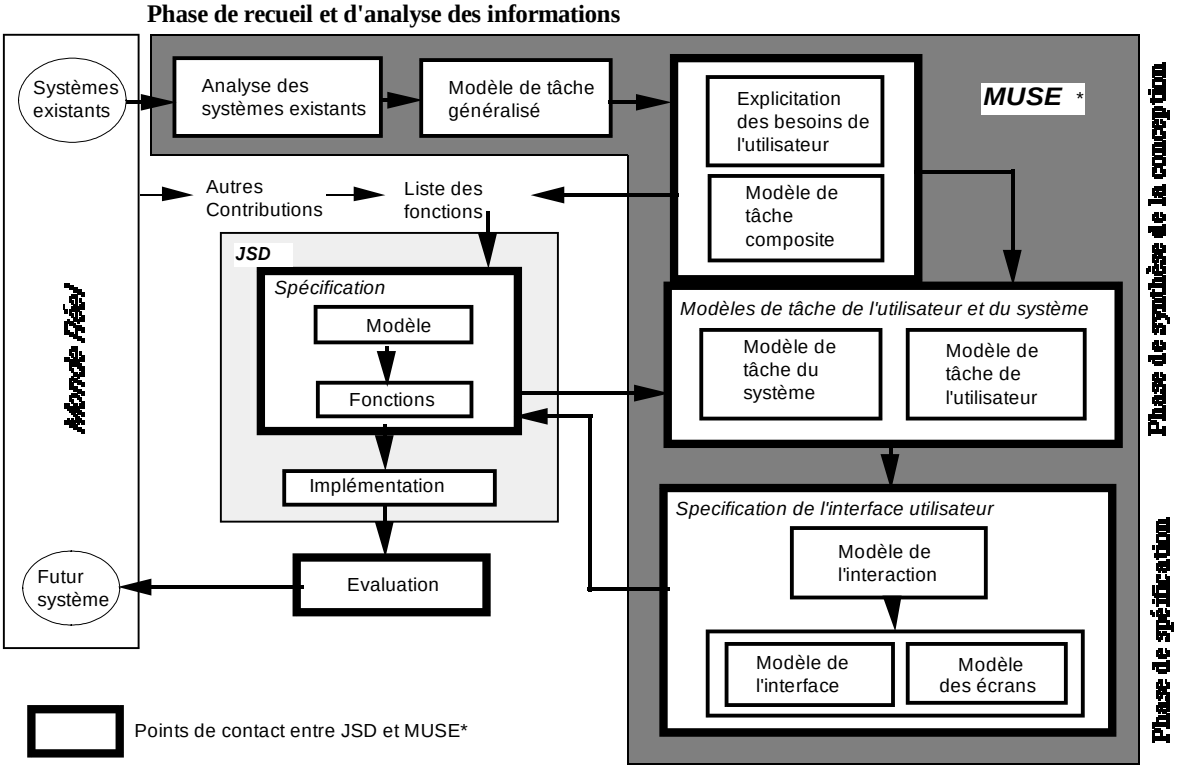


Figure 3.9 - La méthode MUSE/JSD

Avantages principaux de MUSE :

- Utilisation intégrée des besoins demandés directement par l'utilisateur et des besoins recueillis via l'analyse de la tâche ;

- Utilisation de la même notation (JSD) pour la représentation et la documentation des résultats (intermédiaires ou finaux) des deux méthodes (Facteurs Humains et GL) ;

Inconvénients principaux de MUSE :

- La méthode JSD - à la base de MUSE - est orientée surtout pour le développement structuré de systèmes et elle est inadéquate pour le développement orienté objets ;

- La participation de l'utilisateur dans le processus de décision n'est pas incitée ni prévue;

- C'est une méthode pour les ergonomes et pas pour les informaticiens, ce qui implique que les méthodes d'analyse de la tâche et les règles ergonomiques ne sont pas explicites : le savoir-faire des ergonomes est une condition requise pour sa démarche.

10.1.4ADEPT

ADEPT (*Advanced Design Environment for Prototyping with Task Models*) est une approche pour le développement de systèmes interactifs à partir d'une analyse des tâches actuelles de l'utilisateur. Cette analyse (sous la forme d'un modèle de tâches existantes) est utilisée d 'abord pour définir les tâches interactives futures réalisées avec le nouveau système (sous la forme d'un modèle de tâches envisagées) et après pour

concevoir un modèle abstrait mais détaillé de l'interface du système, comme montré dans la figure 3.10. Au contraire de la plupart des approches basées sur les tâches, qui ont plusieurs étapes, ADEPT adopte un processus en deux étapes pour passer du modèle des tâches envisagées à une interface exécutable : premièrement, la transition jusqu'à un modèle d'interface abstraite; deuxièmement, la traduction de ce modèle abstrait jusqu'à un prototype exécutable. Des prototypes d'outils ont été développés pour soutenir le développement avec ADEPT - en général des outils d'édition des modèles de tâches sauf un outil en particulier pour la génération automatique d'une interface exécutable à partir du modèle d'interface abstraite.

19)

20)

Figure 3.10 - Le processus de conception de ADEPT (d'après [P. Johnson et al. 95] et [Wilson & P. Johnson 96])

Analyse des tâches existantes

Le modèle de tâches existantes en ADEPT est la description des tâches et des rôles qui les réalisent, dans la perspective des buts et activités de l'utilisateur. La notation utilisée est TKS (*Task Knowledge Structures*), dont les principaux éléments sont les buts, sous-buts, procédures, actions et les objets impliqués pour la réalisation de la tâche. Un point important est le développement de modèles de tâches existantes en deux phases : premièrement, recueillir les caractéristiques individuelles et les variations dans la réalisation de tâches spécifiques ; deuxièmement, la construction d'un modèle composé de tâche, qui inclut les différentes façons qui ont été identifiées pour arriver aux buts.

Au début, ADEPT avait un seul modèle complémentaire au modèle de tâches existantes : le modèle de l'utilisateur. Le modèle de l'utilisateur est la description des caractéristiques (habiletés, âge, sexe, familiarité avec systèmes informatiques, etc) des individus ou groupes qui réalisent les tâches [P. Johnson et al. 95] et prenait la forme d'une liste de règles utilisée pendant la génération de l'interface exécutable. Aujourd'hui, d'autres aspects du domaine du problème sont aussi capturés et enregistrés sous la forme de besoins, contraintes de conception et idées de conception, mais sans suivre aucun modèle ou format de représentation [Wilson & P. Johnson 96]. Cependant, malgré l'affirmation que ces informations influencent les changements des tâches, la littérature sur ADEPT consultée ne montre aucune indication de comment ADEPT les utilise.

Définition des tâches Envisagées

Les tâches envisagées sont conçues à partir du modèle de tâches existantes (élaboré dans l'étape antérieure), la définition du problème basique à résoudre et les besoins et contraintes pour le système. Le concepteur doit identifier où et comment il peut introduire des changements pour arriver à une solution au problème, sans compromettre la qualité du travail. Cette définition ne spécifie pas comment l'interaction va arriver ou comment les informations seront montrés à l'écran mais surtout quelles tâches seront supportées, comment elles seront supportés et quelle est la raison des probables changements, où il y aura interaction et avec quel utilisateur.

21)

Modèle de l'Interface

Le modèle de l'interface est une description de haut niveau de l'interface, en termes d'objets d'interaction abstraits (y compris les actions associées et leur combinaison pour créer objets plus complexes) et d'information de dialogue (y compris les séquences d'actions). En fait, il consiste en une description de la structure de l'interface sans considérer les détails concrets d'implémentation.

Avantages principaux d'ADEPT:

- La participation de l'utilisateur est incitée surtout comme évaluateur ; en plus, une nouvelle version d'ADEPT plus participative est en cours de développement ;

Inconvénients principaux d'ADEPT

- Comme ADEPT n'est pas basée sur une méthode informatique, il n'est pas clair comment les critères de qualité orientés GL sont considérés, définis et respectés ;

- Le manque d'indications de comment utiliser les informations des besoins, contraintes et idées sur le système pendant la conception ;

10.1.5ICO

L'objectif du travail de [Bastide & Palanque 95] est de vérifier si le modèle de conception de l'application interactive soutien le modèle de tâche construit pendant la phase de définition des besoins. La modélisation de la tâche utilise le formalisme UAN (User Action Notation) [Hartson et al. 90] pour décrire le comportement d'interaction utilisateur-système. Le modèle de tâche décrit en UAN, pour la proposition de vérification, doit être traduit pour utiliser le formalisme de réseau de Petri. Le schéma de traduction est présenté en [Palanque et al. 95].

Pour la modélisation du système la notation orienté objet ICO (Interactive Cooperative Objects) est utilisée. L'ICO considère qu'un objet est composé de :

- une structure de données (attributs);

- un ensemble d'opérations (services internes ou services disponibles comme messages);

- une structure de contrôle d'objet (ObCS) pour modéliser le comportement, définie à l'aide d'un réseau de Petri de haut niveau;

- une présentation, pour décrire son apparence externe et qui est structuré comme un ensemble d'interacteurs (boutons, barre de défilement, etc).

Comme les deux descriptions - la description de la tâche et la description du

comportement du système - ont une base commune on peut, à travers des techniques formelles associées à la théorie des réseaux de Petri:

- les valider de manière isolée, c'est à dire, montrer que chaque ICO est capable de fournir ses services; ou

- les comparer et valider leur coopération . Pour évaluer la coopération avec les utilisateurs, chaque utilisateur est considéré comme un ICO.

Le principe basique est d'utiliser un protocole client-serveur pour assurer, pour chaque ICO, que l'ensemble de services demandés aux autres objets (appelé **demande**) est inclut dans l'ensemble des services offerts par eux (appelé **offre**). En particulier, on peut assurer que la demande de l'utilisateur est incluse dans l'offre du système.

Avantages principaux de ICO:

- Elle permet, à travers des techniques d'analyse mathématique de la théorie des Réseaux de Petri, une validation a priori de l'exactitude (*correcteness*) du système, particulièrement intéressante pour les systèmes réactifs critiques ;

- Elle utilise le même formalisme pour la description des tâches de l'utilisateur et du système ;

- Il existe déjà des outils logiciels disponibles pour l'automatisation de l'analyse mathématique .

Inconvénients principaux de ICO :

- La description formelle du modèle du système en utilisant ICO est très difficile à élaborer bien que le modèle de tâche soit le résultat d'une traduction à partir de UAN ;

- La notation formelle ICO n'est pas adaptée au niveau de comprehension de l'utilisateur ;

- Les tâches modélisées sont des tâches interactives et non des tâches existantes, donc cette approche ne sert pas à l'IB mais à la conception du système : les besoins doivent être déjà établis avant son utilisation.

- La participation de l'utilisateur dans le processus de décision n'est pas incitée ni prévue;

10.2Critères de comparaison

Comme nous l'avons vu, il n'y a pas à notre connaissance de classification des approches d'IB qui permette de les comparer et de choisir l'approche la plus adaptée à une situation en fonction du contexte de développement. En fait, la plus grande difficulté est l'hétérogénéité des sujets usuellement considérés comme faisant partie de l'IB. Pour cela, nous avons choisi de présenter la comparaison selon des critères associés aux efforts de recherche associés aux activités de l'IB.

Les efforts de recherche en IB ont, suivant la proposition de [Zave 95/, au moins deux dimensions : 1) problèmes de l'IB ; 2) contributions à la recherche en IB. Bien que la première dimension fasse l'objet d'un consensus, la deuxième est relativement

conflictuelle parce qu'elle peut être confondue avec autres sujets du GL [Zave 95].

Ainsi, nous proposons une classification basée sur les critères suivants:

- l'information - modèles, documents ou artefacts - utilisés/crées par les différentes approches comme données d'entrée, données temporaires ou données de sortie pendant le processus de l'IB. ;

- les alternatives de solution aux problèmes typiques de l'IB présentés par chaque approche ;

- les propriétés d'ordre général comme par exemple le type de l'utilisateur de l'approche (ergonome ou informaticien) et l'existence ou non d'outils de support.

En fait, il semble logique de retrouver les deux premiers critères au centre de toutes les approches, qui utilisent différents types d'information et créent des solutions alternatives (partielles ou totales) à quelques uns de ces problèmes dans le but de réaliser les activités de l'IB. Les propriétés d'ordre général servent à indiquer des caractéristiques de l'utilisation des approches.

10.2.1L'information

Comme nous avons vu au chapitre 1, il y a 3 types d'information pour l'IB :

- **l'Information contextuelle**, qui représente les connaissances sur l'univers et qui correspond au point de départ et aux informations en entrée du processus ;

- **l'information opérationnelle**, qui représente les connaissances utilisées pour le processus d'IB et qui correspond aux représentations intermédiaires du processus ;

- **l'information sur le système**, qui représente les connaissances spécifiques du système et qui correspond aux résultat du processus. Ici, il est particulièrement important d'identifier les composants structuraux (dialogue, présentation, sémantique) d'un système que l'approche manipule.

Le cas échéant, l'ensemble des informations est souvent associé à la fois à la méthode informatique de base de l'approche et à son domaine d'application . C'est pour cela que ces deux sont aussi pris en compte comme critères.

10.2.2Les alternatives de solution aux problèmes typiques de l'IB

Comme nous l'avons vu au chapitre 2, les principaux problèmes de l'IB sont :

- Soutien à la traçabilité

- Soutien à la Communication utilisateur-analyste

- Participation de l'Utilisateur dans le processus ;

- Soutien au '*Design Rationale*'

10.2.3Les propriétés d'ordre général

Les propriétés retenues pour la classification sont :

- l’utilisateur de l’approche (ergonome ou informaticien) ;
- l’existence ou non des outils de support .

Bien sur, ces propriétés dépendent des circonstances dans lesquelles l’approche est appliquée et ainsi peuvent devenir plus ou moins importants.

10.3Classification des approches d’integration

Le figure 3.11 présente la classification des approches d’intégration GL-IHM. Chaque approche est présentée selon les critères définis dans la section 3.1, de manière à pouvoir les comparer. Les informations citées sont les informations obtenues à partir de la description des approches présentes dans les références utilisées. Le symbole ‘ ?’ indique que l’information sur un critère n’est pas disponible ou n’est pas mentionnée dans les références consultées.

La variété de types d’**information contextuelle** est un indicateur des diverses interprétations de contextes. Les plus grandes quantités d’information contextuelle appartiennent aux approches DIANE, qui utilise les concepts attachés à Merise, et TRIDENT, qui modélise les objets métiers, les tâches mais aussi les caractéristiques des utilisateurs et leur environnement physique . MUSE et ADEPT modélisent surtout les tâches utilisateur existantes pendant que ICO ne modélise que les objets métiers.

L’**information opérationnelle** de la plupart des approches est surtout une information comportementale, comme les modèles de tâches (MUSE et ADEPT), le graphe d’enchaînement des activités (TRIDENT) et le modèle de dialogue avec les objets OPAC de DIANE.

Les approches basées sur une méthode informatique (MUSE, DIANE et TRIDENT) sont liées à des méthodes informatiques traditionnelles (voir le critère **méthode de base**), ce qui constitue une limitation assez importante aujourd’hui due à l’utilisation intensive du paradigme objet. ADEPT n’est pas attachée à une méthode spécifique et cela est remarqué même par ses auteurs comme une déficience pour la systématisation de sa démarche. ICO, par son rigueur mathématique et son déterminisme, est une approche très intéressante pour la conception et la vérification formelle a priori des propriétés du système mais son utilisabilité et utilité pour l’IB est faible.

En rapport aux critères relatifs aux problèmes fondamentales de l’IB, comme soutien à la communication utilisateur-analyste, le soutien à la traçabilité, la participation de l’utilisateur et le soutien au design rationale, toutes les approches sont très faibles . En effet, cela n’est pas une surprise parce qu’aucune approche n’a été conçu pour remplir ces critères.

Le domaine d’application le plus commun est la conception des systèmes d’information. L’exception est ICO, qui est applicable aux systèmes réactifs critiques (trafic aérien, p. ex), d’où la nécessité de plus formalité.

Par rapport au support d’outil, MUSE est le seule approche qui ne le possède pas.

Critères	**DIANE+**	**MUSE**	**ADEPT**	**TRIDENT**	**ICO**
Information Contextuelle	Les modèles de Merise : MCD, MCT, MOD, MOT ;	Modèle de Tâche Généralisé	Modèle de Tâche Existante (TKS), Modèle de	Modèle des objets Métiers (ER), Modèle de Tâche Interactive (TKS), profile des	Modèle à Objets

			l'Utilisateur	utilisateurs ; description de l'environnement physique	
Information Opérationnelle	Modèle de Dialogue et le modèle des objets OPAC	Modèle de Tâche Composite ; Modèle de Tâche Spécifique	Modèle de Tâche Future	Graphe d'enchaînement des activités	**?**
Information du Système	Dialogue, sémantique	Dialogue, sémantique	Dialogue, présentation, sémantique	Dialogue, présentation, sémantique	Dialogue, sémantique
Méthode de Base	Merise	JSD	non	ER	Réseau de Petri
Soutien à la Communication utilisateur-analyste	non	non	non	non	non
Soutien à l traçabilité	non	non	non	non	non
Participation de l'Utilisateur	passive	passive	active (nouvelle version)	passive	passive
Design Rationale	non	non	non	non	non
Domaine de Application (fiabilité des cas appliqués)	Systèmes d'Information	Systèmes d'Information	?	Systèmes d'Information	Systèmes Critiques
Suporté par Outil	oui	non	oui	oui	oui
Utilisateur	informaticien	ergonome	**?**	informaticien	informaticien

Figure 3.11 - Classification des approches d'integration GL-IHM

Sauf MUSE, conçue pour les ergonomes, et ADEPT, qui adopte la notion plus étendue de '*designer*' sans être précise, toutes les approches sont orientés vers l'utilisation par les informaticiens (voir le critère utilisateur).

11.Synthèse

Bien que les résultats obtenus par tous ces approches sont très satisfaisants, cette comparaison montre qu'il y a des lacunes importantes, notamment:

- il n'y a pas d'approches conçues spécifiquement pour l'IB des systèmes interactifs;

- l'information contextuelle modélisée est surtout basée sur les modèles des tâches et/ou modèles d'objet du domaine; cette information n'est pas assez riche pour représenter les aspects organisationnels du travail de l'utilisateur;

- les problèmes typiques de l'IB ne sont pas adéquatement traités;

- entre les diverses tentatives d'intégration entre les approches du GL et d'IHM présentées, il manque notamment la recherche sur l'intégration à une méthode orienté objet; car le paradigme orienté objet est le plus utilisé à l'heure actuelle pour le développement des applications informatiques et pour les outils de construction/maquettage d'interface utilisateur.

- il y a une grande difficulté pour définir exactement quels sont les besoins essentiels pour un système particulier sans savoir comment le système sera utilisé;

même si l'information du modèle de tâche est fondamentale et doit être prise en compte, pour déterminer les besoins du système (y compris les besoins fonctionnels et de l'interaction) il faut considérer l'interaction utilisateur-système dans une situation spécifique d'utilisation. Le fil conducteur du processus d'IB doit être à la fois prendre en compte l'information du modèle de tâche et être suffisamment dynamique pour permettre de envisager différentes situations d'utilisation possibles.

Notre travail est un effort pour remplir ces lacunes. Notre approche TAREFA sera présentée dans la partie suivante.

Partie II : La Proposition de TAREFA

Chapitre IV - TAREFA: Fondements et Vue d'Ensemble

12.Introduction

Dans la partie précédente, nous avons résumé quelques concepts relatifs aux systèmes interactifs et l'ingénierie des besoins. Nous avons également passé en revue quelques approches existantes pour l'IB des systèmes interactifs. Cette analyse essentiellement descriptive nous a permis de constater quelques déficiences dans ces approches et établir quelques axes pour la proposition de notre approche TAREFA, présenté dans cette partie en quatre chapitres.

Celui-ci est le premier chapitre et son but est présenter les fondements et une vue de l'ensemble de TAREFA.

Les chapitres suivants présentent avec plus de détailles l'approche TAREFA et les caractéristiques présentées seront plus approfondies. Le chapitre 5 présente la démarche et les modèles associés à la macroactivité d'Analyse. Le chapitre 6 présente la démarche et les modèles associés à la macroactivité de Synthèse. Le chapitre 7 présente le modèle de processus émergent proposé pour les activités coopératives.

Dans ce chapitre tout d'abord dans la section 2 nous présentons les fondements de

TAREFA : les concepts de besoins et facteurs de qualité des systèmes interactifs adoptés, et les caractéristiques principales de TAREFA, comme la richesse de modèles du contexte de travail, les propositions de solutions aux problèmes typiques de l'IB, l'adoption d'un concept de cas d'utilisation qui permet un lien fort avec le modèle de tâches et l'intégration à une méthode orientée objet du Génie Logiciel.

Ensuite dans la section 3 nous précisons les modèles adoptés par TAREFA pour la modélisation du contexte, des besoins et pour enregistrer les informations du processus.

Finalement, dans la section 4 nous présentons le processus de TAREFA et ses macroactivités Analyse et Synthèse. Ces macroactivités sont composées à la fois par des activités systématiques - activités qui sont réalisées notamment par l'analyste - et par des activités emergentes - qui impliquent la collaboration entre utilisateur et analystes ne suivant pas une démarche systématique.

13.Les Fondements de TAREFA

TAREFA[9] (TAsk based Requirements Engineering FrAmework) est une approche basée sur tâches pour l'Ingénierie des Besoins des Systèmes Interactifs. C'est en nous basant sur la définition de chacun de ces termes séparés puis sur leur signification associés que nous présenterons les fondements et le vue d'ensemble de TAREFA le long de ce chapitre.

L'Ingénierie des Besoins (*Requirements engineering*).a pour but la détermination des objectifs, fonctions et contraintes d'un système (cf. chapitre 2). Dans le cas des systèmes interactifs, cette détermination devient plus complexe du fait de la prise en compte à la fois des aspects techniques et humains inhérents à ce type des systèmes.

Un framework est une organisation d'éléments et la description de leur interaction. Notre framework est l'organisation d'un ensemble d'approches appliquées à l'Ingénierie des Besoins des systèmes Interactifs. Evidemment le framework repose sur un ensemble de principes, concepts, modèles et recommandations nécessaires à l'IB.

Un framework est dit 'basée sur tâches' selon la définition de Jean Vanderdonckt (voir chapitre 1) si le modèle de tâche est un concept fondamental pour son organisation et si les autres modèles utilisés sont soit un cadre contextuel soit le résultat d'une transformation/dérivation du modèle de tâche.

TAREFA est une approche de l'IB conçue spécifiquement pour les systèmes interactifs et basée sur l'intégration des concepts, modèles et techniques du GL et de l'IHM. Son but est de systématiser le processus de détermination et modélisation des besoins d'un système interactif cible, en particulier envisageant la spécification orientée objet de ce système pour permettre la continuité de la conception selon la méthodologie orientée objet Objectory. Cette systématisation consiste en d'abord comprendre la situation de travail existant et son contexte organisationnel pour après déterminer et représenter les besoins d'un système qui va le soutenir. TAREFA est donc une approche pour la détermination des besoins d'un système futur en tenant compte de la compréhension de l'existant.

Les paragraphes suivants discutent les concepts de besoins (section 2.1) et facteurs de qualité (section 2.2) des systèmes interactifs selon l'approche TAREFA.

TAREFA fait la distinction entre les **besoins** et les **facteurs de qualité**. Les **besoins** correspondent aux exigences (explicites ou implicites) spécifiques des utilisateurs alors que les **facteurs de qualité** correspondent aux propriétés générales de qualité désirées du logiciel interactif. Il y a diverses propositions de facteurs dans la littérature dont le

9 L'acronyme a été délibérément choisi : '**tarefa**' est le mot portugais utilisé couramment au Brésil pour designer '**tâche**'.

but est de fournir un cadre pour l'analyse et l'évaluation de la qualité des systèmes interactifs [Abowd et al. 92], [Gram & Cockton 95]. La différence basique que nous faisons entre les facteurs et les besoins est que les premiers sont établis de façon générique pour un ensemble ou une classe de systèmes alors que les deuxièmes sont établis très spécifiquement pour un système particulier. On peut, par exemple, déduire des besoins à partir des facteurs mais il n'est possible que d'induire un facteur à partir d'un ensemble des besoins de plusieurs systèmes. Evidemment ils sont interconnectées mais dans cette thèse nous nous concentrons surtout sur les **besoins**. Une typologie des facteurs est proposée - voir figure 4.3 - mais l'association des facteurs au système n'est pas approfondie. En effet, la définition précise des facteurs, leur mesure et leur évaluation est encore un sujet de recherche.

Après la revision du chapitre 3, nous pouvons comprendre les propositions basiques de TAREFA, qui sont presentées en détail dans les sections suivantes :

- enrichir la modélisation du contexte de réalisation des tâches (section 2.3);
- proposer solutions aux problèmes typiques de l'IB (section 2.4);
- adoption d'un concept dynamique pour être le fil conducteur du processus de l'IB, en permettant de à la fois d'utiliser l'information du modèle de tâche et de considérer l'interaction utilisateur-système dans une situation specifique d'utilisation. Le concept dynamique adopté est le concept de cas d'utilisation (section 2.5) ;
- l'intégration à une méthode orientée objet (section 2.6).

13.1Les Besoins

Avant de commencer à discuter les détails de TAREFA, on doit comprendre mieux la notion de 'besoin'. Nous avons défini dans le chapitre 2 un besoin comme un prédicat sur les buts à atteindre par le système, c'est à dire "une condition nécessaire pour remplir un certain objectif". Dans cette section, nous proposons les notions adoptées par TAREFA de type des besoins et attributs de besoins.

13.1.1Types de Besoins

Une bonne compréhension des besoins commence par l'identification des besoins isolés de leur origine. Cela veut dire qu'il faut manipuler les besoins comme des entités distingués et non comme des 'strings' composants d'une description. Les besoins isolés peuvent être réunis et organisés pour trouver similarités et différences.

Nous proposons d'organiser des besoins en types, comme proposé aussi par [Kaindl 95a]. Ces types possèdent des propriétés - appelés attributs, comme dans [Thompson 94] et forment les catégories d'une typologie, comme dans [Kaindl 95b] et dans [Thompson 94].

Le principal avantage d'un typologie est l'organisation des besoins en une structure de façon à orienter ou au moins faciliter le processus de leur détermination. Organiser les besoins en types nous permet aussi de découvrir des besoins similaires ou conflictuels des systèmes qui appartiennent à un même domaine de l'application et de les réutiliser dans un autre système.

Il est possible d'expliciter l'existence d'une association entre 2 types de besoins

(p.ex., l'association entre une situation d'utilisation et une fonction du système, ou un problème de sécurité en rapport à une information spécifique). On peut aussi associer un type de besoin à une classe d'objet du domaine du problème - association de traçabilité ou management de changement.

Les types des besoins de TAREFA peuvent être utilisés de deux manières:

1. Utiliser directement les types pré-definis dans la typologie de TAREFA.

2. Raffiner/Spécialiser les types pré-definis pour créer des types plus spécifiques des besoins ;

13.1.2Typologie des Besoins

Comme nous avons vu dans le chapitre 2, il y différents types de taxinomie des besoins. Nous avons décidé de proposer une taxinomie nouvelle (voir figure 4.1) parce que les autres ne sont pas adéquates pour les systèmes interactifs - surtout en rapport aux besoins de l'interaction et facteurs de l'utilisabilité. Notre typologie est fondée sur 2 principes:

- la distinction entre les **besoins** et les **facteurs de qualité ;**

- la distinction des besoins **du système,** besoins **de l'interaction** et besoins **de l'environnement.**

Les **besoins** correspondent aux exigences (explicites ou implicites) spécifiques des utilisateurs alors que les **facteurs** correspondent aux propriétés générales de qualité désirées du logiciel interactif.

Suivant une pratique courante en Ingénierie, dans notre typologie de besoins (voir figure 4.1) les besoins du système sont distincts des besoins de son environnement. En plus, les besoins de l'interaction sont distingués des besoins du système. Comme nous avons montré dans le chapitre 1, l'interaction n'est pas une caractéristique inhérente du système mais une relation entre les utilisateurs et le système donc ses besoins ne font pas partie des besoins du système.

Les **Besoins Fonctionnels du Système** sont décomposés en deux types des besoins :

1. les besoins fonctionnels dynamiques, qui indiquent la fonctionnalité (composant dynamique) du système - ses services et fonctions ;

2. les besoins fonctionnels statiques, qui représentent les besoins sur la structure orientée données (composant statique) du système - les objets et leurs propriétés.

Les **Besoins d'Interaction,** qui représentent les besoins sur le comportement de l'interaction des utilisateurs avec le système - quels services du système sont disponibles à l'utilisateur, quand l'utilisateur peut fournir (ou obtenir) des informations au (du) système et en quelle ordonnancement;

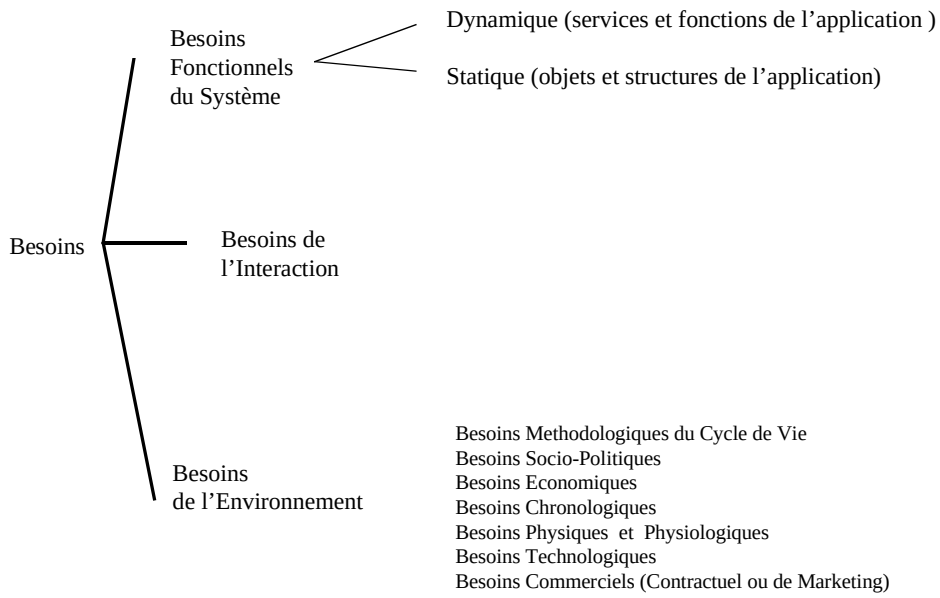


Figure 4.1 - Typologie des Besoins de TAREFA

Les **Besoins de l'Environnement** réunissent plusieurs types de besoins connus, traditionnellement groupés comme besoins non-fonctionnels :

- besoins socio-politiques, relatifs aux demandes du groupe ou aux décisions politiques de l'organisation;
- économiques , relatifs au budget pour la conception du système ;
- du cycle de vie, relatifs à l'adoption d'une méthodologie ou paradigme pour les phases après l'analyse;
- chronologiques, relatifs au temps établi pour la fin du projet (chronogramme) ou pour les phases intermédiaires (*milestones)*;
- physiques, relatifs à la distribution géographique ou spatiale ; et physiologiques relatifs au profil particulier des utilisateurs;
- technologiques, relatifs aux contraintes de matériel et logiciel à utiliser: base de données, système d'exploitation, périphériques d'entrée/sortie, etc);
- commerciales, pour respecter les contrats et les critères de marketing.

Ces différents types de besoins doivent être recueillis selon différentes techniques. En général, l'utilisateur demande explicitement un ensemble des besoins sur la fonctionnalité du système. Cependant, l'adoption de demande explicite des besoins par l'utilisateur n'implique pas forcement l'obtention d'un ensemble complète des besoins. Il y a des besoins qui ne sont pas reconnus ou exprimés par l'utilisateur, comme les **besoins implicites** ("*user needs*"), recueillies en général via analyse de la tâche ou d'autres techniques pour envisager des besoins fonctionnels dans un contexte d'utilisation. En plus, il y a des **besoins imposés par l'organisation**, qui sont identifiés en général par l'analyse des règles organisationnelles attachées aux activités de travail.

Les informaticiens ont souvent une fausse impression: plus fonctions fournis, plus le système sera accepté par l'utilisateur. Cependant, l'acceptation d'un système peut

dépendre plutôt de la qualité de réalisation de quelques fonctions que de la quantité des fonctions disponibles. En fait, [Nickerson 81] a remarqué que la fonctionnalité n'est qu'un facteur pour l'acceptation. La plupart des facteurs, en fait, correspondent plus à la manière d'utiliser le système que au fait d'une fonction particulier être ou pas disponible. Cela est à l'origine des besoins de l'interaction utilisateur-système et est réfléchi dans la définition du comportement des tâches interactives.

Les **besoins d'interaction** sont identifiés par la projection de l'utilisation du système ou par expérimentation avec les premiers prototypes. Les besoins d'interaction servent aussi à envisager des besoins fonctionnels additionnels.

Les **besoins de l'environnement** servent à la fois comme des ressources et des contraintes au développement et à l'utilisation du système parce qu'ils vont délimiter l'espace de alternatives possibles.

Les besoins naturellement ont des relations avec les autres informations du développement. TAREFA considère en particulier les relations suivantes [Thompson 1994] :

- entre besoins (*requirement to requirement*), pour représenter leurs interdépendances ou leur complémentarité;

- entre un besoin et un élément d'un modèle de TAREFA (p.ex. un besoin associé à un schéma de *design rationale*), très utilisée pour la traçabilité;

- entre un besoin et un composant spécifique de l'architecture du système, pour permettre d'identifier les éléments du système qui sont affectés par un changement des besoins, aussi utilisée pour la traçabilité.

13.1.3Attributs des Besoins

Attributs sont des propriétés attachés aux besoins. Les attributs définis pour les besoins de TAREFA et leur description sont montrés dans la figure 4.2. La colonne gauche montre les attributs et la colonne droite leur description. Ensuite, une explication des attributs est presentée.

Un besoin est identifié par un code (**Id Unique**) et un **nom**.

L'attribut **type des besoins** fait reference aux elements de la typologie presentée dans la section precedente. Les valeurs possibles pour cet atribut sont donc "besoin fonctionnel dynamique", "besoin de l'interaction", etc.

L'attribut **origine** fait reference aux differentes sources des besoins, une consequence du fait qu'ils doivent être obtenus selon différentes techniques de recueil. Les valeurs possibles pour cet attribut sont:

- Besoin de l'Utilisateur (BU), qui peuvent être: Besoins demandés Explicitement par les utilisateurs (BUE), Besoins Implicitement identifiés par l'Analyse du Travail (BUI) et Besoins Imposés par l'Organisation (BUO)

- Besoin du Système (BS), qui peuvent être: Besoins Additionnels Découverts (BA) determinés par l'intermediare des cas d'utilisation, Facteurs de Qualité (FQ) dérivés de la typologie des facteurs de qualité et Contraintes Technologiques (CT) dérivés du modèle de plataforme.

Il faut remarquer la distinction entre les attributs 'origine' et 'type': Origine enregistre les informations orientés traçabilité lorsque le Type enregistre la catégorie de la typologie du besoin.

L'attribut **contributions**, inspiré par le travail de [Gotel & Filkenstein 94b], montre les rapports de contribution du besoin:

- **principal**: montre le nom de la personne (un des acteurs ou quelqu'un de l'organisation) qui va valider le besoin;

- **auteur**: montre le nom de la personne (un des acteurs ou quelqu'un de l'organisation) qui a crée/proposé le besoin;

- **source**: montre le nom de la personne (ou d'un département de l'organisation) qui connaît plus de détails sur ce besoin , ou le titre du document où se trouve plus d'informations sur ce besoin.

Ces informations sont très importantes pour la pré-traçabilité des besoins surtout pour les SIIM.

Un besoin peut être décrit informellement (attribut **Description Informelle**), en général en langage naturel non structuré, ou formellement (**attribut Description Formelle**), qui ne doit pas être obligatoirement présente. Cependant, la description informelle est très importante même si la description formelle existe. Il faut remarquer que TAREFA n'impose pas une notation formelle particulière. L'analyste peut choisir la plus adéquate à son modélisation.

Les attributs **version** et **date** enregistrent l'information de configuration des besoins, information très importante pour la gestion des changements de configuration et pour l'évolution des besoins.

Les attributs **Affecté par**, **Fait reference à** et **Utilisé par** representent des informations de connexion un besoin par rapport aux autres besoins, en indiquant respectivement les besoins qui peuvent lui affecter, les besoins qui sont utilisés par lui (p. ex. : les objets/informations utilisés par les services; les services utilisés par les cas d'utilisation, etc) et les besoins qui l'utilisent (p. ex. les services qui utilisent une information).

Finalement, l'attribut de **priorité** indique le poids du besoin en rapport à l'acceptation du système. Les valeurs possibles sont:

- **indispensable**: si ce besoin n’est pas satisfait, le système sera refusé;

- **désirable**: la satisfaction de ce besoin est obligatoire mais n'implique pas le refus du système;

- **pertinent**: la satisfaction de ce besoin est facultative;

- **non pertinent**: le besoin ne s'applique pas au système en question.

Le choix de cette priorité est un compromis résultat de la négociation entre les acteurs, mais en général, l'attribution de priorité suit le standard suivant:

- les besoins indispensables sont souvent les besoins demandés explicitement par l'utilisateur ou imposés par l'organisation;

- les besoins désirables sont souvent les besoins identifiés par l'analyse de la tâche ou par l’intermédiaire des cas d'utilisation singuliers ou concrets (voir chapitre 6);

- les besoins pertinents sont souvent les besoins spécifiés pour d'autres systèmes dans le même domaine ou dans le même contexte organisationnel;

- les besoins non pertinents, sont les besoins jugés non nécessaires par les utilisateurs ou les besoins non mentionnés pendant tout le processus de détermination.

Attributs	Description et Valeurs Possibles
Id Unique	identificateur unique du besoin
Nom	nom du besoin
Type de Besoins	Type du besoin conformément les catégories de la taxonomie
Origine	L'origine du besoin: • Besoin de l'Utilisateur (BU), qui peuvent être: Besoins demandés Explicitement par les utilisateurs (BUE), Besoins Implicitement identifiés par l'Analyse du Travail (BUI) et Besoins Imposés par l'Organisation (BUO) • Besoin du Système (BS), qui peuvent être: Besoins Additionnels Découverts (BA), Facteurs de Qualité (FQ) et Contraintes Technologiques (CT)
Contributions	rapport de contributions **principal**: Nom de la personne **auteur**: Nom de la personne **source**: Nom de la personne ou titre du document
Description Informelle	description informelle du besoin
Description Formelle	description formelle du besoin.
Version - date	Numéro da version du besoin et date de la dernière définition
Affecté par	Identificateurs des besoins qui, si changés, peuvent affecter ce besoin
Fait référence à:	Identificateurs des besoins qui sont utilisés par celui-ci
Utilisé par:	Identificateurs des besoins qui utilisent celui-ci
Priorité	Poids du Besoin en rapport à l'acceptation: **indispensable**, **désirable,** **pertinent**, **non pertinent**

Figure 4.2 - Attributs des besoins de TAREFA

13.2Les Facteurs de Qualité

La taxinomie des facteurs de qualité (voir figure 4.3) suit la distinction des **facteurs de qualité du système**, **facteurs de l'utilisabilité** et **facteurs de qualité du processus de développement du système**.

Les facteurs de qualité du système sont des facteurs relatifs à l'opération du système [McCall 77], comme par exemple l'exactitude (*correctness*), la confiabilité, l'efficience et l'integrité.

Les facteurs de l'utilisabilité sont des facteurs relatifs à la qualité de l'interaction utilisateur-système. Les facteurs d'utilisabilité présentes dans la typologie des facteurs de qualité de TAREFA sont les facteurs proposés par G. Abowd, J. Coutaz et L. Nigay [Abowd et al. 92], dont une définition est présentée dans le chapitre I. En bref, les facteurs d'utilisabilité sont classés en trois groupes:

- facilité d'apprentissage;
- flexibilité d'interaction;
- robustesse d'interaction.

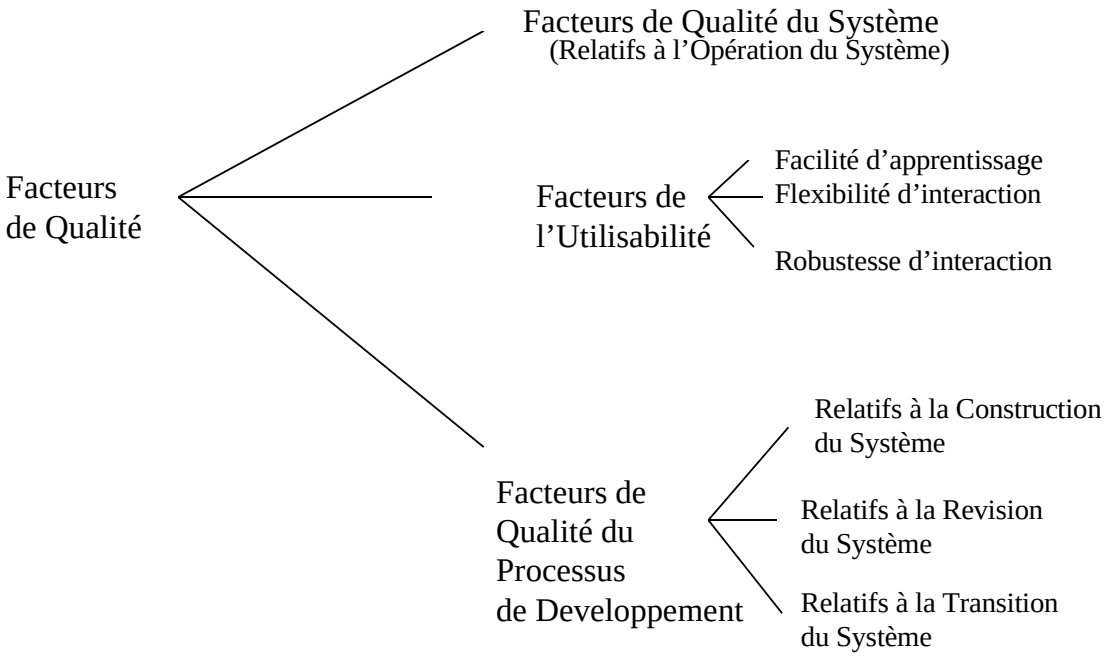


Figure 4.3 - Taxonomie des Facteurs de Qualité de TAREFA

Les facteurs de qualité du processus du dévelopment sont classés aussi en trois groupes:

1. les facteurs relatifs à la construction du système [Ghezzi 91], comme par exemple la verificabilité (verificability), l'evolutibilité (evolutibility), la sécurité (safety), compréhensibilité (understandability);

2. les facteurs relatifs à la revision du système [McCall 77], comme par exemple la repairabilité (repairability), la flexibilité (flexibility) et la testabilité (testability);

3. les facteurs relatifs à la transition du système [McCall 77], comme par exemple la portabilité, la reutilisabilité et l'interoperabilité (interoperability).

En fait, ces facteurs sont fortement liés aux besoins correspondants:

- les facteurs du système sont associés aux besoins fonctionnels du système;

- les facteurs de utilisabilité sont associés surtout aux besoins d'interaction mais parfois aussi aux besoins de l'environnement.

- les facteurs de qualité du processus sont associés aux besoins de l'environnement; bien sur, il y a facteurs qui ont importance pendant tout le processus (comme testabilité), alors qu'il y en a d'autres que sont plus importants dans une phase spécifique (comme interoperabilité);

L'intérêt de la typologie est organiser les facteurs de qualité dans une structure qui permet de fournir des valeurs par défaut lors de la définition des besoins. Un déterminé facteur de qualité peut être typiquement attaché à un besoin et ce fait peut être indiqué comme une orientation à suivre pendant la détermination des besoins. Bien évidemment, donner l'ensemble de ces valeurs par défaut nécessite la mise en oeuvre de TAREFA sur plusieurs études de cas, ce qui est envisagé comme un prolongement de cette thèse.

13.3Richesse des modèles de contexte de travail

TAREFA manipule les 3 types d'information pour l'IB, à savoir, l'Information contextuelle, l'information opérationnelle et l'information sur le système.

En particulier , TAREFA propose un ensemble de modèles pour permettre le recueil de la plus grande quantité de l'information contextuelle et avoir une description plus riche du contexte de travail.

Un système informatique est un sous système technique d'un système socio-technique majeur. La portion de ce système socio-technique qui nous intéresse définit ce qu'on appelle l'**univers du Problème.** Dans le sous système social il y a tous ceux qui ont un rapport avec cet Univers, qui sont appelés **les acteurs** de l'Univers. En bref, l'Univers du Problème, c'est le contexte général dans lequel le système informatique devra être développé et dans lequel le système informatique va fonctionner. C'est pour cela que TAREFA adopte le terme '**Information Contextuelle**' pour l'ensemble des connaissances de l'univers du problème.

Comme nos l'avons vu dans la Synthèse du chapitre 3; l'information contextuelle modélisée par la plupart des approches décrites est surtout basée sur les modèles des tâches et/ou modèles d'objet du domaine. La conception de systèmes interactifs demande une compréhension non seulement du domaine de l'application du système mais aussi du contexte de travail où le système va être utilisé. Ainsi, une représentation plus complète de ce contexte est nécessaire comme montré dans les articles de [CACM 95].

L'information contextuelle dans le sens adopté par TAREFA est constitué par un ensemble de modèles :

- un modèle ontologique du vocabulaire de l'univers du problème ;
- les modèles de tâches de travail existantes ;
- un modèle de contexte organisationnel dont les tâches font partie ;
- un modèle des objets métiers du domaine ;
- un modèle de plateforme.

Ces modèles sont introduits dans la section 3 de ce chapitre et présentés avec plus de détails dans le chapitre V.

13.4Les propositions de solutions aux problèmes typiques de l'IB

TAREFA doit proposer des solutions aux problèmes typiques de l'IB, à savoir:

- pour le problème de la traçabilité, les relations explicites entre l'information contextuelle, l'information opérationnel et les besoins du système permettent de suivre le 'trace' de chaque information utilisée et maintenir la consistance entre les informations même après les inévitables changements;
- pour le problème de la Communication utilisateur-analyste, le modèle ontologique du vocabulaire du domaine permet l'unification de terminologie à la base d'une communication directe efficace;

- pour le problème de Participation de l'Utilisateur, TAREFA adopte un type de participation active de l'utilisateur; en particulier, le soutien à la coopération entre l'analyste et l'utilisateur - compris les discussions, la négociation et les décisions consensuelles - est fourni par un modèle de processus émergent;

- pour le problème de représenter le *design rationale*, le modèle de *design rationale* de TAREFA sert à enregistrer l'historique de la démarche non-coopérative, y compris les problèmes, les décisions prises par l'analyste et les arguments associés.

13.5L'utilisation de cas d'utilisation

Dans l'approche TAREFA, les cas d'utilisation conduisent le processus de détermination des besoins. Ce choix s'explique par diverses raisons :

- l'origine multidisciplinaire des cas d'utilisation est très prometteuse pour permettre de capturer la nature polymorphique des besoins ;

- son organisation narrative permet une compréhension facile par les utilisateurs;

- les cas d'utilisation permettent d'identifier le parallèle entre les activités existantes et les activités futures : considérer les activités typiques et les activités inhabituelles (associés à des incidents ou à des exceptions) qui font partie du scénario de travail et qui peuvent être modifiés par le nouveau système;

- les cas d'utilisation aident à la généralisation des histoires des utilisateurs sur leurs activités en permettant d'incorporer des événements spécifiques et de mélanger les événements de différentes histoires;

- la granularité des cas d'utilisation est flexible : ils peuvent être définis à différents niveaux d'abstraction, en permettant de ne considérer que les éléments relevants à chaque niveau;

- sa modélisation partielle du système permet l'expression de différents points de vue et par conséquence l'élaboration de conjectures et de schémas cognitifs associés à chaque point de vue.

Les cas d'utilisation représentent une situation possible d'utilisation future du système, en permettant à la fois une systématisation rationnelle et une vision ('*insight*') intuitive sur les buts et la façon de les atteindre. En fait, la Psychologie Cognitive a déjà démontré que la connaissance humaine est organisée en '*chunks*', qui reconnaissent et répondent à des situations spécifiques [Anderson 82]. La résolution des problèmes a tendance à être hautement située, c'est à dire, les personnes réagissent aux situations qui apparaissent plutôt que d'exécuter les plans élaborés auparavant [Suchman 87]. Les cas d'utilisation sont des modèles partiels pour élaborer, représenter et réfléchir sur différentes situations dans le but de comprendre et de formuler les buts du système à développer.

En fait, les cas d'utilisation sont une manière diffèrente de considérer le processus de l'IB: les approches conventionnelles sont spécialement adéquates pour les problèmes bien délimités et définis, alors que les cas d'utilisation permettent de manipuler des

problèmes vagues ou mal définis en situations qui changent souvent ou qui présentent des conflits [Aboulafia et al. 94], [Leite 97].

Une contribution originale de TAREFA est que **les cas d'utilisation de TAREFA sont construits à partir des modèles des tâches minimales**, qui sont un type particulier de modèle de tâche (voir chapitre V).

Spécifier les besoins d'un système à partir d'un modèle de tâches existantes n'est pas un travail simple. Il faut considérer qu'un système va proposer des nouvelles actions (interactives) associées aux tâches pour arriver aux buts. Au contraire de [Normand 94], qui utilise une hiérarchie d'abstraction des modèles de tâche comme le fil conducteur pour la spécification des tâches interactives futures à partir des tâches existantes, nous sommes d'accord avec [Jacobson 92], [Carrol 95] et [Rosson & Carroll 95], qui mettent le concept de modèle de cas d'utilisation comme le fil conducteur du développement, en particulier pour l'étape d'Ingenierie des besoins.

TAREFA propose l'utilisation synergique de ces deux techniques pour la détermination des besoins, usuellement utilisés séparément : l'analyse de la tâche et l'approche des cas d'utilisation.

En fait, une critique souvent faite aux méthodes utilisant l'analyse de la tâche est que les connaissances tacites non présentes dans le modèle de tâche ne peuvent pas être considérées. En fait, les approches orientées tâches adoptent le prototypage pour essayer de remplir cette déficience mais sans une processus d'expérimentation systématique des prototypes.

Une critique souvent faite aux approches qui n'adoptent que les cas d'utilisation (voir par exemple [Carrol 95]) est que son utilisation sert plutot à raffiner la conception qu'à générer des besoins ou des connaissances sur le domaine. L'utilisateur, stimulé par le cas d'utilisation, peut identifier les erreurs de conception de l'analyste et lui suggerer modifications. Cependant, malgré la nécessité d'initier le procédé avec un "bon" essai (qui est en général très difficile à définir), la création des cas d'utilisation n'est pas systematisée comme la création des prototypes dans les approchés orientés tâches.

La combinaison de ces deux techniques dans TAREFA se propose à réduire les risques dans l'ingénierie des besoins de logiciels interactifs et consiste en un procédé interactif de construction des cas d'utilisation à partir des modèles de tâches (de la part de ceux qui le développent), d'expérimentation et d'évaluation du cas d'utilisation (par l'utilisateur) et de révision (par ceux qui le développent) des cas d'utilisation d'un système qui vise à satisfaire les besoins de l'utilisateur.

Cette combinaison est donc un procédé à la fois systématique et expérimental d'*essai-erreur*, où l'on essaie d'éclaircir, par les multiples interactions concepteurs-utilisateurs, quelques points obscurs concernant le système, tels que les besoins précis, mais aussi les styles préférés d'interface avec l'utilisateur et les limitations de l'environnement.

13.6L'intégration de TAREFA à la méthode orientée objet Objectory

TAREFA s'intègre à une méthodologie de développement de logiciel orientée objet parce que le paradigme orienté objet est le plus utilisé à l'heure actuelle pour le développement des applications informatiques et pour les outils de construction/maquettage d'interface utilisateur.

Nous avons choisie d'intégrer l'approche TAREFA à la méthode Objectory. Comme nous avons vu au chapitre 2, la méthode Objectory est basée sur un approche conduite par cas d'utilisation pour la conception, réalisation et teste de systèmes. C'est pour cela

que nous l'avons choisie pour l'intégration avec TAREFA, puisque TAREFA utilise aussi les cas d'utilisation comme un concept fondamentale. Les détails de cette intégration sont présentés dans cette section.

22)

23)

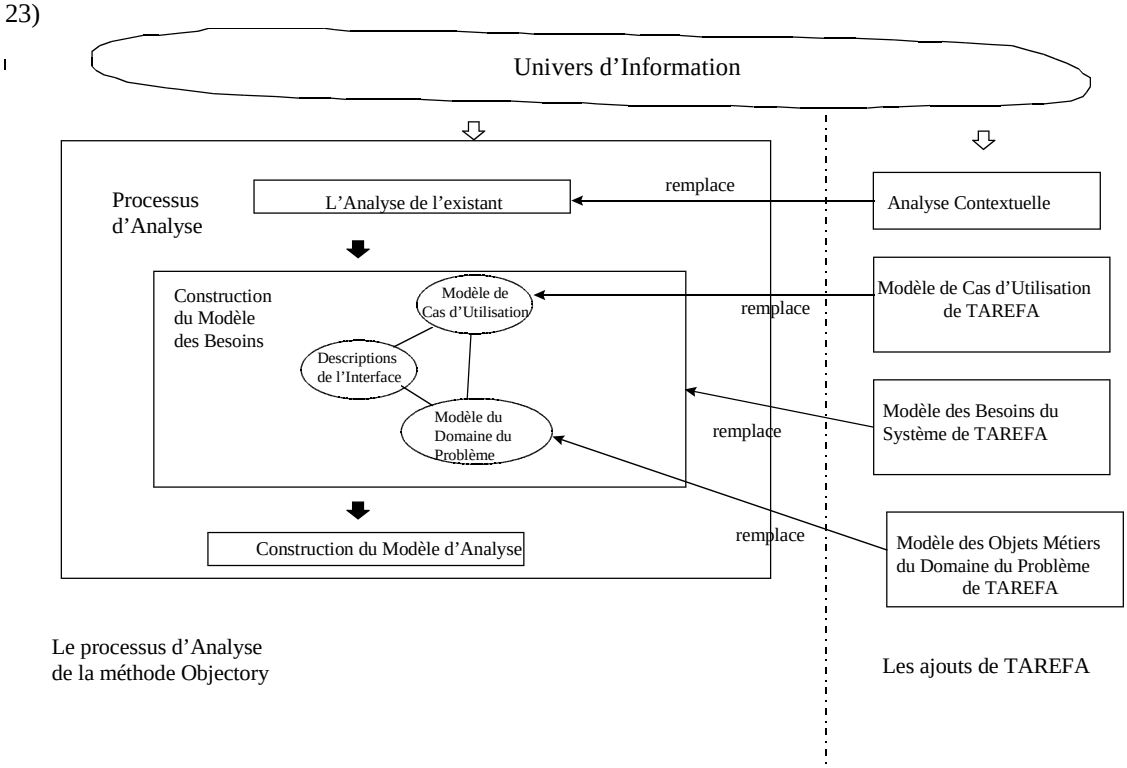


Figure 4.4 - Integration de TAREFA au processus d'Analyse de la méthode Objectory

La proposition d'intégration de TAREFA-Objectory est la suivante, cf. Figure 4.4:

1. Le processus d'Analyse de Objectory est modifié pour s'intégrer avec TAREFA. La figure 4.5 résume une comparaison entre les deux - les items marqués par '+' sont considérés comme point positifs (avantages) et les items marqués par '-' sont considérés comme points négatifs (inconvénients).En résumé :

   - L'Analyse Contextuelle de TAREFA remplace l'analyse de l'existant de Objectory

   - Le modèle de cas d'utilisation de TAREFA remplace le modèle de cas d'utilisation de Objectory;

   - Le modèle des Objets Métiers de TAREFA remplace le modèle des objets du domaine du problème de Objectory;

   - Le modèle de besoins de TAREFA remplace le modèle des besoins (*Requirements model*) de Objectory

2. Les processus de construction et de teste de Objectory et les autres activités d'Analyse non mentionnées ci-dessus ne sont pas modifiés;

24)

Aspects	Objectory	Integration Objectory - TAREFA
1. Cas d'Utilisation	Modèle de cas d'utilisation (Use case model) avec: (-) Problème de Granularité: Taille et niveau d'abstraction de chaque 'use case' sont choisis arbitrairement par l'analyste (-) Problème de Liaisons entre les 'Use Case' ('*extends*' and '*uses*' sont des liaisons de composition et non des liaisons de 'contrôle') (-) Le contexte dans lequel le 'use case' arrive n'est pas décrit (+) Notation informelle, ce qui facilite une communication utilisateur-analyste (+) Les 'use cases' sont utilisés pendant tout le développement (analyse, construction et teste)	Modèle de cas d'utilisation avec: (+) quatre niveaux d'abstraction: • essentiel, dont la construction est dérivée d'une hiérarchie but/sous-but du modèle de tâche minimale; • téléologique, où à chaque pair (but/obstacle) il y a un cas d'utilisation singulier à considérer; • opérationnel, pour décrire le cas d'utilisation en termes de classes d'objets, services et comportement du dialogue • concret, pour décrire une situation spécifique (scénario particulier) (+) Définition explicite de la dépendance et du ordonnancement entre les 'use cases' et leurs actions (+) Définition explicite de: * contexte organisationnel des cas d'utilisation à travers le modèle TOCO (+) La notation informelle initiale est itérativement formalisée • d'abord par une structure de grammaire; • après par une notation rigoureuse (des tableaux) (+) Les cas d'utilisation sont en plus utilisés pour la verification et la validation de la spécification, en addition à l'utilisation de Objectory
2. Objets du Domaine du Problème	Noms et concepts utilisés dans les descriptions de l'utilisateur sont des classes candidates	Les classes candidates sont: • les objets utilisés dans les modèles de tâche; • les signes de catégorie 'Agent' et 'Objet' du modèle ontologique; • les concepts du domaine du modèle de contexte organisationnel; • les objets métier du domaine du problème
3. Types d'Objets	(+) Le Modèle d'Analyse propose une distinction qui est à la base pour la stabilité de l'architecture de l'application: • Entity Object • Control Object • Interface Object	(+) En plus à cette distinction, TAREFA propose la notion d'objet **client** et **serveur** d'un service au niveau opérationnel;;

Figure 4.5 - Comparaison entre l'Integration de TAREFA-Objectory et le processus d'Analyse de la méthode Objectory

14.Les Modèles de TAREFA

Le fossé ('*gap*') entre l'analyse et la conception des systèmes interactifs peut être exprimé, selon [Cockton et al. 96], comme le fossé entre l'information contextuelle et le modèle des besoins du système interactif, voir figure 4.6. Les modèles du contexte representent l'information contextuelle et servent à modéliser les aspects de l'espace problème. Les modèle des besoins servent à représenter les besoins qui doivent être pris en compte par la classe de systèmes appartenant à l'espace solution.

TAREFA est une approche pour 'traverser' ce fossé, c'est à dire, pour établir connexions et permettre transformations systématiques entre l'information contextuelle et le modèle des besoins du système, en passant évidemment par la détermination de ses besoins. Cela exige la compréhension des descriptions contextuelles et des modèles des systèmes, ainsi que une précise détermination des modèles intermédiaires pour guider, structurer et enregistrer les transformations entre les deux espaces ci-dessus.

L'**Information contextuelle** de TAREFA est constituée par un ensemble des

modèles interconnectés pour représenter l'information contextuelle, voir figure 4.6 :

- le modèle ontologique, pour modéliser l'information sur le vocabulaire du domaine ;
- les modèles de tâches, pour modéliser les tâches de travail existantes ;
- un modèle de contexte organisationnel, pour représenter les aspects organisationnels du travail de l'utilisateur ;
- un modèle des objets métiers, pour modéliser les objets métiers du domaine ;
- un modèle de plateforme, pour modèliser les moyens disponibles dans la configuration existante.

Le modèle des besoins du système interactif corresponde au modèle utilisé pour représenter les besoins sur la structure, l'apparence et le comportement (interne et externe) du système.

Le modèle des besoins de TAREFA est son principal résultat final et représente les besoins d'ingénierie du système, c'est à dire, la spécification des caractéristiques que le système doit posséder et qui sont les buts à atteindre lors de sa réalisation. Selon [Kaindl 95], OMT [Rumbaugh 91] [Glinz 95], une description complète des caractéristiques d'un système implique trois types de spécification :

1. spécification de la structure du système (appelée respectivement description structurelle par [Kaindl 95], modèle objet par [Rumbaugh 91] et spécification des données du système par [Glinz 95]) ;
2. spécification de la fonctionnalité du système (appelée respectivement description des intentions et fonctions par [Kaindl 95], modèle fonctionnel par [Rumbaugh 91] et spécification de la fonctionnalité du système par [Glinz 95]) ;
3. spécification des interactions du système (appelée respectivement description comportementale par [Kaindl 95], modèle dynamique par [Rumbaugh 91] et modèle des scénarios par [Glinz 95]) .

Dans l'approche TAREFA les besoins sont structurés autour des cas d'utilisation. A un service (ou épisode) faisant partie d'un cas d'utilisation (dont la description sera présentée dans le chapitre VI) correspond un ensemble des besoins et facteurs de qualité qu'on peut retrouver dans d'autres cas d'utilisation. De manière simplifié, on peut dire qu'un besoin (ou facteur) est attaché à un cas d'utilisation. La structure d'un besoin est décrit par un ensemble d'attributs de la façon montrée dans la section 2.1 dans ce chapitre.

Les besoins attachés à un cas d'utilisation servent à représenter trois types d'information à considérer pour la réalisation de ce cas:

- la description des objets du domaine du problème (et leurs propriétés) et les ressources de la plate-forme qui seront utilisés dans le cas d'utilisation ;
- la fonctionnalité du système, c'est à dire, l'ensemble des services pour manipuler le objets du domaine ou à disposition de l'utilisateur dans le cas d'utilisation;
- les besoins de l'interaction utilisateur-système dans ce cas d'utilisation

Les facteurs de qualité (y compris les facteurs d'utilisabilité pour les interactions) sont aussi associés aux cas d'utilisation, mais dans cette thèse la spécification et l'évaluation des facteurs ne sera pas approfondie.

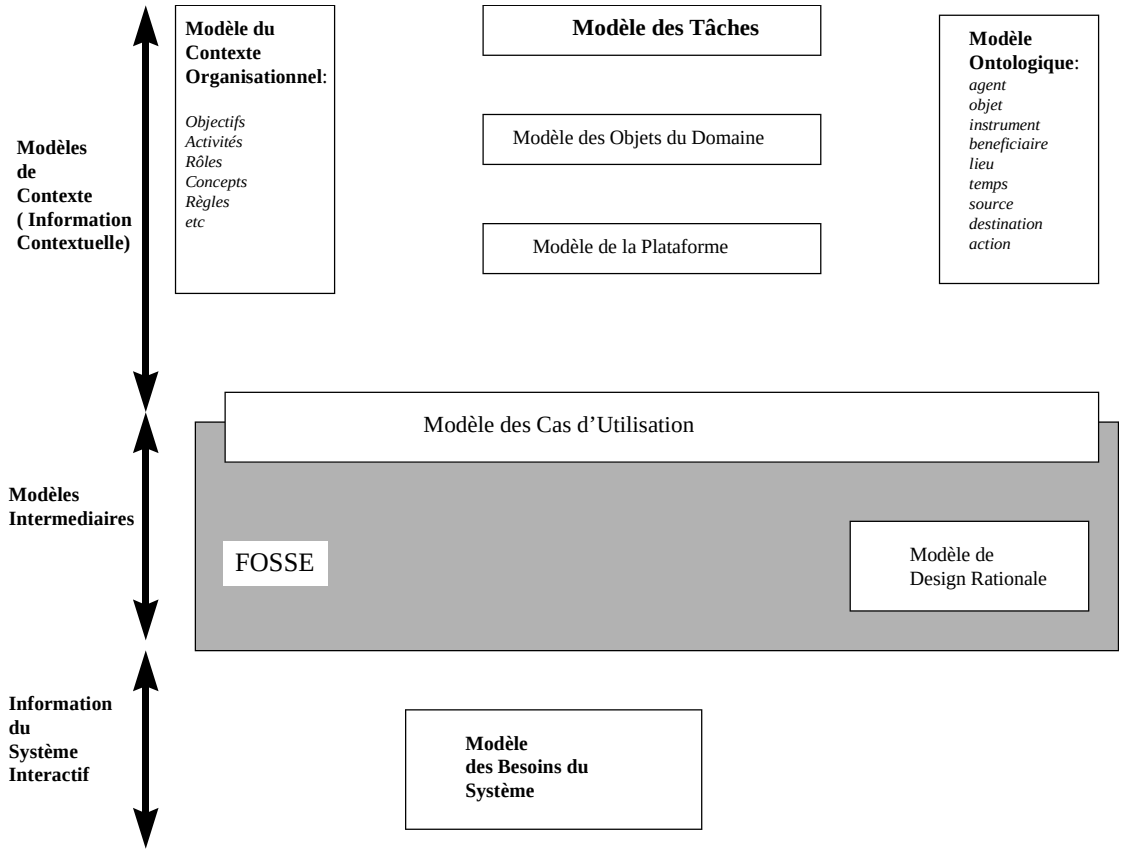


Figure 4.6 - De l'information contextuelle à la spécification des besoins du système (inspirée d'après la figure 1.10 dans le chapitre 1)

TAREFA a deux **modèles intermédiaires**: le modèle de cas d'utilisation et le modèle de *design rationale*, cf. Figure 4.6. Ces modèles sont dites intermédiaires parce qu'ils représentent les résultats intermédiaires (ou partielles) du processus de l'IB. Le modèle de cas d'utilisation est le fils conducteur du processus de détermination des besoins et le modèle de *design rationale* enregistre l'histoire des options et décisions du processus de l'IB.

Les chapitres V et VI présentent en détail le contenu des modèles de TAREFA et comment les construire.

15.Le Processus de TAREFA

Travaux récents en IB reconnaissent l'importance fondamentale de déplacer le point d'attention principal de la modélisation du système à la modélisation du processus de l'IB [Jarke 94, Rolland 94]. La modélisation du système comprend les techniques de modélisation pour représenter notre compréhension des besoins de l'utilisateur et spécifier les composants et propriétés du système en termes abstraits de haut niveau, sans considérer les détails non pertinents. Un modèle du processus sert comme soutien aux activités de construction et transformation des modèles du système, enregistrant l'histoire du développement du système, son contexte, les décisions prises et les arguments qui ont conduits à ces décisions. En fait, cette modélisation des connaissances du processus est très importante en IB pour a) permettre le raisonnement

sur les besoins dynamiques qui changent dans un environnement qui change aussi; b) réutiliser les décisions (non seulement les spécifications) du processus de IB dans d'autres projets; et c) améliorer la manière de travailler de l'équipe multidisciplinaire impliquée au processus.

La manière classique de décrire un processus est l'identification des produits qui passent d'une unité de travail à l'autre. Les modèles pour ce type de description sont appelés modèles de flux de travail (*work flow models*). Dans la littérature sur les processus de logiciel, plusieurs définitions de processus peuvent être trouvées, *comme 'a set of partially ordered steps intended to reach a goal'*[Feiler 93] ou '*a description of the general sequence of activities (...) and of the products involved*' [Conradi 92].

Notre point de vue est qu'un processus est une *manière de travailler*, qui peut être utilisée pour supporter les activités de développement d'un produit. En particulier, le résultat final du processus de l'IB est la spécification du modèle des besoins. La manière d'effectuer ce processus est considérée par TAREFA comme composée par l'entrelacement d'activités systématiques et d'activités émergentes.

15.1Activités Systematiques et Activités Emergentes

TAREFA assume que ces deux types d'activités - systématiques et émergentes - sont complémentaires et doivent être prises en compte. Alors, une démarche pour les activités systématiques et un modèle de processus émergent sont proposés. Une description détaillé de la démarche est présentée dans les chapitres V et VI. Une description du processus émergent est présentée dans le chapitre VII. Dans les paragraphes suivants nos présentons une vue d'ensemble pour résumer leurs points principaux.

Les activités systématiques correspondent aux activités de l'IB qui suivent la structure de contrôle d'une démarche. Ces activités peuvent donc être planifiées a priori et leur contrôle d'exécution est typiquement attaché à structures de ordonnancement et sequencement ou à types spécifiques de choix et conditions de façon presque algorithmique. Par conséquence, elles sont activités facilement supportées par les modèles de processus traditionnels sous la forme d'une démarche. L'ensemble d'activités systématiques d'une démarche fait partie souvent des méthodologies proposées par la plupart des travaux publiés en IB.

Dans l'IB, les activités émergentes correspondent aux activités non-systematiques de la coopération entre les acteurs. Leur coopération désigne une situation dans laquelle des acteurs mettent en oeuvre des actions communes ou interdépendantes en vue d'atteindre les buts communs et où il y a le développement conjoint des idées. Bien sur pour cette coopération, la discussion et la confrontation des plusieurs points de vue sont nécessaires pour intégrer leur compétences multidisciplinaires et pour la négociation, la prise de décision et la résolution de problèmes de manière collective.

Une vue d'ensemble de la démarche pour les activités systematiques et du modèle de processus pour les activités émergentes est présentée ensuite.

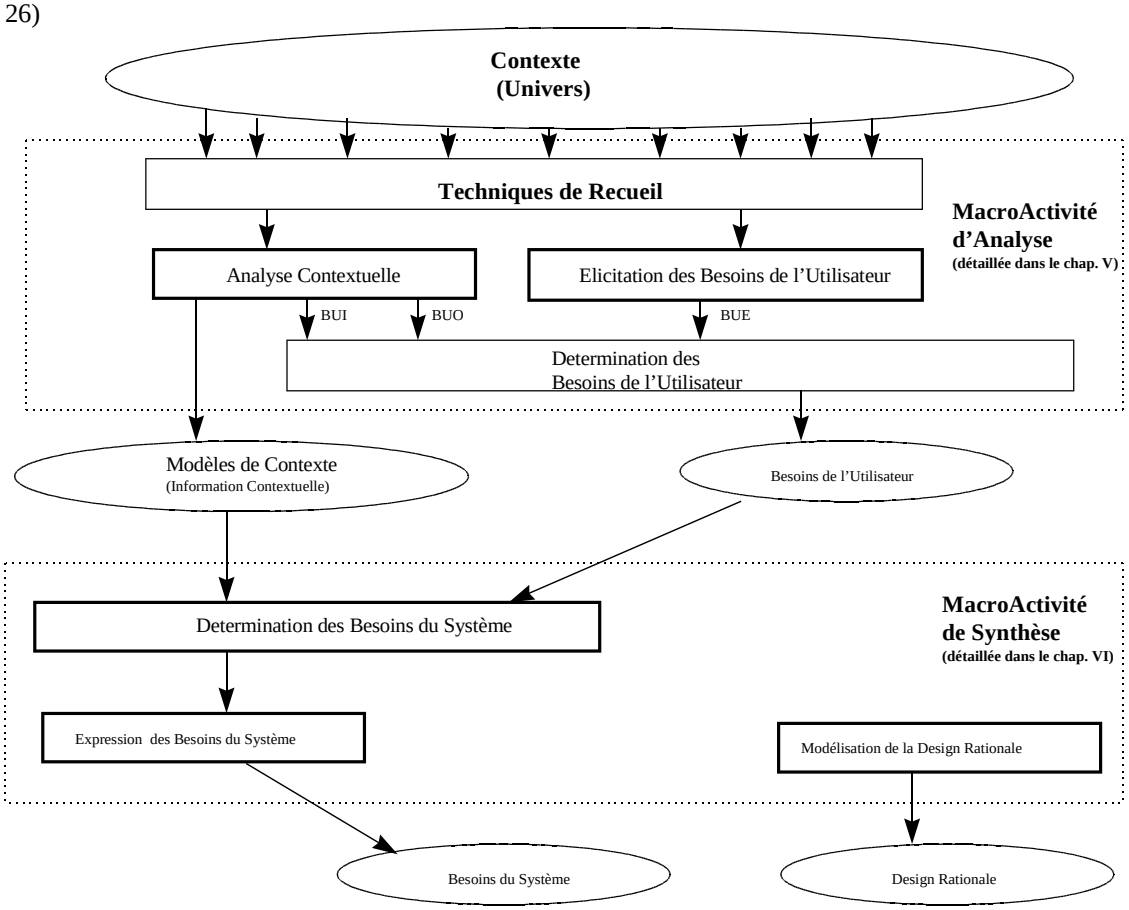
15.2Démarche pour les activités systematiques

Dans TAREFA, les activités systématiques sont activités **exécutées par l'analyste** comme les pas d'une méthode en suivant la démarche proposée. Cette démarche est itérative, en permettant cependant plusieurs points de départ. Dans cette thèse, les activités de la démarche de TAREFA sont présentées de manière linéaire par soucis de

clarté. En pratique, il y a des activités qui peuvent être développées en parallèle ou entrelacées. Une activité est considérée finie lorsqu'il y a suffisamment d'informations recueillies ou des résultats intermédiaires nécessaires à une autre activité. Pendant la réalisation d'une activité il peut arriver souvent qu'une information manque ou que nouvelles questions sont apparus. Cela implique qu'il faut reprendre quelque activité antérieure.

25)

26)



27)

Figure 4.7 - La démarche de l'approche TAREFA

En fait, le processus d'Ingenierie des Besoins avec TAREFA consiste de 2 macroactivités, cf. figure 4.7 :

1. la macroactivité de Analyse et

2. la macroactivité de Synthèse.

L'**Analyse** a pour but comprendre l'univers du domaine du problème et identifier les besoins des utilisateurs. Le terme 'univers' veut dire que l'IB des systèmes interactifs demande en plus de la compréhension du domaine du problème - représenté par un ensemble de entités/objets et/ou activités/processus/fonctions par les approches traditionnelles - une compréhension du contexte du travail où le système va être utilisé, y compris les tâches associées aux activités et les rôles des utilisateurs actuels et futures. Nous pensons que cette compréhension, si bien faite, peut par conséquence faciliter l'identification des besoins et diriger le développement de systèmes utiles et utilisables.

La démarche de la macroactivité d'Analyse est résumé ensuite. Cf. Figure 4.7, elle a trois activités principales:

1. L'Analyse Contextuelle;

2. L'Elicitation des Besoins (explicites) de l'Utilisateur; et

3. La Détermination des Besoins de l'Utilisateur.

Les deux premières utilisent les diverses techniques de recueil pour respectivement acquérir les connaissances sur l'Univers et énumérer les besoins demandés explicitement par l'utilisateur (BUEs). La troisième utilise les besoins explicites et les connaissances de l'univers, y compris les dysfonctionnements observés, les procédures de travail formel, les règles (sociales, politiques, commerciales, etc) internes de l'entreprise), pour arriver à un ensemble des besoins de l'utilisateur, résultat d'une évaluation et classification des besoins demandés explicitement (BUEs), les besoins implicites identifiés par Analyse du travail (BUI) et les besoins de l'Organisation (BUOs).

Le résultat de la macroactivité d'Analyse de TAREFA est donc:

- un ensemble des besoins de l'utilisateur;

- un ensemble de facteurs contextuels et facteurs de qualité qui servent comme ressources et comme contraintes à considérer pour le développement du système;

- un ensemble de modèles pour représenter l'information contextuelle recueillie c'est à dire, l'information sur le contexte organisationnel, l'ontologie du vocabulaire du domaine, les tâches, les objets du domaine et les moyens technologiques disponibles.

La **Synthèse** a pour but déterminer et exprimer les besoins du système pour atteindre les besoins des utilisateurs et représenter le historique des décisions et arguments associés à ces activités. La démarche de la macroactivité de Synthèse est résumé ensuite. Conformément la fig. 4.7, elle a trois activités principales

1. Détermination des Besoins du Système;

2. Intégration des Besoins du Système; et

3. Modélisation du *Design Rationale*.

La première activité est la plus complexe parce qu'elle est cyclique. A partir des besoins de l'utilisateur (l'ensemble de BUEs, BUOs et BUIs) et de l'information contextuelle, les besoins du système sont itérativement découverts. Le fils conducteur pour ce processus sont les cas d'utilisation, définis par TAREFA comme ayant plusieurs niveau d'abstraction: essentiel, singulier, opérationnel et concret. La structure basique de chaque cas d'utilisation (appelée niveau essentiel) est construite à partir du modèle de tâches minimales. Cette structure est raffinée pour créer les cas d'utilisation singuliers par la détermination des problèmes et dysfonctionnements associés aux épisodes et l'identification des possibles actions défensives et correctives de la part du système. Ces cas d'utilisation singuliers peuvent être décrits en termes de objets, méthodes et états (niveau opérationnel) pour permettre génération automatique de prototypes ou directement raffinés en cas d'utilisation concrets. Les prototypes et les cas d'utilisation concrets servent pour la expérimentation/validation avec l'utilisateur. Le deuxième est en fait un type de prototype bas niveau (*low level*).

L'expérimentation avec l'utilisateur peut impliquer changements de cas d'utilisation ou des besoins ou les deux choses. Bien sur, un changement de cas d'utilisation peut

demander des nouvelles informations contextuelles, ce qui implique des activités de la macroactivité d'Analyse. Les cas d'utilisation modifiés sont de nouveau raffinés et soumis à l'utilisateur. Quand ce processus arrive consensuellement à un ensemble des besoins considéré satisfaisant par les participants, ces besoins sont intégrés et exprimés dans la forme d'un modèle des besoins du système. Les principales décisions sont enregistrées par le modèle de *design rationale*. Le processus coopératif de négociation et résolution de conflits est supporté par le modèle de processus émergent, qui sera décrit dans la section suivante.

Le résultat de la macroactivité de Synthèse de TAREFA est:

- un ensemble de cas d'utilisation qui modélisent certaines situations d'utilisation considérées fondamentales;
- un modèle des besoins du système;
- un modèle de design rationale;

La description detaillé des macroactivités d'Analyse et Synthèse de la démarche est présentée respectivement dans les chapitres V et VI.

15.3Modèle de processus émergent pour les activités emergentes

Le processus d'IB est défini dans un espace de trois dimensions (cf. Chapitre II): Représentation, Spécification et Accord (agreement). Récemment, il y a eu un évident progrès en fournir réponses aux problèmes des deux premières dimensions (voir par exemple l'intérêt croissant pour les modèles formels). En contraste, très peu d'attention a été dirigée à la dimension d'accord. Nous assumons que l'émergence est un concept adéquat à fournir un soutien très riche à la dimension d'accord, c'est à dire, aux activités collaborations entre les acteurs. Ces activités sont très difficiles d'automatiser et de contrôler parce qu'elles impliquent la complexité et la nature dynamique du travail de groupe mais il possible de les soutenir. Pour cela, nous avons proposé un modèle de processus pour les activités colaboratives de l'IB basé sur la notion d'émergence.. **Emergence** est un concept dérivé de la Science Cognitive, de la Théorie des Systèmes et de la Ethnomethodologie et qui a aussi un important rôle dans la théorie d'IHM [Suchman 87]

Les processus de collaborations réels de l'IB ne peuvent pas être considérés comme ayant une séquence rigide de procédures et décision suivant les pas d'une démarche centrée sur l'analyste parce leur dynamique est déterminée par choix, décision et participation des acteurs. Une grande variété d'alternatives de actions et interactions peut être combinée pour accomplir les buts du processus. Ces alternatives ont influence du contexte qui forme le espace de ressources pour les stratégies acceptables. En fait, selon [Barghouti 95], la modélisation formelle de processus réels peut montrer inconsistances et inneficiences du processus et il devient difficile de convaincre les participantes à utiliser d'autres manières pour l'exécuter. Cette remarque est particulièrement vraie pour les activités coopératives de l'IB où l'utilisateur n'accepte pas suivre une séquence spécifique pour discuter ses problèmes et décider solutions qui vont l'affecter.

Notre modèle de processus émergent est utile comme support aux processus coopératifs parce qu'il fourni un espace d'objets partagé et un meta-modèle où les plans et buts évoluent dans une façon consensuelle et émergente, en prenant compte à la fois les actions, les communications et la cognition du groupe de travail pendant

l'interaction entre ses membres. L'emphases du modèle est alors plutôt la flexibilité du processus mais tout en prenant compte la traçabilité et la reutilisabilité des discussions et décisions que les aspects de contrôle et guidage, caractéristiques des modèles traditionnels.

Une description detaillé du processus émergent est présentée dans le chapitre VII.

16.Classification de TAREFA par rapport aux autres approches

La figure 4.8 présente la classification des approches réalisée au chapitre III dans laquelle l'approche TAREFA a été rajoutée.

Critères	**DIANE+**	**MUSE**	**ADEPT**	**TRIDENT**	**ICO**	**TAREFA**
Information Contextuelle	Les modèles de Merise: MCD, MCT, MOD, MOT;	Modèle de Tâche Généralisé	Modèle de Tâche Existante (TKS), Modèle de l'Utilisateur	Modèle des objets Métiers (ER), Modèle de Tâche Interactive (TKS), profile des utilisateurs; description de l'environnement physique	Modèle à Objets	Modèle de Contexte Organisationnel, Modèle des Objets Métier du domaine, modèle ontologique du vocabulaire du domaine, modèle de tâches (minimales), modèle de plateforme
Information Opérationnelle	Modèle de Dialogue et le modèle des objets OPAC	Modèle de Tâche Composite; Modèle de Tâche Spécifique	Modèle de Tâche Future	Graphe d'enchaînement des activités	**?**	modèle de cas d'utilisation; modèle de *design rationale*.
Information du Système	Dialogue, sémantique	Dialogue, sémantique	Dialogue, sémantique	Dialogue, présentation, sémantique	Dialogue, sémantique	Dialogue, sémantique
Méthode/Notation de Base	Merise	JSD	non	ER	Réseau de Petri	Objectory
Soutien à la Communication utilisateur-analyste	non	non	non	non	non	modèle ontologique du vocabulaire du domaine du problème
Soutien à l traçabilité	non	non	non	non	non	oui
Participation de l'Utilisateur	passive	passive	active (nouvelle version)	passive	passive	active; à travers un modèle de processus émergent pour soutenir la coopération entre l'analyste et l'utilisateur
Domaine de Application (fiabilité des cas appliqués)	Systèmes d'Information	Systèmes d'Information	**?**	Systèmes d'Information	Systèmes Critiques	Systèmes d'Information 'maison'
Design Rationale	non	non	non	non	non	oui
Supporté par Outil ?	oui	non	oui	oui	oui	oui, à l'heure actuelle partiellement
Utilisateur	informaticien	ergonome	**?**	informaticien	informaticien	informaticien

Figure 4.8 - Classification de TAREFA en rapport aux autres approches d'integration GL-IHM

L'approche TAREFA, comme toutes les approches présentées, se situe dans les approches basées sur les tâches. TAREFA cependant utilise un cadre contextuel plus important: un modèle de contexte organisationnel, un modèle d'objets métier du domaine, un modèle ontologique du vocabulaire du domaine et un modèle de plateforme. Ces modèles complémentaires rend l'approche TAREFA adéquat pour une bonne compréhension de l'univers où les tâches sont réalisées et où le système va être développé et utilisé. La prise en compte de ce contexte permet de considérer aussi des aspects sociaux et organisationnels des tâches dans un processus de travail et pas seulement les aspects individuels de leur réalisation. Dans ce sens, TAREFA est proche

de la méthode DIANE+ qui utilise les concepts organisationnels de Merise. La différence essentielle entre les deux approches est que TAREFA utilise explicitement cette information contextuelle pour la détermination des besoins de comportement alors que DIANE+ ne l'utilise que pour la spécification conceptuelle du composant sémantique.

L'information operationnelle de TAREFA est aussi une différence importante. Au contraire des autres approches, qui utilisent les modèles de tâche comme le fils conducteur pour la détermination et spécification du système futur, nous sommes d'accord avec [Jacobson 92], [Carrol 95] et [Rosson 95], qui mettent le concept de **cas d'utilisation comme le fils conducteur** de ce processus. En fait, spécifier les besoins d'un système à partir d'un modèle de tâches existantes n'est pas évident et TAREFA propose une démarche pour a) construire les cas d'utilisation à partir des tâches existantes et b) utiliser les cas d'utilisation pour représenter et réfléchir sur différentes situations dans le but de formuler les besoins du système futur, en intégrant la préoccupation avec l'utilisabilité et les facteurs humaines dépendants du contexte de travail. Le modèle de *design rationale* de TAREFA sert non seulement à enregistrer l'historique de la démarche et les justificatives des choix effectués mais aussi pour guider la prise de décisions.

A exception de TRIDENT, dont l'état courant est centré sur la spécification et génération du composant de présentation, TAREFA et les autres approches ne prennent en compte ce composant de présentation.

En rapport aux problèmes typiques de l'IB, qui ne sont pas resolus par les autres approches, TAREFA propose des solutions spécifiques, comme nous l'avons montré dans la section 2.4 dans ce chapitre.

Chapitre V - TAREFA : La macroactivité d'Analyse

Vis-à-vis de notre structuration de TAREFA présentée au chapitre précédent, ce chapitre présente spécifiquement la démarche et les modèles associés à la première macroactivité - l'Analyse.

Ici, le mot 'analyse' fait référence à l'examen des caractéristiques, conditions et problèmes du domaine. L'absence de ces connaissances est reconnue comme un problème omniprésent et crucial dans la conception des systèmes [Curtis et al. 88].

TAREFA propose 2 activités principales dans la macroactivité d'Analyse :

1. la compréhension de l'univers du système ;

2. la détermination des besoins de l'utilisateur.

L'information contextuelle recueillie par la première activité et les besoins de l'utilisateur recueillis par la deuxième sont fondamentaux pour le but principal qui est de déterminer les besoins du système.

Dans TAREFA, nous donnons le nom d'Analyse Contextuelle à la compréhension de l'univers dans lequel le système va être développé et utilisé, parce que l'univers correspond en fait au contexte du travail que l'application interactive va devoir soutenir. L'ensemble de l'information recueillie et représentée sur l'univers est appelée Information Contextuelle. L'approche d'Analyse Contextuelle de TAREFA est basée sur un ensemble des modèles interconnectés pour représenter l'information contextuelle et un ensemble d'activités pour acquérir, modéliser et valider cette information contextuelle.

La Détermination des Besoins de l'Utilisateur est généralement une étape très difficile pour les informaticiens parce qu'il n'y a à l'heure actuelle aucune méthodologie pour l'identification (*elicitation)* et l'explicitation des besoins réels des futurs utilisateurs du système à construire. TAREFA adopte une approche coopérative entre analyste et utilisateur, soutenue principalement par le recueil de la plus grande quantité possible d'information contextuelle pertinente, par une bonne utilisation de l'information contextuelle recueillie et par une amélioration de la communication directe entre eux.

Dans ce chapitre, nous introduisons d'abord une étude de cas simple qui sera utilisé pour la présentation de TAREFA dans tous les chapitres de cette partie : le distributeur automatique de billet (dont l'acronyme utilisé dans cette thèse sera ATM, de l'anglais *Automatic Teller Machine*). Ensuite nous présentons l'approche d'Analyse Contextuelle de TAREFA (section 2) et l'approche de TAREFA pour la détermination des besoins de l'utilisateur (section 3).

17.L'étude de Cas Utilisé pour la présentation de TAREFA : l'ATM

Pour illustrer l'utilisation de la démarche dans les chapitres suivants, nous utilisons des extraits de l'étude de cas ATM (de l'anglais *Automatique Teller Machine)*. L'ATM est le terme utilisé pour les distributeurs automatiques de billet, avec la fonction classique de retrait d'argent mais parfois aussi des fonctions secondaires comme l'obtention du solde ou d'un relevé d'opérations, et les fonctions de virement et de dépôt.

Nous avons plusieurs raisons pour choisir ATM :

- La communauté de recherche l'adopté comme un 'benchmark' ([Somerville 95],

[Wirfs-Brock 90], [Rumbaugh 91], [Blandford et al. 93], [Palanque & Bastide 95], [Kaindl 95]);

- Il ne nécessite pas une connaissance spécialisée du domaine pour le comprendre; en fait, nous supposons que les lecteurs ont tous quelque expérience dans l'utilisation d'ATMs;

- L'ingénierie des besoins de l'ATM illustre les problèmes typiques des besoins des systèmes maison réels : il y a quelques caractéristiques très dépendantes de l'organisation - parfois cachées par des suppositions implicites et considérées évidentes, il y a une ample opportunité pour différentes interprétations et différentes points de vues sur les besoins et plusieurs aspects ne sont souvent pas considérés convenablement par les ATM existants;

- Il est représentatif d'une classe de systèmes informatiques appelés 'walk-up and use systems', qui sont des systèmes qui peuvent être utilisés par un large spectre de personnes sans formation spécifique - le grand public. Donc il n'est pas nécessaire de faire une étude approfondie des utilisateurs ni une caractérisation de leurs préférences et expériences, ce qui simplifie la compréhension de l'exemple.

Dans cette thèse, l'ATM est un étude de cas délibérément simple, parce que le but ici est de présenter la démarche et non de concevoir totalement une machine ATM. Nous avons fait quelques suppositions pour le simplifier :

- cette étude de cas se place dans le cadre d'une banque précise. Ainsi, les sections présentant les modèles sur l'étude de cas seront rédigées comme si on appliquait TAREFA pour la conception des ATMs de cette banque spécifique ;

- l'ATM est une machine 'stand-alone', sans connexions réseau;

- l'ATM n'offre pas diverses options de langue pour les utilisateurs;

- l'ATM n'offre pas diverses options de devises.

Cependant, même simplifié, nous pensons que l'exemple est suffisamment complexe pour illustrer à la fois le processus et les modèles de TAREFA.

18.Analyse Contextuelle de TAREFA

L'Analyse Contextuelle est principalement une approche d'analyse (ou plus largement d'étude) de l'existant. En fait, toutes les méthodes d'Analyse de Systèmes et d'Ingénierie des Besoins ont, sous diverses formes, des types d'analyse de l'existant qui essaient de modéliser les entités du monde réel à travers un modèle du domaine de l'application, traditionnellement orienté flux de données, ou orienté objets, ou orienté fonctions.

Cependant, la conception de systèmes interactifs demande une compréhension non seulement du domaine de l'application du système mais aussi du contexte de travail dans lequel le système va être utilisé. Ainsi, une représentation plus complète de ce contexte est nécessaire. Nous appelons Analyse Contextuelle l'ensemble d'activités qui abouti à la construction de différents modèles dont l'ensemble constitue l'Information Contextuelle.

Nous adoptons ici le terme 'Analyse Contextuelle' dans le même sens qu'utilisé par [REAW 91] :

"Analysis of the problem space and application domain of a potential system to be developed. It deals with description of problems only, not solutions. "

Cette analyse a pour but l'investigation et la représentation des connaissances du domaine du problème, c'est à dire,

- la sélection des 'acteurs' participants au processus d'Analyse ;
- la définition du problème à résoudre avec le développement du système (le *problem design*) ;
- la compréhension de la situation concernée et qui est à l'origine des motivations pour la proposition d'un système futur.

18.1Sélection des acteurs participants

Avant tout, il faut déterminer l'équipe du processus. Idéalement, le groupe ne doit pas être trop étendu dans le but de créer un groupe gérable c'est à dire qui ne fournisse pas trop d'information mais qu'il soit représentatif.

La sélection doit suivre des critères :

I. les participants sont divisés de façon à comprendre des représentants du client (le manager par exemple) et des représentants des équipes des utilisateurs finaux, de marketing, de formation, de maintenance et bien sur de conception (l'analyste). L'objectif est d'avoir dans l'équipe la participation de qui a commandé le système, de qui va l'utiliser (fréquemment ou occasionnellement), qui va être affecté par le système et qui va enfin le développer. Le but est d'avoir dans le groupe des points de vue différents et une 'voix de l'expertise' pour chacun des domaines impliqués;

II. chaque participant doit être reconnu comme une 'voix de l'expertise' dans son domaine selon l'expression '*If you cannot afford to lose the person for three days, then that is the person we want*' [Carmel 93] ;

III. la présence des représentants des utilisateurs finaux doit être incitée, surtout pour éviter qu'en respectant le critère II le groupe ne soit composé uniquement que des leaders de chaque groupe;

IV. le rôle de l'analyste n'est pas de devenir le 'leader' du groupe mais plutôt de représenter la voix de l'expertise informatique et de faciliter la recherche de problèmes et de solutions consensuelles.

Normalement la sélection implique le manager du projet ou l'initiateur du projet qui, avec l'analyste, va identifier les personnes participantes selon les critères ci-dessus. Cependant, il n'est pas garanti que les personnes sélectionnées vont participer de façon effective.

18.2Définition du Problème

La définition du problème est faite par une description initiale des objectifs du système. Ces objectifs sont les motivations pour la proposition du système futur : soit une amélioration des services manuels existants, soit incrémenter un système informatique existant, soit une

nouvelle fonctionnalité, soit l'adoption de nouvelles technologies disponibles, etc. Comme proposé par [Newman 95], cette définition peut n'être composée que par une phrase, présentant les quatre composants essentiels qui définissent : l'activité à soutenir, l'utilisateur cible, le niveau de support et la forme de solution demandée. Bien sûr, ces composants ne peuvent pas être dès le début très détaillés mais leur identification est une prise de conscience très importante. Le schéma de définition est :

Définir <*forme de solution*> pour <*niveau de soutien* >, <*activités prises en compte*> à être réalisées par <*réalisateurs des activités, les utilisateurs cibles*>.

Dans le cas de l'ATM :

Définir *le logiciel interactif d'une machine ATM* pour *permettre l'utilisation libre service 24h/24h* pour *les opérations bancaires les plus fréquentes comme le retrait, le virement, la consultation de solde et le relevé d'opérations* pouvant être réalisées par *les clients de la banque X*.

18.3Compréhension de la Situation

La compréhension de la situation est faite par l'identification, la détermination et la représentation de son Univers d'Information, cf. Définition du chapitre II, c'est à dire, d'un sous ensemble de caractéristiques actuelles de l'univers, comme par exemple les activités de l'utilisateur, le contexte organisationnel et les caractéristiques des rôles des utilisateurs. Comme nous l'avons vu dans le chapitre IV, TAREFA adopte le terme '**Information Contextuelle**' pour l'ensemble des connaissances de l'univers du problème.

18.4L'Information Contextuelle

Pour TAREFA, l'Information Contextuelle est composée par :

- le modèle ontologique du vocabulaire de l'univers ;
- les modèles des tâches des utilisateurs dans l'univers, en particulier leurs buts ; cette information est le résultat d'une analyse microscopique du travail ;
- le modèle du contexte organisationnel de ces tâches, pour montrer les relations entre les tâches, les acteurs et les processus de travail de l'univers; cette information est le résultat d'une analyse macroscopique du travail ;
- le modèle des objets métiers de l'univers ;
- le modèle de plateforme, qui décrit les moyens techniques matériels (processeurs, périphériques, etc) et logiciel (systèmes d'exploitation, SGBDs, systèmes de fenêtrage, etc) disponibles pour les activités dans l'univers.

Cet ensemble de modèles sont interconnectés par l'intermédiaire du modèle de Contexte Organisationnel qui joue le rôle d'intégrateur. Le but de cet ensemble de modèles est de comprendre et de documenter ces connaissances d'une manière plus organisée et structurée que les descriptions totalement informelles en langage naturel ou que les descriptions monolithiques. Recueillir les informations contextuelles en TAREFA implique d'utiliser les

techniques de recueil (TBC, TBE et TBO) pour obtenir les informations et d'instancier des modèles à partir de ces informations. Ces modèles représentent parfois des informations délibérément redondantes pour permettre d'établir entre eux des connexions explicites. La redondance est cependant contrôlée, car les relations et connexions explicites permettent de suivre la chaîne des liaisons.

Comme ces modèles sont structurés mais non formels, ils peuvent être facilement compris par les utilisateurs, qui peuvent ainsi collaborer avec le concepteur. Cette collaboration est particulièrement améliorée par la définition et l'utilisation du modèle ontologique du vocabulaire. Par contre, les modèles non formels rendent difficile la vérification automatique (p.ex. consistance) de chaque modèle et les connexions entre les modèles. Cette vérification doit usuellement être faite à travers le dialogue utilisateur-concepteur et par des activités de '*walk through*' des modèles faits par le concepteur. 'Walk-through' est une technique utilisée pour l'évaluation de l'interface utilisateur [Lewis et al. 92] mais nous pensons, comme [Newman 95], que cette technique peut être utilisé pour évaluer la modélisation du système interactif, en particulier le modèle de ses besoins. L'idée basique de cette technique est de parcourir le modèle en examinant les problèmes que l'utilisateur peut rencontrer dans l'utilisation du système pour réaliser ses tâches.

Les sections suivantes montrent les modèles de TAREFA pour les composants de l'univers, leurs descriptions et la démarche à suivre pour leur construction. Enfin, chaque modèle sera exemplifié sur l'étude de cas ATM.

18.5Le Modèle Ontologique

Le modèle ontologique est une représentation de l'ontologie du vocabulaire de l'univers d'informations. Le but est de comprendre le vocabulaire du domaine utilisé par les utilisateurs et ainsi d'établir une terminologie unifiée pour permettre une meilleure communication avec l'analyste. Puisque le sujet de la Sémiotique porte sur les systèmes de signe et les phénomènes de communication et signification, nous avons choisi d'emprunter des concepts et techniques sémiotiques comme base d'une approche pour cette amélioration, que nous appelons Approche Sémiotique. Le modèle ontologique a été proposé tout d'abord dans [Faust & Pimenta 95], mais sa description est plus complète dans [Pimenta & Faust 97].

18.5.1Description du Modèle Ontologique

Puisque plusieurs problèmes de l'IB - en particulier le manque de connaissance sur le domaine de l'application - sont dus à des pannes de communication (*communication breakdowns*) entre utilisateurs et analystes (voir [Curtis et al. 88] pour justificative empirique), il est étonnant qu'il y ait si peu d'effort fait pour améliorer la communication utilisateur-analyste.

Notre approche est basée sur la définition d'un langage commun entre eux plutôt que de développer des traductions entre leur deux langages différents. Nous prenons en compte le problème de la communication à travers la description d'une ontologie qui contient :

a) les concepts qui reflètent les aspects principaux du langage utilisateur et

b) la signification de leur occurrences en rapport aux autres concepts.

Par définition, une **ontologie** est une spécification du discours entre multiples agents sous la forme d'un vocabulaire partagé [Gruber 93], [O'Leary 96], [O'Leary et alli 97]. Cette ontologie peut alors être adoptée pour tout développement dans le domaine.

L'ontologie est centrée sur la détermination d'un langage utilisateur, à savoir **le langage**

utilisateur sur son travail (**LUT**). Ce langage est utilisé pour parler **sur** la situation de travail et il ne doit pas être confondu avec le **langage de travail de l'utilisateur**, un langage opératif utilisé **pendant** la situation de travail [Falzon 84] [Katzenberg 93].

L'unité basique du LUT est la **description du signe**, utilisé pour signifier les concepts réels de l'utilisateur. Signes peuvent être

i) signes qui appartiennent au LUT - qui sont des concepts fondamentaux du LUT qui doivent donc être décrits - et

ii) signes communs - que la plupart des personnes utilisent sans problèmes d'interprétation en situations quotidiennes, comme des prépositions, connecteurs, etc - et qu'il n'est donc pas nécessaire de décrire comme composants du LUT.

La description de chaque signe est composé par :

- a) un **nom** (et l'énumération des synonymes utilisés par l'utilisateur) ;
- b) une **notion** du signe, c'est à dire, un ensemble de phrases qui contiennent un ensemble de signes qui expriment sa définition ; et
- c) un ensemble d'**impacts**, c'est à dire, un ensemble de phrases sur les effets de l'usage ou de l'occurrence du signe décrit.

Notions et **impacts** sont interprétants (selon la terminologie de Peirce [Peirce 31]) de chaque signe du LUT. Autrement dit, ce sont des signes qui nous aident à trouver un signifié approximatif du signe décrit.

Les notions et impacts doivent respecter deux principes, définis par [Leite 93] :

1. *Principe de la Circularité*, qui établit que chaque signe du LUT doit être décrit autant que possible par l'intermédiaire d'autres signes du LUT ;
2. *Principe du Vocabulaire Minimal*, qui établit que l'utilisation de signes qui n'appartiennent pas au LUT doit être évitée ou réduite autant possible.

18.5.2Construction du Modèle Ontologique

L'identification du LUT est faite grâce à un **processus de détermination de signes**. Ce processus est un processus itératif d'acquisition/représentation/validation de signes qu'il faut faire incrementalement et collaborativement entre analystes et utilisateurs**.** L'**acquisition de signes** est faite usuellement par intermédiaire d'interviews informelles ou de verbalisation a posteriori (*retrospective protocole*) mais aussi par immersion des analystes dans le lieu de travail. Pour la verbalisation a posteriori, le travailleur est invité à générer un rapport verbal sur la réalisation de ses activités de travail. L'immersion est la participation active des analystes dans les activités de travail. La **validation des signes** est faite usuellement par l'intermédiaire d'interviews informelles. La **représentation des signes** est d'abord informelle puis est structurée avec la notation Language Extended Lexicon - LEL [Leite 93], en format hypertexte. Un outil (appelée *HyperLex Cover*, décrit dans [Faust 95])a été développé pour aider à la création/manipulation de signes dans ce format. Utiliser la notation LEL est simple :

- une description d'un signe du LUT est une *entrée* ;
- les signes utilisés pour la description des notions et des impacts et qui appartiennent au LUT sont des liaisons (*links*) à leurs entrées respectives ;

- la compréhension du LUT est en fait la 'navigation' dans la description hypertextuelle des signes.

Cas Sémantiques	Définition	Notion	Impact
Agent	Le responsable d'une action	Description de l'agent	Les actions de l'agent
Objet	Ce qui reçoit l'effet d'une action	Description de l'objet	Les actions applicables à l'Objet
Instrument	Un objet ou un processus qui contribue à l'action	Description de l'instrument	Comment l'Instrument est utilisé pour la réalisation de l'action
Bénéficiaire	La personne (ou l'objet) sur qui l'action a un effet	Description du bénéficiaire	Les actions qui l'affectent ? Comment l'affectent-il?
Lieu	L'endroit où se situe une action	Description du lieu	Les actions qui se situent dans ce lieu
Temps	Le moment de l'action	Description du moment	Les actions qui arrivent à ce moment ou l'ordre chronologique relative aux autres actions
Source	L'état au début de l'action	Description de l'état	Les actions qui changent l'état à partir de la Source
Destination	L'état à la fin de l'action	Description de l'état	Les actions que changent l'état pour la Destination
Action	L'action elle-même	L'acteur qui la réalise, ses pré-conditions, la procédure pour sa réalisation (sous-actions, etc.)	Conséquences : Post-conditions ou autres actions (propagation)

Figure 5.1 - Cas sémantiques utilisés pour le processus de détermination de signes et leurs notions et impacts respectifs

Le fondement théorique du processus de détermination de signes est la Grammaire de Cas [Fillmore 1968], une technique d'analyse du discours que nous utilisons d'une façon particulière. Dans la théorie de la Grammaire de Cas, les composants d'une phrase sont identifiés comme un ensemble de cas sémantiques, définis par leur relation avec le verbe. Le nombre exact et la définition précise des cas peut varier[10] et notre choix inclut les cas suivants, (voir figure 5.1) : Agent, Bénéficiaire, Instrument, Objet, Temps, Lieu, Source/Destination, Action. Ces cas sont une manière directe d'identifier les signes pertinents du LUT. Le sens des concepts de notion et d'impacts change selon le cas correspondant, (voir aussi figure 5.1). Dans le cas où un signe appartient à plusieurs cas sémantiques, la notion de ce signe est l'union de toutes les notions pertinentes et l'ensemble des impacts est l'union de tous les impacts respectifs.

18.5.3l'exemple ATM

La figure 5.2 ci-dessous est un extrait de la description des signes du LUT pour l'exemple ATM. Cet extrait contient la description (délibérément incomplète) de 3 signes : CLIENT/ CLIENTE, SOLDE/DEMANDE DE SOLDE/DEMANDE SOLDE/DEMANDER SOLDE/.. et RETRAIT D'ARGENT/FAIT RETRAIT D'ARGENT/RETIRER L'ARGENT.

Signs description (extrait)
sign : CLIENT/ CLIENTE
notions
Personne qui a une COMPTE dans une AGENCE d'une BANQUE. *(Catégorie Agent)*
Personne qui a une CARTE et connaît son MOT-DE-PASSE. *(Catégorie Agent.)*

impacts
As Agent : *(actions d'un agent)*
Client fait un DEPOT.
Client fait un VIREMENT.
Client fait un RETRAIT-D'ARGENT.
Client DEMANDE CHEQUIER.
Client DEMANDE SOLDE.
Client DEMANDE RELEVE D'OPERATIONS.
...

10 Fillmore lui-même a proposé diverses versions qui ont suivi celle de 68 (6 cas en 68, 8 cas en 71, 9 en 71 également). Cependant, il supposait que ces cas étaient toujours en nombre restreint, de l'ordre de la dizaine.

sign SOLDE/DEMANDE DE SOLDE/DEMANDE SOLDE/DEMANDER SOLDE/..
notion *(Catégorie Action)*
Action demandé par le CLIENT.*(Qui la réalise)*
CLIENT S'IDENTIFIE. *(sous tâche de la réalisation de l'action)*
SOLDE ACTUEL est calculé. *(sous tâche de la réalisation de l'action)*
...
impacts
CLIENT PREND LE TICKET de solde. *(Post-condition)*

sign RETRAIT D'ARGENT/FAIT RETRAIT D'ARGENT/RETIRER L'ARGENT/....
notion *(Catégorie Action)*
Action demandé par le CLIENT.*(Qui la réalise)*
CLIENT a une CARTE VALABLE. *(Pré condition)*
CLIENT entre le MOT-DE-PASSE correct. *(sous tâche de la réalisation de l'action)*
...
impacts
SOLDE du COMPTE du CLIENT a changé. *(Post-condition)*
CLIENT demande REÇU DE RETRAIT. *(Action suivante optionnel)*
SOLDE ACTUEL est calculé. *(Action suivante obligatoire)*

Figure 5.2 - Extrait d'un description du LUT de l'ATM

Ces noms (et synonymes) sont présentés séparés par un caractère '/' - '*slash*'. Partout dans la description, la casse majuscule indique que le signe est aussi une entrée de la description. La casse minuscule est utilisé pour les signes communs, qui ne sont pas des entrées. Les commentaires à titre d'explication sont présentés en italique dans la figure 5.2. Dans l'extrait présenté à la figure 5.2, le modèle ontologique contient au moins la description des signes suivants.

client, compte, agence, banque, carte, mot-de-passe, dépôt, virement, retrait d'argent, demande chéquier (demande de chéquier), demande solde (demande de solde), demande relevé d'opérations (demande de relevé d'opérations), s'identifie (s'identifier ou identification du client), solde actuel, prend le ticket (prise du ticket), carte valable, solde, reçu de retrait.

La description de ces signes, même en respectant les principes de la circularité et du vocabulaire minimal, peut bien sûr nécessiter la description de nouveaux signes du LUT.

Le signe 'CLIENT' est classé comme un agent, un bénéficiaire et un objet. Par conséquence, la notion de 'CLIENT' est l'union de la notion de chaque description de 'CLIENT' pour les différentes catégories. Pour les impacts, la procédure est identique. Par soucis de concision, la figure 5.2 ne montre que partiellement les impacts pour 'CLIENT' selon la catégorie Agent, c'est à dire les actions de 'CLIENT'. Par exemple, **'**Client fait un DEPOT' et 'Client fait RETRAIT-D'ARGENT'.

Il faut remarquer que chaque action est aussi une entrée. Si nous nous intéressons, par exemple, aux détails de l'action 'RETRAIT D'ARGENT', il suffit d'en parcourir la description hypertexte.

Le signe 'RETRAIT D'ARGENT' possède aussi des formes verbales (FAIT RETRAIT D'ARGENT et RETIRER L'ARGENT) comme synonymes, un fait normal pour sa catégorie. Les notions de ce signe, suivant le figure 5.1, définissent :

- l'acteur qui la réalise ;
- ses pré-conditions ;
- la procédure pour sa réalisation (sous-actions, etc.).

La figure 5.2 ne montre qu'une instance de chaque type. Les impacts, encore suivant le

figure 5.1, définissent les conséquences de l'action. La figure 5.2 ne montre qu'une post condition et deux actions rendues possibles par l'action : une action optionnelle et une obligatoire.

En fait, les descriptions de signes sont très longues et l'outil hypertexte *HyperLex-cover* implementée est fondamental pour aider à construire et à organiser les structures LEL.

18.6Les Modèles des Tâches Minimales

Comme nous l'avons vu au chapitre I, l'Analyse de la Tâche a émergée de l'Ergonomie comme une aide fondamentale pour les approches de conception des systèmes interactifs. Cependant nous avons aussi souligné qu'il y a autant de définitions de tâche que de modèles de tâche.

En fait, un modèle de tâche est un modèle riche et complexe parce qu'il représente à la fois 2 types d'information [Sperandio 87] :

- l'information sur les buts à atteindre, de type plutôt déclaratif ;
- l'information sur les actions pour arriver aux buts, de type plutôt procédural.

Nous allons ensuite expliquer la différence entre le concept de tâche minimale et le concept de tâche tel que l'on le rencontre dans la littérature des systèmes interactifs.

Dans un modèle de tâches, les actions utilisées pour atteindre un but varient beaucoup en fonction de la technologie disponible. Les buts sont des conditions à accomplir et ils sont plus stables que les actions à réaliser pour les atteindre [Diaper & Addison 92]. En conséquence, il est possible de définir un modèle dans lequel l'élément principal est la structure de buts/sous buts et que nous appelons **modèle de tâches minimales.**

En effet, D. Benyon définit une **tâche** - dans un univers composé seulement par des agents et des mécanismes (*devices*) - comme '*a goal plus a device*', c'est à dire**, les actions orientés buts (*goals*) que l'agent réalise par intermédiaire des mécanismes** [Benyon 92b]. A chaque mécanisme correspond un ensemble de ressources. Le but peut bien sur être atteint par l'utilisation d'une variété de mécanismes. En fonction du contexte (les autres agents et les mécanismes), l'agent formule les buts et sélectionne des mécanismes disponibles pour arriver aux buts. Selon [Benyon 92b], les agents réalisent leurs tâches pour arriver à des buts déterminés par **leur intention**. Avoir l'intention d'agir est un attribut humain qui manque aux ordinateurs : nous disons alors que les ordinateurs réalisent des fonctions ou des processus alors que les tâches sont réalisées par les personnes (ou plus généralement par des agents).

En conséquence, il est clair qu'en suivant la perspective d'outil présentée au chapitre 1, pour bien définir les besoins du système - qui vont permettre de choisir quels mécanismes offrir et quelles caractéristiques ces mécanismes doivent posséder pour soutenir les tâches - il faut tout d'abord connaître les buts de l'utilisateur pour ensuite proposer la spécification des mécanismes.

Cette orientation but est en fait déjà reconnue par d'autres travaux de l'ingénierie des besoins [Salvador et al. 95], [Anton et al. 94], [Van Lamsweerde et al. 95], [Dardenne et al. 93], [Sutcliffe & Maiden 93], [Yu & Mylopoulos 96], [Potts 95]. Cependant peu d'entre eux disent comment identifier les buts, les décomposer et puis les intégrer au processus de l'ingénierie des besoins. Par contre, le modèle de tâches minimales de TAREFA peut être facilement d'un modèle des tâches conventionnel comme montré dans le paragraphe sur la construction des modèles de tâches minimales.

Le modèle de tâches minimales est inspiré par la notion de **procédure minimale**, introduite par M.F.Barthet dans [Barthet 86]. Dans son travail pionnier, [Barthet 86] a proposé différentes types de procédures pour la réalisation d'une tâche avec un système. Cette

variabilité de modes opératoires permet de représenter plusieurs manières de réaliser une tâche identique et d'adapter le niveau de contrôle exercé par le système sur le travail des opérateurs :

- la **procédure prévue** prescrit la manière standard d'effectuer une tâche ;

- les **procédures effectives** décrivent les procédures réellement mises en oeuvre par les opérateurs experts.

- la **procédure minimale** indique les contraintes que l'opérateur doit impérativement respecter dans tous les cas, sans prendre en compte les automatismes développés par l'habitude dans le cas de séquences répétitives ;

Cependant, ce classification ne sert que pour les **tâches interactives**, c'est à dire, pour les tâches réalisées ***avec*** un système. Pour les **tâches existantes de l'utilisateur** réalisés sans système, on propose la classification analogue suivante :

- **tâche prescrite**, la description qui correspond à la notion de travail prescrit (la norme officielle, stable, régulière et reproductible, imposé par les règles de l'organisation); cette description est adopté globalement comme le mode opératoire des débutants ;

- **tâches effectives**, la description des tâches réelles existantes en respectant la façon de les faire; plusieurs tâches effectives sont en principe nécessaires pour rendre compte de la variété possible des pratiques réelles, sous la forme des variations par rapport à la tâche prescrite et des changements (comme automatismes, changement d'ordonnancements, etc) ajoutés par l'expérience de travail.

- **tâche minimale**, le modèle abstrait de buts des tâches en préservant la structure minimal des dépendances causales et d'ordonnancement des buts. Les tâches minimales sont indépendantes de la technologie (comme les tâches conceptuelles ou abstraites de CLG [Moran 81]). Les buts de l'utilisateur sont décomposés hiérarchiquement en sous-buts. Cette décomposition s'arrête quand les sous buts ne peuvent pas être décrits sans faire des hypothèses sur les actions spécifiques du niveau articulatoire de l'interface.

Plusieurs approches basées sur les tâches vus au chapitre I ont créé des abstractions des tâches où la notion d'indépendance de technologie est similaire. Ces abstractions sont respectivement appelées **tâches essentielles** (*essential tasks*) par [Constantine 95], **tâches composées** (*composite tasks*) par [Wilson & P. Johnson 96] ou **tâches généralisées** (*generalized tasks*) par [Lim & Long 94].

18.6.1Description du Modèle des Tâches Minimales

Le modèle de tâches minimales est un modèle abstrait et sans considération de technologie comme les modèles essentiels de [McMenamin & Palmer 84] et [Constantine 95], représentant ce que l'utilisateur veut réaliser et pourquoi. C'est un **modèle des intentions de l'utilisateur** plutôt qu'un modèle de ses actions. Par exemple, les actions et interactions nécessaires pour le but **s'identifier** sont considérablement différentes si l'utilisateur parle avec un employé de la banque au guichet de son agence ou s'il utilise un ATM mais le but reste le même dans les deux situations.

En plus de l'organisation hiérarchique des buts, il y a la description de leur ordonnancement et de leur interdépendance. L'ordonnancement est l'organisation temporelle dans laquelle l'utilisateur réalise ses tâches (séquence, entrelacement, parallélisme, etc).

L'interdépendance est une séquence imposée par les relations entre les entrées et sorties des tâches. Cette description est faite à l'aide de constructeurs. Les constructeurs du modèle des tâches minimales sont inspirés dans les constructeurs du modèle UAN - représentatifs des constructeurs typiques des modèles de tâches présentés dans la figure SIGN (voir chapitre 1).

18.6.1.1Modèle de tâches minimales et l'IB

Comme nous l'avons vu au chapitre 1, les modèles de tâches sont une manière adéquate de considérer le **point de vue de l'utilisateur** pour le développement d'un système parce qu'ils reflètent typiquement la manière par laquelle l'utilisateur conceptualise ses tâches. De telles descriptions peuvent suggérer de nouvelles manières de considérer les éléments et procédures des tâches, en permettant de les améliorer ou d'envisager de nouvelles activités associées à ces tâches. Donc, plus que de décrire l'activité actuelle de l'utilisateur, un modèle de tâche peut également être utile comme un outil d'Ingénierie des Besoins. Autrement dit, à partir de modèles de tâches existants il doit être possible d'envisager les tâches interactives à réaliser avec le soutien d'un système futur.

Pour bien choisir un modèle de tâche prescritif qui sert à l'IB, il faut répondre à 3 questions basiques [Lu 93] :

1. Quelles informations recueillir ?

2. Comment représenter les informations recueillies ?

3. Comment dériver les besoins du modèle de tâche ?

Les deux premières questions concernent le choix d'un modèle de tâches (et une notation associée) et la troisième concerne l'utilisation de modèles de tâches pour l'IB. Les paragraphes suivants vont détailler ces deux aspects.

18.6.1.2Choix d'un modèle de tâches

Une description succincte des modèles de tâches a été présenté au chapitre 1. Des deux modèles cités (à savoir MAD et CLG), nous retenons seulement MAD, pour deux raisons :

1. CLG ne permet pas le raffinement en tâche et sous-tâche, comme [Jambon 96] l'a remarqué; donc il n'est pas adéquat pour la description hiérarchique des tâches existantes ;

2. MAD dispose de méthodes rigoureuses pour l'analyse de la tâche.

18.6.1.3Utilisation du Modèle de tâches choisi pour l'IB

Spécifier les besoins d'un système à partir d'un modèle de tâches existantes n'est pas un travail simple. Il faut considérer qu'un système va proposer des nouvelles actions associés aux tâches pour arriver aux buts. C'est facile de constater qu'il y a plusieurs systèmes possibles pour soutenir le modèle de tâches minimales. Ces systèmes sont différents par rapport à la réalisation des buts, c'est à dire, par rapport aux actions qu'il faut réaliser pour arriver aux buts et qui les réalisent. Les buts de l'utilisateur restent basiquement les mêmes mais chaque réalisation est différente. Cela implique qu'il est possible, à partir des buts, de spécifier les tâches futures avec le système (les tâches interactives).

Dans l'approche TAREFA, les tâches futures sont proposées par l'intermédiaire des cas d'utilisation et les cas d'utilisation sont construits à partir des modèles des tâches minimales.

Les cas d'utilisation permettent d'établir :

- les propositions de nouvelles actions pour arriver à un but déterminé ;
- la découverte de nouveaux buts pour la nouvelle situation future.

Même si les sous-buts ne sont pas préservés pendant le développement, la description du modèle de tâche minimale peut servir comme un **point de départ** de la discussion et de la spécification des buts souhaités, en profitant du support flexible et de la possibilité d'envisager les situations des cas d'utilisation. Plus de détails sur les caractéristiques des cas d'utilisation adoptés par TAREFA et sur leur construction à partir d'un modèle de tâches minimales sont présentés au chapitre 6.

18.6.1.4Représentation des tâches Minimales

Etant donné que le modèle des tâches minimales est un arbre de buts/sous-buts derivé directement d'un modèle des tâches, on peut utiliser une notation hiérarchique similaire aux notations hiérarchiques du modèle de tâche MAD, qui sera utilisé dans le reste de cette thèse. Des exemples de l'utilisation de la notation sont montrés dans les figures 5.4, 5.5 et 5.6.

18.6.2Construction du Modèle de Tâches Minimales

Dans cette section, nous proposons une manière directes d'obtenir un modèle des tâches minimales à partir d'un modèle des tâches conventionnel MAD.

Dans les modèles de tâche par planification hiérarchique, dont un exemple typique est MAD, il y a la distinction buts/mécanismes. Le modèle de planification hiérarchique décrit une tâche sous forme d'un arbre hiérarchique constitué à partir d'items-tâches qui peuvent être décomposés recursivement en prenant en compte des notions de synchronisation et de relations logiques entre tâches. Les procédures sont décrites en faisant appel à un certain nombre de sous-buts, jusqu'au niveaux des actions, le plus bas niveaux explicitable. [Sebillote 87] a montré que la description d'une tâche peut nécessiter, selon les cas, de 2 à 7 niveaux qui se rapportent à quatre niveaux d'abstraction principaux, conformément à la figure 5.3 :

1. le **niveau abstrait** représente la formulation générale de la tache par intermédiaire du but à atteindre ;
2. le **niveau expert** décrit les procédures spécifiques d'un domaine précis ;
3. le **plus haut niveau commun** décrit des procédures générales indépendantes du domaine mais nécessaires pour réaliser les sous-tâches spécifiques ;
4. le **plus bas niveau explicitable** correspond aux actions élémentaires, qui sont indissociables des moyens (de la technologie) nécéssaires pour les réaliser

Un modèle de tâches minimales coïncide avec une description du niveau abstrait jusqu'au **plus haut niveau commun**. Les actions élémentaires du plus bas niveau explicitable sont écartées pour l'IB, même si son utilisation durant la conception est conseillée. Bref, en pratique le dernier niveau de description est éliminé car il est dépendant des technologies disponibles pour réaliser les actions. En fait, l'intention est de retenir les buts (chacun d'entre eux peut être accompli par différentes actions) et **non** les actions spécifiques qui sont réalisées [Potts 95].

28)

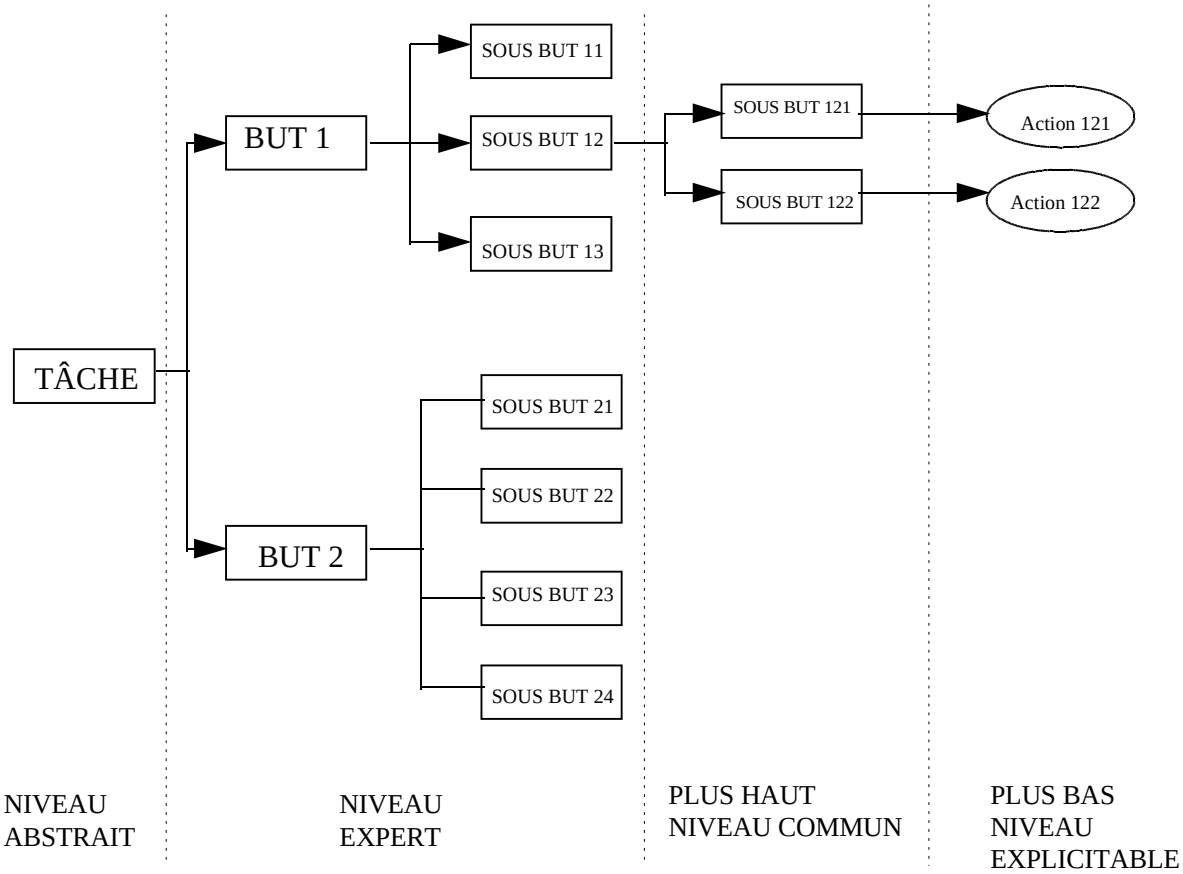


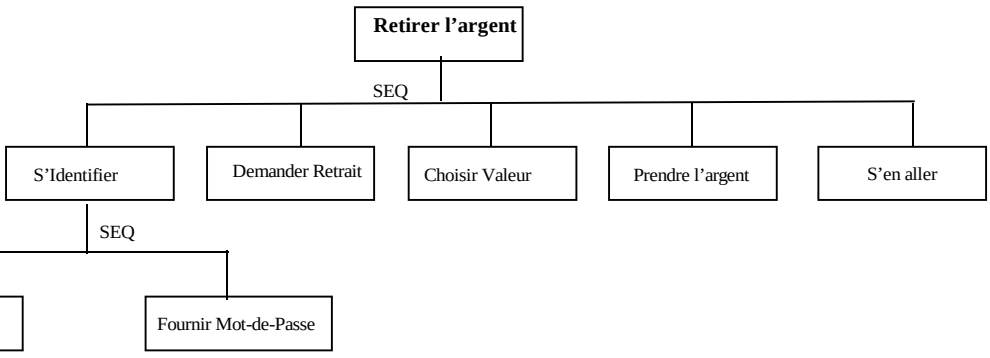
Figure 5.3 - Description d'une tâche par planification hiérarchique et les quatre niveaux d'abstraction

Dans le cas extrême, c'est à dire, s'il n'y a pas une analyse de la tâche faite ou s'il n'est pas possible de le faire, on peut choisir les signes associés à la catégorie 'Action' à partir du modèle ontologique comme une version préliminaire des buts de l'utilisateur.

18.6.3Modèle des tâches minimales de l'ATM

Pour chaque tâche d'utilisateur identifiée et modélisée, à savoir 'Retrait de l'argent', 'Demande de Solde', 'Demande de Relève', un modèle de tâche minimale est construit, cf. respectivement les figures 5.4, 5.5 et 5.5.

29)

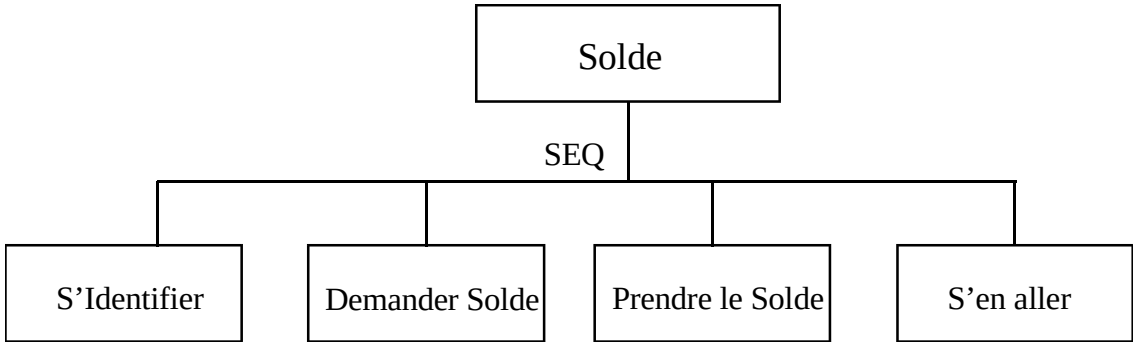


30)

31)

Figure 5.4 - Modèle de tâches minimales de la tâche 'Retrait de l'argent'

32)



33)

Figure 5.5 - Modèle de tâches minimales de la tâche ‘Demande de Solde’

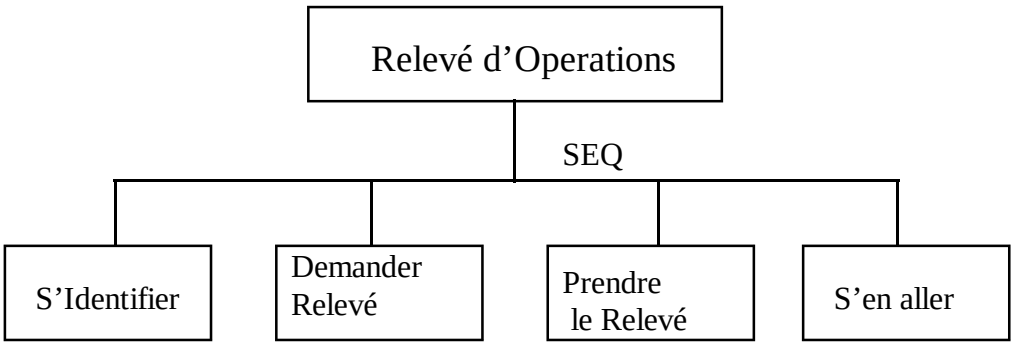


Figure 5.5 - Modèle de tâches minimales de la tâche ‘Demande de Relevé’

18.7Le Modèle de Contexte Organisationnel

L'objectif du modèle du contexte organisationnel de TAREFA est de representer le cadre organisationnel dans lequel les tâches sont réalisées. Nous pensons que quand on veut concevoir un système pour informatiser un ensemble de tâches, il ne suffit pas d’analyser et de modéliser les buts et tâches de chaque utilisateur. Il faut étudier les processus de travail, leurs activités et les conditions pour les réaliser, leurs ressources, les rôles responsables et les possibles relations entre ces plusieurs composants. Le terme générique pour cet étude est ‘Analyse des Systèmes’ [Mumford 83] [Weinberg 88], dont l’origine est bien antérieure à la préoccupation actuelle des IHM .

En fait, la reconnaissance par la communauté IHM du rapport entre IHM et le contexte organisationnel est récente. Traditionnellement, l’IHM avait tendance à considérer le contexte organisationnel comme externe au système donc hors de ses préoccupations. Pourtant, aujourd’hui, la nature sociale de l’utilisation des systèmes (voir les CSCW) et même certains problèmes des systèmes mono-utilisateur dûs probablement à l’influence du contexte organisationnel, font réviser cette position. Aujourd'hui plusieurs auteurs défendent, comme nous, la nécessité d’avoir une vue large de l’organisation comme montre les travaux de [Jones 95], [Hughes 95] et [Nardi 96], qui montrent différentes points de vue sur ce qu'est le contexte organisationnel et comment il influence l’IHM.

La modélisation du contexte organisationnel est très importante pour établir un contexte pour les tâches des utilisateurs (avec ou sans le système) et doit être complémentaire à la modélisation des tâches. En effet, la plupart des méthodes d’analyse de tâche ont un point en commun : l’analyse est basée sur les **aspects individuels** de la réalisation des tâches et leur modèles de tâches représentent les tâches d’un individu. Cependant, au contraire des

procédures exécutées par les machines, les tâches sont réalisées dans un **cadre social et organisationnel**.

C'est dans ce cadre que les notions d'utilisabilité et d'utilité seront déterminées.En effet, les problèmes d'utilisabilité dans les systèmes interactifs doivent être définis plutôt comme un accomplissement infructueux (*unsuccessful fulfillment*) des buts de l'utilisateur et de l'organisation [Gillmore 95] que comme des problèmes de performance humaine, définis typiquement en termes de vitesse et de mesures d'erreur. L'utilisabilité ne peut pas être mesurée par l'étude d'un produit isolé parce que l'utilisabilité n'est pas une propriété d'un système mais surtout une relation entre le système et ses utilisateurs, qui ne peut qu'être déterminé dans un contexte [Goransson 87].

En pratique, la notion de ***contexte*** est très difficile à définir et comme le dit [Cockton et al. 95] la plupart des définitions d'usage général son inadéquates. [Boy 95] fait la distinction entre deux types de contexte à prendre en compte dans une situation d'utilisation : **le contexte permanent** (ou stationnaire), qui est associé à l'environnement social et organisationnel où les activités se déroulent ; et le **contexte éphémère**, qui est associé aux situations émergentes de la dynamique de la situation d'utilisation. Comme le montre les paragraphes suivants, le contexte organisationnel est un **contexte** de type **permanent**.

Dans la plupart des situations concernées, les tâches font partie d'un processus de travail où, soit les tâches impliquent un certain nombre de personnes soit les tâches sont distribuées dans le temps. Ce processus de travail est souvent appelé **processus de l'entreprise** (*business process*) dans la littérature. Comme les tâches simples, des processus de l'entreprise peuvent aussi être supportés par la technologie, en particulier un système informatique, et c'est souvent une manière efficace de résoudre les problèmes de la situation concernée.

Les organisations[11] sont faites d'acteurs sociaux qui coopèrent pour atteindre plusieurs buts impossibles à atteindre sans cette coopération. Ces acteurs dépendent les uns des autres pour des parties de leur travail et ils ne sont pas totalement libres de choisir leurs buts ou les façons de réaliser leurs tâches. Différents cadres sociaux ou organisationnels peuvent mener à des choix différents, parfois très distincts.

Des processus de l'entreprise impliquent la réalisation de plusieurs **activités** pour arriver aux **objectifs**, en analogie aux tâches individuelles pour arriver aux buts. Chaque activité peut dépendre de ressources, parfois connues comme les objets de l'activité [Carey 89]. Cette dépendance provoque des contraintes sur la performance des activités - ***quand*** elles peuvent être réalisées, ***par qui*** et dans ***quelle séquence***. Plus une activité dépend d'une ressource, plus c'est difficile de la réaliser comme une tâche simple et il est probable qu'elle doit être décomposée en plusieurs tâches liées. Elle forme alors un processus de l'entreprise.

En fait, un processus a de ***multiples dépendances*** et sa réalisation change d'un ressource à l'autre en fonction de la disponibilité de ses ressources. Quand une ressource est disponible, l'activité dépendante peut être réalisée ; sinon, l'activité attend sa disponibilité de cette ressource. Cette disponibilité peut être une conséquence de **règles** internes de l'entreprise ou d'**événements** particuliers originaires de la dynamique du processus.

TAREFA considère les mêmes types de ressources que ceux définis dans [Newman 95], mais de façon plus générique. Les types de ressources de TAREFA sont :

- les **concepts**, qui représentent l'information (permanente ou temporaire - comme entrée/sortie) utilisée pour la réalisation d'une activité ;

- les **rôles**, qui représentent les personnes et les groupes qui ont les capacités et des responsabilités spécifiques pour la réalisation des activités ;

11 Le concept d'***organisation*** a ici une interpretation large, en faisant reference à la fois à un domaine limité d'activité d'une entreprise ou à toute l'entreprise concerné.

- les autres processus associés, s'il y en a.

Pour certains auteurs, la modélisation du contexte est une approximation de la modélisation conceptuelle, c'est à dire, la modélisation des concepts de l'utilisateur [Long and Dowell 89]. Dans cette thèse, l'envergure de 'contexte' est délibérément limitée et nous adoptons une définition spécifique au contexte organisationnel.

Les sections suivantes présentent respectivement le modèle de contexte organisationnel TOCO proposé par TAREFA (section 2.7.1), les approches pour sa construction (section 2.7.2) et la modélisation du contexte organisationnel de l'ATM (section 2.7.3).

18.7.1Description du modèle de contexte organisationnel TOCO

La modélisation du contexte organisationnel dans TAREFA commence par l'identification de deux contextes interconnectés :

- **le contexte individuel**, où les aspects pour la réalisation des tâches individuelles sont considérés ; et
- **le contexte organisationnel**, où les aspects pour la réalisation des processus de l'entreprise sont considérés.

Les relations entre le contexte organisationnel et le contexte individuel sont claires : la réalisation d'un processus de l'entreprise implique la réalisation de plusieurs tâches individuelles. Cependant, bien que ces deux contextes soient complémentaires, la détermination du contexte organisationnel est une activité complexe : elle ne peut pas être dérivée à partir de l'union de plusieurs contextes individuels associés.

Nous pensons que les informations sur le contexte doivent être modélisées pour devenir accessibles à l'équipe de développement pendant la conception du système interactif et dans cette thèse nous nous concentrons sur la modélisation du contexte organisationnel.

En effet, le contexte individuel est en partie modélisé par des modèles de tâches plus raffinés (comme par exemple MAD [Scapin & Pierret-Golbreich 89], qui représentent les rôles, les objets de la tâche et les pre-conditions et post conditions pour sa réalisation) et par des modèles de représentation des connaissances (comme les blocs de connaissance [Boy 95],qui représente les pré-conditions de déclenchement, les conditions contextuelles et les conditions anormales d'une procédure).

Le modèle TOCO (Tasks Organisational Context) a pour objectif de modéliser le contexte organisationnel dans l'approche TAREFA. TOCO a été proposé dans [Pimenta & Barthet 96b] mais sa description est plus complète dans [Pimenta & Barthet 96a].

18.7.1.1Eléments de TOCO

Le modèle TOCO a un nombre de composants inter-reliés, chacun représentant un aspect particulier du contexte organisationnel. Les composants du modèle TOCO décrivent :

a) les activités organisationnelles ;

b) les rôles organisationnels ;

c) les objectifs organisationnels ;

d) les concepts du domaine ;

e) les événements organisationnels ;

f) les règles organisationnelles;

g) les perspectives de rôle.

Une **activité organisationnelle** est une généralisation du concept classique d'action de la Théorie de l'Activité. Une activité est une formulation et un ordonnancement particulier de tâches pour accomplir les objectifs organisationnels. Il peut y avoir des combinaisons alternatives de tâches pour la même activité, en fonction des règles organisationnelles ou des événements organisationnels.

Les **rôles organisationnels** définissent les rôles individuels, les groupes ou les unités organisationnelles (et leur relations) responsables d'une activité. Bien sûr, une personne peut jouer plusieurs rôles individuels ou participer à plusieurs groupes ou unités.

Dans les **objectifs organisationnels**, il est décrit à un très haut niveau et d'une manière structurée, pourquoi un ensemble d'activités est réalisé. Malgré la difficulté inhérente de les recueillir et de les représenter, ces objectifs sont une notion très importante pour la compréhension des activités. Usuellement, les objectifs sont fonctionnellement subordonnés à d'autres objectifs, ceux-ci recursivement aussi subordonnés à d'autres. Les objectifs de plus haut niveau correspondent aux objectifs de l'entreprise.

Les **concepts du domaine** définissent l'ontologie du domaine du problème, y compris l'ensemble de types d'objet, leurs relations et attributs. Ces concepts sont naturellement utilisés dans la description des autres composants du contexte organisationnel, comme les activités, les règles et les événements.

La description des **événements organisationnels** consiste en pré-conditions et post-conditions qui définissent la situation respectivement avant et après la réalisation d'une activité. Ces conditions indiquent des situations de succès et d'erreur. Bien sûr, la liste de événements ne peut pas être complète parce que la réalisation d'une activité est située, en fonction de plusieurs caractéristiques émergentes qui influencent et sont influencées par l'activité [Suchman 87]. Cependant, les événements doivent être faciles à tracer pour permettre la modification du système quand la définition de ces conditions change.

Les **règles organisationnelles** définissent les règles de l'entreprise associés aux contraintes, stratégiques ou politiques qui servent à guider ou à restreindre les activités organisationnelles.

La vérification qu'une **situation est ou non appropriée à la réalisation d'une activité** est basée sur les conditions dynamiques - les événements - ou sur les conditions statiques - les objectifs et règles organisationnelles déterminés. Les changements de conditions (par exemple les modifications de l'organisation du travail, les changements de règles politiques internes, les changements de responsabilités de rôles dues à qualification des utilisateurs, etc) peuvent impliquer de nouveaux besoins ou le changement des besoins.

Les espaces de travail et les perspectives sont des mécanismes de structuration pour organiser et interconnecter les composants de TOCO.

Un **espace de travail** est un espace virtuel dans un domaine déterminé du problème avec les concepts du domaine pour la réalisation d'une (ou plus) activité(s) organisationnelle(s) par un (ou plus) acteur(s) jouant un rôle organisationnel afin d'accomplir les objectifs organisationnels, suivant l'application des règles organisationnelles et l'occurrence des événements organisationnels. En fait, une modélisation complète d'un contexte organisationnel est l'union des espaces de travail pour tous les domaines de problème d'une organisation.

Une **perspective** de rôle est une restriction d'un espace de travail selon le point de vue d'un rôle particulier.

18.7.1.2Relations intra-contextuelles et inter-contextuelles

Les relations intracontextuelles sont des relations entre les éléments d'un même contexte

organisationnel. Par exemple, la relation de Responsabilité entre un rôle et une activité - Responsabilité (rôle, activité) d'un contexte organisationnel - ou la relation de Utilisation entre un concept du domaine et une activité d'un contexte organisationnel.

Les relations intercontextuelles sont des relations entre des éléments de modèles faisant partie de l'information contextuelle mais dans differents contextes organisationnels. Par exemple, la relation d'Identité entre un signe du modèle ontologique et un concept du domaine dans le modèle du contexte organisationnel.

TOCO a certaines relations intracontextuelles et intercontextuelles pré-definies - voir figure 5.7 et figure 5.8- mais il permet la modification, la spécialisation ou même l'extension de ces relations si c'est nécessaire. Les relations qui sont detaillées dans la figure 5.8 sont celles qui sont les plus frequemment rencontrés et qui apportent le plus d'information utile pour la construction du modèle.

Relations intracontextuelles pre-definies	**Relations inter-contextuelles pre-definies**
Relation de Responsabilité : *Responsable(rôle, activité)*	Relation d'Identité : *Identité (Rôle de Activité, Rôle de Tâche)*
Relation de Post-Condition : *Post-Condition (Evénement, Activité)*	Relation de Set-Membership : *Set-Membership(Rôle de Activité, Rôle d'une tâche)*
Relation de ¨Pré-Condition : *Pré-Condition (Evénement, Activité)*	Relation de Composition : *Composition(Activité Organisationnel, Tâche)*
Relation d'Utilisation : *Utilisé-en (Concept du Domaine, Activité ou Evénement ou Règle)*	Relation de Généralisation : *Généralisation (Concept du Domaine, Objet de Tâche)*
Relation de Motivation : *Motive (Objectif, Activité)*	Relation de Contrainte, comme en [Cockton et al. 96] : *Contrainte (Règle Organisationnel, Tâche de l'Utilisateur)*
Relation de Set-Association : *Set-Association (Rôle-groupe-social, Rôle-individu)*	
Relation de Règlement : *Règle de (Règle, Activité)*	

Figure 5.7 - Relations intercontextuelles et intracontextuelles pré-definies de TAREFA

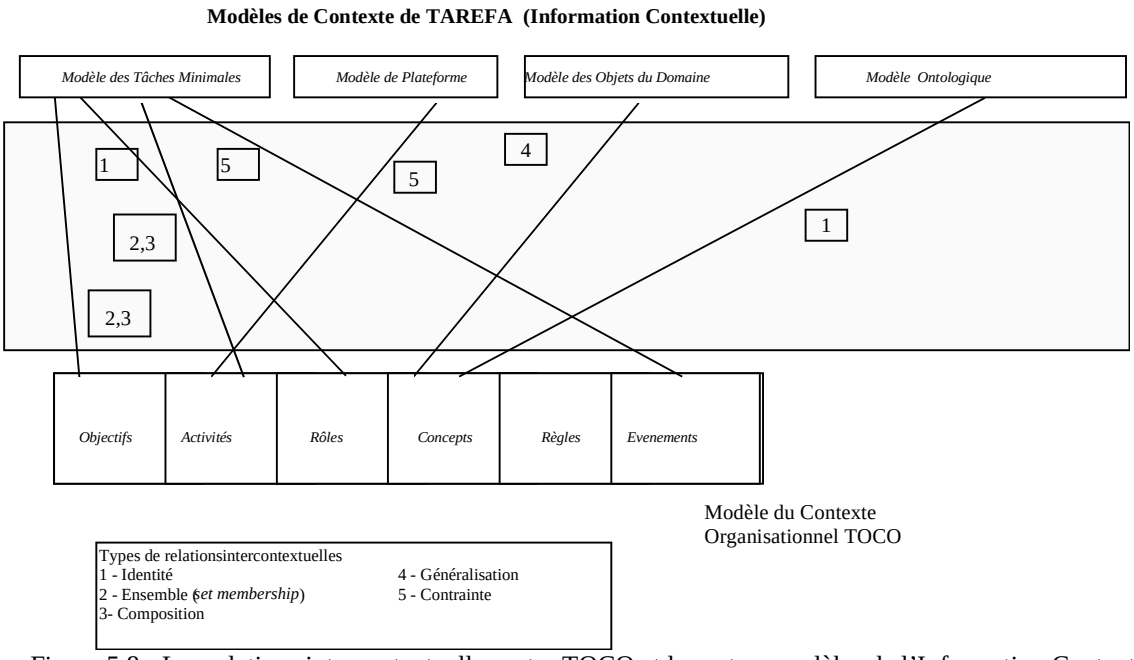


Figure 5.8 - Les relations intercontextuelles entre TOCO et les autres modèles de l'Information Contextuelle

18.7.1.3Relations entre le Contexte et les autres modèles du développement

L'utilisation d'informations sur le contexte pendant le développement du système est un problème complexe [Cockton 95]. Avec TAREFA, cette utilisation est orientée par les relations existantes (les termes utilisés en anglais par [Cockton 96] sont *anchor points* ou *links*) entre le modèle TOCO, les autres modèles du contexte (l'information contextuelle) et les autres modèles du développement (l'information opérationnelle et l'information du système), à travers les deux activités suivantes :

1. construire les modèles pour le développement (les modèles de l'information opérationnelle et de l'information du système)
2. établir les relations entre ces modèles et le modèle TOCO, en utilisant les relations pré-definies - voir figure 5.9 - ou en créant des nouvelles relations.

Les relations entre les autres modèles de contexte et les modèles de développement sont faites par l'intermédiaire du modèle TOCO, qui est un modèle intégrateur de modèles de contexte. En effet, TOCO fourni un cadre organisationnel pour caractériser les espaces de conception (*design spaces*).

Relations pré-definies entre TOCO et les modèles de développement
Contrainte pour (une décision) de développement
Transformation d'une information du contexte en un artefact du développement, la relation la plus courante dans les approches de génie logiciel ;
Critère pour l'évaluation ;
Source de cas de tests;
Identité (*link* 'these are the same thing' [Cockton 96])

Figure 5.9 - Relations pré-définies entre TOCO et les modèles de développement

18.7.2Construction du modèle TOCO

Il y a trois approches principales pour la construction d'un modèle de contexte organisationnel en utilisant TOCO : l'approche descendante, l'approche ascendante et l'approche incrementale.

Avec l'approche descendante, TOCO doit être modélisé avant les autres modèles de contexte, notamment avant le modèle des tâches minimales et le modèle des objets métiers du domaine. Les autres modèles sont un type de 'raffinement' des composants de TOCO,

Avec l'approche ascendante, il faut d'abord commencer par l'identification et la modélisation des tâches pour ensuite analyser et représenter l'information organisationnelle (rôles, règles, événements, etc) autour des tâches sélectionnées (par exemple celles qui seront prises en compte dans un système informatique).

Avec l'approche incrementale, ni les composants de TOCO ni les autres modèles de l'information contextuelle ne sont définis nécessairement dans un ordre déterminé. La construction d'un modèle TOCO est itérative et utilise (simultanément ou non) plusieurs techniques de recueil pour :

- d'abord déterminer un ensemble préliminaire d'éléments de TOCO ;
- à chaque itération ajouter ou corriger des informations ou plus de précisions aux informations préliminaires initiales.

Le choix entre ces trois approches suit les critères suivants :

1. si les connaissances macroscopiques sur le domaine du problème sont plus facilement disponibles - soit sous la forme d'un modèle de domaine, d'un modèle de l'entreprise ou d'un modèle de flux de travail (*workflow*) existants soit les cadres sont disposés à

les fournir - il faut utiliser l'approche descendante ;

2. si les connaissances microscopiques sur les tâches et les procédures sont plus facilement disponibles - soit sous la forme de modèles de tâches existantes soit les opérateurs sont disposés à les fournir - il faut utiliser l'approche ascendante ;

3. s'il n'est pas possible de déterminer quelles sont les connaissances les plus faciles à obtenir ou s'il n'y a pas d'informations suffisantes pour décider, il faut utiliser l'approche incrementale.

18.7.3Modélisation du Contexte Organisationnel de l'ATM

Même si l'ATM est un système typiquement du type '*walk up and use*' [Blandford et al. 93], où l'influence du contexte organisationnel est relativement faible, il est possible de décrire le modèle du contexte organisationnel et de l'utiliser. Un extrait de description du contexte organisationnel de l'ATM est décrit dans le figure 5.10. Même délibérément incomplete, cette description permet de comprendre les principales caractéristiques du modèle TOCO.

A l'heure actuelle, le modèle TOCO est modélisé par une notation tabulaire en utilisant une syntaxe de français structuré pour décrire ses éléments. Cela permet au modèle d'être compris facilement par les acteurs.

Le figure 5.10 montre un extrait du modèle de contexte organisationnel de l'étude de cas ATM. Par simplicité, l'extrait ne montre que les éléments de l'espace de travail Management de Compte. Les autres espaces de travail du domaine de l'exemple, comme par exemple l'espace Investissement et l'espace Devise ne sont pas considérés.

Composant du contexte Organisationnel	**Espace de Travail** : Management des Comptes
Objectifs Organisationnels	O1. Accueil de bonne qualité aux clients O2. Sécurité et Intégrité des transactions O3. Rapidité des Services ....etc.....
Rôles Organisationnels	R1. Client R2. Employé-Accueil R3. Employé-Caisse ...etc..... R4. Equipe d'Accueil (set of R2) R5. Equipe de Caisse (set of R3) R6. Agence
Règles Organisationnelles	U1. SI transaction = retrait de l'argent ET client = client de l'agence ALORS limite = solde_compte SINON limite = 3000 FF. U2. SI le client n'est pas identifié ALORS il ne peut pas prendre l'argent U3. SI transaction = retrait de l'argent ET client = 'client spécial' ALORS limite = 10000 FF. U4. SI la quantité d'argent demandé dépasse la limite ALORS il ne peut que retirer la limite U5. SI le client ne les prend pas ALORS il faut garder sa carte et les billets U6. SI la quantité dépasse le solde du client ET le client n'est pas 'client spécial' ALORS il ne peut que retirer le valeur disponible ...etc..
Activités Organisationnelles	A1. Retrait d'Argent A2. Demande de Solde A3. Demande de Relevé des Opérations A4 Dépôt A5 Virement ...etc... A6. Ouverture de Compte A7. Fourniture de Chéquier A8. Changement de Mot de Passe A9. Fourniture des Cartes
Evénements	E1. Retrait Permis E2. Retrait Interdit par manque de solde E3. Retrait demandé E4. Solde Demandé E5. Solde Calculé E6. Relevé d'Opérations Demandé

Organisationnels	E7. Relevé d'Opérations Disponible ...etc...
Concepts du Domaine	C1. Client (Client-id, client-nom, addresse) C2.Compte (compte-id, compte-type, client-id) C3. Banque (banque-id, banque-num, etc) C4. Agences (agence-id, agence-addresse, agence-responsable-id, etc) C5. Carte (carte-id, client-id, banque-id, agence-id, compte-id, etc) ...etc....
Relation de Responsabilité *Responsable(rôle, activité)*	*Responsable (R3,A1), Responsable (R3,A2), Responsable (R2,A7),* *Responsable (R3,A3), Responsable (R3,A4), Responsable (R2,A8),* *Responsable (R3,A5), Responsable (R2,A6), Responsable (R2,A9)* *,etc*
Relation de Post-Condition *Post-Condition (Evénement, Activité)*	*Post-Condition (E2, A1),* *Post-Condition (E7, A3),* *etc.*
Relation de ¨Pré-Condition *Pré-Condition (Evénement, Activité)*	*Pré-Condition (E1,A1), Pré-Condition (E4,A2),* *Pré-Condition (E3,A1), Pré-Condition (E6,A3),* *etc.*
Relation d'Utilisation *Utilisé-en (Concept du Domaine, Activité ou Evénement ou Règle)*	*Utilisé-en (C1, A1),* *Utilisé-en (C1, E3),* *Utilisé-en (C1, U1),* *etc.*
Relation de Motivation *Motive (Objectif, Activité)*	*Motive (O2, A8)* *etc.*
Relation de Règlement *Règle de (Règle, Activité)*	*Règle de (U1, A1)* *etc.*

Figure 5.10 - Un extrait du modèle de contexte organisationnel du espace de travail Management de Compte de l'étude de cas ATM

Les objectifs organisationnels O1, O2 et O3 expriment des propriétés de très haut niveau qui doivent être considérés pour n'importe quel service offert à un client, soit par les employés d'une agence, soit par une machine. Bien sûr, les notions d'*accueil de bonne qualité*', '*sécurité*', '*intégrité*' et '*rapidité*' doivent être précisés.

Dans une banque, les rôles organisationnels sont les unités (comme une agence R6), les groupes (comme les équipes R4 et R5) ou les rôles individuels (comme R1, R2 et R3). Un acteur peut jouer plusieurs rôles individuels (par exemple, un employé peut être client de sa banque) ou participer à plusieurs groupes (par exemple, un employé peut faire partie de l'équipe R4 et R5).

Les règles organisationnelles dans le figure 5.10 concernent parfois une activité spécifique (comme la règle U1, applicable à l'activité Retrait d'argent) parfois des règles d'ordre général (comme la règle U2). Comme il est souvent le cas dans les organisations, il peut y avoir des règles contradictoires. Par exemple, les règles U1 et U3 du figure 5.10. Si nous considérons le cas d'un client de l'agence qui est à la fois client spécial, nous avons une contradiction. Pour la résoudre, il est possible de changer une des règles ou de décider que les deux sont valables et ajouter une autre règle (par exemple U4 : 'S'il y a une contradiction sur la limite de retrait du client, il faut adopter la valeur la plus petite'). En fait, si les règles organisationnelles peuvent être changées, alors le **contexte** est appelé **révisable**, sinon le contexte est considéré comme un **contexte fixe** [Cockton 95].

Par simplicité, les événements E1 à E7 montrent un sous ensemble de situations strictement associées à l'activité retrait d'argent. (A1).

Les concepts du domaine C1 à C5 sont les concepts considérés nécessaires par l'organisation pour l'activité retrait d'argent (A1) dans l'espace de travail Management des comptes. Ces concepts sont naturellement utilisés par les descriptions des activités organisationnelles, des règles organisationnelles et des événements organisationnels dans le même espace de travail.

Les relations montrées au figure 5.10 sont des exemples des relations intracontextuelles entre les éléments du contexte organisationnel pour le Management des comptes. Par exemple, '*Utilisé-en (C1,A1)*' signifie que le concept C1 (voir les concepts du domaine) est

utilisé dans la description de l'activité A1 (voir les activités organisationnelles).

18.8Modèle des Objets Métiers

Le but de cette modélisation est globalement d'énumérer les objets métiers. Ces objets sont typiquement les objets impliqués dans les tâches (et activités) existantes des utilisateurs et dans les tâches interactives et dans les fonctions du système.

18.8.1Description du Modèle des Objets Métier du Domaine

Un **domaine** est un ensemble d'applications, actuelles et futures, caractérisées par un ensemble commun de données et d'objectifs en relation [Peterson 91]. Il y a autant de types de modèles de domaine que de paradigmes d'analyse : les plus utilisés sont les approches centrées activités, centrées données et centrées objets [Benyon 96]. En général, un domaine possède un ensemble de données et de procédures, qui est commun à toutes les applications de ce domaine. Un **modèle de domaine orienté objet** encapsule ces données et procédures en **objets métiers**. La recherche d'informations appropriées pour une nouvelle conception orientée objet est plus simple si elle est réalisée dans un modèle d'objets métiers d'un domaine adéquat à la nouvelle application. Evidemment, chaque application peut ajouter d'autres objets spécifiques selon ses besoins particuliers. Néanmoins, le modèle de domaine sert comme cadre de référence pour le développement.

Un modèle de domaine orienté objet suit la tendance de modélisation du composant statique des modèles de données orientées objets et il est typiquement composé par :

- un ensemble de classes, qui correspondent aux entités significatives du domaine ;
- l'organisation des classes en hiérarchies de généralisation/spécialisation (hiérarchie 'is-a' avec les notions de classe, superclasse et sous-classe) et agrégation (hierarchie 'part-of' avec les notions de classe composée et de classe composante).

18.8.2Construction du modèle

Dans TAREFA, la modélisation des objets du domaine du problème peut être faite de deux façons différentes :

a) créer un nouveau modèle du domaine, c'est à dire, identifier, acquérir et représenter l'information pertinente sur les objets métiers du domaine ; cette alternative est intéressante s'il n'y a pas un modèle existant ou si le modèle existant n'est pas (ré)utilisable ; pour la construction, l'analyste peut soit utiliser les concepts du modèle ontologique pour identifier les objets métiers du domaine du problème qui appartiennent aux catégories agent ou objet, soit utiliser une méthode d'analyse orientée objet comme par exemple OOA ou OMT ;

b) (re)utiliser les objets disponibles dans un modèle du domaine existant ; cette alternative est intéressante si c'est la première fois qu'on travaille dans ce domaine et s'il y a un modèle de domaine existant. Dans le cas de modèles orientés objets, les modèles existants peuvent être non seulement des modèles résultants d'un processus d'Analyse du Domaine mais aussi des modèles de données produits par une approche non orienté objet (comme p.ex. un modèle ERA [Chen 76]) ou des modèles d'information de l'entreprise (comme p.ex. [Bubenko 94]).

18.8.3Le modèle des objets métier de l'exemple ATM

Dans le cas de l'ATM, nous avons adopté la deuxième alternative parce que nous avons trouvé un modèle existant dans [Wirfs-Brook et al. 90]. Un lecteur plus intéressé pourra trouver également plusieurs modèles de domaine pour l'ATM dans [Benyon 96].

La méthode conduite par responsabilité (*responsability driven*) présentée par [Wirfs-Brook et al. 90] commence par la définition des classes candidates. Son application à l'exemple ATM a identifié cinq types de classes candidates :

1. Classes de Périphériques (*Device Classes*), pour manipuler les dispositifs d'entrée (comme par exemple le clavier numérique) et les dispositifs de sortie (comme par exemple l'imprimante et l'écran);

2. Classes d'Entrée Spéciale (*Key Classes*), pour les dispositifs spéciaux d'entrée (comme par exemple la touche 'Cancel');

3. Classes de Transaction, pour la manipulation des transactions (comme par exemple les Transactions de Dépôt et les Transactions de retrait d'argent) ;

4. Classes d'Interface Utilisateur, pour encapsuler les manipulations génériques d'interface utilisateur (comme par exemple les menus, les messages et les formulaires) ; et finalement

5. Classes du Domaine du Problème, pour les objets métier.

34)

Nous nous sommes interéssés au dernier type de classes, qui fait partie d'un modèle orienté objet du domaine du problème. Les classes du domaine du problème (et leurs attributs) identifiées par [Wirfs-Brook et al. 90] sont donc :

- **Compte** (*Account)* - pour représenter les informations sur le compte :
  - la description du compte du client - y compris l'identification du client, son adresse, le code de l'agence, l'adresse de l'agence, le code de la banque, le type de compte, la limite disponible pour opérations, le solde; et
  - un historique des transactions récentes - y compris la date, le code de l'opération, la valeur de la transaction ;

35)

- **Reçu** (*Receipt)* -pour représenter les informations qui doivent être imprimées sur le reçu, y compris :
  - date, heure, nom du client, code du client, code-ATM, nom-ATM, adresse-ATM, nom-banque, opération, valeur, résultat

36)

- **Carte** (*Card)* - pour représenter l'information présente dans la carte du client, y compris :
  - identificateur de la carte (card ID), identificateur du client (client ID), nom-client, code-banque, code-agence, limite-carte, mot-de-passe du client (client_PIN)

18.9Le Modèle de Plateforme

Le modèle de plateforme (aussi appelé modèle de moyens techniques) est une image de la

configuration existante du système. Cette configuration correspond à l'ensemble des moyens informatiques disponibles pour la réalisation du système, y compris les dispositifs matériels (CPU, périphériques, réseaux, etc) mais également les dispositifs logiciels (systèmes d'exploitation, bases de données, gestionnaires d'interface comme X Windows, Motif, etc.). Le modèle de plateforme ne constitue qu'une abstraction qualitative et non quantitative de la configuration sous-jacente car il ne prend en compte que la nature des ressources et non leur nombre, volume, capacité, puissance, etc.

Le modèle de plateforme est à la fois une ressource et une contrainte de développement. La présentation du modèle de plateforme est une liste des dispositifs, comme cela est illustré sur la figure 5.11. Le modèle de plateforme est raisonnablement stable dans le temps, sauf en cas de changements successifs de paliers technologiques ou en cas d'une augmentation de budget permettant l'achat de nouveaux dispositifs.

La construction du modèle de plateforme est l'élaboration de la liste à partir des informations recuillies dans l'environnement de travail.

18.9.1l'exemple ATM

L'extrait du modèle de plateforme de l'ATM montré dans la figure 5.11 est constitué uniquement d'un ensemble de dispositifs matériels disponibles actuellement dans la banque. Les logiciels ne sont pas considérés dans cet extrait.

A chaque element du modèle est attribué un identificateur (Id du Matériel) et une description succincte est faite (voir figure 5.11).

- MATERIEL
  - Utilisation d'une carte pour chaque utilisateur de l'ATM
  - ATM composé par :
    - Un clavier alphanumérique
    - Un écran
    - 'Cash Container' pour manipuler/compter les billets
    - 'Cash Dispenser' pour éjecter les billets
    - Lecteur de Carte pour l'insérer/éjecter la carte
    - Une imprimante 40 colonnes pour les reçus

Id du Matériel	Description
M1	Carte Bancaire (physique)
M2	Un clavier alphanumérique
M3	Un écran
M4	'Cash Container' pour manipuler/compter les billets
M5	'Cash Dispenser' pour éjecter les billets
M6	Lecteur de Carte pour insérer/éjecter la carte
M7	Une imprimante 40 colonnes pour les reçus

Figure 5.11 - Modèle de Plateforme de l'ATM

19.Déterminer les Besoins de l'Utilisateur

La Détermination des Besoins de l'Utilisateur est une étape très difficile pour les informaticiens parce qu'il n'y a aucune méthodologie pour l'identification (*elicitation*) et l'explicitation des besoins réels des futurs utilisateurs du système à construire.

En effet, dans la plupart des approches traditionnelles, cette détermination est laissé au 'bon analyste', qui possède une longue liste de qualités et de capacités tel un super-homme. C'est en général un analyste expérimenté ayant déjà développé un certain nombre de systèmes

dans le même domaine de l'application. Ces approches sont appelées Approches Centrés Analystes et reposent sur des suppositions implicites. Par exemple, "1. l'utilisateur est un expert du domaine du problème et il sait très bien ce qu'il veut d'un système " ; 2. " Si le bon analyste veut déterminer les besoins de l'utilisateur, il suffit de les lui demander correctement". En pratique, le terme 'capture des besoins ' est souvent utilisé pour ces approches. Pourtant nous pensons, comme [Bubenko 94], que le terme est inadéquat parce qu'il indique que les besoins sont 'déjà là' et qu'il ne faut qu'une méthode ou un outil pour les obtenir à partir des utilisateurs (en anglais, cela s 'appelle '*elicit requirements **from** user*').

Ce rôle central de l'analyste dans le processus de détermination est abandonné en faveur d'une participation plus ample de l'utilisateur adopté par les Approches Coopératives entre Analyste et Utilisateur. Dans ces approches les analystes déterminent les besoins avec les utilisateurs (en anglais, cela s'appelle '*elicit requirements **with** user*').

Cette coopération est typiquement centrée sur la communication analyste-utilisateur. En général cette communication est directe (oral) et considérée comme très difficile par les deux parties impliqués. Cette difficulté, comme nous avons vu à la section ? ? - problèmes de l'IB, est due au fait que chacun a :

- un vocabulaire particulier, où pour différentes personnes un même mot peut avoir différentes significations ou un même objet peut avoir différents noms.

- une vision très particuliere du problème, de l'univers et des besoins du système informatique futur ;

TAREFA propose l'utilisation du modèle ontologique pour une unification du vocabulaire du domaine. Même si un analyste expérimenté a plus de facilité pour comprendre/être compris par l'utilisateur, nous adoptons une solution plus générique et non attaché aux personnes.

TAREFA propose l'utilisation du modèle de processus émergent pour les processus de négociation et de résolution des conflits à cause des différentes visions (voir chapitre IV)

En plus de l'amélioration de la communication, TAREFA utilise l'information contextuelle disponible comme un complément aux besoins demandés explicitement par l'utilisateur. En effet, il y a une différence subtile mais très importante entre ce dont les utilisateurs ont besoin et ce qu'ils disent souhaiter. Donc, nous pensons qu'il faut faire une distinction entre les besoins **explicites** de l'utilisateur, demandés par l'utilisateur, et ses besoins **implicites**, qui sont nécessaires pour l'utilisabilité en considérant l'utilisateur, l'environnement et le système informatique comme un ensemble. En résumé, il faut utiliser de manière intégrée:

- les **besoins demandés** explicitement par l'utilisateur ("user requirements" et dans cette thèse BUEs),

- les **besoins implicites** de l'utilisateur ("user needs" et dans cette thèse BUIs), recueillies en général via l'analyse de la tâche [Long 94], et

- les **besoins** imposés par les règles **de l'organisation (**BUOs**),** identifiés en général par l'analyse des procédures de travail formel.

19.1Détermination des besoins de l'utilisateur dans TAREFA

L'approche proposée par TAREFA pour la détermination des besoins de l'utilisateur est résumée comme suit. Initialement, une liste informelle des BUEs est formulée, en associant un numéro pour identifier les besoin et en indiquant l'origine (en général la personne

responsable) des besoins. Ensuite, ces BUES sont associés aux tâches et activités courantes décrites respectivement par les modèles de tâches et le modèle de contexte. L'objectif est de vérifier la correspondance avec l'objectif général du système. Finalement, pour chaque tâche courante modélisée, il s'agit :

a) d'identifier les problèmes ou dysfonctionnements qui seront à l'origine des besoins implicites (BUI) et

b) de vérifier les contraintes/règles imposées par l'organisation (BUO).

Ces activités sont appliquées ci-dessous à l'étude de cas ATM.

19.2Formulation de la Liste Informelle des Besoins (BUEs)

D'abord il faut identifier les acteurs qui sont à l'origine des besoins demandés. Dans cet exemple, les utilisateurs (les clients de la banque qui vont utiliser l'ATM) : sont représentés en hypothèse par un échantillon des utilisateurs finaux interviewés appelés ici acteur #1 et acteur #2.

Besoins demandés par les utilisateurs finaux :

- (BUE #1) utilisation pour les opérations plus fréquentes, comme retrait, solde, relevé d'opérations ; Origine : acteur #1
- (BUE #2) possibilité de choisir le mot-de-passe ; Origine : acteur #2
- (BUE #3) éviter l'excès d'informations à taper ; Origine : acteur #1

19.3Associer les BUEs aux activités organisationnelles courantes

Le but de cette activité est d'identifier les activités organisationnelles qui seront considérées comme candidates pour les modifications afin de satisfaire les besoins des utilisateurs.

Bien sûr, l'association n'est pas si évidente. En fait, comme nous le montrons dans cet extrait de l'exemple ATM dans la figure 5.12, les besoins préliminaires des utilisateurs sont souvent d'ordre général.

37)

BUEs	Activités Organisationnelles Associées	
BUE #1	A1. Retrait d'Argent A2. Solde A3. Relevé des Opérations	A4 Dépôt A5 Virement
BUE #2	A8. Changement de Mot de Passe A9. Fourniture des Cartes	
BUE#3	A1. Retrait d'Argent A2. Solde A3. Relevé des Opérations	A4 Dépôt A5 Virement

Figure 5.12 - Association des BUE aux Activités Organisationnelles

19.4Pour chaque activité organisationnelle, identifier les problèmes non-mentionnés (BUI) et vérifier les contraintes/règles (BUO)

Le but est d'associer aux activités organisationnelles:

1. les problèmes détectés par l'intermédiaire de l'analyse de la tâche, qui forment les besoins implicites de l'utilisateur (BUIs);

2. les besoins imposés par les règles de l'organisation (BUOs), identifiés par l'intermédiaire de l'analyse des procédures de travail formel lors de la modélisation du contexte organisationnel.

Une spécification d'un système interactif doit permettre la vérification de la viabilité et la validité des besoins explicites en considérant ce qui est connu sur l'exécution des activités organisationnelles et les contraintes imposées par le contexte d'utilisation. Cela permet de connaître ce que l'utilisateur peut ou ne peut pas faire et dans quelles circonstances.

C'est pour cela que nous recueillons de manière intégrée aux activités, les besoins demandés explicitement par l'utilisateur ("user requirements"), les besoins implicites de l'utilisateur (BUIs), et les besoins imposés par les règles de l'organisation (BUOs).

Un extrait des besoins implicites et des besoins imposés par l'organisation pour les activités A1, A3, A4 et A5 de l'exemple ATM sont présentés dans le figure 5.13.

38)

Activités Organisationnelles	**Besoins Implicites (BUIs)**	**Besoins Imposés par l'Organisation (BUOs)**
A1. Retrait d'Argent	BUI#1	BUO #1, BUO #2, BUO #3, BUO #5, BUO #6, BUO #7, BUO #8
A3. Demande de Relevé des Opérations	BUI#2	BUO #1, BUO #2, BUO #3, BUO #4, BUO #5, BUO #6, BUO #7, BUO #8
A4 Dépôt	BUI#3	BUO #1, BUO #2, BUO #3, BUO #4, BUO #5, BUO #6, BUO #7, BUO #8, BUO#9
A5 Virement	BUI#3	BUO #1, BUO #2, BUO #3, BUO #4, BUO #5, BUO #6, BUO #7, BUO #8, BUO#9

Figure 5.13 - Extrait des besoins implicites (BUIs) et des besoins imposés par l'organisation (BUOs) de l'exemple ATM

Les codes des BUIs du figure 5.13 ont la signification suivante :

- (BUI#1) Consultation du solde avant de décider de la valeur à retirer
- (BUI#2) Définition flexible de la période du relevé
- (BUI#3) Affichage des informations du compte à créditer pour confirmation de l'opération.

Les codes des BUOs du figure 5.13 correspondent aux besoins suivants :

- (BUO #1) Une opération (soit dans l'ATM, soit dans l'agence) doit être une transaction, dans le sens 'tout ou rien', c'est à dire, ou une transaction est complètement finie ou l'état du système (et des comptes) doit rester inaltéré ; Origine : manager #1
- (BUO #2) l'utilisation de la carte est indispensable (la concurrence oblige) ; Origine : manager #2 ;
- (BUO #3) éviter l'excès d'information à taper ; Origine : manager #1

- (BUO #4) le fonctionnement distingué pour les clients de la banque (fonctions additionnelles) ; Origine : manager #1

- les procédures de sécurité :

    - (BUO #5) mot-de-passe pour chaque carte ; Origine : manager #2

    - (BUO #6) blocage de la carte en cas de essais de mot-de-passe incorrect ; Origine : manager #2

    - (BUO #7) enregistrement des transactions initiées (finies ou pas) ; Origine : manager #2

    - (BUO #8) confirmation de toutes les opérations ; Origine : manager #1

    - (BUO#9) l'information privée d'un client ne doit pas être visible aux autres clients ; Origine : manager #2

Evidemment, il y a des besoins en conflit comme par exemple les besoins BUI#3 et BUO#9 pour l'activité de 'Virement' (A5). Ces conflits ne doivent pas être résolus maintenant mais lors de la définition négociée de leur priorité après l'expérimentation des cas d'utilisation pour l'activité en question.

Chapitre VI - TAREFA : La macroactivité de Synthèse

20.Introduction

Après la macroactivité d'Analyse, présentée au chapitre précédent, ce chapitre présente spécifiquement la démarche et les modèles associés à la deuxième macroactivité de TAREFA : la Synthèse.

Ici, le mot 'synthèse' fait référence au processus de détermination et de modélisation des besoins du système et à la modélisation du *design rationale* de ce processus.

TAREFA propose 4 activités principales dans la macroactivité de Synthèse (voir figure 6.1) :

- La définition des besoins initiaux du système;
- La détermination des besoins du système;
- L'intégration des besoins du système;
- La modélisation du *design rationale*.

La définition des besoins initiaux du système (section 2) a pour but d'identifier les besoins initiaux du système à partir des besoins de l'utilisateur, qui peuvent ou ne peuvent pas être accomplis par un système informatique. Pour cela, deux activités complémentaires sont réalisées : la Re-ingénierie des tâches et l'allocation des fonctions.

La définition des besoins de l'utilisateur est fondamentale mais non suffisante pour la définition des besoins du système. En fait, les besoins de l'utilisateur ont tendance à être très généraux donc ils ne fournissent pas d'indications ou de détails sur le comportement interne ou externe du système. TAREFA reconnaît ce fait et propose l'utilisation d'un processus itératif et interactif de détermination des besoins du système (section 3). Ce processus a pour but de découvrir ces besoins grâce à leur confrontation aux différentes situations exprimées par l'intermédiaire de cas d'utilisation. La détermination des besoins du système est donc composée par les sous activités de construction des cas d'utilisation, d'expérimentation des cas d'utilisation, de modification des cas d'utilisation et d'intégration des cas d'utilisation.

L'intégration des besoins du système (section 4) a pour but la modélisation des besoins du système dans une forme compatible avec le modèle des besoins de la méthode Objectory. Ainsi, les besoins déterminés avec TAREFA peuvent être utilisés par les autres phases d'Objectory.

La modélisation du design rationale (section 5) a pour but à la fois de documenter et de guider le processus de détermination des besoins, en rendant explicite le raisonnement associé aux décisions de ce processus.

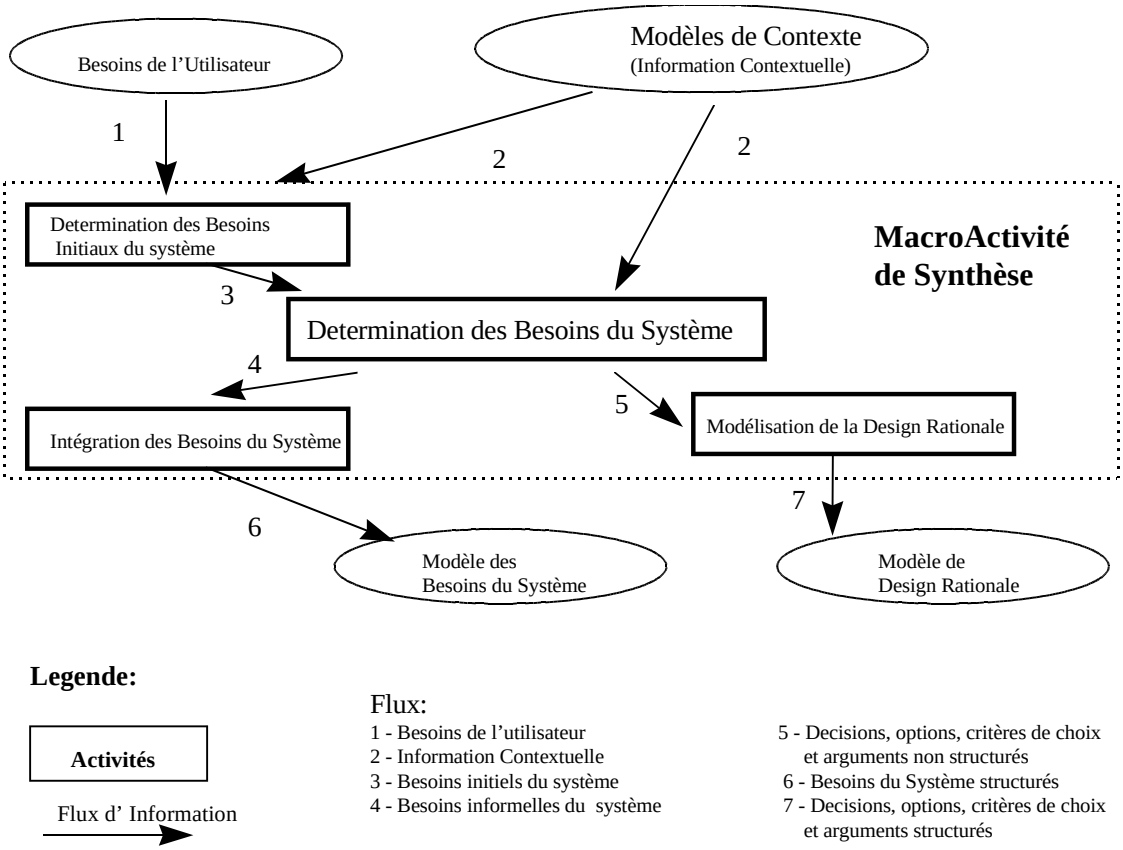


Figure 6.1 - La macroactivité de synthèse

Finalement, à la section 6, nous montrons comme les résultats de TAREFA peuvent être utilisés pour la spécification conceptuelle du système, en particulier en utilisant la méthode Objectory et la méthode DIANE+.

21.Définition des Besoins Initiaux du Système

Après la détermination des besoins de l'utilisateur dans la macroactivité d'Analyse, TAREFA propose une Re-Ingénièrie des tâches pour tous les tâches auxquelles sont associés des besoins demandés explicitement par les utilisateurs (les BUEs) ou des besoins implicites recueillis via l'analyse de la tâche (les BUIs).

La Re-ingénierie des tâches est l'application de techniques de Re-ingénierie aux modèles de tâches. Pour les tâches que l'on veut modifier, il faut les réorganiser pour éliminer les redondances et les inefficacités, avant de proposer une solution informatique. Cette re-ingénierie est basée sur l'information contextuelle recueillie, notamment le modèle de contexte organisationnel TOCO. Après la re-ingénierie, certains problèmes seront résolus, mais pas tous.

Ensuite, pour toutes les tâches dans lesquelles des problèmes persistent, une Allocation des Fonctions est réalisée, pour déterminer quelles tâches seront manuelles, automatiques ou interactives.

Après la réingenierie des tâches et l'allocation des fonctions, les besoins initiaux du système sont déterminés. Ils sont appelés 'initiaux' parce qu'ils ne représentent que le point de vue de l'utilisateur et doivent être complétés dans l'étape suivante par les besoins d'ingénierie, qui correspondent aux caractéristiques internes du système.

39)

21.1Re-ingénierie des Tâches

Parfois la solution à un problème est le résultat d'une modification organisationnelle ou d'une attitude de travail plutôt que d'une innovation technologique. Ce constat est à l'origine des techniques de Re-ingénierie. L'adoption de techniques de Re-ingénierie pousse une entreprise dans la voie d'une révision radicale de son organisation interne comme de ses rapports avec son environnement.

Les techniques de Re-ingénierie ont une double application :

1. remodeler le fonctionnement d'une situation de travail d'une entreprise pour en améliorer les performances, connue comme Ré-ingenierie des processus de l'entreprise (business process reengineering) et dont le but est, de manière simplifiée, de 'modifier le processus', c'est à dire, de le reconfigurer :
   - en remettant en cause certaines fausses contraintes;
   - en diversifiant les variantes et en répartissant le travail différemment entre les acteurs, grâce à l'emploi prévu de technologies nouvelles.
2. mener à bien un projet de réorganisation, dont le but est, de manière simplifié, de 'changer de processus'; ce changement est encore plus radical puisque il ne s'agit plus de changer uniquement de solution mais aussi de problème.

Nous retenons le premier type d'application de techniques pour l'appliquer sur les tâches et activités décrites par les modèles de l'information contextuelle. La Re-ingénierie des tâches est donc l'application de techniques de Re-ingénierie aux modèles de tâches. Le but est de remodeler un modèle de tâche minimale à partir des règles du contexte. Pour les tâches que l'on veut modifier, il faut les réorganiser pour éliminer les redondances et les inefficacités. Cela implique les activités suivantes :

- séparer les détails d'implémentation de l'essentiel;
- identifier comment les activités/tâches peuvent devenir plus faciles, en supprimant les contraintes non-nécessaires (p.ex. séquences imposées ou habituelles);
- identifier les informations nécessaires pour chaque tâche et quand elles sont nécessaires;
- vérifier si les problèmes sont résolus.

Un exemple de suppression des contraintes non-necessaires est la re-ingénierie des tâches du modèle de tâches minimales 'Retrait d'Argent', où les buts ne sont pas nécessairement séquentiels.. Dans la figure 6.2(b), l'identification du client doit être faite avant la prise de l'argent (dans la séquence TPAR arrive avant le but '*Prendre l'argent*') mais non nécessairement avant la séquence qui groupe la demande de la transaction et le choix de la valeur à retirer (TSEQ peut être réalisé en parallèle avec '*S'identifier*'). En effet, la règle U2 du modèle de contexte organisationnel TOCO (à savoir, 'SI le client n'est pas identifié ALORS il ne peut pas prendre l'argent' - voir figure 5.10 dans le chapitre 5) n'impose pas la séquence montrée dans la figure 6.2(a), qui est probablement la conséquence de l'utilisation fréquente de cette séquence. La figure 6.2 montre le modèle de tâche minimale respectivement avant (a) et après (b) la Re-ingénierie des tâches en utilisant la notation MAD

40)

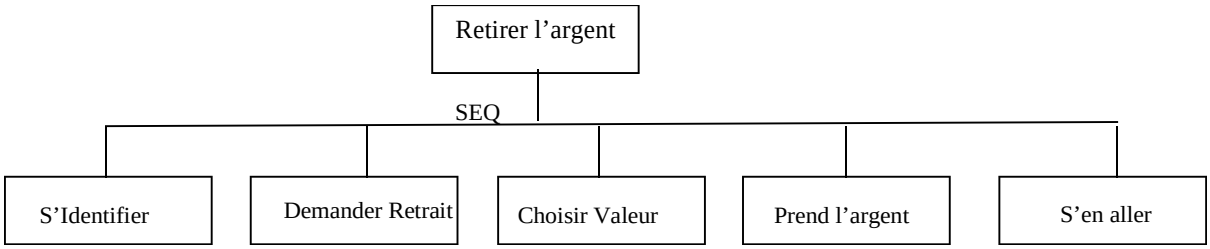
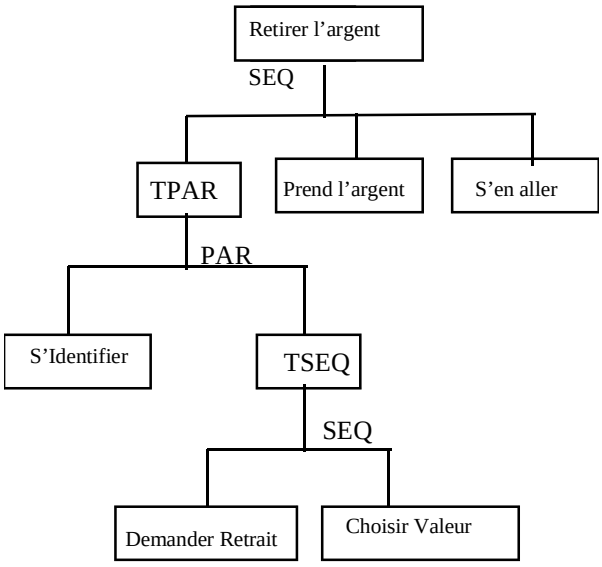


Figure 6.2 - (a) Modèle de Tache Minimale 'Retirer l'argent avant la Re-ingénierie de tâches

41)



42)

Figure 6.2 - (b) Modèle de Tâche Minimale 'Retirer l'argent' après la Re-ingénierie des tâches

Par ailleurs, décider quelles activités seront reformulées est au dessus de l'envergure de ce travail. Pour cela il vaut mieux prendre en compte le cadre conceptuel de la Re-ingénierie des systèmes (*Business Process Reengineering*). Par exemple, la décision de permettre à l'utilisateur de choisir son mot de passe (BUE#2) est une décision administrative de changement de l'activité d'ouverture de compte.

21.2Allocation des Fonctions

Après l'ingénierie des tâches, il faut réaliser l'allocation des tâches pour définir :

- les tâches manuelles
- les tâches automatiques
- les tâches interactives

Le cadre conceptuel de la Théorie de l'Activité [Nardi 1996] aide à comprendre comment les buts des tâches de chaque utilisateur interagissent entre eux dans un domaine de l'application pour constituer une **activité** ou un processus. Cela permet de situer ces trois types de tâche dans le contexte organisationnel, pour permettre de savoir quelles activités de l'organisation impliquent des tâches manuelles, automatiques ou interactives et aussi pour évaluer l'impact du changement d'un type.

L'allocation des fonctions entre des humains et les systèmes se situe dans un intervalle (*spectrum*) de types possibles d'allocation qui va des tâches manuelles à des

tâches automatiques (ou totalement informatisés), conformément dans la figure 6.3. Les tâches manuelles sont des tâches exécutées par les utilisateurs avec peu ou aucun soutien du système informatique et les tâches totalement informatisées sont des tâches exécutées par le système avec peu ou aucune intervention humaine [Macaulay 95]. Dans cet intervalle, il y a des tâches interactives, dont l'exécution est partagé entre l'utilisateur et le système.

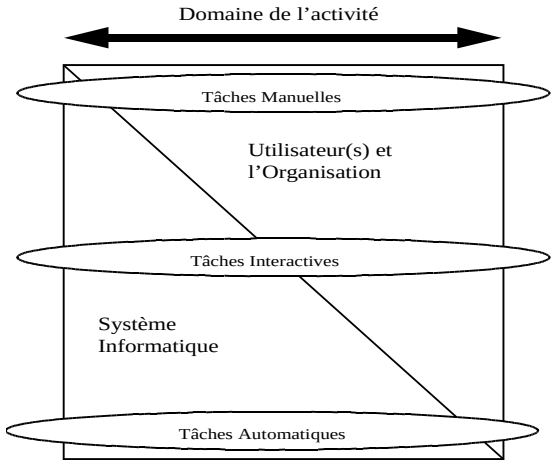


Figure 6.3 - L'Intervalle de l'Allocation des fonctions

En fait, les différentes propositions de l'automatisation des tâches sont classées en trois catégories générales [Wickens 92 p. 531] :

1. Exécuter des fonctions que l'opérateur humain ne peut pas exécuter à cause de ses limitations inhérentes (p.ex. dans un système de contrôle de fusée, l'utilisateur ne peut pas répondre suffisamment rapidement);

2. Exécuter des fonctions que l'opérateur humain peut réaliser mais sans précision ou avec une surcharge d'effort;

3. Augmenter ou assister la performance de l'opérateur dans des domaines où les humains montrent des limitations. Cette catégorie est similaire à la précédente mais ici l'automatisation n'a pas pour but de remplacer intégralement toute la tâche mais surtout d'aider les tâches (en général périphériques) nécessaires pour la réalisation de la tâche principale.

Les deux premiers cas sont associés aux tâches automatiques et le troisième, aux tâches interactives. Cependant, le degré d'automatisation n'est pas le seul aspect à considérer en allocation de fonction. Il faut aussi déterminer qui controle la tâche (l'utilisateur ou le système) et le type de soutien du système (actif ou passif).

Le contrôle de la tâche est immédiat à déterminer pour les tâches manuelles et automatiques, c'est respectivement l'utilisateur ou le système qui le possède. Le problème est de déterminer le contrôle des tâches interactives. Une discussion interessante sur ce sujet peut être trouvée dans la littérature des UIMS [Hartson & Hix 89].

Le type de soutien du système peut être *passif* ou *actif*. Le soutien passif implique le système comme un ensemble d'entités à manipuler et de tâches à réaliser. Par exemple, les logiciels de '*spreadsheets*' sont des applications commerciales utilisées pour soutenir les tâches financières et comptables. Leurs éléments sont des valeurs numériques et monétaires, dates et tableaux et les tâches à réaliser sont des opérations

pour manipuler ces entités comme des fonctions financières et statistiques. Le *soutien actif* implique, en plus des entités et des tâches, une assistance du système à la réalisation des tâches, par l'intermédiaire de suggestions, critiques ou guidage [Fischer 90].

Les décisions de l'allocation sont fondamentales à ce stade de l'analyse parce que l'allocation détermine quelles activités sont affectées par l'introduction du système et quel sera le rôle du système comme soutien de chaque activité. Bien sûr, elle est aussi une aide importante pour évaluer les alternatives coût/bénéfices du système pour les utilisateurs et l'organisation.

L'approche d'allocation de fonction retenue suit une tradition en ergonomie : les dévelopeurs énumèrent les différentes alternatives de soutien informatique pour chaque activité et les utilisateurs peuvent évaluer les plus adéquates pour leur travail et les plus réalisables avec les ressources mises à disposition. L'énumération est faite en présentant, pour chaque alternative d'allocation, les fonctions de l'utilisateur, les fonctions du système, le coût et l'avantage, conformément dans la figure 6.4.

Alternative	**Fonction de l'utilisateur**	**Fonction du Système**	**Coût**	**Avantage**
Nom de l'alternative (en explicitant ses caractéristiques principales)	La liste des fonctions de l'utilisateur pour cette alternative	La liste des Fonctions du système pour cette alternative	Les difficultés (techniques, politiques, économiques, etc) pour son adoption et son utilisation quotidienne	Les avantages de son adoption et utilisation quotidienne

Figure 6.4 - Explication des Informations de la Liste des Alternatives d'Allocation de Fonctions

Dans l'exemple de l'ATM décrit, il n'y pas de tâches manuelles significatives. Les tâches de l'ATM sont soit interactives soit automatiques. En plus, ce sont des tâches très particulières parce que :

- chaque tâche est une transaction de courte durée, sans problème de surcharge physique ou cognitive;

- chaque tâche est très structurée, composée d'une petite série de sous-tâches pour arriver au but principal;

- l'ATM est un système créé pour être indépendant et autosuffisant; il n'y a pas une liaison avec d'autres systèmes/activités bancaires en permanence, cette liaison n'est établie que de façon périodique afin d'actualiser les informations; en conséquence, il n'y pas d'entrelacement avec d'autres activités ni besoin d'interruptions planifiés ('*closure events*' dans [Wickens 92]) pour couper une longue séquence en intervalles réguliers.

Un exemple d'allocation de fonction pour l'émission des reçus des transactions de l'ATM est donnée dans la figure 6.5, selon la perspective de la banque.

Alternative	Fonction de l'utilisateur	Fonction du Système	Coût	Avantage
Emission du reçu pour toutes les transactions sans demander l'avis de l'utilisateur	Pas d'intervention de l'utilisateur	Imprimer les reçus pour toutes les transactions	Prix de papier	La durée d'utilisation est minimisé, en favorisant le nombre de transactions dans la même unité de temps
Emission du reçu selon choix de l'utilisateur	Décider s'il veut ou non le reçu et répondre la question posée	Demander à l'utilisateur s'il veut ou pas le reçu de ses transactions; Selon la réponse, Imprimer les reçus souhaités	L'utilisation est prolongée à cause du dialogue plus sophistiqué;	Utilisation réduite de papier L'impression de l'utilisateur d'avoir un contrôle sur la machine est renforcé

Figure 6.5 - Un exemple d'Allocation de Fonctions pour l'émission des reçus des transactions de l'ATM

21.3Les Besoins Initiaux du système

C'est la liste des besoins initiaux dérivés directement des Besoins de l'Utilisateur (BUs) et après les phases de Re-ingénierie des tâches et d'allocation des fonctions.. Ces besoins initiaux répresentent le point de vue de l'utilisateur sur les besoins du système.

Ici, les expressions 'orientées utilisateur' des BUs sont traduites par l'analyste en expressions 'orientées système' des BS. La figure 6.6 illustre cette traduction pour des expressions extraites de l'exemple de l'ATM.

43)

Besoins de l'utilisateur	Besoins Initiales du Système
BUO#1 :ATM_transationnelle	BS#1 : ATM_Initialisable
BUO#2 : Utilisation-de-Carte	BS#2 :Système doit supporter l'utilisation de la carte magnétique bancaire pour toutes les opérations
BUO#3 :Peu-Taper	BS#3 : Le nombre de touches et la trajectoire de l'interaction de chaque transaction doivent être minimisées
BUO#4 : Notre-Client-Fait-Plus	BS#4 : Client de la banque propriétaire de l'ATM a des fonctions additionnelles disponibles
BUO#5 : Mot-de-passe-pour-cartes; :Si le compte est conjoint, chaque personne a son mot-de-passe	BS#5 : Chaque carte bancaire (ou chaque personne) a une mot-de-passe
BUO#6 :Identification garantie du client	BS#6 :Propose 3 essais de saisie du code
BUO#7 :Enregistrer Opérations	BS#7 : Enregistrer Opérations
BUO#8 : Confirmer Opérations	BS#8 : Confirmer Opérations
BUO#9 : Usage Privée des d'Informations	BS#9 : Sécurité d'Informations
BUE#1 Opérations-Fréquentes	BS#9 : Opérations-Fréquentes
BUE#2 : Choix de Mot de Passe	BS#11 : Proposer saisie de mot de passe
BUE#3 : Peu d'Information à taper	BS#3 (voir description ci-dessus)
BUI#1 : Consultation du solde avant de décider de la valeur à retirer	BS#10 : Pouvoir réaliser une transaction de consultation entrelacée avec une autre transaction
BUI#2 :Définition flexible du période du relevé	BS#11 : Permettre la définition flexible de la période du relevé
BUI#3 : Affichage des informations du compte à créditer pour confirmation de l'opération	BS#12 : Afficher des informations du compte à créditer pour confirmation de l'opération

figure 6.6 - Traduction d'expressions des besoins de l'utilisateur en besoins initiaux du système pour l'exemple ATM.

22.Détermination des Besoins du Système

Le but de cette activité est de spécifier les besoins d'ingénierie du système, c'est à dire, de spécifier les caractéristiques que le système doit posséder pour satisfaire les

besoins des utilisateurs et qui doivent être considérés par l'équipe de conception. En fait, les besoins initiaux doivent être exprimés en termes des caractéristiques internes du système de façon à fournir le comportement souhaité par l'utilisateur. Ces caractéristiques internes correspondent aux trois types d'information sur le système mentionnés comme composants du modèle des besoins du système interactif dans le chapitre 4 : la structure des informations manipulées, sa fonctionnalité et ses interactions avec les utilisateurs dans une situation d'utilisation. Déterminer ces caractéristiques n'est pas trivial et TAREFA adopte les cas d'utilisation comme le concept fondamental pour ce processus de détermination (section 3.1).

TAREFA utilise un processus itératif et interactif de détermination des besoins du système. Ce processus a pour but la découverte des besoins en confrontation à différentes situations exprimées par l'intermédiaire de cas d'utilisation. Conformément dans la figure 6.7, la détermination des besoins du système est composée par les sous activités de construction des cas d'utilisation (section 3.2), d'expérimentation des cas d'utilisation (section 3.3), de modification des cas d'utilisation (section 3.4) et d'intégration des cas d'utilisation (section 3.5).

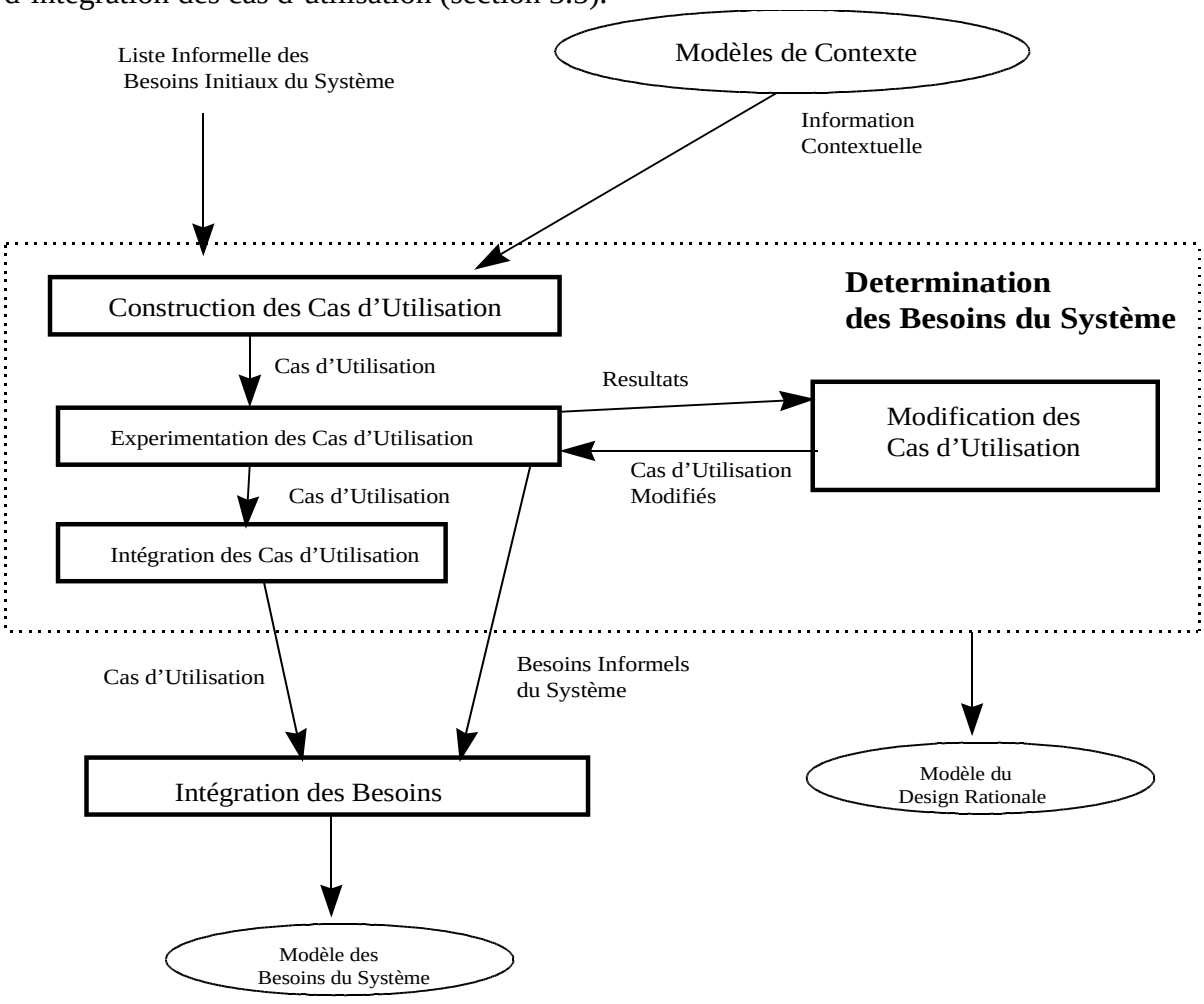


Figure 6.7 - L'activité de détermination des besoins

22.1Modèle de Cas d'Utilisation de TAREFA

Un **cas d'utilisation** modélise une **situation d'usage** où un (ou plus) service(s) du

système informatique sont utilisés par un (ou plusieurs) utilisateur(s) pour arriver au(x) but(s). Un cas d'utilisation peut modéliser les cas normaux (le but est atteint) mais aussi les cas singuliers, dont le but n'est pas atteint à cause de singularités.

Un cas d'utilisation, comme toute narration, peut être divisé en parties appelées **épisodes**. Le même épisode peut apparaître dans plusieurs cas d'utilisations. Un épisode consiste en **actions** exécutées sous des **conditions** par les acteurs pour arriver aux **résultats**. Les résultats sont des états qui sont normalement les sous-buts de la décomposition du but principal du cas d'utilisation, mais qui peuvent aussi être d'autres états atteints à cause de l'occurrence d'une singularité dans la réalisation d'un cas d'utilisation.

Les actions sont décrites par la paire (initiateur, nom de l'action), où **nom de l'action** peut représenter une action de type composée ou de type primitive. Une action composée est formée par un ordonnancement (temporel ou de dépendance logique) d'actions primitives. Une action primitive est l'application directe d'une opération à un objet et elle n'est pas décomposable.

L'**initiateur** explicite qui a l'initiative et le contrôle d'exécution de l'action. Les initiateurs typiques des systèmes interactifs sont :

- l'utilisateur, qui envoie des stimulus au système (c'est à dire, des messages de l'utilisateur au système);

- le système, qui répond aux stimulus de l'utilisateur (c'est à dire, des messages du système à l'utilisateur) et de l'environnement; ou qui exécute des opérations internes sans communication avec l'utilisateur;

- l'environnement, qui est le déclencheur typique des événements qui ne sont pas des stimulus de l'utilisateur (p.ex. les événements temporels comme le début d'un mois).

Ce type de définition d'initiateur est similaire à la définition de déclencheur proposé par [Barthet 88], où une action (opération dans les termes de DIANE) peut être déclenché par l'utilisateur ou par l'ordinateur.

Les **conditions** sont des prédicats sur les états du système qui doivent être considérées avant (appelées dans ce cas **pré-conditions**) ou après (appelées dans ce cas **post-conditions**) l'exécution des actions.

Une action primitive dont l'initiateur est le système peut être encore raffinée pour décrire avec plus de détail son comportement. Dans ce cas ce comportement est décrit en termes d'opérations sur des objets. Donc une **opération** sur un **objet** est associée à un cycle d'état de l'objet et à l'information nécessaire à son exécution (entrées) et aux informations résultantes (sorties).

Les actions de l'utilisateur porraient être raffinées à niveau manipulation physique mais ceci n'est pas considéré dans TAREFA.

Ces raffinements successifs montrent qu'un cas d'utilisation peut être décrit à différents niveaux d'abstraction. Les niveaux proposés pour TAREFA et leurs caractéristiques sont présentées dans les sections suivantes.

22.1.1Les quatre Niveaux de Cas d'Utilisation de TAREFA

Pour TAREFA, un cas d'utilisation est une spécification à la fois de la structure, de la fonctionnalité et du comportement d'une situation d'usage. Comme toute spécification, elle a deux aspects : l'aspect technique, parce qu'elle sert de point de départ pour la conception et la réalisation du système, et l'aspect social, parce qu'elle

sert à la communication entre les acteurs, en particulier l'analyste et l'utilisateur.

Ces deux aspects exigent qu'un cas d'utilisation puisse être décrit à différents niveaux d'abstraction. Notre approche considère quatre niveaux, associés à 4 types de cas d'utilisations interconnectés **: essentiel, téléologique, opérationnel** et **concret**, voir figure 6.8. Chaque niveau est adapté à un but déterminé et concerne un type spécifique de connaissance et de représentation, donc une décomposition différente du cas d'utilisation. En particulier, le niveau **essentiel** défini les cas d'utilisation essentiels (et sa division en épisodes) qui doivent être accomplis afin de satisfaire certains buts dans un contexte organisationnel des activités de l'utilisateur; le niveau **téléologique** définie les cas d'utilisation singuliers, qui décrivent les différentes trajectoires des épisodes du système, avec toute la complexité des différentes variations et exceptions à considérer et qui peuvent éloigner l'interaction de la trajectoire 'normal'; le niveau **opérationnel** présente l'aspect comportemental de chaque épisode en termes d'objets et d'actions; finalement le niveau **concret** représente des occurrences d'exécution particulière d'un cas d'utilisation.

44)

45)

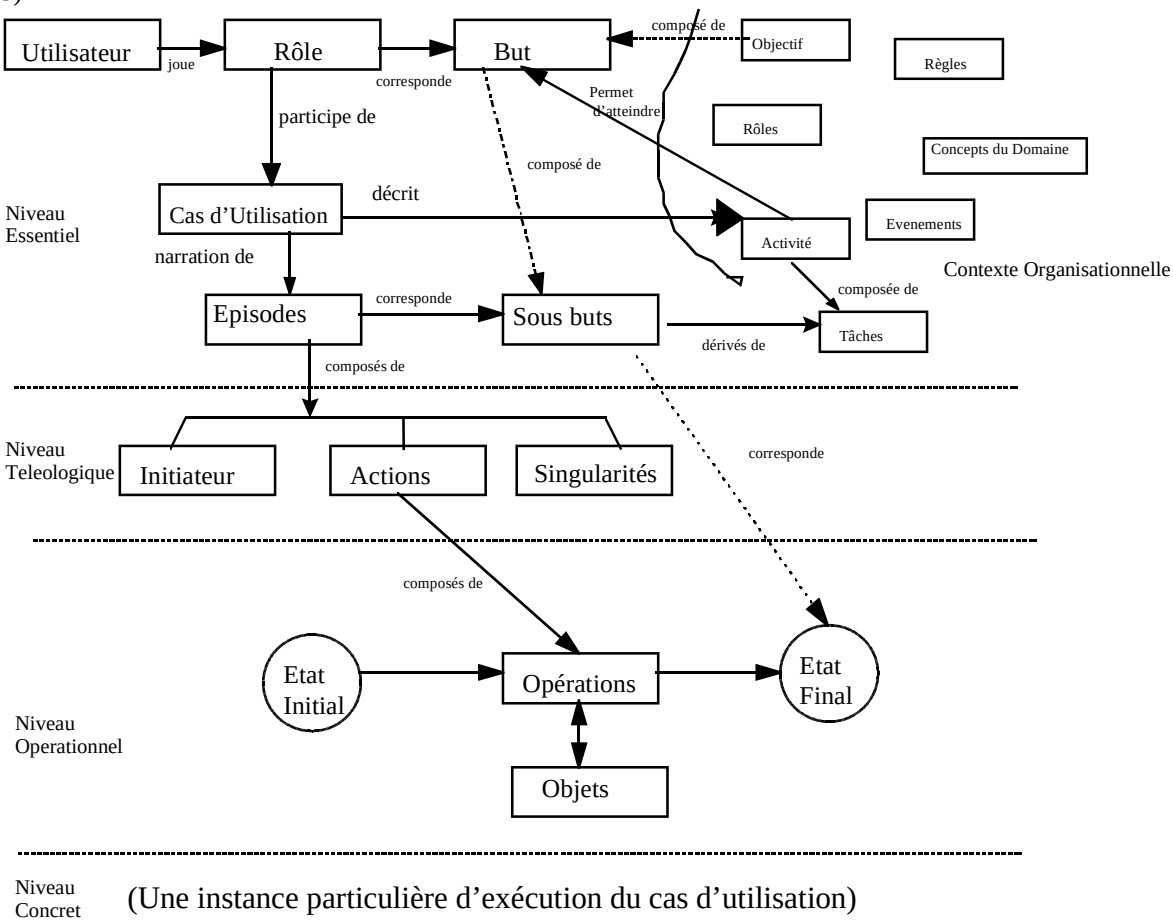


Figure 6.8 - Les quatre niveaux de description d'un cas d'utilisation

Il n'est pas nécessaire de faire la description complète d'un cas d'utilisation selon les quatre niveaux. Par exemple, on n'a pas besoin de faire un raffinement pas a pas d'un cas d'utilisation essentiel jusqu'au concret : le cas d'utilisation opérationnel est facultatif et son utilisation n'est intéressante que pour poursuivre la conception orientée objet. Il est possible de passer des cas d'utilisations Singuliers aux cas d'utilisations concrets par génération assistée (semi-automatique) ou prototypage.

22.1.2Format de Description des Cas d'Utilisation

Il y a plusieurs façons de représenter un cas d'utilisation. Par exemple, [Jacobson 92] utilise une notation textuelle informelle, [Rubin & Goldberg 92] utilisent une notation tabulaire, [Hsia 94] montre qu'un cas d'utilisation peut être représenté par un langage régulier, et [Glinz 95] adopte une représentation par 'statecharts'.

A chaque niveau d'abstraction, TAREFA définit une notation specifique :

- au niveau essentiel, est adoptée une narration structurée en deux colonnes, raisonnablement adéquate pour la description des aspects principaux des cas d'utilisation, car permettant en plus une communication facile avec l'utilisateur;

- au niveau singulier, est adoptée une notation qui suit un 'schéma grammatical', raisonnablement adéquate pour la description des singularités des cas d'utilisation, car permettant aussi une communication facile avec l'utilisateur;

- au niveau opérationnel, est adoptée une notation tabulaire, qui permet de structurer les informations détaillées préparées pour la suite de la conception;

- au niveau concret, est adoptée la combinaison d'une description textuelle informelle et d'un prototype 'bas-niveau', adéquats pour exemplifier les aspects concrets des cas d'utilisation pour l'utilisateur.

22.1.3Le Niveau Essentiel

Un **Cas d'utilisation Essentiel** est dit 'essentiel' parce qu'il capture l'essence du cas de l'utilisation dans une structure idéale sans considérer les détails dépendants de la technologie (appelée 'supposition de technologie parfaite' selon [McMenamin & Palmer 84]) et est basé sur les intentions de l'utilisateur et les services fondamentaux du système. La structure est ***idéale*** parce que les cas d'utilisation essentiels modélisent les cas normaux où le but est atteint, c'est à dire, il se concentre sur la trajectoire normale d'une interaction dont le but de la tâche est accompli. La supposition de ***technologie parfaite*** permet d'oublier des limitations non nécessaires et prématurément restrictives. Par conséquence, le modèle de cas d'utilisation est plus flexible, il permet la recherche de diverses options de réalisation et s'adapte plus facilement aux changements de technologie. L'identification des ***intentions de l'utilisateur*** permet de se concentrer sur les aspects essentiels de ses activités et comment les prendre en compte de façon directe et efficace avec un système informatique. Ce niveau est donc adéquat pour associer la situation d'usage à un contexte de travail : les cas d'utilisation essentiels mettent en valeur les activités devant être accomplies afin de satisfaire certains buts dans un contexte organisationnel. En plus, comme un cas d'utilisation essentiel est adapté au ***niveau d'abstraction de l'utilisateur***, il peut servir comme un moyen de communication efficace entre analyste et utilisateur.

Identifier le but ou l'intention d'un cas d'utilisation est une question importante. Il y a déjà des travaux reconnaissant que nous ne pouvons comprendre un cas d'utilisation que lorsque nous connaissons son but [Kaindl 95b]. Par exemple, lorsque quelqu'un s'approche d'une ATM, on connaît son intention potentielle : réaliser un transaction bancaire. Cela parait évident parce que nous utilisons des ATMs quotidiennement. Cependant pour un système nouveau et encore inconnu, ***connaître le but*** est une information fondamentale pour une compréhension du cas d'utilisation.

Evidemment, on pourrait avoir différentes manières d'interagir avec l'utilisateur,

c’est à dire, différentes réalisations possibles du cas d’utilisation en fonction de la technique (styles, périphériques, etc) d’interaction utilisée. Le cas d’utilisation essentiel est un schéma qui permet de raisonner sur les aspects essentiels de l’interaction et donc sur les services du système. Ainsi un cas d’utilisation essentiel est un mécanisme schématique pour aider à réfléchir sur la manière dont le logiciel sera utilisé par les utilisateurs. Un exemple de cas d’utilisation essentiel est illustré dans la figure 6.9.

46)

Buts de l’Utilisateur	Services du système
	ATM prêt à être utilisé
S’Identifier	Accepter Identification
	Vérifier Identification
	Offre Services
Demander Retrait	Accepter Demande
	Demander Valeur
Choisir Valeur	Accepter Valeur
	Vérifier Soldes (ATM et Carte)
	Faire Sortir argent, carte et reçu
Prendre les choses	ATM prêt pour une autre opération
S’en aller	ATM prêt à être utilisé

Figure 6.9 - Exemple : Cas d’utilisation essentiel Retrait d’Argent de l’étude de cas ATM

Dans ce sens, un cas d’utilisation essentiel est une projection de l’utilisation future du système par le(s) utilisateur(s). Il représente les séquences de transactions qu’il a besoin de réaliser en dialogue avec le système : chaque séquence complète de transactions commencé par un utilisateur est définie comme un cas d’utilisation et l’ensemble de tous les cas par tous les utilisateurs correspond à l’utilisation complète imaginée pour le système, comme illustré par la figure 6.10 pour l’exemple ATM.

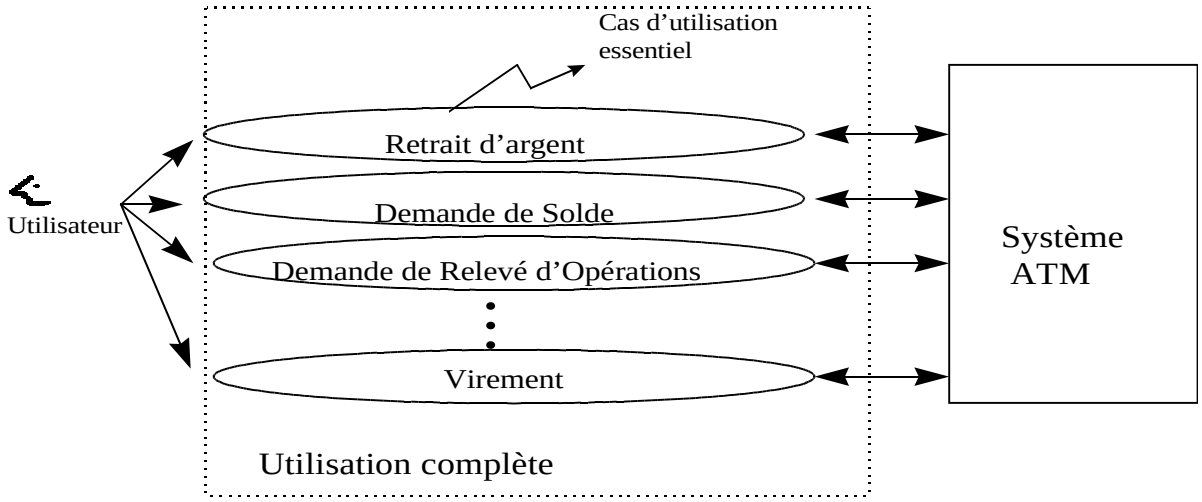


Figure 6.10 - Les Cas d’Utilisation Essentiels de l’ATM

22.1.4Le Niveau Téléologique

Le niveau **téléologique** exprime le rapport entre un réseau de buts/sous-buts et les différents épisodes des cas d’utilisation du système, avec toute la complexité des différentes variations à considérer. Dans ce niveau, sont définis les cas d’utilisation singuliers. Les **Cas d’utilisation Singuliers** décrivent non seulement la trajectoire 'normale' d’un cas essentiel mais aussi les ‘déviations’ qui peuvent éloigner l’interaction de la trajectoire ‘normale’. Ces déviations - y compris les anomalies, les exceptions, les

interruptions et les erreurs - sont appelés ***singularités*** dans [Jambon 96/, d'où vient l'expression 'cas d'utilisation singulier'. Evidemment, la notion de singularité englobe toutes les conditions qui 'singularisent' le cas d'utilisation normale et pas seulement les cas anormaux[12]. Cette idée de considérer ces conditions singulières comme composants incontournables de la modélisation est déjà présente dans les Blocs de Connaissance de G. Boy [Boy 89], un modèle de représentation des connaissances pour les systèmes hommes-machines multi-agents et qui est appliqué avec succès dans le domaine aérospatial (par exemple l'informatisation du processus de pilotage d'avion voir [Boy 95]).

Un cas d'utilisation singulier est associé à des situations spécifiques et singularités d'un cas d'utilisation essentiel. La concentration sur des points spécifiques aide à raffiner la compréhension sur les besoins du système futur. C'est une difficulté substantielle dans la conception de systèmes : bien supporter l'activité principale et à la fois considérer les cas singuliers. Malgré l'évidente importance de la trajectoire normale de l'activité principale, la considération des cas singuliers peut être cruciale pour l'utilisabilité du système.

22.1.4.1Le format de description des cas singuliers

Le format de 'schéma grammatical' est présenté dans la figure 6.11. La notation tabulaire est présentée plus tard avec la description des caractéristiques du niveau opérationnel. L'utilisation des prototypes 'bas niveau' est présentée plus tard avec la description des caractéristiques du niveau concret.

Un schéma grammatical n'est pas une grammaire générative, dans le sens connu de la Théorie des Langages, parce qu'il permet de laisser indéterminée la structure de certaines éléments - typiquement à travers une description informelle. Ce format a été choisi principalement parce que :

- les cas d'Utilisation sont des narrations, donc une description typiquement associé à un langage. Les grammaires sont des formalismes adéquats à la description des langages.

- cette idée a été déjà utilisée avec succès par d'autres auteurs : les 'schematas' de Colin Potts [Potts 95] et le 'langage naturel structuré' de [Kaindl 95b] en sont des exemples.

12 Une erreur , par inadvertance (comme les *slips*, *lapses* ou *mode errors* /Wickens 92/) ou par changement d'intention (comme les *mistakes* /Wickens 92/), doit être considéré comme un événement normal d'une interaction. C'est pour cela que nous utilisons plutôt le terme 'cas singulier' et non 'cas anormal'.

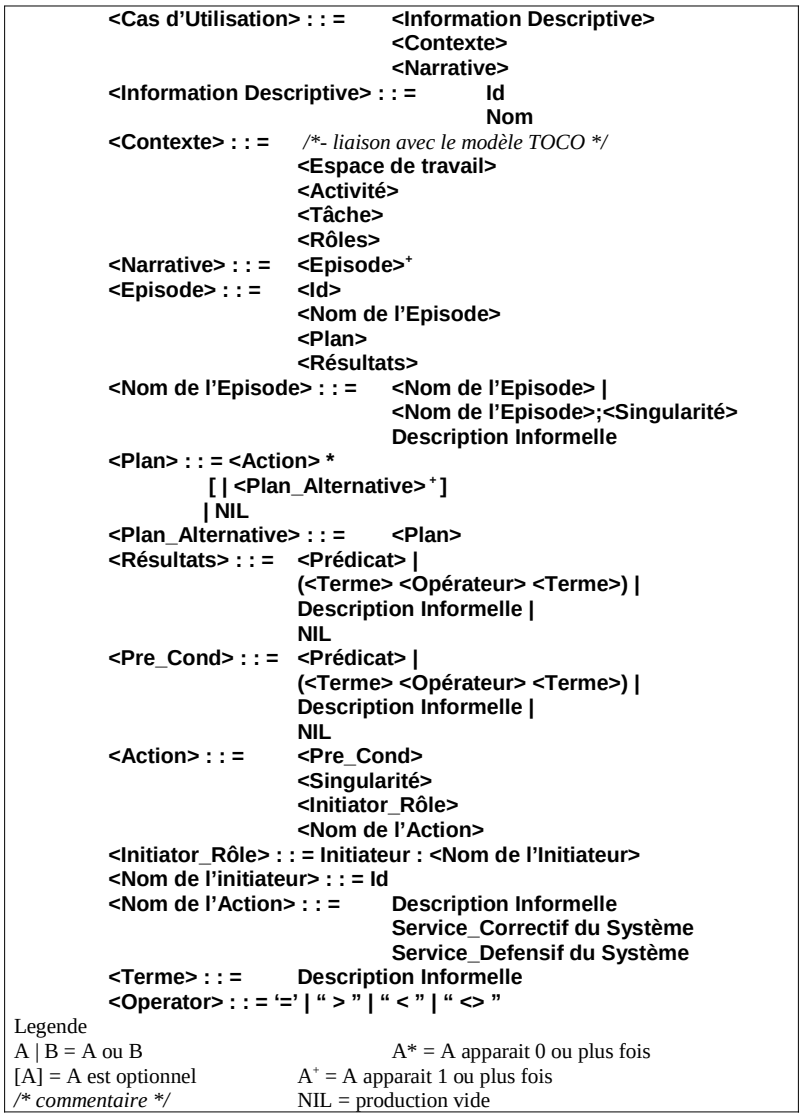


Figure 6.11 - Schéma grammatical pour la description textuel des cas d'utilisation

22.1.4.2Cas d'Utilisation Singuliers et Singularités

Un cas d'utilisation singulier modélise les cas normaux (où le but est atteint) mais aussi les cas d'exception, dont le but n'est pas atteint à cause de singularités. Par définition, une **singularité** est **la cause** d'un **problème**. Au niveau des besoins, un problème est une différence sémantique entre la situation actuelle et une situation désirée. Donc, il faut toujours associer une singularité à un problème et un problème à un but. Un exemple d'un épisode de cas d'utilisation singulier est montré dans la figure 6.12. C'est le premier épisode (Episode 1 - S'Identifier) du cas d'utilisation singulier qui modélise le cas normal (UC0) de la transaction 'Retrait d'argent' de l'ATM. Cette description suit le format grammatical décrit précédemment dans la figure 6.11.

Dans cet exemple, il est montré qu'un épisode décrit l'interaction entre l'utilisateur et l'ATM pour que le but de l'utilisateur (S'Identifier) soit accompli. Cette interaction est décrite par un plan, composé par des actions initiées par l'utilisateur ou par l'ATM (voir Initiator de chaque action). A chaque action, sont associées des conditions. Les **conditions** sont des prédicats sur les états du système ou des événements qui doivent être considérés avant (appelées dans ce cas preconditions) ou après (appelées dans ce

cas postconditions) l'exécution des actions de l'épisode. Dans le cas où deux actions ont la même pré-condition, les **deux** peuvent être réalisées simultanément si la pré-condition est validée.

47)

Cas d'utilisation Retrait d'Argent
UC0 - Sans singularité - Cas Normal
Episode1 : S'Identifier (Identify Self)
Plan :

Pre_Cond : Prêt à Utiliser (ATM)
Initiator : User
Action : Fournir Identification (Id) du Client
Post_Cond : Fourni (Id) ou Non_Fourni (Id)

Pre-Cond : Prêt à Utiliser (ATM)
Initiator : ATM
Action : Accepter (Id)
Post_Cond : Accepté (Id) ou Non_Accepté (Id)

Pre-Cond : Fourni (Id) AND Accepté (Id)
Initiator : ATM
Action : Verifier (Id)
Pos_Cond : Valid (Id) ou Invalide (Id)

Pre-Cond : Valide (Id)
Initiator : ATM
Action : Demander (Mot-de-Passe)
Post_Cond : Mot-de-Passe_Demandé(ATM)

Pre-Cond : Mot-de-Passe_Demandé(ATM)
Initiator : user
Action : Fournir (Mot de Passe)
Post-Cond : Fourni (Mot-de-Passe)

Pre-Cond : Mot-de-Passe_Demandé(ATM)
Initiator : ATM
Action : Accepter (Mot-de-Passe)
Post_Cond : Accepté (Mot-de-Passe)

Pre-Cond : Accepté (Mot-de-Passe)
Initiator : ATM
Action : Verifier (Mot-de-Passe)
Post_Cond : Valid (Mot-de-Passe) ou Invalid (Mot-de-Passe)

Resultats : Si Valid (Mot-de-Passe) l'utilisateur est un client valid

Figure 6.12 - Exemple de Cas d'utilisation singulier - L'Episode 1 du cas 'normal' de 'Retrait d'argent' de l'ATM

Identifier une singularité est une question très importante. Tout but peut être 'bloqué' par des conditions ou des événements du système ou de l'environnement. Envisager les singularités possibles oblige le concepteur (et aussi l'utilisateur) à réfléchir sur des solutions flexibles et robustes pour ces situations réelles (où l'utilisation se déroule d'une façon non anticipée par le concepteur) et non pour des situations idéalisées. A chaque singularité d'un cas d'utilisation singulier des **services défensifs** (pour essayer d'éviter ou de prévenir la singularité) et des **services correctifs** (pour essayer de corriger la singularité) sont proposés.

Un exemple de singularité est montré dans la figure 6.13.

Id de la Singularité	Action/ Episode	Service où la singularité arrive	Problème	Singularité (Cause du Problème)	Services (D)efensifs ou (C)orrectifs du Système
Singularité 4	S'Identifier (épisode 1)	Vérifier Mot-de-Passe	Invalide (Mot-de-Passe)	Mot-de-Passe fourni par l'utilisateur n'est pas la correcte	c) Le système permet 3 essais pour fournir le mot-de-passe correcte; a) Après les 3 essais, la carte est bloqué et gardé par l'ATM

Figure 6.13 - Exemple de Singularité pour le cas d'utilisation essentiel 'Retrait d'Argent'

22.1.4.2.1Délimitation aux singularités intéressantes

Un inconvénient est bien sûr le nombre de cas d'utilisations qui seraient nécessaires pour représenter les différentes possibilités de déroulement d'une interaction. Pour un système complexe, il peut y avoir un nombre astronomique de cas d'utilisations possibles, parfois sans exemplifier adéquatement le comportement du système. La solution adopté par TAREFA est de définir un sous ensemble de cas d'utilisation en termes de singularités et de combinaisons de singularités considérées 'intéressantes', parce qu'ils ont des caractéristiques qui aident à exemplifier des aspects cruciaux du comportement du système. Nous définissons que les singularités intéressantes peuvent appartenir à deux catégories :

- les **singularités critiques**, qui sont les obstructions aux épisodes du cas d'utilisation essentiel ; ces obstructions sont dites critiques parce que ces épisodes sont centrales pour atteindre les buts;

- les **singularités représentatives**, qui sont les obstructions qui arrivent le plus fréquemment;

Evidemment ces catégories sont non-orthogonales, c'est à dire, qu'i y a des intersections entre elles. De plus, une combinaison de singularités est considérée comme une singularité elle-même. Les singularités critiques sont typiquement associés aux conditions des épisodes.

TAREFA propose un ensemble d'heuristiques pour aider à identifier les cas d'utilisations singuliers. Les cas d'utilisations singuliers seront alors les cas d'utilisations où :

- Chaque but dans la hiérarchie de buts (plan) doit être accompli avec succès par un épisode dans au moins un cas d'utilisation;

- Les dépendances logiques et temporelles de la hiérarchie des buts sont respectées : il n'y a pas besoin de considérer les buts hors ordonnancement;

- L'ensemble total des cas d'utilisations doit contenir au moins une instance de chaque catégorie de singularité - critique ou représentative - mais chaque singularité est associée à un cas d'utilisation seulement, sans répétitions.

•

Description du Cas d'utilisation singulier UC5 - 'Selecting Get Cash;Incorrect Option'
Episode 1 - OK (comme la description du cas normal - UC0)
Episodes 3, 4 et 5 : Non réalisés
Episode 2 :[2] Demander Retrait - Occurrence de la Singularité 5
Plan :
Pre_Cond : : Prêt à Utiliser (ATM)
Initiator : ATM
Action : Rendre Visibles les transactions
Post_Cond : Transactions visibles à l'utilisateur

Pre_Cond : Transactions visibles à l'utilisateur
Initiatior : User
Action : Commander transaction 'Demander Retrait'

Pre_Cond : Transactions visibles à l'utilisateur
Initiatior : ATM
Action : Accepter le commande de 'Demander Retrait'

Pre_Cond : Transaction_commandée <> Options Disponibles
Singularité : Singularité#5 (Incorrect Option)
Initiator : ATM
Service (D)efensif du Système :
D1. Ne rendre visibles que les transactions disponibles
Service (C)orrectif du Système :
C1. Permettre l'option d'Annulation' ou 'Cancel' de l'opération, suivie d'une confirmation

Resultat : Aucune transaction est initiée

Figure 6.14 - Description d'un cas d'utilisation singulier en utilisant le schéma grammatical

La description d'un cas d'utilisation singulier (le cas d'utilisation singulier UC5 '**Selecting Get Cash;Incorrect Option'**) en utilisant le schéma grammatical est montré dans figure 6.14. Les épisodes réalisés sans aucune singularité (comme l'épisode 1) suivent le déroulement de l'épisode du cas normal UC0, et cela est indiqué par l'expression 'OK'. Comme la singularité #5 arrive à l'épisode 2 dans le cas UC5 montré, ses actions, l'identification de la singularité et la proposition des services défensifs et correctifs sont décrits. Les épisodes 3,4 et 5 ne sont pas réalisés parce qu'aucune transaction est initiée - un résultat de l'occurrence de la singularité - et cela est indiqué par l'expression 'Non réalisé' dans la description.

22.1.5Le Niveau Opérationnel

C'est la description du cas d'utilisation au niveau plus technique, au niveau de la conception objet de haut niveau : les objets clients et serveurs qui interagissent, leurs attributs candidats et leurs services, tous décrits de façon abstraite puisque nous sommes dans l'IB. D'autres classes d'objets vont apparaître plus tard pendant la conception détaillée et la réalisation. Ce niveau là est plutôt adéquat **aux concepteurs** pour raffiner et identifier les informations précises sur le comportement du système et permettre l'allocation préliminaire des données (attributs) et des services aux objets. Le système est considéré comme un ensemble d'objets qui participent aux cas d'utilisation. En général, il n'y a pas besoin de montrer ces cas d'utilisation opérationnels à l'utilisateur parce qu'il n'est pas intéressé par ce niveau de détail du système. Mais ce niveau permet de définir une architecture orientée objet préliminaire pour la conception.

TAREFA adopte une description de ce niveau pour faciliter l'intégration avec la méthode orientée objet Objectory. Comme nous l'avons vu dans le chapitre 2, Objectory a deux étapes pour l'analyse : la première pour construire le modèle des besoins (y compris le modèle de cas d'utilisation, le modèle du domaine du problème et une description préliminaire des interfaces) et la deuxième pour construire le modèle

d'analyse, où une architecture basée sur les trois types d'objet (Objet d'Interface IOB, Objet de Contrôle COB et Objet Entité EOB) est proposée. TAREFA permet la classification des objets selon les trois mêmes types et en plus l'identification de classes des objets responsables des services, tout cela à partir de la description des cas d'utilisation au niveau opérationnel. Tout le comportement défini pour le système dans un cas d'utilisation est distribué entre les différents objets qui coopèrent à sa réalisation.

Dans TAREFA, les cas d'utilisation opérationnels sont décrits sous forme de **tableaux**. L'utilisation de tableaux pour les représenter n 'est pas une nouveauté. [Rubin & Goldberg 92] a utilisé des tableaux pour sa méthode d'analyse orienté objet OBA. [Dano et al. 96], pour une approche orientée objet, utilisent deux types de tableaux pour décrire les cas d'utilisations, qui contiennent respectivement l'ensemble des paires (fonctions, déclencheur) pour réaliser le cas d'utilisation et la configuration des fonctions (état des objets et conditions de déclenchement) de chaque cas d'utilisation.

Les avantages de ce type de description en tableaux sont multiples :

- Cette description en tableaux d'un cas d'utilisation peut être facilement intégrée à une approche orientée objet comme cela a été déjà fait par OBA [Rubin & Goldberg 92] et [Dano et al. 96];

- Ce type de notation est considérée comme rigoureuse bien que non formelle : la structure en tableau permet de minimiser les problèmes d'ambiguïté et d'inconsistance des descriptions en langage naturel;

- Une outil informatique pour assister l'édition et la manipulation d'une notation en tableau est facile à développer (p.ex. en utilisant une base de données relationnelle);

- La notation permet un raffinement et une formalisation itérative pour la construction d'une spécification conceptuelle du système (p.ex. en utilisant DIANE+ [Tarby & Barthet 96] voir section 6.2) ou même pour une spécification formelle (par exemple en utilisant ICO [Palanque 92]).

Une description de cas d'utilisation au niveau opérationnel est faite à travers un ensemble de tableaux :

- le tableau Script

- le tableau Objets/Rôles

- le tableau Alias

- le tableau Services

- le tableau Attributs

- le tableau Etat de l'Objet

Le but et la description des colonnes de chaque tableau sont montrés dans la figure 6.15. Les caractéristiques de chaque tableau sont expliquées ensuite.

Un tableau **Script** est construit pour chaque épisode d'un cas d'utilisation Singulier, voir figure 6.16. Les actions qui font partie de l'épisode sont énumérées. Tout script commence par une action fantôme (*dummy*) 'Initiate' et finit par une action fantôme 'Terminate', comme proposé par [Dano et al. 96]. En général, l'initiation est considérée

comme un message d'activation et l'initiateur est soit l'utilisateur soit un objet du système. Le client et le type de l'objet responsable (serveur) de chaque action sont determinés. Parfois il est encore tôt pour déterminer la classe responsable de la fourniture d'un service qui va réaliser l'action parce que c'est une décision en général prise pendant la conception. Cependant, si c'est possible, le nom peut être indiqué dans la colonne Object responsable.

La détermination du type de l'objet responsable est faite grâce à 3 heuristiques, adaptés de la proposition de [Jacobson 92]. Les heuristiques sont les suivantes :

1. Chaque objet présent dans le Modèle d'Objets Métiers du Domaine du Problème est un Objet Entité (EOB);
2. Chaque interaction entre l'utilisateur et le système est réalisée par l'intermédiaire d'au moins un Objet d'Interface (IOB); En général, les périphériques particuliers (comme l'imprimante, le clavier, l 'écran ou un autre périphérique spécial) sont associés à l'interaction utilisateur-système et peuvent être manipulés par des IOB particuliers;
3. Pour chaque opération ou décision non supportées par les heuristiques 1 et 2 ci-dessus il est défini un Objet de Contrôle (COB) pour les fournir; Chaque transaction (dans un système orienté transactions) est contrôlée par un COB spécifique, c'est à dire, l'invocation des services des objets d'interface (IOB) et des objets entité est contrôlé par un objet de contrôle.

Tableau	But	Colonnes (Informations décrites)
Script	Décrire les épisodes des cas d'utilisation	• ***Action Id***, un identificateur de l'action; • ***Episode/Action name***, indique le nom d'un episode et le nom de l'action; • ***Initiateur***, pour indiquer l'agent qui a initié (ou invoqué) l'action et qui sera son **client**; • ***Type de l'Objet responsable***, typiquement pour indiquer si l'objet est un Objet Entité (*Entity Object* ou EOB), un Objet Contrôle (*Control Object* ou COB) ou un Objet Interface (*Interface Object* ou IOB); • ***Objet responsable*** : Si identifié, le nom de la classe qui a la responsabilité de fournir un service pour réaliser l'action et qui sera son **serveur**;
Objets/ Rôles	Décrire chaque participant en termes du rôle qu'il joue dans le système.	• ***Nom*** correspond au nom générique des participants; ce choix doit être documenté dans le tableau **Alias;** • ***Type***, pour indiquer si l'objet est un Agent (l'utilisateur ou un autre participant actif des interactions), un Objet Entité (*Entity Object* ou EOB), un Objet Contrôle (*Control Object* ou COB) ou un Objet Interface (*Interface Object* ou IOB); • ***Définition*** est une définition informelle d'un participant; • ***Traces*** sont les Ids des scripts dans lesquels l'agent participe; les traces sont la représentation explicite d'une relation 'utilisé en'; • ***Rôle de l'agent***, choisi entre les suivantes valeurs possibles : **I**nitiateur (client), **R**esponsable (serveur) or les deux - **I/R**).
Alias des agents	Garder la correspondance entre les noms génériques et les noms spécifiques des agents	• ***Nom générique***, utilisé dans les cas d'utilisation, en général utilisé pour décrire le rôle. • ***Noms spécifiques***, utilisés souvent dans les interviews et descriptions textuelles et qui font référence à individus ou éléments particuliers
Services	Décrire les action initiées par le système.	• ***Service Id***, un code identificateur du Service; • ***Nom du Service***, le nom qui identifie le service; • ***Définition***, une définition informelle du service; • ***Informations***, l'ensemble des informations dont le service a besoin pour être réalisé; typiquement il y a deux types d'informations : **entrée** ou **contrôle** • ***Objet Responsable***, l'identificateur du objet qui est le responsable de la fourniture de ce service, c'est à dire, son serveur.
Attributs	Décrire quelles sont les informations nécessaires pour chaque service.	• ***nom*** de l'attribut = le nom qui identifie l'attribut; • ***définition*** = une définition informelle de l'attribut; • ***propriété de*** = le nom de la classe qui est caractérisée par l'attribut; • ***accès*** = si la classe va fournir un service pour permettre la consultation de l'attribut, cette colonne contient le nom du service; sinon 'non'; • ***modification*** = si la classe va fournir un service pour permettre modifier l'attribut, cette colonne contient le nom du service; sinon 'non' • ***collection/unitaire*** = si l'attribut représente une collection, cette colonne contient la cardinalité de la collection; sinon 'unitaire'; • ***état*** = si l'attribut défini l'état de l'objet, cette colonne contient une référence au tableau de l'Etat de l'Objet; sinon 'non' • ***visible*** = si l'attribut est montré dans l'interface utilisateur, le nom de l'objet de l'interface (IOB) associé; sinon 'non';
Etat de l'Objet	Décrire le cycle d 'états d'un objet; Il faut un tableau pour chaque objet.	• *Id de l'action* = un code identificateur du script qui contient l'action et de l' action (invocation de service) ou l'objet change d'état • pour chaque action, il est indiqué *l'état initial* et *l'état final* pour faire référence respectivement à l'état de l'objet avant et après l'exécution de l'action.

Figure 6.15 - Description du contenu des tableaux qui décrivent les cas d'utilisation au niveau opérationnel

Un exemple du tableau Script SCRIPT 1 pour l'Episode 1 - S'Identifier - du cas d'utilisation Singulier 'Retrait de l'Argent' - Cas Normal' (UC0) est montré dans la figure 6.16. L'action Id est composé de l'identification du cas d'utilisation singulier (UC0), l'identification de l'épisode (EP1) et l'identification de l'action (0, 1, 2, etc). Comme annoncé précédemment la première action est toujours l'action fantôme 'Initiate' identifiée par '0' est la dernière- l'action 8 - est l'action 'Terminate'.. L'initiateur de l'action 'Initiate' est le Gestionaire des Transactions d'ATM (GTATM) pour indiquer qu'il y a un objet de contrôle GTATM qui contrôle l'activation des transactions et à qui retourne le contrôle après l'action 'Terminate' (voir l'objet responsable de l'action 8). L'objet responsable pour la transaction d'identification qui commence avec cette action 'Initiate' est un objet de contrôle (COB) de nom 'Identification'. Cet objet va demander des services à d'autres objets (voir les actions 2, 3, 4, 6 et 7) pour la réalisation de l'identification du client, ce qui implique des services d'interface (comme les action 2, 4 et 6 dont l'objet responsable a le type IOB) et services de domaine (comme les actions 3 et 7 dont l'objet responsable a le type EOB). Bien sûr, le client est l'initiateur des actions 1 et 5, parce qu'il fournit son code (action 1) et son mot-de-passe (action 5) mais l'affichage et le contrôle de l'entrée de ces informations sont sous la responsabilité d'un objet d'interface (IOB). Parfois la colonne Objet Responsable n'est remplie que après la définition des services dans le tableau Services. C'est pour cela qu'il y a quelques champs vides.

Une description complète comprend aussi d'autres scripts comme par exemple les scripts pour les épisodes de Demande de Retrait, Choix_Valeur, Prise des Objets (carte, argent, reçu) et Finalisation dans le cas d'utilisation 'Retrait d'Argent'.

SCRIPT 1

action Id	Nom	Initiateur (client)	Type de l'Objet Responsable	Objet Responsable (serveur)
UC0.EP1.0	Initiate	GTATM	COB	Identification
UC0.EP1.1	Fournir Id du Client	Client	IOB	
UC0.EP1.2	Accepter Id	Identification	IOB	
UC0.EP1.3	Verifier Id	Identification	EOB	
UC0.EP1.4	Demander Mot-de-Passe	Identification	IOB	
UC0.EP1.5	Fournir Mot-de-Passe	Client	IOB	
UC0.EP1.6	Accepter Mot-de-Passe	Identification	IOB	
UC0.EP1.7	Verifier Mot-de-Passe	Identification	EOB	
UC0.EP1.8	Terminate	Identification	COB	GTATM

Figure 6.16 - Exemple de tableau Script pour l'Episode 1 - S'Identifier - du cas d'utilisation Singulier 'Retrait de l'Argent' - Cas Normal' (UC0)

Le tableau **Objets/Rôles** (voir figure 6.17) sert pour décrire le rôle que joue chaque participant dans le système. La colonne ***Nom*** est remplie d'abord par le nom des acteurs particuliers; ensuite, par des noms génériques choisis pour classer les participants spécifiques. Ce choix doit être documenté dans le tableau **Alias.**

Un exemple du tableau **Objets/Rôles** est montré dans la figure 6.17. Il décrit un sous-ensemble des participants aux actions de l'ATM. Ces participants sont soit des agents (comme Client et l'opérateur de l'ATM) soit des objets (comme l'IOB *Customer Panel*, l'EOB Compte et l COB Identification). Pour chaque participant, une définition informelle est donnée (dans la colonne **Définition**) et leur rôle est défini : typiquement les utilisateurs sont des **I**nitiateurs, les objets entités (EOBs) sont des **R**esponsables et

les objets d'interface (IOBs) et de contrôle (COBs) sont parfois Initiateurs parfois Responsables (**I/R**). La colonne **Traces** montre l'identification des scripts dans lesquels chaque participant intervient; par exemple, le client participe aux scripts 1, 3 et 4. En fait, nous pouvons remarquer que le client fournit son code et son mot-de-passe (voir respectivement l'action 1 et l'action 5 dans la figure 6.16) dans le script SCRIPT 1.

Objets/Rôles

Nom	**Type**	**Définition**	**Traces**	**Rôle**
Client	Agent	La personne qui utilise l'ATM	1, 3, 4	I
Operateur_ATM	Agent	La personne qui doit mantenir l'ATM (provision de l'argent, imprimer le 'log' de transactions, etc)	8	I
ATM	Agent	la machine que le client utilise pour executer les transactions bancaires	1,	I/R
Compte	EOB	Objet du domaine du problème	3,	R
Reçu	EOB	Objet du domaine du problème	4	R
Carte	EOB	Objet du domaine du problème	1	R
Customer Panel	IOB	Objet qui réalise les interactions via l'écran et le clavier	1,2,3,4,5	I/R
Dispenser_controller	IOB	Objet qui contrôle le 'cash dispenser'	3	I/R
Container_controller	IOB	Objet qui contrôle le matériel 'cash container'	3	I/R
Lecteur_controller	IOB	Objet qui contrôle le lecteur de carte	1,5	I/R
Imprimante_controller	IOB	Objet qui contrôle l'imprimante	4	I/R
Identification	COB	Objet qui contrôle la transaction correspondant à l'épisode 1 (S'Identifier)	1	I/R
Retrait	COB	Objet qui contrôle la transaction correspondant à l'épisode 2 (Demander Retrait)	2	I/R
Choix_Valeur	COB	Objet qui contrôle la transaction correspondant à l'épisode 3 Choisir Valeur	3	I/R
Prise	COB	Objet qui contrôle la transaction correspondant à l'épisode 4 Prendre les Choses	4	I/R
Finale	COB	Objet qui contrôle la transaction correspondant à l'épisode 5 S'en aller	5	I/R
Solde	COB	Objet qui contrôle la transaction correspondant à l'épisode 6 Demander Solde	6	I/R
Relevé	COB	Objet qui contrôle la transaction correspondant à l'épisode 7 Demander Relevé	7	I/R

Figure 6.17 - Exemple de tableau Objets/Rôles pour l'exemple ATM

Le tableau **Alias** sert pour garder la correspondance entre les noms génériques des participants (les noms utilisés dans le tableau **Objets/Rôles**) et les noms spécifiques souvent utilisés dans les interviews et descriptions textuelles.

Un exemple du tableau **Alias** est montré dans la figure 6.18, qui décrit un extrait des noms génériques (pour l'étude de cas ATM) en rapport aux noms spécifiques utilisés comme synonymes dans les descriptions des cas.

Alias

Nom générique	Nom spécifique
Client	utilisateur, client de la banque, propriétaire de compte
Opérateur	operateur_ATM, Jules (un opérateur particulier)
ATM	machine, Automatic Teller Machine, distributeur automatique de billets, distributeur automatique, point d'argent

Figure 6.18 - Exemple de tableau **Alias**

Le tableau **Services** décrit les actions initiées par le système, y compris les services défensifs et correctifs associés aux singularités dans le niveau téléologique. Les colonnes ***Id du service***, ***Nom du service*** et ***Definition informelle*** servent à identifier le service et ses buts. .La colonne ***Informations*** énumère l'ensemble des informations dont le service a besoin pour être réalisé. Typiquement il y a deux types d'informations : **entrée** et **contrôle** qui sont les informations qui servent respectivement comme données à manipuler et comme paramètres utilisées pour les décisions (sélections, itérations, etc) [Sutcliffe 96]. La détermination de la classe de l'objet responsable est faite à travers les 4 heuristiques suivantes :

1. Classer les services en trois types : services de domaine, services d'interface et services de contrôle. Ces types sont similaires à la typologie associé aux objets. En fait, l'objet responsable pour un service sera du même type que le service;
2. A chaque service d'interface dépendant d'un périphérique particulier, l'objet contrôleur de ce périphérique est le responsable; toute manipulation de l'écran et du clavier, omniprésente dans la plupart des interfaces standard des ordinateurs, sera attribuée à un objet générique appelé 'Customer Panel', dont la définition précise du comportement et de la présentation doit être faite pendant l'étape de conception;
3. A chaque service de domaine, l'objet du domaine correspondant est le responsable;
4. A chaque service de contrôle, l'objet de contrôle correspondant est le responsable.

48)

Un exemple du tableau **Services** est montré dans la figure 6.19. Il décrit un sous ensemble des services de l'ATM identifiés par une identification (**Id du service**) et un nom (**Nom du service**). Pour chaque service, il faut décrire son but (**Définition Informelle**), les informations nécessaires pour sa réalisation (**Informations Nécessaires**) et l'Objet Responsable par sa réalisation (**Objet responsable**). Par exemple, Le service S1 - de nom 'Accepter Id' consiste de récupérer les codes de la banque, de l'agence et du client ainsi que le mot-de-passe; l'Objet Responsable peut bien être le *Customer Panel* (si le client tape toutes ces informations) ou le *Lecteur-Controller* (si le client utilise la carte pour les codes). Ce choix est à définir plus tard au moment de choix des alternatives de conception.

49)

Id du Service	Nom du Service	Définition Informelle	Informations Nécessaires	Objet Responsable
S1	Accepter Id	Recevoir les Informations pour Identification du Client	Code Banque, Code Agence, Code Compte, Mot-de-Passe	*Customer Panel* OR *Lecteur Controller*
S2	Vérifier Id du Client	Vérifier si le Id du client est valide et s'il a une permission d'utilisation	Code Banque, Code Agence, Code Compte, Mot-de-Passe	Carte

Figure 6.19 - Exemple de tableau Services

Une description complète comprendrait aussi d'autres services comme par exemple : S3 - Demander Mot-de-Passe, S4 - Accepter Mot-de-Passe, S5 - Vérifier Mot-de-Passe, S9 - Demander Valeur, S10 - Accepter Valeur, pour ne citer que quelques uns.

Le tableau **Attributs** montre les attributs candidats des objets de l'ATM, qui sont les informations nécessaires pour chaque service.

Un exemple de tableau Attributs est montré dans la figure 6.20. Dans ce tableau, tous les attributs sont des propriétés de l'objet entité (EOB) Compte. Les 4 premiers ne peuvent qu'être lus (voir les noms des services qui permettent accès dans la colonne **accès**). Le solde, par contre, est une information qui peut être modifiée (voir la colonne **Modification** qui contient le nom du service qui permet la modification) et pour cela il est intéressant de définir son cycle d'états par l'intermédiaire d'un tableau **Etat de l'Objet** (voir le nom de l'objet dans la colonne **Etat**). En plus, tous ces attributs sont unitaires (voir colonne **Collection/unitaire**) et visibles (voir colonne **Visible**).

Attributs

Nom	Définition	Propriété de	Accès	Modification	Collection/unitaire	Etat	Visible
Code Banque	Code de la banque où est la compte du client	Compte	lire_banque	non	unitaire	non	*Customer Panel*
Code Agence	Code de l'agence où est la compte du client	Compte	lire_agence	non	unitaire	non	*Customer Panel*
Code Compte	Code de la compte du client	Compte	lire_compte	non	unitaire	non	*Customer Panel*
Mot-de-Passe	Mot-de-Passe du Client	Compte	lire_mot_de_passe	non	unitaire	non	*Customer Panel*
Solde	Solde disponible dans le compte du client	Compte	consulter_solde	changer_solde	unitaire	Compte	*Customer Panel*

Figure 6.20 - Exemple de tableau Attributs

Le tableau **Etat de l'Objet** décrit le cycle d 'états d'un objet. Il faut construire un tableau pour chaque objet, en décrivant les états initiaux et finaux associés aux actions sur cet objet. Dans l'exemple montré dans la figure 6.21, un extrait du cycle de l'objet Compte est décrit. L'action *UC0.EP3.7 Verifier Solde* n'altère pas l'état de l'objet parce elle est une consultation; mais l'action *UC0.EP3.9 Actualiser solde (nouveau_solde)* change la valeur du *solde* ancien par le valeur spécifiée par le paramètre *nouveau_solde*. Cette action peut être invoquée dans une transaction de retrait (où le *nouveau solde* est plus petit que le *solde*) ou dans une transaction de virement (où le *nouveau solde* est plus grand que le *solde*).

Etat de l'Objet : Compte

Id de l'action	Etat Initial	Etat Final
UC0.EP3.7 consulter Solde	solde	solde
UC0.EP3.9 Actualiser solde (nouveau_solde)	solde	nouveau solde

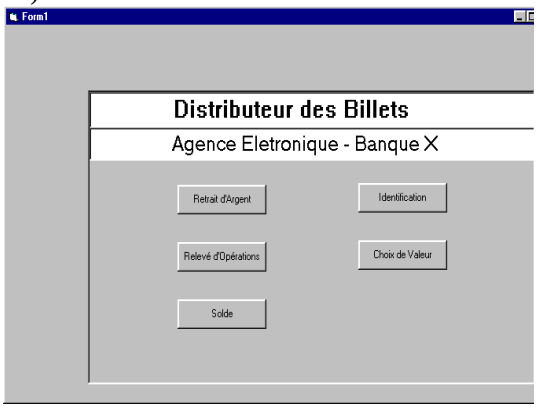
Figure 6.21 - Exemple de tableau Etat des Objets pour l'Objet Compte.

22.1.6Le Niveau Concret

Au niveau concret, les Cas d'utilisation Concrets sont définis**.** Un **cas d'utilisation concret** est une occurrence d 'exécution particuliere d'un cas d'utilisation singulier **du point de vue de l'utilisateur**. Il correspond à la description concrète d'une situation d'utilisation (actuelle ou future) et il contient l'occurrence ou l'instanciation de ses caractéristiques concrètes (comportement détaillé, conditions d'utilisation, singularités)

sans prendre en compte les détails techniques spécifiques du système au niveau opérationnel. Ce concept correspond à la définition conventionnelle de 'scénario' cf. Chapitre 3.

50)

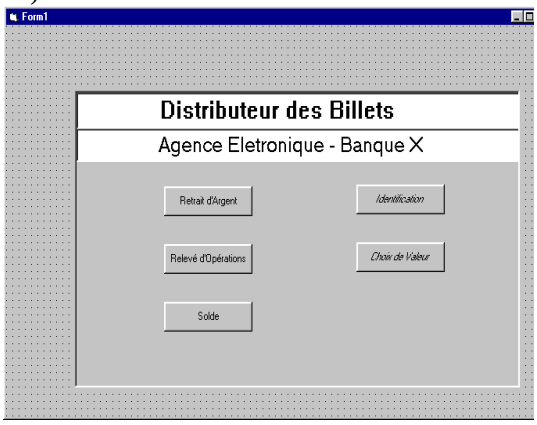


51)

(a) - *Storyboard* #1

52)

53)



(b) - *Storyboard* #2

64)

54)

55) Narration du cas concret.

56)

57)

Jules arrive à l'ATM, qui est prêt à être utiliseé et qui montre les options de transaction (storyboard #1). Jules veut faire un retrait d'argent et il demande l'option 'Choisir valeur'. L'ATM vérifie l'option choisie et détecte que cette option n'est pas disponible parce qu'elle doit être réalisée après l'option 'Retrait d'Argent'. La transaction n'est pas commencée.

58)

59) Ce cas exprime l'occurrence de la singularité #5.

60) Une variation de ce cas concret est la prise en compte du service défensif D1 - 'Ne rendre visibles que les transactions disponibles' (dont la réalisation est possible) :

61)

62)

63)

Jules veut faire un retrait d'argent et il demande l'option 'Choisir valeur'. L'ATM n'accepte pas cette option parce qu'elle n'est pas disponible : c'est indiqué par le bouton grisé. (storyboard #2).

Une autre variation suggérée par l'utilisateur est supprimer de l'écran les options non disponibles, à la place de les montrer grisées.

Figure 6.22 - Un exemple de Cas d'utilisation concret de l'ATM en utilisant l'outil Visual Basic pour construire les *storyboards*.

A ce niveau, on ne se préoccupe pas encore de l'apparence détaillé de l'interface mais uniquement de ce qu'elle fait. Donc, il est possible de dessiner des images d'écran (les objets d'interaction comme les menus, les boutons, etc) sur papier ou de développer des prototypes bas-niveau. Le mot 'papier' ici peut signifier divers types de matériaux, comme les feuilles de papier, les Post-Its, les transparents, etc. Ce travail est un type de prototypage informel très utilisé quand on veut illustrer le comportement du système ou quand on veut générer des idées rapides (*brainstorming)*. Pour les prototypes bas-niveau, on peut utiliser des outils de prototypage rapide - comme par exemple Visual Basic (PC) ou Macromedia Director ou l'Hypercard (Apple Macintosh) - ou encore des outils pour la construction de storyboards ou de sketches, comme par exemple SILK [Landay & Myers 95].

Le choix bien sûr est déterminé par les ressources (matérielles et logicielles) disponibles dans l'organisation et décrits dans le Modèle de Plateforme. Cependant, si les ressources ne sont pas encore disponibles (si on doit encore les proposer et les acheter) le papier peut servir pour stimuler la création flexible et créative des nouvelles

interfaces, sans les contraintes associées aux outils utilisés.

Un exemple de cas d'utilisation concret pour l'exemple de l'ATM fait avec l'outil Visual Basic est montré dans la figure 6.22. Ce cas d'utilisation concret est dérivé du cas d'utilisation singulier UC5.

22.2Construction des Cas d'Utilisation

L'approche adoptée par TAREFA est de spécifier les besoins d'un système à partir d'un modèle de tâches existantes. Au contraire de [Normand ergo IA 94/, qui utilise une hiérarchie d'abstraction des modèles de tâche comme fil conducteur pour cette spécification des tâches interactives futures, nous sommes d'accord avec [jacobson 92/, [Carrol 95] et [Rosson 95/, qui mettent le concept de modèle de cas d'utilisation comme fil conducteur de la conception, en particulier dans l'étape de l'ingénierie des besoins. Ainsi, un cas d'utilisation est construit pour déterminer les besoins de chaque tâche Interactive et automatique.

La construction des cas d'utilisation suit une approche descendante, qui se résume en :

- **construire les cas d'utilisations essentiels** à partir de la structure de buts/sous buts du modèle des tâches minimales et de la définition des services essentiels du système;

- **construire les cas d'utilisation singuliers** en définissant les singularités associées aux buts et aux services de chaque cas d'utilisation essentiel;

- si on veut commencer la conception orientée objet**, construire les cas d'utilisation opérationnels** en définissant les objets clients et serveurs des services du système décrits au cas d'utilisation singulier;

- **construire** à partir des cas d'utilisation singuliers des propositions **des cas d'utilisation concrets** qui seront expérimentés itérativement avec l'utilisateur.

Chaque activité de cette approche sera plus détaillée dans les paragraphes suivants.

22.2.1Construction des cas d'utilisations essentiels

Comme nous l'avons vu dans le chapitre I, le comportement d'un système interactif peut être vue comme l'entrelacement d'événements réceptifs et expressifs. C'est un des buts de l'Ingénierie des Besoins que de déterminer quelles performances supportera le système, le pourquoi de ce choix, et comment seront-elles réalisées en termes de formulations des utilisateurs et fonctions du système.

Du point de vue adopté par TAREFA, un cas d'utilisation represente ce comportement en modélisant:

- les événements dont l'utilisateur est le déclencheur, c'est à dire, les formulations des utilisateurs;

- les événements du système qui sont soit une réception d'une formulation (les fonctions réceptives du système) soit une réponse à une formulation (les fonctions expressives du système).

Construire les cas d'utilisations essentiels implique de définir les deux composants

de l'interaction utilisateur-système : les formulations de l'utilisateur et les services - dits essentiels à ce niveau d'abstraction - du système. Le premier composant, les formulations, est dérivé directement à partir de la structure de buts/sous buts du modèle des tâches minimales. Le deuxième, les services essentiels, sont proposés comme la réaction du système à la formulation de l'utilisateur.

Pour chaque tâche interactive représentée par un cas d'utilisation, la formulation est construite à partir du plus haut niveau commun du modèle des tâches de l'utilisateur et les fonctions expressives sont construites à partir des services du système qui composent la réaction à ces formulations. A ce plus haut niveau de description d'un cas d'utilisation, un service est une abstraction des actions à être exécutées par le système pour transformer l'état courant en l'état final désirée où le but de la tâche interactive est atteint.

La description des services essentiels est la description de ce comportement de haut niveau, c'est à dire, les services exécutés par le système comme des composants des interactions utilisateur-système afin d'arriver aux buts des activités de travail. Le mot 'essentiel' ici a le même sens utilisé dans les cas d'utilisation essentiels : la préoccupation de l'essence du comportement du système sans considérations technologiques. Ces services essentiels sont, en citant Larry Constantine [Constantine 95] *'the essence of 'WHAT' the system does in the course of the use case '.*

La définition de ces services essentiels est sans doute un problème. [J. Johnson & Nardi 96] affirme que *'developing application software that supports well all aspects of a task is extremely difficult' !*

Les services essentiels sont des services qui peuvent être offerts par une 'classe' de systèmes. L'utilisation d'une technologie spécifique est une façon d'operationaliser ces services. Plusieurs systèmes peuvent être conçus pour supporter les buts du modèle de tâche minimale (les intentions de l'utilisateur), c'est à dire, une classe des systèmes avec la même fonctionnalité. Ils sont distingués par **la façon** dont les buts sont operationnalisés, quelles actions sont exécutées pour arriver aux buts. Quand la fonctionnalité est la même, les cas d'utilisations vont permettre de définir avec plus de précision le 'comment' l'utilisateur veut utiliser les tâches interactives. Les services d'un système vont soutenir le tâches interactives pour arriver aux buts. En vérité, ces services devront être raffinés pendant tout le processus de développement jusqu'à la définition du comportement détaillé du système.

Ainsi, chaque tâche primitive du modèle de tâches minimales sera considérée comme une formulation. A chaque formulation, les fonctions essentielles réceptives et expressives du système sont définies, comme nous le montrons dans la figure 6.23. Une ligne de séparation plus grosse indique que pour chaque formulation le système peut avoir un ensemble (possiblement unitaire) de fonctions réceptives et expressives.

65)
66)
67)
68)

Formulation (tâches minimales)	**Fonctions Réceptives du système**	**Fonctions Expressives du système**
S'Identifier	Accepter Identification	Vérifier Identification
Demander Retrait	Accepter Demande	Demander Valeur
Choisir Valeur	Accepter Valeur	Vérifier Soldes (ATM et Carte)
		Faire Sortir argent, carte et reçu
Prendre l'argent		Devenir prêt pour autre opération
S'en aller		Devenir prêt à utiliser pour autre utilisateur

Figure 6.23 : Formulations et Services Essentiels du Système associés pour l'exemple de l'ATM

Un cas d'utilisation essentiel est décrit par une narration simple structurée en deux colonnes : une avec les formulations des utilisateurs et l'autre avec les services du système correspondants aux formulations. L'ordonnancement des événements suit l'ordonnancement logique et temporel du modèle de tâches, donc des formulations. Evidemment l'utilisateur peut réaliser plusieurs tâches en parallèle (taper au clavier et parler au téléphone par exemple) mais cela change si ces tâches sont réalisées avec le soutien d'un système informatique : même s'il réalise plusieurs tâches entrelacées (interrompues et reprises plusieurs fois), en fait à un instant donné il ne traite qu'une seule tâche interactive à la fois, cela indépendamment des possibilités de parallélisme des fonctions de l'ordinateur. C'est pour cela que TAREFA favorise les séquences des événements comme narrations. Alors, pour construire les cas d'utilisation il faut construire des narrations de formulations et fonctions du système, c'est à dire, des séquences événements utilisateur-système. Un cas d'utilisation type est le résultat d'un choix de sequencement, qui peut être détaillé au fur et à mesure du développement. Les heuristiques pour choisir les sequencements à proposer sont :

- le sequencement habituel de l'utilisateur;
- le sequencement prévu (formel) de l'organisation
- des sequencements nouveaux proposés pour essai de l'utilisateur;

69)
Nous avons choisi le sequencement habituel de l'utilisateur,comme illustré dans la figure 6.24.

70)

Tâches Minimales	Services du système
S'Identifier	ATM prêt à être utilisé Accepter Identification Vérifier Identification Offre Services
Demander Retrait	Accepter Demande Demander Valeur
Choisir Valeur	Accepter Valeur Verifier Soldes (ATM et Carte) Faire Sortir argent, carte et reçu
Prendre les choses	ATM prêt pour une autre opération
S'en aller	ATM prêt à être utilisé

Figure 6.24 : Le Cas d'Utilisation Essentiel pour Retrait d'Argent de l'exemple ATM

Comme [Constantine 95], nous adoptons l'ordre du tableau toujours comme la séquence d'exécution simple des interactions. Cependant, chaque formulation peut avoir différentes relations temporelles avec les autres formulations d'un même cas d'utilisation. Par exemple, la formulation ***S'Identifier*** peut être exécutée de manière entrelacée avec les formulations ***Demander Retrait et Choisir Valeu****r*, cf. le modèle de tâches minimales montré dans la figure 6.2(b***).***

L'ensemble des signes définis pour TAREFA est inspiré de la notation UAN [Hartson et al. 90] et est montré dans la figure 6.25.

Relations Temporelles	**Signes de la Notation adopté par TAREFA**
1.Groupe	(A)
2.Choix (Alternative)	A | B
3.Parallele	A | | B
4.Repetition	A^{+}, A^{n}, A^{*}
5.Interruption	$B \rightarrow A$
6.Entrelacement	A ||| B

7.Simultaneité	A, B
8 Ordre Indépendant	A & B
9.Attente d'un temps ***n***	A (t > ***n*** seconds) B

Figure 6.25 - Signes de relations temporelles d'ordonnancement de TAREFA

La plupart de ces relations sont adoptées par les modèles des tâches et les modèles de dialogue (voir la figure 1.9 dans le chapitre 1). La notion de groupe a été ajouté pour factoriser l'application des autres relations.

La figure 6.26 montre un exemple de l'utilisation de ces relations temporelles entre les formulations. Cette expression sera importante plus tard pour le raffinement du cas d'utilisation.

```
Retrait d'Argent :
        S'Identifier ||| (Demander Retrait Choisir Valeur)
        Prendre les choses
        S'en aller
```

Figure 6.26 - Expression des relations temporelles entre les formulations

22.2.2Construction des Cas d'utilisations Singuliers

D'abord, le cas d'utilisation singulier qui correspond au 'cas normal' est construit par l'intermédiaire du raffinement des épisodes du cas d'utilisation essentiel. Le raffinement à cette étape est réalisé par :

- l'identification de pré-conditions et post-conditions pour chaque action de l'épisode; cette identification est faite à partir du modèle de tâche. La décomposition d'une action du cas d'utilisation essentiel est faite soit à partir du modèle de tâche (les sous actions d'une action), soit à partir de la décomposition des informations manipulées par l'action. P.ex. S'Identifier est décomposé en Fournir Id et Fournir Mot-de-Passe.
- la décomposition des actions en actions plus élémentaires.

Un exemple d'un épisode raffiné d'un cas d'utilisation singulier qui modélise le cas normal a été montré dans la figure 6.12.

Ensuite, il faut identifier les singularités. En général, une façon intuitive d'identifier les singularités d'un cas d'utilisation est d'analyser le cas d'utilisation normal et de répondre aux questions :

' What can go wrong with this action ?'
'What would happen if... ?'.

La procédure pour construire les cas d'utilisation singuliers dans TAREFA est une systématisation de ces questions. Les pré et post conditions d'une action sont évaluées et les singularités identifiées. Une liste de singularités identifiées est faite. L'identification des singularités isolées est déterminée structurellement par l'évaluation des pre conditions, indépendamment du domaine de l'application. Cependant l'identification de combinaisons intéressantes d'un ensemble (deux ou plus) de singularités est faite cas par cas en fonction du domaine de l'application. Cela veut dire que l'occurrence d'une singularité conjointement avec une autre peut avoir une signification spéciale dans un domaine particulier.

Dans la figure 6.27, un extrait de l'identification des singularités du cas d'utilisation Retrait d'Argent de l'exemple ATM est montré. Les actions prises en compte sont les actions du cas d'utilisation singulier 'normal' identifié comme UC0 (voir figure 6.12).

Id de la Singularité	Action/Episode	Service du Système où la singularité arrive	Problème	Singularité (Cause du Problème)	Services (D)efensifs ou (C)orrectifs du Système
Singularité 1	S'Identifier (épisode 1)	Accepter Id	Prêt à Utiliser (ATM) est faux	ATM n'est pas en service	
Singularité 2	S'Identifier (épisode 1)	Accepter Id	Prêt à Utiliser (ATM) est faux	ATM est déjà en service	(d) L'insertion n'est pas permise pendant une transaction
Singularité 3	S'Identifier (épisode 1)	Verifier Id	Id Invalide	Utilisateur n'a pas la permission d'utiliser l'ATM	(c) Exhiber message
Singularité 4	S'Identifier (épisode 1)	Verifier Mot-de- Passe	Invalide (Mot-de-Passe)	Mot-de-Passe fourni par l'utilisateur n'est pas correct	c) ATM permet de repeter l'action 3 fois et a) après 3 fois, la carte est bloquée
Singularité 5	Demander Retrait (épisode 2)	Commander transaction 'Demander Retrait'	Invalide (transaction)	Transaction commandée incorrectement ou option invalide	(d) faire choisir l'utilisateur dans une liste de transactions (c) ATM permet de repeter l'action
Singularité 6	Choisir Valeur (épisode 3)	Accepter Valeur	Non_Accepté (valeur)	Syntatic (incorrect format) or Utilisateur a touché des boutons incorrects	(c) Répéter la saisie, annuler et faire nouvelle sélection
Singularité 7	Choisir Valeur (épisode 3)	Verifier Solde Client	Solde Negatif	la valeur désirée dépasse le limite/solde du client	(d) Exhiber la valeur limite de la carte
Singularité 8	Choisir Valeur (épisode 3)	Verifier Solde ATM	Solde Negatif	la valeur désirée dépasse le solde disponible à l'ATM	(c) Exhiber message
Singularité 9	Prendre les choses (épisode 4)	Faire sortir l'argent	Timeout	L'utilisateur a oublié de le prendre	(c) Bloquer l'argent
Singularité 10	S'en aller (épisode 5)	Faire sortir la carte	Timeout	L'utilisateur a oublié de la prendre	(c) Bloquer la carte
Singularité 11	S'Identifier (épisode 1)	Accepter Mot-de-Passe	Non_Accepté (mot-de-Passe)	Timeout	c) ATM permet de répéter l'action 3 fois et a) après 3 fois, la carte est bloquée
Singularité 12	Demander Retrait (épisode 2)	Commander transaction 'Demander Retrait'	Transaction Non_commandée	Timeout	(c) faire sortir la carte (c) exhiber message d'erreur
Singularité 13	Choisir Valeur (épisode 3)	Accepter Valeur	Non_accepté (Valeur)	timeout	c) ATM permet de répéter l'action 3 fois et a) après 3 fois, la carte est bloquée
Singularité 14	Choisir Valeur (épisode 3)	Accepter Confirmation du Valeur	Non_accepté (Confirmation)	timeout	(c) Répéter la saisie, annuler et faire nouvelle sélection
Singularité 15	Prendre les choses (épisode 4)	Imprimer reçu, faire sortir le reçu	Timeout	L'utilisateur a oublié de le prendre	Rien, car le reçu n'est pas si important
Singularité 16	Prendre les choses (épisode 4)	Imprimer reçu, faire sortir le reçu	Non_Sorti (reçu)	Problème avec l'imprimante	
Singularité 17	S'en aller (épisode 5)	Faire sortir la carte	Non_Sorti (carte)-	Problème avec le 'card reader'	-
Singularité 18	S'Identifier (épisode 1)	Accepter Id	Non_Accepté (Id)	Timeout	a.
Singularité 19	Choisir Valeur (épisode 3)	Préparer l'argent pour sortir	Non_Preparé (argent)	Problème avec le 'cash container'	-
Singularité 20	Prendre les choses (épisode 4)	Faire sortir l'argent	Non_Sorti (argent)	Problème avec le 'cash dispenser'	(d) Première fois, rien est fait (d) autres fois, prévenir a travers le message 'Pas de Reçu'

Figure 6.27 - Identification des Singularités pour le cas d'Utilisation 'Retrait d'Argent'

Un cas d'utilisation singulier est défini pour chaque exception et pour chaque

combinaison d'exceptions considérée comme intéressante, en décrivant si un épisode a été affecté par la conséquence de la singularité. Il y a trois types de conséquence :

1. l'épisode est bien réalisé ('Ok');

2. l'épisode est non réalisé ('NR'), parce que une singularité est arrivé dans l'épisode antérieur;

3. l'épisode est changé pour considérer la singularité.

Le changement d'un épisode implique la proposition de **services défensifs** et de **services correctifs** qui peuvent être définis en plus des services essentiels.

A chaque cas d'utilisation singulier identifié, il est attribué un identificateur (Id) et un nom. Le nom doit être suggestif du comportement du cas d'utilisation. Typiquement le cas normal est appelé 'Cas Normal' est son identificateur est 'UC0'. Pour les autres cas, nous adoptons le nom d'un cas d'utilisation comme la combinaison du dernier épisode réalisé et de la singularité associée à l'épisode courant. Un exemple de définition de cas d'utilisation singulier pour le cas d'utilisation essentiel 'Retrait d'Argent' de l'ATM est montré dans la figure 6.28.

Id du Cas d'Utilisation Singulier	**Nom du Cas d'Utilisation Singulier**	**Episode 1**	**Episode 2**	**Episode 3**	**Episode 4**	**Episode 5**
UC0	Normal Case (it is all OK)	OK	OK	OK	OK	OK
UC1	Inserting Card; System Off	Singularité 1	NR	NR	NR	NR
UC2	Inserting Card; System in Use	Singularité 2	NR	NR	NR	NR
UC3	Identifying Client; Invalid Client	Singularité 3	NR	NR	NR	NR
UC4	Entered PIN; Incorrect PIN	Singularité 4	NR	NR	NR	NR
UC5	Selecting Get Cash; Incorrect Option	OK	Singularité 5	NR	NR	NR
UC6	Amount Entered; Invalid Amount	OK	OK	Singularité 6	NR	NR
UC7	Amount Entered; Beyond Card Limit	OK	OK	Singularité 7	NR	NR
UC8	Amount Entered; Beyond ATM Limit	OK	OK	Singularité 8	NR	NR
UC9	Cash Ejected; Timeout	OK	OK	OK	Singularité 9	NR
UC10	Card Ejected; Timeout	OK	OK	OK	OK	Singularité 10
UC11	Entering PIN; Timeout	Singularité 11	OK	OK	NR	NR
UC12	Selecting Get Cash; Timeout	OK	Singularité 12	OK	NR	NR
UC13	Entering Amount; Timeout	OK	OK	Singularité 13	NR	NR
UC14	Confirming Amount; Timeout	OK	OK	Singularité 14	NR	NR
UC15	Receipt Ejected, Timeout	OK	OK	OK	Singularité 15	NR
UC16	Printing Receipt; Error Print Receipt	OK	OK	OK	Singularité 16	NR
UC17	Ejecting Card; Error Eject Card	OK	OK	OK	OK	Singularité 17
UC18	Identifying Client; Timeout	Singularité 18	NR	NR	NR	NR
UC19	Amount Entered; Cash not available	OK	OK	Singularité 19	NR	NR
UC20	Amount Entered; Cash not ejected	OK	OK	OK	Singularité 20	NR

Figure 6.28 - Exemple de definition des cas d'utilisation singulier.

Une liste des cas d'identification singuliers identifiés pour 'Retrait d'Argent' est montré dans la figure 6.29.

Id	Nom du Cas d'Utilisation Singulier
UC0	Cas Normal
UC1	Inserting Card; System Off
UC2	Inserting Card; System in Use
UC3	Inserted Card; Invalid Card
UC4	Entered PIN; Incorrect PIN
UC5	Selecting Get Cash; Incorrect Option
UC6	Amount Entered; Invalid Amount
UC7	Amount Entered; Beyond Card Limit
UC8	Amount Entered; Beyond ATM Limit
UC9	Cash Ejected; Timeout
UC10	Card Ejected; Timeout
UC11	Entering PIN; Timeout
UC12	Selecting Get Cash; Timeout
UC13	Entering Amount; Timeout
UC14	Confirming Amount; Timeout
UC15	Receipt Ejected; Timeout
UC16	Printing Receipt; Error Print Receipt
UC17	Ejecting Card; Error Eject Card
UC18	Identifying Client; Timeout
UC19	Amount Entered;Cash not available
UC20	Amount Entered;Cash not ejected

Figure 6.29 - Liste de Cas d'Utilisation Singuliers identifiés

Ensuite, chaque cas d'utilisation singulier est décrit de façon raffinée selon le format grammatical d'un Cas d'utilisation. Un exemple - le cas d'utilisation singulier UC5 'Selecting Get Cash;Incorrect Option' - a déjà été montré dans la figure 6.14.

22.2.3Construction des Cas d'Utilisation Operationnels

La construction des cas d'utilisation opérationnels à partir des cas d'utilisation singuliers n'est pas un processus mécanique (qui ne peut donc être automatisé) mais clairement un processus créatif, où l'analyste doit identifier les rôles des agents et décider l'allocation des responsabilités de chaque objet pour fournir les services proposés. Cette allocation est guidée par les heuristiques présentées dans la section 3.1.5 et bien sur par l'expérience de l'analyste en matière d'Analyse Orientée Objet. En pratique, la construction des cas d'utilisation opérationnels consiste à remplir les tableaux Scripts, Objets/Rôles, Alias, Services, Attributs, Etat de l'Objet.

22.2.4Construction des Cas d'Utilisation Concrets

La construction des cas d'utilisation concrets est faite directement à partir de cas d'utilisation Singuliers : chaque action d'un épisode d'un cas d'utilisation choisi est 'instanciée' par la détermination de ses caractéristiques concrètes qui serviront à l'expérimentation avec l'utilisateur. Il s'agit ici de construire des prototypes décrivant plus ou moins précisément les présentations du future système et des narrations de situations correspondantes à ces presentations, comme montré dans la figure 6.22 dans la section 3.1.6.

22.3Expérimentation avec les Cas d'Utilisations

L'expérimentation avec des cas d'utilisation concrets est faite par l'exécution du cas d'utilisation concret. Exécuter un cas d'utilisation concret consiste à simuler l'interaction en suivant le séquencement décrit par le cas d'utilisation, et en utilisant les dessins des écrans et les prototypes développés. Evidemment, les fonctions non implementées du système qui n'existent pas encore sont également simulées.

Ceci a pour but de :

- permettre au concepteur de voir la réaction de l'utilisateur et de travailler sur plusieurs ensembles de détails à la fois; il est important de reconnaître qu'un système peut être bon en théorie mais mauvais en pratique à cause de détails, même infimes;

- permettre aux utilisateurs de voir ce que sera le système final et faire des critiques;

Cette simulation permet à l'utilisateur d'évaluer à la fois la fonctionnalité et l'utilisabilité du système. Les critiques des utilisateurs demandent des modifications des cas d'utilisation, détaillées dans le paragraphe suivant. Nous avons quelques recommandations à faire pour les sessions d'expérimentation avec l'utilisateur :

- **Utiliser le cas d'utilisation concret dans le contexte d'utilisation réel** ! Ainsi, l'utilisateur peut plus facilement associer le cas à expérimenter à son contexte de travail;

- **Utiliser un cas d'utilisation à chaque fois** ! L'expérimentation doit utiliser un cas d'utilisation précis à chaque session pour permettre l'assimilation de la situation expérimentée et ne pas créer des confusions;

- **Expliquer simplement les alternatives possibles** ! Si l'utilisateur le demande, on peut expliquer les autres alternatives du comportement du système et les critères adoptés pour le choix. Cependant, ces explications doivent être simples et uniquement présentées en termes d'aide aux tâches de l'utilisateur;

- **Laisser l'utilisateur parler**! Les réactions et critiques de l'utilisateur doivent être considérées comme une opportunité de recueillir plus de besoins et de facteurs de qualité pour le cas d'utilisation; pour cela, il faut le laisser parler et ne pas commencer à défendre ses idées.

22.4Modification des Cas d'Utilisation

Les critiques des utilisateurs sont à l'origine des modifications des cas d'utilisation. Ces modifications peuvent être petites et génèrer des variantes du cas d'utilisation concret en changeant uniquement ses caractéristiques concrètes; ou bien être grandes et provoquer des changements au niveau opérationnel, singulier ou même essentiel du cas d'utilisation.
Il y a deux procédures pour la réalisation de ces modifications des Cas d'utilisations :

- Procédure Descendante (*top-down*) : à partir des modifications réalisées à un niveau, faire les changements correspondants dans les niveaux plus inférieurs;

- Procédure Ascendante (*bottom-up*) : à partir des expérimentations avec les cas d'utilisations concrets, modifier les niveaux plus abstraits.

Pour les deux procédures, les informations de traçabilité entre les niveaux d'abstraction des cas d'utilisation permettent de savoir quels éléments changer et comment remonter à l'origine du problème.

22.5Intégration des plusieurs Cas d'Utilisation

Jusqu'à maintenant chaque cas d'utilisation est utilisé comme un soutien pour la détermination des besoins additionnels du système dans la perspective d'une situation spécifique d'usage, donc dissocié des autres cas d'utilisation. L'unification consiste à intégrer les cas pour un acteur spécifique dans un Modèle UNifié D'utilisatiOn (MUNDO). Un MUNDO est crée par l'identification des épisodes communs entre les cas d'utilisation, par l'attribution de nouveaux numéros et par leur intégration. Ainsi, un MUNDO est le point de vue d'utilisation total pour un acteur du système.

Dans l'exemple ATM, l'épisode Identification/S'Identifier est le même pour les Cas d'Utilisation Essentiels Retrait d'Argent, Demande de Solde, et Demande de Relevé d'Opérations.

La liste d'épisodes avant l'intégration est la suivante :

Cas 'Utilisation Essentiel 'Retrait d'Argent' :
- Episode 1 : S'Identifier
- Episode 2 : Demander Retrait
- Episode 3 : Choisir Valeur
- Episode 4 : Prendre les Choses
- Episode 5 : S'en aller

Cas 'Utilisation Essentiel 'Demande de Solde' :
- Episode 1 : S'Identifier
- Episode 2 : Demander Solde
- Episode 3 : Prendre les Choses
- Episode 5 : S'en aller

Cas 'Utilisation Essentiel 'Demande de Relève' :
- Episode 1 : S'Identifier
- Episode 2 : Demander Relevé
- Episode 3 : Prendre les Choses
- Episode 5 : S'en aller

Les épisodes sont globalement renumerotés et le système ATM est maintenant composé des épisodes suivants :

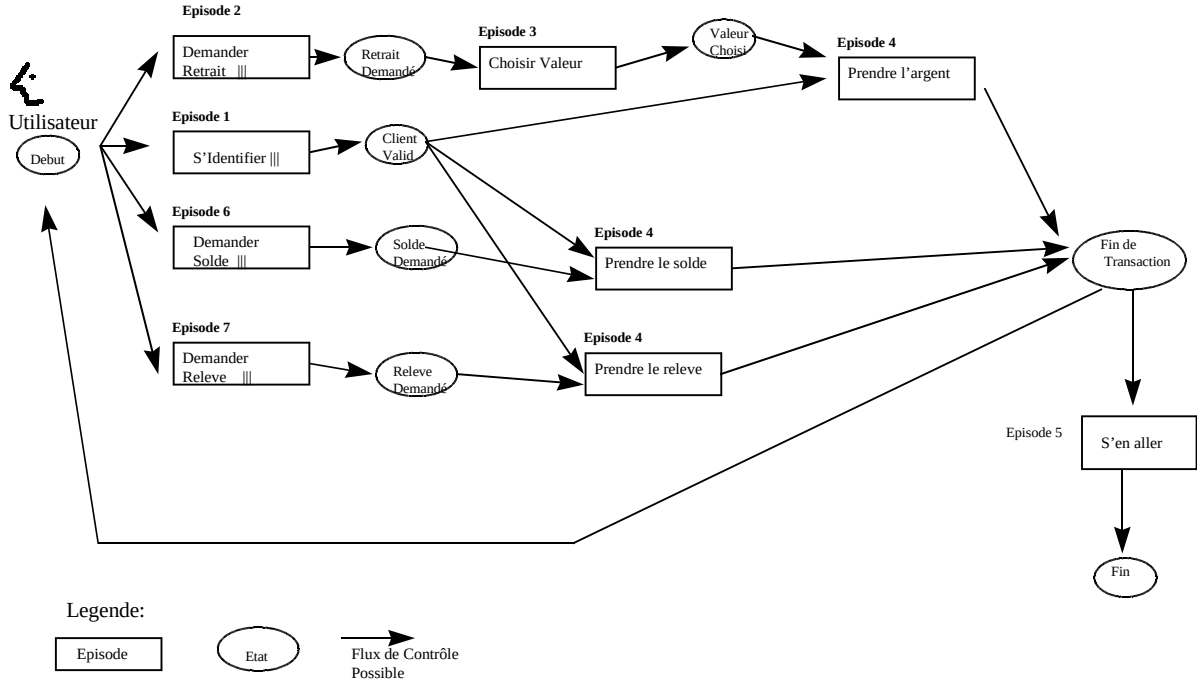
- Episode 1 : S'Identifier
- Episode 2 : Demander Retrait
- Episode 3 : Choisir Valeur
- Episode 4 : Prendre les Choses
- Episode 5 : S'en aller
- Episode 6 : Demander Solde
- Episode 7 : Demander Relevé

Les cas d'utilisations sont (cf. Figure 6.30) ainsi configurés :

Cas 'Utilisation Essentiel 'Retrait d'Argent' :
Episode 1 : S'Identifier
Episode 2 : Demander Retrait
Episode 3 : Choisir Valeur
Episode 4 : Prendre les Choses
Episode 5 : S'en aller
Cas 'Utilisation Essentiel 'Demande de Solde' :
Episode 1 : S'Identifier
Episode 6 : Demander Solde
Episode 4 : Prendre les Choses
Episode 5 : S'en aller
Cas 'Utilisation Essentiel 'Demande de Relevé :
Episode 1 : S'Identifier
Episode 7 : Demander Relevé
Episode 4 : Prendre les Choses
Episode 5 : S'en aller

Le MUNDO pour l'utilisateur de l'ATM est montré dans le schéma suivant. Ce schéma montre tous les épisodes des cas d'utilisation de l'ATM, mais par souci de clarté il ne montre qu'un extrait des états intermédiaires dans la réalisation des cas et des flux de contrôle (séquences) possibles. Evidemment, ces informations peuvent aussi être décrites selon le format grammatical.

71)
72)



73)

Figure 6.30 - Intégration de plusieurs cas d'utilisation

23.L'Intégration des besoins de TAREFA

L'intégration des besoins a pour but d'organiser les besoins recueillis ou identifiés par le processus de détermination décrit ci-dessus. Pour cette organisation, TAREFA propose un modèle de besoins qui est un reflet des différentes informations prises en compte pendant tout le processus et qui suit la philosophie 'tous les besoins doivent être

enregistrés’.

23.1Le modèle des besoins du système

Le modèle des besoins de TAREFA est une structure pour décrire les besoins du système, en suivant le concept de besoins comme un prédicat sur les buts à atteindre par le système.

Le modèle des besoins de TAREFA est structuré également autour du concept des cas d’utilisation. A chaque cas d’utilisation singulier, sont associés :

- les besoins fonctionnels du système identifiés, c’est à dire, les besoins sur les objets et leurs propriétés (composant statique) et sur les services et fonctions (composant dynamique) du système;

- les besoins de l’interaction, c’est à dire, les besoins sur le comportement souhaité du système en interaction avec l’utilisateur;

- les besoins de l’environnement et les facteurs de qualité associés au système, c’est à dire, les besoins non fonctionnels qui serviront aussi pour le choix entre les alternatives de conception du système.

Chaque besoin est décrit par l’ensemble des attributs décrits dans la figure 4.2 au chapitre IV, ce qui fourni l’information de configuration (version, date), l’information de traçabilité (affecté par, fait référence à, utilisé par) et pré-traçabilité (contributions et origine) et le degré d’importance (priorité). Un exemple est montré dans la figure 6.31.

Id Unique	S1
Nom	Accepter Identification
Type	Besoins Fonctionnel Dynamique (BFD)
Source	BUI (fonction receptive à une formulation)
Contributions	non applicable
Description Informelle	Recevoir les Informations pour l’Identification du Client
Description Formelle	non applicable
Version - date	V.1, 20/1/97
Affecté par besoins	<liste de ids>
Fait reference à :	Code Banque, Code Agence, Code Compte, Mot-de-Passe
Utilisé par :	<liste de Ids>
Priorité	**desirable**

Figure 6.31 - Description des attributs des besoins

Evidemment, il y a des besoins qui sont associés à plusieurs cas d’utilisation.Un cas d’utilisation peut aussi être associé à divers besoins, parfois en conflit les uns avec les autres. La résolution de ce conflit est un processus social (comme toutes les décisions des acteurs). Donc les acteurs doivent négocier, examiner les alternatives, faire des compromis et choisir les besoins à accomplir. Ces besoins sont structurés par l’analyste mais ils doivent être validés par les acteurs.

A chaque épisode (ou même à chaque service) d’un cas d’utilisation singulier, on peut :

- associer des facteurs de qualité qui peuvent être :

a) des contraintes pour (une décision de) la conception
b) des critères pour l'évaluation de épisode

- associer des besoins qui peuvent être :

c) des contraintes pour (une décision de) la conception

d) des composants, qui sont des ressources pour le développement et qui sont des besoins fonctionnels dynamiques (en général des services du système et des actions de l'utilisateur) ou statiques (en général des informations).

Ces associations sont illustrés sur la figure 6.32.

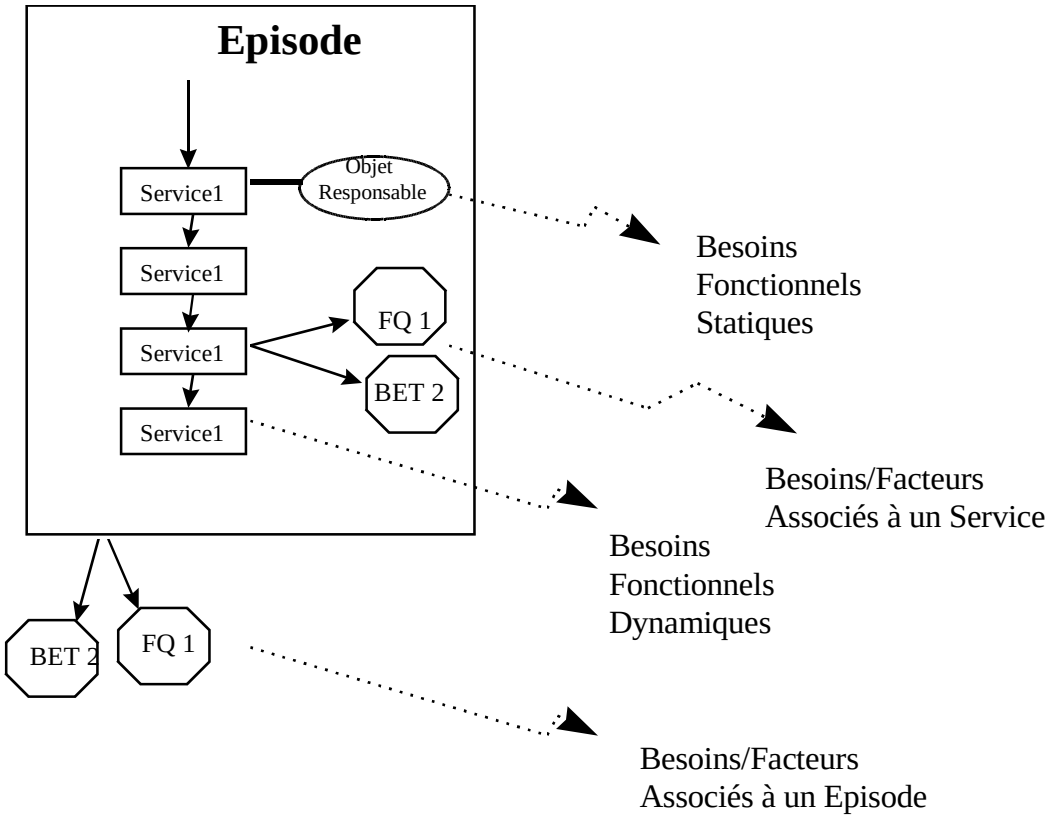


Figure 6.32 - Associations des besoins et facteurs aux épisodes des cas d'utilisation

Un extrait du modèle des besoins du cas d'utilisation singulier UC0 - 'Cas Normal' est montré ensuite dans la figure 6.33. Les codes utilisés font référence à la typologie des besoins et facteurs de qualité proposée dans le chapitre 4, à savoir :

Codes des Besoins de l'ATM

Besoins Fonctionnels du Système (BFS)	**Besoins d'Interaction -(BDI)**	**Besoins de l'Environnement**
Dynamiques (BFSD) : Services S1 à S19 Statiques (BFSS) : Objets et Attributs	Cas d'utilisation UC0 à UC20	socio-politiques (BESP) économiques (BESE) du cycle de vie (BECV) chronologique (BEC) physiques (BEPY) physiologique (BEPS) technologiques (BET) : les ressources du modèle de plateforme M1 à M7

Les codes des Facteurs :

Facteurs de Qualité du système (FQS)	**Facteurs de Qualité de l'interaction (FQI)**	**Facteurs de Qualité du processus de développement du système (FQP) :**
Fiabilité (FQS1) Integrité (FQS2) Securité (FQS3) Exactitude (FQS4) Efficacité (FQS5) etc.	Facilité d'apprentissage, y compris : • Prévisibilité (FQI1) • Synthèse (FQI2) • Familiarité (FQI3) • Généralisabilité (FQI4)	Relatifs à la Construction du système, y compris : • évolutibilité (FQP1) • compréhensibilité (FQP2) • etc
	Flexibilité d'interaction, y compris : • Initiative de dialogue (FQI5) • Dialogue à fils multiples (FQI6) • Migration des tâches (FQI7) • Substituabilité (FQI8) • Multimodalité (FQI9) • Configurabilité (FQI10)	Relatifs à la révision du système, y compris : • testabilité (FQP3) • flexibilité (FQP4) • etc..
	Robustesse d'interaction, y compris : • Observabilité (FQI11) • Recuperabilité (FQI12) • Conformité à la Tâche (FQI13)	Relatifs à la Transition du système, y compris : • portabilité (FQP5) • interoperabilité (FQP6) • etc

La figure 6.33 montre les besoins et facteurs de qualité associés à l'épisode 1 du cas d'utilisation singulier UC0 - 'Cas Normal' après l 'expérimentation avec le cas d'utilisation concret. Evidemment ces besoins et facteurs ne sont pas universels : des personnes différentes, des organisations différentes peuvent déterminer un autre ensemble. Ces besoins vont servir comme prédicats pour le processus de choix d'alternatives. Par brieveté, dans cet exemple, la plupart des besoins de l'environnement, des facteurs de qualité du système et des facteurs de qualité du processus ne sont pas pris en compte. Nous nous concentrons sur les besoins fonctionnels, les besoins de l'interaction et les facteurs de qualité de l'interaction pour démontrer les caractéristiques du modèle.

Par exemple, l'épisode 1 - S'Identifier est associé aux besoins fonctionnels dynamiques (BFD) suivants : les services S1 - Accepter Id, S2 - Vérifier Id du Client, S3 - Demander Mot-de-Passe, S4 - Accepter Mot-de-Passe et S5 - Vérifier Mot-de-Passe, cf. tableau Services du niveau opérationnel; et aux besoins fonctionnels statiques (BFS) suivants : les Objets du domaine du problème Compte et Carte, et les attributs code_banque, code_agence, code_compte et mot-de-passe. Ces services et informations sont des composants du cas d'utilisation et doivent être présents dans n'importe quelle réalisation d'un système qui supporte ce cas. Des exemples des contraintes à ce cas d'utilisation sont les dispositifs périphériques de l'ATM qui font partie du modèle de plateforme, à savoir M1 - Carte Bancaire, M2 - le clavier alphanumérique, M3 - l'écran et M6 - le lecteur de carte.

Des exemples de critères d'évaluation sont :

- les facteurs de qualité du système FQS1 - Fiabilité et FQS3 - Sécurité, dont l'importance est montré par les Besoins imposés par l'Organisation BUO#1 - ATM_transactionnelle du type 'tout ou rien', BUO#2 - utilisation de carte et BUO#9 - usage privée des informations;

- Les facteurs de qualité de l'interaction FQI3 - Familiarité, FQI4 - Generalisabilité, FQI9 - Multimodalité, FQI5 - Initiative de Dialogue, FQI12 - Recuperabilité, et FQI13 - Conformité à la tâche, dont l'importance est due à l'utilisation de l'ATM par le grand public;

- Les facteurs de qualité du processus FQP1 - evolutibilité et FQP3 - testabilité,, parce que l'insertion future d'autres opérations bancaires est très probable.

Pour les autres épisodes, le raisonnement est similaire.

Cas d'Utilisation Singulier	Episodes	Besoins Fonctionnels Composants (ressources)	Contraintes	Critères d'Evaluation
UC0	1 (S'Identifier)	BFD :S1, S2;S3,S4,S5 BFS : Compte, Carte, code_banque, code_agence, code_compte, mot-de-passe	BET : M1, M6, M2, M3	FQS3, FQS1; FQI3, FQI4; FQI5, FQI9; FQI12, FQI13; FQP1, FQP3

Figure 6.33 - Besoins de l'épisode 1 du cas d'utilisation singulier UC0 - 'Cas Normal'

24.Modèle de Design Rationale

Le *Design Rationale* est fréquemment utilisé comme un outil pour structurer et documenter le raisonnement des concepteurs [Buckingham-Shum 95]. La littérature présente plusieurs modèles de *design rationale*, entre autres IBIS (Issue-Based Information Systems [Yakemovic & Conklin 90/) et QOC (Questions, Options, Critères [MacLean et al. 91/). On peut voir d'autres modèles dans le livre [Moran & Carroll 94].

Sans doute le modèle le plus utilisé est le modèle QOC. QOC est une notation semi-formelle des différentes options de conception et de représentation explicite des raisons qui permettent de choisir entre ces options. Les principaux concepts de QOC sont :

- **Q**uestions, indiquant les principaux sujets de la conception;

- **O**ptions, qui sont des possibles réponses aux questions;

- **C**ritères, qui sont les raisons pour (critères positifs) ou contre (critères négatifs) le choix des options possibles.

Les QOCs sont identifiés usuellement par l'intermédiaire d'enregistrement des sessions de conception. Les créateurs de l'approche QOC soutiennent que le *design rationale* est une structure pour résoudre des problèmes de **documentation des décisions déjà prises**. Cependant, comme la détermination des besoins et de la solution sont souvent intercalés, il peut être intéressant d'utiliser le *design rationale* comme une technique pour organiser les alternatives de solutions.

24.1L’utilisation de QOC en TAREFA

Dans cette thèse, nous adoptons QOC comme un **guidage pour prendre les décisions**, donc comme un soutien méthodologique plutôt que documentationnel. Il faut noter que cette utilisation n’est pas nouvelle. [Suttcliffe DIS 95] a utilisé les techniques de *design rationale* pour capturer les préférences et les choix des utilisateurs durant l’analyse des besoins. [Bellamine 96] a également utilisé une variante de QOC pour justifier ses choix de conception.

Nous avons choisi la notation QOC (Questions, Options, Critères) en l’utilisant de la manière suivante :

- Questions, où chaque question associée à une action ou épisode d’un cas d’utilisation à soutenir est pris en compte; l’action dénote autant une action de l’utilisateur qu'un service du système;

- Options, qui est l’énumération des alternatives pouvant soutenir la réalisation de cette action;

- Critères, où les différents besoins/facteurs de qualité sont associés aux options, pour permettre de réfléchir et de décider de l’alternative à choisir. Les critères positifs sont en général les ressources disponibles pour une option; les critères négatifs sont en général les contraintes associées à une option. En fait, il faut trouver un compromis entre des besoins/facteurs aussi divers et souvent contradictoires que la flexibilité et l’intégrité, l’efficacité et la portabilité, en prenant compte les caractéristiques des utilisateurs, les activités à soutenir et les contraintes pour le développement.

74)

Cette structure est organisée par l’analyste pour illustrer son raisonnement et elle est ensuite montrée délibérément à l’utilisateur pour l’encourager à participer au processus de décision et déclencher des discussions sur les alternatives et les critères de choix. L’adoption des besoins et facteurs comme critères est un composant fondamental de cette discussion.

Dans TAREFA, nous avons décidé de construire un schéma QOC pour chaque service du système dans les épisodes de cas d’utilisations Singuliers. Un exemple de l’utilisation de QOC dans TAREFA est ensuite montré.

24.2L’utilisation de QOC pour l’exemple ATM

Ici, nous montrons un extrait d’un schéma QOC pour le cas d’utilisation ‘Retrait d’argent’. Les services considérés ne sont que les services dont il y a plus d’une option de réalisation. Pour les autres services dont l’option de réalisation est unique - à savoir les services S2,S3,S5,S8,S14,S17,S19 (décrits dans le tableau Services du niveau opérationnel) - ce schéma ne sera pas développé.

Les épisodes considérés sont :

- les épisodes dont le cours d’exécution est normal; pour l’exemple de l’ATM, il y a 5 épisodes, à savoir :

a) Episode 1 -S’Identifier
b) Episode 2 - Demande de Retrait
c) Episode 3 - Choisir Valeur;

d) Episode 4 - Prendre les choses
e) Episode 5 - S'en aller

- les épisodes dans lesquels il y a l'occurrence des singularités; pour le cas d'utilisation 'Retrait d'argent' 20 singularités ont été identifiées;

Dans cet exemple, les options ont été proposées pendant l'expérimentation avec les cas d'utilisation concrets. Les critères positifs sont précédés par le signe (+) et les critères négatifs par le signe (-). Même si en général la notation du schéma QOC est spatiale, nous préférons utiliser une notation tabulaire .

La figure 6.34 montre un exemple de l'utilisation de QOC pour le choix des alternatives de spécification, qui ont été parfois testées comme cas d'utilisation concrets. Plusieurs critères sont en commun, on peut cependant remarquer que les critères différents sont très importants. Par exemple, dans l'option 1 un client sans la carte (M1 dans le modèle de plataforme de l'ATM) ne peut pas utiliser l'ATM, même s'il connaît par coeur les codes de la banque, de l'agence et de son compte. Cela implique que le facteur de qualité de l'interaction FQI9 - multimodalité n'est pas pris en compte pour ce service. Dans l'option 2, au contraire, le client peut utiliser le clavier mais cela ne satisfait pas le besoin de l'organisation qui demandait l'utilisation de la carte (BUO#2) et les besoins BUE#3 et BUO#3 - qui demandent la possibilité de minimiser le volume d'information à taper.

Une troisième option est la combinaison des options antérieures, en laissant à l'utilisateur le choix entre utiliser la carte ou taper les informations, ce qui permet en fait de satisfaire le critère de multimodalité (FQI9), dans ce cas du type synergique.

Questions	Options	Critères
Accepter Identification (S1)	**Option 1** : Client insère la Carte;	(+) BUO #2 (+) BUO #3 (+) BUO#5 (+) BET : M1, M2, M6 (+)FQS3, FQS1; (+)FQI3, FQI4; (+)FQI5; (+)FQI12, FQI13; (+)FQP1, FQP3 (-) M1 (Client sans carte) (-) FQI9
	Option 2 : - Le client tape toutes les Informations, il n'y a pas de carte	(+) BUO#4 (+) BET : M2 (+)FQS3, FQS1; (+)FQI3, FQI4; (+)FQI5, (+)FQI12, FQI13; (+)FQP1, FQP3,; (+) BEE - Il y a déjà le clavier disponible, pas de coût additionnel (-) BUE#3 (-) BUO#3 (-) FQI9
	Option 3 : Le client peut choisir entre utiliser la carte ou taper les informations	(+) BUO#4 (+) BET : M2 (+)FQS3, FQS1; (+)FQI3, FQI4; (+)FQI5, FQI9 (en permettant à l'utilisateur d'utiliser 2 manières différentes d'entrer des informations); (+)FQI12, FQI13; (+)FQP1, FQP3,; (+) BEE - Il y a déjà le clavier disponible, pas de coût additionnel

Figure 6.34 - Exemple de schéma QOC appliqué au Cas d'utilisation Singulier 0, Episode 1; Service S1

La construction de schémas QOC pour les autres décisions sur les services du système suit les mêmes principes.

25.Des Besoins à la Spécification Conceptuelle

Les besoins déterminés et modélisés par TAREFA, ainsi que tous les modèles développés, peuvent être utilisés pendant tout la conception du logiciel. Par exemple, les cas d'utilisation peuvent être utilisés comme cas de test pour l'évaluation du système. Mais sans doute, l'utilisation la plus importante est leur utilisation pour la construction de la spécification conceptuelle du système.

Evidemment, comme TAREFA a été proposé pour une intégration avec la méthode Objectory, le passage de TAREFA à Objectory est très naturel, voir immédiat. Ce passage est décrit dans la section 6.1.

Néanmoins l'orientation cas d'utilisation de TAREFA n'empêche pas l'utilisation du modèle des besoins par d'autres méthodes de spécification orientées objet. Par exemple, la construction d'une spécification conceptuelle avec la méthode DIANE+ est montrée dans la section 6.2.

25.1De TAREFA à Objectory

Comme nous l'avons vu, le niveau opérationnel d'un cas d'utilisation permet de décrire les informations qui sont utiles à la conception orientée objet, en particulier en utilisant la méthode Objectory. De la description des cas d'utilisation au niveau opérationnel de TAREFA, on peut dériver directement une spécification du modèle d'analyse d'Objectory, où une architecture préliminaire du système est proposée par la détermination des propriétés et services des objets de trois types : Interface (IOB), Contrôle (COB) et Entité (EOB). A partir des informations présentes dans les tableaux, on peut construire les diagrammes de la méthode Objectory, qui font partie du modèle d'Analyse. La procédure est simple et est décrite ci-dessous.

Pour chaque Script qui représente un cas d'utilisation au niveau opérationnel, il est généré un diagramme d'objets (*object diagram*) d'Objectory, en montrant les objets participants et leurs interdépendances - la classification des objets participants selon les 3 types (IOB,EOB et COB) est explicitement montré dans ce diagramme. Les informations des tableaux Objets/Rôles, Alias et Attributs sont utilisés pour définir les propriétés initiales de ces objets; un exemple de diagramme d'objets pour le script 'Retrait d'Argent' - Cas Normal est montré sur la figure 6.35, qui montre le diagramme d'objets pour le script 'Retrait d'Argent'. Un diagramme d'objets montre les objets qui réalisent les services du cas d'utilisation et leurs interdépendances. Une interdépendance entre deux objets implique qu'un objet sera client, serveur ou client/serveur (indiqué par le sens des flèches) par rapport à l'autre pour la réalisation d'un service. La notation graphique utilisée dans la figure est la notation proposée par [Jacobson 92]

75)

76)

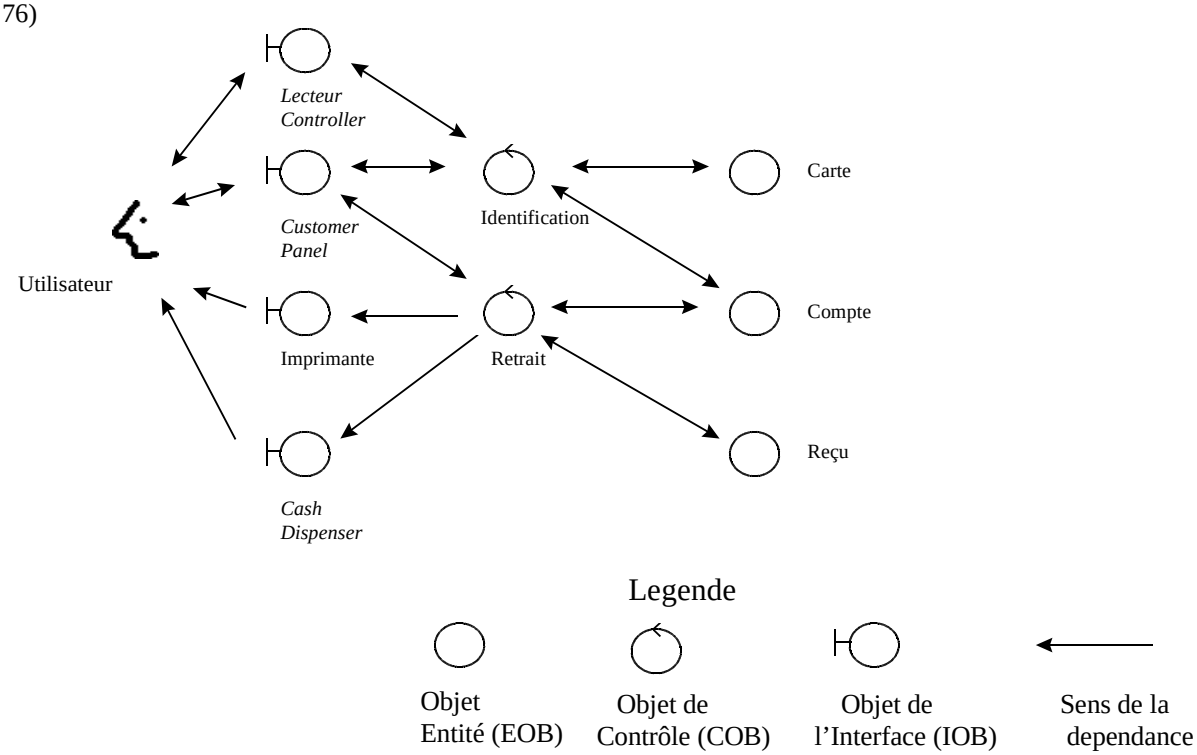


Figure 6.35 - Diagramme d'Objets pour le script 'Retrait d'Argent' - Cas Normal

Ce modèle d'Analyse montre une architecture préliminaire de la partie du système responsable pour la transaction 'Retrait d'argent'. Cette architecture est composée de 9 objets, dont :

- 4 objets de l'interface (IOB) - qui réalisent toutes les interactions utilisateur-système et encapsulent toutes les manipulations des périphériques comme le lecteur de carte (*Lecteur controller,* l'imprimante et *le cash dispenser*);

- 2 objets de contrôle (COB) - qui contrôlent le flux des transactions; et

- 3 objets entités (EOB) - qui servent à stocker les informations soit de façon permanente (comme les informations de **Compte)** soit de façon temporaire (comme les informations de **Reçu** et **Carte**).

Ce processus de correspondance consiste à élaborer des architectures pour chaque transaction. Ces architectures servent de base à la conception de la méthode Objectory, responsable du raffinement des objets, de leurs propriétés et des méthodes et de l'adaptation de ces architecture aux caractéristiques spécifiques de l'environnement de programmation (le langage de programmation, la configuration du matériel, etc).

25.2De TAREFA à DIANE+

DIANE+ est une méthode de spécification conceptuelle de systèmes interactifs, basée sur la définition des interactions (opérations interactives), des actions du système (opérations automatiques) et des actions des utilisateurs (opérations manuelles) ainsi que de leur précédence. Cette définition est utilisée pour construire un modèle de dialogue du système et son interface avec le composant sémantique, toujours en suivant des paramètres ergonomiques pour les décisions des aspects entre autres la précédence, le déclenchement d'opérations, l'obligation de l'exécution d'une opération. Une spécification conceptuelle d'une application interactive en suivant DIANE+ est totalement orientée par ce modèle de dialogue, donc par les interactions utilsateur-système. Cette orientation 'interactions' est une caractéristique fondamentale des cas d'utilisation, c'est pour cela que nous proposons une correspondance entre les cas d'utilisation et DIANE+.

Une spécification DIANE+ peut être dérivée de la description des cas d'utilisation au niveau téléologique de TAREFA, plus particulièrement du cas d'utilisation singulier qui décrit le 'cas normal'. Ce choix précis est dû à deux raisons principales :

1. le cas singulier 'normal' est la description la plus raffinée des actions haut-niveau de l'utilisateur et du système qui permettent une vue conceptuelle des interactions; en fait, le niveau essentiel est trop abstrait pour servir de base à une spécification et le niveau opérationnel contient beaucoup de détails orientés conception qui ne sont pas adéquats à une spécification conceptuelle;

2. le cas singulier 'normal' est décrit sans prendre en compte les singularités; DIANE n'a pas de concepts pour spécifier l'occurrence ou le traitement de singularités, parce la méthode considère que ces aspects seront incorporés au moment de la conception détaillée.

La dérivation suit une correspondance sémantique entre les composants d'un cas d'utilisation singulier de TAREFA et les composants d'un modèle DIANE+. Cette correspondance est montrée dans la figure 6.36 et peut être résumé ainsi :

- Les notions d'**action** et d'**initiateur** du cas d'utilisation correspondent respectivement aux notions d'**opération** et de **déclencheur** de DIANE+;

- Les épisodes du cas d'utilisation singulier seront les procédures interactives composées - qui peuvent être décomposées en procédures primitives - chez DIANE+;

- Les actions interactives du cas d'utilisation singulier seront les opérations interactives primitives chez DIANE;

- Les services du système seront les opérations automatiques chez DIANE+;

- Les cas d'utilisation n'ont pas la notion d'action manuelle, donc il n'y a pas de correspondance pour les opérations manuelles de DIANE+;

- Tous les services du cas d'utilisation normal sont obligatoires; si la spécification prenait en compte les services défensifs et correctifs, ils seraient considérés comme facultatifs;

- Les précédences permanente ou indicatives de DIANE+ indique l'adoption d'un ordonnancement à suivre, ce qui correspond aux alternatives d'ordonnancement associés aux pré conditions et post-conditions de chaque action du cas d'utilisation.

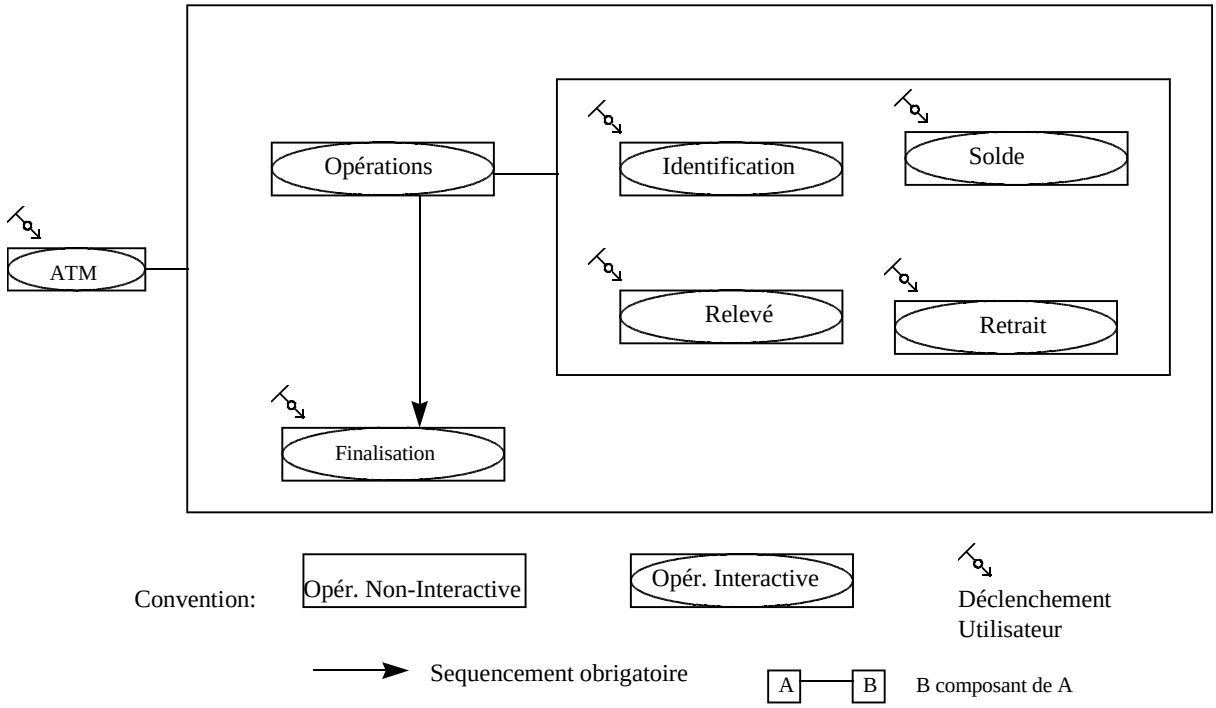
Evidemment la traduction d'un cas d'utilisation de TAREFA à un modèle DIANE+ n'est pas un processus mécanique mais un travail créatif du concepteur guidé par cette correspondance.

Concepts du cas d'utilisation singulier	**Concepts DIANE+**
Episode	Opération interactive composée
Action	Opération interactive primitive
Service du système Initiateur = système ou un agent du système (typiquement un objet)	Opération automatique obligatoire
	Opération manuelle
Services correctifs et défensifs	Opération facultative
Initiateur = utilisateur	Déclenchement par l'utilisateur
Relations entre les pre-conditions et les post-conditions des actions	Précédence permanente ou indicative entre opérations

Figure 6.36 - Correspondances entre les composants des cas d'utilisation de TAREFA et les composants DIANE+

Ainsi, le cas d'utilisation singulier 'cas normal' (UC0) de 'Retrait d'Argent' décrit en TAREFA peut être traduit en DIANE+, en générant la spécification présentée sur les figures 6.37 et 6.38. Par soucis de clarté, les détails des contraintes sur les sous opérations et de contraintes sur le nombre de déclenchements ne sont pas explicités.
La figure 6.37 montre le modèle conceptuel de l'ATM. L'ATM est lui-même une opération interactive composée par les opérations (y compris les transactions d'identification, de retrait d'argent, de solde et de relevé) et par la finalisation, correspondant à l'épisode " S'en aller ".

77)

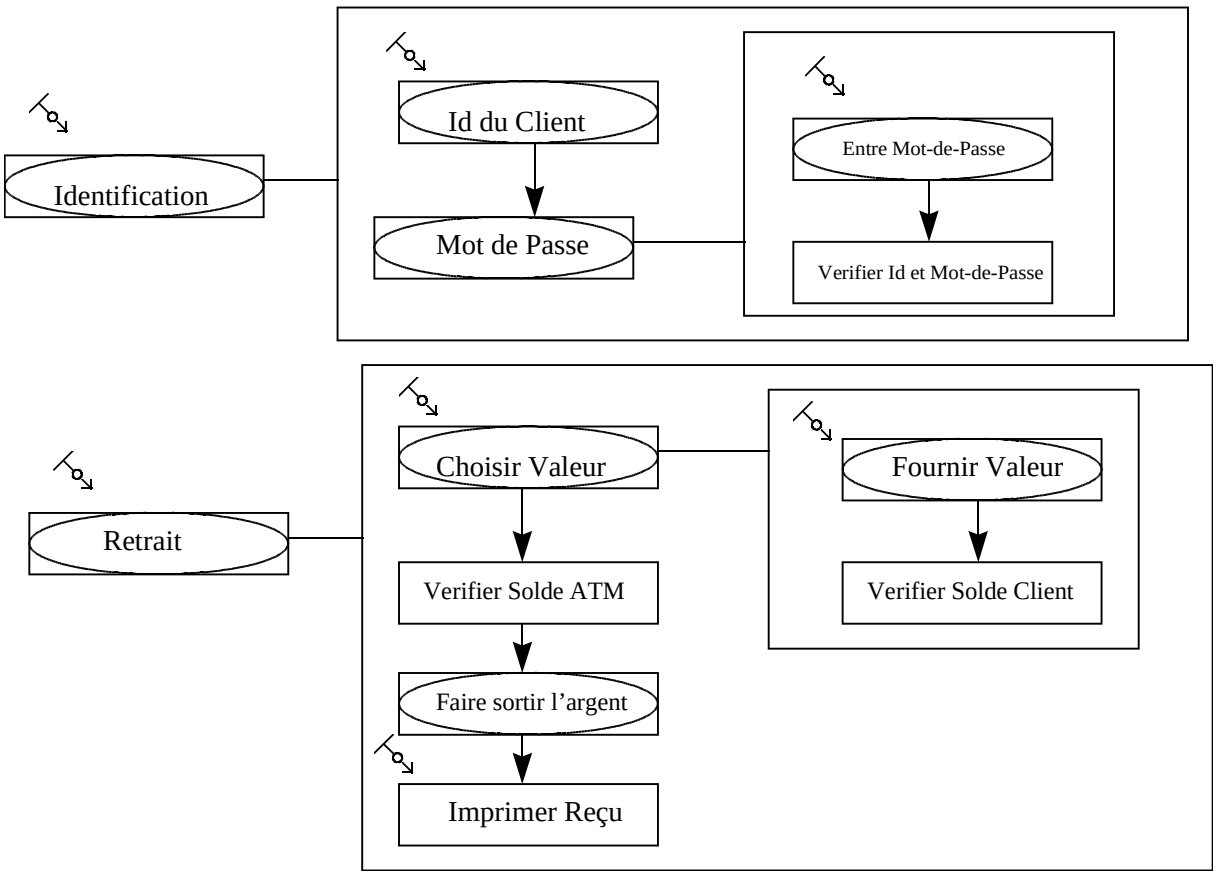


78)

Figure 6.37 - Modèle Conceptuel de l'ATM

La figure 6.38 montre un raffinement du modèle conceptuel de l'ATM, en décomposant les transactions 'Identification' et 'Retrait d'argent'. Ce raffinement montre par exemple que l'utilisateur déclenche l'identification en fournissant son 'Id du client' mais que son 'Mot-de-passe' est demandé automatiquement par le système, pour après être vérifié par une opération du système. Les détails de chaque opération, comme les conditions et les paramètres ne sont pas spécifiés par DIANE+.

79)


Identification
Id du Client
Mot de Passe
Entre Mot-de-Passe
Verifier Id et Mot-de-Passe
Retrait
Choisir Valeur
Verifier Solde ATM
Faire sortir l'argent
Imprimer Reçu
Fournir Valeur
Verifier Solde Client

80)

Figure 6.38 - Modèle Conceptuel de l'ATM - extrait du 2eme niveau Chapitre VII - TAREFA: Le Modèle de Processus Emergent

---

26.Introduction

Des travaux récents en IB reconnaissent l'importance fondamentale de changer le point d'attention principale de la modélisation du système à la modélisation du processus de l'IB [Jarke 94] [Rolland 94]. La modélisation du système comprend les techniques de modélisation pour représenter notre compréhension des besoins de l'utilisateur et spécifier les composants et propriétés du système en termes abstraits de haut niveau, sans considérer les détails non pertinents. Un modèle du processus sert comme soutien aux activités de construction et de transformation des modèles du système, en enregistrant l'histoire du développement du système, son contexte et ses décisions. En fait, cette modélisation des connaissances du processus est très importante en IB pour :

- permettre le raisonnement sur les besoins dynamiques qui changent dans un environnement qui change aussi ;

- réutiliser les décisions (non seulement les spécifications) du processus d'IB dans d'autres projets;

- améliorer la manière de travailler de l'équipe multidisciplinaire impliquée dans le processus.

Le processus d'IB est défini dans un espace à trois dimensions (cf. Chapitre 2) : Représentation, Spécification et Accord (*agreement*). Il y a un évident progrès récemment pour fournir des réponses aux questions des deux premières dimensions (voir par exemple l'intérêt croissant sur les modèles formels). En contraste, très peu d'attention est dirigée vers la dimension d'accord.

Dans ce chapitre, un modèle de processus pour les activités collaboratives de l'IB basé sur la notion d'émergence est présenté. L'**Emergence** est un concept dérivé de la Science Cognitive, de la Théorie des Systèmes et de l'Ethnométhodologie et qui a un rôle important dans la théorie des IHM [Suchman 87]. Nous pensons que l'émergence est un concept adéquat pour fournir un soutien très riche à la dimension d'accord.
Ce travail est le résultat de la préoccupation à supporter les activités coopératives du processus d'IB. Une version antérieure de ce modèle émergent pour l'IB a été présentée dans [Cisse & Pimenta 96]. Ce modèle émergent a été aussi adopté dans les contextes d'ingénierie concurrente [Cisse 95] et CSCW [Cisse 96a] [Cisse 96b] et la description détaillé du métamodèle et d'un outil pour soutenir son utilisation est présentée dans [Cisse 97].

Ce chapitre a la structure suivante. D'abord dans la section 2, nous introduisons la modélisation de processus émergent pour représenter et comprendre les activités coopératives de l'IB, avec une comparaison avec d'autres modèles de processus de

développement de logiciel. La section 3 résume les fondements théoriques de ce modèle. Nous avons choisi de présenter ces fondements séparément de la partie I - l'état de l'art - parce qu'ils sont très spécifiques. Le modèle émergent est présenté plus précisément dans la section 4. Finalement, dans la section 5 une synthèse est présentée.

27.Une perspective émergente dans l'IB

Nous proposons un modèle de processus pour les activités collaboratives entre les acteurs de l'IB basé sur la notion d'émergence. Ces activités sont très difficiles à automatiser et à contrôler parce qu'elles impliquent la complexité et la nature dynamique du travail de groupe mais il possible de les soutenir par l'intermédiaire d'un modèle de processus. En effet, [Sommerville 94] reconnaît que plusieurs activités critiques du développement de logiciel ne peuvent pas être formalisées, comme par exemple :

- les activités imprévisibles détermines par le domaine du problème et par des facteurs organisationnels ; et

- les activités qui sont naturellement collaboratives, dont la dynamique est déterminée par les participants et l'état courant du contexte de l'organisation, qui nous intéressent plus particulièrement.

Les processus de collaborations réels de l'IB ne peuvent pas être considérés comme ayant une séquence rigide de procédures et de décisions suivant les pas d'une démarche centrée sur l'analyste parce leur dynamique est déterminée par le choix, la décision et la participation des acteurs. Une grande variété d'alternatives d'actions et d'interactions peut être combinée pour accomplir les buts du processus. Ces alternatives sont influencées par le contexte qui forme l'espace de ressources pour les stratégies acceptables. En fait, selon [Barghouti 95], la modélisation formelle de processus réels permet de détecter s'il y a d'inconsistances et d'inefficacités du processus. Il peut être difficile de convaincre les participants de changer le processus dont ils ont l'habitude. Cette remarque est particulièrement vraie pour les activités coopératives de l'IB où l'utilisateur n'accepte pas de suivre une séquence déterminée *a priori* pour discuter ses problèmes et décider de solutions qui vont l'affecter.

Notre modèle de processus émergent est utile comme support aux processus coopératifs parce qu'il fournit un espace d'objets partagés et un méta-modèle où les plans et les buts évoluent d'une façon consensuelle et émergente, en prenant compte à la fois les actions, les communications et la cognition du groupe de travail pendant l'interaction entre ses membres. Le point plus important du modèle est alors plutôt la flexibilité du processus (mais tout en prenant compte la traçabilité et la reutilisabilité des discussions et des décisions) que les aspects de contrôle et de guidage, caractéristiques des modèles traditionnels.

Les activités collaboratives de l'IB sont légitimement considérées comme un processus humain intensif déterminé par des conflits et de la coopération. Le développement conjoint d'idées par l'intermédiaire d'un regard multidisciplinaire et un support à la négociation est particulièrement important.

Un **regard multidisciplinaire** est la conséquence du fait que les acteurs ont différents points de vue sur le domaine du problème et différentes attentes par rapport au système futur, dus à leurs différents savoirs (*backgrounds*) et rôles dans l'organisation.

La **négociation** entre ces points de vue et attentes doit être explicitement supporté et

maintenue pendant le processus de l'IB, qui devient un processus de prise de décision coopératif et distribué. Ces activités collaboratives doivent donc être situées dans un modèle de processus coopératif.

Malgré la reconnaissance croissante que les activités collaboratives de l'IB suivent un processus multidisciplinaire et coopératif [Andriole 93], [Sommerville 92], [Macaulay 95], [Filkenstein 92], [Jarke 93], la plupart des efforts de recherche dans la dimension d'accord sont généralement orientés vers les aspects de **contrôle**, tels que l'automatisation de la vérification de la consistance, l'analyse de la complétude (*completness*) ou la détection des conflits entre les points de vue multiples. Cependant, à l'heure actuelle il n'y a pas encore un consensus sur le fait qu'il est possible (ou, dans quelques cas, souhaitable) de déterminer la complétude d'un ensemble des besoins. En plus, il est reconnu que des inconsistances sont inévitables et doivent être tolérées pendant le processus de construction et d'evolution d'un modèle des besoins du système [Arango 88]. En contrepartie, les notions de construction sociale, coopération et négociation - qui sont aussi une partie essentiel de la dimension d'accord - sont fréquemment négligées.

Cette négligence apparaît aussi dans le domaine de modélisation du processus de logiciel (*software process*) - voir par exemple les travaux dans [ESPWT92] et [EWSPT94] - malgré l'existence de quelques essais comme le modèle MASP [Longchamp 92] et le meta-modèle CPCE [Longchamp 94]. Ces modèles ont souvent assimilé les humains à des processeurs neutres à qui fournir une représentation a priori du processus permet de l'accomplir. En fait, les approches conventionnelles de modélisation du processus de logiciel proposent une structure de buts fixes et une planification rigide des étapes du processus - une définition des processus comme une forme particulière de programmation qui conçoit les acteurs comme des processeurs sur lequel s'exécute le code programmé du processus -, ne supportant pas les aspects non systématiques et opportunistiques de la négociation. Donc ces approches ne sont pas adéquates non plus pour la nature dynamique des activités de collaboration entre les acteurs de l'IB. Une différence très importante entre un système de commande comme l'ordinateur et un système autonome comme l'être humain est justement le fait que l'un se contente d'exécuter alors que l'autre interprète et agit en fonction de critères complexes qui sont la base de l'émergence.

L'Emergence, comme montré à la section suivante, est un concept associé aux modèles théoriques de coopération[13] . L'émergence concerne les interactions entre des acteurs autonomes eux-mêmes et leurs contextes. Les activités coopératives réelles impliquent l'apprentissage mutuel, l'adaptation mutuelle et la négociation des conflits et des changements non prévus , tout cela entre tous les acteurs. Ces activités sont imprévisibles parce elles ont une nature émergente. Il n'est pas possible de savoir *a priori* **quand** ni **le contexte exact** de leur occurrence, mais il est possible de prévoir **ce qui** peut arriver et **comment** le soutenir. Une structure rigide d'un plan non modifiable ou le guidage automatique ne sont pas adéquats à sa réalisation.

28.Fondements Théoriques

Cette section résume les fondements théoriques du concept de l'émergence et la motivation de son utilisation pour le modèle de processus proposé. En particulier, nous montrons comme le modèle émergent est associé aux concepts de Cours d'Action [Thereau 92], Autopoiese [Maturana 80] et Cognition Distribué [Sperber 87], [Hutchins 90**].** Ce résumé est adapté du travail originel de [Cissé 97], où le lecteur intéressé peut

13 Ici, coopération est considérée dans une perspective multidisciplinaire et pas seulement dans une perspective technologique.

trouver des explications plus complètes et des informations additionnelles.

Notre idée séminale est : les activités collaboratives de l'IB sont des activités **émergentes**. La notion d'émergence est issue de divers travaux en ethnométhodologie, anthropologie cognitive et sciences de la communication. Dans ces domaines il y a une sorte de convergence pour considérer la communication et l'action coopératives comme des phénomènes émergents, même que avec des noms différents comme gestalt [Minsky 88] et contextualisme [Potts 97].

La nature émergente des activités collaboratives de l'IB peut être perçu dans le cadre de référence à la théorie de l'*autopoièse* [Maturana 80], un récent paradigme pour la cognition des systèmes vivants. Ce paradigme est dérivé des recherches en biologie et il est utilisé aussi dans les sciences sociales. Le concept clé de la théorie de l'*autopoièse* pour l'IB est l'autonomie des acteurs - qui sont des agents responsabilisés et dotés de compétences. Les acteurs autopoiètiques sont des acteurs qui produisent eux-mêmes leur identités en se distinguant de leur environnement - du grec *autos* (soi) et *poiein* (produire). Les acteurs d'un processus de l'IB sont des détenteurs *(holder)* d'une position *(stake)* devant être respecté par les autres acteurs et par le système - comme le montre bien son nom en anglais *stakeholders*. En effet, la vision des membres d'une organisation comme des individus responsables, donc autopoiétiques de fait, nécessite un mode d'interaction fondé sur la conversation, la prise de décision partagée et la réalisation d'un objectif, dont la négociation est une modalité en cas d'incompréhension. Ainsi, bien que leurs positions soient négociables , les acteurs sont indispensables, essentiels, parce qu'ils sont la condition nécessaire de l'existence du système.

Nous avons choisi les acteurs et leurs interactions comme composantes de base d'une situation collaborative de travail. Caractériser une situation de travail collaborative consiste à identifier ces composants. Identifier les acteurs qui interagissent est relativement trivial lorsqu'il s'agit d'un groupe d'individus qui font partie de l'organisation. Par contre, identifier les interactions est un problème difficile due à la complexité des modalités d'interactions possibles. Les interactions comprennent des signes verbaux ou non verbaux, des situations contextuelles, des appréciations subjectives et prennent place dans l'univers même qu'elles définissent. L'interaction entre ces acteurs et leur environnement est un *domaine cognitif* et l'interaction entre deux acteurs produit un *domaine consensuel*. Le langage humain peut être considéré comme un domaine consensuel , tel que l'a établi [Winograd 88] dans la modélisation des actes de parole (*speech acts*) comme une base pour les conversations. Cependant, les contributions de l'ethnomethodologie [Suchman 87, Lave 88], de l'Anthropologie [Thereau 92] et de la Science Cognitive [Sperber 87, Hutchins 90] sont utiles pour compléter la perspective conversationnel avec d'autres notions comme l'action et la cognition. Nous adoptons donc les modes suivants d'interactions collaboratives entre les acteurs : la communication, l'action et la cognition. Les fondements de ces modes sont les approches de l'Action Située, la Cognition distribuée et la Communication Inférentielle, que nous utilisons de façon complémentaire à la notion d'autonomie de l'autopoiese.

L'Action Située [Suchman 87] est un courant ethnométhodologique qui concerne la nature située du travail. Définir une action comme située signifie qu'elle est déterminée par le contexte dynamique des interactions entre les acteurs impliqués - ce contexte est le résultat de la compréhension que chaque acteur a des actions de l'autre ou de la perception des indices provenant directement de l'environnement . Ainsi pour Suchman, la situation d'une action peut être définie comme l'ensemble des ressources disponibles pour véhiculer la signification de ses actions et pour interpréter celles d'autrui. Cette interprétation essentiellement interactionnelle de la situation suppose que:

- Les actions ne sont pas dirigées par des plans. Ceux-ci sont plutôt une ressource à laquelle l'acteur se réfère. C'est ainsi que les méthodes de conduite de projets sont rarement respectées dans les faits, même si les différents documents qui en résultent sont toujours rédigés selon les phases décrites dans la méthode.

- L'objectivité des situations est propre aux acteurs et non fixée de l'extérieur selon un mode prédéfini;

- Le langage est la ressource principale pour atteindre l'objectivité des situations, c'est à dire leur relation au contexte (indexicalité); En pratique, l'indexicalité signifie que le discours a besoin d'être associé soit à une situation soit à un acteur pour être compris.

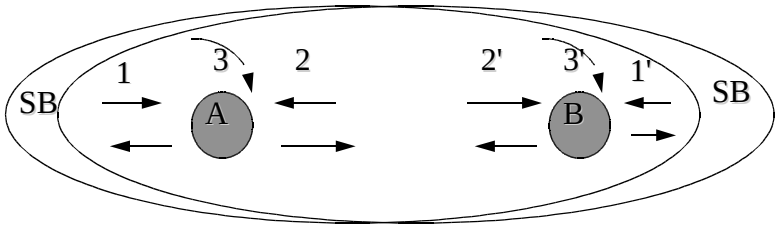
- La compréhension mutuelle est réalisée à chaque instant particulier et non pas une fois pour toutes à travers un ensemble stable de significations partagées.

Par contre, selon la perspective environnementale de Lave, tout savoir et toute aptitude sont ancrés dans l'exploitation de l'environnement. La communication verbale n'est plus le facteur essentiel de contextualisation car [Lave 88]:

- L'action est ancrée dans l'espace, la manipulation d'objets de l'environnement supporte l'exécution de l'action

- L'effet produit par l'action modifie l'environnement immédiat, créant de nouvelles relations spatiales.

Ces deux approches complémentaires permettent de relier l'émergence, fruit de l'autopoièse à l'interaction contextualisée par l'indéxicalité et la spatialité, dans un environnement ou toute norme n'est que ressource et tout objet support potentiel d'action signifiante, voir figure 7.1.



3 est le discours privé de A
3' est le discours privé de B
SA, SB situations respectives de A et B
1,2 relations assymetriques entre A et l'emergent SB
1', 2' relations assymetriques entre B et l'emergent SB
Domaine Consensuel: articulation de 2 et 2'
Domaine Cognitif de A: articulation de 1,2,3
Domaine Cognitif de B: articulation de 1', 2', 3'

Figure 7.1 - Autopoièse (adaptée de Thereau 92)

La théorie de la Cognition Distribuée de Hutchins [Hutchins 90] montre que la connaissance en situation de travail coopératif n'est pas répartie de manière mutuellement exclusive entre les acteurs. Ce serait une situation dans laquelle la

défaillance d'un membre entraînerait forcément celle de l'équipe. En situation réelle, il y a souvent un recoupement partiel entre les connaissances des différents acteurs, ce qui permet la négociation. Ainsi, le domaine consensuel n'est plus l'intersection des domaines cognitifs mais leur union, pour prendre en compte l'imprévisibilité de la diffusion de l'information et l'impossibilité d'anticiper par avance la trajectoire de l'information dans une situation collaborative où la coordination est émergente. La base de la construction de ces connaissances partagées repose sur une distribution de la connaissance, d'où la notion de *cognition distribuée*.

La théorie de la Communication Inferentielle de Sperber and Wilson [Sperber 87], a mis en évidence la nature inférentielle de la communication. Un énoncé suscite un certain nombre d'inférences qui ne sont pas forcément prédictibles parce qu'elles sont dépendantes des stimuli provenant de l'environnement extérieur. C'est donc en opposition aux modèles pragmatiques basés sur les actes de parole qui supposent une sorte de codage/décodage et un sens univoque pour chaque énoncé.

L'univers cognitif d'un acteur comprend les inférences potentielles qu'il peut faire , c'est à dire, des faits ou hypothèses qu'il peut potentiellement faire à un moment donné. L'intersection des environnements cognitifs de deux acteurs est *l'environnement cognitif partagé*, ensemble maximal d'informations partageables par ces deux acteurs.

Les inférences par rapport au contexte sont alors faites pour juger de la pertinence des énoncés reçus par un acteur : un énoncé n 'a pas de pertinence s'il n'a pas *d'effet contextuel* , c'est à dire, l'énoncé n'a aucun lien avec le contexte en cours, ou il est déjà présent dans le contexte ou il est en contradiction avec le contexte sans pouvoir cependant le modifier en profondeur. Le contexte n'est cependant pas fixé avant que le processus d'interprétation n'ait lieu. En effet, le but des humains n'est pas de juger de la pertinence d'une information mais de maximiser l'effet contextuel de l'information pour qu'elle soit productive. Ainsi un énoncé est pertinent si et seulement si il est pertinent dans au moins un des contextes accessibles à cet individu à ce moment .

L'hypothèse générale de l'autopoièse traduit les interactions entre entités autonomes comme les interactions communicatives, mais sont les travaux de Sperber et Wilson qui fournissent un cadre d'analyse non déterministe de la communication. L'environnement cognitif partagé est alors équivalent au domaine consensuel et il est support aux inférences à travers le choix de contextes qu'il permet pour l'interprétation de la pertinence des informations. Cette approche est donc déterminante pour concevoir la communication de façon émergente et liée au contexte.

Enfin, la théorie du Cours d'Action de Jacques Thereau [Thereau 92] permet comprendre comment concevoir d'un point de vue pragmatique le domaine consensuel comme base du travail coopératif. Un cours d'action est l'activité d'un ou plusieurs acteurs engagés dans une situation qui est signifiante pour ce ou ces derniers, c'est à dire racontable et commentable par lui (eux) a tout instant. On voit donc le glissement du domaine consensuel langagier au domaine consensuel centré sur l'action, permettant d'intégrer les résultats des différents travaux.

Pour nous résumer nous dirons qu'une activité collaborative de l'IB est un cours d'action collectif d'acteurs autopoietiques engagés dans le but d'arriver à un consensus sur un aspect déterminé du système. Nous pensons que le modèle de processus émergent est utile pour prendre en compte la nature non déterministique de ce type d'activité. Ce modèle permet de changer d'une perspective individuelle à une perspective sociale, de modéliser l'information nécessaire pour le domaine consensuel et de faire émerger les décisions qui ont une relation avec le processus de l'IB et ses produits.

29.Le modèle de processus émergent

Le modèle de processus émergent proposé ici est une instance d'un méta-modèle développé pour la conception de systèmes de travail coopératifs assistés par ordinateur (CSCW) [Cissé 96b]. Comme [Rolland 94], notre intention est de proposer une approche de méta-modélisation pour définir un processus d'une manière systématique, par l'intermédiaire de l'instanciation[14] de ce meta-modèle. Ce type d'approche permet de manipuler une grande variété de situations, avec les niveaux appropriés de généralité. C. Rolland utilise cette approche pour formaliser des modèles de processus orientés décision, conçus pour fournir un guidage méthodologique de l'exécution des procédures et de la prise de décision durant le cycle de vie du système. Notre modèle, au contraire, concerne la modélisation de processus réels de collaboration, qui à notre avis est un meilleur support pour les aspects sociaux et coopératifs de l'IB qu'une séquence rigide de procédures et de décisions.

En fait, dans les activités collaboratives de l'IB une grande variété d'alternatives sont possibles pour l'accomplissement des buts. Ces alternatives sont influencées par le contexte, qui en réalité délimite l'espace de possibilités pour les stratégies d'action acceptables.

L'idée basique du modèle de processus émergent pour l'IB est que les acteurs autonomes coopérent à travers leur interaction, typiquement par l'intermédiaire de :

- la *conversation* explicite entre les acteurs;
- les *actions* explicites des acteurs sur l'*espace d'objets partagé* ;
- la *cognition*, résultante de l'interprétation et l'inférence d'un acteur des *messages* et *actions* des autres acteurs selon le modèle de leur raisonnement et comportement construit durant *le cours de leur interaction*.

L'interaction entre les acteurs, par ailleurs le traditionnel ensemble d'actions sur l'espace partagé, est donc élargi pour inclure les conversations directes entre les acteurs et la cognition résultant de la cognition sur l'interaction avec les autres acteurs. Nous trouvons une relation analogue dans le travail de [Kirsh 94], où il est proposé une distinction entre les *actions pragmatiques* - actions visant l'accomplissement d'un but - et les *actions epistemiques* - actions réalisés pour découvrir des informations qui sont camouflées ou difficiles d'obtenir. En fait, les activités d'un groupe ne sont pas seulement des manipulations finalisées de produits. Sauf si les acteurs eux-mêmes travaillent dans les multiples disciplines, la communication entre les membres du groupe est cruciale pour permettre la multidisciplinarité à partir de la coopération entre leurs perspectives individuelles.

Nous présentons ensuite un explication du schéma du modèle et leur composants - montrés dans la figure 7.2. Ce modèle a été conçu pour être indépendant de l'approche/ méthode de l'IB mais dans cette explication nous utilisons des modèles et des activités de l'approche TAREFA, cf. Chapitres IV, V et VI.

14 Le terme couramment utilisé dans le domaine de modélisation de processus du logiciel est 'énaction' qui a typiquement le sens d'exécution/guidage. Le terme 'instanciation' nous parait plus adequat pour reflechir le sens de Varela (*faire émerger*), qui propose que cette 'instanciation' est ce qui produit le processus. Il faut remarquer, cependant, que le terme utilisé par Varela pour une transformation dynamique de la structure est 'ontogenèse' qui ne nous parait moins comprehensible pour les informaticiens que l'instanciation.

Figure 7.2 - Le modèle de processus emergent pour les activités collaboratives de l'IB

Les activités collaboratives de l'IB consistent des constructions et des transformations des objets par l'intermédiaire d'interactions orientées buts entre les acteurs. Les activités de l'IB - collaboratives ou non - sont motivées par les buts. Une démarche pour les activités non collaboratives défini une séquence d'actions orientée but devant être suivie par les acteurs mais pour les activités collaboratives les acteurs ne peuvent pas être imposés à suivre un ensemble particulier d'actions, même si leur comportement est aussi intrinsèquement orienté buts.

Un **plan** est décrit en termes procéduraux (et pas de façon déclarative) comme un ensemble de buts qui peut être incrementalement accomplis - sequentiellement ou parallèlement. Les acteurs sont libres de choisir comment accomplir chaque but. Une structure de contrôle n'est pas imposée sur les actions parce que le plan n'est qu'une contrainte pour les résultats du processus et non pour sa réalisation. Un plan est **évolutif**, parce les **buts** peuvent évoluer avec l'expérience et le déroulement du processus, afin d'être orientés vers les situations spécifiques. En fait, les acteurs sont créatifs et cherchent continuellement à améliorer et optimiser la façon de réaliser leurs activités. Un plan évolutif, où les buts peuvent être graduellement découverts, supporte une coopération créative et permet que les descriptions des buts génériques soient flexiblement raffinées pour s'adapter à des situations spécifiques.

Les **acteurs** sont les individus membres de l'équipe multidisciplinaire de l'IB et ils jouent des **rôles** pour accomplir les buts. Un simple acteur peut jouer plusieurs rôles et

donc participer à plusieurs activités du processus. Il faut remarquer qu'un but est attaché à un rôle et non à un acteur. Cependant, les acteurs peut être impliqués dans une conversation, indépendamment de leurs rôles, pour se communiquer avec les autres acteurs.

Un **espace d'objets** partagé consiste en **artefacts** - y compris toute l'information orienté les produits du processus de l'IB - et en **objets orientés social** - y compris les interactions entre les acteurs durant le processus.

Le but final du processus de l'IB est l'élaboration d'un produit appelé modèle des besoins (ou spécification des besoins) , typiquement par l'intermédiaire de la manipulation d'artefacts pour représenter la compréhension sur le domaine du problème, sur les besoins des utilisateurs, les besoins du système, et les contraintes de l'environnement. Plusieurs types différents d'artefacts sont possibles. Les artefacts de l'approche TAREFA sont les :

- modèles de l'information contextuelle ;
- modèles intermédiaires ;
- modèles des besoins du système ;
- documents (structurés ou informels) associés aux modèles ci-dessus ;
- combinaisons ou compositions des artefacts.

Les **actions** sont la manipulation des artefacts, y compris la création, modification, transformation de l'un en autre, suppression, consultation et réutilisation. Donc les actions correspondent exactement à la notion d'activités systématiques présentes dans le démarche des macroactivités d'Analyse et de Synthèse de TAREFA. L'exécution d'une action peut être basée sur les événements qui arrivent de l'espace d'objets partagé. Les pré- et pos-conditions des événements définissent 'quand' une action peut être réalisé et si cette réalisation a réussi. Alors, la réalisation d'une action est à la fois orienté buts et orientée événements.

Les objets orientés social consistent en **conversations**, **messages** et **arguments**. Une conséquence significative de reconnaître ces objets comme une information pertinente est que, à l'opposé de ne considérer que des actions explicites sur les artefacts , les objets orientés social doivent être aussi considérés pour choisir comment aller d'un but à l'autre. Cela veut dire qu'à l'origine d'un changement de l'espace d'objets partagé il faut interpréter à la fois les changements des artefacts et les changements du domaine consensuel dû aux objets orientés social. Ainsi, une séquence de messages entre des acteurs peut changer un argument, une opinion ou un argument sans nécessairement changer un artefact.

Terry Winograd utilise la conversation pour la conception de son système Coordinator [Winograd 88]. Pour lui une conversation est une séquence coordonnée d'actes de langage (*speech acts*). Les messages de modèle de conversation de T. Winograd doivent être catégorisés par leur intention comme ayant pour but une action, un éclaircissement, une orientation ou la génération d'autres conversations. Cependant, dans une situation sociale normal, cet ensemble limité de catégories peut être remis en question. Nous ne pouvons pas considérer tout ce que les personnes font avec le langage comme un acte coordonné ou une conversation structurée. Donc, dans notre modèle, une **conversation** est simplement composé de **messages** envoyés et reçus par/de les acteurs du processus, mais qui sont indexés (dans le sens de l'indexicalité) par leur relation au contexte, notamment à travers les buts. Dans la plupart des cas, l'information

informelle (usuellement non structurée et parfois partiellement inconsistante ou ambiguë) générée durant les conversations est perdue à cause de la difficulté de l'intégrer aux informations représentées par des notations plus formelles. C'est pour cela que nous pensons qu'il est extrêmement utile d'enregistrer les messages des acteurs et ainsi de capturer d'une manière implicite leurs connaissances tacites et implicites comme leurs concepts, définitions, opinions.

En plus, le *design rationale* peut être explicitement enregistré par l'intermédiaire des décisions et arguments. Les **décisions** sont des choix consensuels motivés par les buts, concernant les alternatives d'un cours d'action. Toute décision ou action peut être associé à des arguments (*pro* ou *contre*). Les **arguments** sont une explication consensuelle et non un message individuel ni un type spécial de message echangé entre des acteurs. Les arguments *pro* sont en faveur d'une décision ou d'une action ; les arguments *contre*, le contraire.

Pour nous résumer nous dirons que les acteurs coopèrent par l'intermédiaire de l'espace d'objets partagé de deux façons :

1) directement , à travers les conversations ;

2) indirectement, à travers les résultats des manipulations des artefacts.

Les différentes manières d'arriver aux buts constituent l'espace de possibilités pour la réalisation des actions dans un contexte particulier. Un contexte est considéré à la fois comme ressources et contraintes aux actions. Il est naturellement changé par les interactions entre les acteurs et limité par la disponibilité des ressources pour les actions, par l'information sur l'état actuel des artefacts et les événements qui permettent les actions. L'existence de plusieurs alternatives pour une action dépend d'une correspondance multiple entre les moyens et les buts d'une situation spécifique selon la perception (directe ou inferentielle) de l'acteur qui la réalise. Cet espace de possibilités configuré par les alternatives donc émerge des interactions, ce qui est la principale caractéristique de notre modèle de processus. Si c'était le contraire, les activités collaboratives ne seraient pas émergentes mais totalement déterminées, sans la nécessité de choix, participation et décision de la part des acteurs.
Le chapitre 8 - l'application de TAREFA à l'étude de cas SOHO - présente un exemple pour illustrer quelques aspects de ce modèle. Cet exemple montre le modèle de processus associé à une session de simulation de la détermination des besoins d'un outil de travail coopératif assisté par ordinateur (CSCW) pour supporter les activités de planification hebdomadaire de l'opération du satellite multi-instrument SOHO. Nous pensons que cet exemple réel et complexe aide à comprendre notre modèle émergent.

30.Synthèse

Les activités collaboratives réelles de l'IB ne peuvent pas suivre une séquence rigide et les décisions selon les pas d'une méthodologie centrée concepteur parce que leur dynamique est déterminée par les choix, les décisions et la participation des acteurs membres de l'équipe multidisciplinaire. En fait , dans la plupart des collaborations entre les acteurs, une grande variété d'actions alternatives et de conversations peuvent être combinées pour l'accomplissement des buts. Ces alternatives sont influencées par le contexte qui constitue l'espace de ressources pour les stratégies acceptables de l'IB. Ces propositions sont d'accord avec les idées exposées dans [Barghouti 95] qui montrent que la modélisation formelle des processus réels peut révéler des inconsistances et des inefficacités du processus et ne pas être utilisée. Nous pensons que cela est particulièrement vrai pour les activités collaboratives de l'IB , qui ne peuvent pas être

modélisés par un processus basée sur une structure de buts fixes et une planification rigide des étapes à suivre.

Pour soutenir ce type d'activité nous avons proposé dans ce chapitre un modèle de processus basée sur la notion d'*émergence* qui est conçu pour prendre en compte la complexité, la dynamique et par conséquence la richesse de la coopération des acteurs humains de l'IB. Le modèle de processus émergent prend en compte à la fois - à travers un espace d'objets partagé - l'action, la conversation et la cognition pendant le cours d'interaction entre les acteurs, où les plans et les buts originels évoluent d'une façon consensuelle et émergente.

Partie III : L’Etude de Cas

Chapitre VIII - l'Etude de Cas: l'Ingénierie des Besoins de l'Outil d'Assistance aux Réunions de Planification Hebdomadaire d'Opération du Satellite Multi-Instrument SOHO

31.Introduction

L'objectif de ce chapitre est de présenter succinctement l'application de l'approche TAREFA à une étude de cas réel : l'ingénierie des besoins du système informatique pour soutenir les réunions de planification scientifique des opérations hebdomadaires des instruments à bord du satellite SOHO (*SOlar and Heliospheric Observatory*).

Ce chapitre est structuré en 3 sections. Dans la première section (section 2) , une description de l'étude de cas est présentée, en particulier de la PLAnification HEbdomadaire des Opérations du Satellite Multi-Instrument SOHO (appelée PLAHE-SOHO dans cette thèse), qui nous intéresse plus spécifiquement.

La section 3 est consacrée à la présentation de l'application de TAREFA à l'étude de cas PLAHE-SOHO, plus spécifiquement à l'ingénierie des besoins pour un Outil d'Aide à la planification PLAHE-SOHO. Le plan de cette section suit le plan des chapitres 5 et 6, c'est à dire, intègre une sous-section pour la macroactivité d'Analyse et une sous-section pour la macroactivité de Synthèse et la description des extraits de chacun des modèles produit pour chacune des macroactivités.

Finalement, la section 4 présente une synthèse sous la forme d'une discussion des résultats de l'application de TAREFA à PLAHE-SOHO.

32.Description de l'étude de Cas

Cette section résume les principales caractéristiques de l'étude de cas, réalisée dans le cadre de la mission SOHO. L'objectif de cette section est uniquement de présenter succinctement l'étude de cas. Comme nous ne présentons que des extraits de l'application de TAREFA à l'étude de cas, on ne pourrait pas avoir une idée complète de son fonctionnement sans cette description.

Ce résumé commence par une introduction à SOHO (sous-section 2.1), puis la présentation des caractéristiques générales de la planification scientifique dans le cadre de la mission SOHO (sous-section 2.2) et finalement, des caractéristiques de la PLAnification Hebdomadaire (sous-section 2.3), appelé PLAHE-SOHO dans cette thèse, sur laquelle nous avons appliquée l'approche TAREFA. Due à la complexité du sujet, un travail précèdent a étudié comment introduire un système de travail coopératif au groupe SOHO pour l'assister pendant ses réunions de planification [Bellamine 96]. Un résumé de ce travail précèdent est présenté dans la sous-section 2.4.

32.1Une introduction à SOHO

Cette section résume les principales caractéristiques de la mission SOHO. Une description plus détaillée de la mission SOHO et de ses planifications coopératives est présentée en [Bellamine 96].

Une mission spatiale est composée du satellite lui-même et d'un segment sol qui assure son exploitation.

Un satellite est un engin spatial composé d'une plateforme et d'une charge utile, et appartient à l'une des catégories de missions spatiales suivantes : missions scientifiques, missions d'observation de la Terre, missions de télécommunication ou vols habités. SOHO (SOlar and Heliospheric Observatory) est un satellite multi-instrument destiné à une mission scientifique bien déterminée : l'étude du soleil. C'est un projet de coopération entre les agences spatiales européenne - ESA (European Space Agency), et américaine - NASA (National Aeronautics and Space Administration), qui sont responsables respectivement de la construction du satellite et de son lancement et des opérations en cours de mission.

La charge utile de SOHO est composée de onze instruments, divisés en trois classes selon les domaines de recherche différents et complémentaires auxquels ils sont dédiés : la héliosismologie - l'étude de la structure interne du soleil (3 instruments : GOLF, MDI et VIRGO), l'atmosphère solaire (6 instruments : CDS, EIT, LASCO, SUMER, SWAN et UVCS) et l'origine du vent solaire (2 instruments : CELIAS, CEPAC). Ces plusieurs instruments différents doivent fonctionner de manière coordonnée en vue d'exécuter des programmes scientifiques particuliers. Contrairement à d'autres missions spatiales, où la charge utile est sous le contrôle d'un seul centre dédié et d'une équipe dédiée, SOHO est une mission avec de multiples responsables scientifiques, un pour chaque instrument. En effet, chaque instrument sera exploité en coopération entre les équipes de plusieurs instituts et laboratoires de plusieurs pays.

Le cycle de vie d'une mission spatiale se compose essentiellement d'une *phase de préparation* - construction du satellite et de sa charge utile, préparation des centres et des procédures de son contrôle et son exploitation, etc -, et d'une *phase d'opération-exploitation*, la partie du cycle après le lancement, y compris les sous-phases de mise en trajectoire, placement en orbite, opérations de vérification et bien sur exploitation. SOHO est actuellement dans sa phase d'exploitation et le tout se déroule normalement. Le lancement de SOHO a eu lieu le 2 décembre 1995, à partir de Cap Canaveral en Floride, par une fusée Atlas II AS (voir plus d'informations sur le site http ://sohowww.nascom.nasa.gov). La durée prévue de l'exploitation est de 2 ans, avec une possibilité d'extension de 4 ans.

La phase d'exploitation du satellite se traduit par un dialogue régulier sol-bord basé sur des échanges de données : le satellite transmet au sol les informations qu'il collecte durant sa mission en envoyant des télémesures (désignés par TM); le centre de contrôle communique avec le satellite via des télécommandes (désignés par TC) pour modifier sa configuration. En fait, l'exploitation est une boucle continue: planification des opérations - télécommande des instruments - analyse des données des télémesures.

Un segment sol correspond aux bancs de tests, aux centres de contrôle, et aux centres d'opération qui contribuent à l'intégration et à la validation du satellite au sol, à sa mise à poste lors du tir, et à son exploitation en orbite. En particulier, le segment sol de SOHO est distribué et réparti sur différents sites:

- Les services généraux de la NASA utilisés entre autres par SOHO et qui constituent l'interface entre le centre EOF (Experimenters' Operation Facility) et le satellite SOHO (aussi bien pour l'envoi des télécommandes que pour la

réception de télémesures);

- Le centre d'opérations américain: l'EOF, le centre principal d'opération SOHO, responsable pour la maîtrise du bon fonctionnement de la charge utile et aussi pour assurer la communication entre les équipes instruments et les différents éléments du segment sol de SOHO;

- Le centre d'analyse américain: EAF (Experimenters' Analysis Facility), un centre d'analyse des données SOHO, où les équipes instruments utilisent leurs outils d'analyse et leurs stations;

- Le centre d'opération européen: MEDOC (Multi Experiment Data Operations Center), situé à l'IAS (Institute d'Astrophysique Spatiale) en Orsay, est dedié à la mise en oeuvre des programmes des instruments de SOHO depuis l'Europe. MEDOC fonctionne comme centre auxiliaire du centre principal EOF.

Aux centres EOF et MEDOC, des réunions de planification scientifique sont réalisés pour coordonner les observations des instruments. Il faut souligner que la planification scientifique s'occupe de la charge utile uniquement pendant que la planification de mission s'occupe de la mission dans son ensemble, en intégrant tous ses éléments (segment spatial, segment sol et toute autre contrainte relative à la mission) et en couvrant toutes les opérations de tout le satellite. Dans cette thèse, nous nous concentrons uniquement sur les réunions de planification scientifique.

32.2Planification scientifique des opérations de la mission SOHO : caractéristiques générales

Nous nous intéressons à la planification des opérations de la charge utile du satellite SOHO. En fait, les instruments à bord de SOHO se partagent des ressources satellite, ont des champs de vue complémentaires, sont de complexités différentes, et pointent tous et en permanence vers le soleil pour permettre aux astrophysiciens solaires de l'étudier. Les opérations ont pour but d'envoyer au satellite des commandes et de recueillir les données produites par les instruments.

L'organisation associée à la mission SOHO dans son ensemble, et à chacune des équipes instruments, définit les relations entre les partenaires et précise les responsabilités de chacun d'eux. Outre l'investigateur principal -PI[15]- responsable principal de l'instrument, une équipe est également composée de CoInvestigateurs -CoI-, scientifiques associés -AS[16]-, investigateurs invités -GI[17]-.I : Guest Investigator. Au cours des opérations, les responsabilités sont partagées entre plusieurs membres du projet SOHO. Nous trouvons donc des responsables des agences spatiales - américaine et européenne -, des responsables de la plate-forme et un responsable (PI) par instrument embarqué à bord. Chaque PI est responsable pour le contrôle de ses instruments de bout en bout : de la planification des observations inter-instruments (y compris la gestion des ressources et des temps) à la génération des commandes.

Généralement, une équipe entière participe au développement puis à l'exploitations de chaque instrument de SOHO. Pendant la phase opérationnelle du satellite, les utilisateurs définis sont essentiellement des astrophysiciens. Chaque astrophysicien a un

15PI : Principal Investigator
16AS : Associated Scientist
17GI : Guest Investigator

ensemble d’idées ou de théories à vérifier et des objectifs scientifiques souhaités pour chaque étape de la mission. Ainsi, pour chaque instant de la vie du satellite une séquence unique d’instructions compréhensible par le satellite et par chacun des instruments doit être formulé afin de répondre au mieux aux souhaits des utilisateurs. Pour la mission SOHO, il est nécessaire de tenir compte d’un grand nombre de points de vue des scientifiques intéressés afin de bien exploiter les capacités de cette mission. Afin de converger vers un plan sans conflits répondant au mieux aux demandes des scientifiques, des activités de planification s’imposent. La planification désigne l'organisation d'activités élémentaires et leurs échelonnement dans le temps tout en respectant un certain nombre de conditions.

La planification de mission est définie comme le "processus de planifier et d'ordonnancer toutes les activités et les opérations du segment spatial et du segment sol associé à une mission donnée" [Gasquet et al. 94].

Le grand souci actuel des différentes missions consiste à réduire les coûts, à avoir des systèmes moins complexes et à assurer une bonne planification. Ré-utilisabilité, flexibilité, performance et évolutivité sont les critères essentiels attendus d'un système de planification. Cette planification peut être hiérarchique ou non, parallèle ou séquentielle, centralisée ou distribuée. Ainsi, plusieurs solutions techniques peuvent être proposées : architecture distribuée, utilisation de l'architecture client-serveur, les librairies, l'utilisation des approches orientées objet, développement de systèmes génériques, utilisation des bases de données et des bases de connaissances, etc.

Cependant, le processus de planification considéré dans son ensemble ne se limite pas toujours à un processus de calcul combinatoire où les opérandes en entrée (i.e. activités, requêtes, contraintes, etc.) sont définies, où les opérateurs (règles sur les opérandes, critères de calcul, etc.) sont établis, et où le résultat est calculé - entièrement ou partiellement - par le système. En effet, pour des missions complexes, multi-instrument par exemple, où l'expertise et la connaissance humaine sont difficilement représentables (ni maîtrisables), la planification est en grande partie assurée par des scientifiques impliqués dans la mission. Dans un tel contexte, un outil sert souvent à organiser, et à regrouper les contributions au plan.

Toutes les activités et les étapes se déroulant depuis l’idée de l’astrophysicien jusqu’à l’instruction ou la commande élémentaire envoyée au satellite sont des éléments de la planification de la mission dans son ensemble.

D’après [Domingo et al. 95], la routine général de la planification prévue des opérations de SOHO est décrite comme suit:

" The SOHO Working Team (SWT) will set the overall science policy and direction for mission operations, set priorities, resolve conflicts and disputes, and consider Guest Investigator observing proposals. During SOHO science operations, the SWT will meet every three months to consider the quarter starting in one month's time and form a general scientific plan. If any non)standard DSN contact are required, the requests must be formulated at this quarterly meeting. The three-month plan will then be refined during the monthly planning meeting of the Science Operations Team (SOT), composed of those PI's or their team members with IWS's at the EOF, wich will allocate observing sessions to specific programs. At weekly meetings of the SOT, coordinated timelines will be produced for the instruments, together with detailed plans for spacecraft operations. Daily meetings of the SOT will optimize fine ponting targets in response to solar conditions and adjust operations if DSN anomalies occur"

Une planification se déroule en plusieurs étapes. Chacune de ces étapes correspond à des réunions de planification et la fréquence ainsi que les objectifs (et participants qui y assistent) de ces réunions varient. La figure 8.1 montre une description synthétique des différentes phases de planification.

	Planification Trimestrielle	**Planification Mensuelle**	**Planification Hebdomadaire**	**Planification quotidienne**
Fréquence	Chaque trimestre	Chaque Mois	Chaque Semaine	Chaque Jour
Objectifs	Etablir une stratégie globale d'opération pour le trimestre suivant	Etablir un plan d'opération pour le mois suivant	Etablir un plan détaillé pour la semaine suivante	Définir avec précision le plan du lendemain
Participants	SWT + membres du FOT + SOC + PSs	SPWG + membres du FOT + SOC + PSs	SOL + SOT + membres du FOT + SOC + PS	SOL + SOT + membres du FOT + SOC + PS

Figure 8.1 - Resumé des phases de planification des opérations de SOHO

Après la planification détaillée, l'écriture des commandes de chaque instrument se fait par l'équipe de ce dernier et constitue en principe une traduction fidèle (soit manuelle soit automatique) du plan en commandes.

Dans cette thèse nous nous intéressons plus particulièrement aux planifications hebdomadaires.

32.3La planification Hebdomadaire de SOHO (PLAHE-SOHO)

Les réunions hebdomadaires sont des réunions pour la planification scientifique d'une semaine d'opérations des instruments du satellite SOHO. Ces réunions peuvent être de deux types :

1. directe ou face-à-face, géographiquement centralisé dans un seul centre d'opérations, en général le centre d'opérations américain EOF; ou

2. distribué , géographiquement distribuée entre le centre américain et le centre européen ;

Nous nous sommes intéressés de plus près au premier type : l'étude qui a eu lieu dans le centre d'opérations américain de SOHO.

La réunion hebdomadaire a lieu le vendredi de chaque semaine afin de prévoir la planification de la semaine suivante commençant le lundi suivant. Cette réunion est organisée par l'équipe d'opération SOHO - SOT -, et elle est conduite par un leader (SOL + SOC actuellement) selon un agenda précis, voir figure 8.2.

Cette réunion constitue l'occasion pour les astrophysiciens de présenter et de discuter les possibilités de coordination avec les différents instruments, tout en respectant leurs programmes nécessitant la mise en œuvre de plusieurs instruments. En effet, pendant la réunion, un responsable de chaque instrument présente les demandes de son équipe ainsi que les disponibilités de l'instrument lui-même.

Les caractéristiques principales d'une réunion hebdomadaire se définissent ainsi :

- l'objectif général se résume en 'la planification des opérations d'une semaine d'observations du satellite SOHO'.

- les supports disponibles pendant cette réunion sont les tableaux de la salle, un rétroprojecteur et une tablette LCD, des terminaux X ;

- la taille moyenne du groupe de participants est de 15 personnes ;

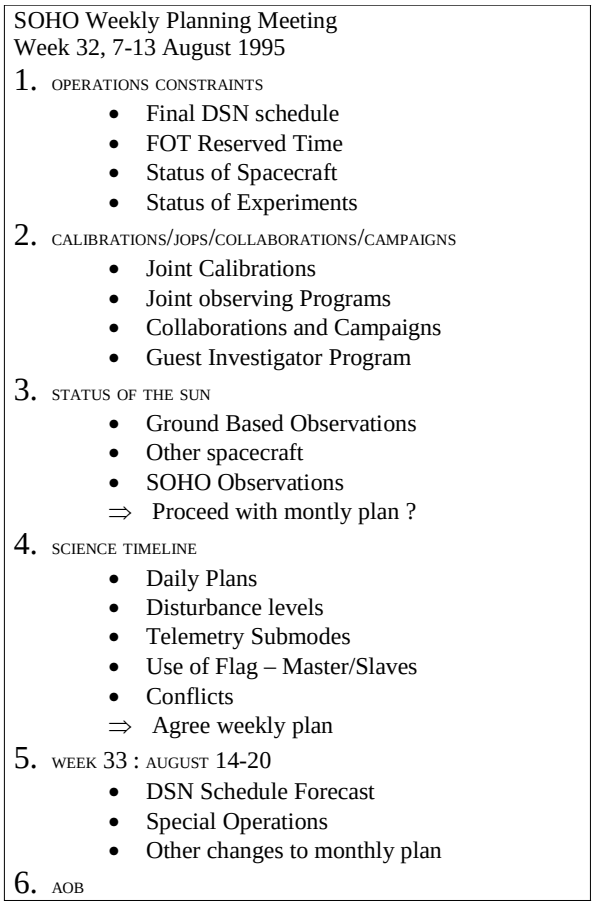


Figure 8.2 - Exemple d'agenda de réunion hebdomadaire de SOHO (d'après [Bellamine 96])

- la composition du groupe est : leader + représentants de chaque instrument (ou sous-système à planifier) + coordinateur ; en fait tous les participants n'ont pas le même rôle et leurs poids sur la prise de décision est diffèrent : par exemple, le PI a une grande liberté et influence, et c'est lui qui a le dernier mot en ce qui concerne son instrument ;

- même avec une composition structurée, le groupe est volatile, c'est à dire, les acteurs jouant des rôles particuliers changent d'une session de réunion à une autre ; cependant, au niveau d'une semaine donnée le groupe est conservée : la même équipe formée pour la réunion de planification hebdomadaire se réunit tous les jours de la semaine pour détailler et mettre à jour la planification quotidienne et veiller à sa bonne mise en oeuvre ;

- La réunion est structurée : elle se déroule en plusieurs phases distinctes (par exemple mise en contexte, étude/présentation de contraintes techniques, programmes coordonnés, identification des objectifs scientifiques, brainstorming, prise de décision, édition de plans), elle est animée par un leader et suit un ordre du jour ;

- la communication entre les membres est directe (il n'y a pas de système informatique pour médiatiser la communication) et synchrone (une personne parle et toutes les autres écoutent)

- la prise de décision est collective et consiste d'un accord de tous les participants sur les choix effectués ;

- le résultat d'une telle réunion est une planification globale (couvrant tous les instruments et la période prévue), commun (le même résultat est diffusé pour tous les participants), consensuelle (approuvée par tous les participants) et cohérente (scientifiquement et techniquement correct et sans conflits).

32.3.1Les rôles de la planification hebdomadaire

Le groupe des participants des réunions de planification hebdomadaire est composé essentiellement des astrophysiciens jouant un ou plusieurs rôles, comme les rôles de représentants des équipes instruments, de représentants des centres d'opération et des centres de contrôle de la mission.

Ces utilisateurs se regroupent en *équipes instrument.* Chaque équipe est sous la responsabilité de son investigateur principal (*Principal Investigator* ou PI), qui est le premier responsable de l'instrument. En plus du PI, chaque équipe est composée de Co-Investigateurs (Co-Investigator ou CoI) , de Scientifiques Associés (Associated Scientist ou AS) et d'Investigateurs Invités (Guest Investigator ou GI). Un CoI est associé au PI pendant toute la mission, ayant le droit d'accès à toutes les données. Un AS est aussi associé au PI mais sa participation est secondaire. Un GI participe occasionnellement pour faire des programmes d'observation en travaillant avec l'équipe de l'instrument.

Ces équipes sont généralement non disjointes, c'est à dire, qu'une même personne peut appartenir à plusieurs équipes à la fois, mais souvent avec des responsabilités différentes (un PI d'un instrument peut être CoI d'un autre par exemple).

Outres les PIs, CoIs, ASs, et Gis formant des équipes de chaque instrument, d'autres rôles et regroupement sont prévus, comme le PS, SOL, le SOC, le SDC, le OEP et les groupes SWT, SOT, SPWG et FOT.

Le Responsable Scientifique du Project (Project Scientist ou PS) contrôle le bon développement des instruments et de leur conformité aux spécifications de la mission et est un rôle partagé par deux personnes : un responsable scientifique américain (NASA) et un responsable scientifique européen (ESA).

Le Science Operations Leader (ou SOL) est le responsable du bon déroulement du plan de la semaine et est un rôle joué par l'un des Pis ou l'un de ses représentants, en général choisi par le SOT (voir définition ensuite). Le SOL commence la préparation des opérations hebdomadaires trois jours avant le début de la semaine, en co-presidant la réunion hebdomadaire et les réunions quotidiennes pendant sa semaine de responsabilité. Sa principal responsabilité est d'identifier les conflits entre les opérations planifiées et les plans de contact avec le satellite.

Le Science Operation Coordinator (ou SOC) est le coordinateur des opérations scientifiques mais il n'est membre d'aucune équipe ; il doit assurer la continuité operationnelle au cours de la mission.

Le Science Data Coordinator (ou SDC) s'occupe de l'archive SOHO à l'EOF.

L'Opérateur d'Editeur de Plans (ou EOP) est le responsable pour l'utilisation d'un outil informatique existant, qui est un éditeur graphique de planning. L'EOP reçoit les demandes verbales des participants, traduit les demandes en termes d'instructions à l'outil et les applique pour mettre à jour les informations du planning.

Le groupe Science Working Team (ou SWT) est composé de Pis et CoIs. Il se réunit une fois tous les trois mois pour établir une politique scientifique générale. Il affecte les

priorités, resoud les conflits et étudie les propositions des divers investigateurs (futurs CoIs ou GIs).

Le groupe Science Operations Team (ou SOT) est composé de représentants de chaque instrument et est responsable de l'exécution des plans scientifiques préparés lors des réunions mensuelles, hebdomadaires et quotidiennes.

Le groupe SOHO Planning Working Group (ou SPWG) est le groupe de planification scientifique des instruments de SOHO composé également des Pis et/ou de leurs représentants. Avant le début des opérations, ce groupe s'occupe de la préparation et validation des plans d'observation conjoints - Joint Observation Plans (JOPs) impliquant plusieurs instruments de SOHO et de collaborations avec les observatoires au sol pour la réalisation d'un objectif commun.

Le groupe Flight Operations Team (ou FOT) est responsable des commandes satellite.

32.3.2Les activités de la planification hebdomadaire

Les activités de haut niveau de la planification hebdomadaire de SOHO suivent le diagramme fonctionnel montré sous la figure 8.3.

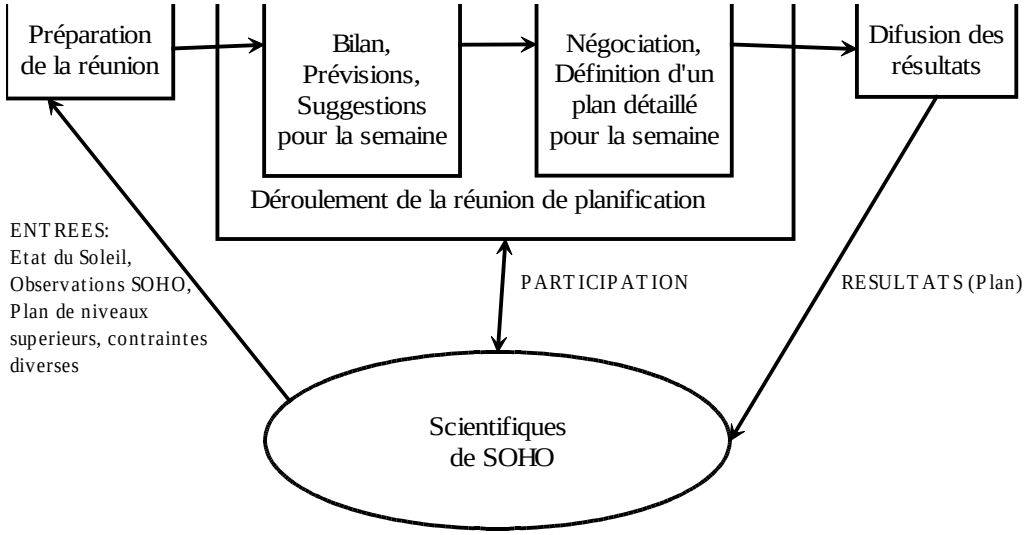


Figure 8.3 - Diagramme Fonctionnel des activités de la réunion de planification hebdomandaire de SOHO

Une description du déroulement d'une réunion sera présentée ensuite.

Déroulement d'une réunion de planification hebdomadaire

Si les participants 'clés' sont présents, le leader SOL annonce le début de la réunion.

L'agenda de la réunion est en général prêt et disponible quelques jours avant sur les pages web SOHO. S'il n'y a pas un accord collectif sur l'agenda, une discussion est commencée pour le modifier.

Ensuite, les bilans sont présentés. D'abord les bilans techniques (par exemple, l'état des ressources satellite ou le bilan des pointages) sont présentés par le FOT ou le SOC ; puis les bilans généraux (par exemple, le rappel des plans mensuels prévus, l'état effectif du soleil pendant la semaine précédente, les bilans de la semaine précédente comme le pourcentage de programmes réalisés par rapport à l'objectif ou bien le bilan des opérations d'acquisition et distribution des données) sont présentés par le PS et/ou

le SOC ; finalement les prévisions scientifiques générales (par exemple, la prévision de l'état du soleil pour identifier les événements solaires importants qui seront cibles candidates des instruments).

En suivant l'ordre du jour, le leader - SOL - s'adresse au représentant de chaque instrument pour lui demander de décrire ses contributions. Les contributions de chaque équipe sont typiquement :

- présentation du bilan de l'instrument, c'est à dire, présenter l'état de l'instrument, les résultats des opérations et le bilan scientifique ;

- présentation du plan hebdomadaire préliminaire, proposé par l'équipe responsable de l'instrument et qui contient aussi la répartition des programmes prévus sur la semaine - jour par jour.

Après les contributions, commence la phase de négociation du plan détaillé consensuel pour la semaine. Cette négociation porte sur l'évaluation collective des conditions ou contraintes sur les ressources ou opérations à fin de modifier les plans préliminaires proposés par les équipes instruments. En général le plan final est l'union des plans (modifiés après la phase de négociation) de chaque instrument.

L'évaluation de conditions et contraintes est faite en quatre étapes :

1. Identifier et définir les programmes d'observations synoptiques, c'est à dire, les images et rapports scientifiques obtenus régulièrement à partir d'observatoires sol et/ou d'autres missions spatiales et qui peuvent être utiles pour la planification SOHO ; Cette définition implique de déterminer le nom de l'observation, la date, l'horaire, la fréquence, la durée et la priorité ;

2. Identifier et définir les programmes spécifiques de chaque instrument individuellement; cette définition implique de déterminer le nom du programme, la date, l'horaire, la durée, la priorité et les conditions particulières de sa réalisation ;

3. Identifier et définir les programmes communs (JOPs), c'est à dire, les programmes d'observation coordonnée qui nécessite la participation de plusieurs instruments ; cette définition implique de déterminer le numéro du JOP, la date, l'horaire, la durée et la priorité.

4. Identification des conditions ou contraintes additionnelles aux programmes proposés; elles sont dites 'additionnelles' en rapport aux contraintes présentées dans l'étape 'Bilans et Prévision' au début de la réunion parce qu'elles ont été ajoutées pendant le processus de négociation ; cependant elles doivent aussi être respectées lors de l'établissement d'un plan d'opérations.

A la fin de la négociation, le résultat est un plan hebdomadaire détaillé.

Pendant la réunion, divers équipements sont utilisés pour soutenir les présentations et la discussion. Par exemple :

- Le SOL utilise le retro-projecteur pour projeter l'ordre du jour et les images du soleil.

- Le SOL écrit au tableau le planning (*timeline*) au fur et à mesure de sa création pendant l'avancement de la réunion.

- Le SOC utilise l'outil de pointage pour visualiser les transparents des images du

soleil (donc le terminal X et la tablette LCD).

- Le reste des interactions dans le groupe est sous forme de parole (communication verbale directe).

32.4Le travail précèdent : EPISCOC

SOHO est une étude de cas riche et complexe. A cause de la complexité du sujet, un travail précèdent a étudié comment introduire un système de travail coopératif au groupe SOHO pour l'assister pendant ses réunions de planification [Bellamine 96]. Un des buts de ce travail était de proposer une méthode pour la conception des systèmes multi-utilisateurs coopératifs pour assister les situations coopératives, plus particulièrement la situation de planification appliqué à SOHO. En fait, la méthode EPISCOC (Etude, Production et l'Introduction d'un Système COopératif à un Collectif) a été proposé en tenant compte de la coopération et des collectifs pour la réalisation de chaque étape qui sont:

- L'Etude, où est proposée la définition de l'univers coopératif et la relation d'interaction entre ses composants humains, matériels et contextuels comme une base pour la caractérisation des situations coopératives complexes; cette caractérisation sert de repère dans l'analyse des dysfonctionnements existants et la spécification des solutions possibles, bien entendu des solutions organisationnelles y compris les changements de processus et procédures intentionnels du collectif (comme les modes de coopération et de contrôle, les moyens et supports de coordination sur ces derniers, etc) et pas seulement des solutions informatiques;

- La Production, où le prototypage rapide d'une solution collecticiel est faite;

- L'Introduction du Système Coopératif au Collectif, où une simulation de travail coopératif assisté par le prototype et une évaluation de la solution collecticiel introduite au collectif sont réalisés.

Comme résultat pratique le prototype d'un éditeur de planification partagé a été conçu à partir d'un éditeur mono-utilisateur existant et ensuite validé avec les utilisateurs réels. Pourtant, ce prototype n'a été qu'une solution partielle, un support à une investigation plus poussée afin de choisir une solution plus complète et adéquate. Notre étude de cas se situe donc dans la continuation de ce travail, et correspond à un approfondissement de l'étape d'Etude sur l'aspect informatique, comprenant les processus de détermination des besoins et la spécification d'une solution informatique.

33.L'application de TAREFA à PLAHE-SOHO

L'objectif de l'étude de cas est de déterminer les besoins d'un outil informatique pour assister le groupe pendant ses activités de planification hebdomadaire d'opérations scientifiques du satellite SOHO.

33.1La macroactivité d'Analyse

Comme nous l'avons vu dans le chapitre 5, l'Analyse a pour but l'investigation et la

représentation des connaissances du domaine du problème. Cela est fait par l'intermédiaire de 3 activités :

1. la sélection des 'acteurs' participants au processus d'Analyse ;

2. la définition du problème à résoudre avec le développement du système (le *problem design*) ;

3. la compréhension de la situation concernée qui est à l'origine des motivations pour la proposition d'un système futur.

L'application de l'Analyse à l'étude de cas PLAHE-SOHO est montré dans les sections suivantes.

33.1.1Sélection des acteurs participants

Les acteurs participants du processus d'IB de l'étude de cas SOHO sont :

- l'équipe de conception, composée de 2 analystes informatiques et un concepteur de système informatique ;

- le groupe des utilisateurs 'finaux', qui sont essentiellement des astrophysiciens jouant un ou plusieurs rôles, comme les rôles de représentants des équipes instruments, de représentants des centres d'opération et des centres de contrôle de la mission.

Les participants typiques des réunions de simulations ont été 1 analyste , le(s) SOC(s), le(s) SOL(s), les PI de chaque équipe instrument, et l'opérateur de l'éditeur de plans.

33.1.2Définition du Problème

Notre objectif étant de concevoir une solution informatique d'assistance au groupe dans les réunions de planifications hebdomadaires, le problème est défini de la façon suivante :

Définir *un outil informatique* pour *assister le groupe* pendant ses *activités dans les réunions de planification hebdomadaire de SOHO* devant être réalisés par *les participants des réunions*.

33.1.3Compréhension de la Situation

La compréhension de la situation est faite par l'identification, la détermination et la représentation de l'Information Contextuelle et implique d'utiliser les techniques de recueil (TBC, TBE et TBO) pour obtenir les informations et d'instancier des modèles à partir de ces informations. Les paragraphes suivants décrivent l'utilisation des techniques de recueil (paragraphe 3.1.3.1) et montrent des extraits de chacun des modèles produit pour la compréhension de la situation. Nous rappelons que les modèles de l'Information Contextuelle sont :

- le modèle ontologique du vocabulaire de l'univers (paragraphe 3.1.3.2);

- les modèles des tâches minimales des utilisateurs dans l'univers (paragraphe 3.1.3.3);

- le modèle du contexte organisationnel de ces tâches (paragraphe 3.1.3.4);
- le modèle des objets métiers de l'univers (paragraphe 3.1.3.5);
- le modèle de plateforme (paragraphe 3.1.3.6).

33.1.3.1Techniques de recueil utilisées

Tout le long de cette première macroactivité dédiée à l'étude et à la compréhension du problème, nous avons recueilli une grande quantité d'information par l'intermédiaire de differentes techniques :

- Techniques basées sur communication (TBC), à savoir des interviews et des conversations informelles avec les membres du groupe ;
- Techniques Basées sur l'Etude, à savoir l'étude des documents de la mission SOHO et l'analyse du système existant, en particulier de l'outil éditeur mono-utilisateur de plannings ;
- Techniques Basées sur l'Observation, à savoir l'immersion, l'utilisation de l'enregistrement vidéo, et l'observation directe ;

La caractéristique la plus importante de la macroactivité d'Analyse de cette étude de cas est la *longue étape d'immersion* d'un analyste (Narjes Bellamine, chercheur et thésard du LIS à l'époque) à l'environnement de travail au laboratoire -IAS[18]. Ce laboratoire IAS est fortement impliqué dans SOHO : il est impliqué dans quatre instruments SOHO, et abrite le centre d'opération MEDOC. L'immersion de l'analyste et a été une étude de type ethnométhodologique qui faisait partie de sa thèse et a durée 2 ans et demi (de Février 1994 à Avril 1997). Pendant ce temps, l'analyste a eu l'occasion de connaître l'environnement de travail (les règles, la politique, les relations sociales, etc,), de discuter avec des astrophysiciens, futurs acteurs à la planification, et de comprendre leurs besoins.

En plus, dans le cadre de la préparation et de l'entraînement pour l'exploitation de SOHO, le groupe SOHO a organisé différentes simulations de réunions de planification. Les réunions simulées sont du type hebdomadaire, et quotidien. L'analyste a été invité pour une série de 3 simulations de réunions de planification avant le lancement du satellite. En particulier :

- l'analyste a participé aux Réunion de Simulation 1 (SIM1) et Réunion de Simulation 2 (SIM2) comme observateur, et des copies de la plupart des documents distribués et utilisés pendant ces réunions ont été obtenues ;
- la Réunion de Simulation 3 (SIM3) a été enregistrée en vidéo, en plus de la participation de l'analyste et des discussions avec les participants.

Pendant les deux premières participations (SIM1 et SIM2), l'analyste s'est contenté d'observer le groupe, sans intrusion, prendre des notes personnelles, et avoir une copie des documents distribués. Pendant la troisième participation (SIM3), les réunions ont été filmés et en plus l'analyste demandait des explications aux membres en faisant attention à ne pas gêner le déroulement de la réunion. Ces simulations ont fourni beaucoup d'information pour l'analyse de l'existant (par exemple la structure et la

18IAS : Institut d'Astrophysique Spatiale

dynamique des réunions) et l'identification des dysfonctionnements.

Un exemple d'un extrait du déroulement de la réunion simulée SIM3 est montré ensuite. Cet extrait a été obtenu à partir des informations recueillis par l'observation pendant la réunion, et à partir des informations additionnelles recueillies après, en regardant la vidéo enregistrée.

Déroulement d'une réunion de planification hebdomadaire -7 Août 1995-
Lieu : EAF -Experimenter's Analysis Facility-
Participants : PS -Project Scientist- , SOC -Science Operations Coordinator-, SOL -Science Operations Leader- , représentants du FOT -Flight Operations Team-, représentants des instruments,
Président : SOL
Déroulement : en 2 parties -1h45 la matinée et idem l'après-midi-
Outils utilisés : rétroprojecteur, terminal X, tablette LCD, pointeur laser
Logiciels utilisés : outil de planification développé par l'équipe CDS -Coronal Diagnostic Spectrometer-, outil de pointage et de manipulation d'image "image_pointing_tool", et netscape
Répartition des participants dans la salle de réunion : les places n'étaient pas affectées avant la réunion

Quand ?	***Qui et Quoi ?***	***Comment ?***	***Commentaires***
	Début du "Weekly Planning Meeting"		
10:05	***SOL** : Introduction de l'agenda prévu pour la réunion hebdomadaire de planification*	*Rétroprojecteur Transparent*	*cet agenda était déjà disponible sur le web*
	1. OPERATIONS CONSTRAINTS		
10:06	***Représentant du FOT** : présentation du "DSN -Deep Space Network- schedule" de la semaine*	*Oralement et en faisant référence à celui distribué sur le web*	*calendrier de contact avec le satellite*
	3. STATUS OF THE SUN		
10:44	*SOL :"Status of the SUN"*		*2ème point de l'agenda*
10:44	*SOC : utilise l'outil de pointage pour projeter des images venant d'observatoires sols ou d'autres missions, projection d'une image, …*	*Terminal X, tablette LCD, pointeur laser, outil de pointage des images*	*une seule image est projetée à la fois, il n'est donc pas facile de comparer les images entre elles !*
10:52	*J. G.(EIT), B. T.(CDS), J. S. (MDI) discutent de l'image projetée : demandent à l'opérateur de faire faire une rotation du soleil, de montrer une autre image pour compléter les informations, …*		*L'assemblée, ou le groupe constitué de tous les participants, s'est "divisé" en petites équipes (plusieurs discussions dans la même salle)*
11:30	*Ph. L. (SUMER) refait remarquer que le temps partagé entre SUMER et EIT n'est pas représenté sur le plan projeté. Il explique que c'est une contrainte très intéressante …*		*demande de précision et argumentation de la demande*

Figure 8.4 : extrait du scénario de la réunion de planification hebdomadaire du 7/8/95

Dans la figure 8.4, nous enregistrons la chronologie et l'évolution de cette réunion. Les trois premières colonnes sont remplies en regardant la vidéo. Le contenu de la dernière colonne complète les précédentes pour une meilleure description du cas : ces informations sont issues de l'observation, des connaissances antérieures de l'analyste sur le contexte , et des discussions ultérieures de l'analyste avec quelques participants à cette réunion.

Cependant cette information était parfois redondante et souvent écrite en langage naturel, avec peu voire aucune abstraction, et représenté sous forme de documents écrits ou manuscrits et sous forme d'enregistrements vidéos. Le processus d'organisation des

informations recueillies est présenté dans [Bellamine 96]. Nous utilisons surtout ces informations organisées pour la construction des modèles de la macroactivité d'Analyse de TAREFA, décrits ensuite.

33.1.3.2Modèle Ontologique

PLAHE-SOHO fait partie d'un domaine très spécifique, les missions spatiales, avec beaucoup de concepts qui n'ont pas d'équivalents dans d'autres domaines plus conventionnels. En fait, dans PLAHE-SOHO un langage très particulier est utilisé par les participants du groupe et même après quelques mois d'immersion, l'analyste avait encore besoin d'établir un petit dictionnaire de ces concepts pour comprendre les informations lues ou écoutées. Le modèle ontologique, même ses premières versions plus primitives, a servi comme dictionnaire, et était souvent consulté par l'analyste pendant les conversations avec le groupe et il a été augmenté et raffiné au fur et à mesure.

Un extrait du modèle ontologique du domaine PLAHE-SOHO est montré dans la figure 8.5.

Signs description (extrait)
sign : JOP/Joint Observing Plan
notions
PLAN D'OBSERVATIONS COORDONNEES qui décrit la contribution de chaque INSTRUMENT SOHO.
PLAN qui précise l'éventuelle COORDINATION avec des OBSERVATIONS SOLS et/ou d'autres MISSIONS SPATIALES.
impacts
La PREPARATION d'un JOP commence avant le DEBUT DE LA MISSION.
La VALIDATION d'un JOP se fait au cours de REUNIONS du SPWG.
...
sign : INSTRUMENT
notions
Appareil embarqué sur le SATELLITE capable d'effectuer un type d'OBSERVATIONS bien précis.
Appareil qui fait partie de la CHARGE UTILE du SATELLITE.
Il existent trois TYPES d'INSTRUMENTS : INSTRUMENTS CORONAUX, INSTRUMENTS HELIOSISMOLOGIQUES et INSTRUMENTS PARTICULES.

Impacts
Un PROGRAMME INDIVIDUEL décrit les OPERATIONS de l'instrument pour la réalisation d'un BUT défini.
Un instrument réalise des ACTIVITES D'OBSERVATION PRE-DEFINIES ou ACTIVITES SPECIALES.
Un instrument a un MODE D'OPERATIONS spécifique à chaque ACTIVITE.
Il y a différents MODES DE COMMANDE de l'instrument, en fonction du TYPE D'INSTRUMENT.
...

Figure 8.5 - Extrait du modèle ontologique relatif au domaine PLAHE-SOHO

33.1.3.3Modèles des tâches Minimales des utilisateurs

Dans la description des tâches existantes dans le domaine PLAHE-SOHO, il faut remarquer la différence entre l'activité de planification (voir figure 8.6) et les activités de la réunion de planification (voir figure 8.7). En fait, certaines tâches sont réalisées pendant la réunion pour assurer son contrôle (par exemple, assurer continuité entre les réunions précédentes), la sauvegarde d'informations (par exemple, l'édition du plan) et la diffusion d'informations (par exemple, diffuser le plan décidé). Ces tâches sont spécifiques de la réunion et ne font pas partie des tâches de la planification elle même.

Le modèle de tâche minimale de la planification hebdomadaire des opérations SOHO est montré sur la figure 8.6 et suit le schéma basique des planifications trimestrielle, mensuelle et quotidienne de SOHO.

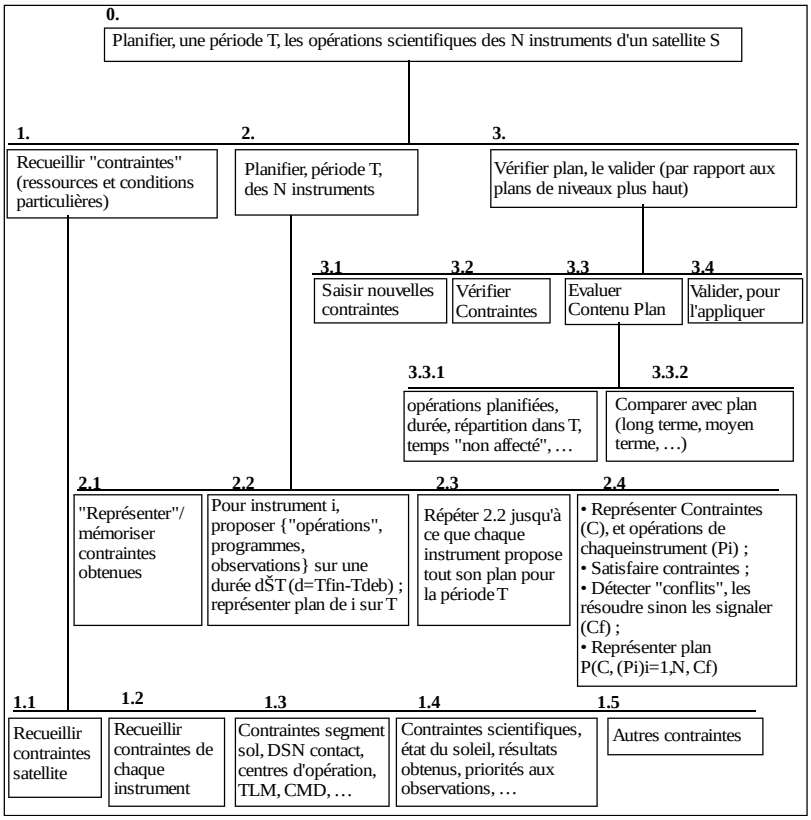


Figure 8.6 - Modèle de Tâche Minimale de Planification Hebdomadaire des Opérations de SOHO (S = SOHO, T = 1 semaine, N = 11)

La réunion de planification étant une tâche de groupe, la description des activités (et déroulement) de la réunion montrée sur la figure 8.7 ne tient pas compte des acteurs ni des rôles nécessaires pour la réalisation de chaque activité.

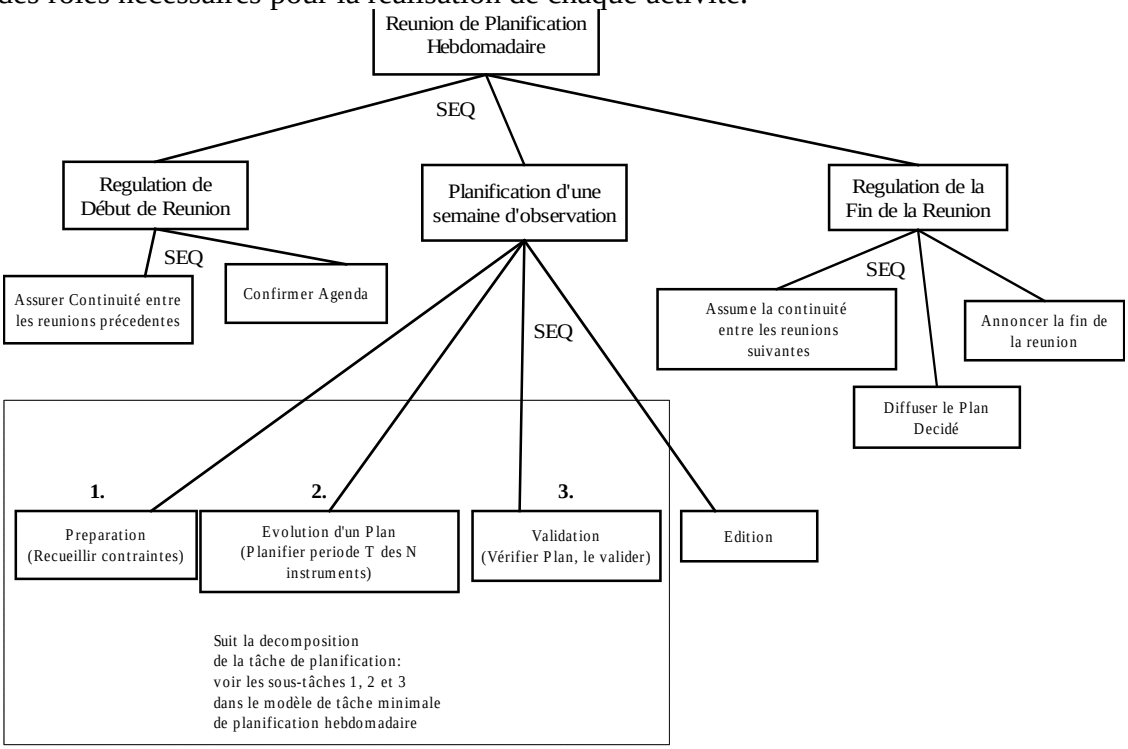


Figure 8.7 - Modèle de Tâche Minimale des Reunions de Planification

La décomposition de ces activités en tâches et sous-tâches n'aide pas à comprendre les rôles joués par chaque acteur de la réunion. Par conséquent, nous identifions les rôles des participants à la réunion et acteurs dans le processus de planification d'une part, nous identifions les unités fonctionnelles à partir des données brutes d'autre part.

Nous appliquons ensuite les méthodes de description de l'activité de réunion orientée-acteur et orientée-collectif de [Anne et al. 94], conçues pour la description des tâches de groupe. Nous présentons ainsi un regroupement par blocs fonctionnels orienté-acteur pour chaque rôle, et un regroupement par blocs fonctionnels orientés-collectifs pour la totalité de la réunion.

La figure 8.8 montre le modèle de tâche minimale pour les activités du leader pendant la réunion.

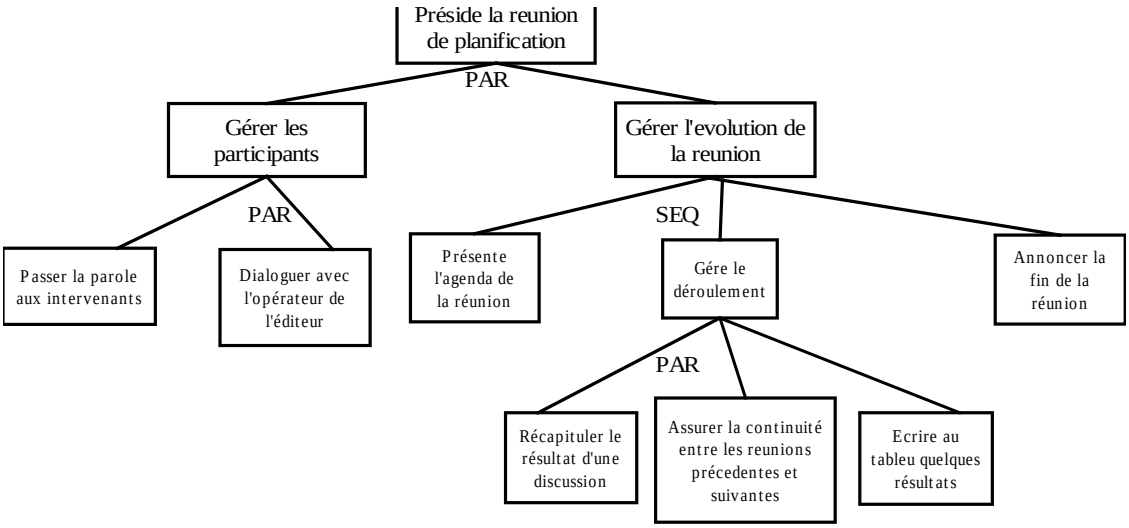


Figure 8.8 - Modèle de tâche minimale des activités du leader pendant la reunion de planification

Chaque instrument embarqué à bord de SOHO est représenté pendant la réunion par une équipe composée de une à plusieurs personnes. Dans une telle équipe, il existe a priori une personne responsable porte parole de l'équipe instrument dans son ensemble.

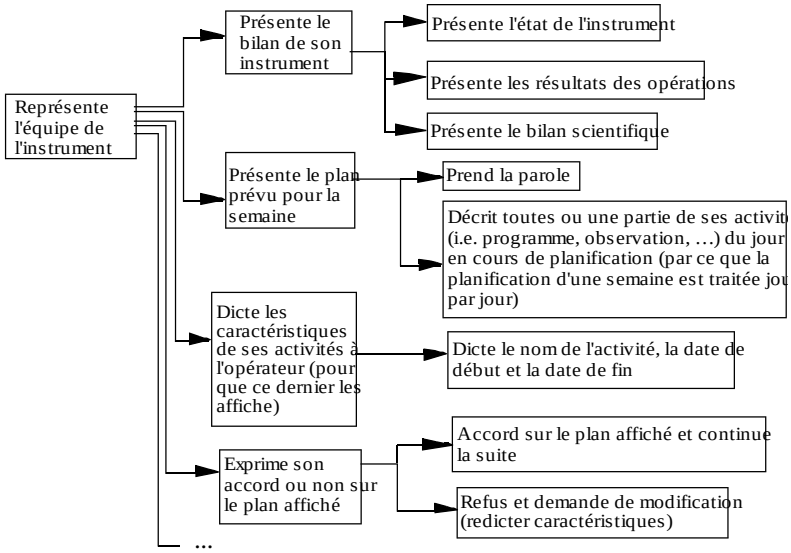


Figure 8.9 - Modèle de tâche minimale des activités d'un representant d'un instrument pendant la reunion de planification

Lorsque le PI est présent, c'est souvent lui qui assure ce rôle. La figure 8.9 montre le

modèle de tâche minimale correspondant au rôle d'un représentant d'un instrument.

Un ou deux coordinateurs participent à la réunion de planification et assistent le groupe pour obtenir des renseignements complémentaires techniques et scientifiques utilisés pour les activités de planification. La figure 8.10 montre le modèle de tâche minimale pour les activités du rôle de coordinateur de réunion de planification.

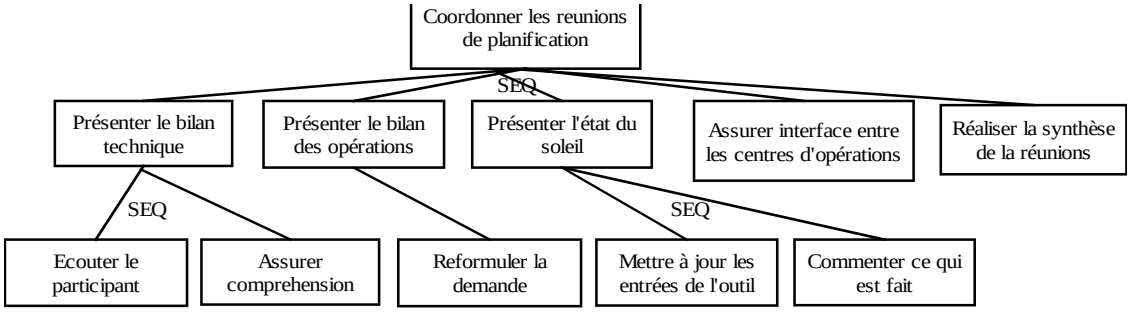


Figure 8.10 - Modèle de tâche minimale d'un coordinateur de reunion de planification hebdomadaire.

Pendant les réunions de planification, un opérateur utilise l'éditeur de plannings, un outil mono-utilisateur existant. Cet opérateur est unique pour tous les instruments et cela impose une séquentialité au travail du groupe : toutes les demandes et entrées doivent être dirigées sur lui, ce qui permet de constater une surcharge d'activités. Outre la surcharge de l'opérateur, certains problèmes de communication sont remarqués, dus aux différents niveaux des ordres (par exemple, depuis un ordre complexe comme l'insertion d'une contrainte à un ordre simple comme le changement d'une date) ou au fait que l'opérateur n'est pas toujours le même le long de toute la réunion. La figure 8.11 montre le modèle de tâche minimale des activités du rôle de l'opérateur de l'éditeur de planning.

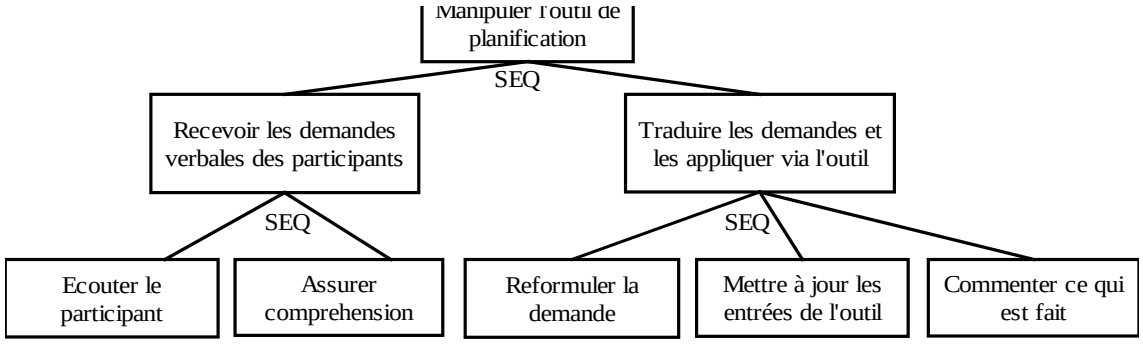


Figure 8.11 - Modèle de tache minimale des activités de l'operateur de l'outil editeur de plannings existant

33.1.3.4Modèle du Contexte Organisationnel

Comme le modèle TOCO complet pour PLAHE-SOHO est très étendu, la figure 8.12 montre un extrait du modèle de contexte organisationnel TOCO de l'étude de cas SOHO. Par simplicité, l'extrait ne montre que les éléments de l'espace de travail PLAHE-SOHO, sans considérer la planification quotidienne souvent associée à la planification hebdomadaire. Les autres espaces de travail du domaine de l'exemple, comme par exemple l'espace Planification Trimestrielle et l'espace Planification Mensuelle ne sont pas considérés.

Composant du contexte Organisationnel	**Espace de Travail** : PLAHE-SOHO (Planification Hebdomadaire des Opérations Scientifiques de SOHO)
Objectifs Organisationnels	O1. Définition des plans d'opérations des instruments de la charge utile, couvrant tous les instruments de SOHO et la période prévue O2. Coopération entre les scientifiques pour la définitions de plans qui soient les résultats de prise de décision collective avec accord de tous les participants ....etc.....
Rôles Organisationnels	R1. PI (générique) R7. SOC R27. Equipe-GOLF R2. CoI R8. SDC R28. Equipe-MDI R3. AS R9. OEP R29. Equipe-VIRGO R4. GI R10. SWT R30. Equipe-CDS R5. PS R11. SOT R31. Equipe-EIT R6. SOL R12. SPWG R32.Equipe-LASCO R13. FOT R14. Leader (de la réunion) R33. Equipe-SUMER R15. PI-GOLF R16. PI-MDI R34. Equipe-SWAN R17. PI-VIRGO R18. PI-CDS R35. Equipe-UVCS R19. PI-EIT R20. PI-LASCO R36. Equipe-CELIAS R21. PI-SUMER R22. PI-SWAN R37. Equipe-CEPAC R23. PI-UVCS R24. PI-CELIAS R38. Tout le Groupe R25. PI-CEPAC R26. Leader d'un JOP R39. Participants-clés (set of R14, R6,R7, R13, R5, R15 à R25) ...etc.....
Règles Organisationnelles	U1. Le groupe est irremplaçable pour l'activité de planification : il n'y pas de stratégie stable à suivre pour créer un plan d'observations. U2. Les scientifiques SOHO ont besoins de se réunir pour travailler ensemble et coordonner leurs opérations et observations. U3. Le groupe SOHO, et surtout les PI des instruments, veulent garder la contrôle de leurs instruments et maîtriser en continu l'exploitation de ceux-ci. U4. Les décisions et les choix doivent être faits par les scientifiques parce qu'ils dépendent de l'état du soleil qui est a priori imprévisible. U5. Le leader doit présider la réunion et gérer les participants, y compris organiser la réunion selon un agenda, passer la parole aux intervenants, synthétiser périodiquement les résultats partiels d'une négociation, demander aux participants d'approfondir certains points : le leader exerce un type de contrôle centralisé selon l'ordre du jour prévu mais certains points non prévus peuvent être discutés lorsqu'ils apparaissent pendant les discussions du groupe. U6. Certains participants peuvent jouer plusieurs rôles. U7. Il n'y a aucune hiérarchie entre les membres du groupe. U8. Le leader d'un JOP est a priori la personne qui historiquement l'a proposé a tous les instruments et géré la coordination entre certains membres de ces équipes pour spécifier leurs contributions à l'objectif principal du JOP : il joue le rôle d'informateur et de personne qui defend le JOP. U9. Si le PI est présent, il est le représentant de l'équipe de son instrument. U10.Il faut enregistrer les résultats partiels (plans préliminaires, arguments, etc) de la négociation, pour éviter qu'ils soient effacées par manque de place aux tableaux de la salle. ...etc..
Activités Organisationnelles	A1. Régulation du Début de la réunion. A2. Présentation de contraintes techniques du satellite A3. Présentation du bilan d'un instrument A4.Présentation du bilan des opérations A5. Présentation de l'état général du soleil A6. Proposition d'un Plan préliminaire d'un instrument A7.Negociation A8. Validation A9. Modification d'un Plan A10. Proposition de l'agenda A11. Modification de l'agenda A12. Diffusion de l'information A13 Régulation de la fin de la réunion A14Proposition de JOP A15 Définition du Plan Final A16 . Evaluation d'un Plan A17. Synthèse d'une négociation A18. Sauvegarde d'informations de la réunion ...etc...
Evénements Organisationnels	E1. Tous les participants clés sont présents E2. Début de la réunion E3. Participant Clé n'est pas présent E4. Agenda de la réunion est prêt E5. Agenda de la réunion n'est pas prêt E6. Bilan de l'instrument est présenté

	E7. Bilan du soleil est présenté E8. Bilan technique est présenté E9. Bilan des opérations est présenté E10. Plan d'observation préliminaire de l'instrument proposé E11. Accord sur un plan est exprimé E12. Désaccord sur un plan est exprimé E13. Demande de modification est proposée E14.Des explications aux autres membres sont demandés E15. Argumentation d'un plan présenté E16. Information additionnelle présentée E17. Récapitulation du plan est présenté E18. Le plan final est diffusé E19. Décision d'une activité du plan est prise E20. Fin de la réunion ...etc...
Concepts du Domaine	C1. Plan_Mission (date_creation, date_debut_execution) C2.Plan_Plateforme (date_creation, auteur, date_debut_execution, durée, hoaraires_contact, horaires_reservés_FOT C3. Plan_Scientifique_SOHO(periode, date_creation, auteur, lieu, date_debut_execution, durée, instruments, contraintes) C4. Instrument (id, nom, type, objectif scientifique,mode de commande, programmes) C5. Programme (id, instrument_à_commander, date_creation, auteur, source_emd_IWS, date_debut_execution, durée, priorité, opérations) C6. Programme_Conjoint_JOP (id, titre, auteur, leader, historique, objectif scientifique, opérations) C7. Bilan_instrument (id_instrument, date_bilan, liste_opérations, objectifs réalisés, evenements_non_prevus) C8. Etat_du_soleil (date_bilan, liste_proprietés, evenements_non_prevus) ...etc....
Relation de Responsabilité *Responsable(rôle, activité)*	*Responsable (R14,A1), Responsable (R27,A3), Responsable (R14,A17),* *Responsable (R13,A2) Responsable (R1 , A6) Responsable(R38,A7)* *,etc*
Relation de Post-Condition *Post-Condition (Evénement, Activité)*	*Post-Condition (E6, A3),* *Post-Condition (E10, A6),* *Post-Condition (E18, A12),* *etc.*
Relation de ¨Pré-Condition *Pré-Condition (Evénement, Activité)*	*Pré-Condition (E1,A1), Pré-Condition (E6,A6),* *Pré-Condition (E5,A10), Pré-Condition (E12,A9),* *etc.*
Relation d'Utilisation *Utilisé-en (Concept du Domaine, Activité ou Evénement ou Règle)*	*Utilisé-en (C7, A3),* *Utilisé-en (C3, A12),* *Utilisé-en (C6, A14),* *Utilisé-en (C5, U10)* *etc.*
Relation de Motivation *Motive (Objectif, Activité)*	*Motive (O2, A7)* *Motive (O1, A6)* *etc.*
Relation de Règlement *Règle de (Règle, Activité)*	*Règle de (U9, A3)* *Règle de (U7, A7)* *Règle de (U8, A14)* *etc.*

Figure 8.12 - Un extrait du modèle de contexte organisationnel du espace de travail PLAHE-SOHO

Les objectifs organisationnels O1 et O2 expriment des propriétés de très haut niveau qui doivent être considérés pour l'assistance aux réunions de planification. Ces objectifs très génériques sont à l'origine de toutes les activités de planification.

Dans PLAHE-SOHO, les rôles organisationnels sont les rôles individuels (comme R5, R6 et R1), les groupes (comme les équipes instruments R27 à R37 ou les groupes scientifiques R11 et R12), le groupe entier (R38) et sa sélection des participants incontournables (R39). Comme il est établi par la règle U6 (à savoir, certains participants peuvent jouer plusieurs rôles), un acteur peut jouer plusieurs rôles individuels (par exemple, un PI d'un instrument peut être à la fois leader d'un JOP et leader de la réunion) ou participer à plusieurs groupes (par exemple, un GI peut faire partie des équipes R29 et R33).

Les règles organisationnelles dans le figure 8.12 concernent parfois une activité spécifique (comme la règle U10, applicable à l'activité A18 - Sauvegarde

d’informations de la réunion) parfois des règles d’ordre général (comme la règle U6).

Les événements E1 à E20 montrent un extrait de situations qui peuvent arriver une fois (comme E2, E20 et E18) ou plusieurs fois (comme E6, E10 ou E15) dans une réunion.

Les concepts du domaine C1 à C8 sont un extrait des concepts considérés nécessaires par l’organisation pour l’espace de travail PLAHE-SOHO. Ces concepts sont naturellement utilisés par les descriptions des activités organisationnelles, des règles organisationnelles et des événements organisationnels dans le même espace de travail.

Les relations montrées dans la figure 8.12 sont des exemples de relations intracontextuelles entre les éléments du contexte organisationnel pour PLAHE-SOHO. Par exemple, Responsable (R14, A1) signifie que le rôle R14 (le leader de la réunion) est responsable pour l’activité A1 (Régulation du début de la réunion).

33.1.3.5Modèle des objets métiers

Les données essentielles manipulées et produites le long du processus de planification sont les informations de bilan et les plans. Dans le cas de PLAHE-SOHO, nous avons réutilisé les objets disponibles dans un modèle du domaine existant construit par [Bellamine 96]. Une hiérarchie de différents types de plans et opérations d’intruments ainsi que leurs structures sont montrés sous la figure 8.13, en utilisant la notation OMT [Rumbaugh 91].

Figure 8.13 - Les objets métiers du domaine PLAHE-SOHO : un modèle des objets Plans et Opérations en utilisant la notation OMT

33.1.3.6Modèle de plateforme

L'extrait du modèle de plateforme de PLAHE-SOHO montré dans la figure 8.14 est constitué d'un ensemble d'équipements, dispositifs matériels et logiciels disponibles actuellement pour les réunions de planification.

A chaque élément du modèle est attribué un identificateur (Id) et une description succincte est faite (voir figure 8.14).

- EQUIPEMENT
  - Rétro-projecteur
  - Tableaux de la salle
  - Transparents d'images du soleil

- MATERIEL
  - Terminal X
  - Tablette LCD
- LOGICIEL
  - Outil de Pointage
  - Netscape
  - UNIX
  - IDL (Interactive Data Language)
  - Editeur monoutilisateur graphique de Planning ‘Sohoparr’

Id	Description
P1	Rétro-projecteur
P2	Tableaux de la salle
P3	Transparents d’images du soleil
P4	Terminal X
P5	Tablette LCD
P6	Outil de Pointage
P7	Netscape
P8	UNIX
P9	Environnement IDL (Interactive Data Language)
P10	Editeur mono-utilisateur graphique de Planning ‘Sohoparr’

Figure 8.14 - Modèle de Plateforme de PLAHE-SOHO

33.1.4Déterminer les Besoins de l’Utilisateur

Dans le domaine SOHO, les différentes techniques de recueil utilisées ont permis d’identifier des besoins importants. En fait, les besoin demandés explicitement par les participants (BUEs) n’étaient pas nombreux et ils ont été complétés par les besoins implicites (BUIs) déterminés par l’immersion de l’analyste et les besoins du contexte de l’organisation (BUOs). Ces besoins sont présentés dans les sections suivantes.

33.1.4.1Formulation de la Liste Informelle des Besoins (BUEs)

D’abord il faut remarquer que, d’entre tous les participants des réunions, il y a eu 4 acteurs qui sont à l’origine des besoins demandés explicitement (appelés ici acteur #1, acteur #2, acteur #3 et acteur #4) et qui correspondent respectivement aux participants qui jouent souvent les rôles de SOL (acteur#1), CoI-SUMER (acteur#2), PI-EIT (acteur#3), et JOP2-leader (acteur#4). En général, ils étaient les porte-paroles de plusieurs participants : ils exprimaient leurs besoins individuels conjointement aux besoins de leurs équipe.

En plus, le nombre des besoins est petit, ce qui confirme notre présupposition de la difficulté des utilisateurs à exprimer leurs besoins sans une période d ‘expérimentation dans une situation réelle d’utilisation.

Un extrait des besoins demandés explicitement (BUEs) est montré ensuite.

- BUE#1. Réduction de la durée de la réunion. Origine : Acteur#1, Acteur#2, Acteur#3 et Acteur#4.

- BUE#2. Assistance à l'intégration des suggestions et à l'affichage. Origine : Acteur#1 et Acteur#4.

- BUE#3. Consistance des informations disponibles à tous , qui doit être l'information validée la plus actuelle, pour suivre la réunion (par exemple, résultats partielles pendant la réunion) et pour garder des résultats finaux (après la réunion). Origine : Acteur#1, Acteur#2, Acteur#3 et Acteur#4.

- ....etc.....

Les activités des réunions sont basées fortement sur la parole et la communication face-à-face qui facilitent l'expression des points de vue et les échanges. Cependant, la difficulté de garder une trace de ce qui est écrit ou dit est à l'origine de plusieurs problèmes de contrôle de la réunion et consistance des informations. C'est pourquoi ces besoins reflètent la nécessité du groupe d'optimiser le déroulement des réunions (BUE#1) et de permettre le partage des informations (partielles ou finales) par les participants (BUE#2 et BUE#3).

33.1.4.2Associer les BUEs aux activités organisationnelles

D'entre les activités organisationnelles décrites dans le modèle de contexte, nous avons identifié lesquelles seront considérées comme candidates pour les modifications afin de satisfaire les besoins initiaux des utilisateurs.

BUEs	**Activités Organisationnelles Associées**
BUE #1	A2. Présentation de contraintes techniques du satellite A3. Présentation du bilan d'un instrument A4.Présentation du bilan des opérations A5. Présentation de l'état général du soleil A6 Proposition d'un Plan préliminaire d'un instrument A7.Negociation A11. Modification de l'agenda A14Proposition de JOP A15 Définition du Plan Final A17. Synthèse d'une négociation A18. Sauvegarde d'informations de la réunion
BUE #2	A6. Proposition d'un Plan préliminaire d'un instrument A9. Modification d'un Plan A12. Diffusion de l'information A14 Proposition de JOP A17. Synthèse d'une négociation A18. Sauvegarde d'informations de la réunion
BUE#3	A2. Présentation de contraintes techniques du satellite A3. Présentation du bilan d'un instrument A4.Présentation du bilan des opérations A5. Présentation de l'état général du soleil A6. Proposition d'un Plan préliminaire d'un instrument A9. Modification d'un Plan A10. Proposition de l'agenda A11. Modification de l'agenda A12. Diffusion de l'information A14Proposition de JOP A17. Synthèse d'une négociation A18. Sauvegarde d'informations de la réunion

Figure 8.15- Association des BUE aux Activités Organisationnelles de PLAHE-SOHO

Nous pouvons remarquer que ces besoins sont associés aux activités d'échange et d'affichage d'informations et non aux activités de contrôle de la réunion. Cela reflète

clairement l'opinion général du groupe de maintenir l'aspect coopératif des discussions et des prises de décisions, voir les règles organisationnelles U1, U2, U3 et U4 dans la figure 8.12.

33.1.4.3Identifier les problèmes non-mentionnés (BUI) et vérifier les contraintes/règles (BUO)

En plus des besoins demandés par les utilisateurs, les techniques d'observation nous ont permis de repérer plusieurs problèmes et dysfonctionnements dans le déroulement des activités des réunions de PLAHE-SOHO. Outre l'observation de l'analyste qui a réalisé l'immersion, une étude de la vidéo enregistré (3h 30 minutes!) nous montre l'apparition de quelques problèmes supplémentaires. En particulier, [Bellamine 96] décrit le travail de transcription de dialogues et extraction de séquences vidéo pour tracer le processus de planification réalisée dans la réunion simulée et identifier les dysfonctionnements.

Les besoins identifiés (BUIs) sont les suivants :

- BUI#1 Résoudre le problème de la surcharge de l'opérateur de l'outil de planification existant (alléger, voire annuler la charge du travail de l'opérateur de plan unique)
- BUI#2 Résoudre le problème de division fréquente du groupe en petits groupes, besoin de discuter avec les membres de la même équipe instrument et besoin de discuter avec les membres d'une autre équipe instrument
- BUI#3 Résoudre le problème d'absence de diffusion d'information commune aux participants
- BUI#4 Résoudre le problème de perte d'informations non sauvegardées . Ces informations appartiennent à 3 catégories principales :
    - contraintes générales sur les programmes coordonnés
    - caractéristiques des programmes conjoints
    - schémas et dessins explicatifs

Ces besoins sont associés à des problèmes typiques de PLAHE-SOHO, décrits avec plus de détails dans [Bellamine 96]. Par exemple, le problème de surcharge de l'opérateur se produit lorsque l'opérateur manipule seul son outil éditeur : les représentants de l'instrument attendent leur tour pour dicter tout ou une partie de leurs plans ; outre la séquentialité de la dictée, certains problèmes de communication sont remarqués, parce que l'opérateur actuel n'est pas un astrophysicien et il ne connaît pas tous les détails et concepts des plans. En fait, l'opérateur est un rôle artificiel parce qu'il a été rajouté par la nécessité de manipuler l'outil éditeur mono-utilisateur et il n'est pas primordial pour le travail du groupe (ce rôle n'existait pas pendant la réunion simulée SIM2).

Les besoins de l'organisation (BUOs) ont été identifiés par l'analyse du contexte et les divers interviews (informelles) avec le participant qui joue souvent le rôle de leader des réunions. Un extrait des besoins de l'organisation est présenté ensuite.

- BUO#1 Assister la coopération entre les membres
- BUO#2 Assister une planification entre groupe distribué (personnes éloignées)

Ces besoins sont aussi présentés avec plus de détails dans [Bellamine 96]. Par exemple, l'assistance à la planification entre groupe distribué peut faire gagner du temps et de l 'argent en évitant les déplacements pour les réunions face-à-face.

Un extrait de l'association des besoins implicites (BUIs) et des besoins imposés par l'organisation (BUOs) ci-dessus aux activités de PLAHESOHO est présenté dans la figure 8.16. Les cellules en blanc signifient qu'il n'y a pas besoins du type correspondent (BUI ou BUO) identifiés pour cette activité.

Activités Organisationnelles	**Besoins Implicites (BUIs)**	**Besoins Imposés par l'Organisation (BUOs)**
A2	BUI#3	
A3	BUI#3	
A4	BUI#3	
A5	BUI#3	
A6	BUI#1, BUI#2	
A7	BUI#2	BUO #1
A9	BUI#1	
A12	BUI#3	BUO #2
A15	BUI#1	
A16	BUI#2	
A17	BUI#3	
A18	BUI#1,BUI#4	BUO #2

Figure 8.16 - Extrait des besoins implicites (BUIs) et des besoins imposés par l'organisation (BUOs) de PLAHE-SOHO

L'association entre les besoins et les activités a été faite par les analystes. Il y a associations immédiates (voir les besoins associés à l'activité A18 - Sauvegarde d'informations de la réunion) mais aussi des associations moins évidentes , comme les besoins associés à l'activité A7 - Négociation. En fait, dans ce cas là, les besoins BUI#2 et BUO#1 sont plutôt associés à la stratégie de négociation adoptée (distribuée, centralisée, avec ou sans un facilitateur, etc) et aux supports de communication disponibles.

33.2La macroactivité de Synthèse

La macroactivité d'Analyse réalisé dans l'univers SOHO et décrite dans la section précédente nous a permis de comprendre l'univers du domaine du problème et d'identifier les besoins (explicites et implicites) des utilisateurs. Nous allons dans cette section décrire la réalisation de la macroactivité de Synthèse , ayant pour but la détermination des besoins d'un outil d'assistance aux réunions de planifications hebdomadaire des opérations de SOHO. Nous commencerons à présenter la définition des besoins initiaux du système (section 3.2.1). Nous présenterons ensuite le processus itératif et interactif de détermination des besoins du système (section 3.2.2), composé par les sous activités de construction des cas d'utilisation, d'expérimentation des cas d'utilisation, de modification des cas d'utilisation et d'intégration des cas d'utilisation. Dans la section 3.2.3 nous présenterons le modèle des besoins résultant de l'activité d'intégration des besoins du système et dans la section 3.2.4 la modélisation du *design rationale*.

33.2.1Définition des Besoins initiaux du Système

Comme nous l'avons vu dans le chapitre 6, la définition des besoins initiaux du système à partir des besoins de l'utilisateur consiste en deux activités complémentaires : la Re-ingénierie des tâches et l'allocation des fonctions. Nous allons présenter ensuite la réalisation de ces activités dans l'étude de cas PLAHE-SOHO.

33.2.1.1Re-ingénierie des Tâches

Le but de la re-ingenierie des tâches est de remodeler un modèle de tâche minimale à partir des règles du contexte. La manière la plus courante de réaliser la re-ingénierie est d'identifier comment les activités/tâches peuvent devenir plus faciles, en supprimant les contraintes non-nécessaires (p.ex. séquences imposées ou habituelles).

Un exemple de suppression des contraintes non-necessaires est la re-ingénierie des tâches du modèle de tâches minimales 'Réunion de Planification' (voir figure 8.7), en particulier la sous-tâche 'Planification d'une semaine d'observation', où les sous-buts 'Evolution d'un Plan', 'Validation' et 'Edition' ne sont pas nécessairement séquentiels. En fait, pour des raisons 'historiques' les points de l'agenda prévu pour la réunion sont traités séquentiellement. Dans la figure 8.7 l'évolution du plan répond au deuxième point de l'ordre du jour dans l'agenda et l'édition du planning correspond au quatrième. En effet, la règle U10 du modèle de contexte organisationnel TOCO (à savoir, 'U10. - Il faut enregistrer les résultats partiels (plans préliminaires, arguments, etc) de la négociation, pour éviter qu'ils soient effacées par manque de place aux tableaux de la salle') n'impose pas cette séquence, qui est probablement la conséquence de l'utilisation fréquente d'un outil mono-utilisateur ou de la centralisation du contrôle d'édition par le leader, pour qui l'agenda est traité de manière séquentielle et l'éditeur de planning n'intervient qu'après l'évolution du plan. En fait, les deux activités peuvent être faites simultanément. Par exemple un membre peut prendre des notes pour le groupe (sur portable, papier, etc.) ou tous les membres peuvent avoir accès à un outil partagé.

La figure 8.17 montre le modèle de tâche minimale de la tâche Réunion de Planification respectivement avant (a) et après (b) la Re-ingénierie des tâches en utilisant la notation MAD.

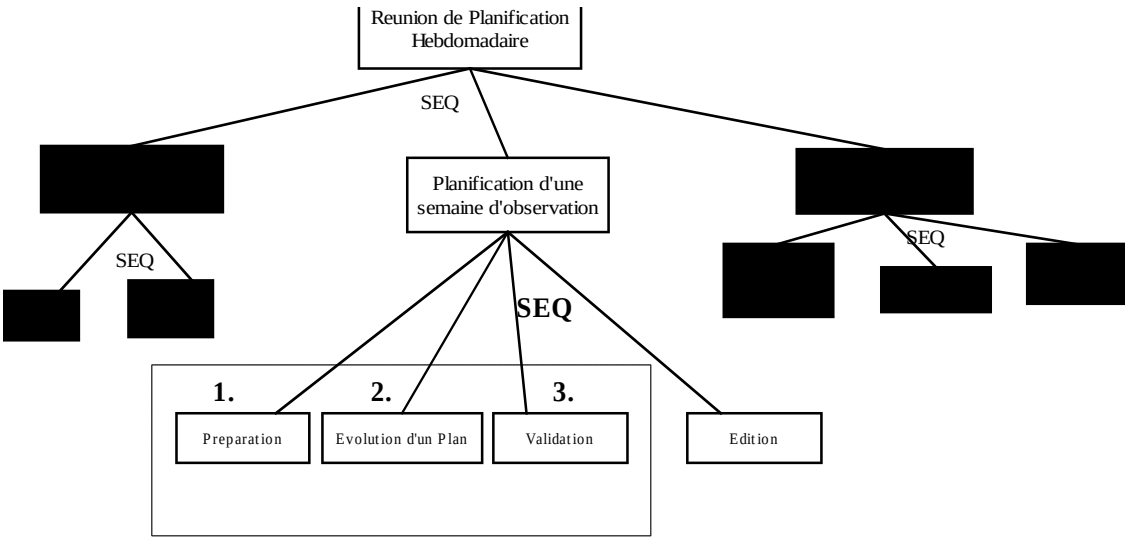


Figure 8.17(a) - Modèle de Tache Minimale 'Reunion de Planification' avant la Re-ingénierie de tâches

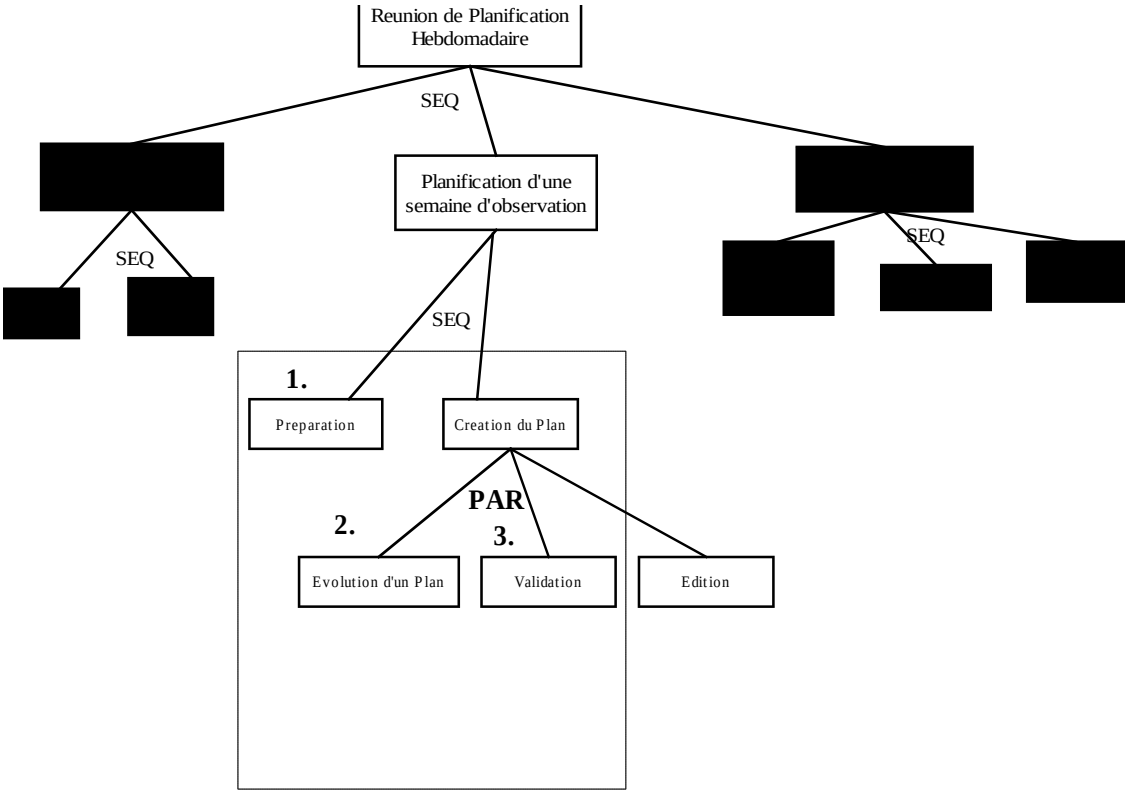


Figure 8.17(b) - Modèle de Tache Minimale 'Réunion de Planification' après la Re-ingénierie de tâches

33.2.1.2Allocation des Fonctions

Une réunion face-à-face de PLAHE-SOHO a pour but principal la planification scientifique d'une semaine d'opérations des instruments du satellite SOHO. Mais en plus elle est une occasion privilégiée, voir unique, pour les astrophysiciens se rencontrer et discuter. C'est pourquoi, le groupe ne souhaite pas être remplacé par un système totalement automatique, mais plutôt utiliser un outil d'assistance aux tâches que chaque participant réalise pendant les réunions.

Dans l'étude PLAHE-SOHO , il y a beaucoup de tâches manuelles significatives, qui font partie de la communication face-à-face. La parole, le tableau (P2 dans le modèle de plateforme) servent de support à cette communication et peuvent être retenues. Par contre , l'outil d'assistance aux réunions qui sera conçu doit remplacer l'éditeur (P10 dans le modèle de plateforme).

Un exemple d'allocation de fonctions pour la vérification de contraintes et plans est donnée dans la figure 8.18.

Alternative	Fonction de l'utilisateur	Fonction du Système	Coût	Avantage
Vérification manuelle (par le groupe) de contraintes et plans	Formulation de contraintes et plans, la saisie, la détection et l'annonce des conflits	Enregistrement des informations sur les contraintes et les plans	Chaque participant doit évaluer les contraintes, participer dans la négociation et discussion scientifique pour confirmer les propositions de plans	La créativité et la coopération du groupe sont stimulés
Vérification semi-automatique de contraintes et plans	Formulation et saisie de contraintes et plans ; Demande de vérification	Enregistrement des informations sur les contraintes et les plans ; Détection et annonce de conflits sur	• Il faut avoir au moins un expert responsable pour formuler, saisir et contrôler les règles de vérification ; A l'heure actuelle ces règles ne sont pas encore définies • Définir les notations pour	• Gagner du temps en permettant plus de discussions sur les points clés • Permettre le déroulement de la reunion même avec l'existence de conflits non résolus, • Aider les participants à valider les plans et aider les non-experts à

		demande du leader	représenter les règles • Risque de ne pas vérifier tous les plans : il faut identifier les moments de vérification de chaque plan dans la réunion • Risque de manque de coopération et créativité et de proposition de nouveaux programmes ;	prendre de bonnes décisions
Vérification automatique de contraintes et plans	Formulation et saisie de contraintes et plans	Enregistrement des informations sur les contraintes et les plans ; Détection et annonce de conflits sans l'intervention de l'utilisateur	• Il faut avoir au moins un expert responsable pour formuler, saisir et contrôler les règles de vérification ; A l'heure actuelle ces règles ne sont pas encore définies • Risque de bloquer l'évolution d'un plan si un conflit est détecté et non résolu ; • Risque de manque de coopération et créativité et de proposition de nouveaux programmes ;	Gagner du temps en permettant plus de discussions sur les points clés

Figure 8.18 - Un exemple d'Allocation de Fonctions pour la verification de contraintes et plans de PLAHE-SOHO

33.2.1.3Les Besoins Initiaux du système

La liste des besoins initiaux de l'outil d'assistance dérivés directement des Besoins de l'Utilisateur (BUEs, BUIS et BUOs) après les phases de Re-ingénierie des tâches et d'allocation des fonctions est montré dans la figure 8.19.

Besoins de l'utilisateur	Besoins Initiales du Système
• BUE#1. Réduction de la durée de la réunion.	BS#1 : Réduire la durée de la réunion
• BUE#2. Assistance à l'intégration des suggestions et à l'affichage.	BS#2 : Assister l'intégration d'affichage et représentations des propositions de plans
• BUE#3. Consistance des informations disponibles à tous , qui doit être l'information validée la plus actuelle, pour suivre la réunion (par exemple, résultats partielles pendant la réunion) et pour garder des résultats finaux (après la réunion).	BS#3 :Maintenir la consistance des informations disponibles
• BUI#1 Résoudre le problème de la surcharge de l'opérateur de l'outil de planification existant (alléger, voire annuler la charge du travail de l'opérateur de plan unique)	BS#4 Etre multi-utilisateur
BUI#2 Résoudre le problème de division fréquente du groupe en petites groupes, pour discuter avec les membres de la même équipe instrument ou pour discuter avec les membres d'une autre équipe instrument	BS#5 : Assister les équipes dans leurs discussions
BUI#3 Résoudre le problème de absence de diffusion d'information commune aux participants	BS#6 Supporter la diffusion ou l'accès des informations partagées
• BUI#4 Résoudre le problème de perte d'informations non sauvegardes. Ces informations appartiennent à 3 catégories principales : • contraintes générales sur les programmes coordonnés • caractéristiques des programmes conjoints • schémas et dessins explicatifs	BS#7 Support au stockage d'informations temporaires ou finales de la réunion
• BUO#1 Assister la coopération entre les membres	BS#8 Etre coopératif (CSCW)
BUO#2 Assister une planification entre groupe distribué (personnes éloignées)	BS#9 Etre distribué

Figure 8.19 - Liste de Besoins Initiaux du Système pour PLAHE-SOHO

33.2.2Détermination des Besoins du Système

Les sections suivantes illustrent le processus de déterminations des besoins de l'outil d'assistance à PLAHE-SOHO. Comme nous l'avons vu dans le chapitre 6, la détermination des besoins du système est composée par les sous activités de construction des cas d'utilisation (section 3.2.2.1), d'expérimentation des cas d'utilisation (section 3.2.2.2), de modification des cas d'utilisation (section 3.2.2.3) et d'intégration des cas d'utilisation (section 3.2.2.4).

33.2.2.1Construction des Cas d'Utilisation

Dans l'étude de cas PLAHE-SOHO, la construction des cas d'utilisation a suivi l'approche descendante décrite au chapitre 6, mais sans la construction des cas

d'utilisation opérationnels. En fait, comme la conception orientée objet n'est pas un but immédiat de PLAHE-SOHO, nous avons décidé de ne pas construire les cas d'utilisation opérationnels. Ainsi, le prototype développé a été conçu à partir des cas d'utilisation concrets dérivés directement des cas d'utilisation singuliers.

La construction des cas d'utilisation pour les niveaux essentiel, téléologique et concret de PLAHE-SOHO sera présentée respectivement dans les section suivantes.

33.2.2.1.1Les cas d'utilisations essentiels

Comme nous l'avons vu dans le chapitre 6, les cas d'utilisation essentiels sont construits pour les tâches interactives et automatiques et il y a donc des buts dans le modèle de tâche minimales qui ne seront pas utilisés pour construire un cas d'utilisation parce qu'ils correspondent aux tâches manuelles (par exemple, 'annoncer la fin de la réunion'). A partir du modèle de tâches minimales de la réunion de planification, nous avons défini les cas d'utilisation essentiels (CUEs) suivants :

Régulation de début de réunion
CUE#1 - Assurance de continuité entre les réunions précédentes
CUE#2 - Définition de l'agenda
Planification d'une semaine d'Observation Préparation
CUE#3 - Présentation des bilans des instruments
CUE#4 - Présentation du bilan technique de support mission
CUE#5 - Présentation du bilan des opérations
CUE#6 - Présentation de l'état du soleil
CUE#7 - Représentation des contraintes dérivées des bilans présentés
Planification d'une semaine d'Observation Evolution d'un Plan
CUE#8 - Proposition de plan préliminaire pour chaque instrument
CUE#9 - Représentation des opérations
CUE#10 - Vérification du Plan (conflits, etc)
CUE#11 - Résolution de Conflits détectés
CUE#12 - Choix de plan
CUE#13 - Justification de choix
Planification d'une semaine d'Observation Validation
CUE#14 - validation du plan
Planification d'une semaine d'Observation Edition
Régulation fin de la réunion
CUE#15 - Assurance de la continuité entre les réunions suivantes
CUE#16- Diffusion des informations

Un cas d'utilisation essentiel est décrit par une narration simple structurée en deux colonnes : une avec les formulations des utilisateurs et l'autre avec les services du système correspondants aux formulations. La description du cas d'utilisation essentiel 'Représentation des opérations d'un plan préliminaire' (CUE#9) est montré dans la figure 8.20.

Représentation des opérations d'un plan préliminaire

Tâches Minimales (Formulations)	Services du système
S'Identifier	Accepter Identification Vérifier Identification

	Vérifier si le rôle est le responsable pour cette activité
Cycle de : • Proposer nom de l'opération	Cycle de : • Accepter nom de l'opération Vérifier nom de l'opération
• Proposer Date de début (Tdeb)	• Accepter Date de début (Ddeb) Vérifier Date de début
• Proposer Date de fin (Tfin)	• Accepter Date de Fin (Dfin) Vérifier Date de Fin
• Proposer Justification/argumentation pour l'opération	• Accepter justification
Proposer Vérification de Conflit Interne du Plan	Accepter Demande de Vérification Vérifier le plan pour trouver des contradictions internes Annoncer le résultat de la vérification
Diffuser la proposition au groupe	Accepter Demande de Diffusion Vérifier si l'agenda est respecté Diffuser la proposition

Figure 8.20 : Le Cas d'Utilisation Essentiel (CUE#9) pour 'Représentation des opérations d'un plan preliminaire ' de PLAHE-SOHO

La représentation commence typiquement par une identification de son responsable (épisode 1) et continue par un cycle de propositions (épisodes 2 à 5): chaque proposition a le nom de l'opération à réaliser sur un instrument et les dates de début et fin de l'opération ainsi qu'une justification pour sa réalisation . Même s'il est fait par des astrophysiciens expérimentés, un plan proposé peut présenter des conflits internes, c'est à dire, des conflits entre les opérations du même instrument. Le terme 'interne' est utilisé par opposition aux conflits externes, qui sont des conflits entre des opérations des divers instruments et qui ne peut qu'être vérifié après la proposition des plans de tous les instruments. Ensuite, une vérification du plan d'un instrument est faite (épisode 6) après le cycle de proposition pour détecter les conflits internes. Finalement, le plan proposé doit être diffusé au groupe (épisode 7). Les services essentiels d'un outil d'assistance à ces tâches sont présentés dans la colonne droite.

La figure 8.21 montre les relations temporelles entre les formulations du cas d'utilisation essentiel 'Représentation des opérations d'un plan préliminaire'(CUE#9).

Représentation de Plan Préliminaire :
S'Identifier
((Proposer nom de l'Opération) |||
(Proposer Date de début) |||
(Proposer Date de Fin) |||
(Proposer Justification | *nil*)) $^{+}$
(Demander Vérification de Conflit | *nil*)
Diffuser Proposition au groupe

Figure 8.21 - Expression des relations temporelles entre les formulations du cas d'utilisation essentiel CUE#9

Suivant l'expression de la figure 8.21, après l'identification et la demande de proposition de plan, un cycle de proposition des composants d'un plan (nom d'opération, date de début, date de fin, justification) commence. Le cycle a au moins un ensemble de composants (voir l'exposant '+'). Les formulations optionnelles sont décrites par l'intermédiaire de l'expression (<*formulation*> | *nil*) que veut dire que soit la formulation est faite soit rien n'est fait. Après le cycle, la demande (optionnelle) de vérification des conflits internes du plan et la diffusion du plan aux membres du groupe sont faites.

33.2.2.1.2Les cas d'utilisation singuliers

Le cas d'utilisation singulier qui correspond au 'cas normal' du cas d'utilisation

essentiel CUE#9 est construit par l'intermédiaire du raffinement de ses épisodes. Un exemple d'un épisode raffiné de ce cas d'utilisation singulier est montré dans la figure 8.22.

Cas d'utilisation 'Représentation des opérations d'un plan préliminaire'
UC0 - Sans singularité - Cas Normal
Episode 7 : Diffuser la proposition au groupe
Plan :

Pre_Cond : Sans_conflits_internes (Plan préliminaire)
Initiator : PI
Action : Proposer Diffusion du plan
Post-Cond : Proposée (Diffusion) ou Non-proposée (Diffusion)

Pre-Cond : Sans_conflits_internes (Plan préliminaire)
Initiator : SOHO-outil
Action : Accepter Demande de Diffusion du plan
Post_Cond : Accepté (Demande) ou Non_Accepté (Demande)

Pre-Cond : Accepté (Demande)
Initiator : SOHO-outil
Action : Vérifier si l'agenda est respecté
Pos_Cond : Respecté (Agenda) ou Non-respecté(Agenda)

Pre-Cond : Respecté (Agenda)
Initiator : SOHO-outil
Action : Actualiser Base de Données Partagé
Post_Cond : Actualisé (Base de Données) ou Non-actualisé (base de données)

Pre-Cond : Actualisé (Base de Données)
Initiator : SOHO-outil
Action : Signaliser Diffusion de Proposition aux participants
Post-Cond : Signalisé (diffusion) ou Non-signalisé (diffusion)
Resultats : Si signalisé (diffusion) la proposition a été diffusé et les participants peuvent regarder le plan proposé

Figure 8.22 - Episode 7 du Cas d'utilisation singulier UC0 - le cas 'normal' de 'Representation des opérations d'un plan preliminaire' (CUE#9) de PLAHE-SOHO

L'épisode présenté est l'épisode 7 - 'Diffuser la proposition au groupe', dont le but est de permettre aux participants de la réunion de regarder le plan préliminaire proposé pour un instrument. Comme les réunions SOHO ont plusieurs rôles et l'utilisateur prévu pour l'outil est le groupe (et leurs rôles individuels composants), l'initiateur peut être le système (ici appelé SOHO-outil) ou les rôles des participants des réunions. Il y des conditions pour cette diffusion :

- elle ne doit être proposée que par un rôle responsable (dans ce cas, le PI de l'instrument) ;
- elle doit suivre l'ordre du jour de l'agenda ; c'est à dire, il y a un moment de la réunion qui doit être respecté pour la diffusion des plans préliminaires ;
- la diffusion est faite en deux pas : actualisation de la base de données partagées et la signalisation de la proposition aux participants .

Comme résultat de la diffusion, le plan proposé peut être regardé par les participants de la réunion.

A partir de la description du cas normal raffiné, commence l'identification des singularités associées au cas d'utilisation. Dans la figure 8.23, un extrait de l'identification des singularités du cas d'utilisation 'Représentation des opérations d'un plan préliminaire' (CUE#9) de PLAHE-SOHO est montré.

Id de la Singularité	**Action/Episode**	**Service du Système où la singularité arrive**	**Problème**	**Singularité (Cause du Problème)**	**Services (D)efensifs ou (C)orrectifs du Système**
Singularité	Diffuser la	Accepter	Sans_Conflits_internes	Discussion entre les	(c) Continuer la diffusion en

18	proposition au groupe (épisode 7)	Demande de Diffusion du plan	(plan préliminaire) est faux	membres de l'équipe instrument n'est pas finie	ajoutant l'information de qu'il y a encore des conflits internes (c) Continuer la diffusion des parties sans conflits du plan, en ajoutant qu'il a encore d'autres opérations à diffuser (d) Diffusion n'est pas permise s'il y a des conflits internes
Singularité 19	Diffuser la proposition au groupe (épisode 7)	Accepter Demande de Diffusion du plan	Rôle Invalide	La demande n'est pas faite par le rôle responsable par la diffusion	(c) Exhiber message
Singularité 20	Diffuser la proposition au groupe (épisode 7)	Vérifier si l'agenda est respecté	Non-Respecté (Agenda)	La demande ne respecte pas l'ordre du jour de la réunion	c) Exhiber message (c) Relaxer le respect à l'ordre du jour et permettre la diffusion
Singularité 21	Diffuser la proposition au groupe (épisode 7)	Actualiser Base de Données Partagé	Non-actualisé (Base de données)	Problème de SGBD	c) Répéter l'action (c) Après quelques essais (déterminer le nombre), ne plus répéter et annuler la actualisation ; la diffusion est considérée comme 'non faite' c) Exhiber message
Singularité 22	Diffuser la proposition au groupe (épisode 7)	Signaliser Diffusion de Proposition aux participants	Non-signalisée (diffusion)	Problèmes de communication du réseau	d) Répéter l'action (c) Exhiber message (c) Après quelques essais (déterminer le nombre), ne plus répéter , et annuller la signalisation ; la diffusion est considérée comme 'faite' (d) Signalisation verbale simultanée
Singularité 23	Diffuser la proposition au groupe (épisode 7)	Signaliser Diffusion de Proposition aux participants	Non-regardé(diffusion)	Participants ignorent la signalisation	

Figure 8.23 - Extrait de l'identification des Singularités pour singulier UC0 - le cas 'normal' de 'Representation des opérations d'un plan preliminaire' (CUE#9)

Dans une situation coopérative, un dysfonctionnement s'identifie lorsque les fonctions de production et/ou régulation du groupe présentent des anomalies. Les dysfonctionnements susceptibles d'apparaître dans un univers coopératif peuvent se manifester par des anomalies au niveau de la coopération et par les résultats de cette coopération. Un dysfonctionnement peut être détecté lorsque le collectif est insatisfait et lorsque le résultat de la coopération ne correspond pas aux objectifs de celle-ci. Par exemple, la singularité 1 - l'existence de conflits internes au moment de la diffusion parce que la discussion n'est pas encore finie - peut montrer la nécessité d'un "environnement privé" par équipe pour permettre le travail en aparté avant de proposer une partie du plan à tout le groupe. Cependant, cela peut avoir le risque de manque de coordination (si chaque équipe travail seule pendant longtemps, le réunion n'est plus synchrone). Une idée (proposée comme service correctif) est de diffuser l'état actuel du plan en ajoutant que les conflits internes persistent. Une autre idée (proposée aussi comme service correctif) est de diffuser la partie sans conflits du plan.

Cependant, il y a des singularités très difficiles à traiter. Par exemple, la singularité 6 - le fait que la diffusion n'est pas regardée - ne peut être résolue que par les changements des attitudes des participants.

Après l'identification des singularités, nous définissons les cas d'utilisation singuliers. Un exemple de définition des cas d'utilisation singulier pour le cas d'utilisation essentiel 'Représentation des opérations d'un plan préliminaire' (CUE#9)

de PLAHE-SOHO est montré dans la figure 8.24, d'après les singularités identifiées précédemment.

Id du Cas d'Utilisation Singulier	**Nom du Cas d'Utilisation Singulier**	**Episode 1**	**Episode 2**	**Episode 3**	**Episode 4**	**Episode 5**	**Episode 6**	**Episode 7**
UC0	Normal Case (it is all OK)	OK	OK	OK	OK	OK	OK	OK
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
UC18	Proposant diffusion ; Conflits internes	OK	OK	OK	OK	OK	OK	Singularité 18
UC19	Proposant Diffusion ; Rôle Invalide	OK	OK	OK	OK	OK	OK	Singularité 19
UC20	Diffusion Proposée ; Agenda non-respecté	OK	OK	OK	OK	OK	OK	Singularité 20
UC21	Actualisant Base de données ; problème d'actualisation	OK	OK	OK	OK	OK	OK	Singularité 21
UC22	Signalisant Proposition ; problèmes signal	OK	OK	OK	OK	OK	OK	Singularité 22
UC23	Proposition diffusé ; non regardée	OK	OK	OK	OK	OK	OK	Singularité 23
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**

Figure 8.24 - Extrait de définition des cas d'utilisation singuliers de PLAHE-SOHO.

Un extrait d'une liste des cas d'identification singuliers identifiés pour 'Représentation des opérations d'un plan préliminaire' est montré dans la figure 8.25.

Id	Nom du Cas d'Utilisation Singulier
UC0	Cas Normal
UC1	Identifiant Participant; Participant inexistant
UC2	Identifiant Participant; timeout
UC3	Identifiant Participant; Participant n'est pas le responsable
UC4	Acceptant Nom d'opération ; Nom invalide
UC5	Acceptant Nom d'opération ; timeout
UC6	Acceptant Nom d'opération ; Opération inexistant
UC7	Acceptant Date début d'opération ; Date invalide
UC8	Acceptant Date début d'opération ; timeout
UC9	Acceptant Date fin opération ; Date invalide
UC10	Acceptant Date fin opération ; timeout
UC11	Acceptant Date fin opération ; durée invalide
UC12	Acceptant Justification ; timeout
UC13	Acceptant Demande de Verification ; Demande Invalide
UC14	Acceptant Demande de Verification ; timeout
UC15	Verification de conflits internes ; opération invalide
UC16	Proposant diffusion ; proposition invalide
UC17	Proposant diffusion ; timeout
UC18	Proposant diffusion ; Conflits internes
UC19	Proposant Diffusion ; Rôle Invalide
UC20	Diffusion Proposée ; Agenda non-respecté
UC21	Actualisant Base de données ; problème d'actualisation
UC22	Signalisant Proposition ; problèmes signal
UC23	Proposition diffusé ; non regardée
UC24	Actualiser Base de Données ; timeout

Figure 8.25 - Extrait de la liste de Cas d'Utilisation Singuliers identifiés

33.2.2.1.3Les cas d'utilisations concrets

La construction des cas concrets a été influencé par le choix de l'outil de base : l'environnement IDL (Interactive Data Langage) a été retenu pour 3 raisons :

- il était largement répandu dans l'institut IAS ;
- il tourne sur plateformes Unix, VMS, Windows et Macintosh, donc il peut être utilisé par les différentes centres de SOHO ;
- l'apprentissage et le développement avec ce langage sont très rapide.

Dans cette étude de cas, les cas d'utilisation concrets ont été dérivés directement des cas d'utilisation singuliers. Cependant, au contraire de l'exemple ATM, la construction a été réalisé **après** l'intégration des cas d'utilisation singuliers : nous sommes arrivés alors à un outil intégré sous la forme d'un éditeur de planning partagé permettant un processus de planification basée sur une boucle composée des deux pas : édition de propositions en parallèle puis validation par le groupe. Ainsi l'élaboration d'un plan consisterait en une série d'itérations de deux types d'activités :

- activités parallèles, où la proposition de programmes est faite par chaque équipe instrument ;
- activités de groupe, où tous ses membres discutent pour ajuster, négocier et valider les programmes.

Puisque cet outil permet à plusieurs participants à la réunion d'éditer leur

contributions en même temps, elle induit une modification dans les rôles : le rôle de l'opérateur de planning disparaît, et un membre de chaque équipe instrument se trouve alloué un rôle d'opérateur d'outil. L'expérimentation de cette solution nous semble intéressante parce que le rôle de l'opérateur était un rôle artificiel et concernant l'équipe de chaque instrument, elle a l'impression de participer plus activement et de maitriser le contrôle de ce qu'elle propose.

Le prototype décrit dans cette section a été présenté dans [Bellamine 96]. Pourtant, nous allons résumer ses aspects principaux.

L'architecture du prototype multi-utilisateur consiste en :

- différentes sessions IDL sont lancées sur la machine de chaque utilisateur et une copie de l'outil est lancée par chaque opérateur.
- ces différentes sessions communiquent avec une seule base de données IDL partagées.

Ainsi, les données définissant le plan global en cours son partagées par ces différentes sessions.

L'interface utilisateur est composée de deux zones :

1. Une zone de contrôle (composée de boutons, menus et champs éditables) offrant des fonctionnalités d'affichage et d'édition de plans ;
2. Une zone graphique, composées de plusieurs sous-parties réservées à la représentation des ressources qui ne peuvent pas être modifiés par les utilisateurs, et plusieurs parties (une ligne par instrument, et une couleur par instrument) permettant l'affichage et l'édition des plans de chaque instrument. Le contenu de la zone graphique est personnalisable. En effet, un participant peut choisir les lignes qu'il souhaite afficher dans sa fenêtre, pour permettre à chaque personne d'afficher les parties relatives aux instruments aux quels il s'intéresse.

Il y a des modes d'utilisation : mode d'édition de plans (*Edit mode*) ; et mode d'affichage seul (*Display mode*) de plans.

En mode d'édition de plans, la zone de contrôle comprend des boutons supplémentaires permettant d'ajouter, modifier, effacer un programme de la zone graphique. L'édition des programmes pour chaque instrument peut se faire soit via la zone de contrôle uniquement, soit par manipulation directe de la zone graphique avec la souris pour fixer l'instrument et les horaires de début et de fin, et compléter la description des champs restants par la zone de contrôle.

En mode d'affichage seul, pour permettre la visualisation du plan global à tout instant, un bouton de mise-à-jour (*Update display*) est disponible, mettant à jour les plans affichés sur la fenêtre. A cette mise-à-jour de l'affichage correspond une re-lecture de la base de données partagées. Ainsi, l'interface du groupe peut ne pas être WYSIWIS. Dans le cadre des réunions SOHO, cette option peut être utilisé par tout participant souhaitant suivre l'évolution des plans, par exemple un participant non autorisé à modifier les plans ou le leader pendant la phase des activités parallèles des autres participants.

Comme l'outil sert à l'édition et l'affichage, il faut pouvoir différencier ce qui est une proposition écrite par un membre et ce qui est une décision confirmée par tout le groupe. Cette distinction est fondamentale pour le déroulement et l'avancement de prise de décisions pendant la réunion. Par conséquent, une différenciation graphique est proposée :

- chaque programme proposé est représenté par un rectangle vide ayant la couleur de l'instrument ;
- chaque programme décidé et validé est représenté par un rectangle plein ayant la couleur de l'instrument également .

Cette validation graphique des programmes se fait à l'aide d'un bouton supplémentaire 'Valider' (*Validate*) utilisable en mode d'édition et ce surtout suite à l'accord du groupe pendant les activités de groupe.

L'outil est le même pour tous les participants et utilisateurs éventuels pendant une réunion de planification, y compris le leader, qui n'a pas une interface super-privilègée.

33.2.2.2Expérimentation des Cas d'Utilisation

L'experimentation des cas concrets de PLAHE-SOHO implique la réalisation conjointe des activités face-à-face du groupe. C'est pourquoi nous avons réalisé une simulation d'une réunion avec l'aide du prototype. Cette simulation a créé un univers artificiel mais le plus proche possible de celui supposé être l'univers réel :

- les acteurs sont les acteurs réels, c'est à dire, des vrais participants des réunions de planification de SOHO ;
- le prototype multi-utilisateur est mis à leur disposition ;
- le contexte de planification est le même : aboutir à la planification des opérations d'une semaine de SOHO et à l'édition partagée d'une partie de ces plans ; cependant, les résultats de cette planification ne son pas évidemment appliqués.

Tous les détails de la préparation humaine (trouver les volontaires disponibles, les motiver pour jouer la simulation, proposer une nouvelle structuration du travail avec l'outil multi-utilisateur, définir le rôle attendu de chaque participant, etc.) et de la préparation technique (organisation de la salle de la réunion, préparation de l'ordre du jour prévu pour la réunion et sa distribution quelques jours avant celle-ci, etc) de la simulation ainsi que les remarques générales sur le déroulement de la réunion simulée sont présentés dans [Bellamine 96].

Nous allons ensuite nous concentrer sur quelques résultats de la simulation concernant l'ensemble des besoins initiaux de l'outil conçu :

- la réunion se déroule sous la forme d'une alternance entre les interventions parallèles des représentants de chaque instrument pour éditer un programme et une discussion globale de toute l'équipe pour ajuster les horaires, préciser les caractéristiques et valider le programme ajouté ; donc les deux modes d'édition et d'affichage ont été validés ;
- l'utilisation de l'éditeur partagé de planification par plusieurs personnes en même temps a permis des résultats plus complets et meilleurs : la quantité (7 jours discutés et 3 jours édités) et la qualité (beaucoup de programmes conjointes, le contenu détaillé des plans) des résultats ont été considérés supérieurs à ceux des autres réunions de simulation de durée équivalente d'après les jugements des participants ayant participé à tous les réunions ; par conséquent, l'outil peut être utile pour réduire la durée totale des réunions, considérée auparavant trop élevée.
- certains participants (surtout les utilisateurs de l'outil) souhaitent que le bouton de validation graphique (*Validate*) ne soit autorisé qu'aux utilisateurs de l'équipe de l'instrument (ou leurs représentants) ou au leader;

- certains programmes ont été identifiés comme ‘programmes critiques’, à cause de la possibilité de perturbation inter-instruments. Alors, certains participants (surtout les participants qui ne sont pas les utilisateurs de l’outil) ont suggéré que la proposition et la validation d’un programme critique soient affichés de manière spéciale et que les justifications pour ces programmes ne soient pas optionnelles mais obligatoires pour fournir des explications devant être repris a posteriori .

Ces remarques sont à la base des modifications des cas d’utilisation montrés au paragraphe suivant.

33.2.2.3Modification des Cas d’Utilisation

Les critiques et suggestions des participants de l’expérimentation décrite précédemment (surtout les deux dernières) nous ont permis de faire les modifications suivantes :

- Validation réservée : contrôle d’accès au niveau des modes et au niveau des utilisateurs, pour ne permettre les validations des programmes que par les rôles responsables (typiquement les représentants de chaque équipe instrument et le leader) ;

- Programmes ‘critiques’ : les programmes critiques sont affichés en rouge dans le plan et le remplissage du champs textuel *Notes* fournissant des explications est obligatoire.

33.2.3Le modèle des besoins du système

En plus des besoins initiaux du système, qui ont un caractère générique, le modèle des besoins permet l’association des besoins aux cas d’utilisation, afin de décrire les besoins spécifiques du système pour chaque situation d’utilisation illustrée par un cas, comme par exemple les besoins de fonctionnalité (objets et services), de comportement (interactions) et les facteurs de qualité et d’évaluation du système.

La figure 8.27 montre un extrait du modèle des besoins du cas d’utilisation singulier UC0 - ‘Cas Normal’ de ‘Représentation des opérations d’un plan préliminaire’ après l ‘expérimentation avec les cas d’utilisation concrets. Dans cet extrait les besoins et facteurs de qualité associés au cas d’utilisation sont montrés. En particulier, nous nous concentrons sur les besoins fonctionnels, les besoins de l’interaction et les facteurs de qualité de l’interaction.

Les codes utilisés font référence à la typologie des besoins et facteurs de qualité décrits pour l’exemple ATM dans le chapitre 6. Cependant, les codes des services utilisés dans le modèle des besoins sont montrés dans la figure 8.26. En général, ces codes sont les identificateurs du tableau Services au niveau opérationnel mais dans le cas où il n’y pas la description à ce niveau, nous citons les codes.

S1.Afficher l'agenda de la réunion	S25.Afficher un bilan des instruments	S49.Accepter Identification de participant
S2.Ajouter l'agenda de la réunion	S26.Ajouter un bilan des instruments	S50.Verifier Identification d'un participant
S3.Modifier l'agenda de la réunion	S27.Modifier un bilan des instruments	S51.Verifier rôle responsable par activité
S4.Supprimer l'agenda de la réunion	S28.Supprimer un bilan des instruments	S52.Accepter nom d'opération
S5.Sauvegarder l'agenda de la réunion	S29.Sauvegarder un bilan des instruments	S53.Verifier nom d'opération
S6.Afficher un plan mensuel prévisionnel	S30.Afficher une opération d'un plan	S54.Accepter Date début
S7.Afficher l'état de la réalisation d'un plan mensuel prévisionnel	S31.Ajouter une opération d'un plan	S55.Verifier Date Début
S8.Afficher un plan trimestriel prévisionnel	S32.Modifier une opération d'un plan	S56.Accepter Date Fin
S9.Afficher l'état de la réalisation d'un plan trimestriel prévisionnel	S33.Supprimer une opération d'un plan	S57.Verifier Date Fin
S10.Afficher un bilan des opérations	S34.Sauvegarder une opération d'un plan	S58.Accepter justification
S11.Ajouter un bilan des opérations	S35.Afficher un plan (entier)	S59. Afficher une justification
S12.Modifier un bilan des opérations	S36.Ajouter un plan (entier)	S60.Ajouter une justification
S13.Supprimer un bilan des opérations	S37.Modifier un plan (entier)	S61.Modifier une justification
S14.Sauvegarder un bilan des opérations	S38.Supprimer un plan (entier)	S62.Supprimer une justification
S15.Afficher un bilan scientifique (l'état du soleil)	S39.Sauvegarder un plan (entier)	S63.Change mode d'utilisation
S16.Ajouter un bilan scientifique (l'état du soleil)	S40.Afficher un bilan technique (support mission)	S64. Mise à jour les plans affichés
S17.Modifier un bilan scientifique (l'état du soleil)	S41.Ajouter un bilan technique (support mission)	S65.Accepter Demande de Vérification
S18.Supprimer un bilan scientifique (l'état du soleil)	S42.Modifier un bilan technique (support mission)	S66.Verification pour détecter des conflits internes
S19.Sauvegarder un bilan scientifique (l'état du soleil)	S43.Supprimer un bilan technique (support mission)	S67.Annoncer résultat de vérification de conflits
S20. Classer les plans anciens par attribut donné	S44.Sauvegarder un bilan technique (support mission)	S68.Accepter Demande de Diffusion
S21. Classer les plans anciens par attribut donné	S45.Calculer pourcentage de temps libre par instrument	S69.Verifier si l'agenda est respecté
S22. Afficher plans anciens	S46. Calculer le temps alloué aux programmes synoptiques	S70.Atualiser Base de Données Partagée
S23. Copier une partie d'un plan ancien	S47. Calculer le temps alloué aux opérations coordonnées	S71. Signaliser la Diffusion aux participants
S24. Copier un plan ancien (entier)	S48. Calculer le taux de réalisation d'un programme en cours	S72.Validation des programmes

Figure 8.26 - Codes des services de PLAHE-SOHO

Dans la figure 8.27 l'épisode 7 (Diffuser la proposition au groupe) est associé aux besoins fonctionnels dynamiques (BFD) suivants : les services S49 (Accepter Identification de participant) à S71 (Signaliser la Diffusion aux participants) et aux besoins fonctionnels statiques (BFS) suivants : les Objets Métiers du domaine du problème Plan_Scientifique_SOHO, Instrument, Programme, Programme_Conjoint_JOP et leurs attributs . Ces services et informations sont des composants du cas d'utilisation et donc des ressources pour la réalisation d'un système qui supporte ce cas.

Des exemples des contraintes de ce cas d'utilisation sont les équipements, les matériels et les logiciels existants qui font partie du modèle de plateforme et qui sont disponibles pour l'épisode en question, à savoir le P1 (rétro-projeteur), le P2 (tableaux de la salle de réunions), P4 (terminal X), P8 (Unix) et P9 (Environnement IDL). Le logiciel P10 (Editeur mono-utilisateur graphique de Planning 'Sohoparr') sera remplacé

par l'outil multi-utilisateur dont nous déterminons les besoins.

Des exemples de critères d'évaluation sont :

1. les facteurs de qualité du système FQS2 - Intégrité et FQS3 - Sécurité, dont l'importance est montré par le besoin demandé BUE#3 - Consistance d'informations disponibles. La consistance est directement associée à ces deux facteurs : l'intégrité est par définition la cohérence et la véracité des données du système /Theron 88/ ; et la sécurité est la propriété d'un système se protégeant des altérations de son code ou de ses fichiers par des accès extérieurs (personnes, événements, autres systèmes, etc) [Ghezzi 91]. La sécurité se décompose en contrôle des accès (contre lecture) et protection des accès (contre l'écriture).

2. Les facteurs de qualité de l'interaction FQI2 - Synthèse, FQI4 - Generalisabilité, FQI9 - Multimodalité, FQI8 - Substituabilité, FQI12 - Recuperabilité, et FQI13 - Conformité à la tâche, dont l'importance est due à l'utilisation de l'outil par des utilisateurs qui sont astrophysiciens sans formation spécifique en informatique;

3. Les facteurs de qualité du processus FQP1 - evolutibilité et FQP3 - testabilité, parce que l'insertion future d'autres opérations est très probable et FQP5 - portabilité et FQP6 - interoperabilité parce que le changement de plateforme pour exécuter l'outil dans les différents centres SOHO est certain.

•

Cas d'Utilisation Singulier	**Episodes**	**Besoins Fonctionnels Composants (ressources)**	**Contraintes**	**Critères d'Evaluation**
UC0	7 (Diffuser la proposition au groupe)	BFD : les services S49 (Accepter Identification de participant) à S71 (Signaliser la Diffusion aux participants) BFS : Objets Métiers Plan_Scientifique_SOHO, Instrument, Programme, Programme_Conjoint_JOP et leurs attributs	BET : P1, P2, P4, P8, P9	FQS2, FQS3; FQI2, FQI4; FQI8, FQI9; FQI12, FQI13; FQP1, FQP3, FQP5, FQP6

Figure 8.27 - Besoins de l'épisode 7 (Diffuser la proposition au groupe) du cas d'utilisation singulier UC0 - 'Cas Normal' de 'Représentation des opérations d'un plan préliminaire' (CUE#9)

33.2.4Modèle de Design Rationale

La figure 8.28 montre un extrait d'un schéma QOC pour le service S52 - Accepter nom d'opération de l'épisode 2 - Proposer le nom de l'opération du cas d'utilisation 'Représentation des opérations d'un plan préliminaire' (CUE#9) de PLAHE-SOHO. Le schéma QOC est décrit sous la forme d'une notation tabulaire avec trois colonnes représentant respectivement les questions, les options et les critères, dont les critères positifs sont précédés par le signe (+) et les critères négatifs par le signe (-).

Dans cette figure, la question (Q) est la spécification pour le service S52 (Accepter nom d'opération). Les alternatives de spécification sont montrées comme options (O) et sont associés aux critères (C) positifs et négatifs qui peuvent orienter le choix. Par exemple, les options 1 , 2, 3 et 4 montrent différentes façons (évidemment nous ne prétendons pas l'exhaustivité) pour que le participant fournisse le nom d'une opération dans l'édition d'un plan. L'option 3 par exemple a comme un des critères positifs le

facteur de qualité de l'interaction substituabilité parce qu'elle permet de remplacer des valeurs d'entrée par des valeurs de sortie choisies dans un plan affiché. La quatrième option est la combinaison des options antérieures, en laissant à l'utilisateur le choix entre taper le nom, choisir dans une liste ou encore faire 'copier-coller', ce qui permet en fait de satisfaire les critères de dialogue à fils multiples (FQI6) et de multimodalité (FQI9), qui est dans ce cas du type synergique.

Questions	Options	Critères
S52 - Accepter nom d'opération	**Option 1** : Participant tape le nom de l'opération (comme il est usuellement utilisé)	(+) BUE #2 (+) BUE #3 (+) BUI#1 (+) BET : P4, P8, P9 (+)FQI13, FQI14; (-) FQI9
	Option 2 : Participant fait le choix dans une liste d'opérations classées par instrument	(+) BUE #2 (+) BUE #3 (+) BUI#1 (+) BET : P4, P8, P9 (+)FQI3, FQI4; (+)FQI11, FQI13; (-) FQI9
	Option 3 : Participant fait 'copier-coller' à partir d'un plan affiché	(+) BUE #2 (+) BUE #3 (+) BUI#1 (+) BET : P4, P8, P9 (+)FQI3, FQI4; (+)FQI5, FQI6; (+)FQI8, FQI13; (-) FQI9
	Option 4 : Combinaison des options 1, 2 et 3 antérieures	(+) BUE #2 (+) BUE #3 (+) BUI#1 (+) BET : P4, P8, P9 (+)FQI13, FQI14; (+)FQI11, FQI8 (+)FQI5, FQI6; (+) FQI9 (en permettant à l'utilisateur d'utiliser 3 manières différentes d'entrer des informations);

Figure 8.28 - Exemple de schéma QOC appliqué au Cas d'utilisation Singulier UC0, Episode 2; Service S52 -Accepter Nom de l'opération

33.3Un exemple de Processus Emergent pour la Coopération dans l'Ingénierie des Besoins d'un Outil d'Assistance de Réunion pour PLAHE-SOHO

Cet exemple est intentionnellement incomplet pour des raisons de longueur. Il s'agit de décrire une séance de travail coopératif pour la définition des besoins d'un système destiné à assister les réunions de planification de l'utilisation du satellite héliosphérique SOHO.

Les données suivantes ont été recueillies sous forme d'enregistrements vidéo (voir [Bellamine 96]). Nous avons selectionné un extrait d'une séance de travail du groupe destiné à établir les besoins pour un nouveau système d'assistance aux réunions de planification.

Ce processus est en notre sens le type même du processus émergent. Les différents acteurs abordent le problème de leurs points de vue respectifs et le cours de leur interaction est entièrement indéterminé. Cette réunion correspond à un type d'interaction face-à-face. Nous pensons que le type d'interaction n'est pas une donnée majeure et

n'entraîne aucun biais pour la description de ce processus émergent avec un nombre réduit de concepts simples couramment utilisés pour la description des processus. Nous illustrons aussi le type d'évolution envisageable sur la base de cet exemple.

Nous avons laissé les différentes interventions en anglais parce que le lieu de la réunion était le centre américain EOF mais nous insérons des commentaires en italique pour illustrer les concepts orientés processus. La présentation suit la chronologie du processus observé.

33.3.1Présentation du Processus

Au début de cette séquence les différents acteurs ont négocié les différents objectifs de leur réunion et ils ont donc un ensemble de buts précis.

Plan: - Know the problem domain and the actors (missions, group, roles, etc.)
- Define the boundary relevant of the system
- Identify specific planning problems of identified roles
- Understand user needs
- Understand environmental (social, organizational, physical) constraints
- Refine the requirements list and to detect ambiguities and inconsistencies
- Integrate the requirements list across the various group members
- Simulate a planning meeting
- Investigate possible solutions

Agents et Rôles:

- A : l'analyste qui observe et étudie la réunion
- U_i : est un membre du groupe qui représente les utilisateurs d'un instrument donné du satellite et futur utilisateur du système en cours de conception,
- D : concepteur de système informatique

But: identify operator problems

	Message 1 = {A : "During our analysis of the group activities, we notice that the operator has a heavy task. As shown on this graph -Art1-, he is overloaded."}
	Message 2 = {D : "How and Why is he so overloaded ?"}
Conversation C1:	**Message 3** = {U : "During this meeting, we were many scientists, involved in SOHO, discussing operations, proposing coordinated observations to be performed jointly by different instruments in the same time. So, the timeline had to be updated at every moment: every change, of every program, of every instrument had to be made rapidly, seen by all participants becoming our new reference for further discussions and new proposals. So, during our last meeting, as the operator was alone and was receiving different requests from different participants, and sometimes he did not know the instruments, the names of the activities to be added, so for all these reasons he couldn't satisfy all demands rapidly and for many times we were obliged to wait for his updates and lose time ..."}
	Message 5 = {A : "This is very clear on the video recorded during this meeting. we have some sequences here -Art2-..."}

Commentaires

L'artefact Art1 est un graphe représentant les activités des participants au réunions en fonction du temps. Il est manifestement indexés par le message 1.

L'artefact Art2 est un enregistrement vidéo montrant les dysfonctionnements durant les réunions.

But : Understand specific requirements requested explicitely by group members

Commentaires

*Ici nous assistons à un changement dynamique du but courant. le But "*Understand specific requirements requested explicitly by group members*" , non prédéfini au départ apparaît dans l'argumentaire des messages 6 et 7. En fait on peut dire que ce but a émergé au cour de l'interaction*

	Message 6 = { U_1 = "I would like to use directly the tools to record my plan changes"}
Conversation C2:	**Message 7** = { U_2 = "But I would like also to see the up-to-date information in addition to my own data"}

But: - Investigate possible solutions

Commentaires

ici, une suggestion d'une solution possible est proposée durant le processus, changeant le but courant.

	Message 8 = {A : "In brief, I think that integrating a multi-user planning editor, which can be shared by meeting participants would provide better rendering of such meeting."}
	Message 9 = {U : "But How ?"}
	Message 10 = {D : "By the means of a shared editor every participant would be able to add, change, delete, ... his part in the timeline"}
Conversation C3	**Message 11** = {U : "Yes, this would be a nice thing, every one will be like an operator but it couldn't work because we have only 11 instruments, and every instrument is represented by 1 to 4 people team in this kind of meetings, and I think about two problems. First we will need as many terminals as people attending the meeting instead of one terminal and one LCD display and overhead projector used now ? This will be very expensive I think. Second, it would be difficult to do something if all people can change every thing..."}
	Message 12 = {A : "As I could understand, and as it is prescribed, during this meeting only one representative of an instrument team have to speak on the of the whole team to take decisions, so we can introduce only one terminal per represented instrument and by this way only one person per instrument would make updates"}

But: Investigate possible solutions

Conversation C4	**Message 13** = {D : "We can also suggest with this tool two modes : Editing mode to be used by those allowed to make changes to the shared planning, and Display mode to others, and also keep the use of LCD display and overhead projector to project continuously shared and updated timeline."}
	Message 14 = {A : "Ah ! this is the case of the planning editor used now. We have two modes : display only and editing"}

But: - Understand specific requirements requested explicitly by group members

Conversation C5	**Message 13**= {U : "It will be a nice thing if we could have the same interface as the one of the tool we are using now. It is a good one : easy to use and very intuitive,... I like it because I was able to use rapidly, after few trials it was OK..."}

Finalement, un artéfact représentant un résumé du travail déjà accompli peut être créé et édité par le concepteur comme un relevé de besoins à être analysé comme des besoins demandés explicitement par les utilisateurs.

Finally we can conclude as follows :
- First : a multi-user and shared planning editor will be introduced to lighten operator task, to optimize meeting period
- Second : two modes -editing and display only- have to be offered and editing mode would be enabled only for allowed people, to control flow of changes in the plan
- Third : the user interface may look like the one of the existing single user tool because it seems good and would facilitate the integration of a new tool in this setting
- Forth : the re-usability of the existing tool will be studied in order to reduce development time and cost
- Fifth : a rapid prototype have to be done and tested with actual users as soon as possible

33.3.2Commentaires

La présentation de cette séquence de travail selon un mode processuel basé sur des composants comme rôles, agents, artefacts, conversations et messages illustre l'adéquation de cette conception du processus avec une séquence de travail réelle entièrement déterminée par les interactions des différents agents. Nous avons montré comment l'indéxicalité, le changement de but courant ou l'ajout de but nouveau (toutes choses qui ne sont pas conscientes au niveau des agents que nous avons enregistrés) apparaît au cours du processus. Notre exemple permet d'une part de bien fixer la notion de processus émergent, et d'autre part la généricité des concepts utilisés pour la description du processus. Le fait que l'ensemble des personnes présentes soit en mesure d'entendre les autres et de partager les mêmes informations illustre la notion de domaine consensuel qui est créé en même temps que le processus suit son cours.

La richesse du modèle et ses notions de Décisions, d'arguments Pour ou Contre servent à ajouter une structure argumentaire au processus.

Decision: installer un éditeur partagé
Pro-Argument chacun pourra mettre à jour le planning
Con - Argument pas assez de terminaux pour chacun
Décision un éditeur partagé avec un terminal par équipe et non par participant

Ces concepts sont sémantiquement proches des acteurs et leur permettent d'adapter le

modèle à leur mode de fonctionnement au cours du processus sans que la distance cognitive entre la tâche de redéfinition dynamique du modèle et le processus lui-même ne soit un obstacle à l'adaptabilité du système. De la même manière on aurait pu penser à la structuration des questions à décider, les alternatives et les arguments sous la forme de diagrammes QOC. Cela permettrai d'utiliser le même format adopté pour enregistrer le *design rationale* mais avec la différence que le modèle de processus émergent sert à conduire la recherche des questions, une recherche qu'il n'est pas prévue dans l'usage courant de QOC. Cet aspect illustre l'importance de l'ouverture du modèle pour son évolutivité et pour son adaptabilité du point de vue de l'utilisateur final.

34.Synthèse

Cette section présente une synthèse sous la forme d'une discussion des résultats de l'application de TAREFA à PLAHE-SOHO. Ces résultats sont structurés par les paragraphes suivants :

- techniques de recueil (paragraphe 4.1)
- modélisation riche du contexte (paragraphe 4.2)
- utilisation des cas d'utilisation (paragraphe 4.3)

34.1Techniques de recueil

L'utilisation synergique de différentes techniques de recueil nous a permis de recueillir une grande quantité d'information d'une grande qualité tout le long du processus d'ingénierie des besoins. En particulier, l'immersion d'un analyste a donné l'opportunité pour les conversations informelles avec les participants du groupe et surtout pour les observations de l'environnement de travail et du déroulement des réunions, comprenant l'information tacite difficile à recueillir d'une autre façon.

En fait, cela a validé notre idée de que les techniques basées sur l'observation (TBO) sont fondamentales pour la conception d'un système interactif. En fait, les concepteurs ont tendance à utiliser des techniques plus analytiques au détriment de techniques plus écologiques et concrètes pour l'observation et prise de conscience du contexte de travail du système.

Cependant, l'utilisation des TBO est chère. L'effort de placer un analyste dans l'environnement est énorme : les problèmes sociaux et politiques d'adaptation au groupe, les problèmes d'apprentissage du vocabulaire spécifique du domaine et les problèmes de l'entraînement pour les activités de travail ne sont pas faciles à traiter.

En plus, recueillir les besoins par l'intermédiaire des TBO n'est pas suffisant : observer et comprendre la pratique du travail courant n'aide pas à évaluer les conséquences de propositions de conception qui ne sont pas encore réalisés. C'est pourquoi le processus de détermination des besoins utilisant les cas d'utilisation a été très important (voir paragraphe 4.3 ensuite).

34.2Modélisation riche du contexte

Cette étude nous a confirmé que la compréhension de la situation concernée dans un environnement réel est très complexe. Les analystes se trouvent souvent submergés par

une grande quantité d'informations et de données sur le terrain. Face à de telle circonstances, l'utilisation des modèles de TAREFA pour représenter l'information contextuelle recueillie nous a permis d'identifier ce qui est pertinent et nécessaire pour l'élucidation du problème à résoudre avec le développement du système et pour la compréhension et la modélisation des caractéristiques actuelles de l'univers du problème.

Plus spécifiquement :

- le modèle ontologique a été très consulté par l'analyste pour les conversations avec le groupe mais aussi pour raffiner la compréhension et orienter les questions à poser. En fait, même les autre participants du groupe l'ont utilisé éventuellement, grâce à l'utilisabilité de sa structuration ;

- les modèles de tâches minimales reflètent la qualité des observations de l'analyste qui a fait l'immersion ; cependant, ces modèles ne décrivent pas les liens qui existent entre les différentes activités qui expliquent la coopération entre les participants ni les échanges (dialogues ou messages) entre eux ; cette déficience s'explique en partie par la déficience des modèles des tâches existants pour représenter la complexité des tâches coopératives ;

- le modèle de contexte organisationnel nous a permis d'expliciter des informations très importantes pour le développement de l'outil (comme par exemple les règles organisationnels U1, U2 et U3, toute la richesse et complexité des rôles organisationnels, et la relation de responsabilité) ;

- le modèle des objets métiers a servi pour la modélisation conceptuelle de la base de données partagée ;

- le modèle de plateforme a délimité les éléments (matériels, équipements, outils, logiciels) utilisés dans la situation coopérative à une échelle individuelle ou collective.

34.3Utilisation des cas d'utilisation

D'après notre expérience concrète, nous n'avons aucune doute que les cas d'utilisation sont très utiles pour le processus d'ingénierie des besoins.

Cette étude de cas nous a montré que :

- certaines questions sur les besoins ne sont répondues qu'après l'investigation des singularités et l'expérimentation avec les cas d'utilisation concrets ;
- l'utilisation des cas d'utilisation permet d'envisager les situations de travail et de considérer comment elles seront affectés par le système même pour les situations non usuelles mais quand même significatives (comme les situations où il y a l'occurrence de singularités) ;
- l'utilisation des buts des modèles de tâches minimales pour systématiquement construire les cas d'utilisation nous a permis de

envisager des cas qui ne seraient pas pris en compte par d'autres méthodes *ad hoc* de construction de cas d'utilisation, comme par exemple les cas de définition de l'agenda (CUE#2) et de la représentation des opérations (CUE#9).

Conclusion

Cette conclusion s'articule autour de trois thèmes. La première section résume les principales contributions de cette thèse, tandis que la deuxième en dresse ses limites. Enfin, la troisième propose des nouvelles perspectives pour la continuité de ce travail.

35.Contributions

Notre travail contribue à l'ingénierie des besoins de systèmes interactifs, particulièrement avec la proposition de l'approche TAREFA. La contribution de TAREFA peut être résumée par les aspects suivants:

1. L'intégration de différents concepts, modèles et techniques - originaires surtout du domaine du GL et de celui de l'IHM - lors du processus de l'Ingénierie des Besoins. Parmi ces approches, certaines font partie de la contribution de cette thèse (résumées ci-dessous), d'autres sont déjà existantes. La réutilisabilité d'approches existantes, utilisées avec succès dans des travaux antérieurs, est un choix délibéré parce qu'il très important profiter de leur expérience. Ainsi, nous (ré)utilisons par exemple les concepts et techniques de la méthode Objectory (particulièrement les concepts et techniques de son processus d'Analyse), le modèle et la notation QOC pour représenter et documenter les alternatives et les choix de conception (*design rationale*), les attributs d'utilisabilité de la littérature IHM et les techniques pour allocation des fonctions et la réingénierie des tâches de l'Ergonomie. Cependant, l'utilisation complémentaire de toutes ces approches constitue une contribution originale.

2. Nous avons proposé un modèle ontologique pour représenter le vocabulaire du domaine employé par les utilisateurs et ainsi établir une terminologie unifiée pour permettre une meilleure communication avec l'analyste.

3. Nous avons proposé le modèle de contexte organisationnel TOCO, pour représenter l'environnement social et organisationnel dans lequel les activités des utilisateurs se déroulent. Le modèle TOCO est constitué de plusieurs composants (à savoir les objectifs, les activités, les rôles, les concepts du domaine, les règles organisationnelles, les événements organisationnels, les espaces de travail, et les perspectives de rôle) chacun représentant un aspect particulier de ce contexte, et qui sont en relation entre eux et avec les autres modèles utilisés. Le chapitre 5 a présenté en détail chacun de ces composants ainsi que les relations les plus importantes qu'ils entretiennent.

4. Nous avons proposé un modèle de cas d'utilisation structuré en quatre niveaux d'abstraction interconnectés - **essentiel, téléologique, opérationnel** et **concret** - chacun concernant un type de connaissance spécifiques. Ce modèle est fondamental pour déterminer les besoins du système, ce qui est réalisé au moyens d'expérimentations et de modifications itératives des cas d'utilisation. En plus, une approche pour construire les cas d'utilisation à partir d'un modèle particulier de tâche - le modèle minimal de tâche - est aussi proposée. Dans cette thèse nous avons montré que la combinaison des deux techniques (modèles de tâche et cas d'utilisation) est très intéressante pour construire les cas d'utilisation, pour après les utiliser pour déterminer besoins et les

valider par expérimentation;

5. Dans le domaine du support aux activités collaboratives de l'IB, nous avons proposé un modèle de processus basé sur la notion d'émergence. Ce modèle ne vise pas à automatiser ni à guider ces activités mais à servir de support à la fois structuré et flexible à la coopération dans laquelle les plans et buts évoluent de façon consensuelle et émergente en prenant compte les actions, les communications et la cognition du groupe pendant que ses membres interagissent.

Dans l'Introduction Générale, nous avons défini l'hypothèse générale que sous-tend notre travail de thèse. Après l'exposé du contenu de ce mémoire et les contributions de TAREFA, voici les conclusions que nous pouvons en tirer comme contributions par rapport à cette hypothèse.

TAREFA aide à recueillir plus d'information (en définissant les étapes, l'enchaînement et les données échangés entre chaque étape) et à les organiser par un ensemble de modèles dont le contenu et la technique de description sont précisément définis. Cependant, TAREFA n'apporte pas de réponse à la question de la garantie d'utilité et d'utilisabilité des systèmes construits en suivant cette approche. En effet, la réponse à cette question est liée à la difficulté inhérente de validation des méthodes de l'informatique. Cette difficulté est à l'origine des diverses questions posées lors d'une validation : Combien d'études de cas sont nécessaires pour valider une méthode ? Faut-il la comparer avec d'autres méthodes appliqués à la même étude de cas? Comment faire la distinction entre les variations dues à la méthode et les variations dues au savoir faire de l'analyste ou au contexte de l'application (équipe qui a appliqué la méthode, période d'application, aspects politiques internes, etc). Ainsi, la réussite d'une méthode ou d'une approche en informatique à l'heure actuelle est encore mesurée par la qualité du système développé, par la satisfaction des utilisateurs de la méthode (dans notre cas, les analystes) et la satisfaction des utilisateurs du système. Bien que le progrès sur la définition et formalisation des mesures de qualité d'un système soit aujourd'hui important , l'évaluation de la qualité est encore la plupart du temps faite de façon subjective.

Cette discussion est le coeur même du problème de la définition d'une méthode dans le travail d'une thèse. Dans cette thèse, nous avons montré que TAREFA est utile car nous avons montré qu'en suivant l'approche on a pu traiter l'étude de cas ATM. Nous avons aussi montré qu'elle est utilisable car nous avons pu l'appliquer sur une étude de cas. Nous garantissons est que TAREFA permet de recueillir, d'organiser et prendre en compte plus d'informations que les autres méthodes existantes. La structure intrinsèque de TAREFA est conçue spécifiquement pour l'Ingénierie des Besoins de façon à permettre de recueillir, d'organiser et prendre en compte les informations du contexte (même les informations informelles) plus tôt dans le cycle de vie, tout en permettant la facile intégration des informations contextuelles aux méthodes actuelles de l'IB et en réutilisant des approches existantes qui ont été utilisées avec succès comme modélisation de tâche, cas d'utilisation ou modèle de *design rationale*. En plus, TAREFA propose une manière d'intégrer ses résultats aux méthodes de développement existants comme par exemple Objectory.

Les problèmes qui se posent en suite sont ceux de la *scalability* de l'approche:

- est-elle utilisable par d'autres personnes ? Nous n'avons pas encore de réponse a cette question ;
- est-elle utilisable pour d'autres études de cas ? C'est la raison pour laquelle nous avons traite PLAHE-SOHO, mais il est clair que c'est insuffisant ;
- est-elle utile par rapport a d'autres approches ? Nous avons réussi a représenter tous les besoins que nous avons trouve dans des modèles de TAREFA ce qui n'était le cas avec aucune des autres méthodes que nous avons étudié). Par contre le fait de savoir si le résultat est utile (i.e. le système produit sera meilleur) n'a pas été prouve.

En fait tous ces problèmes se posent parce que TAREFA est le produit d'une recherche académique. L'avantage d'une telle recherche est que les concepts de la méthode sont clair et précisément définis (ce qui n'est pas forcement le cas d'une recherche industrielle) mais que le nombre d'application n'est pas suffisant pour tirer des conclusions définitives. Ceci est la raison pour laquelle la première des perspectives de continuation du travail (voir section Perspectives) est dirigée vers l'application de TAREFA dans un contexte industriel.

36.Limites

Les contributions résumées à la section précédente ont néanmoins quelques limitations.

Parce que TAREFA a beaucoup de modèles et d'interrelations entre eux et une démarche avec plusieurs activités, elle est une approche 'poids lourd' par rapport à des approches abrégées (prototypage rapide, par exemple) où l'effort de modélisation est très réduit. Cependant, cela est allégé par la possibilité de réutilisation des modèles pour la définition des besoins concernant des systèmes similaires dans le même domaine ou dans la même organisation, une situation fréquente dans les cas de systèmes 'maison'. En plus, la possibilité d'adopter un sous ensemble des modèles de TAREFA peut fournir une version 'raccourcie' de l'approche. Ce sous ensemble contient typiquement les modèles incontournables de TAREFA (le modèle minimal des tâches et le modèle de cas d'utilisation) pour l'élaboration et l'expérimentation rapides des cas d'utilisation. Cette version 'raccourcie' peut seulement être appliqué dans des conditions déterminées (par exemple si la liaison entre le poste de travail et l'organisation est très tenue, l'influence du contexte organisationnel est réduite - exactement comme l'exemple du distributeur de billets ATM) et la qualité du résultat est sensiblement différente. Dans les conditions normales, tous les modèles définis sont nécessaires (par exemple, si on ne prend pas en compte le contexte organisationnel, on pourra avoir un poste de travail qui ne s'intègre pas de façon adéquate à l'organisation).

Le manque d'application de TAREFA dans un contexte industriel est une autre limitation et il est clair que c'est une étape nécessaire à la validation de l'approche. Le domaine de l'étude de cas PLAHE-SOHO (du chapitre 8) constitue un pas dans cette direction. Pourtant, nous sommes tout a fait conscients des limites de notre travail et nous préférons d'abord bien mettre au point la méthode avant de nous lancer dans des applications industrielles.

37.Perspectives

Après avoir évoqué les contributions et limitations de notre travail, nous proposons maintenant de nouvelles perspectives de recherche :

1. l'utilisation en plus d'études de cas réelles surtout dans le domaine industriel ; Une des principales limites de TAREFA est son manque d'application dans le domaine industriel mais cela constitue aussi une des premières directions de prolongement de notre travail;

2. le développement d'outils d'assistance à l'édition des modèles ou à la génération (assistée ou le cas échéant automatique) de certains modèles (comme les scénarios concrets à partir de cas d'utilisation opérationnels) mais aussi à leur vérification;

3. Plus précisément , il y a encore quelques nouvelles possibilités pour les contributions spécifiques:

    - l'utilisation de la connaissance structurée dans le modèle ontologique pour la construction de modèles du domaine. Cette travail a été déjà commencé (voir

par exemple [Pimenta & Faust 97] et [Faust & Pimenta 95]) par la définition d'un ensemble de heuristiques pour cette transformation mais nous pensons que cet ensemble peut être plus élaboré ;

- chercher des possibilités d'utilisation du modèle de contexte organisationnel pour construire ou valider les modèles d'information de l'organisation;

- la proposition d'une démarche de formalisation itérative pour la construction de spécifications formelles du système (par exemple en utilisant ICO [Palanque 92]) ;

- le raffinement du modèle de processus émergent et l'investigation de sa potentialité par l'intermédiaire de son application à des nouveaux études de cas. Notre intention est aussi d'intégrer l'outil associé à ce modèle - à l'heure actuelle un prototype d'un outil de Travail Coopératif Assisté par Ordinateur (CSCW) - à l'environnement de support aux activités de la démarche de TAREFA.

Bibliographie

[Aboulafia et al. 94] Aboulafia, A. ; Klausen, T. ; Jorgensen, A. *Towards a Theoretical Underpinning of Scenarios* , Proc. of the 17th Information Systems Research seminar In Scandinavia (IRIS 17), Finland, August 6-9 1994., pp 561-571.

[Abowd et al. 92] Abowd, G.; Coutaz, J.; Nigay, L. *Structuring the Space of Interactive System Properties*, In: Engineering for Human-Computer Interaction, Larson, J.;Unger, C. (eds), North Holland, Amsterdam, 1992, pgs. 113-28.

[ACM SIGCHI 92] ACM SIGCHI Bulletin Octobre, V. 24, N. 4, October 1992

[Andersen 90] Andersen, P. B., *A Theory of Computer Semiotics* , Cambridge University Press, Cambridge, 1990.

[Andersen 92] Andersen, P. B., *The force dynamics of interactive systems: Towards a computer semiotics*. Department of Informatic and Media Science - University of Aarhus, 1992.

[Andriole 93] Andriole, S.J.; Freeman, P. *Software Systems Engineering: the Case for a New Discipline*. Software Engineering Journal, May 1993, pp 165-179.

[Anne & Falzon 94] P. Anne et P. Falzon Méthode de description de l'activité de réunion orientée-acteur et orientée collectif IHM'94, 114-118

[Anton et al. 94] Anton, A. ; McCracken, W. ; Potts, C.. Goal Decomposition and Scenario Analysis in Business Processing Reengineering. Proc. Of the Sixth Conference on Advanced Information Systems Engineering (CAiSE' 94), Utrecht, Netherlands (June 6-10), Springer Verlag, 1994.

[Arango & Freeman 88] Arango, G.; Freeman, P. *Application of Artificial Intelligence*. ACM Software Engineering Notes, V. 13, N.1, January 1988, pp 32-38. (Workgroup Report of Fourth International IEEE Workshop on Software Specification and Design, April 1987).

[Arango, 1989] Arango, G. and Prieto-Díaz, R. *Domain Analysis : Concepts and Research Directions*.. Computer Society Press Tutorial, Pittsburgh, 1989.

[Arbaoui & Oquendo 94]Arbaoui, S.; Oquendo, F. *Goal Oriented vs. Activity Oriented Process Modeling and Enactment: Issues and Perspectives*. In: Proc. of Third European Workshop Software Process Technology, EWSPT'94 B C Warboys (ed), LNCS 772, Springer-Verlag, 1994.

[Ashworth 89] Ashworth, C.M. Using SSADM to Specify Requirements. In IEEE Colloquium on 'Requirements capture and Specification for Critical Systems' (Digest No. 138, 3/1-3/3. Institution of Electrical Engineers, November 1989.

[Atwood 95] Atwood, M. Position Paper at ACM SIGCHI Panel on User Centred Design. Apud [Karat 96].

[Aubin et al. 94] Aubin, F; Robert, J-M; Engleberg, D. From Task Analysis to User Interface Design. In: Proc. of 12th Triennal Congress of the IEA., V. 4, August 15-19 1994, Toronto, 397-399.

[Balzert et al. 96] Balzert, H. et alli. The JANUS Application Development Environment - Generating More than the User Interface. In:[CADUI 96].

[Bannon 1985]Bannon, L. J. *Extending the Design Boundaries of Human-Computer Interaction*, San-Diego : Institute of Cognitive Science, 1985.

[Barghouti 95] Barghouti, N. et alli. *Two Case Studies in Modeling Real, Corporate Processes*. Software Process - Improvement and Practice, Pilot Issue, August 1995.

[Barnard 93] Barnard, P. *Modeling Users, Systems and Design Spaces*. In: Human-Computer Interaction: Applications and Case Studies (19 A), Smith, M.J.; Salvendy, G. (Eds.) Advances in Human Factors/Ergonomics, Elsevier, 1993. Proc. of HCI' 93, V.1, pp 331-336.

[Barthe 95] Barthe, M. Ergonomie des Logiciels, Masson, 1995.

[Barthet 1988]Barthet, M.-F. *Logiciels Interactifs et Ergonomie - Modèles et Méthodes de Conception*, Dunod Informatique, 1988.

[Barthet 1993]Barthet, M.-F. *L'Intégration de l'Ergonomie aux Mèthodes Informatiques de Conception de Logiciels*. Anais do Segundo Congresso LatinoAmericano de Ergonomia, Florianópolis, 1993 (Présentation Invitée).

[Barthet 86] Barthet, M.F. Conception d'Applications Conversationnelles Adaptée à l'Utilisateur. Thèse de Doctorat d'Etat, INPT, France, 1986.

[Basili & Musa 3 91] Basili, V.; Musa, J. *The Future of Software Engineering: A Management perspective*. IEEE Computer, 24(9), September, pp 90-96, 1991.

[Bass & Coutaz 91] Bass, L.; Coutaz, J. *Developing Software for the User Interface*. Addison-Wesley, 1991.

[Bastien 96] Bastien, J.M.C. Les Critères Ergonomiques: un Pas vers Une Aide Métjodlogique à l'Evaluation des Systèmes Interactifs, Thèse en Ergonomie Cognitive, Univ. René Descartes (Paris V), Decembre 1996.

[Bellamine et al. 96] Narjes Bellamine, Marcelo Pimenta, Remi Bastide.Approche Orientee Scenario pour l'Etude du Travail Cooperatif:Une Etude de Cas, In: Actes du 5emeColloque Ergonomie et Informatique Avancee ERGO-IA 96, 9-11 October, Biarritz, France, 1996. (In French)

[Benyon & Palanque 95] Benyon, D.; Palanque, P.(eds) Critical Issues in User Interface Systems Engineering, Springer Verlag, Berlin , 1995.

[Benyon 92a] Benyon, D. The Role of task Analysis in Systems Design. Interacting with Computers, 4(1), 1992.

[Benyon 92b] Benyon, D. *Task Analysis and System Design: the Discipline of Data*. Interacting with Computers, 4(2), 246-59, 1992.

[Benyon 95] Benyon, D. A Data Centred Framework for User Centred Design. In: Proc. of INTERACT'95, pp 197-202.

[Benyon 96 livre Phil] texto de Benyon dans le livre Benyon, D ; Palanque, P. (eds). Critical Issues in User Interface Systems Engineering, Springer-Verlag, 1996,

[Bevan 95] Bevan, N. Usability is Quality of Use. In: Anzai, Y. (ed) Symbiosis of Human and Artifact. Proc. of the Sixth Intl. Conf. On Human Computer Interaction (HCI International '95), Tokyo, Japan, 9-14 July 1995, Volume 2.

[Biébow 92]Biébow, B. Szulman, S. *DISERT - A Tool for the Detection of Incoherences in Software Engineering Requirements Texts*. Proc. of 5th International Conference on Software Engineering and Its Applications, Toulouse 1992, pp 577-88.

[Blanford et al. 93] Blanford et alli Integrating User Requirements and System Specification. AMODEUS Project Document BC7, April 1993.

[Bodart et al 94] Bodart, F. Et alli. A Model-based Approach to Presentation: A Continuum from task Analysis to Prototype. In: Proc. of 1st Eurographics Workshop on Design, Specification and Verification of Interactive Systems, Bocca di Magra (La Spezia), 8-10 juin 1994, Paterno, F. (ed) ,Springer Verlag, Berlin.

[Bodart et al. 1994]Bodart,F. et alli. *Dimensions clé pour une méthodologie de développement d'applications interactives*. Proc. of 1st International Eurographics Workshop on Design, Specification and Verification of Interactive Systems, Bocca di Magra (La Spezia), 8-10 juin 1994.

[Bodart et al. 95] Key Activities for a Development Methodology of Interactive Applications.

In:[Benyon & Palanque 95].
[Bodker 91]Bodker, S.; Gronbaek, K. *Cooperative Prototyping : Users and Designers in Mutual Activity*. Int. J. Man-Machine Studies, 34 (1991), 453-78.
[Boehm 81] Boehm, B. Software Engineering Economics, Prentice-Hall Inc, Englewoods Cliffs, N Jersey, 1981.
[Boehm 88] Boehm, B. The Spiral Model of Software development and enhancement. IEEE Computer V.21 N. 5, pp 61-72.
[Boehm 96] Boehm, B. Identifying Quality-Requirement Conflicts, IEEE Software March 1996, pp 25-35.
[Booch 91]Booch, G. *Object-Oriented design with Applications*. Redwood City, Benjamim/Cummings, 1991.
[Booch 94] Booch, G. Object Oriented Analysis and Design with Applications, the Bejnamim/Commings Publishing Company, 1994.
[Boy 95] Boy, G. Coopération Homme-Machine dans les Systèmes Complexes: Agents et Aide à la Decision. In: Actes des Septiemes Journées sur L'Ingenierie de l'Interaction Homme-Machine (IHM '95), Toulouse, France, 11-13 octobre 1995, Cepadues editions.
[Boy 97a] Boy, G.A. Cognitive function analysis: an example of human-centered re-design of a flight management system. International Ergonomics Association 13th Triennial Congress Proceedings, June 29-July 4, Tempere, Finland, 1997.
[Boy 97b] Boy, G.A. Active technical documents. Proceedings of ACM DIS'97.Amsterdam, Holland, August, 1997.
[Boy 97c] Boy, G.A. Cognitive Function Analysis in Human-Centered Automation. HCI International'97. 7th International Conference on Human-Computer Interaction, San Francisco, CA, August 24-29, 1997.
[Boy, 97d] Boy, G.A. (Book to appear in 1997). Cognitive function analysis. Ablex Publishing Corporation, Greenwich, CT, 1997.
[Brandford et al.] Blandford, A. et alli Integratig User Requirements and System Specification. AMODEUS Project Document BC7, 1993.
[Brooks 87/ Brooks, F.P. *No Silver Bullet: Essence and Accidents of Software Engineering*. IEEE Computer, p. 10-19, April 1987.
[Brown 97] Brown, J. Exploring Human-Computer Interaction and Software Engineering Methodologies for the Creation of Interactive Software. In: ACM SIGCHI Bulletin, V. 29, N. 1, January 1997, pp 32-35.
[Brun & Beaudoin-Lafon 95] Brun, P. ; Beaudoin-Lafon, M. A taxonomy and evaluation of formalisms for the specification of interactive systems. Proc. of HCI 95- 10th Annual conference of the British Human-Computer interaction Group, University of Huddersfield, UK, August-september 1995.
[Brunet 91]Brunet, E. *KADS: Méthode d'ingénierie de la Connaissance*. Génie Logiciel & Systèmes Experts, N. 23, Juin 1991.
[Bubenko 94] Bubenko Jr, J. *Enterprise Modelling*. Ingénierie des Systèmes d'Information, V. 2, N. 6, 1994, 657-77 (Special Issue on Requirements Engineering).
[Buckingham- Shum 95] Buckingham-Shum, S. Analysing the Usability of a Design Rationale. In: Design Rationale: Concepts, Techniques and Use. Moran, T.; Carroll, J. (eds) Lawrence Erlbaum Assoc., Hillsdale, 1995.
[CACM 93] *Communications of ACM*, V. 36 N. 4, June 1993. Special Issue on Participatory Design.
[CACM 95] CACM V. 38, N. 9, September 95. Special Issue on Representations of Work.
[CADUI 96] Proc. of 2th International Workshop on Computer-Aided Design of User Interfaces (CADUI'96), Namur, Belgique, 5-7 June 1996, Vanderdonckit, J. (ed.), Presses Universitaires de Namur.

[Campbell 92] Campbell, R. L. *Will the Real Scenario Please Stand Up ?* ACM SIGCHI Bulletin, V.24 N.2, pp 6-8.

[Card et al. 83] Card, S.; Moran, T.; Newell, A. The Psychology of Human Computer Interaction, Lawrence Erlbaum Associates, 1983.

[Carey et al 89] Carey, M. Et alli. Human Computer Interaction Design: the Potential and Pitfalls of Hierarchical task Analysis. In:[Diaper 89].

[Carmel 93] Carmel, E.; Whitaker, R.D.; George, J. *PD and Joint Application Design : A Transatlantic Comparaison*. In :[CACM 93[, pp 40-48.

[Carneiro et al. 93] Carneiro, L.M.F. et alli User Interface High-Order Architectural Models, technical report CS-93-14, University of Waterloo, Ontario, Canada,1993. http://csg.uwaterloo.ca/~statford/ADV/theory.html

[Carrol 91] Carrol,J (ed.) *Designing Interaction : Psychology at the Human-Computer Interface*. New York, Cambridge University Press, 1991.

[Carroll 91] J. Carrol, and M. Rosson Getting Around the Task-Artifact Cycle : How to Make Claims and Design by Scenario. ACM Transactions on Information Systems, 10 (2), 181-212

[Carroll 95] Carroll, J. (ed) Scenario-Based Design: Envisioning Work and Technology in System Development, John Wiley & Sons, London, 1995.

[Chapman 87] Chapman, D.; Agre, P. *Abstract Reasoning as Emerging from Concrete Activity*. In : Georgeff, M.P.;Lansky,A. (eds.) Reasoning about Actions and Plans, Los Altos, Morgan Kaufmann Publisher, pp 411-424, 1987.

[Chen 76] The Entity-Relationship Model - Toward a Unified View of Darta, ACM Transactions on Database Systems, V. 1, N. 1, 1976, 9-36.

[Chin 88] Chin, D.N. KNOMé : Modelling what the user knows in UC. In : Kobsa, A .; Wahlster, W. (eds), User Models in Dialog Systems, Springer-Verlag, pp 74-107, 1988.

[Christel & Kang 92] Christel, M.; Kang, K. Issues in Requirements Elicitation. Technical Report CMU/SEI-92-TR-12, , Software Engineering Institute, Carnegie Mellon University, September 1992.

[Chung 94] Chung, L et alli Using Quality Requirements to Systematically Develop Quality Software. Proc. of Fourth International on Software Quality (ICSQ 94), McLean USA, October 3-5 1994.

[Cisse & Pimenta 96] Alassane Cisse, Marcelo Soares Pimenta. Requirements Engineering as an Emergent Process, In: Proceedings of European Workshop on Software Process Technology EWSPT'96, 9-11 October 1996, Nancy, France, Lecture Notes in Computer Science N. 1149, Springer Verlag.

[Cissé 95] Cissé , A; Link-Pezet, J *Assister le travail coopératif en ingénierie simultanée* (Supporting Cooperative Work in Concurrent Engineering), Colloque Afcet 95, Toulouse, France, October 1995 (In French).

[Cisse 96a] Cissé, A.; Souleymane, N. ; Link-Pezet, J. *Process Oriented Cooperative Work: An Emergent Framework*. International Symposium and Workshop on Engineering of Computer Based Systems (ECBS 96), March 11-15, 1996, Friedrichshafen, Germany.

[Cisse 96b] Cisse A; Ndiaye S; Pezet J L. *Process Oriented CSCW: Dealing With Emergence*, to appear in Proc. of International Conference on Applied Ergonomics (ICAE'96), May 21-24 Istanbul Turkey, 1996

[Cissé et al. 96] Cissé, A.; Souleymane, N. ; Link-Pezet, J. *Process Oriented Cooperative Work: An Emergent Framework*. Proc. of International IEEE Symposium and Workshop on Engineering of Computer Based Systems (ECBS 96), March 11-15, 1996, Friedrichshafen, Germany.

[Clarke 91] Clarke, L. The Use of Scenarios by User Interface Designers. AMODEUS Project Document, CP 17, 1991.

[Clement 93]Clement, A.; van den Besselaar, P. *A Retrospective Look at PD Projects.* In[CACM 93[, pp 29-37.

[Coad 91]Coad, P.; Yourdon, E. *Object-Oriented Analysis*. Prentice-Hall, Englewood Cliffs, 1991.

[Cockton 95] Cockton, G; Clarke, S.; Gray, P. *Theories of Context influence the System Abstractions Used to Design Interactive Systems*. Proc. of HCI'95, Huddersfield, August , 1995, Cambridge Universtity Press, pgs. 387-405.

[Cockton et al. 96] Cockton, G. ; Clarke, S. ; Gray, P. ; Johnson, C. Literate Développement : Weaving Human Context into Design Specifications. In : Benyon, D ; Palanque, P. (eds). Critical Issues in User Interface Systems Engineering, Springer-Verlag, 1996, pags 227-248.

[Cockton et al. 95] Cockton, G; Clarke, S.; Gray, P. *Theories of Context influence the System Abstractions Used to Design Interactive Systems*. Proc. of HCI'95, Huddersfield, August , 1995, Cambridge Universtity Press, pgs. 387-405.

[Conradi 92] Conradi, R. et alli Towards a Reference Framework for Process Concepts. In: [EWSPT 92].

[Constantine 95] Constantine, L. Essentiel Modeling : Use Cass for User Interfaces. ACM Interactions, V.2, N.2, April 1995

[Coutaz 90] Coutaz, J. Interfaces Homme-Ordinateur: Conception et Réalisation, Dunod Informatique, 1990.

[Cox 93]Cox, K.; Walker, D. *User Interface Design*., Prentice Hall, Simon & Schuster Asia, Singapore, 1993.

[Cross 73] Cross, D. The Worker Opinion Survey : A Mesure of Shop-Floor Satisfaction. Occupational Psychology, V.47, N. 3-4, 193-208 , 1973.

[Curtis & Hefley 94]Curtis, B. Hefley, B. *A WIMP No More*. ACM Interactions, V.1 N.1 Jan 1994, pp 22-34.

[Curtis et al. 88] Curtis, B.; Krasner, H.; Iscoe, N. *A Field Study of the Software Design Process for Large Systems*. CACM, 31(11), 1988

[Dankel et al.92] Dankel, D. et alli. *A Model for Capturin Requirements*.Proc. of 5th International Conference on Software Engineering and Its Applications, Toulouse 1992, pp 589-98.

[Dardenne et al. 93] Dardenne, A. ; Van Lamsweerde, A, Fickas, S. Goal-Directed Requirements Acquisition. Science of Computer Programming, 20(1), pp 3-50, 1993.

[Davis 93] Davis, A.M. *Software Requirements: Objects, Functions and States*, Prentice-Hall, 1993.

[Dayton 91] Dayton, T. *Cultivated Eclecticism as the Normative Approach to Design*. In: Taking Software Design Seriously, Karat, J. (Ed.), 1991, Academic Press, USA.

[De Marco 78] De Marco, T. *Structured Analysis and Systems Specification*, Yourdon Press, 1978.

[Denning 86] Denning, W.é. Out of the Crisis, MIT Center for Advanced Engineering, Cambridge, MA, 1986.

[Denning 94] Denning,P.J.; Dargan, P. *A Discipline of Software Architecture*. ACM Interactions, V. 1, N.1, Jan 94, pp 55-65.

[Denning 95] Denning, P. Can There Be a Science of Information ? ACM Computing Surveys, V.27, N. 1, March 1995, 23-25.

[Diaper & Addison 92] Diaper, D.; Addison, M. Task Analysis and Systems Analysis for Software Development. Interacting with Computers 4(1), 1992, 124-39.

[Diaper 89] Diaper, D. (ed) Task Analysis for HCI, Ellis Worwood, Chichester, 1989.

[Dix et al. 93] Dix, A. et alli. *Human-Computer Interaction*, Prentice-Hall International, 1993.

[Dowell & Long 89] Dowell, J.; Long., J. *Towards a Conception for an Engineering Discipline of Human Factors*. Ergonomics **32**, 1513-35, 1989.

[Draper 85]Draper, S.W.; Norman, D. *Software Engineering for User Interfaces* IEEE Transactions on Software Engineering, SE-11, 252-58, 1985.

[Draper 93] Draper, S. The Notion of Task. In Proc. of INTERCHI 93, Ashlund, S. (eds), Addison-Wesley, 93.

[DSVIS-95] Proc. of 2nd Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS 95), Chateau de Bonas, 7-9 June 1995, Bastide, R.; Palanque, P. (eds), Springer Verlag, Vienna, 1995.

[Dubois & Hagelstein 88] Dubois,E.; Hagelstein, J. A Formal Requirements Engineering with ERAE. Philips journal of Research 34(3/4):393-414, 1988.

[Due et al. 91] Due, B.; Jorgensen, A.; Nielsen, J. An Observational Study of User Interface Design Practice. AMODEUS Project Document D 16, 1991.

[Eden 89] Eden, C. Using Cognitive Mapping for Strategic Options Development and Analysis (SODA). In : Rosenhead, J. (ed) Rational Analysis for a Problematic World : Problem Structuring Methods for Complexity, Uncertainty and Conflict, Chichester : John Wiley, 1989, pp 21-42.

[El Emam 95] El Emam, K. Measuring the success or Requirements Engineering Process. In: Proc. of the 2nd IEEE Intl Symposium on Requirements Engineering, 1995, pp 204-211.

[El Emam et al 96] El Emam, K. et alli User Participation in the Requirements Engineerign Process: An Empirical Study. Requirements Engineering Journal, V. 1, N. 1, 1996.

[Elwert & Schlungbaum 95] Elwert, T.; Schlungbaum, E. Modelling and Generation of Graphical User Interfaces in the TADEUS Approach, In:[DSVIS-95].

[EWSPT'94] Proc. of Third European Workshop Software Process Technology, *EWSPT'94* B C Warboys (ed), LNCS 772, Springer-Verlag, 1994.

[EWSPT'92] Proc. of Second European Workshop Software Process Technology, *EWSPT'92* J C Derniame (ed), LNCS 635, Springer-Verlag, 1992

[Falzon 84] Falzon, P. *The Analysis and Understanding of an Operative Language*. In: Shackel, B. (ed.) Proc. of INTERACT ' 84, pp 437-41, Amsterdam, North-Holland, 1984.

[Falzon 90] Falzon, P. *Human-Computer Interaction: Lessons from Human-Human Communication*. Cognitive Ergonomics: Understanding, Learnig and Designing Human-Computer Interaction, pp 51-66, European Association of Cognitive Ergonomics, 1990.

[Farenc et al.95] Farenc, C.; Palanque, P.: Vanderdonckt, J. *Ergonomic Evaluation of Software Usability: Is it Ever Usable?* Proc. of HCI International '95, Tokyo, 1995.

[Faust & Pimenta 92]Faust, R.; Pimenta, M. *Software Development as a Learning Process*. In : INFONOR'92, International Symposium on Applications of Information Technology, 1992, Antofagasta, Chile.

[Faust & Pimenta 93]Faust, R., Pimenta, M.S. *Rumo a uma Visão Ergonômica do Desenvolvimento de Software*. (Towards to an Ergonomic Viewpoint of Software Development). Anais do Segundo Congresso LatinoAmericano de Ergonomia, 1993, pp 312-14. (In Portuguese).

[Faust & Pimenta 95] Faust,R.; Pimenta, M.S. *An Ergonomic Approach to Software Development: Focusing on Communication*. In: Proc. of International Ergonomics Association World Conference 1995, Rio de Janeiro, Brazil, October 16-20, 1995, p. 402-406.

[Faust & Pimenta 96a] Faust, R.; Pimenta, M; *Interactive Systems Requirements Elicitation: A Semiotic Approach* In: Proc. of User Centred Requirements Engineering Workshop:Integrating Methods from Software Engineering and Human Computer Interaction, 15 January 1996, York, United Kingdom, (unpaged).

[Faust & Pimenta 96b] Richard Faust, Marcelo Soares Pimenta. A Semiotic Approach to

Information System Requirements Elicitation, In: Proc.of First International Conference on Applied Ergonomics (ICAE'96), May21-24 Istanbul Turkey, 1996.

[Faust 92] Faust, R. *Domain Object Model - Definition and a Case Study*. Technical Report 03/92. Computer Science Department, Universidade Federal de Santa Catarina, Brazil, 1992. (In Portuguese). Available directly with the author.

[Faust 95] Faust, R., *Software as Sign Interpretation: A User Linguistic-Register-Centered Strategy for Software Development*, Master Thesis, Federal University of Santa Catarina, Brazil , 1995 (In Portuguese).

[Feiler 93] Feiler, P.; Humphrey, W. Software Process Development and Enactment: Concepts and Definitions. Proc. of ICSP 1993.

[Filkenstein et al. 92] Filkenstein, A. et alli. *Viewpoints: a Framework for Integrating Multiple Perspectives in System Development*. Intl. Journal on Software Engineering and Knowledge Engineering, 2(1), 1992.

[Fillmore 68] Fillmore, Ch. J.,*The case for case*. In: Bach,E. & Harms,R.T. (eds.), Universal in Linguistic Theory, Holt, Rinehart and Winston, London, 1-90, 1968.

[Fischer 89] Fischer, G. *Human-Computer Interaction Software: Lessons Learned, Challenges Ahead.* IEEE Software, January 1989, pp. 44-51.

[Fischer 90] Fischer, G.; Lemke, A.C.; Mastaglio, T.; Morch, A. Using Critics to Empowers Users', CHI'90, Seattle, WA, 1990, pp 337-347.

[Fischer et al. 93] Fischer , G. Et alli Embedding Computer Based Criticas in the Context of Design. In: Proc. of the Conference on Human Factors in Computing Systems INTERCHI 93, Amsterdam 24-29 April 1993, S. Ashlund (ed), ACM Press, New York, 1993.

[Floyd 88] Floyd, C. *Outline of a Paradigme Change in Software Engineering*. ACM SIGSOFT Software Engineering Notes, V.13, N. 2, April 1988, 25-38.

[Floyd, 1989]FLOYD, C. *Out of Scandinavia : Alternative Approaches to Software Design and System Development*, Human-Computer Interaction, V. 4, pp. 253-350, 1989.

[Foley 82]Foley, J. ; Van Dam, A. *Fundamentals of Interactive Computer Graphics*. Reading, Mass : Addison-Wesley, 1982.

[Foley 87]Foley, J. *Transformation on a Formal Specification of User Computer Interfaces.* Computer Graphics, 21(2), April 87, pp 109-113.

[Foley et al. 91] Foley, J. e alli. UIDE - An Intelligent user Interface Design Environment. In: Sullivan , J. (ed) Intelligent User Interfaces, Addison Wesley, ACM Press, 1991.

[Franco & Leite 92] Franco, A. P. and Leite, J. *Uma Estratégia de Suporte à Engenharia de Requisitos*[A Supporting Strategy for Requirements Engineering]. XII Congresso da Sociedade Brasileira de Computação, Rio de Janeiro, RJ, Brazil, October 1922, pp. 200-213.[in Portuguese]

[Freund 85] Freund, R. Definitions and Basic Quality Concepts. In : Sepehri, M . (ed). Quest fort Quality : Managing the Total System, Atlanta, Georgia, Industrial Engineering and Management Press, 1985.

[Galegher 92]Galegher, J.; Kraut, R.. *Computer-Mediated Communication and Collaborative Writing : Media Influence and Adaptation to Communication Constraints*. Proc. of CSCW'92, Toronto, pp 155-62, 1992.

[Garfinkel 67] Garfinkel H. *Studies in ethnomethodology*, Englewood Cliffs, Prentice Hall, 1967.

[Ghezzi 91] Ghezzi, C. et alli *Fundamentals of Software Engineering*, Prentice-Hall, 1991.

[Gilmore 95] Gilmore, D. *Interface Design: Have We Got It Wrong?* Proc. of INTERACT' 95, 5th IFIP Intl. Conf. on Human-Computer Interaction, Lilhammer, Norway, June 1995, pp 173-78.

[Goguen 94] Goguen, J. Requirements Engineering: Reconciliation of Technical and Social Issues. Tech. Rep., Centre for Requirements and Foundations, Oxford University

Computing Lab, Cambridge, UK, 1992.
[Goransson et al. 87] Goransson, B. et alli. *The interface is Often Not the Problem*. In: Carrol, J; Tanner, P (eds.). Proc. of CHI+GI '87 Human Factors in Computing Systems and Graphics Interface, ACM Press, New York, 133-36, 1987.
[Gotel et Filkenstein 94a] Gotel, O. ; Filkenstein, A. An Analysis of the Requirements Traceability Problem. Proc . of First IEEE International Conference on Requirements Engineering (ICRE 94), Colorado Springs, Colorado, 18-22 April, pp 94-101.
[Gotel et Filkenstein 94b] Gotel, O. ; Filkenstein, A. Contribution Structures, Technical Report, Department of Computing, Imperial College, 1994 (ftp://dse.doc.ic.ac.uk/dse-papers)
[Gould 88] Gould, J. How to design Usable Systems. In: Helander, M. (ed) handbook of Human Computer Interaction, North Holland, 1988.
[Gram & Cockton 95] Gram, C.; Cockton, G. Design Principles for Interactive Software, Chapman & Hall, 1995.
[Gray et al 94] Gray, P. et alli. XUAN: Enhancing UAN to Capture temporal relationships among Actions, In: Proc. of British Conference on Human Computer Interaction HCI 94, Glasgow, 23-26 August 1994, Cockton, G (ed), Cambridge University Press, Cambridge, 1994.
[Gray et al. 94] Gray, P.; England, D.; McGowan, S. XUAN: Enhancing the UAN to capture temporal relation among actions. Department of Computing Science, University of glasgow, February 1894, Department research report IS-94-02.
[Green 86]Green, M. *A Survey of Three Dialogue Models*. ACM Transactions on Graphics, New York, V. 5, N. 3, p. 244-75, July 1986.
[Grenn 86] Green, M. A Survey of Three Dialogue Models, ACM Treansactions on Graphics, V. 5, N. 3, July 1986, 244-275.
[Gronbaek 93]Gronbaek, K.; Kyng, M.; Mogensen, P. *CSCW Challenges : Cooperative Design in Engineering Projects*. In : [CACM 93[, pp 67-77.
[Gronquist et al. 92] gronquist, B. A Component Approach for Reuse. In : Proc. of Intl. Conference on Software Engineering and Its Applications, Toulouse 1992, pp 359-67.
[Gruber 93] Gruber, T. A translational approach to portable ontologies. Knowledge Acquisition V.5, N.2, 1993, pag. 199-220.
[Grudin 91]Grudin, J. *Interactive Systems : Bridging the Gaps between Développers and Users*. IEEE Computer, April 1991, pp 59-69.
[Hackman et Oldhan 80] Hackman, J. ; Oldham, G. Work Redesign, Reading Massachussetts, Addison Wesley ,1980.
[Hammouche 95] Hammouche, H. De la modélisation des tâches utilisateurs à la spécification conceptuelle d'interfaces Homme-Machine, Thèse de doctorat, Université Paris 6, 1995.
[Hartson & Hix 1989] Hartson, R ; Hix, D. *Human-Computer Interface Development : Concepts and Systems for Its Management*. ACM Computing Surveys, New York, v.21, n. 1, p. 5-92, March 1989.
[Hartson & Hix 89] Hartson, R.; Hix, D. Human-Computer Interface Development: Concepts and Systems for Its Management. ACM Computing Surveys, 21(1),pp 5-92.
[Hartson et al 90] Hartson, R. et alli The User Action Notation: A User-Oriented Representation for Direct Manipulation Interfaces, ACM Transactions on Information Systems, V. 8, N. 3, 1990, 181-203.
[Hayenhjelm 93] Hayenhjelm,H. *The Ergonomic Design Process - the User Involved in Problem Solving and the Design Évolution*. Proc. of HCI'93, pp. 966-69, 1993.
[Hix & Hartson 93] Hix, D., Hartson, H.R. *Developing User Interfaces Ensuring Usability through Product and Process*, John Wiley & Sons, 1993.
[Hix et Hartson 93] Hix, D.; Hartson, R. Developing User Interfaces - Ensuring usability

Through Product and Process, John Wiley & Sons, New York, 1993.
[Hoare 85] Hoare, C.A.R. Communicating Sequentiel Process, Englewood Cliffs, NJ, Prentice-Hall, 1985.
[Holbrook 90] Holbrook, H. A Scenario-Based methodology for Conducting Requirements Elicitation. ACM SIGSOFT Software Engineering Notes 15(1) :95-104, January 1990.
[Hsia et al. 94] Hsia, P. et alli. Formal Approach to Scenario Analysis, IEEE Software, March 1994.
[Hughes 95] Hughes, J. ; King, V. ; Rodden, T. ; Anderson, H. The Role of Etnography in Interactive Systems Design. ACM Interactions, V. 2, N. 2, April 1995.
[Hutchins 90] Hutchins E, *The Technology of Team Navigation*, in Intellectual Teamwork, J Galagher, R E Kraut & C Egido (Eds), Lawrence Elbaum Associates, 1990.
[ICSE 94] Proc. of the Software Engineering and Human-Computer Interaction ICSE 94 Workshop, Sorrento, 16-17 May 1994, Coutaz, J. (ed) Lecture Notes in Computer Science, V. 896, Springer Verlag, Berlin, 1995.
[IEEE 83] IEEE Standard Glossary of Software Engineering Terminology, IEEE Standard 729, 1983.
[IEEE 84] IEEE Guide to Software Requirements Specification. IEEE/ANSI Standard 830-1984, Institute of Electrical and Eletronics Engineers, New York, 1984.
[IEEE 90] Institute of Electrical and Eletronics Engineers. IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard 610.12-1990 (revision and redesignation of IEEE Std. 729-1983), Institute of Electrical and Eletronics Engineers, New York, 1983.
[Iris et al. 92] Iris, J.M.; Costes, B. *Formalisation de l'Expression du besoin : la méthode ARC2*. Proc. of 5th International Conference on Software Engineering and Its Applications, Toulouse 1992, pp 81-92.
[ISO 94] ISO Draft International Standard 9241-11. Ergonomic Requirements for Office Work with Visual Display Terminals - Part 11 Guidance on Usability, September 94.
[Jackson 93] Jackson, M. System Development, Prentice Hall, London,1983.
[Jackson 95] Software Requirements and Specifications, Addison-Wesley, Readings 1995.
[Jacobson 92] Jacobson, I. et alli. *Object Oriented Software Engineering*. ACM Press, Addison-Wesley, 1992.
[Jambon 96] Jambon, F. Erreurs et Interruptions du Point de Vue de l'Ingenierie de l'Interaction Homme-Machine, Thèse en Informatique, Université Joseph Fourier (Grenoble 1), Decembre 1996.
[Jarke et al. 93]Jarke, M. et alli. *Requirements Engineering : An Integrated View of Représentation, Process and Domain*. Proc. of 5th European Software Engineering Conference, Garmish-Paternkirchen, Germany, September 1993.
[Jarke et al. 94]Jarke, M. et alli. *Requirements Information management : the NATURE approach*. Ingénierie des Systèmes d'Information, V. 2 N. 6, pp 609-637, 1994. (Special Issue on Requirements Engineering).
[John & Kieras 94] John, B. ; Kieras, D. The GOMS family of analysis techniques : Tools for Design and Evaluation. Carnegie Mellon University School of Computer Science Technical Report No. CMU-CS-94-181. Also appears as the Human Computer Institute Technical Report No. CMU-HCII-94-106.
[P. Johnson 92] Johnson, P. *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering*, Mc-Graw Hill, Maidenhead, UK, 1992.
[H. Johnson & P. Johnson 90] Johnson, H.; Johnson, P. Designers-identified requirements for tools to support task analyses. In: Proc. of INTERACT 90, 3rd IFIP Conference on HCI, August 1990, Elsevier North-Holland, 1990.
[H. Johnson & P. Johnson 91] Johnson, H.; Johnson, P. Task Knowledge Structure:

Psychological basis and Integration into System Design. Acta Psychologica, 78, 1991, pp 3-26.

[P. Johnson & H. Johnson 93] Johnson, P. ; Johnson, H. ADEPT - Advanced Design Environment for Prototyping with Task Models. Human factors in Computing Systems (INTERCHI 93), Amsterdam, 24-29 april 1993, vol. Adjunct Proc(2),p. 56.

[J. Johnson & Nardi 96] Johnson, J. ; Nardi, B. Creating Presentation Slides : A Study of User Preferences for Task- Specific versus Generic Application Software. ACM Transactions on Computer-Human Interaction, V. 3, N. 1, Mar 1996, 38-65.

[P. Johnson 94] Johnson, P. *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering*, Mc-Graw Hill, Maidenhead, UK, 1994.

[C. Johnson 96] Johnson, C. The Namur Principles: criteria for the evaluation of user interface notations. Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS 96), Namur, Belgique, 1996.

[C. Johnson & Jones 97] Johnson, C; Jones, S. Human Computer Interaction and Requirements Engineering: papers from an interdisciplinary workshop, ACM SIGCHI Bulletin, V. 29, N. 1, January 1997.

[P. Johnson et al 95] Johnson, P et al. Rapid Prototyping of User Interfaces Driven by Task Models. In: [Carroll 95].

[P. Johnson et al. 88] Johnson, P.; Johnson, H.; Waddington, R. and Shouls,A. *Task-Related Knowledge Structures: Analysis, Modelling and Application*. In: D.M.Jones; R. Winder (eds.). People and Computers: From Research to Implementation, HCI'88, Cambridge University Press, pp 137-55.

[Jones 95] Jones, M. Organizational Analysis and HCI. In: Perspectives on HCI: Diverses Approaches. Monk, A.; Gilbert, N. (eds) Computer and People Series, 1995.

[Jordan 94] Jordan, P.W. What is Usability ? In : Robertson, S.A (ed) Contemporary Ergonomics. Taylor & Francis, 1994. Proc. Of the Ergonomics Society's 1994 Annual Conference, Univ. Of Warwick, 19-22 April 1994.

[Jordan 94] Jordan, P.What is Usability? In: Robertson, S. (ed) Contemporary Ergonomics, Taylor & Francis, 1994. Proc. of the Ergonomics Society's 1994 Annual Conference, Univ. of Warwick, 19-22 April 1994.

[Kaindl 95a] Kaindl, H. An approach to hypertext-based requirements specification and its application. In: Proc. of 5th Intl. Conf. On Human Computer Interaction (INTERACT 95), Lillehammer, Norway, June 1995, IFIP.

[Kaindl 95b] Kaindl, H.An Integration of Scenarios with Their Purposess in Task Modelling. Proc. of DIS 95, ACM Symposium on Designing interactive systems: Processes, practices, methods & techniques,Ann Arbor, Michigan, 23-25 august 1995.

[Kammersgaard 1993]Kammersgaard, J. *Four Différents Perspectives on Human-Computer Interaction*, International Journal of Man-Machine Studies, V. 28, pp. 343-362, 1988.

[Karat 91]Karat, J. (ed.)*Taking Software Design Seriously*, Academic Press, Hancourt Brace Jovanovich, 1991.

[Karat 92] Karat, C.M. ; Karat, J. *International Perspectives : Some Dialogues on Scenarios*. ACM SIGCHI Bulletin, V.24 N.4, pp 7-17.

[Karat 96/ Karat, J. User Centred Design : Quality or Quackery ? ACM Interactions, July-August 1996, pp 19-20.

[Katzenberg 93] Katzenberg, B.; Piela, P. *Work Language Analysis and the Naming Problem*. Comm. of ACM, V.36, N. 4, June 1993.

[Kavakli et al. 96] Kavakli, E. ; Loucopoulos, P. ; Filippidou, D. Using Scenarios to Systematically Support Goal-Directed Elaboration for Information System Requirements. Proc . of ECBS 96, pp 308-314.

[Kensing 93]Kensing, F.; Munk-Madsen, A. *PD : Structure in the Toolbox*. In : [CACM 93],

pp 78-85.
[Kieras 88] Kieras, D. Towards a practical GOMS model methodology for user Interface design. In : M. Helander (ed.) handbook of Human Computer Interaction, North-Holland Elsevier, 1988, pp 135-158.
[Kirsh 94] D. Kirsh, P. Maglio. *On Distinguishing Epistemic From Pragmatic Action* . Cognitive Science, 18, 513-549, 1994.
[Krasner 88] Krasner, H. Requirements Dynamics in Large Software Projects, A Perspective on New Directions in the Software Engineering Process. Proc of IFIP, Elsevier, New York, 211-16.
[Kuhn, 1969] Kuhn. T.S. *The Structure of Scientific Revolutions.* University of Chicago Press, 1969.
[Kuuti & Bannon 93] Kuuti,K.; Bannon,L. *Searching for Unity among Diversity : Exploring the "Interface" Concept.* Proc. of INTERCHI'93 Conference on Human Factors in Computing Systems, Amsterdam, the Netherlands, 24-29 April 1993, New York, ACM Press, pp 263-68, 1993.
[Kyng, 1991]KYNG, M. *Designing for Cooperation : Cooperating in Design.* Communications of the ACM, pp. 65-73, December 1991.
[Lakatos, 1970] Lakatos, I. and Musgrave, A. eds. *Criticism and the Growth of Knowledge.* Cambridge University Press, Cambridge, UK, 1970.
[Landay & Myers 95] Landay, J. ; Myers , B. *Interactive sketching for the early stages of user interface design*. In : Proc. of CHI 95 : Human Factors in Computing Systems, Denver, May 1995, pp 43-50.
[Laurel 90]Laurel, B.(ed.) *The Art of Human-Computer Interface Design*, Addison-Wesley, 1990.
[Laurel, 1991] Laurel, B. *POSTSCRIPT: On Vision, Monsters and Artificial Life* in Laurel, B. ed. *The Art of Human-Computer Interface Design*. Addison-Wesley, NY, 1990.
[Lave 88] Lave J, *Cognition in practice*, Cambridge University Press, 1988.
[Lehmann 80]Lehmann, M.M. Programs, *Life Cycles and Laws of Software Évolution.* Proc. IEEE 68, 9 (1980).
[Leite & Franco 93] Leite, J.C.S.P.; Franco, A.M. A Strategy for Conceptual Model Acquisition. Proc. of the First IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 243-46.
[Leite 87] Leite, J.C.S.P. A Survey on Requirements Analysis. Technical Report RTP-071, University of California at Irvine, Departement of Information and Computer Science , 1-235, June 1987.
[Leite 90]Leite, J.C.S. *Validação de Requisitos : o Uso de Pontos de Vista*. (Requirements Validation : the Use of Viewpoints) Revista Brasileira de Computação, V. 6 N. 2 Out/Dez 1990 pp 39-52. (In Portuguese)
[Leite 93] Leite, J.C; Franco, A.P.M. *A Strategy for Conceptual Model Acquisition*. Proc. of the First IEEE International Symposium on Requirements Engineering: (1993), IEEE Computer Society Press, pgs. 243-46.
[Leite, 1989] Leite,J. *Application Languages: A Product of Requirements Analysis*. Departamento de Informática, PUC/RJ, Brazil, January 1989.
[Leite, 1991] Leite,J. Franco A. *Re-Engenharia de Software: Um Estudo de Caso[*Software Re-engineering: A Case Study]. V Simpósio Brasileiro de Engenharia de Software, Ouro Preto, MG, Brazil, 1991. (in Portuguese)
[Lewis et al. 92] Lewis, C. et alli. Cognitive Walkthroughs: a Method for Theory-based Evaluation of User Interfaces, CHI'92, Tutorial Notes, Monterey 4 May 1992, NY, ACM Press, 1992.
[Lim & Long 94] Lim, K.; Long, J. The MUSE Method for Usability Engineering, Cambridge

University Press, Cambridge, 1994.
[Lim 92]Lim, K. Y.; Long, J. B. *Instanciation of Task Analysis in a Structured Method for User Interface Design*. Proc. of Task Analysis in HCI, Austria, June 1992.
[Lim et al. 92] Lim, K. et alli. Integrating Human Factors with the Jackson System Development Method: An Illustrated Overview, Ergonomics, V. 35, N. 10, 192, 1135-1161.
[Lindquist 85]Lindquist, Timothy. *Assessing the usability of Human-Computer Interfaces*. IEEE Software, pp 74-82, Jan 1985.
[Long 95] Long, J. Integrating Human factors with Software Engineering for Human Computer Interaction. 7emes Journées de l'Ingenierie de l'Interaction Homme-Machine, (IHM '95), 11-13 octobre 1995, Toulouse, 1995. (Conférencier Invité).
[Long and Dowell 89] Dowell, J.; Long., J. *Towards a Conception for an Engineering Discipline of Human Factors*. Ergonomics **32**, 1513-35, 1989.
[Longchamp 92] Longchamp J. *Supporting social interaction activities of software processes*. LNCS N. 635: Software Technology, J C Derniame (ed), Springer-Verlag, 1992.
[Longchamp 94] Longchamp J. *A Process-Centered Framework for Asynchronous Collaborative Work*. In:[ESWPT94].
[Loucopoulos & Potts 96] Loucopoulos, P. ; Potts, C. Editorial. Requirements Engineering Journal, V.1, N.1, 1996.
[Lu 93] Lu, T. User Interface Software Requirements Specification, Technical Report, Queen's University, Canada, April 1993.
[Macaulay 95] Macaulay, L. *Human-Computer Interaction for Software Designers*. International Thomson Publishing, London, UK, 1995.
[MacLean et al. 91] MacLean, A. ; Young, R. ; Bellotti, V. ; and Moran, T. *Questions, Options, and Criteria : Elements of Design Space Analysis*. In : Human-Computer Interaction 6(3-4) :201-250, 1991.
[Madsen 93]Madsen, K. H.; Aiken, P. *Experiences using Cooperative Interactive StoryBoard Prototyping*. In : [CACM 93[, pp 57-64.
[Maiden 92]Maiden, N.; Sutcliffe, A. *Exploiting Reusable Specifications Through Analogy*. CACM, V.35, N. 4, April 92, pp 55-64.
[Marcus 92] Marcus, A. *Graphic Design for Eletronic Documents and User Interfaces*. New York, ACM Press, 1992.
[Mattos, 1989]Mattos, N. *An Approach to Knowledge Base Management*. University of Kaiserslautern, Germany, 1989.
[Maturana 80] Maturana, H.; Varela, F. *Autopoiesis and cognition*, D. Reidel, Boston, 1980.
[Maturana 87] Maturana, H. ; Varela, F. *The Tree of Knowledge*, Shambala, Boston, 1987.
[McMenamin & Palmer 84]McMenamin, S.M.;Palmer, J.F. *Essential Systems Analysis*. Englewood Cliffs, NJ : Yourdon Press, 1984.
[Meyer, 1988] Meyer, B. *Object-Oriented Software Construction.* Prentice-Hall, New Jersey, 1988.
[Mogensen 93]Mogensen, P; Trigg, R. *Artifacts as triggers for Participatory Analysis*. Proc. PDC 92 (Boston, Dec 1991), Kuhn, Muller and Meskill (eds.), pp 55-62.
[Moran & Carroll 94] Moran, T. ; Carroll, J. Design Rationale : Concepts, techniques and Use. Hillsdale, NJ : Lawrence Erlbaum, 1994.
[Moran 81]Moran, T. *The Command Language Grammar : A Représentation for the User Interface of Interactive Computer Systems*, International Journal of Man-Machine Studies, v. 15, pp. 3-50, 1981.
[Mottay 96] Mottay, Didier . Contribuition à l'Etude de l'Impact de la mmodification du système d'information sur les caracteristiques du travail, ESUG-Université Toulouse 1, Juin 96.

[Mullery 96/ Mullery, G. The Perfect Requirement Myth. Requirements Engineering Journal 1 :132-134, Springer Verlag, 1996.
[Mumford 83] Mumford, E. Designing Human Systems for New Technology: the ETHICS Method, Manchester Business School Press, Manchester, 1983.
[Mumford 83] Mumford, E. Designing Human Systems for New Technology: The ETHICS Method. Manchester, Manchester Business School Press, 1983.
[Myers 88] Myers, B. *Creating Users Interfaces by Demonstration*. Academic Press, 1988.
[Myers 94]Myers, B. *Challenges of HCI Design and Implementation*. ACM Interactions, V.1 N.1 Jan 94, pp 73-83.
[Nadin 88] Nadin, M. *Interface Design: a Semiotic Paradigm*. Semiotica 69, 3[4, pp 269-302, 1988.
[Nardi 1996] Nardi, B. Context and Consciousness : Activity Theory and Human-Computer Interaction, MIT Press, Cambridge, Mass, 1996.
[Newman & Lamming 95] Newman, W.; Lamming, M. Interactive System Design, Addison-Wesley, 1995.
[Nielsen 93] Nielsen, J. Usability Engineering, Academic Press, 1993.
[Norman 86] Norman, D. Cognitive Engineering. In: Norman, D.; Draper, S. (eds) user Centred System Design: New perspectives on Human-Computer interaction, Hillsdale, NJ:Erlbaum, 1986.
[Norman 88] Norman, D. Psychology of Everyday Things, Basic Books Publi., 1988.
[Normand 1992]Normand, V. *Le Modèle SIROCO : de la spécification conceptuelle des interfaces utilisateur à leur réalisation*.. Thèse de Doctorat, Université Joseph Fourier - Grenoble I, 1992.
[Nosek 88] Nosek, J. ; Schwartz, R. User Validation of Information System Requirements : Some Empirical Results. IEEE Transactions on Software Engineering,14 (9) : 1372-1375, September 1988.
[Nygaard 87]Nygaard, K. *Program Development as a Social Activity*. In : Fuchs,K.; Kittowski, D. (Eds.) System Design for ,Human Development and Productivity : Participation and Beyond, Berlin, 1987. (apud[Floyd 89[).
[O'Leary 96] O'Leary, D. Impediments in the use of explicit ontologies for KBS development. International Journal Human Computer Studies, 1996, http://www.usc.edu/dept/sba/atisp/AI/ontology/
[O'Leary et alli 97] O'Leary, D. ; Kuokka, D. ; Plant, R. Artificial Intelligence and Virtual Organizations, CACM V.40 N.1, Jan 1997, pag. 52-59.
[Olsen 95] Olsen, D. Interacting with Information. In:[DSVIS-95]
[Olson & Olson 90] Olson, J.R. ; Olson, G.M. The Growth of Cognitive Modelling in Human Computer Interaction since GOMS, *Human Computer Interaction*, 5, 221-265, 1990.
[O'Neill 95] O'Neill, E. CUSTARD: A Participatory Approach to Usability Requirements Synthesis for Software Design. In: RE'95 Doctoral Consortium, Proc. of 2nd IEEE International Symposium on Requirements Engineering, March 27-29 1995, York, UK, IEEE CS Press, 1995.
[Palanque & Bastide 95] Palanque, P.;.Bastide, R. Verification of an Interactive Software by analysis of its formal specification, IN: Proceedings of the IFIP Human-Computer Interaction conference (Interact'95) Lillehammer, Norway., 27-29 June 1995, p. 191-197.
[Palanque 92]Palanque, P. *Modélisation par Objets Coopératifs Interactifs d'Interfaces Homme-Machine Dirigées par l'Utilisateur*. Thèse de 3ème Cycle en Informatique, Université Toulouse I, Septembre 1992.
[Palanque 94] Palanque, P. et alli. Ergonomic Application Design: A Method for Computer Science and a Method for Humand Factors. In: Proc. of ERGO-IA 94, Biarritz October 1994. (In French).

[Palanque 95]Palanque, P. et alli. *Task Model - System Model : Towards an Unifying Formalism*. HCI International '95, Tokyo, 1995.

[Palanque et al. 96] Palanque, P. et alli. Towards an Integrated Proposal for Interactive Systems Design Based on TLIM and ICO. In: Proc. of 3rd Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS 96), Namur, Belgique, 5-7 June 1996, Bodart, F.; Vanderdonckt, J. (eds), Springer Verlag, New York, 1996.

[Payne & Green 86] Payne, S.J.; Green, T.R. *Task Action Grammar: a Model of the Mental Representation of Task Languages*, Human-Computer Interaction, 2, pp 93-133, 1986.

[Payne 84] Payne, S. Task Action Grammars. In: Shackel, B. (ed) Proc. of Human Computer Interaction - INTERACT 84, North Holland, 1984.

[Peterson 91] Peterson, A.S. *Model-Based Software Reuse Terminology*. Software Engineering Notes, 16(2), pp 45-51, April 1991.

[Pierce 31] Pierce , C. S. Collected Papers, Harvard University Press, cambridge, 1931-1958.

[Pimenta & Barthet 96a] Marcelo Soares Pimenta, Marie-France Barthet. Task Organizational Context Modelling: An Approach for Improving Interactive Systems Usability , In: Proc. of First International Conference on Applied Ergonomics (ICAE'96), May 21-24 Istanbul Turkey, 1996.

[Pimenta & Barthet 96b] Marcelo Soares Pimenta, Marie-France Barthet. Context Modelling for an Usability Oriented Approach to Interactive Systems Requirements Engineering , In: Proc. of IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'96), Friedrichshafen, Germany, March 1996.

[Pimenta & Faust 97] Pimenta, M.S.; Faust, R. Eliciting interactive Systems Requirements in a Language-Centred User Designer Collaboration. ACM SIGCHI Bulletin, V. 29, N. 1, January 1997.

[Pimenta et al. 92] Pimenta, M.; Gontijo, L.; Rosa, S. *An Ergonomic Approach for User Interface and Application Integrated Development: a Case Study*. INFONOR'92, International Symposium on Applications of Information Technology, Antofagasta, Chile, 1992. (In Spanish).

[Pinheiro & Goguen 96] Pinheiro, F.K; Goguen, J. An Object-Oriented Tool for Tracing Requirements. IEEE Software, March 1996.

[Pohl 93]Pohl, K. *The Three Dimensions of Requirements Engineering*. Proc. 5th International Conference Advanced Information Systems Engineering, Paris, 1993.

[Poltrock & Grudin 94] Poltrock, S.E.; Grudin, J. *Organisational Obstacles to Interface Design and Development: Two Participant Observer Studies*. ACM Transactions on Computer-Human Interaction V.1, N.1, March 1994.

[Potts 94] Potts, C. ; Takahashi, K., Anton, A. Inquiry-Based Scenario Analysis of System Requirements. Georgia Tech College of Computing Technical Report, GIT-CC-94/14, January 1994.

[Potts 95] Potts, C. Using Schematics Scenarios to Understand User Needs. Proc. of Symposium on Designing Interactive Systems: Processes, Practices, Methods & Techniques DIS'95, Ann Arbor, USA, August 23-25, 1995, pp. 247-56.

[Potts 97] Potts, C. Requirements Models in Context. In:Proc. of 3rd IEEE Int. Symposium on Requirements Engineering (RE 97), Annapolis, 6-10 january 1997, pp 102-104.

[Potts et al. 94] Potts, C. Et alli. Inquiry-Based Requirements Analysis, IEEE Software, March 1994.

[Prieto-Diaz 1990] Prieto Diaz, R., Domain Analysis : an Introduction. Sw Eng Notes 15, 2 April 1990, pp 47-54

[Prieto-Díaz, 1987/ Prieto-Díaz, R. *Domain Analysis for Reusability*. COMPSAC'87 Proceedings, Tokyo, Japan, October 1987, pp. 23-29.

[Procter & Willians 92] Procter, R.N.; Williams, R.A. *HCI: Whose Problem Is IT Anyway?* Engineering for Human-Computer Interaction, Larson, J.;Unger, C. (eds), North Holland, Amsterdan, 1992, pgs. 385-96.

[Puerta 96] Puerta, A. The MECANO Project: Comprehensive and Integrated Support for Model-based interface Development. In:[CADUI 96].

[Rasmussen 83] Rasmussen, J. Skills, Rules and Knowledge, Signals, Signs and Symbols, and Other Distinctions in Human Performance. IEEE Transactions on Systems, Men and Cybernetics, SMC-13, 3, 233-257, 1983.

[REAW 91] Proc. of Requeriments Engineering and Analysis Workshop. Technical Report CMU/SEI-91-TR-30, Software Engineering Institute, Carnegie Mellon University, December 1991.

[Regnell et al. 95] Regnell, B. et alli. Improving the Use Case Driven Approach to Requirements Engineering. Proc. of Second International Symposium on Requirements Engineering, York, UK, March 1995.

[Rolland 94] Rolland, C. *Reformuler les Démarches de Conception des Systèmes d'Information* (Revising Information Systems Design Approaches), Ingénierie des Systèmes d'Information (Special Issue on Requirements Engineering), V. 2, N. 6, 1994.

[Rosson & Carroll 95] Rosson, M.; Carroll, J. Integrating Task and Software Development for Object Oriented Applications. In: Proc. of Conference of Human Factors in Computing Systems CHI 95, Denver 7-11 May 1995, Katz, I. (ed), ACM Press, New York, 1995.

[Rosson & Carroll 95] Rosson, M.B.; Carroll, J. Integrating task and Software Development for Object-Oriented Application. In: Proc. of CHI 95 Human Factors in Computing Systems, Denver 7-11 May 1995, ACM Press.

[Rubin & Goldberg 92]Rubin, K.S.; Goldberg, A. *Object Behaviour Analysis*. CACM, Sept 1992, pp 48-62.

[Rumbaugh 91] Rumbaugh, J. et alli. *Object Oriented Modelling and Design*. Englewood Cliffs, Prentice-Hall, 1991.

[Rzepka 89/ Rzepka, W. A Requirements Engineering Testbed : Concept, Status and First Results. In : Shriver, B (ed) Proc. of the Twenty-Second Annual Hawaii International Conference on Systems Sciences, 339-347. IEEE Computer Society, 1989.

[Sabah 88] Sabah, G. L'Intelligence Artificielle et le Langage, Hermes, 1988.

[Salber 95] Salber, D. De l'Interaction Homme-Machine Individuelle aux Systèmes Multi-Utilisateurs: l'Exemple de la Communication Homme-Homme Mediatisée, Thèse en Informatique, Université Joseph Fourier (Grenoble I), 1995.

[Salvador 95] Salvador, A.C.; Scholtz, J.C. Systematic Creativity: a Methodology for Integrating User, Market and Engineering Requirements for Product Definition, Design and Usability Testing. In: Bass, L.; Unger, C. (eds) Engineering for Human-Computer Interaction, Chapman & Hall, 1996. Proc. Of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, Yellowstone Park, USA, August 1995.

[Scapin & Pierret-Golbreich 89] Scapin, D.; Pierret-Golbreich, C. *Towards a Method for Task Description: MAD*. In: Work With Display Units'89, Amsterdam, Elseiver, 1989.

[Scapin 86]Scapin, D. *Guide Ergonomique de Conception des Interfaces Homme-Machine*. Rapport de Recherche N.77, INRIA - Rocquencourt , 1986.

[Scapin 93]Scapin, D. *The Need for a Psycho-Engineering approach to HCI*. 2nd Latin American Conference on Ergonomics, October 10-13, 1993, Florianopolis, Brasil. (Invited Lecture).

[Schlungbaum & Elwert96] Schlungbaum, E.; Ewert, T. Automatic user Generation from Declarative Models. In:[CADUI 96].

[Schwab & Cummings 73] Schwab, D. ; Cummings, L. Theories of Performance and

Satisfaction : A Review. In : W.é.Scoot & L. Cummings (eds) Readings in Organisational Behaviour and Human performance, Homewood : Richard D. Irwin, Inc., 1973.
[Sebillote 88] Sebillote, S. Hierarchical Planning as Method for Task Analysis: the Example of Office Task Analysis, Behaviour and Information technology, V. 7, N. 3 1988, 275-93.
[Sebillote 91] Sebillote, S. Task Description according User's Objectives, Le Travail Humain, V. 54, N.3, 1991, 193-223.
[Seely-Brown and Duguid 94] Seely-Brown, J. ; Duguid, P. ; 'Borderline Issues : Social & Material Aspects of Design' In : Human-Computer Interaction, 9(1), 1994.
[Selic et al. 94] Selic, B. Et alli. Real Time Object Oriented Modeling, Wiley & Sons, 1994.
[Shackel 91] Shackel, B. Usability - Context, Framework, Definition, design and Evaluation. In: Shackel, B & Richardson, S. (eds), Human Factors for Informatics Usability, Cambridge University Press, 1991.
[Shlaer 88]Shlaer, S.; Mellor, S.J. *Object Oriented Systems Analysis*. Englewood Cliffs, Prentice-Hall, 1988.
[Shneiderman 87] Shneiderman, B. Designing the User Interface : Strategies for Effective Human-Computer Interaction, Addison-Wesley, Readings, 1992.
[Shneiderman 87]Shneiderman, B. *Design the User Interface : Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Reading Mass, 1987.
[Siddiqi 96] Siddiqi, J. ; Chandra Shekaran, M. Requirements Engineering : the Emerging Wisdom. IEEE Software, Mar 1996.
[Silvant 92] Silvant, E. et alli. *KADS & IEM : Un Pas de Plus Vers l'Intégration du Génie Cognitif dans la Conception des Systèmes d'Information* . In : Proc. of Fifth International Conference Software Engineering and Its Applications, Toulouse, pp. 131-44,1992.
[Simon 81] Simon, H. The Sciences of the Artificial, Cambridge , MIT Press, 1981.
[Siochi & Hartson 89] Siochi, A.; Hartson, R The UAN: A Task-Oriented Notation ofr User Interfaces. In: Proc. of Conference of Human Factors in Computing Systems CHI'89, ACM Press, 1989.
[SOHO 93] SOHO Science Operqtion Plan, Issue 1.1, June 1993
[Sommerville & Monk 94] Sommerville, I; Monk, S. *Supporting Informality in the Software Process*. In:[ESWPT94].
[Sommerville 92a]Sommerville, I. et alli. *Sociologists can be surprisingly Useful in Interactive Systems Design*. Research report CSCW[1[1992, Lancaster University.
[Sommerville 92b] Sommerville, I. et alli. *Integrating Ethnography into the Requirements Engineering Process*. Research Report CSCW/17/92, Computing Department, Lancaster University, 1992.
[Sommerville 95] Sommerville, I. Software Engineering, Addison-Wesley, Reading, 1995.
[Souza 1993]Souza, C. S. *The Semiotic Engineering of User Interface Languages*, International Journal of Man-Machine Studies, V. 39, pp 753-773, 1993.
[Sperandio 87] Sperandio, J.-C. *Software Ergonomics of Interface Design*. Behaviour and Information Technology, V. 6, N.3, pp 271-8, 1987.
[Sperber 87] Sperber, D.; Wilson, D. *La pertinence*, Paris, Editions de minuit, 1987
[Srivasta 77] Srivasta, S. et al. Job Satisfaction and Productivity. Comparative Administrations research Institute. Kent State University Press, Kent, Ohio, 1977.
[Storrs 95] Storrs, G. The Notion of Task un Human-Computer Interaction. In: Proc. of HCI 95- 10th Annual conference of the British Human-Computer interaction Group, University of Huddersfield, UK, August-september 1995.
[Suchman 87] Suchman, L. *Plans and Situated Actions - The Problem of Human-Machine Communication*, Cambridge University Press, 1987.
[Suchman 93] Suchman L, *Do categories have politics? The language/action perspective reconsidered*, Proceedings of ECSCW'93, 13-17 September, Milan, Italy, 1993

[Sutcliffe & Maiden 93] Sutcliffe, A. ; Maiden, N. Bridging the requirements Gap : Policies, Goals and Domains. Proc. of First International Symposium on Requirements Engineering RE'93, IEEE Computer Society Press, San Diego, USA, 1993.

[Sutcliffe 95] Sutcliffe, A. Human Computer Interface Design, Macmillan Press, London, 1995.

[Swartout 82]Swartout, W.; Balzer, R. *On the inevitable interwining of specification and implementation*. Communications of the ACM, 25(7), 438-45, 1982.

[Szekely 96] Retrospective and Challenges for Model-based Interface Development. In: [CADUI 96].

[Szekely et al. 92] Szekely, P. et alli. Facilitating the Explorations of Interface Design Alternatives: the HUMANOID Model of Interface Design. In Proc. of CHI'92 : Human Factors in Computing Systems, Monterrey, 1992, ACM Press.

[Szekely et al. 96] Szekely, P. et alli. Declarative Interface Models for User Interface Construction Tools: the Mastermind Approach, Engineering for Human Computer Interaction, 1996.

[Tanner 83]Tanner, P.;Buxton,W. *Some Issues in Future User Interface Management Systems (UIMS) Development.* IFIP Working Group 52 Workshop on User Interface management, Seeheim, Nov 1983.

[Tarby & Barthet 96] Tarby, J-C.; Barthet, M.-F. The DIANE+ Method. In:[CADUI 96].

[Tarby & Barthet 95] Tarby, J.C; Barthet, M.F. *Specifier les Taches et Concevoir les Objets*, Journées Internationales sur les Nouveautés en Génie Logiciel (J.I.N.G.L) - 13-15 décembre 1994 - CNIT, PARIS - Publié dans La Lettre ADA, n°78, pp.8-12, novembre 1995.

[Tarby 93]Tarby, J-C. *Le Dialogue Homme-Machine : des Spécifications à la Gestion Automatique*. Thèse de Doctorat, Université Toulouse 1, Toulouse, Septembre 1993.

[Thereau 92] Thereau J, *le Cours d'Action, Analyse Semiologique* (Action Course - Semiotic Analysis*)*, Peter Lang, 1992.(In French).

[Theron 88] Paul Theron, Guide Practique du Genie Logiciel , ed. Eyrolles, Paris , 1988.

[Thompson 94] Thompson, J. What You Need to Manage Requirements, IEEE Software, Mar 94.

[USDD 91] United States Department of Defense. *Software Technology Plan: Volume II - Plan of Action* (Technical Report Draft 5, 8/15/91), U.S. Department of Defense, August 1991.

[Van Lamsweerde et al. 95] Van Lamsweerde, A. ; Darimont, R. ; Massonet, P. Goal-Directed Elaboration of Requirements for a Meeting Scheduler : Problems and Lessons Learnt. Proc. of Second International Symposium on Rquirements Engineering RE'95, IEEE Computer Society Press, York, UK, March 28-30, 1995.

[Vanderdonckt 96] Vanderdonckt, J. Current Trends in Computer Aided Design of user Interfaces. In:[CADUI 96].

[Walsh 24 89] Walsh, P. *Analysis for Task Object Modelling (ATOM)*. .In: Diaper, D. (ed.) Task-Analysis for Human-Computer Interaction, Ellis Horwood, 1989.

[Wegner 95] Wegner, P. Interaction as a Basis for Empirical Computer Science. ACM Computing Surveys, V.27, N. 1, March 1995, 45-48.

[Wegner 97] Wegner, P. Why Interaction is Moer Powerful than Algorithms. CACM V. 4à, N. 5, May 1997.

[Weinberg 88] Weinberg, G.M. Rethinking Systems Analysis and Design. New York, Dorset House, 1988.

[Weiser 94]Weiser, M. *The World is not a Desktop*. ACM Interactions, V.1 N.1 Jan 94, pp 7-8.

[Wickens 92] Wickens, Christopher . Engineering Psychology and Human Performance, 2nd

ed, Harper Collins, NY, 1992.
[Williams 93]Williams, M; Begg, V. *Translation between Software Designers and Users*.. CACM V. 36 N. 4 Jun 1993, pp 102-103.
[Wilson & P. Johnson 96] Wilson, S.; Johnson, P. Bridging the Generation Gap: From Work Tasks to user Interface Design. In:[CADUI 96].
[Wilson 96] Wilson, S. Reflections on Model Based Design: Definitions and Challenges. In: [CADUI 96].
[Winograd 88] Winograd T, *A language action perspective on the design of cooperative work*, In: Computer Supported Cooperative Work: a book of readings, I. Greif[ed], Morgan Kaufman Publishers, 1988.
[Wirfs-Brock et al 90] Wirfs-Brock, R. Et alli Designing Object Oriented Software, Englewood Cliffs, NJ, Prentice-Hall, 1990.
[Wohlin et al. 94] Wohlin, C. Et alli. User Centred Software Engineering - A Comprehensive View of Software Development. Proc. of Nordic Seminar on Dependable Computing Systems, Denmark, August, 1994.
[Workshop 94]Workshop on Software Engineering and Human-Computer Interaction : Joint research Issues, 1994.
[Wroblewski 91]Wroblewski, D. *The Construction of Human-Computer Interfaces Considered as a Craft*. In :[Karat 91], pp 1-20.
[Yakemovic & Conklin 90] Yakemovic, K ; Conklin, J. 'Report on a development project use of an issue-based information system' IN : Proc. of CSCW' 90, October 1990, pp 105-118.
[Yourdon 89]Yourdon,E. *Modern Structured Analysis*. Yourdon Press, 1989.
[Zave 95] Zave, P. Classification of Research Efforts in Requirements Engineering. Used as classification scheme for the Second IEEE Intl. Symposium on Requirements Engineering 1995.
[Zultner 92a] Zultner, R.E.; The Denning Way : Total Qualit Management for Software, Proc. Of TQM for Software Conference, April 1992, Washington.
[Zultner 92b] Zultner, R.E. Quality Function Deployment (QFD) for Software: Structured Requirements Exploration. In: Schulmeyer, G. (ed) Total Quality management for Software, New york, 1992, pp 297-317.

Résumé

Notre travail traite de l'intégration des concepts de Génie Logiciel (GL) et de l'Interaction Homme-Machine (IHM). Le but de cette thèse est de proposer **TAREFA** (Task Analysis based Requirements Engineering FrAmework), une approche pour l'Ingénierie des Besoins des systèmes interactifs. TAREFA possède des caractéristiques spécifiques pour les systèmes interactifs, comme (a) une riche modélisation du contexte de réalisation des tâches de l'utilisateur ; (b) des solutions à des problèmes comme la traçabilité des besoins, la communication utilisateur-analyste et la modélisation des choix réalisés lors de la conception (*design rationale)* ; et (c) l'adoption de cas d'utilisation pour envisager différentes situations de l'utilisation du système.

Cette thèse est structurée en trois parties distinctes.

La *première partie* est consacrée à la présentation de l'état de l'art. Nous débutons tout d'abord au *chapitre I* par une revue des concepts des systèmes interactifs qui interviennent dans la compréhension de notre travail. Le *chapitre II* fait un tour d'horizon de l'ingénierie des besoins, selon une perspective Génie Logiciel. Ensuite, nous examinons dans quelle mésure ces concepts doivent être modifiés pour correspondre aux particularités des systèmes interactifs. Le *chapitre III* présente les approches les plus significatives utilisées pour l'ingénierie des besoins dans les domaines du GL et de l'IHM, et compare les travaux visant à l'intégration de ces deux domaines.

La *deuxième partie* est consacrée à la description de l'approche TAREFA. Le *chapitre IV* présente les fondements et une vue de l'ensemble de TAREFA, en particulier son organisation en deux macroactivités d'Analyse et Synthèse en prenant en compte l'entrelacement des activités systématiques et des activités coopératives; le *chapitre V* présente la démarche et les modèles associés à la macroactivité d'Analyse ; le *chapitre VI* présente la démarche et les modèles associés à la macroactivité de Synthèse; le *chapitre VII* présente le modèle de processus basé sur le concept de l'émergence, proposé pour les activités coopératives de TAREFA.

La troisième partie contient le *chapitre VIII*, qui présente une étude de cas de l'application de TAREFA : l'ingénierie des besoins pour un Outil d'Aide à la Planification Coopérative des Opérations du Satellite Multi-Instrument SOHO.

Mots-Clés

Ingénierie des Besoins, Systèmes Interactifs, Interaction Homme-Machine, Génie Logiciel, Modèles des Tâches, Cas d'Utilisation, Contexte, Analyse Orientée Objet.