MASTER EN SCIENCES INFORMATIQUES

Amélioration du support à la rédaction d'exigences, GenSpec

Harmel, Delphine; Cavillot, Guillaume

Award date:
2008

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique

Année Académique 2007-2008

Amélioration du support à la rédaction d'exigences, *GenSpec*

Delphine Harmel et Guillaume Cavillot

Mémoire présenté en vue de l'obtention
du grade de maître en informatique

Résumé

L'importance de l'ingénierie des exigences (IE) est de plus en plus reconnue et l'outil *GenSpec*, développé au sein de l'unité Conception-Automatismes (CA) d'Hydro-Québec (HQ) depuis 2001, est un outil de support à ce processus. Il permet notamment l'entrée des exigences dans une base de données, quelques vérifications de ces exigences et la génération de documents d'exigences. Notre travail consistait à améliorer l'outil *GenSpec* via (1) l'ajout de types d'exigences et d'énoncés par défaut ainsi qu'un module de gestion des types, et (2) la génération automatique de la Spécification des Exigences d'Interface (SEI) sur base de la Spécification des Exigences Système (SES). La première fonctionnalité permet de solutionner les problèmes d'uniformisation des exigences, de structuration de celles-ci, de complétude des documents d'exigences ou encore de redondance au sein de ceux-ci. La deuxième fonctionnalité permet de résoudre certains problèmes récurrents à la rédaction de documents d'exigences : le manque de cohérence entre SES et SEI, le manque de complétude de la SEI et le manque de traçabilité entre SES et SEI.

Ce mémoire contient une mise en contexte du domaine de l'IE, un résumé des différentes façons d'appréhender la phase d'IE, une analyse du processus de l'unité CA qui nous a permis d'identifier certains problèmes, la mise en avant de la problématique, un état de l'art des outils de support à l'IE et de la littérature, et enfin une description de notre mode de développement. Nous terminons avec la description des solutions proposées et une évaluation de celles-ci.

Mots clés - Ingénierie des Exigences, Outil de Support à l'Ingénierie des Exigences, Types d'Exigences, Génération de la SEI sur base de la SES.

Abstract

The importance of requirements engineering (RE) is increasingly recognized and the tool, *GenSpec*, developed by the Conception-Automatismes (CA) unit of Hydro-Québec (HQ) since 2001, helps to support this process. This tool supports requirements definition in a database, some verifications of these requirements and requirements documents generation. Our job was to improve the tool *GenSpec* through (1) the addition of new requirements types, their default statements and a module to manage these types, and (2) the automatic generation of the Interface Requirements Specification (IRS) based on the System Requirements Specification (SRS). The first feature helps solving problems to standardize requirements, to structure them, in the completeness of requirements documents or redundancy within them. The second feature solves some problems in the drafting of requirements documents : the lack of coherence between SRS and IRS, the lack of completeness of IRS and the lack of traceability between SRS and IRS.

This dissertation also contains an overview of the context in the field of RE, a summary of the different ways apprehending the phase of RE, an analysis of CA unit process which has allowed us to identify some problems, the setting before of the problem, a state of the art of different commercial RE-tools and literature, and finally a description of our way of development. We conclude with a description of proposed solutions and an assessment of them.

Keywords - Requirements Engineering, Requirements Engineering Tools, Requirements Types, Generation of the IRS based on the SRS.

Avant-propos

Ce mémoire est le résultat d'un stage de 4 mois au sein de l'unité Conception-Automatismes (CA) d'Hydro-Québec (HQ), Montréal, Canada. Nous y avons travaillé sous la tutelle de Monsieur René Bujold, ingénieur, et nous avons apporté des améliorations à l'OSIE, *GenSpec*. Cet outil a notamment reçu de l'office québécois de la langue française, le prix des mérites du français dans les technologies de l'information en mars 2007, dans la catégorie des applications logicielles de Grande organisation et Petite et Moyenne organisation. Nous avons ensuite entamé la rédaction de ce mémoire au sein des Facultés Universitaires Notre-Dame de la Paix, Namur, Belgique.

Pour cela, nous voudrions remercier :

Notre promoteur, Monsieur Patrick Heymans, pour le temps qu'il nous a accordé, sa supervision, son aide et ses conseils judicieux.

Monsieur René Bujold pour sa supervision lors de notre stage, ses conseils et son aide pour la rédaction de notre article.

Nous tenons à remercier Monsieur Daniel M. Berry pour son aide et sa sympathie.

Nous tenons également à remercier toute l'équipe CA d'HQ, dont notamment Monsieur Harold Ratté, Monsieur Marc Lacroix et Madame Claire Levesque pour leur sympathie et leur accueil lors de notre stage.

Table des matières

Table des figures

Liste des tableaux

Glossaire

Artefact	Tout document (règlement, graphique, procédure, ...) identifié au sein d'un processus.
Cas d'utilisation (Use Case)	Représentation du comportement affiché par le système sous certaines conditions de manière à satisfaire un objectif de l'un des acteurs.
Document d'exigences	Terme français employé à Hydro-Québec (HQ) pour Software Requirements Specification (SRS). Appelé également document de spécification ou document de spécification d'exigences.
Domaine d'application	Environnement dans lequel les actions du système à construire vont être observées et évaluées. Les exigences sont écrites relativement au domaine d'application.
Élicitation	Terme provenant de l'anglais ("Elicitation") faisant référence aux activités de collecte d'informations visant à identifier et clarifier le problème à l'origine du développement.
Extrant (Output)	Résultat d'une fonction, non nécessairement final. Par exemple, la fonction qui calcule l'aire d'un demi-cercle pourrait fournir en extrant l'aire du cercle, qu'elle calcule avant de la diviser par deux.
Fonctionnalité	Dans le cadre de ce mémoire, le travail à réaliser se décompose en deux fonctionnalités : - Ajout de types d'exigences, d'énoncés par défaut et d'un module permettant de les gérer ; - Génération automatique de la SEI à partir de la SES.
Ingénierie des exigences (IE)	Ensemble d'activités importantes qui s'intègrent tout au long d'un processus de développement logiciel. Elle a pour but de définir ce que le système doit faire (le "quoi") plutôt que de définir comment il doit le faire.
Intrant (Input)	Paramètre d'une fonction, élément utilisé par une fonction. Par exemple, les grandeurs des deux côtés d'un rectangle sont des intrants de la fonction qui calcule l'aire de ce rectangle.
Langage naturel	Langage utilisé pour la communication entre êtres humains. Par exemple la français ou l'anglais.

Modèle Représentation d'un canevas d'exigences (IEEE-Std-830, Volere, etc.) dans *GenSpec*, se présentant comme un ensemble d'exigences structuré respectant un canevas prédéfini. Dans *GenSpec*, toute création d'un nouveau projet *GenSpec* se fait à partir d'un modèle.

Outils CASE (Computer Aided Software Engineering) Ensemble de logiciels organisés autour d'une base de données de spécifications (repository). Leur objectif est de fournir une assistance au développement et à la maintenance de logiciels ou de certains composants d'un logiciel.

Outils de Support à l'Ingénierie des Exigences (OSIE) Outils logiciels fournissant une assistance automatisée durant le processus d'Ingénierie des Exigences (IE).

Parties prenantes Ensemble des acteurs associés à un projet. Il s'agit des clients, utilisateurs, concepteurs, développeurs, managers et toute autre personne concernée de près ou de loin par l'utilisation ou le développement du système. Correspondant au terme anglais "Stakeholders".

Spécification des exigences d'interfaces (SEI) Terme français employé à Hydro-Québec (HQ) pour Interface Requirements Specification (IRS). Une SEI a pour but principal de définir le format de tous les intrants et extrants définis dans la spécification des exigences système (SES) concernant l'interface que l'on veut décrire dans la SEI ainsi que les intrants et extrants spécifiques à celle-ci. Il peut s'agir d'une interface utilisateur, d'une interface entre deux logiciels, etc.

Spécification des exigences système (SES) Terme français employé à Hydro-Québec (HQ) pour System Requirements Specification (SRS). La SES doit décrire précisément les fonctionnalités du système à implémenter.

Système Un système ne se limite pas à un système informatique (software et hardware), mais inclut également les activités humaines. C'est par le support de celles-ci que le système est utile, on parle alors de "Software-Intensive System". Le terme système correspond, par concision, dans ce mémoire à "Software-Intensive System".

Types d'exigences Les types d'exigences permettent de structurer l'ensemble des exigences d'un document. Des exigences de même type ont des caractéristiques communes, notamment leur formulation, l'utilisation de termes spécifiques et leurs liens avec les autres exigences.

Acronymes

AGL	Atelier de Génie Logiciel
BD	Base de Données
CA	(unité) Conception-Automatismes
CASE	Computer Aided Software Engineering
CMMi	Capability Maturity Model Integration
ERP	Enterprise Resource Planning
ESIGETEL	Ecole Supérieure d'Ingénieurs en Informatique et GEnie des TE-Lécommunications
FUNDP	Facultés Universitaires Notre-Dame de la Paix
GL	Génie Logiciel
HQ	Hydro-Québec
IE	Ingénierie des Exigences
IRS	Interface Requirements Specification (traduction anglaise de Spécification d'Exigences d'Interface)
IV&V	Independant Verification and Validation (traduction anglaise de Vérification et Validation Indépendante)
MDA	Model Driven Architecture
MS	Méthodes Structurées
OMG	Object Management Group
OSIE	Outil de Support à l'Ingénierie des Exigences
PMBOK	Project Management Body of Knowledge

PMI	Project Management Institute
RE	Requirements Engineering (traduction anglaise d'Ingénierie des Exigences)
ROI	Return On Investment (traduction anglaise de Retour sur l'Investissement)
SEI	Spécification des Exigences d'Interface
SES	Spécification des Exigences Système
SI	Système d'Information
SRS	System Requirements Specification (traduction anglaise de Spécification des Exigences Système)
SWEBOK	SoftWare Engineering Body Of Knowledge
TE	Type d'Entité
UML	Unified Modelling Language
V&V	Vérification et Validation

Introduction

L'ingénierie des exigences (IE) est la phase initiale de tout développement logiciel. Elle met en oeuvre un ensemble de techniques permettant aux utilisateurs d'exprimer leurs besoins et à l'organisation de développement de les comprendre. Le déroulement de cette phase et la qualité des documents produits sont donc particulièrement critiques.

Si la phase d'IE est négligée, il en résulte une augmentation des coûts et des délais de réalisation du produit logiciel ainsi qu'une diminution de la qualité de ce dernier. En effet, une mauvaise compréhension d'un besoin émis par le client peut entraîner des perturbations au cours du développement. Ces perturbations occasionnent bien souvent une augmentation des délais et/ou des coûts. De plus, si la correction de certains besoins incompris entraine de trop grandes perturbations, celle-ci peut ne pas être réalisée lors du développement. Dans ce cas de figure, la qualité du produit final est altérée.

Afin de soutenir cette phase cruciale du développement d'un logiciel, des outils de support à l'ingénierie des exigences (OSIE) sont développés (Telelogic Doors, RaQuest, etc.). Le logiciel *GenSpec*, développé au sein de l'unité Conception-Automatismes (CA) d'Hydro-Québec (HQ) depuis 2001, fait partie de cette catégorie d'outils. Il est développé sur base de normes internationales (IEEE-Std-830 [Sof 98b] et ISO/IEC 12207 [Int 95]) et offre un large panel de fonctionnalités afin de produire des documents d'exigences de qualité.

Via l'utilisation de *Genspec*, les exigences sont rédigées en langage naturel, ce qui ne nécessite aucune formation ou connaissance particulière. Toutefois, il est à constater que plusieurs problèmes peuvent survenir lors de la rédaction des exigences avec *GenSpec*.

Un document d'exigences doit être le plus complet possible, aucun besoin du client ne doit être oublié et ce afin d'éviter tout problème en cours de développement. *GenSpec* doit offrir un modèle de document d'exigences et les fonctionnalités nécessaires à l'utilisateur lui permettant d'éviter au mieux ces oublis.

L'énoncé d'une exigence doit respecter des règles d'uniformité et être complet. Un exemple de règle d'uniformité pourrait être : "*L'énoncé d'une exigence devrait être écrit comme une phrase déclarative affirmative simple qui a seulement un verbe principal*". Ces règles permettent d'uniformiser les exigences en déterminant certains éléments de structuration ou de rédaction à respecter. Cela permet d'obtenir des énoncés complets, concis et compréhensibles aussi bien pour le client que pour les personnes amenées à consulter ces exigences. Il faut également éviter d'être redondant lors de la rédaction des exigences. Pourtant, aucun OSIE ne fournit d'énoncés modèles ou de règles permettant de formuler les exigences et *GenSpec* ne fournit pas non plus d'aide à ce niveau-là.

La structuration des exigences est également très importante. En effet, un regroupement logique des exigences au sein d'un document d'exigences facilite la lecture et la compréhension du client. Certains OSIE, tout comme *GenSpec*, offrent des possibilités de regrouper les exigences selon des types mais ceux-ci sont généralement trop peu nombreux. De plus, seuls quelques rares OSIE offrent une fonctionnalité permettant de créer et gérer des types d'exigences.

L'un des buts des OSIE est d'accélérer le processus d'IE. *GenSpec* offre déjà plusieurs fonctionnalités permettant cette accélération mais certaines lacunes sont à constater au niveau de la spécification des exigences d'interface (SEI). Une SEI a pour but principal de définir le format de tous les intrants et extrants définis dans la spécification des exigences système (SES) concernant l'interface que l'on veut décrire dans la SEI ainsi que les intrants et extrants spécifiques à celle-ci. Il peut s'agir d'une interface utilisateur, d'une interface entre deux logiciels, etc. Plusieurs éléments de celle-ci doivent être rédigés de manière manuelle, sans *GenSpec*, et il en résulte une perte de temps importante.

La liaison des exigences est un point important lors de la rédaction des documents d'exigences afin d'assurer une bonne traçabilité. *GenSpec* permet de créer différents types de liens entre les exigences afin d'en connaître l'origine, l'impact, etc. Il est également possible de créer des liens entre les exigences d'un même document ou vers un document externe au projet. Toutefois, les possibilités offertes ne sont pas assez développées et il est à constater que beaucoup d'OSIE ne fournissent que peu de fonctionnalités au niveau de la gestion des liaisons.

Après avoir récolté et analysé l'ensemble des besoins, nous avons conçu et implémenté des solutions aux problèmes de *GenSpec*.

Un ensemble de types par défaut ainsi que leurs énoncés associés ont été ajoutés à l'outil. De plus, un module permettant de gérer ceux-ci a été développé. Certaines fonctionnalités comme la génération automatique d'un énoncé lors de la sélection d'un type d'exigences ont été implémentées. Cet ensemble de solutions permet de résoudre des problèmes liés à la qualité des énoncés des exigences (complétude, uniformisation et non-redondance) ainsi que des problèmes de structuration au niveau du regroupement des exigences. Cela permet également, via la génération automatique d'un énoncé lors de la sélection d'un type d'exigences, de répondre en partie au problème d'accélération du processus d'IE. De plus, les types par défaut et la hiérarchie mise en place permettent à l'utilisateur de ne pas oublier certains types d'exigences et ainsi de fournir un document plus complet.

Une option de génération automatique de la première mouture de la SEI sur base de la SES a été développée. Afin que cette génération soit la plus complète possible, nous avons ajouté un module de gestion des références ainsi que d'autres fonctionnalités liées à ces dernières comme des vérifications automatiques. Ces nouvelles possibilités répondent en partie au problème d'accélération du processus d'IE ainsi qu'à certains problèmes relatifs aux liaisons. Les vérifications automatiques permettent également de répondre à des problèmes de complétude des documents d'exigences en signalant à l'utilisateur des oublis d'exigences dans la SEI.

Ce mémoire, subdivisé en 9 chapitres, présente l'ensemble de nos recherches ainsi que les diverses activités que nous avons réalisées afin d'élaborer ces solutions.

Le *Chapitre 1* décrit le contexte dans lequel le stage a été réalisé. En partant du plus général et en allant vers le plus spécifique, nous définissons le génie logiciel (GL), l'IE et les OSIE.

Le *Chapitre 2* présente les différentes activités de l'IE ainsi que diverses techniques permettant de les réaliser.

Le *Chapitre 3* fournit une description du logiciel *GenSpec*. Nous situons *GenSpec* par rapport aux différentes manières d'envisager l'IE décrites au chapitre précédent. Pour terminer, nous discutons également dans ce chapitre de l'utilisation qui en est faite ainsi que les évolutions envisagées pour cet outil.

Le *Chapitre 4* résume l'analyse que nous avons faite du processus de GL de l'unité CA d'HQ. On y retrouve notamment, une description de ce processus, une évaluation de type CMMi et les avantages et inconvénients de celui-ci.

Le *Chapitre 5* identifie les différents problèmes rencontrés par les utilisateurs de *GenSpec* auxquels nous devions apporter une solution.

Le *Chapitre 6* reprend un état de l'art de plusieurs OSIE commerciaux et de la littérature réalisé dans l'optique d'une recherche de solutions aux problèmes énoncés au chapitre 5.

Le *Chapitre 7* décrit notre mode de développement de manière globale et détaillée.

Le *Chapitre 8* illustre la solution que nous avons apportée à *GenSpec* via un ensemble de copies d'écran accompagnées de descriptions. L'intégration des nouvelles fonctionnalités est également discutée.

Le *Chapitre 9* évalue la solution que nous avons apportée à *GenSpec*, du point de vue de ses avantages et de ses limites, ainsi que le mode de développement que nous avons suivi.

Au terme de ces différents chapitres, nous tirons les conclusions et perspectives de notre travail.

Chapitre 1

Contexte

1.1 Génie logiciel

1.1.1 Définition

Si l'on se réfère à l'arrêté ministériel français du 30 décembre 1983 relatif à l'enrichissement du vocabulaire de l'informatique (Journal officiel du 19 février 1984), on appelle génie logiciel (GL) "*L'ensemble des activités de conception et de mise en oeuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi*".

Une autre définition pourrait être la suivante :

> "*Application pratique d'une approche systématique, disciplinée, quantifiable au développement, l'opération et la maintenance du logiciel*".

De manière synthétique, il s'agit donc de l'application de l'ingénierie au logiciel [IEE 90].

La norme Software Engineering Body of Knowledge (SWEBOK) du IEEE [IEE 04] définit les champs de connaissance du GL, tout comme le Project Management Body of Knowledge (PMBOK) du Project Management Institute (PMI) [Pro 04] le fait, pour la gestion de projet.

1.1.2 De la crise du logiciel à l'ingénierie

Cette sous-section est largement inspirée de [Hugues 02] et [Solnon 97].

Évolution de l'informatique

L'informatique n'est pas une discipline comme les autres. Celle-ci est encore très jeune et s'est organisée sur quelques dizaines d'années. Durant les années 45-58, la programmation était encore un domaine réservé à quelques ingénieurs mathématiciens qui programmaient en assembleur des programmes de calcul scientifique sur mesure. Ces programmes nécessitaient alors des machines volumineuses de puissance inférieure à celle d'une calculette actuelle.

Dans les années 58-75 l'informatique connait une période de développement considérable à la fois du côté scientifique (langage FORTRAN) et dans le domaine de la gestion (langage COBOL). Les premiers systèmes d'exploitation multi-utilisateurs voient le jour (IBM 360, etc.) durant cette période tout comme les premiers systèmes temps réel en informatique industrielle. Dans les entreprises, l'importance de la direction informatique n'a jamais été aussi grande et les investissements matériels sont énormes. Parmi les entreprises les plus avancées comme la NASA, par exemple, certaines perçoivent la crise du logiciel (cf section 1.1.2) qui va apparaître (1969).

Pendant les années 75-90, l'informatique va devenir une discipline à la portée des utilisateurs grâce aux innovations apportées par les bases de données (BD) relationnelles, la

micro informatique et l'avènement des réseaux. C'est également durant cette période que la nécessité de maîtriser le développement de logiciels apparait dans les esprits alors que le prix du hardware diminue et que le coût de l'informatique prend en compte non seulement les coûts de développement mais aussi les coûts de maintenance (corrective et évolutive). On passe des systèmes centralisés aux systèmes distribués. L'industrie du logiciel devient plus complexe alors qu'elle n'en est qu'à ses débuts. Avec la prise de conscience de la crise du logiciel et de l'importance stratégique des systèmes d'information (SI), la direction informatique cède progressivement la place à une direction de l'informatique et de l'organisation (début des années 80) dans les grandes entreprises.

Durant les années 90, l'omniprésence d'Internet ou d'Intranets dans les applications oblige à reconsidérer les architectures client-serveur au profit d'architectures 3-tiers prenant en compte des serveurs d'applications et non plus seulement des serveurs de données. Au niveau des applications, l'informatique de gestion se dote de progiciels de gestion de type Enterprise Resource Planning (ERP) qui intègrent les opérations les plus courantes et offrent à leurs clients des produits standardisés à 80% et paramétrables pour les 20% restant. Les outils d'aide à la décision font également leur apparition ainsi que le commerce électronique. Cette période voit les coûts et délais de développement devenir radicalement inférieurs en comparaison avec le passé.

A l'heure actuelle, le logiciel est présent partout et sa complexité augmente de façon exponentielle. Il faut savoir que la crise du logiciel apparue à l'aube des années 70 n'est pas encore totalement résolue même si de grands progrès ont été réalisés.

La crise du logiciel

Mise en évidence au début des années 70, la crise se caractérise par l'absence de maîtrise des projets, une augmentation des coûts et des délais ainsi que la mauvaise qualité des produits : les produits ne répondent pas aux besoins définis et des erreurs résiduelles persistent dans le produit final. On se rend compte, à l'époque, que les méthodes de développement logiciel en vigueur ne s'appliquaient pas aux grands projets.

La liste des logiciels défaillants produits après cette prise de conscience de l'existence d'une crise du logiciel est longue. La liste ci-dessous reprend plusieurs exemples marquants :

- Les sondes perdues vers Vénus (années 60) et vers Mars (1999),
- L'abandon du projet Advanced Logistic System (années 80),
- Les anti-missiles Patriotes (Guerre du Golfe 1991),
- Le système ambulancier londonien (1992),
- L'échec d'Ariane 5 (1996),
- Erreur dans le comptage des votes à Schaerbeek (2003),
- Blocage des réservations SNCF (2004).

Des enquêtes réalisées après plusieurs années mettent en avant certains chiffres révélant l'ampleur de cette crise. Parmi celles-ci, on retrouve notamment l'étude américaine du General Accounting Office réalisée en 1992 [Gen 92] et des études CHAOS réalisées par le Standish Group International depuis 1994 [Sta 04]. Il faut savoir que les études du Standish Group International ont été réalisées aux États-Unis sur un grand échantillon de projets. Plus précisement, 8380 projets de plus de 350 organisations ont été étudiés.

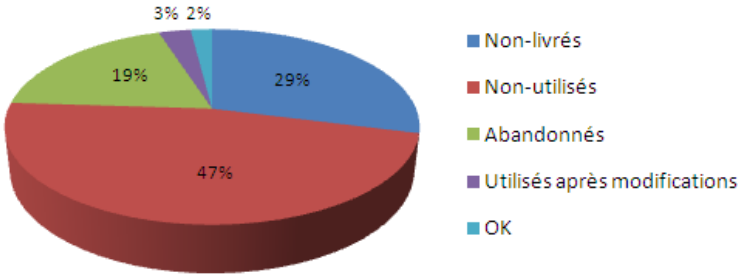

FIG. 1.1 - Résultats de l'étude du General Accounting Office (1992) [Gen 92]

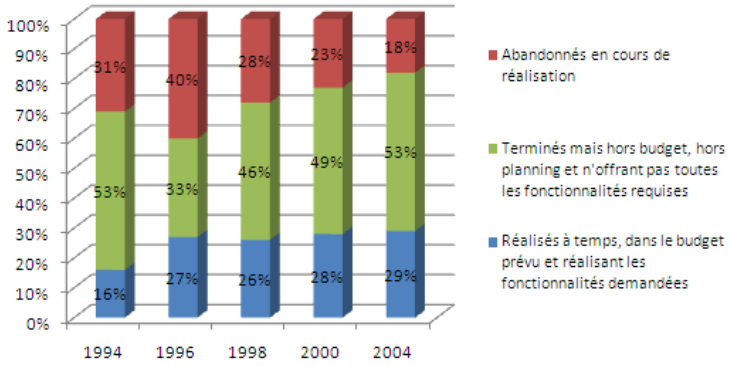

FIG. 1.2 - Résultats des études CHAOS (1994-2004) [Sta 04]

Dans les figures 1.1 et 1.2, nous constatons que des progrès ont été réalisés. Notamment, le nombre de projets réussis à temps dans le budget prévus et réalisant les fonctionnalités demandées est passé de 2% en 1992 à 29% en 2004. Toutefois, les chiffres nous prouvent également que la crise du logiciel n'est pas encore totalement résolue et qu'il faut continuer d'améliorer les méthodes de développement.

Les réponses à la crise

Cette sous-section est largement inspirée de [Hugues 02].

Afin de résorber cette crise, il faut tout d'abord acquérir la maîtrise du processus de développement. Pour cela, une organisation doit choisir et appliquer un modèle de processus définissant :

- Les phases du développement : définition des besoins, conception, codage, tests, etc.,
- Les produits intermédiaires : prototype, documentation, etc.,
- Les critères de changement de phase,
- Un cadre pour la gestion de projet.

Ces conditions sont nécessaires mais non-suffisantes. Il faut également obtenir le soutien de la direction, avoir du personnel compétent à disposition, etc.

Désormais, l'importance des phases d'IE et de conception est reconnue. La figure 1.3 de la page suivante permet de visualiser le modèle de développement qui fut créé au début des années 70, le développement en cascade [Royce 70].

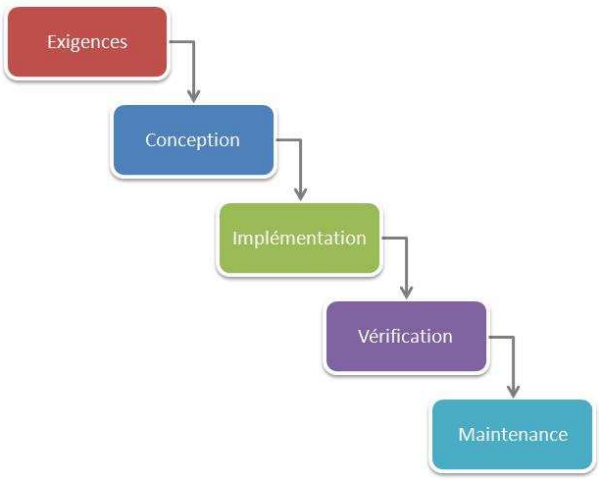

FIG. 1.3 - Modèle de développement en cascade [Royce 70]

Ce modèle a servi de base pour les autres modèles de développement qui ont suivi : incrémental, itératif, etc. A l'époque, l'IE était considérée comme l'étape initiale de tout développement logiciel tandis que les modèles développés ultérieurement considèrent l'IE comme un ensemble d'activités à réaliser tout au long du développement d'un logiciel, ce qui permet de gérer les projets pour lesquels les exigences peuvent changer en cours de projet. La section 1.2 permet d'approfondir le sujet de l'IE.

Dans les années 75-90, comme décrit par Laurent Audibert [Audibert 08], les méthodes structurées (MS) se sont développées ainsi que les métriques et outils permettant de mettre en oeuvre ces méthodes. Les MS utilisent intensivement les raffinements successifs pour produire des documents d'exigences dont l'essentiel est sous forme de notation graphique en diagrammes de flots de données. Le plus haut niveau représente l'ensemble du problème (sous forme d'activité, de données ou de processus, selon la méthode). Chaque niveau est ensuite décomposé en respectant les entrées/sorties du niveau supérieur. La décomposition se poursuit jusqu'à arriver à des composants maîtrisables. Les outils de mise en oeuvre de ces méthodes sont appelés Ateliers de Génie Logiciel (AGL) ou outils CASE (Computer Aided Software Engineering). Un AGL intègre des outils adaptés aux différentes phases du développement d'un logiciel et facilite la communication et la coordination entre ces différentes phases. Un AGL est basé sur des méthodes qui formalisent le processus logiciel et chacune des phases qui le composent. Les AGL apportent une réelle solution à certains problèmes du GL et contribuent à l'amélioration de la productivité et de la qualité du logiciel, notamment en faisant le suivi des différentes phases du processus logiciel et en offrant un cadre cohérent et uniforme de production.

L'apparition de nouveaux paradigmes contribue également à l'amélioration des processus de développement. En effet, durant les années 90, les MS s'avéraient inéfficaces pour traiter des programmes de 500.000 voire 5.000.000 de lignes. Or les développements sont de plus en plus complexes et de plus en plus longs. On se rend compte alors de la nécessité de réutiliser les connaissances et de les partager. Dans le domaine de la maintenance également, les MS n'ont pas fait leurs preuves. Le problème vient du fait que les MS sont soit orientées données, soit orientées actions, et que la modification d'un élément du logiciel impacte d'autres éléments de manière relativement imprévisible. Un nouveau paradigme, apparu en 1980 avec Smalltalk, devient très populaire dans les années 90, c'est le paradigme "objet" qui encapsule actions et données. Ceci facilite la maintenance car la localisation de l'erreur se fait plus facilement. Chaque objet est autonome. De plus, la définition de l'héritage et de la généricité permet la réutilisation d'objets déjà développés dans un autre contexte. Dans les années 2000, c'est la programmation à base de composants réutilisables qui est l'approche la plus en vue. Cette approche peut être vue comme un assemblage de briques logicielles prédéfinies conçues dans le but d'être réutilisées. La programmation à base de composants

est donc l'extension naturelle du paradigme "objet".

1.1.3 Tendances actuelles en GL

Cette sous-section s'inspire fortement de [Wikipédia 08].

- **Programmation orientée aspect :** la programmation orientée aspect est un paradigme de programmation qui permet de séparer les aspects ("Concern" en anglais) techniques des aspects métier dans une application. Ce type de programmation permet d'extraire les dépendances entre modules concernant des aspects techniques entrecroisés et de les gérer depuis l'extérieur de ces modules en les spécifiant dans des composants du système à développer qui sont développés à un autre niveau d'abstraction. Les chercheurs travaillent actuellement pour comprendre comment utiliser la programmation orientée aspect pour concevoir du code ayant un but général.

- **Méthode Agile :** Les méthodes agiles permettent de guider des projets de développement logiciel qui évoluent rapidement avec des attentes changeantes et des marchés où la rapidité de mise en place est critique. Ces méthodes se veulent plus pragmatiques que les méthodes traditionnelles. Elles visent la satisfaction réelle du besoin du client, et non d'un contrat établi préalablement.

- **Architecture dirigée par les modèles :** L'architecture dirigée par les modèles ou MDA (pour l'anglais Model Driven Architecture) a pour principe de base l'élaboration de modèles indépendants de plate-formes et la transformation de ceux-ci en modèles dépendants de plates-formes pour l'implémentation concrète du système. Les techniques employées sont donc principalement des techniques de modélisation et des techniques de transformation de modèles.

- **Lignes de produits logiciels :** l'approche "lignes de produits logiciels" est une manière systématique de produire des familles de systèmes logiciels, au lieu de créer une succession de produits complètement individuels. Elle regroupe donc les activités de développement d'un ensemble de logiciels appartenant à un domaine particulier. La réutilisation est le maître mot de cette approche, celle-ci devient réellement planifiée.

1.2 Ingénierie des exigences

1.2.1 Définition

Afin de définir l'IE, nous avons choisi les deux définitions ci-dessous. L'une étant celle proposée par le Centre d'Excellence en Technologie de l'Information et de la Communication (CETIC), l'autre étant le fruit de Steve Easterbrook. Ces deux définitions mettent en évidence la différence entre les modèles de développement considérant l'IE comme une étape initiale à tout développement, pour la première, et ceux considérant l'IE comme un ensemble d'activités réalisées tout au long d'un projet, pour la deuxième.

> "*L'IE est l'étape initiale de tout développement d'un logiciel. Elle met en oeuvre un ensemble de techniques permettant aux utilisateurs d'exprimer leurs besoins et à l'organisation de développement de les comprendre. Cette étape débouche sur un document contractuel pour le développement ultérieur : le cahier des charges. Le déroulement de cette phase et la qualité des documents produits sont donc particulièrement critiques.*" [Cen 08]

> "*L'IE est un ensemble d'activités concernées par l'identification et la communication des objectifs d'un système logiciel et le contexte dans lequel il sera utilisé. Ainsi, l'IE relie les besoins réels des utilisateurs, des clients et des autres parties*

prenantes affectées par le système logiciel avec les capacités et opportunités offertes par les technologies logicielles." [Easterbrook 05]

Cette deuxième définition met en évidence certaines caractéristiques du domaine de l'IE :

- L'IE représente un **ensemble d'activités** et non pas une phase ou une étape du processus de développement. Les exigences étant sujettes à des changements fréquents durant la mise au point d'un système.

- La **communication** est aussi importante que l'analyse en IE.

- La qualité d'un logiciel correspond à sa **capacité à atteindre les objectifs** qui ont été déterminés pour le projet. Il est impossible de juger la qualité tant que l'objectif est incompris ou mal défini.

- Les concepteurs doivent savoir où et comment le système sera utilisé, autrement dit, connaître son **contexte**.

- Les exigences représentent **ce qui est désiré**, d'une part, et **ce qui est possible**, d'autre part. L'IE agit tel un pont entre ces deux domaines.

- **L'ensemble des parties prenantes** doit être pris en compte. Il ne faut pas se limiter aux clients et aux utilisateurs du système, les besoins de tout intervenant touché de près ou de loin par le système doivent être intégrés afin d'augmenter les chances d'acceptation du produit.

1.2.2 Importance

Une mauvaise compréhension d'un besoin émis par le client peut entraîner certaines perturbations au cours du développement. En effet, certains besoins incompris ne peuvent être corrigés, faute de temps et/ou d'argent, en cours de développement ou après la livraison du produit. La qualité du produit en est donc diminuée. Si ce n'est la qualité du produit qui diminue, ce sont les coûts et/ou délais qui augmentent en raison de ces perturbations. En résumé, si les activités d'IE sont négligées, il en résulte une augmentation des coûts et des délais de réalisation du produit logiciel ainsi qu'une diminution de la qualité de ce dernier.

Les activités d'IE ne doivent donc pas être prises à la légère afin d'établir un compromis entre le coût et la qualité du produit en accord avec le client.

Nous reprenons, dans les sous-sections 1.2.2 et 1.2.2, les résultats de deux études qui ont permis de mettre en avant l'importance du domaine de l'IE :

- Étude CHAOS du Standish Group [Sta 95],
- Étude réalisée par James Martin [Martin 84].

Étude CHAOS

L'étude CHAOS réalisée par le Standish Group, déjà citée dans la section précédente, a permis également de mettre en évidence les facteurs de réussite des projets ainsi que les facteurs d'échec de ceux-ci. On trouvera ci-dessous les tableaux 1.1 et 1.2 reprenant les résultats de l'étude relatifs à ces facteurs.

Facteurs de réussite d'un projet	
Facteur de réussite	*% des réponses*
Participation des utilisateurs	15,9 %
Soutien de la direction	13,9 %
Énoncé clair des exigences	13 %
Bon planning	9,6%
Exigences réalistes	8,2%
...	...

TAB. 1.1 - Facteurs de réussite d'un projet [Sta 95]

Facteurs d'échec d'un projet	
Facteur d'échec	*% des réponses*
Exigences incomplètes	13,1 %
Manque de participation des utilisateurs	12,4 %
Manque de ressources	10,6 %
Attentes irréalistes	9,9%
Manque de soutien de la direction	9,3%
Exigences changeantes	8,7%
...	...

TAB. 1.2 - Facteurs d'échec d'un projet [Sta 95]

On constate que 32,1% des facteurs de réussite sont liés à l'IE : participation des utilisateurs, énoncé clair des exigences et exigences réalistes. Nous constatons également que 44,1% des facteurs d'échec sont liés à l'IE : exigences incomplètes, manque de participation des utilisateurs, attentes irréalistes et exigences changeantes.

Étude de James Martin

Une étude réalisée par James Martin en 1984 [Martin 84], qui était alors membre du comité de conseil scientifique en matière de logiciel du département de la défense aux USA, met en évidence la manière dont les bugs d'un logiciel sont distribués ainsi que la manière dont l'effort pour les résoudre est distribué. On trouvera ci-dessous les figures 1.4 et 1.5 reprenant les résultats relatifs à ces distributions.

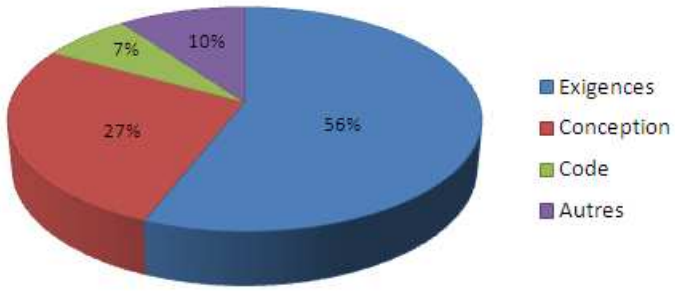

FIG. 1.4 - Distribution des bugs d'un logiciel [Martin 84]

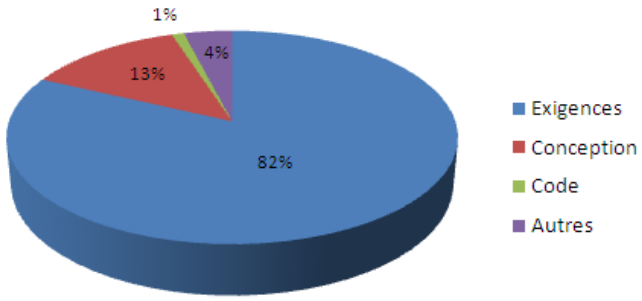

FIG. 1.5 - Distribution de l'effort pour résoudre les bugs d'un logiciel [Martin 84]

Grâce à ces graphiques, on constate que 56% des erreurs sont commises au niveau des exigences. On constate également que 82% de l'effort de correction des erreurs est lié à des erreurs commises au niveau des exigences.

1.2.3 Activités de l'ingénierie des exigences

Les activités liées au processus d'IE sont :

- Élicitation
- Prioritisation
- Documentation
- Validation/Vérification
- Négociation

Le chapitre 2 décrit plus en profondeur les différentes activités de l'IE et les diverses méthodes permettant de les réaliser.

1.2.4 Atouts et limites

Cette sous-section est également inspirée de [Wallonne 04].

Atouts

- L'IE est une activité permettant de prendre en compte de manière systématique et formalisée les besoins et les exigences des utilisateurs d'un système. Cette prise en compte des besoins et exigences des utilisateurs est indispensable pour la réussite du développement du système quel qu'il soit.

- L'IE permet de déceler, le plus rapidement possible, les erreurs de définition des exigences, qui autrement deviennent les erreurs les plus difficiles, les plus nombreuses et les plus coûteuses à corriger.

- L'IE doit contribuer à garantir un développement plus fluide, car elle diminue fortement le risque de devoir apporter des modifications fondamentales à l'application durant son développement.

Limites

- L'IE est un ensemble de techniques qui, selon la complexité et la taille du projet, peut s'avérer assez lourd et long à mettre en oeuvre et qui demande des compétences à la fois spécifiques et variées. Le recours à la sous-traitance peut donc souvent s'avérer nécessaire.

- Même si elle permet d'éviter bon nombre d'erreurs coûteuses, l'IE n'est ni une arme absolue, ni une assurance tout risque contre les erreurs de définition des exigences des utilisateurs, ni (surtout) contre les inévitables changements qui pourraient survenir.

- Vu les contraintes qu'elle engendre, l'IE est davantage adaptée au développement de gros systèmes qu'à la réalisation d'applications de petite taille et peu complexes même si cela dépend de la manière dont elle est pratiquée.

1.3 Outils de support à l'ingénierie des exigences

Cette sous-section est tirée principalement de [Firesmith 03].

1.3.1 Définition

Les OSIE sont des logiciels qui fournissent une assistance automatisée durant les activitées de l'IE.

> "*Seuls les outils supportant l'entièreté des activités de l'IE sont considérés comme des OSIE pour certains. Un logiciel de traitement de texte ne peut être considéré comme tel, d'après ces personnes, car il permet de supporter l'activité de documentation mais non l'entièreté du processus.*" [Matulevicius 05]

Toutefois, force est de constater qu'à l'heure actuelle, le nombre d'outils supportant l'entièreté des activités de l'IE est encore limité et nous considérerons comme OSIE, tout outil capable de supporter plus d'une activité de l'IE.

1.3.2 Évolution historique des OSIE mis à la disposition des utilisateurs

Le travail de l'ingénieur des exigences étant de plus en plus complexe, il est logique que les outils supportant celui-ci évoluent. Voici la manière dont les OSIE ont évolué :

- Outils de traitement de texte tels que MS Word,
- Feuilles de calcul telles que MS Excel,
- Outils de traitement de texte avec des feuilles de calcul intégrées,
- Outils possédant une BD et des possibilités de génération de rapport,
- Outils d'élicitation des exigences,
- Outils supportant plus d'activités de l'IE que l'unique élicitation des exigences,
- Outils qui font partie d'un environnement de développement intégré.

1.3.3 Génération automatique des documents d'exigences

La génération automatique des documents d'exigences est, sans doute, une des fonctionnalités les plus importantes d'un OSIE. En effet, celle-ci permet de produire des documents de manière standardisée et rapide. Cette génération va au-delà du fait d'utiliser des canevas internationaux tels que l'IEEE-Std-830 [Sof 98b]. Cela inclut la création de documents d'exigences pour la gestion des exigences et les autres activités du processus de développement utilisant ces documents. Cela devrait également inclure la génération de l'entièreté des documents d'exigences à publier et non pas la génération d'une partie de ceux-ci qui requiert, par la suite, des efforts manuels significatifs pour être complétée. Cela permet d'obtenir des documents d'exigences synchronisés avec les exigences présentes dans la BD et de générer ceux-ci de manière éléctronique, ce qui permet d'épargner de grandes quantités de papier. Cet avantage se remarque particulièrement lors de l'utilisation d'un cycle de développement itératif durant lequel les exigences changent de manière journalière.

1.3.4 Propriétés d'un OSIE

Au-delà du fait de posséder une BD permettant, entre autres, de générer automatiquement les documents d'exigences, les OSIE devraient avoir certaines propriétés critiques selon [Firesmith 03] :

- **Interface pour les utilisateurs :** l'une des meilleures manières pour énoncer des exigences est d'utiliser une interface graphique conviviale pour l'utilisateur qui permet d'entrer et maintenir des exigences individuelles, des tableaux, des modèles incluant du texte et des diagrammes, etc. Il est important de noter que cela est très différent de la technique consistant à développer un document d'exigences en utilisant un traitement de texte et puis d'encoder ce document dans la BD. L'interface ne devrait pas uniquement supporter l'introduction et la maintenance des exigences mais également la génération de documents d'exigences et de rapports incluant la création de "modèles", l'interrogation, etc.

- **Support de l'IE :** un OSIE devrait être complet dans le sens où il devrait supporter toutes les activités de l'IE incluant, l'élicitation, la modélisation, la vérification, etc. Il devrait supporter différentes approches d'analyse des exigences et donc plusieurs types de modèle tels que les cas d'utilisation, les diagrammes d'états, etc. Il devrait également supporter tous les types d'exigences : fonctionnelles, données, qualité, interface et contraintes. Les OSIE devraient offrir la traçabilité des exigences entre elles et vers les sources.

- **Support aux activités liées :** un OSIE devrait supporter les tâches d'autres activités si celles-ci sont liées au processus d'IE. Cela inclut la gestion de configuration, l'ingénierie de la qualité, etc. La gestion de configuration consiste à gérer la description technique d'un système ainsi qu'à gérer l'ensemble des modifications apportées au cours de l'évolution du système. L'ingénierie de la qualité, elle, concerne la qualité des processus de développement et la qualité des processus d'ingénierie des systèmes, notamment mis en oeuvre par le GL. De ce fait, un OSIE ne devrait pas être un outil indépendant mais un composant critique d'un environnement de développement intégré.

- **Développement en équipe :** le processus d'IE est mieux assuré si celui-ci est traîté par une équipe. Tous les membres d'une équipe chargée des exigences devraient être capables de travailler simultanément sur ces dernières. De la même manière, les autres membres de l'équipe de développement ont besoin d'avoir un accès simultané dans le but d'apprendre, évaluer et approuver.

- **Développement distribué :** de nos jours, il n'est pas inhabituel qu'une application soit développée par plusieurs équipes et organisations qui sont éloignées géographiquement. Un OSIE devrait offrir la possibilité de réaliser un document d'exigences de manière distribuée.

- **Sécurité :** les exigences de beaucoup d'applications contiennent des informations privées, des secrets, etc. Tout OSIE devrait supporter la sécurité des exigences incluant l'identification, l'authentification et l'autorisation d'effectuer des tâches spécifiques selon les rôles de ses utilisateurs. Il devrait également inclure la confidentialité, l'intégrité et la non-répudiation des exigences et de leurs mises à jour.

- **Partie d'un environnement de développement intégré :** un OSIE adéquat est plus qu'un simple outil de modélisation indépendant ou qu'une BD d'exigences. Il devrait être une partie d'un environnement de développement intégré.

- **Autres facteurs de qualité :** dans un sens, un OSIE est "une application comme une autre". De ce fait, les OSIE devraient remplir d'autres objectifs de qualité que la

sécurité et l'interopérabilité. Ainsi, lors de l'évaluation de ces outils, il faut examiner d'autres facteurs typiques de qualité : l'internationalisation, les performances, l'évolutivité, la simplicité d'utilisation, la convivialité, etc.

– **Réutilisation des exigences :** les OSIE devraient permettre l'incorporation d'exigences existantes dans la BD afin de ne pas toujours tout reprendre à zéro. Puisque ces exigences ont été probablement générées en utilisant des approches traditionnelles, les OSIE doivent être capable d'analyser d'anciens documents d'exigences, reconnaître les exigences potentielles et les incorporer de manière à ce quelles soient facilement révisables.

1.4 Résumé

Ce chapitre a permis de définir et comprendre le contexte dans lequel nous avons travaillé. Il couvre le GL, l'IE ainsi que les OSIE.

Le GL est une discipline relativement jeune. Il représente l'application de l'ingénierie au logiciel. Cette discipline est apparue suite à un besoin. Par le passé, les logiciels étant plus petits, ils pouvaient être développés de manière plus artisanale et une approche systématique et disciplinée n'était pas encore nécessaire à leur développement et à leur maintenance. Toutefois, l'informatique se développant à grande vitesse et les logiciels devenant de plus en plus complexes, une approche telle que décrite précédemment s'avère nécessaire. Les progrès réalisés en GL ont permis d'améliorer la qualité des projets et, de ce fait, la satisfaction des clients. Cependant, les diverses études réalisées dans le domaine montrent que des efforts sont encore à faire.

L'IE fait partie intégrante du processus de développement d'un logiciel. Cette discipline est également assez jeune, tout comme le GL. L'importance de l'IE n'est reconnue que depuis peu. Pourtant, comme les études du Standish Group International et de James Martin l'ont montré, une partie importante des facteurs de réussite et d'échec d'un projet sont étroitement liés à l'IE. Les erreurs commises au niveau des exigences sont plus coûteuses à corriger que celles réalisées lors de la conception ou l'implémentation, par exemple. L'amélioration du processus d'IE permet de réduire les coûts et délais d'un projet et d'augmenter la qualité de celui-ci. Toutefois, l'IE n'est pas le remède miracle aux problèmes que peuvent connaître les acteurs d'un projet.

Les OSIE fournissent une assistance aux personnes concernées par les activités de l'IE durant un projet de développement logiciel. Le nombre d'activités de l'IE qui sont couvertes par un OSIE peut différer mais ceux-ci sont en pleine évolution. L'atout principal d'un OSIE est de permettre une génération automatique des documents d'exigences. Les autres propriétés décrites à la section 1.3.4 ne sont pas encore toutes intégrées au sein des outils mais leur évolution permettra sans doute, dans le futur, d'offrir toutes ces possibilités.

Chapitre 2

Activités de l'IE

Ce chapitre décrit les différentes activités présentes au sein du processus d'IE. Pour chacune d'entre elles nous débutons par une decription pour ensuite discuter des problèmes et enfin des méthodes permettant d'appréhender l'activité.

2.1 Élicitation

2.1.1 Description

Hickey et Davis [Hickey 03] définissent l'activité d'élicitation comme :

> "*L'apprentissage, l'extraction, ou la découverte des besoins du client, des usagers et des autres acteurs concernés*".

Une alternative à cette définition est celle de Wiegers [Wiegers 99] :

> "*L'élicitation consiste à identifier les besoins à partir de diverses sources au moyen d'entrevues, d'ateliers, de workflows d'analyse de documents, ainsi que d'autres mécanismes*".

De ces définitions nous pouvons retirer que l'élicitation concerne l'apprentissage et la compréhension des besoins, des désirs et des attentes des utilisateurs et des clients, avec comme objectif de les communiquer à toutes les parties prenantes et en particulier aux développeurs du système.

Rzepka décompose notamment l'activité d'élicitation comme suit [Rzepka 89] :

1. **Identifier les parties concernées** qui sont les sources d'exigences. Les parties peuvent être un utilisateur final, un système d'interfaces ou des facteurs environnementaux.
2. **Rassembler la liste des besoins** pour chaque partie concernée. Cette liste est susceptible de contenir à l'origine des ambiguités, des incohérences, des exigences irréalisables et des exigences non testables, ainsi que d'être probablement incomplète.
3. **Documenter et affiner la liste des besoins** pour chaque partie concernée. La liste des besoins comprend toutes les activités importantes et les données. Durant l'élaboration de celle-ci, il est important de réitérer à plusieurs reprises les processus permettant d'éviter les incohérences. La liste est généralement de haut niveau, spécifique au domaine du problème et établie avec des termes propres à l'utilisateur.
4. **Intégrer la liste des besoins** à travers les diverses parties concernées et résoudre les conflits entre les points de vue. Une vérification constante est un élément important de ce processus. Les listes des besoins ou objectifs doivent être vérifiées afin de s'assurer de leur faisabilité.

5. **Déterminer les exigences non fonctionnelles**, telles que les performances et les problèmes de fiabilité et les intégrer au document d'exigences.

Ces activités sont communes à la plupart des définitions du processus d'élicitation d'exigences trouvées dans la littérature.

2.1.2 Problèmes

> "*Il existe de nombreux problèmes liés à l'IE, y compris les problèmes de définition de la portée du système, les problèmes dans la compréhension des buts des différentes communautés touchées par le développement du système donné, et les problèmes dus à la nature changeante des exigences. Ces problèmes peuvent conduire à des exigences de mauvaise qualité voire à l'annulation du développement du système, ou encore à un développement du système qui sera plus tard jugé insatisfaisant ou inacceptable, avec des coûts d'entretien élevés, ou ayant subit de fréquents changements.*" [Michael 92]

Comme explicités par [Michael 92], les problèmes concernant l'élicitation des exigences peuvent être regroupés en trois catégories :

- Les problèmes de **portée**, les exigences peuvent contenir trop ou trop peu d'information,
- Les problèmes de **compréhension**, au sein des groupes ainsi qu'entre les groupes tels que les utilisateurs et les développeurs,
- Les problèmes de **volatilité**, c'est-à-dire la nature changeante des exigences.

Problème de portée

Les techniques d'élicitation doivent permettrent d'établir les conditions limites du système ciblé. Ne pas prendre en compte les éléments liés au contexte peut mener à des exigences incomplètes, non vérifiables, inutiles et inutilisables. De la même façon, mettre l'accent sur les activités de conception et non sur les besoins réels des utilisateurs peut conduire à de mauvaises exigences.

L'élicitation des exigences nécessite une compréhension de l'organisation dans laquelle le système va être déployé, ainsi qu'une compréhension des objectifs du système dans son contexte organisationnel.

Il s'agit donc de prendre en compte :

- Les **contraintes environnementales** qui sont introduites parce que le système en cours de développement est rarement indépendant et interagit avec un système plus vaste [Leite 87].

- Le **contexte d'un projet** puisque celui-ci affecte aussi les exigences et le processus d'IE. Les facteurs pouvant influencer sont notamment le style de gestion du projet, la hiérarchie en place, les contraintes imposées aux personnes réalisant le projet, etc.

La compréhension de l'organisation, de l'environnement et du contexte du projet fournit ainsi un bon point de départ pour l'élicitation des exigences.

Problème de compréhension

> "*56% des erreurs dans les systèmes proviennent d'une mauvaise communication entre l'utilisateur et l'analyste dans la définition des besoins, et ces types d'erreurs sont les plus coûteuses à corriger et utilisent jusqu'à 82% de temps du*

personnel disponible." [Goodrich 90]

Les problèmes de compréhension au cours de l'élicitation peuvent conduire à des exigences ambigües, incomplètes, incohérentes, voire erronées car elles ne correspondent pas aux vrais besoins des intervenants.

On peut séparer les problèmes de compréhension en trois points :

- Les **parties concernées** par l'élicitation des exigences peuvent venir de milieux très différents, avoir des expériences différentes et des connaissances variées. Dès lors, il est difficile pour un analyste d'interpréter et d'intégrer les informations recueillies auprès des diverses parties.

- Le **langage utilisé** pour exprimer les exigences de chacune des parties peut être trop ou trop peu formel pour répondre aux besoins de chacun des groupes concernés, et ce, encore une fois, à cause de la diversité des groupes.

- La grande **quantité d'informations** recueillie au cours de l'élicitation nécessite d'être structurée. La compréhension de cette structure est dépendante des caractéristiques des groupes d'intervenants.

L'élicitation concerne divers participants tels que l'utilisateur, les analystes, les développeurs, etc. Il est donc important de ne pas recueillir de l'information uniquement auprès d'un seul groupe ou à un seul niveau, car elle serait susceptiblement biaisée par le niveau d'abstraction à partir duquel les gens conçoivent le problème, leurs connaissances, leurs préjugés personnels, leurs objectifs et leurs responsabilités. Par conséquent, comme l'exprime [Mittermeir 90] :

> "*La vraie image du problème à résoudre ne peut être obtenue qu'à partir d'une collecte d'information émanant de toutes les parties concernées.*"

Problème de volatilité

Les exigences changent. Durant la mise en place d'un système, les besoins de l'utilisateur peuvent évoluer. Les parties prenantes peuvent se rendre compte de besoins imprévus en raison des pressions environnementales ou organisationnelles. Si ces changements ne sont pas pris en compte, l'ensemble des exigences initial deviendra incomplet, incompatible avec la nouvelle situation et potentiellement inutilisable car l'information sera devenue obsolète.

2.1.3 Méthodes

Cette section inspirée de [Coulin 07], fournit un bref aperçu des méthodes les plus utilisées à l'heure actuelle. Nous les avons regroupées en quatre sections et expliquons pour chacune d'entre elles le principe et les techniques disponibles.

Méthodes traditionnelles

Ces techniques sont utilisées depuis le début de l'ingénierie du logiciel. Elles comportent notamment :

- Les **interviews**, qui consistent à interroger diverses parties prenantes, de manière structurée ou non, afin de collecter de l'information. La qualité des résultats provenant d'interviews dépend fortement des capacités de la personne les réalisant à interagir avec les participants, à ne pas se focaliser sur des questions techniques, etc.

- Les **questionnaires**, qui permettent de collecter rapidement des informations de diverses parties prenantes en utilisant une liste de questions (ouvertes ou fermées). Cependant, ils ne fournissent que peu d'information concernant le contexte et on observe

un manque d'interaction avec les participants.

- Les **lectures de fond**, qui permettent à l'analyste de comprendre l'organisation, peuvent fournir des exigences détaillées sur le système actuel. Cependant, ces documents contiennent bien souvent beaucoup d'informations inintéressantes.

Méthodes cognitives

Les techniques cognitives permettent d'éliciter les exigences en représentant et en structurant la connaissance des parties prenantes en fonction de leur vision du problème et de la solution. Ci-dessous quelques-unes de ces méthodes :

- L'**analyse des tâches** utilise une approche top-down où les tâches de haut niveau sont décomposées en sous-tâches voire en séquences détaillées. Les principaux objectifs de cette technique sont de construire une hiérarchie des tâches que les utilisateurs ou le système doivent réaliser et de déterminer les connaissances utilisées ou nécessaires à la réalisation de ces tâches. L'analyse des tâches fournit des informations sur les interactions entre les utilisateurs et le système concernant les tâches à réaliser, ainsi qu'une description contextuelle des activités. Dans la plupart des cas, un effort considérable est nécessaire pour accomplir une analyse des tâches approfondie. Il est donc important d'établir le niveau de détail nécessaire et savoir quand les composants des tâches doivent être étudiés plus en profondeur.

- Pour réaliser le **card sorting**, les parties concernées doivent répartir en groupe une série de cartes contenant le nom d'entités du domaine selon leur propre compréhension. Ils doivent ensuite expliquer oralement la façon dont ils ont réparti les cartes. Il est important pour l'efficacité du card sorting que toutes les entités soient incluses dans le processus. De plus, cela n'est possible que si le domaine est suffisamment compris à la fois par l'analyste et les participants. Si le domaine n'est pas bien établi, des travaux de groupe peuvent être utiles pour identifier ces entités. Cette technique est généralement plus utilisée pour le classement et la clarification des exigences que pour leur explicitation. L'inconvénient est qu'elle limite fortement l'information aux entités identifiées.

Méthodes participatives

Les techniques participatives nécessitent la présence de différentes parties prenantes afin qu'elles travaillent ensemble et génèrent des idées et des spécifications pour le système à réaliser. Ci-dessous, quelques exemples de telles techniques :

- Le principe du **brainstorming** est que des participants provenant de divers groupes de parties prenantes s'engagent dans une discussion informelle dont le but est de générer le maximum d'idées possibles sans se cibler sur l'une d'entre elles. Ensuite, il s'agit de trier les idées, de les évaluer et de raffiner les plus pertinentes d'entre elles.

- Les **Focus Groups** sont constitués, en général, d'au maximum une douzaine de participants et utilisent des matériaux (questionnaires, prototypes,...) pour engendrer la discussion entre ces participants. Les sessions ne durent jamais plus de deux heures et elles abordent généralement un sujet précis ou un problème particulier. L'animateur de la session a un rôle passif et le niveau de structure requis pour cette dernière est souple. Les focus groups permettent d'avoir des conversations plus naturelles que des interviews, par exemple. Cette technique permet l'élicitation d'idées importantes, d'opinions précises et de perceptions sur le système ciblé. Bien que relativement pratique et économique, les focus groups ne sont pas adaptés pour la recherche de solutions concernant des problèmes complexes ou des systèmes critiques.

Méthodes contextuelles

Les méthodes contextuelles ciblent la collecte d'information au contexte spécifique du système à réaliser. On y retrouve notamment :

- L'**observation des participants** qui permet d'obtenir une connaissance généralisée du domaine. Le problème principal de cette technique est que les participants ont tendance à modifier leur manière d'agir en sachant qu'ils sont observés et provoquent ainsi un certain biais dans les résultats. De plus, cette technique est assez coûteuse et exige un effort de la part de l'analyste qui doit interpréter et comprendre les actions réalisées par les participants.

2.2 Prioritisation des exigences

2.2.1 Description

Cette section fait de très larges emprunts à [Firesmith 04].

Les buts de la prioritisation des exigences sont de délimiter l'importance relative à chaque exigence, d'aider à éliminer les exigences secondaires via la négociation et le consensus, et enfin de planifier l'implémentation des exigences.

Pour certains auteurs dont [Sommerville 97], la prioritisation des exigences signifie la catégorisation de celles-ci :

- Les **exigences essentielles** qui doivent être incluses dans le système,
- Les **exigences utiles** qui diminuent l'efficacité du système ou la satisfaction du client si elles sont mises de côté,
- Les **exigences désirables** qui rendent le système plus attrayant pour certaines parties prenantes.

Cette catégorisation implique que certaines exigences sont hautement désirées ou importantes à avoir, ce qui peut sembler en contradiction avec la signification même d'exigence qui exprime leur caractère obligatoire ou exigé.

Les bénéfices de la prioritisation sont notamment :

- d'**augmenter la satisfaction** des commanditaires en mettant l'accent sur les exigences les plus importantes,
- de **diminuer les risques** d'abandon du projet en implémentant et en délivrant en premier les exigences les plus importantes.

2.2.2 Problèmes

Lors de l'activité de prioritisation des exigences, différents problèmes peuvent apparaître :

- Le **changement** continu des exigences. Comme nous avons pu le voir précédemment les exigences d'un projet évoluent constamment, certaines s'ajoutent tandis que d'autres se modifient ou sont supprimées. Nous avons vu également que l'IE devait être considérée comme un ensemble d'activités à réaliser tout au long du projet et l'activité de prioritisation ne fait pas exception à cette règle. En effet, si les besoins évoluent, il est important de réitérer le processus de prioritisation afin de rester fidèles aux attentes des clients. De plus, un contrôle des besoins en ressources doit être réalisé afin de s'assurer que le projet, dont les exigences ont changé, respectera les délais et les coûts fixés préalablement. Au besoin, une négociation avec le client devra être effectuée.

– Les **avis divergents** des parties prenantes concernant les priorités accordées aux exigences. Tout comme il est difficile de récolter des exigences provenant de différentes personnes impliquées dans un projet, il n'est pas aisé d'obtenir l'accord de ces dernières quant aux priorités accordées aux exigences. Cependant, il est important de trouver un compromis et de fixer ces priorités car le développement du projet est directement influencé par celles-ci.

– Les **priorités incompatibles** d'exigences interdépendantes. Certains besoins peuvent paraître secondaires et ne faire l'objet, a priori, que d'une faible priorité. Toutefois, ces besoins peuvent être étroitement liés, voir être nécessaires, à la réalisation d'autres exigences ayant une priorité plus élevée. Les interdépendances entre les différentes exigences doivent donc être prises en compte afin de réaliser une bonne prioritisation.

2.2.3 Méthodes

Il existe de nombreuses méthodes pour réaliser la prioritisation des exigences. Le lecteur pourra trouver plus d'informations sur ces méthodes dans [Mead 06]. Nous ne présentons ici que deux d'entre elles.

Il existe la méthode de l'**assignation numérique** pour laquelle tous les participants doivent attribuer une pondération aux exigences (par exemple, allant de 1 = exigence sans importance à 6 = exigence obligatoire). Ensuite, on réalise la moyenne de tous les chiffres pour chaque exigence et on obtient ainsi les exigences les plus importantes selon les différentes parties concernées.

Une autre méthode largement utilisée est la méthode **AHP** (Analytical Hierarchical Prioritization). Cette méthode utilise un système de matrice de comparaison en paires pour calculer la valeur et le coût relatif des différentes exigences les unes par rapport aux autres. Il s'agit ensuite de les représenter sous forme de points dans un diagramme coût-valeur comme celui ci-dessous (Fig. 2.1).

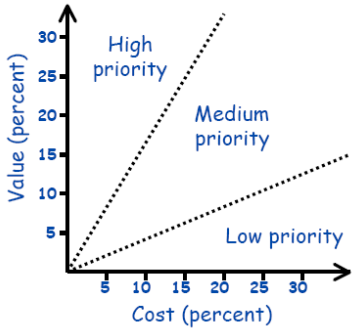

FIG. 2.1 - Exemple de diagramme coût-valeur [Easterbrook 04a]

2.3 Documentation des exigences

2.3.1 Description

Les informations reprises dans cette section proviennent des diverses sources suivantes : [Wallonne 04], [Grosjean 07], [Young 01], [Robertson 99] et [Wiegers 99].

La documentation des exigences consiste à formuler de manière synthétique les exigences dégagées lors de l'élicitation et à enregistrer, sous un ou plusieurs formats (textes, tableaux, modèles, schémas, etc.) et sur des supports persistants (BD, fichiers...), les informations

relatives aux exigences d'un SI afin de rendre celles-ci accessibles de façon durable aux différentes parties prenantes.

Une fois le document d'exigences réalisé, il doit être validé, vérifié, négocié et amendé pour pouvoir finalement être transmis aux développeurs. De plus, ce document peut faire l'objet de plusieurs itérations afin de le compléter ou encore de l'améliorer, par exemple.

Les intérêts d'une telle documentation peuvent varier mais ne sont pas pour autant mutuellement exclusifs. Elle peut constituer :

- Une **base de communication** pour expliquer le domaine d'application et le système qui doit être développé,
- Une **base contractuelle** pour s'assurer de l'accord des différentes parties prenantes,
- Une **base pour l'évaluation** du logiciel (support à l'activité de V&V) qui fournit assez d'information afin de s'assurer que le système délivré rencontre les exigences,
- Une **base de comparaison** pour le contrôle du changement des exigences,
- Une **base pour la rédaction** du manuel d'utilisation.

2.3.2 Problèmes

Les problèmes de la documentation des exigences sont nombreux. Nous présentons les principaux ci-dessous :

- Sélection de la **représentation**. Beaucoup de facteurs peuvent entrer en compte dans le choix du format de la documentation. En fonction des parties concernées, du type et de la maturité des exigences, des compétences des analystes, du caractère critique ou non de l'application, etc., un certain format peut être jugé plus ou moins adéquat.

- **Problèmes inhérents** à la documentation. Il existe une série de problèmes inhérents tels que l'ambiguïté, l'imprécision, la contradiction, l'incomplétude, etc. Il est important de prendre conscience de ces problèmes et d'essayer de les minimiser.

- **Inflation** des exigences. L'un des problèmes récurrents est l'inflation du nombre d'exigences. Celui-ci est fortement lié à la prioritisation et à l'élicitation des exigences. Le commanditaire a tendance à en demander de plus en plus et il est dès lors important de limiter cette pratique. Il faut prendre en considération les ressources disponibles et fixer les limites avec les parties concernées.

- La **traçabilité** des exigences. Il est essentiel de rendre les éléments documentés traçables les uns par rapport aux autres mais également par rapport à d'autres documents internes ou externes. On doit pouvoir facilement retrouver la source des exigences voire les liens entre chaque exigence.

- **Propriétés nécessaires** sur les exigences. Pour chaque exigence, les documents doivent fournir le statut de celle-ci (son niveau de priorité, si elle est approuvée ou non, etc.) ainsi que son contexte. On doit savoir, entre autres, le but de chaque exigence, pourquoi et par quelle partie prenante elle est demandée. Autant d'éléments sont essentiels afin de savoir à tout moment les origines et les enjeux de chaque exigence.

- **Gestion des changements et des versions**. Un document d'exigences peut varier à travers le temps, il est donc important de garder une trace des changements et des différentes versions de celui-ci. Si aucune gestion des changements n'est réalisée, il est impossible de retrouver l'origine ou la date des changements effectués. Dans certains cas, il est intéressant voire essentiel de pouvoir recouvrir rapidement les informations quant aux changements réalisés ou aux différentes versions délivrées.

- **Gestion des accès**. Plusieurs intervenants peuvent avoir accès aux documents d'exigences. Cependant, tous n'ont pas les mêmes droits concernant ces derniers. Par exemple, les développeurs n'ont qu'un droit de consultation alors que les analystes ont le droit d'écriture et de modification. Si une bonne gestion des accès n'est pas réalisée, certaines personnes n'ayant pas les droits pourraient, par exemple, modifier les documents et engendrer ainsi une certaine confusion quant aux exigences à réaliser.

- **Problèmes techniques**. Le problème d'interopérabilité entre les outils logiciels utilisés est un exemple de problèmes techniques que l'on peut rencontrer lors de l'activité de documentation. Tous les outils ne fournissent pas un moyen de réaliser des documents à la fois textuels et graphiques. Il s'agit alors d'utiliser différents logiciels pour réaliser un document complet comprenant à la fois une partie écrite et une partie modélisée. De plus, ces outils peuvent avoir une syntaxe et une sémantique propres ce qui peut engendrer un certains nombres de problèmes de cohérence entre les représentations.

2.3.3 Méthodes

Méthode par cas d'utilisation détaillés

Cette section se base fortement sur [Heymans 06].

La méthode par cas d'utilisation détaillés est une autre technique pour documenter les exigences potentielles d'un nouveau système ou les changements d'un logiciel. [Cockburn 01] fournit une définition :

> "*Un cas d'utilisation représente le comportement affiché par le système sous certaines conditions de manière à satisfaire un objectif de l'un des acteurs.*"

En d'autres termes, les cas d'utilisation décrivent, sous la forme d'actions et de réactions, le comportement d'un système du point de vue d'un acteur. Un cas d'utilisation est une manière spécifique d'utiliser un SI. Il s'agit de l'image d'une fonctionnalité d'un système déclenchée en réponse à la stimulation d'un acteur externe (au système) dans le but de satisfaire un objectif de ce dernier.

Comme le montre le schéma ci-dessous (Fig. 2.2), les cas d'utilisation partitionnent les exigences des utilisateurs.

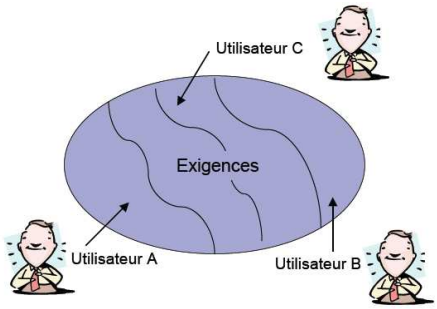

FIG. 2.2 - Partionnement des exigences des utilisateurs [Thiran 05]

Pour décrire un cas d'utilisation, on a besoin (au minimum) de deux notations :

- Les diagrammes de cas d'utilisation pour **décrire une "carte" des cas d'utilisation** permettant de représenter les fonctions et la communication (Fig. 2.3).

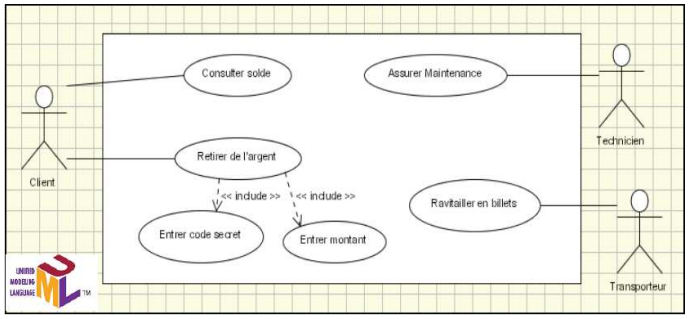

FIG. 2.3 - Diagramme de cas d'utilisation [Heymans 06]

– D'une autre notation pour **décrire le contenu des cas d'utilisation**, c'est-à-dire la spécification des fonctions et la dynamique. Typiquement, on utilise un texte structuré selon un canevas (Fig. 2.4). Cependant on peut également utiliser des diagrammes UML comme les diagrammes d'activités, d'états, de séquence,... ou bien encore des réseaux de Pétri.

Client	Système
1- Le client entre sa carte dans le système	
	2- Le système vérifie la carte du client et son droit à faire des opérations
3- Le client indique la somme qu'il veut retirer 4- Le client entre son code secret	
	5- Le système vérifie le code secret 6- Le système vérifie que le compte est bien approvisionné et qu'il reste assez d'argent dans l'appareil
8- Le client prend sa carte et son argent	7- Le système rend la carte et livre l'argent demandé
	9- Le système met à jour le compte du client

FIG. 2.4 - Description du contenu d'un cas d'utilisation [Heymans 06]

Chaque cas d'utilisation fournit un ou plusieurs scénarios qui expriment la manière dont le système doit intéragir avec l'utilisateur ou un autre système afin de réaliser un but spécifique. Lors de l'utilisation de cas d'utilisation, on évite généralement les termes techniques et on utilise un langage proche de l'utilisateur ou de l'expert du domaine. Les cas d'utilisation sont souvent rédigés conjointement avec les ingénieurs des exigences et les différentes parties prenantes.

La méthode des cas d'utilisation est relativement simple pour décrire le comportement d'un système ou d'un logiciel et elle permet de représenter simplement les différentes étapes que l'utilisateur doit suivre afin de réaliser une certaine tâche.

L'intérêt d'une telle méthode est donc de capturer, représenter et valider les fonctions d'un SI, de les spécifier et de donner un apercu de la dynamique et des interactions. Elle permet également de stimuler la découverte d'autres types d'informations comme la description du domaine d'application ou encore les exigences/spécifications non-fonctionnelles. Les cas d'utilisation peuvent constituer un input pour le planning du projet et l'estimation de son coût, la conception, la rédaction des manuels d'utilisation et l'aide en ligne ou encore les tests d'acceptation. Dans certains cas, le document d'exigences se résume à un ensemble de cas d'utilisation comme par exemple dans le cadre des méthodes Agile.

Avantages :

- *Compréhensibilité* :
  - écrit en langage naturel,
  - écrit dans un style narratif/scénarique,
  - facilement compréhensible par tous les intéressés.

- *Abstraction* :
  - fonctions, dynamique et communication/interactions entre les acteurs et la machine,
  - vue externe (black-box) et orientée utilisateur,
  - on ne s'encombre pas de détails,
  - un cas/flux à la fois,
  - vue d'un acteur à la fois.

- *Coût* : faible coût d'apprentissage,

- *Flexible*,
- *Réutilisable* pour des phases ultérieures (conception, tests,...),
- *Utilisation répandue*.

Inconvénients :

- *Coût* élevé si modélisation complète et cela à cause de la profusion des pêchés capitaux et l'explosion combinatoire du nombre de cas,

- *Précision* :
  - utilisation d'une notation informelle ce qui favorise les imprécisions,
  - opérations, conditions et autres concepts à définir séparément,
  - risque d'incomplétude des scénarios.

- *Prédictibilité* : le côté informel des cas d'utilisation peut mener à une utilisation automatique très limitée et à une exécution impossible,

- *Abstraction* : utilise parfois des concepts non définis,

- Ne constitue pas une spécification précise,
- Méthode peu adaptée aux projets critiques.

Méthode par énoncés d'exigences

La méthode par énoncés d'exigences permet de décrire les exigences en utilisant une structure claire et précise. Il existe une table des matières prédéfinie qui sert de base pour la rédaction des documents d'exigences. Pour chaque section, on dispose d'un descriptif de ce qu'elle doit contenir. Chaque exigence possède un numéro d'identification unique et une brève description. L'organisation du document d'exigences est un point essentiel car il s'agit de maximiser la lisibilité et la vérifiabilité. Le langage utilisé pour la rédaction des documents d'exigences est le langage naturel.

Il existe plusieurs façons de structurer un tel document :

- *Par modalités* : certains systèmes se comportent de manière très différente selon le mode de fonctionnement. Par exemple, un système de contrôle peut avoir des ensemble de fonctions différents selon le mode de fonctionnement normal ou d'urgence. Pour chaque mode, on énonce les exigences fonctionnelles, les exigences des interfaces externes et les exigences de performance.

- *Par classes d'utilisateurs* : certains systèmes prévoient différents ensembles de fonctions selon les différentes catégories d'utilisateurs. Par exemple, un système de contrôle d'un ascenceur fournit différentes options s'il s'agit d'un passager, d'un employé de la maintenance ou d'un pompier. On énonce donc ici les exigences fonctionnelles pour chaque classe d'utilisateurs.

- *Par objets* : les objets du monde réel sont des entités qui ont un homologue au sein du système. Par exemple, dans un système de suivi des patients, les objets sont des patients, des capteurs, des infirmières, des chambres, des médecins, des médicaments, etc. A chaque objet est associé un ensemble d'attributs et de fonctions qui peuvent être interprétés par cet objet. Ces fonctions sont aussi appelées des services, des méthodes ou des procédés. Notez que les ensembles d'objets peuvent partager des attributs et des services. Ceux-ci sont regroupés sous forme de classes. Pour chaque classe ou objet, on énonce les attributs directs ou hérités, les fonctions (services et méthodes directs ou hérités) et les messages (communications reçues ou envoyées).

- *Par caractéristiques du système* : une caractéristique est un service extérieur souhaité par le système qui peut exiger une séquence d'intrants pour obtenir le résultat souhaité. Par exemple, dans un système téléphonique, les caractéristiques comprennent un appel local, le renvoi d'appel et la conférence téléphonique. Chaque caractéristique est généralement décrite dans une séquence de stimulus-réponse. Pour chaque caractéristique, on énonce son but, la séquence de stimulus/réponse et les exigences fonctionnelles associées.

- *Par stimulus* : certains systèmes peuvent être mieux organisés en décrivant leurs fonctions en termes de stimuli. Par exemple, les fonctions d'un système d'atterrissage automatique d'un avion pourraient être organisées selon la perte de puissance, le cisaillement du vent, la vitesse excessive, etc. Pour chaque stimulus, on énonce les exigences fonctionnelles.

- *Par réponses* : certains systèmes peuvent être mieux organisés en décrivant toutes les fonctions nécessaires à la génération d'une réponse. Par exemple, les fonctions d'un système de gestion du personnel peuvent être organisées en sections qui correspondent à toutes les fonctions associées à la génération des salaires des patrons, toutes les fonctions associées à la création d'une liste actuelle des salariés, etc. Pour chaque réponse, on énonce les exigences fonctionnelles.

- *Par hiérarchie fonctionnelle* : lorsqu'aucun des schémas organisationnels ci-dessus ne convient, l'ensemble des fonctionnalités peut être organisé dans une hiérarchie de fonctions communes organisées selon les intrants, les produits communs ou l'accès commun aux données internes. Des diagrammes de flux de données et des dictionnaires de données peuvent être utilisés pour montrer les relations entre les fonctions et les données.

- *Mélange de structures.*

La norme IEEE-Std-830 [Sof 98b] est l'une des plus utilisée dans ce mode de représentation. Elle décrit les structures possibles, le contenu attendu et les qualités désirables d'un document d'exigences. La structure de cette norme est reprise à la figure 2.5 :

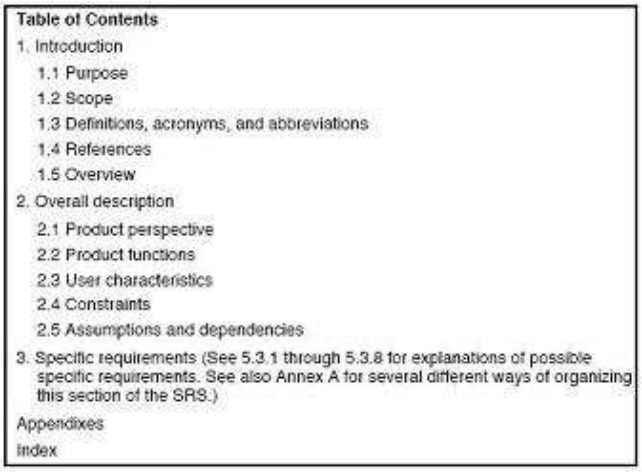

FIG. 2.5 - Structure du gabarit IEEE-Std-830 [Sof 98b]

Avantages :

- Facilité de lecture, de compréhension et de gestion en général des documents d'exigences,
- Bonne structuration du document d'exigences.

Inconvénients :

- Problèmes inhérents au langage naturel (ambiguïté, etc.).

Méthode par tableaux d'exigences

Le principe de cette technique est de décrire les exigences sous forme de tableaux. Chaque exigence dispose d'un numéro d'identification unique, d'une description et d'une série d'autres informations. Tout comme la technique précédente, les documents d'exigences sont rédigés en langage naturel.

Le gabarit Volere [Robertson 06] permet la découverte, l'organisation et la communication des exigences via la réalisation des documents d'exigences. Ce modèle classe tout d'abord les exigences selon les types d'exigences suivants :

- Les **Exigences fonctionnelles** qui sont les exigences fondamentales ou essentielles concernant le projet à réaliser.

- Les **Exigences non-fonctionnelles** qui concernent les propriétés que le futur système doit avoir comme, par exemple, la performance et la facilité d'utilisation.

- Les **Contraintes de projet** qui sont les restrictions imposées au produit en raison du budget et/ou du temps disponible pour réaliser le système.

- Les **Contraintes de conception** qui concernent les restrictions sur la manière dont le produit doit être réalisé.

- Les "**Project Drivers**" qui concernent, entre autres, le but du produit et les parties prenantes du projet.

- Les "**Project Issues**" qui défininissent les conditions sous lesquelles le projet sera réalisé.

Ci-dessous, vous trouverez la structure du gabarit Volere (Fig. 2.6) :

FIG. 2.6 - Structure du gabarit Volere [Robertson 06]

Ce gabarit fournit également un canevas pour rédiger chaque exigence. Ce canevas comprend les paramètres suivants :

- *Numéro de l'exigence* : identificateur unique de l'exigence,
- *Type de l'exigence* : type provenant du gabarit ci-dessus (Fig. 2.6),
- *Numéro(s) de l'évènement/cas d'utilisation* : liste des évènements ou cas d'utilisation qui ont besoin de l'exigence,
- *Description* : explication du but de l'exigence (en une seule phrase),
- *Logique* : justification de l'exigence,
- *Origine* : la personne qui a émis l'exigence,
- *Critère de mesure* : mesure de l'exigence de telle façon que l'on peut tester si la solution répond à l'exigence initiale,
- *Satisfaction du client* : degré de satisfaction des parties prenantes si l'exigence est implémentée avec succès (échelle de 1 = inintéressé à 5 = très intéressé),
- *Mécontentement du client* : mesure de mécontentement des parties prenantes si l'exigence ne fait pas partie du système final (échelle de 1 = sans importance à 5 = très mécontent),
- *Priorité* : notation de la valeur pour le client,
- *Conflits* : autres exigences qui ne peuvent être implémentées si cette exigence l'est,
- *Document de support* : pointe vers le document qui illustre ou explique cette exigence ;
- *Historique* : création, changement, suppression, etc.

Ci-dessous un exemple d'exigence fonctionnelle rédigée selon le canevas fournit par Volere (Fig. 2.7) :

FIG. 2.7 - Exemple de représentation d'une exigence fonctionnelle [Robertson 06]

Avantages :

- Facilité de lecture,
- Utilisation répandue, [Mascaro] fournit un ensemble de témoignages d'utilisateurs,
- Documentation simple.

Inconvénients :

- Problèmes inhérents au langage naturel (ambiguïté,...).

Méthode de modélisation UML des exigences

Cette section fait de larges emprunts à [Heymans 06].

> "*Comme les systèmes deviennent plus complexes, il devient de plus en plus difficile d'expliquer leur comportement de façon non ambigüe. L'une des raisons de cette ambiguïté est l'ambiguïté inhérente à toute langue naturelle. Un modèle fournit tout simplement un ensemble de constructions plus riches, de niveau plus élevé, et avec une sémantique plus précise qu'un langage naturel. L'utilisation de modèles permet de réduire l'ambiguïté, facilite la vérification de l'incomplétude et cela peut parfois améliorer la compréhension globale.*" [Chambers 06]

La modélisation des exigences permet de se concentrer sur la conception de haut niveau. Elle permet de penser et de communiquer clairement sur les questions les plus importantes, sans se soucier des détails de conception. Les documents d'exigences, bien souvent rédigés en langage naturel et illustrés de schémas ad-hoc, ont tendance à être ambigüs et incohérents.

Un modèle est d'ailleurs défini comme :

> "*Un système physique, mathématique ou logique représentant les structures essentielles d'une réalité et capable à son niveau d'en expliquer ou d'en reproduire dynamiquement le fonctionnement.*" [Heymans 06]

Comme l'exprime [Selic 03], la modélisation est le moyen traditionnel pour communiquer avec le client, réduire les risques et diriger le projet. Avant de réaliser quelque chose, les ingénieurs construisent des modèles et en tirent un certain apprentissage.

Un modèle est donc une représentation réduite, simplifiée et abstraite d'un aspect d'un système. Le but étant d'aider à comprendre et raisonner sur un problème (ou une solution), de communiquer des informations sur le problème et de diriger la conception de la solution. En ingénierie des SI, un modèle conceptuel est une représentation abstraite d'un (aspect d'un) SI au travers d'une notation standardisée.

Il y a quatre manières de décrire un aspect d'un SI :

- En langage naturel,
- En utilisant des notations ad-hoc,
- En utilisant des notations semi-formelles,
- En utilisant des notations formelles.

De plus, [Selic 03] décrit les qualités attendues d'un modèle :

- **Abstrait** : focalisation sur les aspects importants en oubliant les autres. Un modèle est toujours une simplification, une caricature, un point de vue. L'abstraction est le moyen le plus efficace de gérer la complexité croissante des SI.

- **Compréhensible** : la forme utilisée doit être intuitive pour l'audience. La fonction de l'expressivité est la capacité de transmettre une idée complexe au travers de peu d'information directe.

- **Précis** : fidélité avec laquelle le système modélisé est représenté. On ne peut en juger que si la forme, elle-même, ne présente pas d'ambiguïté. Mais attention, l'aspect formel n'induit pas forcément la précision ! On dit aussi que le modèle doit dire la vérité (correct), toute la vérité (complet) et rien que la vérité (pas de bruit ou de sur-spécification).

- **Prédictif** : le modèle peut être utilisé pour déduire des propriétés correctes et non triviales du système. Cela dépend fortement de la précision et de la forme.

- **Peu coûteux** : le modèle doit être beaucoup moins coûteux à construire et à étudier que le système réel, que celui-ci existe déjà ou non.

Un modèle peut être réalisé selon différentes vues :

- Les *vues classiques* :

  - Orientées données dont le contenu est informationnel et les propriétés statiques. Exemples : diagramme entité-association, les diagrammes de classes (sans opérations), etc.,
  - Orientées fonctions, c'est-à-dire selon les services, tâches, opérations. Il s'agit d'une découpe hiérarchique (fonctions → sous-fonctions → ...). Cela comprend les diagramme de cas d'utilisation, les pré-post conditions, VDM, Z, etc.,
  - Orientées dynamique (comportements) commme les diagrammes d'états, d'activités, de séquence, de collaboration et les réseaux de Pétri.

- Les *vues agrégées* :

  - Orientées objet, c'est-à-dire les données, les fonctions, etc. On les regroupe en entités réactives. On retrouve ici les diagrammes de classes (avec opérations),
  - Orientées agent où l'on regroupe les données, fonctions, etc. en entités actives,
  - Les notations à tout faire comme les notations logiques.

- Les *vues moins classiques* :

- Orientées objectifs qui ciblent le "pourquoi" et représentent les intentions, les argumentations et les alternatives. Exemples : i*, KAOS, etc.,
- Autres modèles qui représentent le contexte, la structure, la commmunication, etc.

L'Unified Modeling Langage (UML) est un ensemble de notations standardisées par l'Object Management Group (OMG) qui sont essentiellement graphiques et destinées à la modélisation des SI, à la fois du problème (domaine) et de la solution (logiciel). UML est actuellement le leader incontesté du marché de la modélisation, du moins en nombre d'adeptes et d'outils logiciels adaptés à celui-ci.

UML n'est pas une méthode mais un processus associé à un ensemble de notation et à des outils logiciels. A l'heure actuelle, UML propose une dizaine de diagrammes :

- Diagrammes de classes (class diagrams) et d'objets (objects diagrams),
- Diagrammes de paquetages (package diagrams),
- Diagrammes de composants (component diagrams),
- Diagrammes de déploiement (deployment diagrams),
- Diagrammes d'activités (activity diagrams),
- Diagrammes de cas d'utilisation (use case diagrams),
- Diagrammes d'états (state diagrams),
- Diagrammes de séquences (sequence diagrams),
- Diagrammes de collaborations (collaboration diagrams).

Ces différents diagrammes correspondent chacun à l'une des différentes vues et utilisent chacun une notation précédemment citée. Le choix de l'un ou l'autre diagramme dépend ensuite des concepteurs, des besoins de documentation, etc.

Nous allons maintenant voir les différents avantages et inconvénients de la modélisation UML des exigences.

Avantages :

- Le standard UML est reconnu et largement accepté par l'industrie,
- Polyvalent,
- Disponibilité de nombreux processus et outils logiciels,
- Volonté d'unification plait davantage que les particularités d'autres langages.

Inconvénients :

- Nécessite une connaissance minimum des techniques de modélisation,
- Unification par union plutôt que par intersection,
- Manque de formalité,
- Manque de coordination entre notations,
- Pas de notations pour certains aspects et peut-être trop pour d'autres,
- Aucun outil logiciel ne respecte la spécification à 100%.

2.4 Vérification et validation

2.4.1 Description

Comme expliqué par E. Tran [Tran 99], la vérification et la validation (V&V) sont essentielles dans le cycle de vie de tout système. Le développement de tout système n'est pas complet sans des tests rigoureux et la vérification que le produit est compatible avec les spécifications. La V&V a pris de l'importance, en particulier dans le logiciel, car la complexité des systèmes et des logiciels a augmenté et la planification de la V&V est nécessaire dès le début du cycle de développement.

Au cours des 20-30 dernières années, le développement de logiciels a évolué, passant de petites tâches impliquant un faible nombre de gens à des tâches gigantesques, impliquant un grand nombre de personnes. En raison de cette évoluation, la V&V a également fait l'objet de changements. Auparavant, la V&V d'un processus informel était effectuée par l'ingénieur logiciel lui-même. Toutefois, étant donné la croissance de la complexité des systèmes, il est devenu évident que la poursuite de ce type de tests se traduirait par des produits peu fiables. Il est devenu nécessaire d'examiner la V&V comme une activité distincte de l'ensemble du cycle de vie du développement logiciel. Le V&V d'aujourd'hui est sensiblement différente du passé car elle est pratiquée sur l'ensemble du cycle de vie du logiciel.

Les termes "Vérification" et "Validation" prennent des significations différentes selon les documents. La norme IEEE-Std-610 [IEE 90] les définit de la façon suivante :

- La **vérification** est

  "Le processus d'évaluation d'un système ou de tout autre document ou artefact produit pendant le développement qui satisfait les conditions imposées au début de cette phase."

- La **validation** est

  "Le processus d'évaluation d'un système ou de tout autre document ou artefact produit pendant ou à la fin du processus de développement afin de déterminer si on satisfait les exigences spécifiées."

La norme IEEE-Std-1012 [IEE 98] contient notamment plus d'informations concernant le processus de V&V.

Le schéma suivant illustre l'activité de V&V :

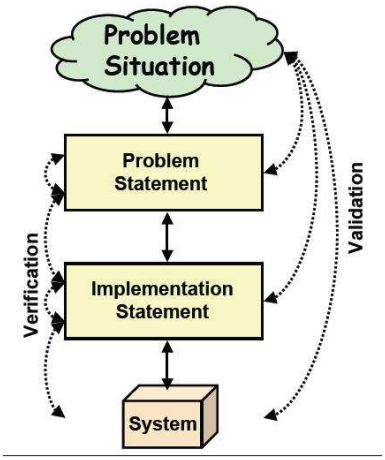

FIG. 2.8 - Processus de V&V [Easterbrook 04b]

D'un côté, nous avons le domaine d'application ("Problem Situation") qui est composé des propriétés du domaine et des exigences. De l'autre, nous avons le système qui est composé de la machine et du programme. Les spécifications jouent le rôle de pont entre les deux domaines, les propriétés du domaine sont des choses qui sont vraies à tout moment et les exigences sont ce que l'on doit rendre vrai. Les spécifications sont une description du comportement que le programme doit avoir pour satisfaire les exigences.

L'activité de validation consiste à s'assurer de la correspondance entre le problème de départ et les besoins du monde réel. Pour cela, il faut également s'assurer que l'on a bien dé-

couvert les exigences importantes et les propriétés du domaine d'application. Il s'agit donc de valider la correspondance entre les documents d'exigences par rapport aux exigences réelles.

Le but de la vérification est de s'assurer que la solution (le système) résoud bien le problème de départ. Pour cela, le système doit satisfaire les spécifications et les spécifications doivent correspondre au domaine d'application donné. L'activité de vérification consiste donc à vérifier (en interne) les documents d'exigences afin de s'assurer qu'ils ne contiennent pas, par exemple, d'incohérences.

De plus, comme l'explique [Storey 96], l'activité de V&V peut être effectuée par le même organisme que celui chargé de la conception, du développement et de l'implémentation, mais parfois elle est réalisée par un organisme indépendant. C'est ce qu'on appelle la **vérification et la validation indépendantes** (IV & V). Ces agences doivent normalement être accréditées par une organisation de plus haut niveau afin d'être sûr que leurs résultats sont fiables.

2.4.2 Problèmes

Comme l'exprime [Habra 07], deux problèmes majeurs sont à relever dans l'activité de V&V :

- La recherche d'un compromis entre les **avis divergents** des différents acteurs. Tout comme pour la prioritisation, il peut y avoir des avis divergents voire contradictoires entre les intervenants. Certains peuvent estimer que le produit satisfait entièrement les exigences initiales alors que d'autres non. L'activité de V&V doit prendre en compte l'opinion de toutes les parties prenantes afin de s'assurer que ces dernières sont toutes satisfaites par le produit réalisé.

- La recherche d'un compromis entre le **temps, les coûts et l'apport** de l'activité de V&V. Les bénéfices de la V&V doivent surpasser le temps et les coûts de sa mise en oeuvre. En effet, bien que la V&V doive s'assurer de la bonne qualité du système, celle-ci doit être réalisée tout en respectant les délais et les coûts qui lui sont initialement attribués.

2.4.3 Méthodes

Nous allons maintenant discuter de diverses méthodes de V&V. Cette sous-section se base essentiellement sur [Tran 99].

Revues, inspections et walkthroughs

Comme l'explique la NASA [NASA 08], les revues sont effectuées pendant et à la fin de chaque phase du cycle de vie afin de déterminer si les exigences établies sont satisfaites. Les revues consistent à présenter les documents réalisés à un panel de personnes . Les revues sont plus efficaces si le panel est composé de personnes qui n'ont pas été directement impliquées dans le développement du logiciel en cours de révision.

Une inspection ou walkthrough est un examen détaillé d'un produit étape par étape ou ligne de code par ligne de code. Le but de ces deux techniques est de trouver un maximum d'erreurs. Le groupe qui réalise l'inspection ou le walkthrough est composé de personnes venant du développement, des tests et de l'assurance qualité. La principale différence entre l'inspection et le walkthrough est que l'inspection suit un plan détaillé alors que la walkthrough suit un ordre aléatoire pour examiner le produit.

Prototypage

Cette sous-section se base sur les sources suivantes : [Davis 92], [Crinnion 91], [Haag] et [SPC 97].

Le prototypage est le fait de rapidement mettre en place un modèle de travail (un prototype) dans le but de tester les différents aspects de la conception, d'illustrer des idées ou des fonctionnalités et de recueillir plus vite les commentaires des utilisateurs. Souvent un ou plusieurs prototypes sont fabriqués dans un processus de développement itératif et incrémental où chaque prototype est influencé par la performance des modèles précédents. De cette manière, les problèmes ou les lacunes dans la conception peuvent être corrigés.

On distingue deux catégories de prototypage :

- Le **prototypage jetable** qui réfère à la création d'un modèle qui sera finalement jeté plutôt que de devenir une partie du logiciel final. Après une première collecte des exigences, un simple modèle de fonctionnement du système est conçu de façon à montrer aux utilisateurs à quoi ses exigences peuvent ressembler lorsqu'elles sont mise en oeuvre dans un système. Ce modèle permet donc de valider les exigences mais également de les éliciter.

- Le **prototypage évolutif** dont l'objectif est de construire un prototype très robuste de manière structurée et de constamment l'affiner. Lorsque le prototype est suffisamment affiné et répond aux besoins de fonctionnalité, de robustesse, d'industrialisation et d'autres objectifs de la conception, le produit est prêt pour la production.

2.5 Résumé

Tout au long de ce chapitre nous avons vu les différentes activités présentes au sein du processus d'IE : élicitation, prioritisation, documentation et V&V. L'IE étant encore une discipline relativement jeune, il existe de nombreuses définitions de ce processus. Certaines incluent d'autres activités que celles précédemment citées ou utilisent d'autres termes pour les nommer telles que "Spécification" ou "Analyse" au lieu de "Documentation". Nous avons donc choisi d'utiliser les termes qui nous semblaient les plus pertinents.

Pour chaque activité, nous avons tout d'abord fournit une brève description pour ensuite parler des problèmes liés à l'activité. Enfin, nous terminons chaque section par une présentation de diverses méthodes permettant de réaliser l'activité. Il existe de nombreuses techniques permettant de réaliser chaque activité du processus. Nous avons essayé de présenter les plus pertinentes et/ou les plus utilisées.

Au final, cette brève présentation des activités de l'IE a permis de démontrer la complexité de ce processus. Il n'existe aucune technique valable pour tous les projets et pour toutes les entreprises. Il s'agit de prendre en compte continuellement le contexte, l'organisation et l'environnement propres au projet à réaliser.

Le prochain chapitre vise à présenter l'OSIE amélioré. Tout d'abord, nous tenterons de classifier *GenSpec* parmis les diverses méthodes présentées tout au long de ce chapitre. Ensuite, nous expliquerons les classes d'utilisateurs d'un tel outil et les diverses fonctionnalités que ce dernier peut offrir.

Chapitre 3

GenSpec

Ce chapitre fournit une description du logiciel *GenSpec* ainsi qu'une présentation des utilisateurs de ce dernier. Nous discutons ensuite des améliorations envisagées pour l'outil et de l'utilisation qui en est faite (par qui, à quelle étape du processus, etc.).

3.1 Description de *GenSpec*

3.1.1 Classification de *GenSpec*

Dans le chapitre précédent nous avons vu différentes techniques pour réaliser les diverses activités du processus d'IE. Nous allons maintenant essayer de classifier *GenSpec* parmi celles-ci.

Tout d'abord, *GenSpec* est utilisé afin de réaliser les documents d'exigences. On peut donc dire que *GenSpec* est intéressant principalement pour la phase de **documentation**. Il faut cependant noter qu'il peut également être utilisé pour d'autres activités telles que la réalisation des tests via sa fonctionnalité d'écriture et de vérification de procédures de tests.

Parmi les différentes méthodes disponibles pour réaliser l'activité de documentation, la **méthode par énoncés exigences** est celle mise en place par *GenSpec*. En effet, l'outil se base sur deux normes, la IEEE-Std-830 [Sof 98b] et la IEEE-Std-12207 [Int 95], afin de réaliser ses documents d'exigences. Les documents générés à l'aide de l'outil respectent ces normes au niveau de la structuration des exigences.

Comme explicité dans la section 1.3, les OSIE offrent de nombreuses fonctionnalités intéressantes telles que, notamment, la gestion continue des exigences. Il est plus aisé de gérer les exigences, leurs interdépendances, leurs changements via l'utilisation de tels logiciels.

3.1.2 Contexte, objectifs et couverture du système actuel

L'unité CA d'HQ ayant perçu la nécessité de faciliter la gestion du processus d'IE, elle a développé, dès 2001, l'outil *GenSpec*. Il faut savoir que plusieurs outils existent déjà sur le marché mais l'unité a préféré concevoir son propre outil qui est disponible gratuitement aussi bien à l'intérieur qu'à l'extérieur de l'entreprise.

Les objectifs attendus à travers son utilisation sont les suivants :

- **La réduction des coûts :** automatisation, facilitation des modifications, facilitation de la réutilisation, etc.

- **La facilitation de la lecture :** présentation progressive (de la vue d'ensemble à la vue détaillée, format normalisé, etc.)

– **La réduction de la quantité d'erreurs :** contrôle et vérification automatique des exigences, etc.

– **Le respect des normes internationales :** IEC/IEEE/MIL/...

3.1.3 Fonctionnalités du système actuel

Nous allons tout d'abord présenter les fonctions principales de l'outil et nous terminerons par les fonctions secondaires. La nomination principale ou secondaire de ces fonctionnalités dépend de l'aspect optionnel de ces dernières. De plus, les fonctionnalités principales sont celles sur lesquelles l'accent a été mis, notamment en matière d'implémentation. Nous terminons cette section en classifiant les diverses fonctionnalités explicités selon les activités de l'IE qu'elles réalisent.

Ci-dessous, une illustration de l'aspect général de l'outil (Fig. 3.1).

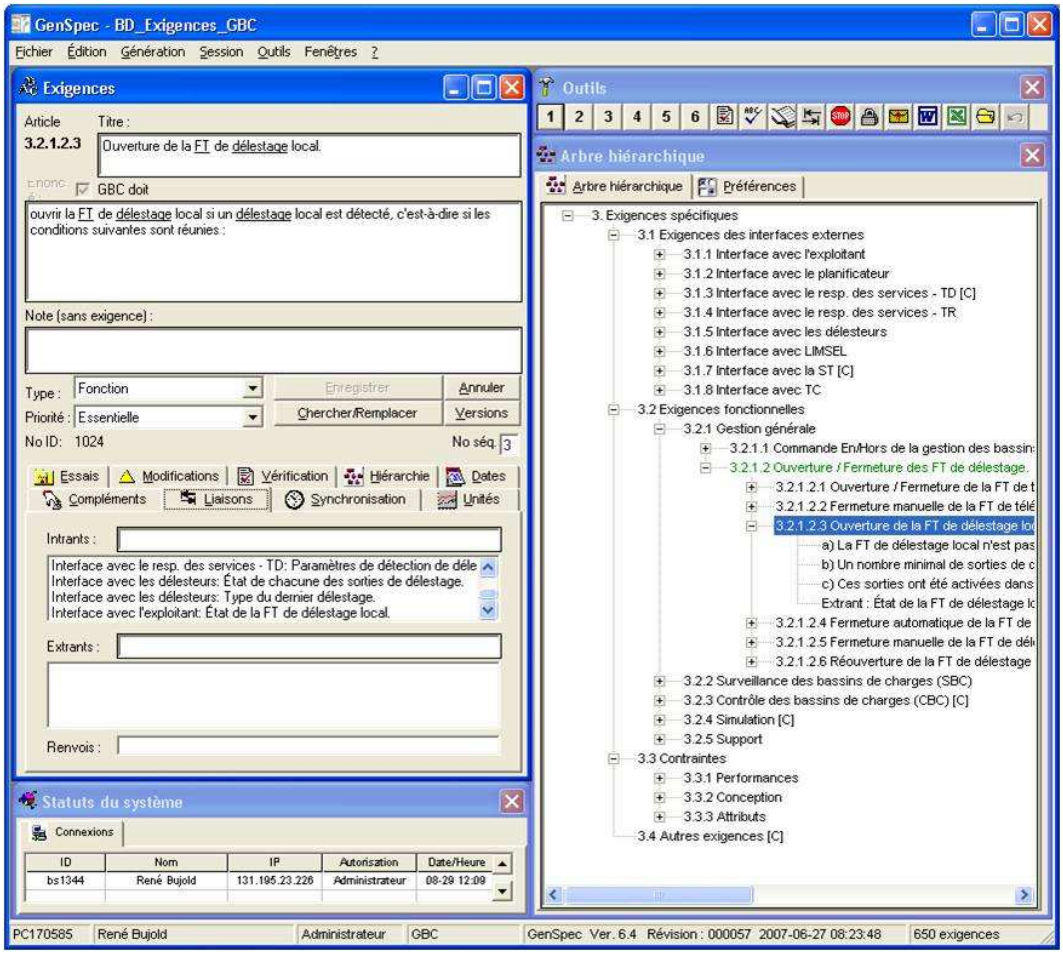

FIG. 3.1 - Aspect général de l'outil *GenSpec*

Fonctionnalités principales de *GenSpec*

La première fonction est la ***définition des exigences***. Il s'agit de créer un projet qui pourra ensuite contenir une liste d'exigences. Plusieurs utilisateurs peuvent entrer ou modifier des exigences en même temps dans un même projet. On peut également chercher et remplacer une exigence. Enfin, sur base de ces définitions, il est possible de générer des

documents Word ou Excel : spécification, arbre d'exigences, tableau reprenant les exigences selon leur numéro, etc.

La seconde fonctionnalité principale est la ***caractérisation des exigences*** qui permet d'entrer un numéro unique, l'identification de l'auteur, une priorité, un commentaire, un document joint ou encore une note pour chaque exigence (Fig. 3.2). De plus, *GenSpec* utilise un type d'exigences afin de caractériser celle-ci. L'utilisation de types d'exigences permet de structurer l'ensemble des exigences d'un document. Par exemple, les contraintes de sécurité et de disponibilité d'un produit, doivent être regroupées sous des titres différents, en l'occurence, sous sécurité et disponibilité. Et pour être précis, au sein des exigences de disponibilité, si nous avons des exigences liées aux pannes et certaines liées au démarrage, il faut également les placer sous des types d'exigences différents. Des exigences de même type ont des caractéristiques communes, notamment leur formulation, l'utilisation de termes spécifiques, leurs liens avec les autres exigences et leur position dans l'arbre des types.

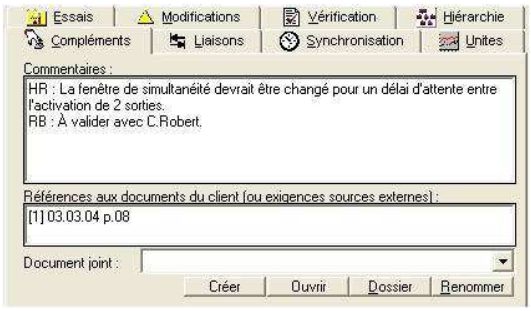

FIG. 3.2 - Caractérisation d'une exigence via l'onglet Compléments

Ensuite, nous avons la ***restructuration et la liaison des exigences*** (Fig. 3.3). Il est possible, avec *GenSpec*, de structurer et lier facilement les exigences par de simples "cliquer/glisser". La synchronisation de celles-ci et la navigation en sont facilitées. La liaison d'exigences est réalisée à l'aide du numéro attribué à chaque exigence. Par exemple, pour créer un renvoi d'une exigence vers une autre, il suffit de cliquer/glisser l'exigence souhaitée vers le champ "Renvois" de l'autre exigence. Le renvoi s'affiche sous la forme "Nom de l'exigence glissée". Dans les documents d'exigences, le numéro de l'exigence glissée est utilisé (Renvoi : [3.1.1.1]).

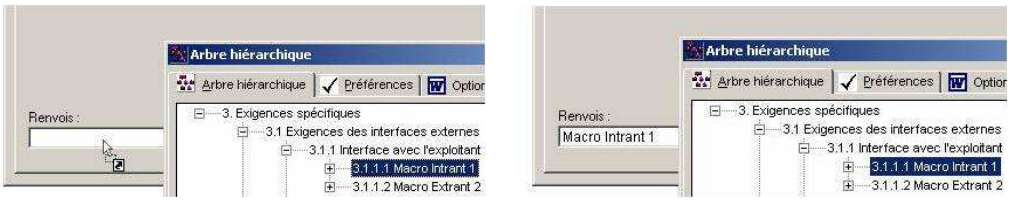

FIG. 3.3 - Structuration et liaison des exigences

La dernière fonction principale est ***l'évaluation de conformité*** de l'implémentation par rapport aux exigences. Grâce à elle, l'utilisateur peut créer des procédures d'évaluation de conformité pour chaque exigence et générer un document contenant les exigences, les procédures d'évaluation et les résultats de ces dernières (Fig. 3.4). Lorsque les tests ont été réalisés, il est ensuite possible d'insérer le résultat dans les documents d'exigences en utilisant les cases "Réussi" et "Échoué" ainsi qu'en indiquant la date de réalisation des tests.

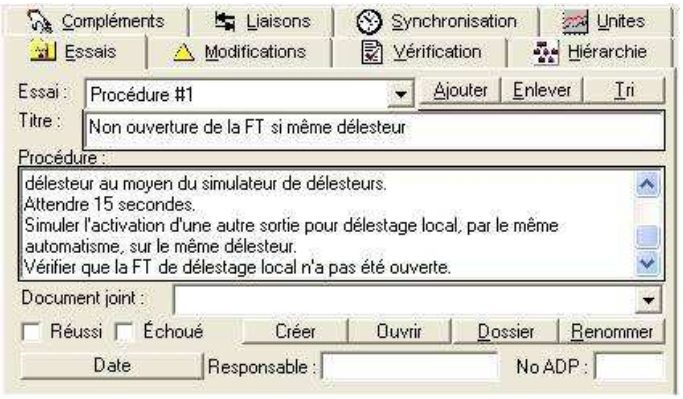

FIG. 3.4 - Évaluation de conformité à l'aide de procédures d'essais

Fonctionnalités secondaires de *GenSpec*

Le ***contrôle et l'analyse des exigences*** empêche d'introduire des incohérences de hiérarchie ou de liaison d'exigences et ce, à l'aide de règles de hiérarchie et de liaisons paramétrables. Par exemple, si l'on désire supprimer une exigence à laquelle d'autres renvoient, le vérificateur avertit l'utilisateur qu'il doit d'abord supprimer les renvois pour pouvoir supprimer l'exigence. De plus, *GenSpec* offre un vérificateur d'exigences, y compris un vérificateur d'orthographe et de grammaire mais également une série de vérifications comme le montre la figure 3.5.

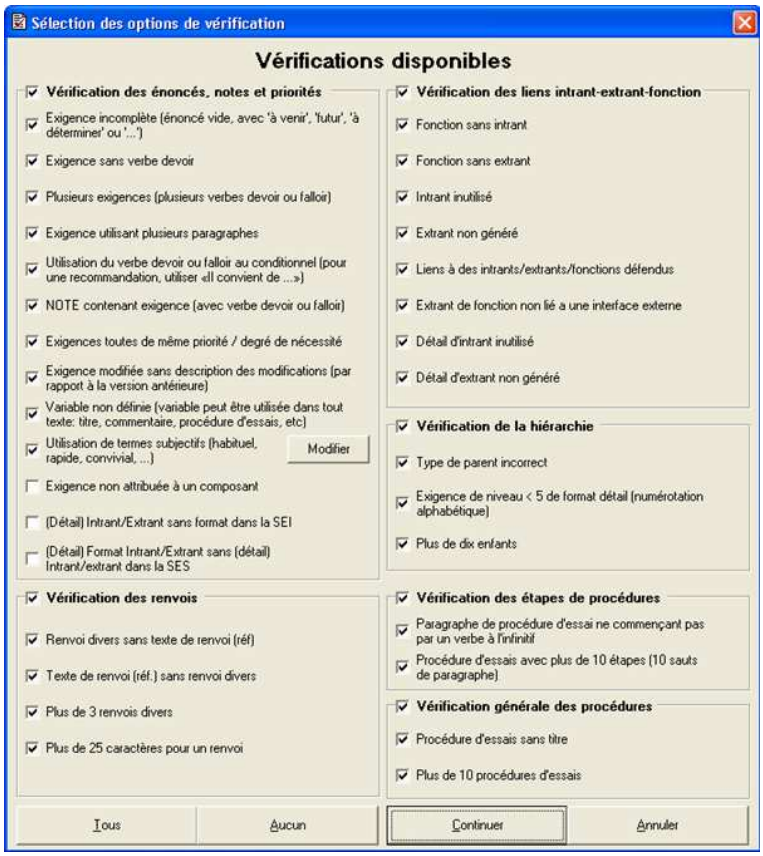

FIG. 3.5 - Fenêtre de vérification des exigences

Une deuxième fonctionnalité secondaire concerne la ***normalisation des exigences*** qui

permet de générer automatiquement des énoncés d'exigences selon le type d'exigences sélectionné. Elle permet également de définir, dans un glossaire, les "termes" utilisés et leurs synonymes, puis détecte automatiquement l'utilisation de ces synonymes et propose les termes à utiliser.

La configuration des documents d'exigences offre, quant à elle, une grande quantité d'options de formatage des documents générés, par exemple :

- Options de génération (Fig. 3.6) : niveaux à générer, etc.,
- Options des styles pour les documents à générer,
- Paramétrage dans la BD, des textes et styles générés.

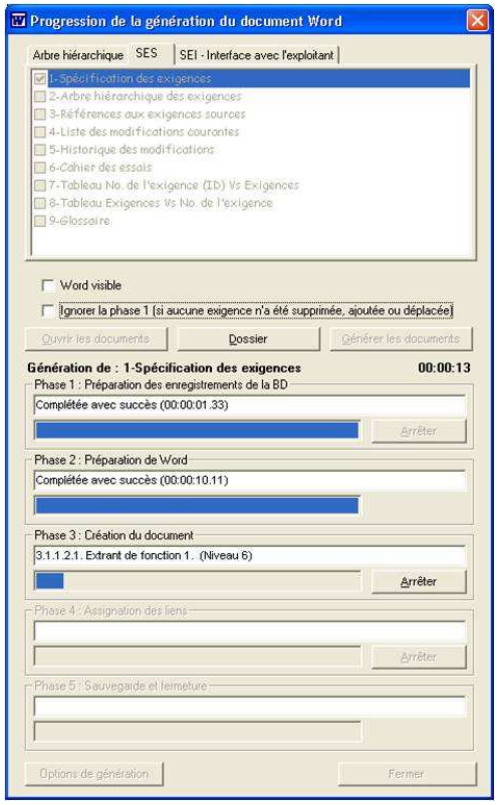

FIG. 3.6 - Option de génération de documents d'exigences

Enfin, elle permet notamment d'exclure des exigences et d'en inclure d'autres avec la mention "Non applicable".

La gestion de l'historique des exigences offre de nombreuses possibilités :

- Enregistrer une version formelle de l'ensemble des exigences,
- Entrer et visualiser la raison de modification d'une exigence par rapport à la version antérieure,
- Comparer la version actuelle avec une version antérieure,
- Ramener une ou toutes les exigences telles qu'elles étaient à une version antérieure,
- Générer un tableau reprenant l'historique des modifications des exigences,
- Enregistrer de simples copies de sécurité des exigences.

Enfin, la dernière fonctionnalité secondaire concerne ***l'attribution des exigences à des composants***. Cela permet d'attribuer chacune des exigences à chacun des composants du produit, aidant à la traçabilité des exigences. Ce mécanisme signale à l'utilisateur les

composants sans exigences attribuées, les exigences non attribuées à un composant et les exigences modifiées depuis l'attribution. Cette fonction permet également de générer un tableau présentant ces attributions.

Classification des fonctionnalités

Cette sous-section vise à classifier les fonctionnalités offertes par l'outil *GenSpec* selon les activités de l'IE (TAB. 3.1).

Fonctionnalités	Elicitation	Prioritisation	Documentation	V&V
Définition des exigences	-	-	X	-
Caractérisation des exigences	-	-	X	X
Restructuration et liaison des exigences	-	-	X	X
Evaluation de conformitié	-	-	-	X
Contrôle et analyse des exigences	-	-	-	X
Normalisation des exigences	-	-	X	-
Configuration des documents d'exigences	-	-	X	-
Gestion de l'historique des exigences	-	-	X	-
Attributions des exigences aux composants	-	-	-	-

TAB. 3.1 - Classification des fonctionnalités selon les activités de l'IE

Nous pouvons ressortir de ce tableau que la majorité des fonctionnalités permettent la documentation des exigences. Certaines d'entre elles permettent également l'activité de V&V. Enfin, nous observons que la dernière fonctionnalité ne fait pas partie du processus d'IE mais relève de la conception. Cette fonctionnalité permet de grouper les exigences devant être implémentées en même temps.

3.1.4 Caractéristiques techniques de l'outil

GenSpec est actuellement développé en langage Visual Basic 6.0. Afin d'implémenter les différents modules, l'éditeur Visual Basic est utilisé. Bien que l'on envisage la reprogrammation de l'outil en Java, cette modification n'est pas jugée prioritaire et n'est donc pas réellement planifiée à court ou long terme.

L'outil interagit avec une BD Microsoft Access. Ces BD recouvent l'entièreté des informations concernant un projet, des utilisateurs aux exigences, en passant par des informations purement techniques.

Afin de générer les divers documents (documents d'exigences, cahier des essais, etc.), *GenSpec* utilise l'éditeur Microsoft Word. Une modification est également envisagée à ce sujet. Il s'agirait d'utiliser un logiciel libre tel que OpenOffice pour l'affichage et l'édition des documents générés. Tout comme pour la reprogrammation, cette modification n'est cependant pas prioritaire.

Enfin, *GenSpec* tourne uniquement sur le système d'exploitation Microsoft Windows XP. Une amélioration possible et envisagée est de le faire fonctionner sur les autres versions de Microsoft telle que Microsoft Windows Vista, mais également sous Linux.

3.1.5 Profil des classes d'utilisateurs et structure organisationnelle

Cette section décrit les différents profils des classes d'utilisateurs en termes de responsabilités liées à chaque type d'usager (Fig. 3.7).

Le schéma de la structure organisationnelle ci-dessous permet de mieux percevoir la hiérarchie découlant de ces responsabilités.

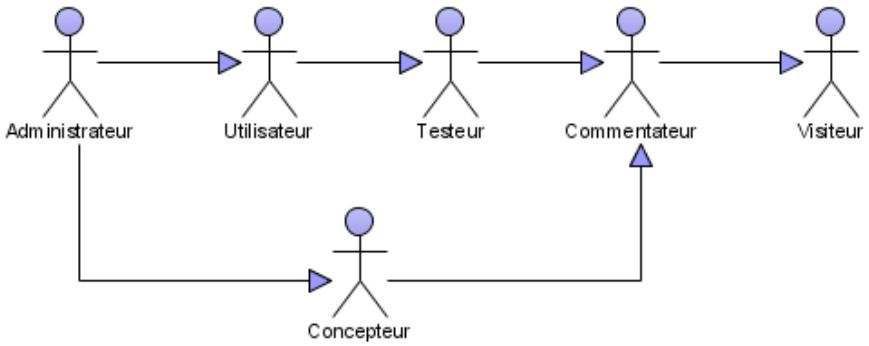

FIG. 3.7 - Structure organisationnelle

- **Administrateur** : l'administrateur se situe au sommet de la hiérarchie. L'administrateur est chargé de la gestion d'un projet dans sa totalité. Il est responsable du contrôle de la réalisation des spécifications et de mener à bien ce processus.

- **Utilisateur** : l'utilisateur, que l'on retrouve en-dessous de l'administrateur dans le schéma, est chargé d'établir les exigences et l'ensemble des informations nécessaires qui leurs sont associées : priorité, auteur, note,...

- **Testeur** : le testeur est chargé de vérifier si toutes les exigences ont bien été satisfaites. C'est pourquoi il est le seul, mis à part l'administrateur, à pouvoir écrire dans l'onglet Essais.

- **Commentateur** : le commentateur se retrouve à l'avant-dernier étage de la hiérarchie. Ceci étant du à la faible responsabilité qui lui est accordée : commenter l'ensemble des exigences, et ce s'il le souhaite.

- **Visiteur** : le visiteur se situe tout en bas de la hiérarchie. Celui-ci ne possède aucune responsabilité réelle vis-à-vis du processus. Il peut uniquement lire les documents d'exigences.

- **Concepteur** : le concepteur se situe entre l'administrateur et le commentateur. Sa responsabilité est de réaliser des regroupements d'exigences en composants et, s'il le souhaite, d'établir des commentaires sur les exigences.

3.1.6 Classes d'utilisateurs

Cette partie décrit les différentes classes d'utilisateurs de l'outil *GenSpec*. Auparavant, il n'en existait que 5 mais l'ajout de la classe Concepteur est apparue suite à l'apport de la fonctionnalité gérant l'aspect multi-documents par Olivier Pire et Nicolas Pirmez [Pire 07]. Contrairement à la section précédente, il s'agit ici de classer les différents usagers en termes de droits associés à l'utilisation des fonctionnalités de *GenSpec*.

Les différentes classes sont :

- **Administrateur** : un administrateur possède tous les droits sur un projet. Il peut accéder à toutes les fonctionnalités de *GenSpec* et peut également déterminer les droits d'accès des autres usagers.

- **Utilisateur** : un utilisateur possède tous les droits d'un administrateur mis à part les droits suivants :
  - Génération de versions officielles,
  - Modification des termes subjectifs (ex : convivial), couverts par les Options de vérification,
  - Ajout d'un usager,
  - Retrait d'un usager,
  - Modification du profil d'un autre usager,
  - Changement des droits d'accès.

- **Visiteur** : un visiteur possède uniquement les droits de lecture de tous les champs et de génération des documents.

- **Commentateur** : un commentateur a les mêmes droits qu'un Visiteur plus les droits d'écriture sur les champs Commentaires.

- **Testeur** : un testeur a les mêmes droits qu'un Commentateur plus les droits d'écriture sur les champs de l'onglet Essais.

- **Concepteur** : un concepteur peut uniquement ajouter des commentaires aux exigences, attribuer des exigences à des composants et fonctions et, enfin, générer dans un document le tableau d'attribution des exigences à des composants.

Toutes ces classes correspondent aux différents types d'intervenants par rapport à un projet. Sur le schéma décrivant la structure organisationnelle présenté à la sous-section suivante, il est possible de situer toutes ces classes.

3.2 Améliorations de *GenSpec*

GenSpec est en amélioration continue depuis 2001 et plusieurs projets d'amélioration sont menés actuellement ou prévus dans un futur proche.

3.2.1 Améliorations passées

A la fin de l'année 2006, deux étudiants des FUNDP, en stage chez HQ, ont implémentés les deux fonctionnalités suivantes [Pire 07] :

- Un glossaire, pour le traitement de l'ambiguïté.

- Une gestion multi-documents d'exigences, pour rassembler l'ensemble des exigences, mécanisme d'allocation d'exigences à une hiérarchie de composants et une liste de fonctions fournissant un support à l'activité de conception.

Ces deux fonctionnalités ont ensuite fait l'objet d'améliorations qui ont été réalisées fin août 2007 par des étudiants de l'université ESIGETEL et de l'université La Sorbonne, également en stage chez HQ. Il s'agissait d'apporter quelques améliorations au glossaire préalablement implémenté, en particulier, y ajouter les possibilités suivantes :

- Importer un glossaire à partir d'un autre projet *GenSpec*, d'une autre BD ou d'un document Word,
- Générer un document "Glossaire" sous forme de tableau,
- Formater les termes du glossaire apparaissant dans les documents d'exigences dans un style particulier (en relief par défaut),
- Ajouter automatiquement les définitions du glossaire dans les documents d'exigences selon trois options :

- *Option 1* : génération automatique des définitions à la première apparition du terme, intégrant automatiquement le principe de présentation graduelle des idées dans les documents d'exigences, facilitant davantage la documentation et, au besoin, la restructuration des exigences,
- *Option 2* : génération automatique des définitions à toutes les apparitions du terme, pour faciliter au besoin la lecture,
- *Option 3* : aucune génération de définition, pour raccourcir le document d'exigences. Après plusieurs lectures, les définitions deviennent souvent inutiles.

La deuxième partie des améliorations concerne l'attribution des exigences aux composants. Cela consiste en :

- L'amélioration de la convivialité de la fenêtre d'attribution des exigences aux composants,
- L'ajout de la possibilité, à partir d'une exigence sélectionnée, d'identifier quels sont les composants auxquels elle a été attribués,
- La vérification de l'absence de termes subjectifs dans les exigences (ex : convivial, habituel ou rapide) et ajout de cette option dans les options de vérifications automatiques,
- Limitation de cette vérification aux titres et énoncés d'exigences.

GenSpec a ensuite fait l'objet de nos améliorations qui seront présentées dans le chapitre 8.

3.2.2 Améliorations futures

Dans cette section nous pouvons différencier les améliorations à court terme, à long terme et celles qui ne sont actuellement qu'envisagées.

Amélioration à court terme

L'une des améliorations à court terme, réalisée par l'École Polytechnique de Montréal jusqu'à la fin avril 2008, consiste en la création de vidéos d'aide à l'utilisation de *GenSpec*.

L'Institut National Polytechnique de Grenoble est, quant à lui, chargé de l'intégration d'une fonctionnalité de gestion des risques liés aux exigences.

La dernière amélioration à court terme qui sera réalisée par l'Université de Sherbrooke au Canada d'ici la fin avril 2008, consiste en la génération automatique d'une première ébauche des spécifications de composants, à partir des :

- Exigences système,
- Attributions des exigences système aux composants,
- Liaisons Intrants-Fonctions-Extrants des exigences système.

Il s'agira également d'ajouter une vérification automatique plus poussée des liens Intrants-Fonctions-Extrants :

- Vérification de la présence des noms des intrants/extrants liés aux exigences fonctionnelles dans les énoncés de ces exigences fonctionnelles,
- Vérification de la présence des liens aux intrants/extrants pour les intrants/extrants mentionnés dans les énoncés des exigences fonctionnelles.

Enfin, une amélioration générale de la convivialité de l'outil est envisagée, via :

- La génération de la totalité du document d'exigences, pas seulement la partie exigences mais également les parties "Introduction", "Description générale", etc.

- La facilitation de l'insertion des documents générés par *GenSpec* dans un autre document,
- La facilitation de l'ouverture et de la création d'un nouveau projet.

Améliorations à long terme

Les améliorations prévues avant la fin 2010 sont la gestion des versions par exigence, en plus de la gestion des versions par spécification, la révision générale des interfaces externes de l'outil pour le rendre, en outre, utilisable par la minorité anglophone et, enfin, une refonte complète de l'outil, possiblement via l'utilisation de logiciels libres.

Autres améliorations envisagées

Une autre solution envisagée par le créateur de *GenSpec*, monsieur René Bujold, est l'ajout d'une fonction d'importation de documents. Cela permettrait tout d'abord d'importer le document d'expression des besoins du client et de faciliter la liaison des exigences avec les besoins. Ensuite, on pourrait également importer les documents d'exigences faits avec Word. Cette dernière modification augmenterait un peu plus les possibilités de l'outil.

3.3 Utilisation de *GenSpec*

L'outil *GenSpec* est principalement utilisé au sein de l'unité qui l'a conçu, l'unité CA d'HQ. Une douzaine de personnes utilise quotidiennement le logiciel afin de réaliser les documents d'exigences qui leur sont assignés.

Il est difficile de recenser le nombre exact d'utilisateurs en dehors d'HQ puisqu'aucun suivi n'est réalisé. Cependant, une centaine de demandes de l'outil ont été traitées. De plus, l'École Polytechnique de Montréal utilise *GenSpec* comme outil pédagogique.

Le fait que l'outil ne soit pas encore traduit en anglais limite la distribution de ce dernier à la francophonie. On retrouve les utilisateurs de *GenSpec* essentiellement en France, en Belgique, au Maroc et bien évidemment au Canada.

La direction de l'unité CA tente cependant de généraliser l'utilisation de l'outil *GenSpec*. Dans ce but, le Comité Qualité ainsi que le Comité de pilotage *GenSpec*, chargé entre autres de promouvoir l'outil, essayent de faire en sorte que tous les ingénieurs de l'unité utilisent systèmatiquement le logiciel pour l'activité d'IE. Une fois que cela sera réalisé et contrôlé, ils espèrent étendre l'utilisation de *GenSpec* aux autres unités d'HQ.

En ce qui concerne l'activité où est utilisé *GenSpec*, il semblerait que la majorité des utilisateurs l'utilise principalement pour l'activité de documentation.

3.4 Résumé

Ce chapitre a permis de décrire l'outil *GenSpec* et les fonctionnalités qu'il offre. Nous avons pu remarquer que, via l'utilisation de ce logiciel, certains avantages pouvaient être dégagés : la réduction des coûts, la facilitation de la lecture des documents d'exigences, la réduction de la quantité d'erreurs au sein de ces derniers et le respect de normes internationales.

Nous avons également décrit les différentes classes d'utilisateurs de *GenSpec* : administrateur, utilisateur, visiteur, commentateur, testeur et concepteur. Ces différentes classes d'utilisateurs permettent la gestion des droits d'accès aux fonctionnalités.

Par la suite, nous avons présenté les améliorations réalisées ou envisagées pour ce logiciel qui fait l'objet d'une évolution continue depuis 2001.

Finalement, nous avons décrit l'utilisation qui est faite de l'outil par les clients ainsi que les activités où le logiciel est utilisé. Aucun suivi n'étant réalisé, il n'est pas aisé de connaître le nombre réel d'utilisateurs de *GenSpec* mais celui-ci connait une popularité croissante.

Le chapitre suivant contient l'analyse du processus de GL de l'unité CA. Cette analyse nous a, entre autres, permis d'identifier les problèmes auxquels nous devions apporter une solution.

Chapitre 4

Analyse du processus de GL de l'unité CA

Ce chapitre fournit l'analyse du processus de GL que nous avons réalisée au sein de l'unité CA d'HQ. Ce chapitre reprend les éléments les plus pertinents de cette analyse. La version complète peut être trouvée en annexe (Annexe A).

4.1 Définition des concepts de SES et SEI

Le concept de SEI est propre au vocabulaire d'HQ. Ces termes étant utilisés dans l'analyse, il est donc important de comprendre ce qu'est une SEI et de cerner les différences qu'il existe entre une SEI et une SES. De plus, il s'agit des deux artefacts principaux du processus d'IE chez HQ.

4.1.1 SES

La SES doit décrire précisément les fonctionnalités du système à implémenter, en évitant autant que possible d'y inclure des solutions et des détails techniques de la conception du système. De cette façon, la SES sera valide quelles que soient la conception et la plate-forme matérielle retenues, permettant ainsi d'être indépendante et réutilisable pour une nouvelle conception et plate-forme matérielle.

La SES doit contenir des exigences qui sont simples, précises, avec un regroupement fonctionnel compréhensible et logique basé sur le domaine d'affaire du client. Et cela afin que le client puisse en approuver son contenu et qu'elle soit utilisable, lors des autres étapes du processus, autant par des fournisseurs internes que par des fournisseurs externes.

4.1.2 SEI

Tout comme la SES, la SEI doit être rédigée avec des exigences qui sont simples, précises, avec un regroupement fonctionnel compréhensible et logique basé sur le domaine d'affaire du client. Et cela afin que le client puisse en approuver son contenu et qu'elle soit utilisable autant par des fournisseurs internes que par des fournisseurs externes lors des autres étapes du processus.

La SEI produite, subdivisée en exigences de format et de synchronisation, doit décrire précisément les fonctionnalités de l'interface de l'utilisateur ou du système externe à implémenter, en évitant autant que possible d'y inclure des solutions et des détails techniques de la conception du système. De cette façon, la SEI sera valide quelles que soient la conception et la plate-forme matérielle retenues, permettant ainsi d'être indépendante et réutilisable pour une nouvelle conception et plate-forme matérielle. Elle doit aussi identifier les contraintes de configuration, d'Interface Personne Machine (IPM), de conception, de performance, de

disponibilité et de sécurité spécifiques à l'interface externe. De plus, elle doit expliciter le protocole d'échange utilisé ainsi que les objets utilisés pour les données dans le protocole.

La SEI devra être élaborée de telle façon qu'elle soit détachée des autres SEI d'un même système et qu'elle ne contienne que des exigences vérifiables par le client afin que la conception, l'implémentation et la vérification du service de l'interface restent indépendantes des autres.

Afin de mieux comprendre les similarités et différences entre une SES et une SEI, le lecteur trouvera les structures de ces deux types de document à la Fig. 4.1 :

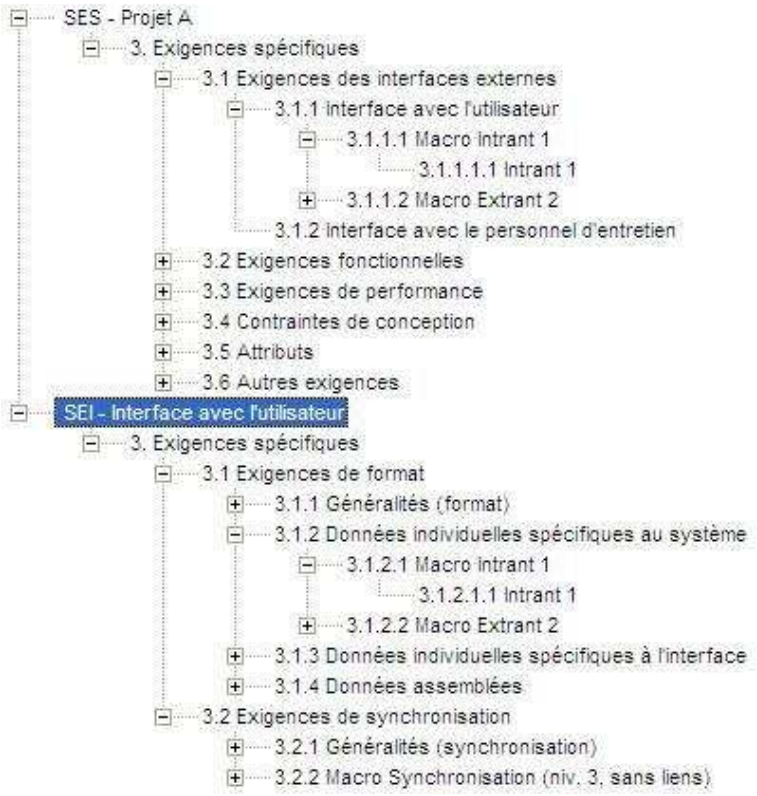

FIG. 4.1 - Exemple de structure d'une SES et d'une SEI

De manière plus précise, voici un exemple d'exigence de la SES accompagné de l'exigence correspondante dans la SEI (TAB. 4.1) [Uni 04] :

Exigence de la SES :

3.1.1.2.1 Indication de l'état EN/HORS manuelle de l'opération du délesteur
TDST doit fournir une indication de l'état EN ou HORS fonction de l'opération du délesteur.

Exigence de la SEI :

Intrants/Extrants de la SES	Format
3.1.1.2.1. Etat En/hors du système TDST	En : Vert ; Hors : Rouge ; VD : Hors

TAB. 4.1 - Exemple d'exigence d'une SEI

4.2 Élaboration du processus

Le processus de GL de l'unité CA d'HQ est le fruit d'une collaboration entre des membres du personnel : madame Véronique Chu et messieurs Harold Ratté, Michel Vincelette et Jean-Pierre Turgeon ainsi que le professeur Pierre Robillard de l'École Polytechnique de Montréal. La création de celui-ci a commencé en 2005 et il fût officialisé en 2006.

Afin de réaliser cette standardisation du processus, les personnes citées précédemment se sont basées sur certaines normes comme la ISO 9001 [Int 00] ou encore la ISO/IEC 12207 [Int 95]. La décision de s'inspirer de normes plutôt que de modèles de développement reconnus telles que les méthodes Agile a été prise car ces méthodes ont été considérées comme des phénomènes de mode tandis que les normes sont des standards plus stables et plus facilement adaptables aux besoins réels de l'entreprise.

La description du processus qui est donnée est volontairement laissée de haut niveau afin de permettre aux différents chefs de projet de réaliser les développements en suivant le modèle qu'ils désirent. Le processus est adaptable à tout type de modèle de développement logiciel. Le but est de définir les responsabilités de chacun lors d'un projet. Il permet également de décrire l'ensemble des étapes à suivre pour réaliser chaque pratique ainsi que les inputs et outputs de celles-ci. Les chefs de projet doivent réaliser l'ensemble des pratiques décrites et produire les différents artefacts qui y sont liés.

4.3 Modèle de pratique du processus normalisé

Cette section a été écrite sur base de [Vincelette 06].

4.3.1 Description

Un processus de GL est un ensemble structuré de pratiques. Avant de commencer la synthèse de l'ensemble du processus suggéré dans l'unité, il faut savoir que toutes les étapes de celui-ci suivent le même schéma. Ce schéma (Fig. 4.2) a servi de base pour synthétiser l'ensemble des étapes du processus [Vincelette 07].

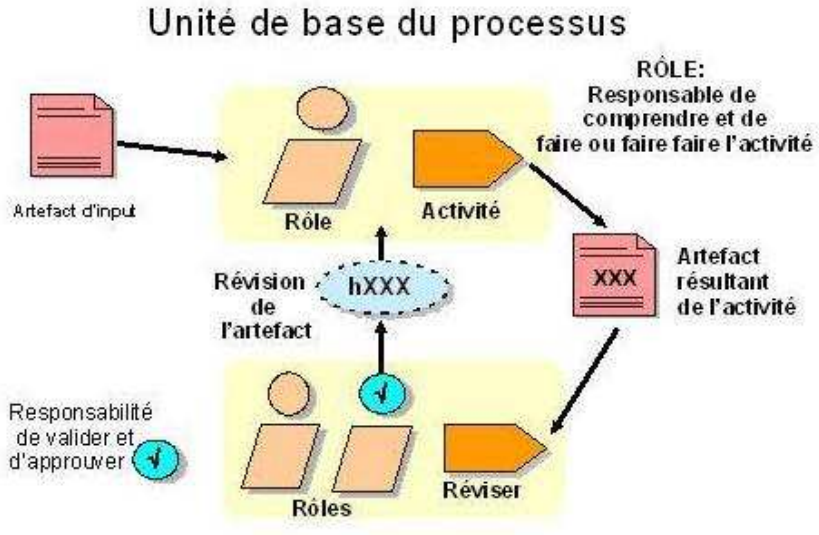

FIG. 4.2 - Modèle de pratique du processus de GL [Vincelette 07]

Chaque pratique peut être décomposée en plusieurs éléments :

- Artefacts d'input,
- Activité et participants à celle-ci,
- Artefacts d'output (résultant de l'activité),
- Révision et participants à celle-ci.

Chaque pratique identifiée doit résulter en la création ou le développement d'un artefact et celui-ci doit obligatoirement servir d'input à une pratique subséquente sauf s'il s'agit de la fin du processus et que l'artefact est un livrable final. Au besoin, les pratiques peuvent être exécutées en parallèle.

4.3.2 Artefacts d'input

S'il s'agit de la première activité du processus, les artefacts d'input sont l'engagement de base et éventuellement l'avis de problème relatif aux pratiques subséquentes. Dans tous les autres cas, les artefacts d'input sont les artefacts d'output des pratiques antérieures.

Cependant, lorsqu'il s'agit d'un projet affectant un système existant ou d'un projet correspondant à une nouvelle version d'un système, plusieurs autres documents peuvent être pertinents : normes de référence, SES ou SEI de l'ancien système, etc. Tous ces artefacts sont utilisés par le responsable et les participants à l'activité afin de la mener à bien.

4.3.3 Activité et responsables

Le responsable et les autres participants à la réalisation d'une activité doivent comprendre les artefacts d'input. Le responsable a pour objectif de réaliser ou faire réaliser l'activité identifiée. Toutefois, s'il assigne l'activité à une autre personne, il devra s'assurer de son niveau d'expérience et, au besoin, la former. Cette personne devra alors faire valider son travail par un expert du domaine.

4.3.4 Artefacts d'output

Les artefacts d'output d'une activité sont soumis à une révision (cf section 4.3.5) afin de s'assurer qu'ils rencontrent les exigences spécifiées et sont suffisants pour permettre aux responsables des activités postérieures de réaliser celles-ci.

4.3.5 Révision et participants

Comme nous l'avons vu, la boucle de révision a pour but de vérifier le contenu des artefacts d'output. Tout au long de cette revue, les participants doivent fournir une preuve de vérification composée d'un ensemble de documents décrivant le déroulement et les résultats de la révision. Afin de réaliser cette activité, les acteurs suivants sont requis : le responsable de l'activité, le responsable des inputs, le responsable de l'activité découlant de celle-ci et, éventuellement, le responsable Qualité.

A chaque itération, les participants ne traitent que la partie différentielle du contenu. Cette boucle est répétée autant de fois que nécessaire pour en arriver à l'approbation finale des participants. La fréquence des revues dépend de la complexité de l'artefact à produire.

4.4 Synthèse du processus

Cette sous-section a pour but de présenter chaque étape du processus sous forme d'un schéma et il est conseillé de se référer à la figure représentant le modèle de pratique du processus de GL (Fig. 4.2) afin de pouvoir comprendre le déroulement de chaque étape. Les rôles et responsabilités de chacun des acteurs du processus sont également décrits.

4.4.1 Description du processus

Nous donnons ici une vue d'ensemble du processus. Tout d'abord, afin de pouvoir se situer au mieux, nous fournissons un schéma (Fig. 4.3) représentant l'ensemble des étapes du processus de GL [Vincelette 06].

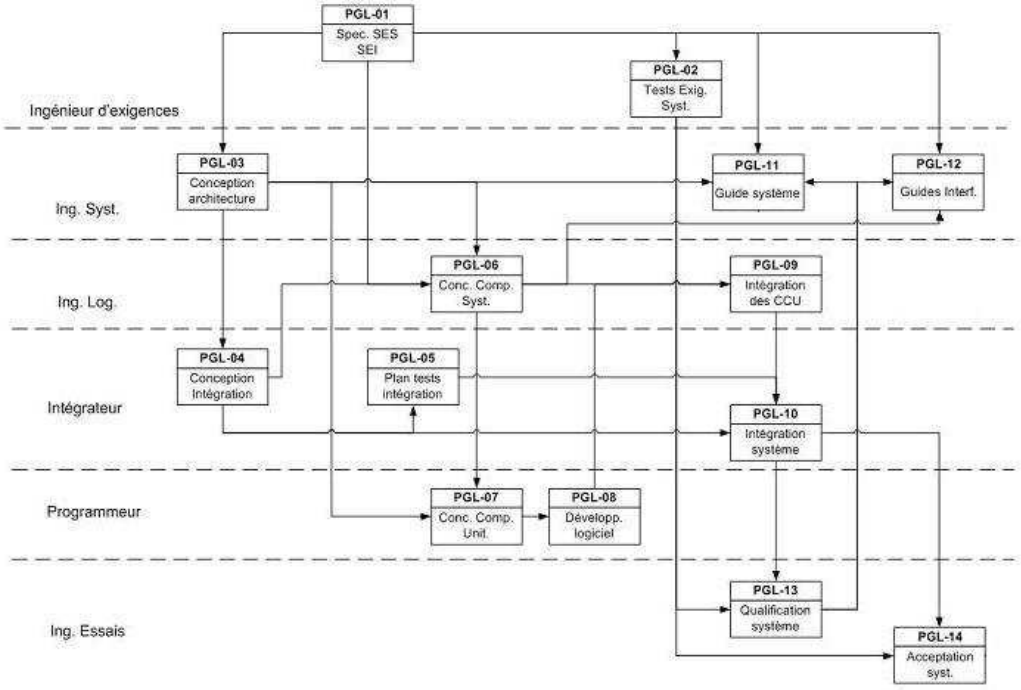

FIG. 4.3 - Étapes du processus de GL [Vincelette 06]

Ce schéma illustre la manière dont s'enchainent les différentes étapes décrites dans le processus normalisé de GL. Il permet également de percevoir les dépendances existantes entre les étapes du processus. Par exemple, l'étape PGL-13 (Fig. 4.3) de qualification du système nécessite la terminaison des étapes PGL-02 et PGL-10 (Fig. 4.3) qui sont les tests d'exigences système et l'intégration du système.

Étapes du processus dirigées par l'ingénieur des exigences

Cette sous-section présente uniquement les deux étapes du processus dirigées par l'ingénieur des exigences selon le modèle de pratique du processus (Fig. 4.2). L'ensemble de l'analyse comprenant toutes les étapes du processus est repris en annexe (Annexe A).

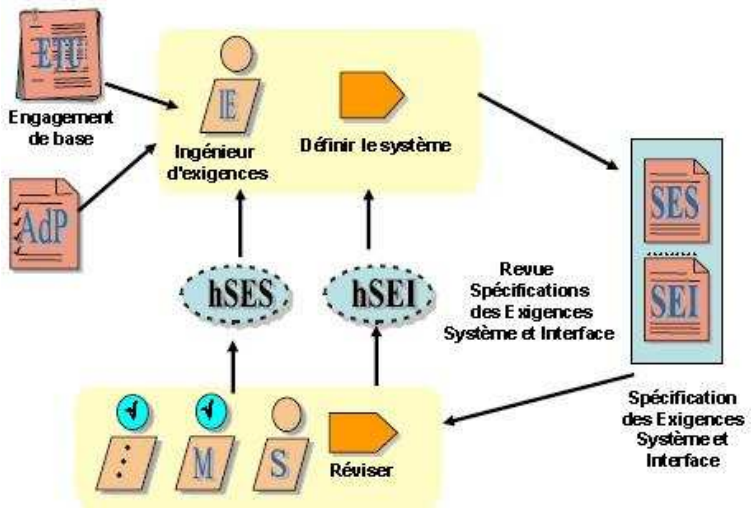

FIG. 4.4 - Spécification des exigences système et interface [Vincelette 06]

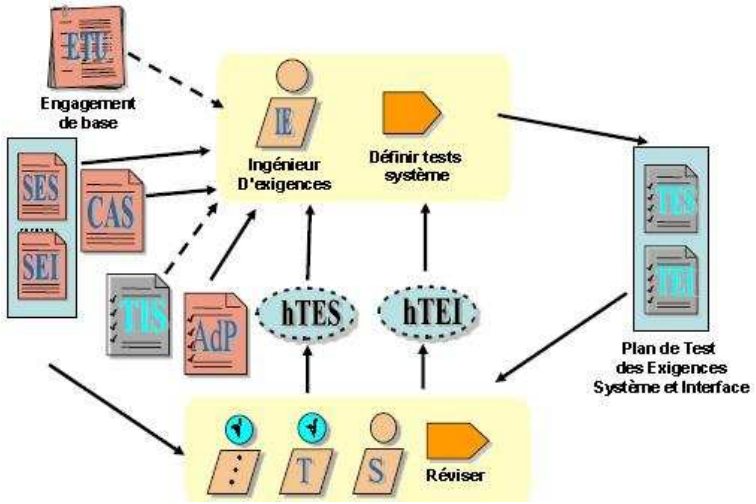

FIG. 4.5 - Planification des tests d'exigences système et interface [Vincelette 06]

4.4.2 Rôles et responsabilités

Le tableau ci-après (TAB. 4.2) reprend les différents rôles et responsabilités associés aux acteurs intervenant durant le processus de GL.

Rôles et responsabilités des acteurs du processus de GL			
Rôle	***Description***	***Responsabilité de production***	***Responsabilité de révision***
Client / Utilisateur	Expert du domaine applicatif du système (principal utilisateur)	/	Approuver les exigences, les plans, les cas de test et le produit final
Responsable de mandat ou de lot	Expert du processus de GL	Gérer les ressources et le projet ou le lot selon son assignation	Approuver les exigences et la qualification du système
Expert du domaine	Expert du domaine applicatif et du fonctionnement du système	Spécifier le fonctionnement du système et assister l'ingénieur d'exigences	/
Ingénieur d'exigences	Expert en SES et SEI	Rédiger la SES et la SEI	Approuver l'architecture et la qualification du système
Ingénieur système	Expert du domaine applicatif (caractéristiques)	Concevoir l'architecture du système	Approuver les composants et l'intégration
Ingénieur logiciel	Expert en fabrication du logiciel	Concevoir le système logiciel et ses composants	Approuver l'intégration, les composants unitaires, le code produit et les cas de test
Intégrateur	Expert en spécifications et fonctionnalités du système	Planifier la construction et intégrer les composants	Approuver les composants, les essais unitaires et le logiciel
Programmeur	Expert en spécifications du système et en programmation	Programmer les composants du système logiciel	/
Ingénieur des essais	Expert des stratégies de test et des fonctionnalités des composants	Tester le système	Approuver le contenu des guides techniques
Responsable Assurance Qualité	Expert du processus de GL et des normes en vigueur	Gérer le processus, le respect des normes et la qualité	/

TAB. 4.2 - Rôles et responsabilités des acteurs du processus de GL

4.5 Analyse et évaluation du processus

4.5.1 Méthode

La démarche que nous avons suivie est orientée selon deux grands axes. Le premier, une analyse du processus sur base de documents que nous avons récoltés et sur base d'entrevues que nous avons réalisées. En effet, nous avions à notre disposition des documents décrivant le processus de développement normalisé. De plus, nous avons eu l'occasion d'interroger monsieur René Bujold, ingénieur au sein de l'unité et n'ayant pas participé à l'élaboration du processus de GL. Ensuite, nous avons interviewé monsieur Harold Ratté, chargé de l'équipe d'Automatismes de Réseau et ayant participé à l'élaboration du processus.

Le deuxième axe est une évaluation de type CMMi. Celle-ci a été produite grâce aux entrevues avec les deux personnes citées précédemment ainsi que monsieur Michel Vincelette, responsable de l'Assurance Qualité et ayant participé à l'élaboration du processus.

Grâce à l'évaluation de type CMMi et notre connaissance du domaine du développement logiciel, nous avons déterminé les avantages et inconvénients du processus normalisé. Nous avons également proposé différentes améliorations pouvant être apportées à ce processus. Cette analyse a été répartie sur les trois premières semaines de notre stage.

4.5.2 Situation actuelle

Actuellement, les procédures décrites dans le document de support au processus de GL servent uniquement de guide aux différents chefs de projet pour les développements au sein de l'unité. En effet, au moment de l'analyse, l'ensemble des projets en cours avaient tous débuté avant l'élaboration et la mise en place du processus normalisé. Ces procédures n'étant pas obligatoires, la majorité des chefs de projet et des employés ont préféré suivre leur propre méthode de travail basée sur leur expérience personnelle afin d'éviter toute perturbation au sein de l'organisation des projets.

Le processus étant très jeune, aucun audit de son utilisation n'avait été réalisé au moment de la réalisation de cette analyse. Toutefois, des audits étaient prévus pour le début de l'année 2008 et un processus d'amélioration continue est mis en place. On peut donc s'attendre à ce que le processus s'impose petit à petit dans les projets à venir.

Révision de conformité

A l'heure actuelle, le processus est soumis à une révision visant à assurer sa conformité par rapport à la norme ISO/IEC 12207 [Int 95]. Un certain nombre de clauses sont déjà respectées par le processus actuel et d'autres font l'objet d'une étude préalable à leur intégration au processus. Certains éléments ne seront jamais pris en compte car ceux-ci sont réalisés par d'autres organismes ou ne sont pas jugés nécessaires pour l'unité. Pour plus d'information au sujet de la norme ISO/IEC 12207, l'analyse complète (Annexe A) apporte plus de précisions.

4.5.3 Évaluation de type CMMi

Présentation

Cette sous-section est tirée de [Borderie 06].

Le CMMi, pour *Capability Maturity Model Integration*, est une extension de la spécification CMM première, créée pour le ministère de la Défense américain en 1989 afin de déterminer si un projet interne ou tiers serait terminé dans les temps, selon le budget et les spécifications. CMMi étend CMM en y intégrant les avancées d'autres spécifications proches, et établies entre-temps pour pallier les manques de CMM.

CMMi est donc avant tout un référentiel d'évaluation de la capacité à gérer et à terminer un projet correctement, proposant nombre de bonnes pratiques liées à la gestion, au développement et à la maintenance d'applications et de systèmes. Le but étant de fixer un niveau de maturité au processus global de développement d'un projet.

En ce qui concerne l'évaluation ci-dessous, nous nous sommes limités au processus d'IE. Nous avons fait ce choix pour plusieurs raisons. Tout d'abord, une évaluation CMMi complète doit être réalisée par des experts en la matière. De plus, ce travail peut prendre plusieurs semaines et la priorité de notre stage n'était pas cette analyse. Enfin, notre stage s'effectuant dans le domaine de l'IE, nous avons privilégié cette partie de CMMi.

Les cinq niveaux de maturité CMMi

A chaque niveau, on retrouve un ensemble de critères à respecter et pour passer à un niveau supérieur. Tous les critères des niveaux inférieurs doivent être remplis afin d'accéder à un niveau plus élevé. Le tableau 4.3 reprend l'ensemble de ces niveaux. Toutefois, les critères d'évaluation concernant l'IE sont répertoriés uniquement aux niveaux 1, 2 et 3. C'est pourquoi nous avons dû adapter CMMi au contexte qui était le nôtre.

Niveaux de maturité CMMi	
Niveau	***Signification***
Niveau 1 : Initial	Lorsque l'on se situe au niveau 1, les résultats sont imprévisibles. L'atteinte des résultats repose plus sur les hommes, sur leur engagement et leur bonne volonté que sur l'application disciplinée de bonnes pratiques définies. Les succès passés ne sont pas pris en compte lors de la réalisation de nouveaux projets.
Niveau 2 : Répétitif	On se base sur l'expérience des projets antérieurs, en faisant appel à une certaine discipline et à une gestion de base du projet (gestion des exigences, estimations de charge argumentées, suivi de projet, mesure d'indicateurs, contrôle qualité, etc.). Certains processus sont maîtrisés ; il est possible de les répéter.
Niveau 3 : Défini	A ce niveau, l'organisation dispose d'un ensemble de processus standards, qui sont adaptés par chaque projet. Chaque projet capitalise son expérience et permet de bonifier le capital collectif.
Niveau 4 : Géré	A ce niveau, les processus clés sont sous contrôle statistique (surveillance d'indicateurs quantitatifs, et actions correctrices si dérives). Les performances des processus sont prévisibles en quantité comme en qualité.
Niveau 5 : Optimisé	Les processus sont constamment améliorés de manière incrémentale et innovante. Les objectifs sont revus en permanence pour rester proches des besoins du marché. Les évolutions sont anticipées et gérées de bout en bout.

TAB. 4.3 - Niveaux de maturité CMMi [Product Team 07b]

Ces niveaux ont pour but de donner une évaluation de la maturité d'un processus de développement de projet. Une étude du Software Engineering Institute [Product Team 07a], réalisée sur 1.377 organisations, montre qu'environ 70% des entreprises se situent aux niveaux 2 et 3.

Méthode d'évaluation

Afin de réaliser l'évaluation de type CMMi, nous avons interviewé trois personnes : messieurs Harold Ratté (Ingénieur dans l'unité CA et responsable de l'équipe d'Automatismes de Réseau) et Michel Vincelette (Ingénieur dans l'unité CA et responsable de l'Assurance Qualité) qui ont participé à l'élaboration du processus de GL de l'unité, et monsieur René Bujold (Ingénieur dans l'unité CA et responsable du projet *GenSpec*) qui n'a pas participé

à l'élaboration de celui-ci.

Pour récolter leurs résultats respectifs, nous leur avons demandé, pour chaque critère CMMi concernant l'IE, de l'évaluer avec une échelle allant de 1 à 5. Le but étant de faire correspondre ces chiffres au niveaux de maturité de CMMi. Le chiffre 5 correspond à une pratique réalisée complètement de manière formelle et le chiffre 1 correspond à une pratique inexistante dans le processus de l'unité. Le détail des autres critères se trouvent dans la version complète de l'analyse en annexe (Annexe A). Le but était d'évaluer le degré de formalité avec lequel la pratique décrite dans chaque critère était réalisée.

Nous leur avons demandé d'effectuer cette évaluation aussi bien pour le processus normalisé que pour le processus utilisé actuellement. Nous avons ensuite calculé des moyennes sur base des résultats des trois participants tout en gardant la distinction entre le processus théorique et le processus réel. Ces calculs fournissent un chiffre entre 1 et 5 correspondant aux niveaux de maturité de CMMi.

Résultats

Comme nous l'avons dit précédemment, nous nous sommes limités au processus d'IE. Pour mieux comprendre les résulats, nous présentons les critères issus de [Product Team 07b] concernant l'IE (Fig. 4.4) :

Critères CMMi concernant l'IE	
ID	***Pratique***
REQM-SP1.1	Développer une compréhension commune des exigences et de leur signification avec ceux qui les ont formulées.
REQM-SP1.2	Obtenir l'engagement des participants sur les exigences.
REQM-SP1.3	Gérer les modifications aux exigences au fur et à mesure de leur évolution en cours de projet.
REQM-SP1.4	Maintenir une traçabilité biunivoque entre les exigences d'une part et les plans du projet et les produits de sortie d'autre part.
REQM-SP1.5	Identifier les incohérences entre les plans du projet et les produits de sortie d'une part et les exigences d'autre part.
REQM-GP2.1	Établir et maintenir une directive organisationnelle traitant de la planification et de la mise en oeuvre du processus.
REQM-GP2.2	Établir et maintenir le plan pour la mise en oeuvre du processus.
REQM-GP2.3	Fournir les ressources adéquates pour la mise en oeuvre du processus, le développement des produits de sortie et la prestation des services couverts par le processus.
REQM-GP2.4	Assigner les responsabilités et l'autorité pour la mise en oeuvre du processus, le développement des produits de sortie et la prestation des services couverts par le processus.
REQM-GP2.5	Former selon les besoins les personnes qui mettent en oeuvre ou soutiennent le processus.
REQM-GP2.6	Gérer en configuration, au niveau approprié, les produits de sortie du processus.
REQM-GP2.7	Identifier et impliquer, tel que planifié, les parties prenantes concernées par le processus.
REQM-GP2.8	Suivre et contrôler le processus vis-à-vis de son plan de mise en oeuvre et mener les actions correctives appropriées.

ID	*Pratique*
REQM-GP2.9	Évaluer de manière objective la conformité du processus à sa description, ses normes et ses procédures et traiter les non-conformités détectées.
REQM-GP2.10	Passer en revue avec les responsables de niveau supérieur les activités, le statut et les résultats du processus et résoudre les points soulevés.
REQM-GP3.1	Établir et maintenir la description d'un processus personnalisé.
REQM-GP3.2	Collecter les produits de sortie, les descriptions de mesures, les résultats de mesures et les retours d'expérience dérivés de la planification et de la mise en oeuvre du processus en vue de soutenir l'usage futur des processus organisationnels et des actifs associés ainsi que leur amélioration.

TAB. 4.4 - Critères CMMi concernant l'IE [Product Team 07b]

Il faut savoir que plusieurs de ces critères sont subdivisés en sous-critères. Toutefois, nous avons préféré agréger les résultats par soucis de clarté. Seul les sous-critères du critère REQM-SP1.1 ont été distingués car l'un d'eux pose problème au niveau du processus utilisé actuellement et du processus théorique.

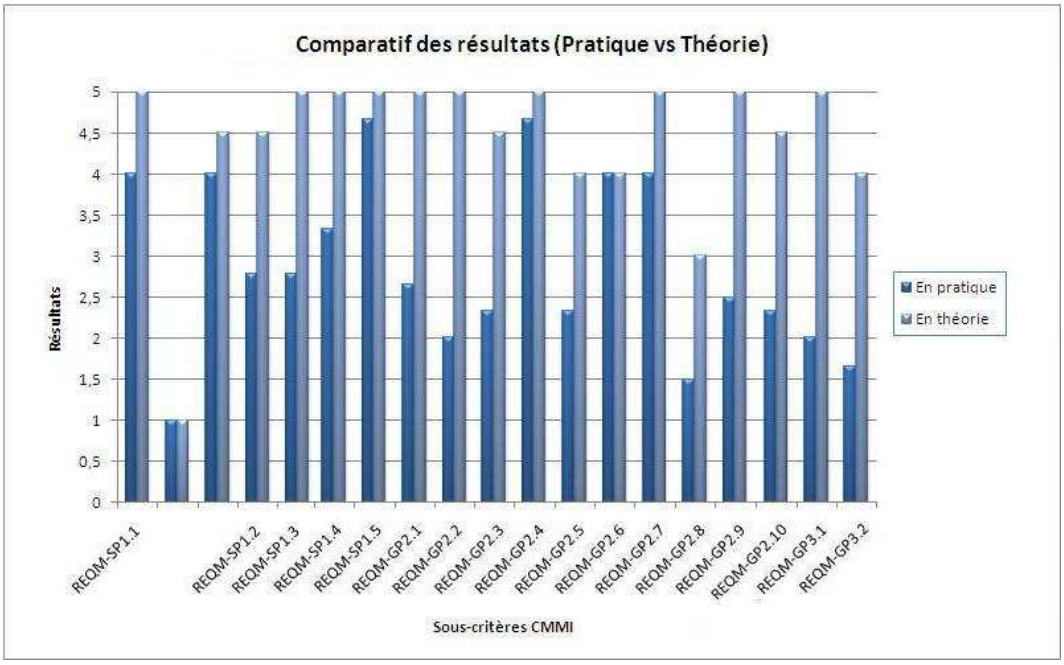

FIG. 4.6 - Résultats de l'évaluation de type CMMi

Processus utilisé actuellement

Si on observe le graphique des résultats de l'évaluation de type CMMi (Fig. 4.6), nous pouvons constater qu'il existe quelques lacunes au niveau du processus suivi au sein de l'unité. A l'heure actuelle, la moyenne des scores est de 2,80 points.

Les critères REQM-SP1.1 à REQM-SP1.5 mettent en évidence les efforts réalisés afin d'améliorer la compréhension du client (REQM-SP1.1) et afin de s'assurer de son accord vis-à-vis des documents d'exigences (REQM-SP1.2). En effet, la SES de tout projet doit être signée par les deux parties. De plus, pour gérer les modifications apportées aux exigences (REQM-SP1.3), les chefs de projet de l'unité CA utilisent un document reprenant les modifications techniques apparues en cours de projet. Il faut maintenir une traçabilité

entre les exigences, d'une part, et les plans du projet et les produits de sortie, d'autre part (REQM-SP1.4). Comme nous l'avons vu, *GenSpec* offre des fonctionnalités afin de gérer cela (cf section 3.1.3). Enfin, au niveau de la gestion des incohérences entre les exigences et les produits de sortie (REQM-SP1.5), l'unité utilise, entre autres, les avis de problème qui permettent d'identifier les problèmes rencontrés en cours de projet. Toutefois, le sous-critère concernant l'utilisation de métriques (deuxième sous-critère du REQM-SP1.1) semble poser problème.

Pour les critères REQM-GP2.1 à REQM-GP3.2, seuls les aspects des critères REQM-GP2.4, REQM-GP2.6 et REQM-GP2.7 semblent être bien gérés. Les critères REQM-GP2.4 et REQM-GP2.7 montrent que la répartition des rôles est bien définie et appliquée lors du développement d'un projet. En effet, chaque personne impliquée sait ce qu'elle doit faire et produire afin que le projet aboutisse. Le critère REQM-GP2.6 concerne la gestion en configuration, au niveau approprié, des produits de sortie du processus. Celui-ci est mis en oeuvre via l'utilisation de gabarits et de méthodes de classement de documents (numérotation, gestion des versions, etc.). Les faibles résultats obtenus pour les autres critères mettent en avant le manque de standardisation du processus actuel et la sous-utilisation des mécanismes permettant de récolter les informations utiles en fin de projet.

En ce qui concerne le niveau de maturité du processus IE, nous estimons que le processus utilisé actuellement n'est que de niveau 2.

Processus normalisé

Au niveau du processus normalisé, la majorité des critères ont un score supérieur à 4 et la moyenne atteint 4,52 points. Seuls deux critères semblent poser problème.

Le deuxième sous-critère du REQM-SP1.1 concernant les critères d'évaluation et d'acceptation des exigences n'est pas réellement pris en compte au sein de l'unité. En effet, un processus d'évaluation et d'acceptation des exigences est réalisé mais sans faire appel à des critères déterminés et réutilisés lors de différents projets.

Le point REQM-GP2.8 concernant le suivi du processus et de sa mise en oeuvre, ainsi que les actions correctives n'est pas encore très au point dans le processus normalisé. Une gestion du suivi de la qualité et du respect des normes est en train d'être mise en place mais, le processus étant soumis à l'heure actuelle à une révision de conformité à la norme ISO 12207 [Int 95], celle-ci n'est pas encore optimale.

En ce qui concerne le niveau de maturité du processus IE, nous pensons que le processus normalisé atteint le niveau 4.

4.5.4 Avantages et inconvénients

En ce qui concerne les avantages et inconvénients du processus, l'accent a été mis sur le processus décrit de manière normalisée. Ceux-ci ont été déterminés grâce à l'évaluation de type CMMi et notre connaissance du développement logiciel. Les inconvénients sur lesquels nous nous sommes penchés plus particulièrement durant notre stage sont repris au chapitre 5.

Avantages

- Le processus est normalisé et conforme aux normes en vigueur telle que la norme ISO 12207 [Int 95]. Il permet de mettre en pratique des **méthodes efficaces et réutilisables** pour le développement de projets. Il permet également de **délimiter l'ensemble des étapes** du processus.

- La **définition des rôles et des responsabilités** décrite dans le processus permet à chacun des acteurs de savoir quand, comment et avec qui il doit intervenir au cours du développement d'un projet.

- Les **inputs et outputs** de chaque étape du processus sont **clairement explicités**. De ce fait, les responsables chargés de valider chaque étape savent ce qui doit être produit afin de ne pas entraver le bon déroulement du processus de développement.

- Grâce au schéma décrivant les liens existants entre les différentes pratiques du processus (Fig. 4.3), il est possible de **déterminer les étapes qui peuvent être réalisées en parallèle et celles qui doivent être mises en attente**. Il est également plus aisé de **transférer une partie du travail** à d'autres groupes.

- Grâce à l'utilisation des gabarits de documents suggérés par le processus, la **gestion des documents** est **uniformisée**.

- Grâce à l'**utilisation d'un logiciel tel que *GenSpec***, il est possible d'améliorer la rédaction des exigences, la vérification de celles-ci et leur traçabilité.

- Lors du départ d'un employé, la standardisation du processus permet de **connaître** exactement quelles sont **ses responsabilités et comment réaliser l'activité** dont il était chargé. De ce fait, un nouvel employé arrivant dans l'unité sera plus facilement intégré. En effet, il lui suffira de consulter les documents concernant le processus afin de comprendre la manière de travailler de l'unité et la manière d'appréhender la tâche qui lui est dévolue.

- Actuellement, 80% des projets de l'unité dépassent les délais et/ou les coûts estimés en début de projet. Avec les pratiques décrites dans le processus, la partie concernant l'analyse préliminaire est plus complète. Les **prévisions en terme de délais et de coûts** seront donc **plus fiables** et le pourcentage de projets les respectant sera plus élevé.

- Un dernier avantage est de pouvoir élaborer des **statistiques sur base de métriques**. Grâce à cela, les nouveaux projets pourront se baser sur ces données afin de réaliser la répartition du temps de travail, par exemple.

Inconvénients

- L'un des inconvénients majeurs se situe au niveau de la formalisation des exigences. De nombreux employés ont en effet des **lacunes en IE** ce qui aboutit, lors de la formalisation des exigences, à des documents d'exigences mal conçus voire inutilisables. Cela résulte en une perte de temps importante et des spécifications inadéquates.

- Au niveau de l'unité, **aucune méthode précise** n'est utilisée **afin de gérer les risques**. Il y a une analyse des risques qui est faite lors de l'élaboration du plan d'ingénierie mais durant le projet, les problèmes sont réglés au cas par cas et ce même s'ils avaient été envisagés.

- La **gestion du temps de travail** imparti à chaque employé n'est également **pas très optimale**. Actuellement, l'unité utilise un système à cartes permettant de relever le nombre d'heures prestées par chacun toutes les deux semaines. A la fin de cette période chaque personne doit répartir ses heures de travail dans différentes catégories générales. Toutefois, il devrait être possible de connaître le détail de ces dernières. De plus, cette comptabilisation doit être effectuée à la fin des deux semaines et les employés éprouvent des difficultés à attribuer leurs heures de manière précise.

- A l'heure actuelle, **l'entreprise ne possède pas un bon système centralisé pour**

rendre compte de l'expérience acquise au cours d'un projet. En fin de projet, une réunion concernant les leçons apprises est effectuée mais les résultats de celle-ci ne sont pas utilisés de manière optimale.

- La standardisation du processus impose également une **quantité importante de documentation à générer** durant un projet. Cela peut sembler contraignant voire inutile pour certains employés désirant éviter toute perte de temps.

4.5.5 Améliorations

Suite à l'analyse du processus de GL, nous avons proposé certaines améliorations. Les améliorations ayant fait l'objet de notre stage sont décrites au chapitre 8 et ne sont pas reprises ici. Dans cette sous-section nous reprenons uniquement les principales améliorations que nous avons proposées et que nous n'avons pas implémentées : la gestion des risques et la gestion des connaissances. Certaines adaptations étaient déjà envisagées au sein de l'unité. Pour connaître le détail des propositions qui ont été faites ainsi que des améliorations envisagées au sein de l'unité, nous renvoyons le lecteur à l'analyse complète qui se trouve en annexe (Annexe A). Cette sous-section n'en présente que l'essentiel.

Améliorations envisagées au sein de l'unité

- **Mise en place d'une équipe spécialisée en IE** pour encadrer les personnes rédigeant les spécifications afin de former ces dernières qui pourront ensuite, à leur tour, aider d'autres employés.

- **Réalisation d'audits** afin de connaître l'opinion de chacun au sujet du processus normalisé. L'utilisation d'audits fait partie du processus d'amélioration continue mis en place.

- **Utilisation d'un logiciel de gestion de projet** afin de suivre au quotidien l'avancement des tâches pour chaque employé et ceci de manière détaillée.

- **Utilisation de gabarits** pour les documents de projet afin de parfaire l'uniformisation au sein de l'unité.

Gestion des risques

Cette recommandation s'inspire largement de [Morley 97] et [Pro 04].

Afin de résoudre les problèmes liés à la gestion des risques, il serait opportun de mettre en place un réel processus de gestion des risques. Le but d'un tel processus est de minimiser la probabilité et les conséquences des éléments défavorables à un projet. Le processus que nous décrivons ci-dessous est inspiré du guide du PMBOK. Afin de pouvoir mettre en place un tel processus de gestion des risques, il faut une implication des divers utilisateurs, le soutien de la hiérarchie et une formalisation claire des besoins. Il faut savoir que suite à cette suggestion, monsieur René Bujold, créateur du logiciel *GenSpec*, a décidé d'ajouter à l'outil un module de gestion des risques (cf section 3.2.2).

La gestion des risques peut être scindée en plusieurs étapes à réaliser lors de chaque projet :

- **Planification de la gestion des risques :** Le but de la planification de la gestion des risques est de décider comment les risques seront gérés. Pour cela, il faut organiser une ou plusieurs réunions de planification durant lesquelles les méthodes utilisées pour analyser et gérer les risques seront déterminées. Il faut également décider du nombre de fois où les risques seront réévalués au cours du projet.

- **Identification des risques :** L'identification des risques doit être réalisée lors de l'étude précédant le commencement du projet. Le but de cette identification est de parvenir à déterminer tous les risques pouvant survenir durant le projet et leurs impacts sur ce dernier. Les techniques utilisées pour identifier les risques sont la revue de documentation, la collecte d'informations (méthode Delphi, analyse SWOT, etc.), l'analyse de listes de contrôle ou encore des techniques utilisant des diagrammes (diagrammes d'Ishikawa, etc.). Il est également intéressant de déterminer une typologie des risques (réutilisable dans plusieurs projets). Sur base de tout cela, il est possible de déterminer le profil du projet.

- **Analyse qualitative et quantitative des risques :** Pour réaliser une analyse qualitative des risques, la technique qui semble être la plus répandue est celle de la matrice de probabilité-impact. Cette matrice a comme ordonnée la probabilité de survenance du risque et, en abscisse, l'impact de celui-ci sur le projet. Grâce à cela, un classement des risques peut être réalisé. Il est également envisageable d'utiliser des critères et des métriques qui seraient alors à déterminer.

  Pour l'analyse quantitative, on peut faire appel à des techniques telles que l'analyse de sensibilité, l'utilisation d'arbres de décision, etc. La technique d'analyse de sensibilité permet de déterminer les facteurs ayant le plus d'impact potentiel. Elle peut déboucher sur un diagramme en tornade qui illustre le calcul de la variation du risque en fonction de la variation d'un paramètre (de sa valeur la plus faible à sa valeur la plus grande).

- **Planification des réponses aux risques :** Afin de ne pas perdre de temps en cours de projet, il est opportun de planifier les réponses aux risques qui ont été jugés importants. Pour ces risques, il faut donc décider de ce qui sera fait dans le cas où ils se concrétisent ou prendre des dispositions avant le début du projet. Il existe plusieurs types de réponses aux risques :

  - *Éviter* : modifier le plan de gestion de projet pour éliminer la menace du risque,
  - *Transférer* : détourner l'impact négatif et la responsabilité d'établir une réponse vers un tiers (prise d'une assurance, etc.),
  - *Atténuer* : modifier le plan de gestion de projet pour réduire la probabilité et/ou les conséquences d'un risque,
  - *Accepter* : ne pas modifier le plan de gestion de projet et accepter le risque,
  - *Réponse conditionnelle* : préparer un plan de réponse mis en oeuvre sous certaines conditions.

- **Surveillance et maîtrise des risques :** La surveillance et la maîtrise des risques consistent à suivre les risques identifiés, surveiller les risques résiduels, identifier les risques nouveaux, exécuter les plans de réponse et évaluer leur efficacité. Pour cela il faut utiliser des techniques telles que les réunions de revue de projet, la réévaluation des risques, les audits, etc.

Gestion des connaissances

La gestion des connaissances est un point important pour une entreprise car elle représente sa mémoire et permet une amélioration continue. La gestion des connaissances est l'ensemble des méthodes et des techniques permettant de percevoir, d'identifier, d'analyser, d'organiser, de mémoriser, et de partager des connaissances entre les membres des organisations, en particulier les savoirs créés par l'entreprise elle-même ou acquis de l'extérieur [Dening 04]. Nous pouvons distinguer deux types de connaissances, celle de l'entreprise et celle relative à un projet. La connaissance de l'entreprise est en fait le regroupement des expériences acquises lors de divers projets.

La gestion des connaissances dans le cadre d'un projet consiste à fournir un rapport à la fin du projet qui est élaboré de manière continue tout au long de celui-ci. Ce rapport

contiendrait un résumé des procédures, des outils, des méthodes utilisés pour le projet ainsi qu'une critique de chacun. Il serait alors possible de savoir les avantages et les inconvénients qu'on eut les choix réalisés et de pouvoir réutiliser ses informations pour des projets futurs. Plus précisement, ce rapport devrait faire état de la gestion des délais, des coûts, des risques et des changements mais aussi de la manière dont les employés ont collaboré. Les expériences apprises pourraient ensuite être rassemblées dans une BD pour servir ultérieurement.

La mémoire d'une entreprise peut ainsi être créée et permet, dans beaucoup de cas, de gagner un temps précieux. La BD des connaissances est une source pour les chefs de projets ou employés à la recherche de solutions. Cela évite de faire les mêmes erreurs que dans le passé et donc d'apprendre sur base de ces expériences.

Pour plus d'information au sujet de la gestion des connaissances, nous renvoyons à [Dening 04].

4.5.6 Conclusions de l'analyse et de l'évaluation du processus

L'analyse du processus et l'évaluation de type CMMi de celui-ci, nous ont permis de découvrir le contexte dans lequel nous avons dû travailler. Cela nous a également permis de cibler les problèmes auxquels nous devions apporter une solution durant notre stage (Chapitre 5) et de percevoir les possibilités d'amélioration que nous avons implémentées par la suite (Chapitre 8).

L'analyse et l'évaluation ont été menées grâce à diverses entrevues. Toutefois, parmi les personnes interrogées, deux ont participé à l'élaboration du processus normalisé. Il faut donc tempérer les résultats que nous avons obtenus pour l'évaluation de type CMMi. En effet, les réponses données par ces personnes sont peut-être biaisées par le fait qu'elles ne veulent pas trop critiquer un processus qu'elles ont contribué à élaborer.

Malgré ce risque, l'analyse et l'évaluation ayant suivi ces entrevues ont été réalisées de la manière la plus objective possible par nos soins.

4.6 Résumé

Ce chapitre a présenté l'essentiel de l'analyse du processus de GL réalisée dans l'unité CA d'HQ.

A l'heure actuelle, un processus normalisé est en train de se mettre en place, même si celui-ci n'est pas encore vraiment utilisé. En effet, les projets en cours au sein de l'unité, au moment de l'analyse, avaient tous commencé avant l'élaboration de ce processus.

Les avantages et inconvénients du processus de GL ont été analysés et, suite à cela, des améliorations ont été proposées. Parmi les améliorations que nous ne devions pas implémenter, la gestion des risques a été considérée comme la plus intéressante par le comité *GenSpec*. Cette suggestion se base sur le processus de gestion des risques décrit dans le guide du PMBOK et fera l'objet d'une future amélioration du logiciel *GenSpec* (cf section 3.2.2). Les problèmes auxquels nous avons apporté des solutions sont présentés au chapitre 5 et les améliorations que nous avons apportées à l'outil *GenSpec* sont décrites au chapitre 8.

L'évaluation de type CMMi réalisée a permis de comparer le processus utilisé actuellement au sein de l'unité et le processus normalisé qui sera bientôt mis en place. Les résultats ont démontré les améliorations pouvant être apportées par le processus normalisé, celui-ci obtenant d'excellents résultats.

Chapitre 5

Problématique

Ce chapitre décrit les problèmes rencontrés par les utilisateurs de *GenSpec* au sein de l'unité CA d'HQ auxquels nous avons apporté une solution. Pour plus de clarté, nous avons subdivisé ce chapitre en deux grandes parties représentant les domaines sur lesquels notre travail était axé. Premièrement, l'amélioration de la qualité des documents d'exigences via l'ajout de types d'exigences et d'énoncés par défaut. Deuxièmement, la rédaction des exigences d'interface.

5.1 Qualité des documents d'exigences

Différents problèmes peuvent surgir lors de la rédaction et de la gestion des exigences, nous nous sommes penchés sur quatre d'entre eux :

- Redondance et incohérence des exigences,
- Manque de complétude,
- Manque de structuration,
- Manque d'uniformisation des exigences.

Ces problèmes ne sont pas les seuls problèmes auxquels nous nous sommes intéressés. En effet, d'autres problèmes liés à l'outil *GenSpec* ont été constaté et pris en compte lors de notre travail. Ceux-ci sont exposés dans la sous-section 5.1.5.

5.1.1 Redondance et incohérence des exigences

Une même exigence peut être formulée de façon différente selon les personnes. Il est possible que deux personnes ajoutent la même exigence mais avec une formulation différente. De ce fait, les documents d'exigences peuvent comporter certains éléments redondants. Ce type d'erreur est à proscrire car le client pourrait croire que les besoins qu'il a émis ont été mal compris ou, du moins, émettre certaines remarques quant à la qualité des documents d'exigences.

De plus, la redondance peut augmenter le risque d'incohérence. En effet, si l'on est redondant dans la rédaction des documents d'exigences, une modification apportée à une exigence peut produire une incohérence si on ne modifie pas de la même façon les exigences dites redondantes.

Par exemple, voici deux exigences redondantes exprimées de manière différente :

- Les interfaces de l'application X doivent offrir un temps de réponse inférieur à 1sec.
- Le temps de réponse des interfaces de l'application X ne devront pas excéder 1sec.

Un problème d'incohérence peut alors surgir, si l'on ne supprime pas l'une de ces exigences et que l'on modifie l'une d'elle. Par exemple, en modifiant la première on peut avoir des temps de réponse exigés différents :

- Les interfaces de l'application X doivent offrir un temps de réponse inférieur à 2sec.
- Le temps de réponse des interfaces de l'application X ne devront pas excéder 1sec.

5.1.2 Manque de complétude

Les documents d'exigences doivent être les plus complets possibles. Ceux-ci servent de base contractuelle avec le client. Il faut donc qu'ils contiennent l'ensemble des besoins énoncés par ce dernier. Toutefois, certains clients peuvent ne pas être habitués à énoncer des besoins tout comme certaines personnes chargées de réaliser des documents d'exigences peuvent ne pas être habituées à formuler les besoins d'un client. Un problème au niveau de la complétude des documents d'exigences peut entrainer une implémentation tardive et coûteuse d'exigences essentielles ayant été oubliées. Il en résulte alors une augmentation des coûts et des délais de réalisation. Un manque de complétude des documents d'exigences peut également diminuer la qualité du produit si certaines exigences ayant été oubliées ne sont pas implémentées.

Par exemple, les personnes chargées de réaliser les documents d'exigences peuvent oublier de mentionner cette exigence :

- Les interfaces de l'application X doivent offrir un temps de réponse inférieur à 1sec.

Si cette exigence est considérée comme essentielle, il faudra vérifier et corriger l'ensemble de l'application afin que le temps de réponse maximal soit respecté. De ce fait, les coûts et délais de réalisation seront augmentés.

5.1.3 Manque de structuration

Lorsque l'on consulte divers documents d'exigences réalisés par un ensemble de personnes différentes, on constate que la manière de structurer les exigences peut grandement différer. Certaines personnes regroupent plusieurs exigences sous un même titre. Par exemple, elles regroupent toutes les contraintes sous un même titre alors que d'autres personnes formulent une seule exigence sous un titre. On obtient alors une différenciation entre les contraintes de conception, les contraintes d'implémentation, etc. Chaque exigence se retrouve donc sous un titre qui lui est propre. Cela nécessite l'utilisation d'un plus grand nombre de niveaux de titre. Il ne faut pas négliger cet aspect lors de la rédaction des documents d'exigences car une bonne structure facilite la lecture et la compréhension du client et des concepteurs.

5.1.4 Manque d'uniformisation

Lors de la rédaction des documents d'exigences, des règles d'uniformité doivent être respectées. Ces dernières permettent d'énoncer des exigences respectant les normes (telles que les directives ISO/CEI [Dir 04]) et de les rendre compréhensibles par un plus grand nombre de personnes.

> *"Des exigences identiques doivent être rédigées de façon identique, et des exigences analogues doivent être rédigées de façon analogue."*

En effet, cela facilite la lecture, la compréhension et l'approbation des exigences par un client mais aussi par les concepteurs.

Par exemple, voici deux exigences exprimant le même type d'information de manière non-uniformisée :

- Les interfaces avec les utilisateurs de type A de l'application X doivent offrir un temps de réponse inférieur à 1sec.
- Le temps de réponse des interfaces dédiées aux utilisateurs de type B de l'application X ne devront pas excéder 2sec.

Ces deux exigences ont pour but de fixer un temps de réponse maximal pour les interfaces d'un type d'utilisateur précis. Il faut donc les uniformiser de la manière suivante :

- Les interfaces avec les utilisateurs de type A de l'application X doivent offrir un temps de réponse inférieur à 1sec.
- Les interfaces avec les utilisateurs de type B de l'application X doivent offrir un temps de réponse inférieur à 2sec.

5.1.5 Problèmes liés à l'outil

Les problèmes liés à l'outil de support *GenSpec* qui ont été rencontrés dans le cadre de la rédaction d'exigences sont les suivants :

- Manque de types d'exigences et absence de gestion des types,
- Aucune génération automatique d'énoncé par défaut.

Manque de types d'exigence et absence de gestion des types

Lors de notre arrivée au sein de l'unité CA, la version courante du logiciel *GenSpec* offrait des possiblités trop limitées au niveau des types d'exigences. En effet, il n'existait qu'une trentaine de types disponibles. Certaines fonctionnalités étaient présentes afin de guider la sélection d'un type d'exigences. Par exemple, une exigence de type "Contrainte de Conception" ne pouvait être créée sous une exigence de type "Exigence de Performance". Toutefois, ces possiblités étaient trop peu développées et aucune gestion des types n'était offerte aux utilisateurs. La création de nouveaux types, la modification ou la suppression de ceux-ci n'étaient pas permises.

Les types avaient été créés principalement dans le but de permettre certaines vérifications automatiques. Par exemple, vérifier que toute fonction possède au moins un intrant et un extrant. Les problèmes de structuration évoqués précédemment étaient donc plus difficiles à éviter car les utilisateurs ne disposaient pas d'une structure de types d'exigences suffisamment complète avec un bon niveau de granularité.

La figure (Fig. 5.1) nous montre que lors d'un ajout d'exigence sous une exigence de type "Exigences de Performance", les seuls types disponibles sont "Détail" et "Exigences de Performance". Par exemple, pour créer une subdivision entre les exigences de performance statiques et dynamiques, il faut créer deux exigences ayant pour type "Exigences de Performance".

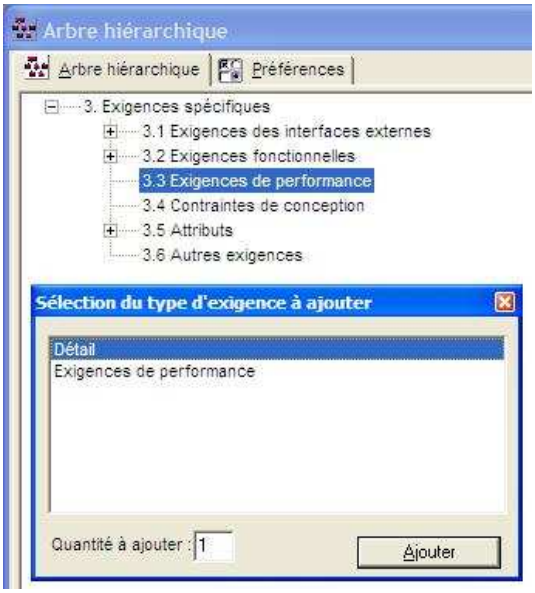

FIG. 5.1 - Types disponibles lors de l'ajout d'une exigence

Aucune génération automatique d'énoncé par défaut

Comme il a été dit ci-dessus, les types d'exigences servaient à la réalisation d'un ensemble de vérifications automatiques. La sélection d'un type lors de la création d'une exigence permettait de générer un début de phrase par défaut, associé au type, mais celui-ci était trop peu développé. Le début de phrase le plus courant étant "Le produit doit". Tout le reste de l'énoncé de l'exigence devant être entré manuellement par l'utilisateur. De ce fait, les problèmes d'uniformisation expliqués précédemment (cf section 5.1.4) étaient plus difficilement gérables. En effet, les utilisateurs de *GenSpec* ne connaissent généralement pas l'ensemble des règles de rédaction des exigences.

5.2 Rédaction des exigences d'interface

Différents problèmes peuvent surgir lors de la rédaction et de la gestion des exigences d'interface, nous nous sommes penchés sur trois d'entre eux :

- Manque de cohérence,
- Manque de complétude,
- Manque de traçabilité.

Ces problèmes ne sont pas les seuls problèmes auxquels nous nous sommes intéressés au niveau de la rédaction des exigences d'interface. En effet, certains problèmes liés à l'outil *GenSpec* devaient être résolus afin d'améliorer le processus de rédaction des exigences d'interface. Ceux-ci sont exposés dans la sous-section 5.2.4.

5.2.1 Manque de cohérence

Lorsque les exigences système et les exigences d'interface sont spécifiées dans des documents distincts, la SES et la SEI, certains problèmes de cohérence peuvent apparaître. En effet, la SEI est réalisée sur base des exigences spécifiques d'interface reprises dans la SES. Ces deux documents doivent donc être cohérents car ils sont présentés au client afin de s'assurer de la compréhension des besoins émis par ce dernier. Ces documents sont également fournis aux développeurs qui doivent les comprendre et les utiliser afin d'implémenter les besoins qui y sont décrits. Un problème de cohérence entre les deux documents d'exigences

peut nécessiter leur correction si cette incohérence est détectée, ou conduire à une implémentation contraire à ce que le client désire si celle-ci n'est pas détectée. De plus, si des éléments sont mis à jour au sein de la SES, il faut s'assurer que la SEI ne nécessite pas, elle aussi, d'être mise à jour afin d'éviter toute incohérence. Les problèmes de cohérence peuvent donc surgir lors de la rédaction proprement dite des documents mais également lors des mises à jour de ceux-ci.

Par exemple, une exigence de la SES concernant une interface est mise à jour et l'exigence correspondante dans la SEI de cette interface n'est pas modifiée. Si les développeurs découvrent que l'exigence correspondante dans la SES a été mise à jour, ils perdront du temps en contactant les personnes ayant réalisé les documents d'exigences afin de s'assurer que l'exigence de la SEI soit mise à jour. Si les développeurs ne se rendent pas compte que l'exigence correspondante dans la SES a été mise à jour, il est possible que cette exigence soit implémentée de manière incorrecte par rapport à la SES.

5.2.2 Manque de complétude

Tout comme pour le problème précédent, le fait que la SES et la SEI soient deux documents distincts peut poser problème, cette fois, au niveau de la complétude de la SEI. Lorsque la SEI est rédigée sur base de la SES de manière non automatique, il faut veiller à ne rien oublier. En effet, à toute exigence spécifique d'interface de la SES (ex : exigence d'extrant) doit correspondre une exigence de la SEI (ex : exigence de format d'extrant). Les problèmes de complétude peuvent donc également surgir aussi bien au moment de la rédaction proprement dite des documents que lors des mises à jour de ceux-ci.

Par exemple, une exigence de la SES concernant une donnée d'une interface est ajoutée et aucune exigence n'est ajoutée dans la SEI. Les développeurs ne possèdent alors aucune information sur le format de cette donnée. Ils devront donc faire appel aux personnes chargées de réaliser les documents d'exigences afin que ces derniers soient mis à jour.

5.2.3 Manque de traçabilité

Un autre type de problème pouvant survenir lors de la rédaction des documents d'exigences est le manque de traçabilité, en particulier entre la SES et la SEI. Comme il a été dit précédemment, à chaque exigence spécifique d'interface de la SES correspond une exigence de la SEI. Puisque ces documents sont distincts, il faut créer des renvois entre la SES et la SEI. Cela permet au client et aux personnes travaillant sur le projet de pouvoir passer d'un document à l'autre plus rapidement. Toutefois, lors de la rédaction de ces documents, il n'est pas toujours aisé de maintenir ces liens.

5.2.4 Problèmes liés à l'outil

Les problèmes liés à l'outil de support *GenSpec* qui sont rencontrés dans le cadre de la rédaction de la SEI sont les suivants :

- Lacunes dans la gestion multi-documents,
- Manque de possibilités au niveau des références,
- Rédaction manuelle de certaines parties de la SEI.

Lacunes dans la gestion multi-documents

L'un des problèmes de *GenSpec* était qu'il ne permettait pas de créer, de manière claire et explicite, deux documents d'exigences dans un même projet. Il était possible de manipuler plusieurs documents dans un même projet mais les possibilités offertes à ce niveau là devaient être révisées. Si une première mouture de la SEI doit être générée sur base de la SES, il est préférable que celle-ci se trouve dans le même projet que la SES correspondante. Les risques d'incohérence, d'incomplétude et de mauvaise traçabilité peuvent d'ailleurs être plus facilement évités grâce à cela.

Problème au niveau des références

L'outil permettait déjà de lier les exigences d'un même document. Cependant, le fait que l'on puisse désormais regrouper plusieurs documents au sein d'un seul projet a engendré un besoin au niveau des possibilités offertes pour la gestion des références. En effet, il fallait permettre de créer des liens entre plusieurs documents et, pour cela, une référence vers le numéro de l'exigence voulue ne suffisait plus. Il faut tout d'abord identifier le document et, ensuite, l'exigence au sein de ce document. Sans cette possibilité, la traçabilité entre SES et SEI ne peut être assurée.

Rédaction manuelle de certaines parties de la SEI

Une SEI a pour but, entre autres, de décrire le format des données spécifiques au système et des données spécifiques à l'interface. Ces éléments apparaissaient sous la forme de tableaux, comme celui présenté ci-dessous (TAB. 5.1).

Numéro	Intrants/Extrants de la SES	Format	Exemple
0	*3.1.2. Interface avec TC*	-	-
...	...	...	...
100	*3.1.1. Interface avec l'utilisateur TDST*	-	-
101	*3.1.1.1. Intrants de la vue d'ensemble*	-	-
102	3.1.1.1.1. Rappel de la fenêtre de temps TDST	Bouton	Fig. 2
...	...	...	...
104	3.1.1.4.7. a) Délai de temporisation de la fenêtre de temps TDST	UN : minutes ; VD : 120 ; PR : s.o. ; RÉ : 1 ; PL : 0 à 240	Fig. 4
...	...	...	...

TAB. 5.1 - Exemple de tableau explicitant le format des données spécifiques au système

Le problème est que ces tableaux sont des pièces jointes liées à des exigences et ils sont créés de manière manuelle, en dehors de *GenSpec*, par la personne réalisant la SEI. Les exigences pouvant être modifiées en cours de projet, il est impossible, sans une grande perte de temps, de maintenir ces tableaux en ordre de la SES. A l'heure actuelle, les tableaux détaillés des données sont reclassés en ordre de la SES lorsque celle-ci est considérée comme définitive. Cette méthode représente une perte de temps et il faut remédier à cela. De plus, en cas d'oubli lors de la mise à jour des documents d'exigences, la traçabilité et la cohérence des exigences de la SES et de la SEI ne sont plus assurées.

5.3 Résumé

À l'issue de ce chapitre, les problèmes concernant la qualité des documents d'exigences et la rédaction des exigences d'interface ont été passés en revue ainsi que les problèmes liés à l'outil ayant un rapport avec ces deux aspects.

Les problèmes liés à la qualité des documents d'exigences sur lesquels nous nous sommes penchés sont les suivants :

- Redondance et incohérence des exigences,
- Manque de complétude,
- Manque de structuration,
- Manque d'uniformisation des exigences,
- Problèmes liés à l'outil
  - Manque de types d'exigences et absence de gestion des types,
  - Aucune génération automatique d'énoncé par défaut.

Les problèmes liés à la rédaction des exigences d'interface auxquels nous nous sommes intéressés sont les suivants :

- Manque de cohérence,
- Manque de complétude,
- Manque de traçabilité,
- Problèmes liés à l'outil
  - Lacunes dans la gestion multi-documents,
  - Manque de possibilités au niveau des références,
  - Rédaction manuelle de certaines parties de la SEI.

Les améliorations proposées du logiciel *GenSpec*, présentées au chapitre 8, ont pour but d'apporter une solution à ces différents problèmes.

Chapitre 6

État de l'art

Dans ce chapitre, nous allons présenter un état de l'art relatif aux deux fonctionnalités à implémenter, à savoir l'ajout de types d'exigences et d'énoncés par défaut d'une part et, d'autre part, la génération de la SEI sur base de la SES. Pour chacune d'entre elles, l'état de l'art est divisé en deux parties : état de l'art des OSIE et état de l'art de la littérature.

6.1 Ajout de types et d'énoncés par défaut

Cette analyse nous a permis de trouver quelques idées et de consolider notre point de vue sur la solution à proposer. Ci-dessous, nous reportons les résultats obtenus.

6.1.1 Etat de l'art des OSIE

Avant d'énoncer des solutions quant à la réalisation d'une fonctionnalité supplémentaire pour un outil, il est judicieux de se pencher sur les divers concurrents afin de récolter des idées de solutions.

Nous avons choisi d'utiliser les OSIE les plus renommés ainsi que ceux dont nous pouvions obtenir rapidement une version d'évaluation.

Cette section se base notamment sur l'étude INCOSE [INC 07].

RaQuest 2.4

RaQuest un outil de gestion des exigences proposé par la société Sparx System. La dernière version de ce produit fournit une nouvelle fonctionnalité pour configurer les types et le statut des exigences. Il est désormais possible d'ajouter de nouveaux types d'exigences (Fig. 6.1) et de créer des exigences utilisant ces nouveaux types (Fig. 6.2).

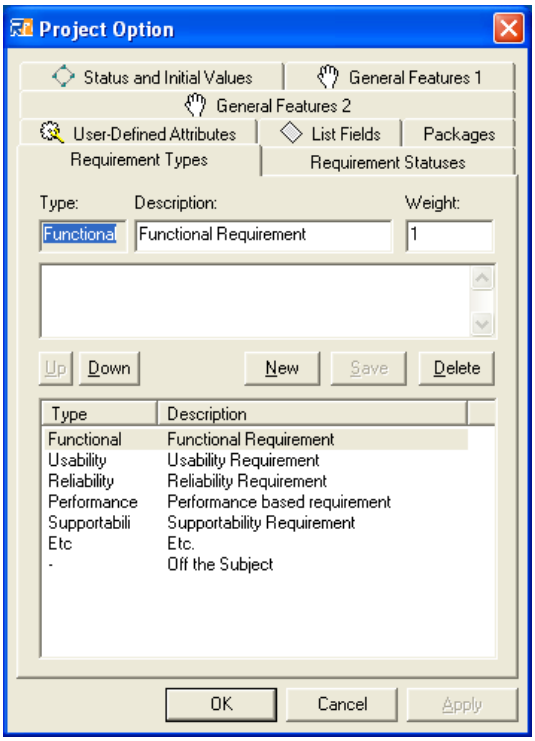

FIG. 6.1 - Ajout de nouveaux types d'exigences avec RaQuest

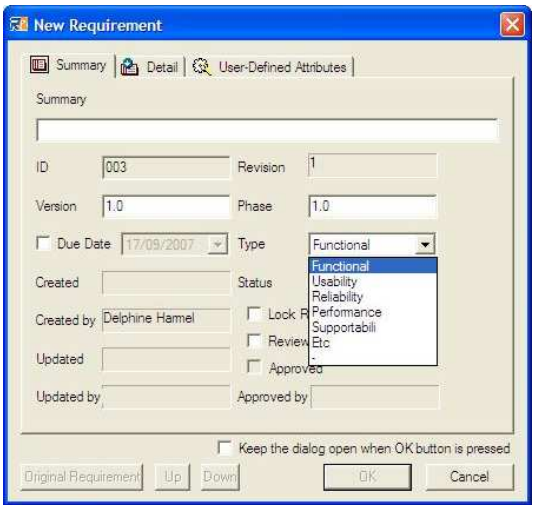

FIG. 6.2 - Création d'une exigence avec RaQuest

Les nouveaux types ajoutés sont ensuite disponibles lors de la création d'une nouvelle exigence. Cependant, cet outil ne fournit aucun énoncé par défaut, quel que soit le type d'exigences. Les types d'exigences par défaut sont les suivants : "functional", "usability", "reliability", "performance", "supportability", "etc" et "-".

Entreprise Architect 7.0

Le logiciel Entreprise Architect basé sur UML 2.1 est également un produit de la société Sparx System, qui se base sur l'outil précédemment présenté : RaQuest. Il dispose donc également de la possibilité de fournir une série de types d'exigences par défaut mais offre aussi l'opportunité de les modifier, de créer de nouveaux types propres à un projet, et enfin de gérer complètement la liste des types en fonction des besoins de l'organisation. Voici les

interfaces proposées pour cette fonctionnalité (adaptation des interfaces de RaQuest) (Fig. 6.3 et 6.4) :

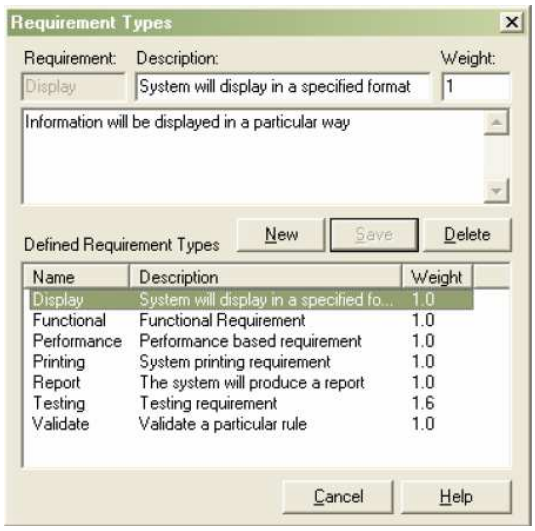

FIG. 6.3 - Ajout de nouveaux types d'exigences avec Entreprise Architect

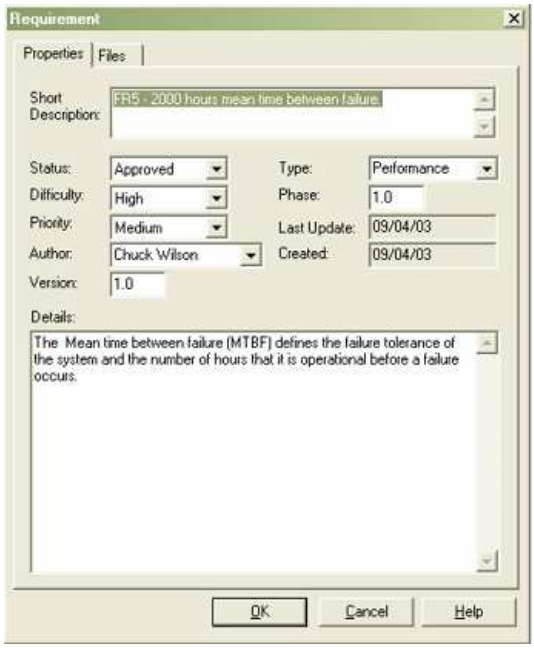

FIG. 6.4 - Création d'une exigence avec Entreprise Architect

Les types par défaut proposés sont les suivants : display, functionnal, performance, printing, report, testing et validate.

Ici non plus, il n'existe aucun énoncé par défaut pour ces types d'exigences. De plus, l'un des inconvénients est que la longueur du nom du nouveau type d'exigences est limité à douze caractères.

Requirements Management Database 2.5

Cet outil permet de créer des exigences, d'en fournir une description facilement rééditable mais aucun type d'exigences par défaut n'est attribué à l'exigence. Il n'existe pas non

plus d'énoncé par défaut associé à la création d'une nouvelle exigence. En conclusion, les possibilités sont assez limitées.

FIG. 6.5 - Assignation d'une exigence à un groupe avec RMD

FIG. 6.6 - Assignation d'une exigence à un sous-groupe avec RMD

Les deux images ci-dessus illustrent la fonctionnalité de "création d'une exigence". Il s'agit tout d'abord d'assigner l'exigence à un "groupe" : hardware, software ou system integration (Fig. 6.5), puis à un sous-groupe de ce dernier (Fig. 6.6). On peut ensuite entrer le nom d'une exigence et les divers autres champs, ainsi qu'une description facilement modifiable de l'exigence. Aucun énoncé par défaut n'est cependant généré quel que soit le type d'exigences.

TopTeamAnalyst 2.0.8

TopTeamAnalyst est développé par la société TechnoSolutions. La création d'un document d'exigences se fait par l'insertion directe de nouvelles exigences dans ce document. La création d'une exigence consiste en l'ajout d'un nom d'exigence, auquel est automatiquement associé un identifiant. Il s'agit ensuite d'ajouter une description à l'exigence créée. Il n'y a donc aucun type d'exigences, et encore moins d'énoncé par défaut.

Cradle REQ 5.6

Cradle REQ est un outil de gestion d'exigences développés par 3SL. Il fournit la possibilité de capturer les exigences dans des documents (Excel ou Word). Ces exigences, capturées ou directement créées, seront tout d'abord réparties dans des groupes non extensibles (business requirements, operating requirements, user requirements, system requirements) qui seront eux-mêmes subdivisés en types comprenant les exigences fonctionnelles, de performance, environnementales, commerciales, etc. Cet outil ne fournit cependant pas d'énoncés par défaut associés aux différents groupes et sous-groupes.

Telelogic Doors 8.1

L'outil Telelogic Doors se distingue sur l'axe "Faciliter la saisie des exigences" puisqu'il permet de récupérer des exigences dans des documents de divers formats tels que MSWord et PostScript. Ces fonctions facilitent la création et l'intégration d'exigences préexistantes. Cependant, comme le montre la figure 6.7, la création d'une exigence ne demande aucun type d'exigences.

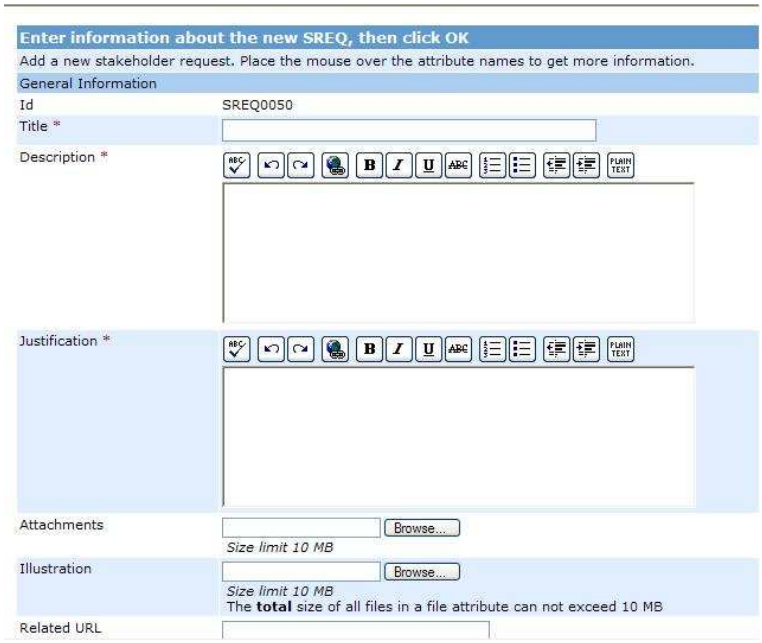

FIG. 6.7 - Création d'une exigence avec DOORS

Aucun énoncé par défaut n'est donc associé à un type d'exigences.

RTM Workshop

RTM Workshop est adapté pour l'importation de documents existants. Le produit dispose d'un assistant qui permet de choisir si les exigences sont à créer, à ajouter aux exigences existantes ou plutôt destinées à les remplacer. Un fichier de log liste toutes les actions et évènements associés. Ceci permet de contrôler la cohérence de l'importation des exigences. Il n'y a aucun énoncé par défaut associé au type d'exigences et la création de nouveaux types d'exigences ne semble pas permise.

Requisite Pro 7.0.1

Rational Requisite Pro est un outil proposé par IBM. Quelques types d'exigences sont proposés tels que "Feature", "Default for documents without requirements", "Software Requirement", "Stakeholder Request" et "Glossary Item" mais il n'y a pas d'énoncé par défaut associé à la création de l'exigence. Une des options intéressantes est de pouvoir à tout moment modifier le niveau de hiérarchie de l'exigence et son type (le changer en l'un des autres types proposés). La figure 6.8 fournit une illustration de l'ajout d'une exigence avec RequisitePro.

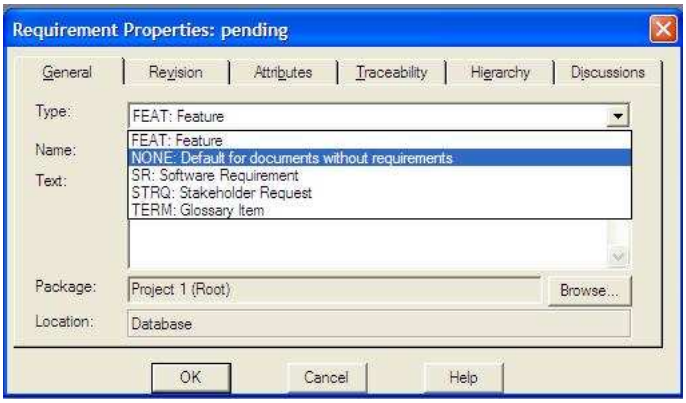

FIG. 6.8 - Ajout d'une exigence avec RequisitePro

Wibni 1.3.1

Wibni est un outil d'aide à la gestion de projet. Il s'agit d'un logiciel libre basé sur un fichier Excel et sur une BD. Les exigences peuvent être entrées au moyen d'une interface qui offre la possibilité de choisir le type de l'exigence (functionnal, performance, utilisability, etc.) mais aucun énoncé par défaut n'est généré suite à la sélection de celui-ci.

6.1.2 Etat de l'art de la littérature

La deuxième partie concerne l'état de l'art de littérature. Nous avons procédé en plusieurs étapes :

- Récolte de documents d'exigences,
- Réalisation d'un arbre des types,
- Récolte d'un ensemble de règles,
- Recherche d'énoncés et amélioration de l'arbre des types.

L'annexe J regroupe l'ensemble des recherches effectuées dans le cadre de l'état de l'art de la littérature.

Récolte de documents d'exigences

Afin de guider notre recherche de types et d'énoncés par défaut, nous avons récolté un bon nombre de documents d'exigences. En effet, nous avons eu l'occasion de parcourir plus de septante documents de ce type afin d'en extraire un maximum d'information (les références de ces divers documents peuvent être trouvées dans l'annexe J). Nous avons tout d'abord ciblé nos recherches sur les gabarits de SES et de SEI mais nous n'avons pu en trouver que très peu. C'est pourquoi nous avons ensuite collecté des SES et des SEI réalisées par diverses entreprises comme la NASA, HQ, etc. Ces documents récoltés pouvaient différer dans leur structure, leur qualité, etc. mais on a pu observer des structures ainsi que des types d'exigences récurrents. Cette première collecte nous a donc permis de nous faire une première idée concernant la structure à suivre et les types devant être inclus.

Réalisation d'un arbre des types

La première chose que nous avions à réaliser, lorsque la recherche de documents fut terminée, consistait à trouver les types d'exigences les plus pertinents, les plus utilisés. Il avait été décidé avec notre superviseur, monsieur René Bujold, que nous mettrions l'accent sur les types d'exigences non fonctionnels et hors interfaces. Comme il a été dit précédemment, les documents récoltés différaient au niveau de la structure. Certains documents étaient organisés avec plusieurs niveaux de titre et une manière de regrouper les exigences bien précise.

D'autres, par contre, possédaient une structure très primaire, plusieurs exigences étant regroupées sous des titres plus généraux, avec moins de précision. Toutefois, à la lecture d'un énoncé, il est tout à fait possible de déterminer un type et ces différences de structure n'ont finalement pas été néfastes à notre recherche.

L'arbre de types, que nous avons conçu, a comme noeud racine "Document". En-dessous, nous retrouvons le type "Exigences Spécifiques", cela étant en accord avec l'arborescence présente au sein de *GenSpec*. Ensuite, nous avons choisi les types de plus haut niveau et se retrouvant dans la quasi-totalité des documents, comme les exigences de performance, les contraintes de conception,... Ces types d'exigences servent en fait à réaliser des regroupements d'exigences. De plus, il nous faut préciser que nous avions convenu avec monsieur René Bujold que toutes les feuilles de l'arbre se trouveraient à un même niveau, à savoir dans notre cas, le niveau 6. Afin d'atteindre un tel niveau de granularité, il nous fallu trouver des types de niveau intermédiaire pour obtenir un arbre complet et bien équilibré. Pour cela, nous avons parcouru l'entièreté des documents, et réalisé un tableau reprenant l'ensemble des types pertinents observés. Par pertinent, nous entendons les types qui sont assez généraux et ne dépendent pas d'un projet précis. En effet, nous avons rencontré certains types d'exigences qui étaient inutilisables dans d'autres domaines que celui dans lequel ils étaient employés. Plusieurs types étaient fréquemment repris dans les divers documents d'exigences tels que temps d'installation, ressource mémoire nécessaire, ... Ces derniers sont dès lors présents dans notre arbre des types.

L'exemple d'arbre de types, provenant de l'arbre que nous avons réalisé, apparaît à la figure 6.9 :

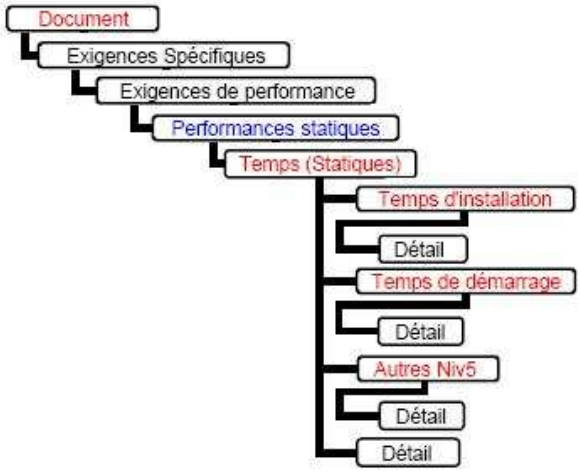

FIG. 6.9 - Exemple d'arbre des types

Cet arbre constitue la structure possible d'un document. Lorsqu'un type d'exigences est le fils d'un autre dans l'arbre, cela veut dire qu'il est possible de créer une exigence ayant ce type sous une exigence ayant le type père. Si on prend l'exemple proposé ci-dessus (Fig. 6.9), si un document comporte une exigence de type "Temps (Statiques)", on ne pourra créer sous celle-ci que des exigences de type "Temps d'installation", "Temps de démarrage", "Autres Niv5" ou "Détail". Cette structure a pour but de s'assurer que l'utilisateur ne regroupera pas sous un seul type d'exigences des exigences relatives à plusieurs sous-types. Lors de notre recherche de documents, nous avons observé un document où le type supérieur était "Contraintes" et sous ce type, on décrivait toutes les contraintes possibles et imaginables pour le système à réaliser. Il aurait cependant été préférable de définir des sous-types tels que "Contraintes de conception", "Contraintes légales", etc. afin de structurer les exigences. Une exigence ne peut donc être répertoriée que sous un seul type.

Récolte d'un ensemble de règles

Après avoir produit une première mouture de l'arbre des types (cf Annexe J), nous devions trouver des énoncés pour chacun d'eux mais, pour cela, il était important de trouver un ensemble de règles afin de formuler les énoncés. Nous avons tout d'abord formulé des règles générales à respecter lors de l'élaboration des énoncés. Ensuite, nous avons traduit les 27 règles émises par Berry, Tjong et Hartley dans leur article "Extended Disambiguation Rules for Requirements Specifications" [Berry 07]. Toutes ces règles nous ont permis d'envisager l'étape suivante de notre travail. Ci-dessous, un exemple des règles suivies pour l'élaboration de nos énoncés :

> "*Soit S, l'énoncé de l'exigence.*
> ***Règle 1 :*** *S devrait être écrit comme une phrase déclarative affirmative simple qui a seulement un verbe principal.*
> ***Règle 2*** *: Évitez d'écrire S à la voix passive, surtout si aucun personnage dynamique de l'action n'est spécifié. Au lieu de cela écrivez S à la voix active, avec le personnage dynamique de l'action comme sujet de S*
> ***Règle 3*** *: Évitez d'écrire S sous la forme "Il y a X dans Y", ou "X existe dans Y". Au lieu de cela écrivez qu' "Y a X" (verbe avoir). "*

Nous avons donc veillé à respecter ces règles lors de l'élaboration de nos énoncés.

Recherche d'énoncés et amélioration de l'arbre des types

Tout comme pour l'élaboration de l'arbre des types, nous nous sommes plongés dans les documents récoltés afin de regrouper tous les énoncés présents pour un même type (Fig. 6.10). Le tableau réalisé comporte 4 colonnes. La première colonne reprend la hiérarchie du type, la deuxième est le type d'exigences, la troisième est l'énoncé trouvé (traduit en français s'il était en anglais) et enfin la référence où l'énoncé a été trouvé. Toutes les références ont été définies au préalable et se sont vues asssignées un numéro [N°]. Ce numéro correspond à un numéro de référence dans le document de recherche disponible dans l'annexe J.

Hiérarchie	*Type*	*Enoncé(s)*	*Référence*
Exigences spécifiques Exigences de performance Performances statiques Temps (Statiques)	Temps d'installation	Pour faciliter l'installation, $ doit s'installer en moins de 20 minutes avec des exigences matérielles minimum. For ease of installation, $ must install in under 20 minutes on the minimum hardware requirements.	[37]
	Temps de démarrage	$ should start within 30 seconds. $ doit démarrer dans les 30 secondes.	[45]

FIG. 6.10 - Exemple de collecte d'énoncé pour un type d'exigences

Par après, nous avons réalisé une synthèse de tous ces énoncés afin de trouver le meilleur énoncé possible en respectant les règles qui avaient été définies préalablement. Lors de la formulation, nous nous sommes principalement focalisés sur les énoncés pour les types du plus bas niveau. En effet, les types de niveau supérieur nécessitent un énoncé mais sont surtout utilisés afin de réaliser des regroupements. Tandis que les énoncés pour les types du plus bas niveau doivent être plus précis et permettre de formuler, de manière tout à fait compréhensible, les exigences du client. Lors de cette recherche, certains types ont été ajoutés ou supprimés afin d'obtenir un arbre des types plus complet et mieux équilibré. Au final, nous avons obtenu un arbre de 203 types d'exigences et autant d'énoncés par défaut.

Il nous faut signaler quelques observations. Tout d'abord, beaucoup de documents de spécification utilisent les types "Intrant" et "Extrant", avec les énoncés respectifs suivants : "Le système doit recevoir ..." et "Le système doit fournir ...". Deuxièmement, la majorité des documents d'exigences utilisent des exigences simples, c'est-à-dire avec une seule phrase, et

qui sont hiérarchisées. Un exemple est fournit par [TCS 99] :

> *"3.2 System Capability Requirements*
>
> *3.2.6 AV Maintenance Function*
>
> *TCS shall be capable of executing AV maintenance software and displaying appropriate status results."*

Le contre-exemple suivant, tiré de [Tagger 05], montre une série d'exigences reprises sous le même type d'exigences et qui devraient être distinguées sous différents types d'exigences comme "Temps de démarrage", "Délai d'interaction" et "Temps de réponse".

> *"3.2 Performance Requirements*
>
> *This section aims to specify some of the timing and performance constraints to which the product will aim to adhere.*
>
> – *The program should start within 30 seconds. This depends on the number of data points that are to be plotted.*
>
> – *The interaction with the data points should have a delay of no longer than 2 seconds.*
>
> – *The response to a change in the orientation should be fast enough to avoid interrupting the user's flow of thought."*

6.1.3 Conclusion de l'analyse

Outils	Types d'exigence	Enoncés par défaut	Gestion des types
Raquest 2.4	Oui	Non	Oui
Entreprise Architect 7.0	Oui	Non	Oui
RMD 2.5	Non	Non	Non
TopTeamAnalyst 2.0. 8	Non	Non	Non
Cradle REQ 5.6	Oui	Non	Non
Telelogic Doors 8.1	Non	Non	Non
RTM Workshop	Oui/Non	Non	Non
Requisite Pro 7.0.1	Oui	Non	Non
Anvil Designer 1.4.4.0	Non	Non	Non
Wibni 1.3.1	Oui	Non	Non
BorlandCaliber	-	-	-
IRqA	-	-	-

TAB. 6.1 - Résultats de l'analyse des OSIE

Suite à l'analyse de divers OSIE et d'après le tableau (TAB. 6.1), nous pouvons tirer les conclusions suivantes :

– La moitié des outils analysés utilisent des types d'exigences mais aucun d'entre eux n'y associe des énoncés par défaut,

– Certains outils (RaQuest et Entreprise Architect) permettent d'ajouter de nouveaux types d'exigences et de les gérer via un module de gestion des types.

Ce premier état de l'art nous a donc permis d'observer ce qu'il en était pour les autres outils actuellement sur le marché. Il faut cependant noter que certains outils n'ont pu être évalués en profondeur (Cradle et Telelogic Doors), voire pas du tout évalués (BorlandCaliber et IRqA) car aucune version d'évaluation n'était disponible ou ne nous a été accordée.

Les outils analysés sont assez variés, certains sont très renommés, d'autres moins, et un cas de logiciel libre (Wibni 1.3.1) a également été observé.

En ce qui concerne l'état de l'art de la littérature, il nous a permis d'analyser les structures de documents d'exigences existantes et les types les plus fréquemment utilisés. Il a servi de base lors de l'élaboration de notre arbre des types et des énoncés par défaut. Comme nous l'avons déjà dit précédemment, la synthèse de nos recherches a donné lieu à un arbre de 203 types d'exigences et autant d'énoncés par défaut associés (cf Annexe J). Comme il l'a dit dans la section 5.1.5, seulement une trentaine de types étaient disponibles dans *GenSpec* avant notre recherche.

6.2 Génération de la SEI sur base de la SES

Cette section présente l'ensemble des informations qui ont pu être collectées par l'analyse de divers OSIE et de documents récoltés dans la littérature ou au sein de l'unité CA.

6.2.1 État de l'art des OSIE

Comme pour la première fonctionnalité, nous avons observé une dizaine d'OSIE. Plusieurs d'entre eux offrent des fonctionnalités permettant la génération de documents d'exigences : Telelogic Doors, Requisite Pro, Top Team Analyst, etc. Généralement, il s'agit de la génération de la SES sous un format pdf, doc ou encore rtf. Cette génération utilise un gabarit permettant d'organiser les informations de manière claire et certains outils offrent même la possibilité de créer un tel gabarit (Enterprise Architect, par exemple). Ce genre de fonctionnalités est déjà présent dans l'outil *GenSpec* et il est à constater qu'aucun des outils étudiés ne permet de générer la SEI ou une partie de celle-ci, sur base de la SES.

La fonctionnalité de génération à apporter à *GenSpec* permettrait de générer la SEI sur base de la SES à l'écran, dans un premier temps, et ensuite de générer celle-ci dans un document.

La génération de la SEI sur base de la SES est une fonctionnalité assez spécifique à l'unité CA d'HQ et c'est pourquoi l'état de l'art des OSIE n'a rien apporté d'intéressant. Nous avons tout de même réalisé une analyse des documents disponibles dans la littérature afin de rechercher d'éventuelles études, conférences ou autres ayant pour objet la génération de la SEI sur base de la SES.

6.2.2 Analyse de documents concernant la génération de la SEI sur base de la SES

Afin de récolter un maximum de documents, Internet semble être l'outil le plus adéquat. Nous avons dès lors utilisé des moteurs de recherches afin de trouver des documents relatant d'une génération d'une SEI sur base de la SES. Bien évidemment, nous avons entré différents termes autant en français qu'en anglais afin de maximiser les chances de trouver des documents. Voici quelques exemples de mots-clés utilisés : "SES", "SEI", "SRS", "IRS", "System Requirement Specification", "génération IRS", "Specification",... Toutefois, la recherche de documents concernant la génération de la SEI sur base de la SES fut aussi infructueuse que l'état de l'art des OSIE. Plusieurs moteurs de recherche ont pourtant été utilisés et des sites spécialisés (IEEE, CAiSe, ACM, etc.) ont été consultés.

La recherche via les moteurs de recherche est probablement restée infructueuse car nous étions limités dans le temps. De plus, certains articles peuvent discuter d'une telle fonctionnalité mais ne pas être centrés sur celle-ci et nous n'avons donc pas pu les retrouver via nos recherches.

6.2.3 Analyse de documents concernant le format d'une SEI

Les recherches précédentes ayant été soldées par des échecs, une recherche de documents décrivant le format d'une SEI a été envisagée. En effet, via l'élaboration d'une structure adéquate, il serait possible d'agencer la SEI au niveau du programme en s'inspirant d'une partie de celle-ci.

Norme MIL-STD-498 (IRS) [U.S 95]

Le premier gabarit qui a été trouvé est le MIL-STD-498 (IRS) qui est une norme de l'armée américaine. Cette norme militaire peut paraitre obsolète, puisqu'elle date de 1995, mais elle constitue une base de connaissances intéressante. Voici la structure de cette norme :

1. Scope :
   1.1 Identification
   1.2 System overview
   1.3 Document overview
2. Referenced documents
3. Requirement :
   3.1 Interface identification and diagrams
   3.2 Project-unique identifier of interface
   ...
   3.x Project-unique identifier of interface
   3.y Precedence and criticality of requirements
4. Qualification provisions
5. Requirements traceability
6. Notes A
7. Appendixes

Cette norme possède une structure similaire à celle des standards suivants :

- IEEE Recommended Practice for Software Requirements Specifications (IEEE std 830-1998)
- British Standard Guide to Specifying User Requirements for a Computer-Based Standard (BS6719 - 1986)
- Canadian Standard, Basic Guidelines for the Structure of Documentation of System Design Information (Z242.15.4-1979)

Ce qui est plus particulièrement intéressant dans ce standard concerne la manière dont sont décrites les exigences de chaque interface (les points 3.2 à 3.x).

Chaque interface est décrite en sept sous-points :

- La priorité de chaque entité d'interface,
- Les exigences sur le type d'interface à implémenter,
- Les caractéristiques des données individuelles,
- Les caractéristiques des données assemblées,
- Les caractéristiques des méthodes de communication,
- Les caractéristiques des protocoles,

– Tous les autres types de caractéristiques.

Chacun de ces sous-points est lui-même détaillé en différents éléments tels que : noms/identifiants, type de données, taille/format, unité, précision, etc.

Grâce à ces détails, il est possible d'identifier les différentes caractéristiques qui peuvent être demandées à l'utilisateur pour chaque exigence relative à la SEI afin de fournir un document le plus complet et le plus précis possible.

Gabarit SEI d'un projet de l'unité CA : TDST [Uni 04]

Plusieurs projets sont en cours dans l'unité CA dont le projet concernant le Télé-Délestage en Sous-Tension (TDST). Ce projet a pour objectif de prévenir l'effondrement du réseau électrique du Québec par la détection d'une sous-tension sur le réseau autour de Montréal. Afin de mieux cerner le processus de création d'une SEI, un exemplaire de la SEI de ce projet nous a été fourni.

1 Introduction :
  1.1 Objet
  1.2 Portée
  1.3 Définitions, acronymes et abréviations
  1.4 Références
  1.5 Vue d'ensemble
2 Description générale :
  2.1 Format des données
  2.2 Synchronisation des échanges
  2.3 Caractéristiques des utilisateurs
  2.4 Hypothèses et dépendances
3 Exigences spécifiques :
  3.1 Exigences de format
    3.1.1 Généralités
    3.1.2 Données de l'utilisateur TR
  3.2 Exigences de synchronisation
    3.2.1 Généralités
    3.2.2 Commandes spécifiques à l'interface
    3.2.3 Signalisations spécifiques à l'interface
    3.2.4 Gestion des droits d'accès

Annexe A : Intrants / Extrants en ordre de la SES
Annexe B : Arbre des exigences

Cependant, les SEI réalisées pour l'instant dans l'unité ne sont pas toutes structurées de la même manière. Leur format est encore à standardiser.

Il est intéressant dans ce document de se pencher sur les exigences de format, au niveau des données spécifiques de l'utilisateur (point 2.3 dans la structure ci-dessus). Dans cette partie, on retrouve les illustrations des différentes interfaces dont les données sont ensuite détaillées dans un tableau (TAB. 6.2) à quatre colonnes : numéro, intrants/extrants de la SES, format et exemple.

Détails des données			
♯	Intrants/Extrants de la SES	Format	Exemple
0	3.1.2. Interface avec TC	-	-
...	...	...	...
100	3.1.1. Interface avec l'utilisateur TDST	-	-
101	3.1.1.1. Intrants de la vue d'ensemble	-	-
102	3.1.1.1.1. Rappel de la fenêtre de temps TDST	Bouton	Fig. 2
...	...	...	...
104	3.1.1.4.7. a) Délai de temporisation de la fenêtre de temps TDST	UN : minutes ; VD : 120 ; PR : s.o. ; RÉ : 1 ; PL : 0 à 240	Fig. 4
...	...	...	...

TAB. 6.2 - Exemple des détails des données [Uni 04]

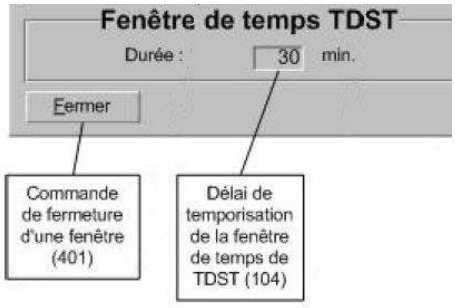

FIG. 6.11 - Exemple de prototype d'interface [Uni 04]

Le numéro (première colonne) correspond au nombre se trouvant dans le prototype d'interface (par exemple, le numéro 104 dans la figure 6.11). Les intrants/extrants de la SES (deuxième colonne) correspondent à la référence vers l'exigence de la SES. Le format (troisième colonne) reprend l'unité, la valeur par défaut, la précision, etc. L'exemple renvoit à la référence vers le prototype d'interface correspondant (par exemple, Fig.2 dans le cas du tableau 6.2).

Par après, nous avons les exigences de synchronisation qui sont spécifiques à l'interface et qui représentent les exigences entre deux systèmes et, enfin, nous avons les annexes du document qui sont composées du tableau des données dans l'ordre de la SES ainsi que de l'arbre hiérarchique des exigences.

Le plus important dans cet exemple est de constater la présence d'une référence vers l'exigence correspondante dans la SES. L'idéal est, en fait, de générer une SEI ayant la même structure que la SES. De ce fait, le lien existant entre ces deux types de spécification n'en sera que plus clair puisqu'il y a un même niveau de granularité dans l'arbre des exigences. Pour prendre un exemple, on sait qu'une exigence de niveau 4 dans la SES sera également une exigence de niveau 4 dans la SEI.

Gabarit SEI en construction de l'unité CA [Ratté 07]

Comme expliqué à la section 4.5.2, le processus de GL de l'unité fait l'objet d'une standardisation et les SEI ne font pas exception. Le gabarit de SEI a, par exemple, été modifié pour la dernière fois en juillet 2007. Voici un résumé de la table des matières de cette dernière version :

1. Introduction
   1.1 Objet
   1.2 Portée
   1.3 Définitions, acronymes et abréviations
   1.4 Références
   1.5 Vue d'ensemble
2. Description générale
   2.1 Mise en contexte
   2.2 But du l'interface externe
   2.3 Environnement
   2.4 Regroupement de traitements de données
   2.5 Exigences non fonctionnelles
   2.6 Hypothèses et dépendances
3. Exigences spécifiques
   3.1 Intrants - Extrants de la SES
   3.2 Intrants - Extrants spécifiques à l'interface externe
   3.3 Traitement des données
   3.4 Format et organisation des affichages
   3.5 Exigences non fonctionnelles
4. Protocole de communication

Dans ce gabarit, on retrouve une structure classique qui est inspirée de la norme IEEE-Std-830 [Sof 98b] qui est également utilisée afin de structurer les documents de la SES. Un tel gabarit est très complet, il permet de réaliser des documents comme celui présenté à la section précédente (cf section 6.2.3) avec certaines informations supplémentaires.

Au niveau des exigences spécifiques, la structure utilisée diffère également de celle de la norme MIL-STD-498 [U.S 95]. Toutefois, les détails demandés pour chaque intrant/extrant correspondent assez bien aux détails demandés pour chaque donnée dans la norme militaire : identifiant, but, provenance, unité, précision, etc.

6.2.4 Conclusion de l'analyse

Il n'existe actuellement aucun outil fournissant une génération automatique de la SEI sur base de la SES. Les seuls éléments intéressants ont pu êtres trouvés dans des gabarits de SEI existants. En effet, ceux-ci nous ont permis d'avoir une idée de la structuration qu'une SEI pourrait avoir.

De plus, nous n'avons trouvé que peu de SEI réalisées à l'aide des standards énoncés au chapitre 2, IEEE-Std-830 [Sof 98b] et Volere [Robertson 06]. HQ utilise le standard IEEE-Std-830 pour réaliser ses SEI mais nous n'avons trouvé aucune SEI réalisée selon le gabarit Volere.

Il semble que cette fonctionnalité soit trop spécifique à HQ, ce qui expliquerait le peu d'informations qui ont pu être trouvées à ce sujet.

6.3 Résumé

Dans ce chapitre, nous avons présenté un état de l'art pour chacune des fonctionnalités à réaliser :

– Concernant la première fonctionnalité, nous avons vu que certains OSIE utilisent des types d'exigences. Certains disposent même d'une gestion de ces types. Cependant, aucun d'entre eux ne fournit des énoncés par défaut clairs et structurés pour chaque type d'exigences. Ce premier état de l'art nous a donc permis de voir ce qu'il en

était actuellement sur le marché et nous a donné quelques pistes concernant les types d'exigences et la manière de les gérer. L'état de l'art de la littérature nous a permis d'observer les différentes structures et types d'exigences utilisés. Il a servi de base à l'élaboration de notre arbre des types et de nos énoncés par défaut.

– L'état de l'art de la deuxième fonctionnalité n'a, quant à lui, apporté que très peu d'informations concernant la génération proprement dite de la SEI sur base de la SES. Aucun OSIE ne dispose de cette option et très peu de documents trouvés parlent de cette possibilité. Comme nous l'avons dit précédemment, seule la recherche réalisée dans les gabarits de SEI existants nous a permis de trouver quelques éléments intéressants, notamment concernant la structuration d'une SEI.

Le chapitre suivant explique le mode de développement mis en oeuvre lors du stage pour implémenter les fonctionnalités. Il explique les différentes activités que nous avons réalisées et les documents qui ont été produits pour chacune d'entre elles.

Chapitre 7

Mode de développement

Ce chapitre a pour but de décrire le mode de développement que nous avons suivi afin d'améliorer l'outil *GenSpec*. Nous commençons par une description générale avant de proposer une description détaillée de celui-ci.

7.1 Description générale

7.1.1 Personnes impliquées

Les personnes concernées par le mode de développement sont les stagiaires et le comité *GenSpec*. Les stagiaires sont Delphine Harmel et Guillaume Cavillot. Le comité *GenSpec* est composé de Jean-Pierre Turgeon, Véronique Chu, Harold Ratté, Michel Vincelette, Luc Tétreault et René Bujold. Ce dernier étant le créateur et responsable principal du projet *GenSpec*, la majorité des activités ont été réalisées en étroite collaboration avec lui.

7.1.2 Étapes du mode de développement

Le schéma ci-dessous (Fig. 7.1) montre les différentes étapes du mode de développement que nous avons suivi.

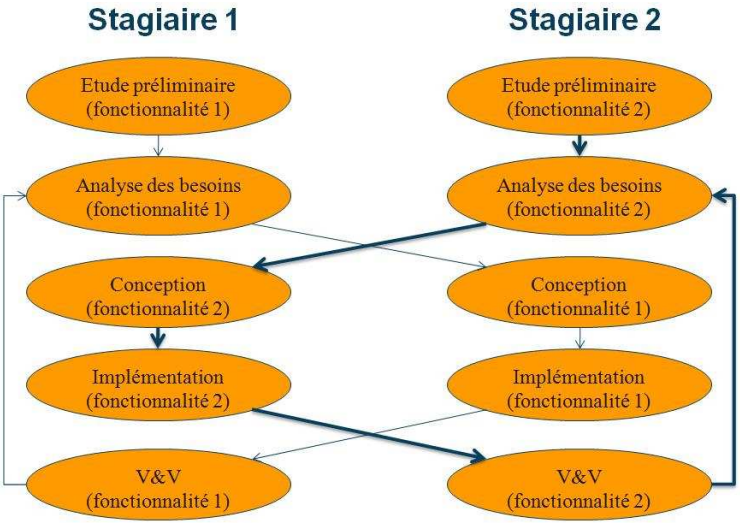

FIG. 7.1 - Schéma de développement général

Nous rappelons que nous avions deux fonctionnalités à apporter à l'outil (cf Chapitre 8) : l'ajout de types et d'énoncés par défaut (fonctionnalité 1) et la génération de la SEI sur base de la SES (fonctionnalité 2).

Sur le schéma les flèches en gras représentent le développement de la fonctionnalité 2 et les flèches normales représentent le développement de la fonctionnalité 1. Le stagiaire 1 est Delphine Harmel et le stagiaire 2, Guillaume Cavillot. Le planning a été réalisé par les deux stagiaires. Par la suite, nous avons travaillé de manière indépendante mais il est arrivé que nous nous aidions en cas de problème afin d'éviter tout retard. On constate également qu'après la phase d'analyse des besoins, un premier échange est réalisé ainsi qu'après la phase d'implémentation. Par exemple, Delphine Harmel a réalisé l'étude préliminaire et les activités d'analyse des besoins et de V&V de la fonctionnalité 1. Elle était également responsable des activités de conception et d'implémentation de la fonctionnalité 2. Cette manière de procéder avait été suggérée par les stagiaires de l'année précédente, Olivier Pire et Nicolas Pirmez [Pire 07].

Cette pratique possède plusieurs avantages :

- Nécessité d'avoir une **analyse des besoins de bonne qualité** afin que l'autre personne puisse la comprendre et l'implémenter.

- **Réduction du risque d'intégrer des solutions d'implémentation** dans l'analyse des besoins ou de négliger cette dernière.

- **Meilleure objectivité** au niveau des tests car la personne chargée des tests n'est pas celle qui réalise le codage d'une fonctionnalité.

Nous avons organisé notre développement de manière itérative et incrémentale :

- **Itérative** car les deux fonctionnalités, que sont l'ajout de types et d'énoncés par défaut et la génération automatique de la SEI sur base de la SES, ont été développées progressivement et améliorées à chaque itération.

- **Incrémentale** car certaines fonctionnalités secondaires ont été développées en entier lors de certains incréments.

Nous avons réalisé trois incréments/itérations pour la première fonctionnalité et cinq pour la seconde. Nous avons essayé de diviser le travail en incréments/itérations de même taille. Toutefois, le premier incrément/itération de la fonctionnalité 1 et le dernier de la fonctionnalité 2 étaient plus simples et nécessitaient donc moins de temps de travail que les autres. La description des différents incréments/itérations est donnée par la suite (cf section 7.2.3).

7.1.3 Répartition du temps

Afin de mener au mieux notre développement, nous avons essayé de respecter une répartition du temps de travail optimale. Pour cela, nous avons consulté les chiffres proposés par M. Zelkowitz [Zelkowitz 95]. Toutefois, celui-ci prenait en compte la maintenance dans sa répartition et nous avons dû l'adapter. En effet, aucune maintenance de l'outil *GenSpec* n'a été réalisée durant notre stage. Le développement de celui-ci étant réalisé par les stagiaires successifs de l'unité CA, une grande partie du temps de développement est consacrée au processus d'intégration et de tests. La maintenance est réalisée en grande partie par monsieur René Bujold.

Sur le schéma de la page suivante (Fig. 7.2), on constate que notre répartition (en orange) respecte assez bien celle proposée par M. Zelkowitz (en bleu).

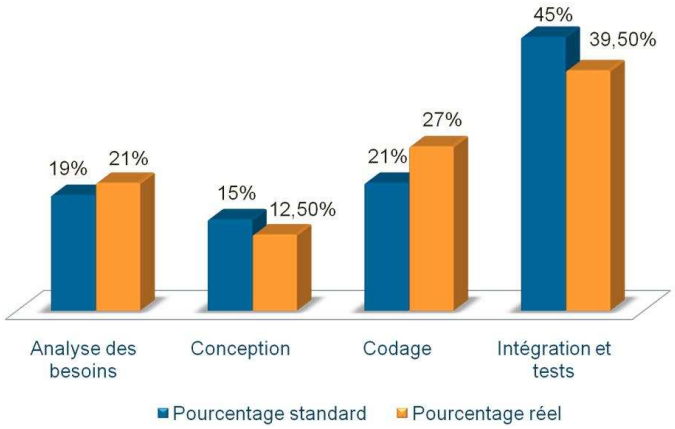

FIG. 7.2 - Répartition du temps de travail

7.2 Description détaillée

Le schéma de la page suivante (Fig. 7.3) montre le détail des différentes activités que nous avons réalisées lors du développement.

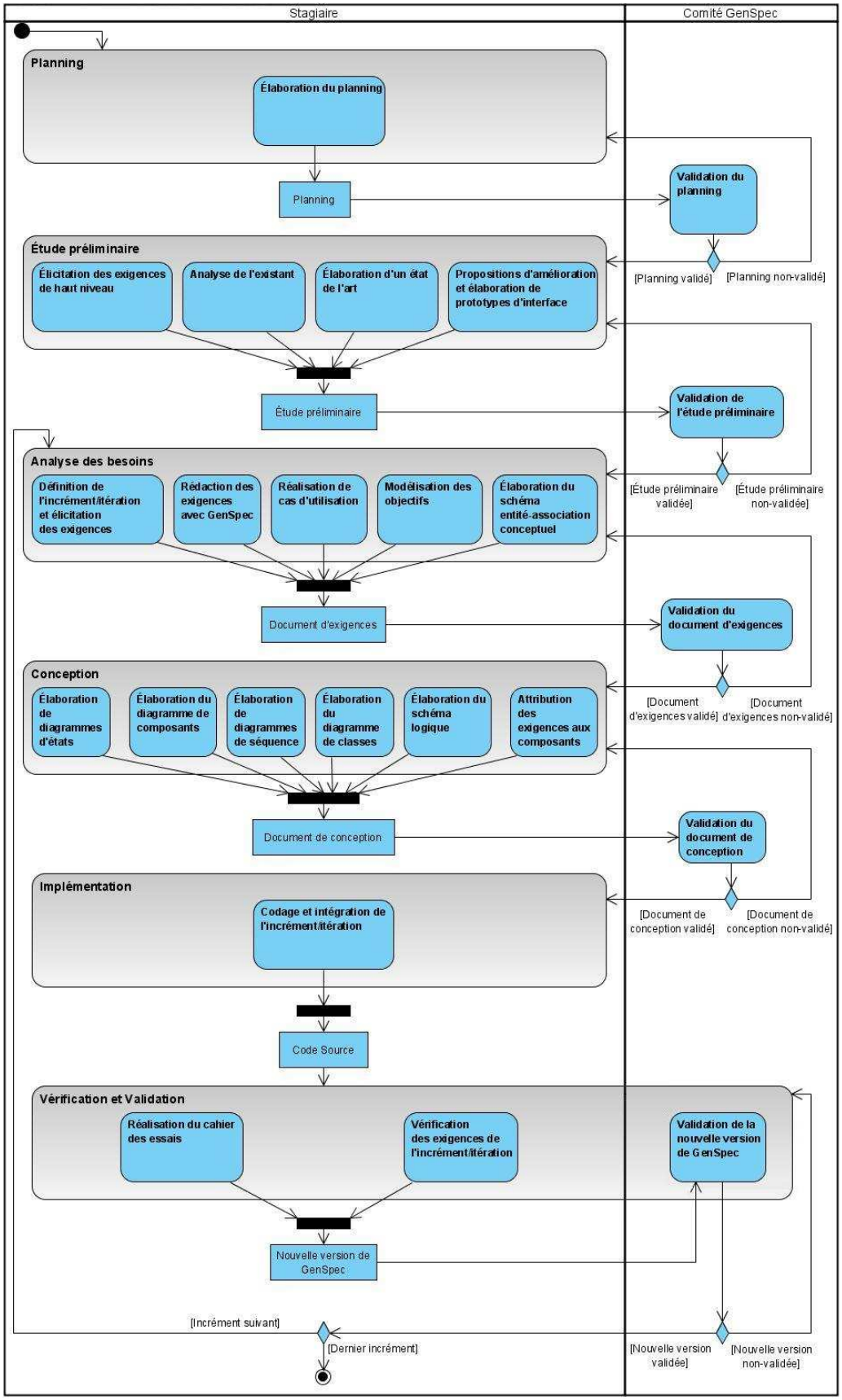

FIG. 7.3 - Schéma de développement détaillé

Les sous-sections suivantes présentent les différentes activités du schéma (Fig. 7.3) rassemblées selon les grandes étapes du développement que nous avons décrites précédemment (Fig. 7.1).

7.2.1 Planning

Élaboration du planning

L'élaboration du planning était la première étape à réaliser lors de notre développement. Cette activité a de nombreuses utilités :

- Définir l'ensemble des tâches à réaliser afin d'arriver au terme du projet,
- Déterminer les ressources (en particulier le temps) nécessaires à chacune de ces tâches,
- Déterminer la séquence d'exécution des tâches,
- Fixer les responsabilités de chacune des personnes impliquées dans le projet,
- Suivre le projet au jour le jour afin de s'assurer que celui-ci sera terminé dans les délais fixés.

Afin de réaliser cette activité, nous avons utilisé l'outil Microsoft Office Project qui permet de réaliser un planning très détaillé. De plus, cet outil offre de nombreuses fonctionnalités permettant de gérer un projet au jour le jour : gestion du temps de travail restant, complétion des tâches, etc.

7.2.2 Étude préliminaire

Lors du développement de l'outil *GenSpec*, il est toujours demandé aux stagiaires de l'unité CA d'HQ de réaliser une étude préliminaire. Cette étude se base sur le standard IEEE-Std-1362 de documents de conception opérationnel [Sof 98a]. Elle permet de déterminer les objectifs/la vision des clients, de comprendre le contexte dans lequel le projet est réalisé et d'envisager les solutions à implémenter afin de satisfaire les besoins des clients. Le but de celle-ci est de s'assurer de la compréhension commune des stagiaires et du comité *GenSpec* afin de démarrer le développement sur une bonne base. Le lecteur trouvera en annexe les deux études préliminaires correspondant aux deux fonctionnalités (Annexes B et C).

Élicitation des exigences de haut niveau

Afin de pouvoir débuter les études préliminaires, il nous fallait cerner les objectifs des différentes parties prenantes. C'est pourquoi plusieurs interviews et réunions ont été organisées avec le comité *GenSpec*, et plus particulièrement monsieur René Bujold.

Nous avons pu y dégager les exigences de haut niveau. Toutefois, les détails des différentes fonctionnalités n'ont pas été discutés à ce moment-là. En effet, comme nous devions proposer des améliorations, les exigences ont été volontairement énoncées à un haut niveau afin de pouvoir envisager toute possibilité de solution. Comme nous l'avons déjà dit, l'étude préliminaire sert uniquement de base au développement.

Analyse de l'existant

Les exigences de haut niveau ayant été élicitées, nous pouvions nous concentrer sur le contexte dans lequel nous allions devoir travailler. L'analyse de l'existant a permis de cerner le fonctionnement de l'outil *GenSpec*, la manière dont il était utilisé au sein de l'unité CA et les problèmes ou limitations de celui-ci. Avant d'envisager une solution à un problème, il est important de comprendre et maîtriser celui-ci.

Plusieurs possibilités étaient envisageables afin de réaliser cette analyse. Voici celles que nous avons utilisées :

- **La lecture de documents concernant l'outil**. En effet, différents articles et présentations ont été mis à notre disposition afin de nous donner une première idée de *GenSpec*.

- **La manipulation du logiciel *GenSpec*** nous a permis de comprendre les diverses fonctionnalités offertes par celui-ci mais également de mettre en évidence les limites de ce logiciel.

- **La lecture de documents** écrits par des membres de l'unité CA **décrivant le processus de développement logiciel**. Grâce à cela, il était possible de déterminer lors de quelles étapes le logiciel *GenSpec* était utilisé et quelles étaient les personnes qui l'utilisaient.

- **La lecture de documents produits via *GenSpec*** fut également très intéressante et nous a permis d'affiner notre compréhension du fonctionnement de *GenSpec*.

Élaboration d'un état de l'art

Nous connaissions les possibilités de l'outil via l'analyse de l'existant mais, afin de trouver les meilleures solutions possibles aux problèmes décrits dans le chapitre 5, nous devions réaliser un état de l'art pour trouver des pistes concernant l'ajout de types et d'énoncés par défaut et la génération de la SEI sur base de la SES.

Pour cela, nous avons travaillé de manière différente pour les deux fonctionnalités :

- Pour l'ajout de types et d'énoncés par défaut, nous avons **analysé plusieurs OSIE disponibles sur le marché**. Certains d'entre eux possédaient des fonctionnalités liées à notre recherche et cela nous a permis de paufiner nos idées de solution. Nous avons également **exploré plusieurs documents d'exigences** afin de récolter un ensemble de types et d'énoncés par défaut.

- Pour la génération automatique de la SEI sur base de la SES, ce fut plus délicat. En effet, aucun outil parmi ceux analysés ne possédait de fonctionnalité liée à notre recherche. Il faut savoir que l'utilisation d'un document tel que la SEI est spécifique à HQ, très peu d'entreprises utilisent ce type de document. De ce fait, pour approfondir notre état de l'art, nous avons **parcouru différentes SEI disponibles au public**. Grâce à cela, nous avons pu déterminer une bonne structure de SEI en analysant les avantages et inconvénients des différents documents que nous avons collectés.

Le chapitre 6 présente en détail l'état de l'art que nous avons réalisé.

Propositions d'amélioration et élaboration de prototypes d'interface

Pour terminer nos études préliminaires, nous avons proposé des ébauches de solutions accompagnées de prototypes d'interface. Ces propositions ont été élaborées sur base des informations récoltées dans les parties précédentes de chaque étude. Le but de celles-ci étant d'harmoniser les différents points de vue des parties prenantes.

Il faut savoir que les prototypes d'interface ont été présentés à plusieurs reprises et modifiés conformément aux remarques des parties prenantes. Les interfaces réellement implémentées s'inspirent fortement de ces prototypes. Toutefois, des différences sont à constater en raison d'améliorations apportées au cours du développement.

L'ensemble des prototypes sont disponibles dans les études en annexe (Annexes B et C). La figure de la page suivante (Fig. 7.4), présente un exemple de prototype d'interface réalisé dans le cadre de la fonctionnalité d'ajout de types et d'énoncés par défaut.

FIG. 7.4 - Prototype d'interface pour l'ajout de types et d'énoncés par défaut

7.2.3 Analyse des besoins

Après avoir réalisé les études préliminaires, nous pouvions débuter le processus de développement des deux fonctionnalités. La première étape de ce processus est l'analyse des besoins. Le lecteur trouvera en annexe les deux documents produits lors de l'analyse des besoins correspondant aux deux fonctionnalités (Annexes D et E).

Définition de l'incrément/itération et élicitation des exigences

Afin de développer les nouvelles fonctionnalités, nous avons procédé de manière évolutive en les découpant en incréments et itérations. Pour chaque incrément ou itération, une élicitation des exigences (cf section 2.1) a été réalisée afin de cibler les besoins à implémenter. A la fin de chacun de ces incréments ou itérations, une nouvelle version stable de *GenSpec* était produite.

Afin de réaliser l'élicitation des exigences à chaque incrément ou itération de chaque fonctionnalité, des réunions étaient organisées entre les parties prenantes afin de dégager leurs besoins. Les itérations et incréments étaient déterminés sur base de cette élicitation car elle permettait de dégager la priorité des exigences à satisfaire. Les exigences essentielles étaient dans les premiers incréments et itérations tandis que les exigences moins importantes ou optionnelles se trouvaient dans les derniers.

Voici les éléments principaux ayant été traités lors des différents incréments/itérations que nous avons réalisés pour la fonctionnalité 1 :

- Gestion de la génération d'un énoncé par défaut à la sélection d'un type et mise à jour de la fonctionnalité de vérification automatique des exigences,
- Ajout de types et d'énoncés par défaut,
- Création d'un module de gestion des types et énoncés par défaut.

Voici les éléments principaux ayant été traités lors des différents incréments/itérations que nous avons réalisés pour la fonctionnalité 2 :

- Gestion du niveau document et amélioration des fonctionnalités de génération de documents Word,
- Gestion de la création d'un nouveau document au sein d'un projet *GenSpec* à l'aide d'un modèle,
- Création d'un module de gestion des références,
- Gestion de la génération d'une SEI,
- Mise à jour de la fonctionnalité de vérification automatique des exigences.

Modélisation des objectifs

La modélisation des objectifs permet de mettre en évidence les différents objectifs d'un projet. Ces objectifs sont regroupés par acteurs et peuvent être aussi bien fonctionnels que non-fonctionnels. Cette modélisation permet également de mettre en évidence certaines relations existantes entre les acteurs. Les modèles d'objectifs sont réalisés sur base de l'élicitation des exigences et offrent un support pour la rédaction des exigences.

Pour plus d'information sur la modélisation i*, veuillez consulter [Yu 05]. Toutefois, afin de mieux comprendre les notations des modèles i*, voici la légende (Fig. 7.5) proposée par cette source :

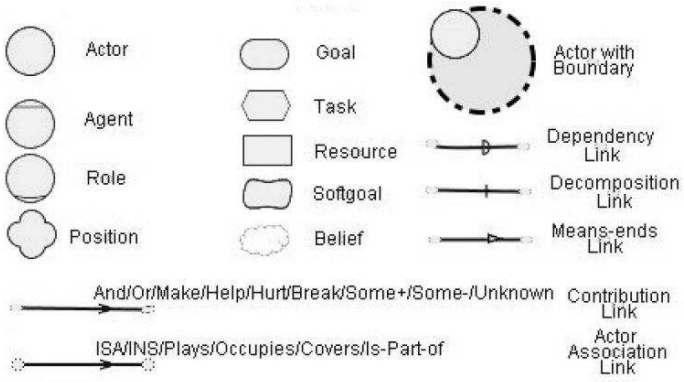

FIG. 7.5 - Légende associée aux diagrammes i*

Nous avons représenté les objectifs grâce à deux diagrammes d'objectifs i* correspondant aux deux fonctionnalités :

- La figure 7.6 correspond à la fonctionnalité d'ajout de types et d'énoncés par défaut.
- La figure 7.7 correspond à la génération automatique de la SEI sur base de la SES.

Les acteurs représentés sur ces diagrammes correspondent aux différents types d'utilisateurs du logiciel *GenSpec* (cf section 3.1.6). On retrouve l'utilisateur, l'administrateur et les autres usagers. Les autres usagers sont le concepteur, le commentateur, le visiteur et le testeur de *GenSpec*. Nous avons préféré rassembler ces différents acteurs ayant les mêmes objectifs afin de ne pas être redondant.

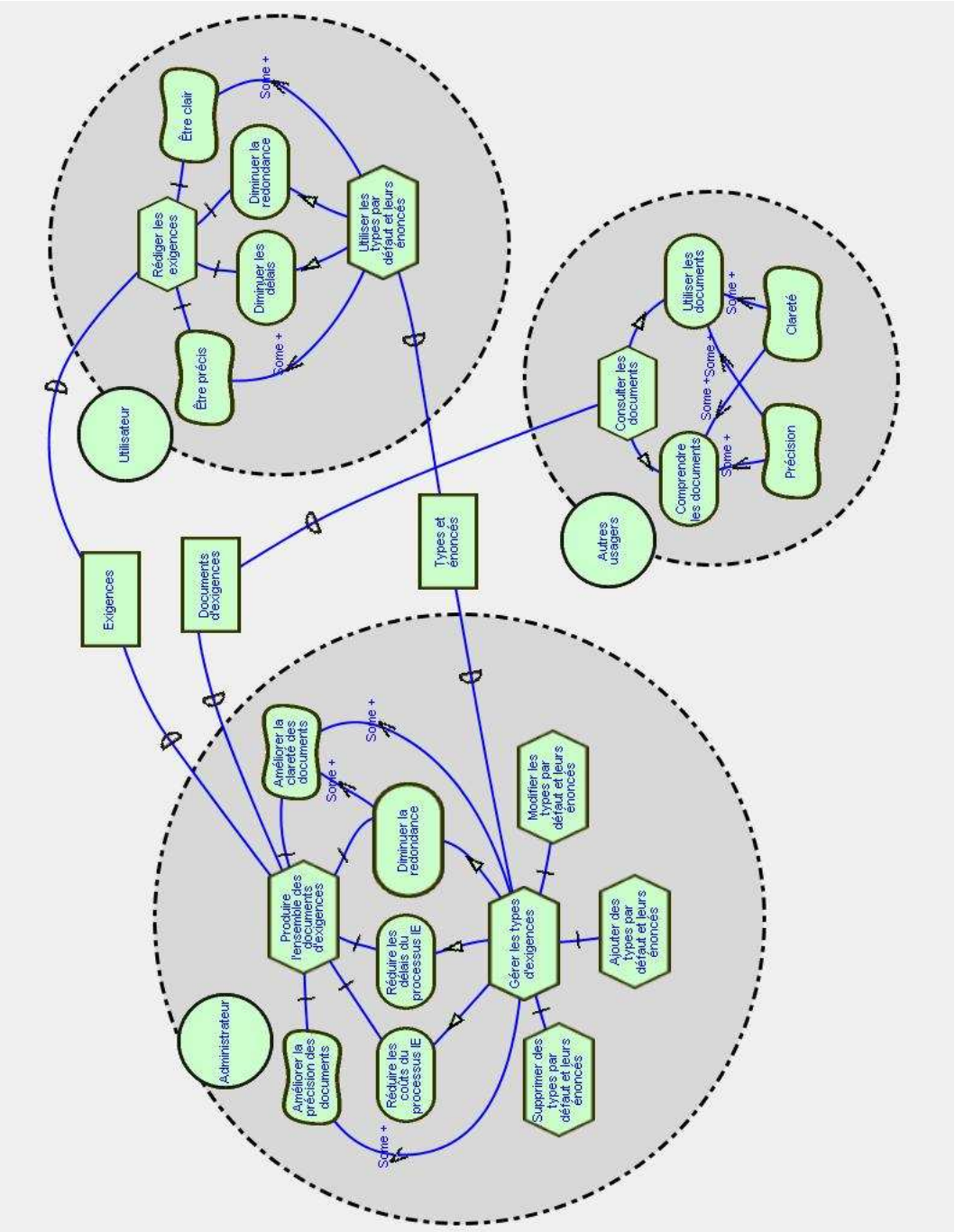

FIG. 7.6 - Diagramme d'objectifs de la fonctionnalité 1

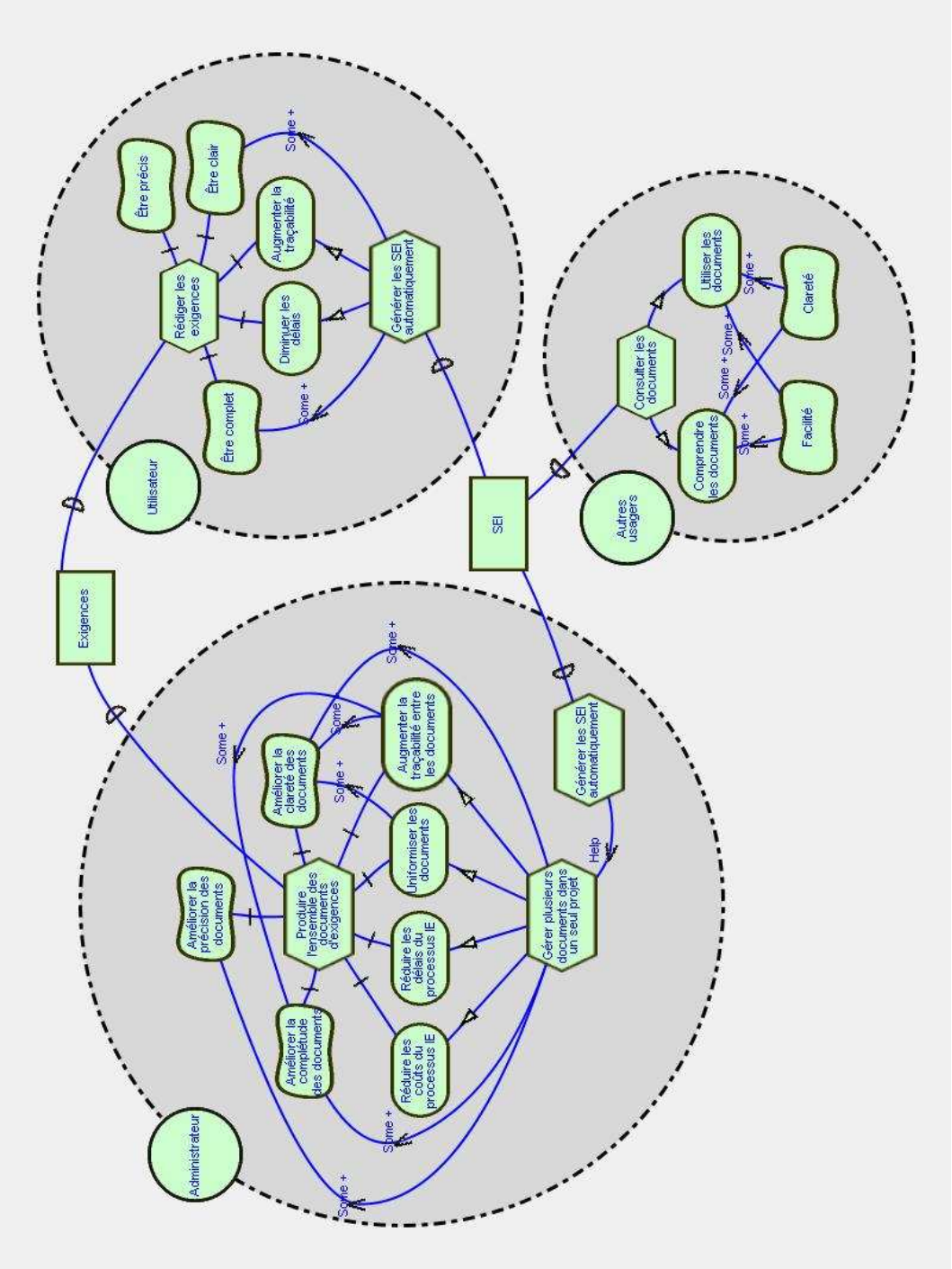

FIG. 7.7 - Diagramme d'objectifs de la fonctionnalité 2

Réalisation de cas d'utilisation

Afin de supporter les activités d'élicitation et de rédaction des exigences, nous avons réalisé des diagrammes de cas d'utilisation (Annexes F et G). Ceux-ci sont accompagnés de scénarios qui expriment la manière dont le système doit intéragir avec l'utilisateur ou un autre système afin de réaliser un but spécifique (cf section 2.3.3). Les modèles d'objectifs sont utilisés comme base afin de réaliser les cas d'utilisation.

A chaque itération ou incrément, un nouveau diagramme était produit ainsi que les scénarios associés à celui-ci. Afin de synthétiser l'ensemble de ces schémas, nous fournissons les diagrammes de cas d'utilisation globaux des deux fonctionnalités (Fig. 7.8 et Fig. 7.9).

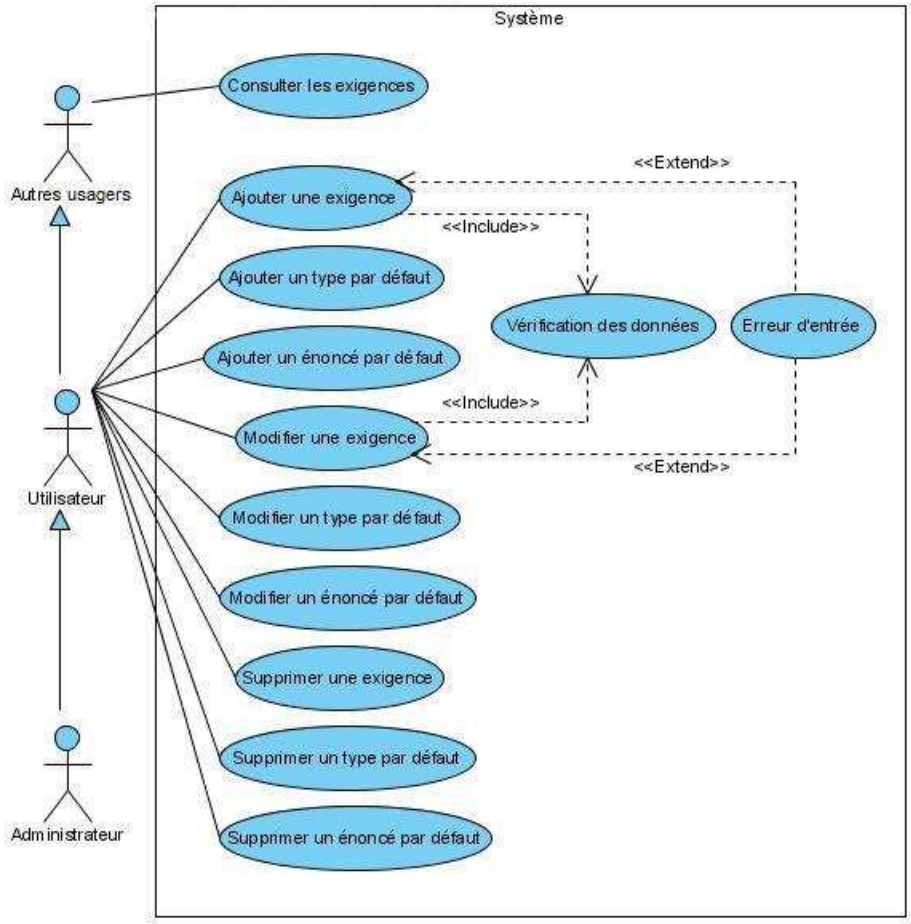

FIG. 7.8 - Diagramme de cas d'utilisation associé à la fonctionnalité 1

Le diagramme ci-dessus (Fig. 7.8) représente les cas d'utilisation de la fonctionnalité 1. On constate que l'utilisateur peut ajouter, modifier et supprimer une exigence, un type ou un énoncé par défaut. On remarque également que tous les usagers peuvent consulter les exigences. Les cas d'utilisation d'ajout et de modification d'une exigence sont étendus par le même cas d'utilisation d'erreur d'entrée car l'ajout et la modification d'une exigence sont très similaires. L'administrateur peut également réaliser tout ce qui est permis à l'utilisateur et aux autres usagers.

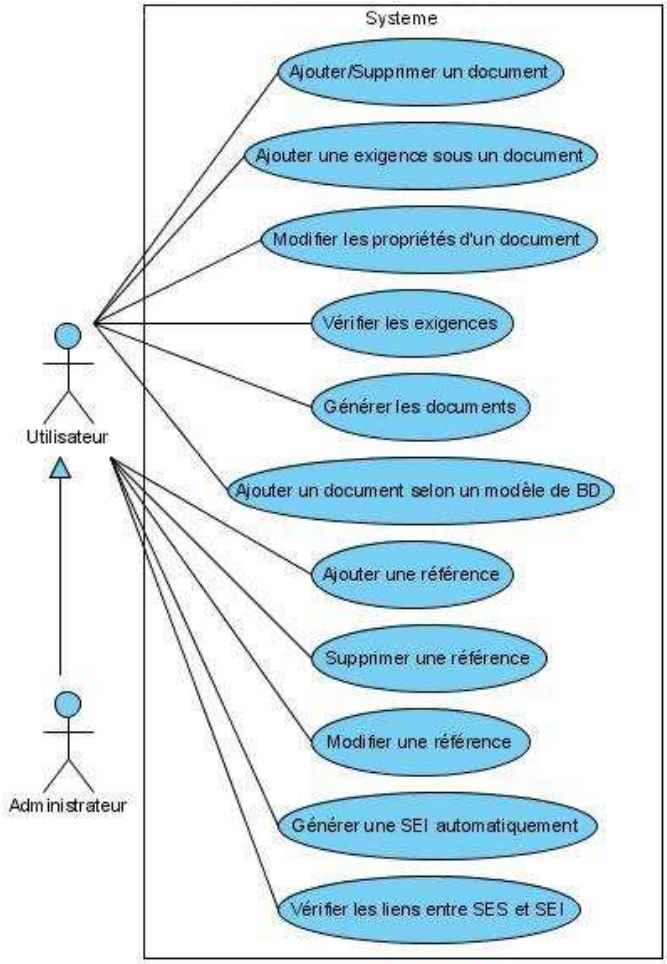

FIG. 7.9 - Diagramme de cas d'utilisation associé à fonctionnalité 2

Le diagramme ci-dessus (Fig. 7.9) représente les cas d'utilisation de la fonctionnalité 2. On constate que l'utilisateur peut effectuer diverses actions relatives aux documents : ajout, modification, suppression et génération. Il peut également ajouter, modifier et supprimer une référence. Enfin, il peut effectuer des vérifications et générer une SEI automatiquement. Toutes ces possibilités sont également offertes à l'administrateur.

Pour plus d'informations concernant les diagrammes et les scénarios associés à ceux-ci, consulter les annexes (Annexes F et G).

Rédaction des exigences avec *GenSpec*

Au sein de l'unité CA d'HQ, les exigences de chaque projet sont rédigées à l'aide de l'outil *GenSpec*. Afin de respecter cette pratique, nous avons également rédigé les exigences de notre projet en langage naturel à l'aide de ce logiciel. Le modèle que nous avons choisi est celui de la norme IEEE-Std-830 [Sof 98b]. Ces exigences ont été rédigées suite aux entrevues avec les parties prenantes et l'élaboration des modèles d'objectifs et des diagrammes de cas d'utilisation (cf section 7.2.3).

Nous avons rédigé les exigences relatives aux interfaces externes ainsi que les exigences fonctionnelles et non-fonctionnelles pour chaque fonctionnalité. Pour la fonctionnalité 1, 135 exigences ont été rédigées contre 150 pour la fonctionnalité 2.

Les documents d'exigences des deux fonctionnalités sont disponibles en annexe (Annexes D et E).

Élaboration du schéma entité-association conceptuel

Le schéma ci-dessous (Fig. 7.10) présente le schéma entité-association conceptuel de la BD du logiciel *GenSpec*. Celui-ci était mis à jour à chaque incrément/itération parallèlement à la rédaction des exigences. Un modèle conceptuel de BD est un formalisme de description des informations relatives à un domaine d'application. Il est, avant tout, un formalisme de représentation de connaissances [Hainaut 03].

Les stagiaires de 2006, Olivier Pire et Nicolas Pirmez, avaient réalisé un schéma entité-association conceptuel complet de la BD associée à *GenSpec* [Pire 07]. Nous avons donc mis à jour ce schéma faisant partie de la documentation de la BD.

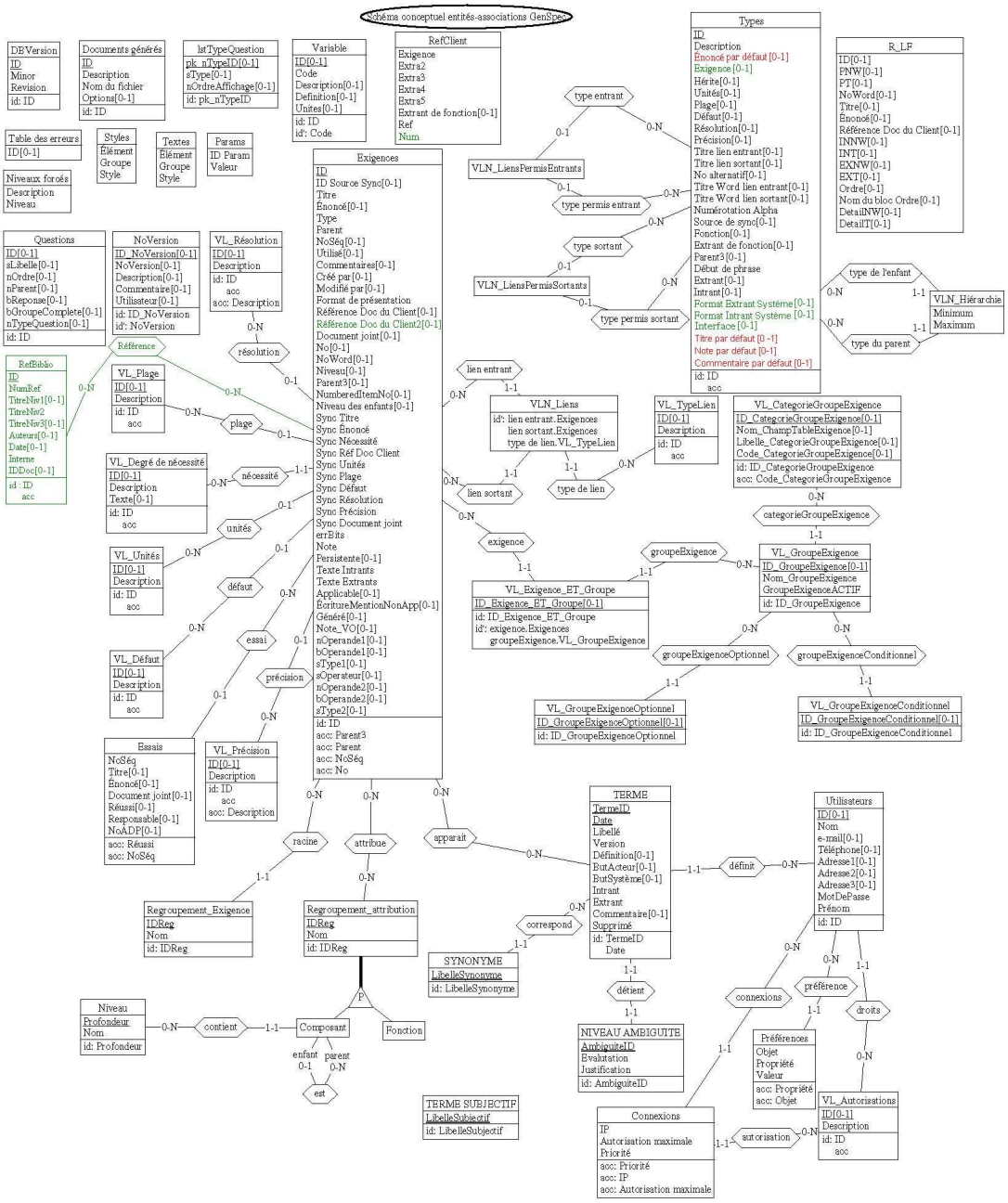

FIG. 7.10 - Schéma entité-association conceptuel de *GenSpec*

Les modifications apportées par les deux fonctionnalités sont mises en évidence :

En rouge : la fonctionnalité d'ajout de types et d'énoncés par défaut.

- Les attributs Énoncé, Titre, Note et Commentaire par défaut ont été ajoutés à la table Types afin que ceux-ci soient générés lors de la sélection d'un type.

En vert : la fonctionnalité de génération automatique de la SEI sur base de la SES.

- Les attributs de type booléen Exigence, Interface, Format Extrant Système et Format Intrant Système ont été ajoutés à la table Types afin de supporter la fonctionnalité de vérification automatique,
- L'attribut "Référence Doc du Client2" a été ajouté à la table Exigences afin qu'il soit possible de distinguer deux types de références aux documents du client,
- La table RefBiblio a été ajoutée afin de gérer les références bibliographiques externes.

7.2.4 Conception

Lors de l'étape de conception du processus de développement, plusieurs types de diagramme ont été élaborés sur base des documents produits durant l'analyse des besoins. Le lecteur trouvera en annexe les deux documents produits lors des différentes phases de conception des deux fonctionnalités (Annexes H et I).

Élaboration de diagrammes d'états

Nous avons produit différents diagrammes d'états que vous pouvez retrouver dans les annexes (Annexes H et I).

Des diagrammes ont été produits pour les objets suivants : une exigence, un type, un document, une référence et une SEI. Cela nous a permis d'envisager les différents changements d'états que peuvent connaitre ces objets, en réponse aux interactions avec d'autres objets, composants ou acteurs.

Élaboration du diagramme de composants

Un diagramme de composants décrit l'organisation du système du point de vue des modules de code. Il permet de mettre en évidence les dépendances entre les composants (qui utilise quoi) et ainsi de mieux organiser les modules. Les dépendances entre composants permettent notamment d'identifier les contraintes de compilation et de mettre en évidence la réutilisation de composants. Les composants sont des agrégations de haut niveau de fragments du logiciel plus petits [Jaton 07].

Le schéma de la page suivante (Fig. 7.11) présente le diagramme de composants et les services principaux associés aux deux fonctionnalités :

- Les services/composants bleus sont créés/modifiés uniquement dans le cadre de la fonctionnalité de génération automatique de la SEI sur base de la SES.

- Les services/composants jaunes sont créés/modifiés uniquement dans le cadre de la fonctionnalité d'ajout de types et d'énoncés par défaut.

- Les services/composants verts sont créés/modifiés dans le cadre des deux fonctionnalités.

Les composants que nous avons créés sont le Gestionnaire de génération SEI, le Gestionnaire des références et le Gestionnaire des types. Les interfaces que nous avons créées sont gestionTypes, gestionRéférences et générerSEI. Les autres composants et interfaces ont

uniquement été modifiés et existaient auparavant.

Les composants relatifs à l'ajout des types et des énoncés par défaut sont :

- Gestionnaire des types : gère toutes les informations relatives aux types d'exigences (ajout, modification et suppression).

Les composants relatifs à la génération automatique de la SEI sur base de la SES sont :

- Gestionnaire de génération SEI : gère la génération automatique de la SEI sur base de la SES,
- Gestionnaire des références : gère toutes les informations relatives aux références (ajout, modification et suppression).

Les composants communs aux deux fonctionnalités sont :

- Gestionnaire des propriétés : gère les exigences et les documents (ajout, modification et suppression),
- Gestionnaire des Intrants/Extrants : gère les interactions avec les utilisateurs. Il fait appel aux autres composants via l'interface homme-machine,
- Gestionnaire de vérification : gère les vérifications automatiques des exigences,
- Gestionnaire de la BD : gère les accès à la BD,
- Gestionnaire de génération des documents : gère la génération des documents d'exigences au format Microsoft Word.

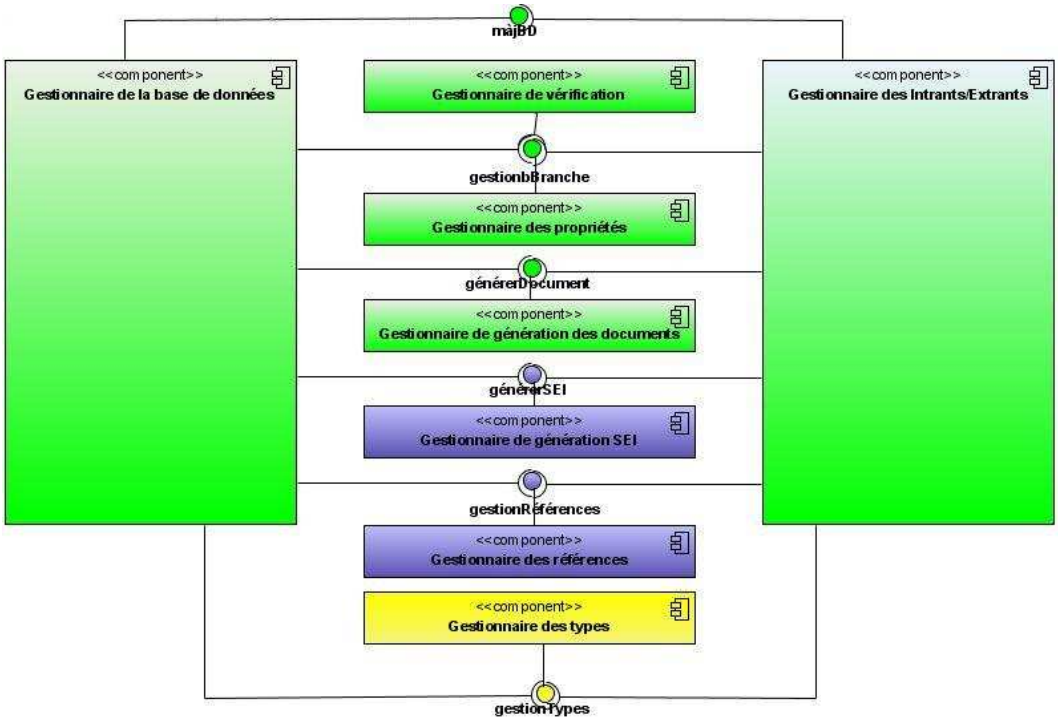

FIG. 7.11 - Diagramme de composants associé aux deux fonctionnalités

Élaboration de diagrammes de séquence

Nous avons produit différents diagrammes de séquence que vous pouvez retrouver dans les annexes (Annexes H et I).

Ces diagrammes représentent l'exécution des cas d'utilisation que nous avions produits lors de l'analyse des besoins (cf section 7.2.3). Ils nous ont permis de déterminer les différentes fonctions à implémenter et les composants dans lesquels il fallait le faire. Nous avons également pu nous rendre compte des interactions nécessaires entre ces composants. Pour cela, nous avons parcouru le code afin de découvrir si les fonctions nécessaires n'existaient pas déjà et pour comprendre et situer celles qui devaient être modifiées. Cela nous a permis de comprendre le code préexistant et d'éviter de créer des fonctions redondantes grâce à la réutilisation.

Élaboration du diagramme de classes

Le schéma ci-dessous (Fig. 7.12) présente le diagramme de classes et les principales méthodes associées aux deux fonctionnalités. Un bon nombre de méthodes moins importantes ont été implémentées mais nous ne les reprenons pas ici par souci de concision. Nous avons utilisé la même convention de couleurs que celle utilisée pour le diagramme de composants (cf section 7.2.4).

Les gestionnaires que nous avons créés sont le Gestionnaire de génération SEI, le Gestionnaire des références et le Gestionnaire des types. Les méthodes que nous avons créées sont celles permettant d'ajouter, modifier et supprimer un type ou une référence ainsi que celle permettant de générer une SEI automatiquement. Les autres gestionnaires et méthodes ont uniquement été modifiés.

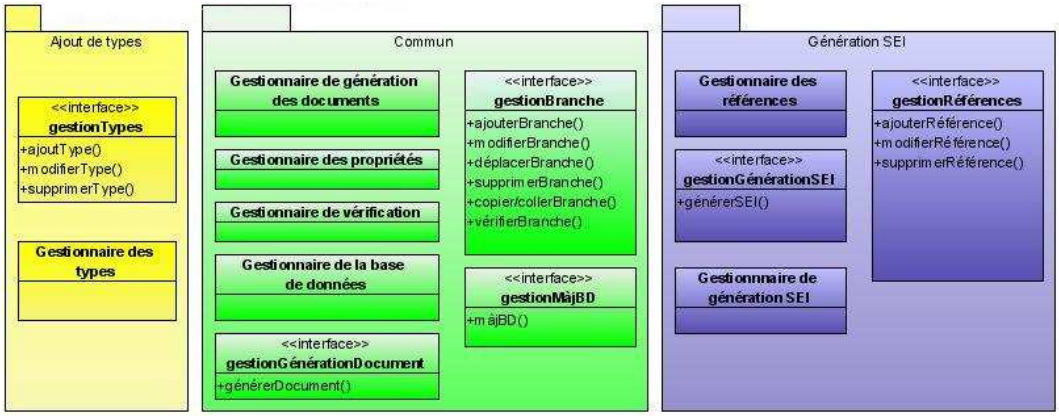

FIG. 7.12 - Diagramme de classes associé aux deux fonctionnalités

Élaboration du schéma logique

Le schéma de la page suivante (Fig. 7.13) présente le schéma logique de la BD du logiciel *GenSpec*. Celui-ci était mis à jour à chaque incrément/itération. Tout comme le schéma entité-association conceptuel, ce sont les stagiaires de 2006, Olivier Pire et Nicolas Pirmez qui l'ont réalisé afin de documenter la BD de *GenSpec* [Pire 07].

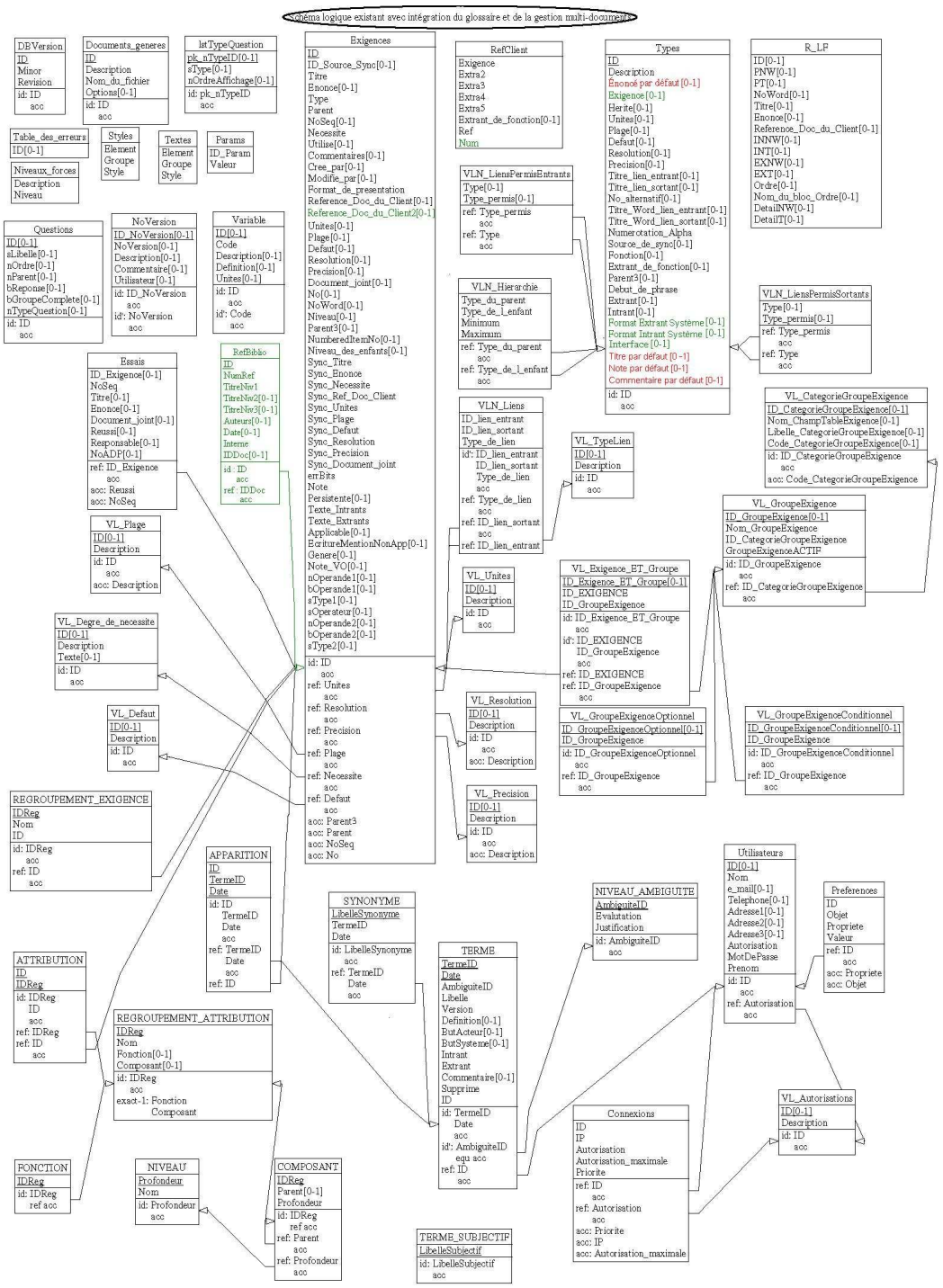

FIG. 7.13 - Schéma logique de *GenSpec*

Les modifications apportées par les deux fonctionnalités sont mises en évidence et sont les mêmes que celles énoncées pour le schéma entité-association conceptuel (cf section 7.2.3) :

- En rouge : la fonctionnalité d'ajout de types et d'énoncés par défaut.
- En vert : la fonctionnalité de génération automatique de la SEI sur base de la SES.

Attribution des exigences aux composants

L'attribution des exigences aux composants a été réalisée via l'outil *GenSpec*. Cette fonctionnalité a été développée par d'autres stagiaires venant des FUNDP, Nicolas Pirmez et Olivier Pire [Pire 07].

Lors de cette activité, le concepteur crée des composants et des fonctions afin de réaliser la conception architecturale du système. Cette découpe du système en hiérarchie de composants est réalisée jusqu'à ce que chaque exigence de l'incrément ou de l'itération soit allouée à un ou plusieurs composants. Pour cela, un parcours du code existant est nécessaire ce qui permet de se plonger au coeur de l'application et d'avoir une vision plus affinée de la manière d'implémenter l'incrément ou l'itération : fonctions à modifier, fonctions à réutiliser, etc.

Grâce à *GenSpec*, il a été également possible de générer un tableau des différentes attributions. Vous pourrez retrouver ceux-ci en annexe (Annexes H et I).

7.2.5 Implémentation

Comme nous l'avons déjà dit précédemment (cf section 7.1.2), la personne qui était chargée de l'implémentation d'un incrément ou itération était la personne ayant réalisé la phase de conception de cet incrément ou itération.

Codage et intégration de l'incrément ou itération

Le codage et l'intégration de chaque incrément ou itération a pour but d'implémenter les exigences spécifiées de cet incrément ou itération. Cette activité est réalisée en se basant sur les documents produits lors des phases d'analyse des besoins et de conception.

L'ensemble du code produit et intégré à l'outil *GenSpec* a été réalisé en Visual Basic 6.0 via l'outil Microsoft Visual Studio.

Le chapitre 8 décrit plus en détail la solution implémentée et explique la manière dont celle-ci a été intégrée à l'application existante.

7.2.6 Vérification et validation

Comme nous l'avons déjà dit auparavant (cf section 7.1.2), la personne qui était chargée de la vérification d'un incrément ou itération était également la personne ayant réalisé l'analyse des besoins de cet incrément ou itération. Elle connaissait donc parfaitement les exigences devant être vérifiées et réalisait les tests de manière supposément plus objective étant donné qu'elle n'était pas responsable de l'implémentation de ces exigences. L'activité de validation était réalisée par le comité *GenSpec* et en particulier monsieur René Bujold.

Réalisation du cahier des essais

Une des fonctions principales de *GenSpec* (cf section 3.1.3) est l'évaluation de conformité aux exigences. Grâce à elle, l'utilisateur peut créer des procédures d'évaluation de conformité pour chaque exigence et générer un document contenant les exigences, les procédures d'évaluation et les résultats de ces dernières. Ce document est le cahier des essais. Pour chaque incrément/itération, nous avons réalisé un cahier des essais à l'aide de *GenSpec* afin de vérifier que les exigences définies étaient réalisées par l'implémentation apportée.

Vérification des exigences de l'incrément ou itération

La vérification des exigences de l'incrément ou itération est réalisée avec le cahier des essais. La vérification a pour but de vérifier le résultat de l'implémentation. Chaque exigence doit être testée et le résultat du test doit être indiqué dans le cahier des essais.

Grâce à cela, il est possible de déterminer quelles exigences ont été oubliées ou incomprises par la personne ayant réalisé l'implémentation de l'incrément ou itération. Il est également possible de découvrir des bugs éventuels dans l'implémentation. Certaines corrections de bugs étaient effectuées durant cette étape avant de générer une nouvelle version de *GenSpec* afin de passer à l'étape de validation.

Validation de la nouvelle version de *GenSpec*

La validation de chaque nouvelle version de *GenSpec* permettait de valider la cohérence entre le résultat de l'implémentation et les besoins réels des parties prenantes. Si la nouvelle version était approuvée, un nouvel incrément ou itération pouvait être développé. Dans le cas contraire, l'incrément ou itération devait être revu jusqu'à l'acceptation de celui-ci.

7.3 Résumé

Ce chapitre a décrit le mode de développement que nous avons suivi afin de fournir une solution à la problématique décrite au chapitre 5. Nous avons décrit celui-ci de manière générale avant de fournir une description détaillée des différentes activités. Pour cela, nous les avons regroupées selon les grandes étapes suivantes :

- Planning,
- Étude préliminaire,
- Analyse des besoins,
- Conception,
- Implémentation,
- Vérification et Validation.

Le chapitre suivant vise à décrire les solutions techniques implémentées afin de solutionner les problèmes décrit au chapitre 5.

Chapitre 8

Solution technique

Ce chapitre a pour but de présenter les solutions techniques que nous avons apportées. Nous présenterons tout d'abord la première fonctionnalité, l'ajout de types d'exigences et d'énoncés par défaut, pour laquelle nous décrivons la solution implémentée et nous parlons ensuite de son intégration. Nous suivons ensuite le même canevas pour expliciter la réalisation de la deuxième fonctionnalité, à savoir la génération de la SEI sur base de la SES.

8.1 Ajout de types et d'énoncés par défaut

Afin de remédier aux différents problèmes exposés au chapitre 5, il a été décidé d'ajouter un ensemble de types d'exigences par défaut, pour chacun, leur énoncé associé et permettre à l'utilisateur de réaliser diverses manipulations à l'aide d'un module de gestion des types.

Comme nous l'avons décrit au chapitre 6 (cf section 6.1.2), nous avons tout d'abord dû réaliser un état de l'art de la littérature en ce qui concerne la structure, les types d'exigences et les énoncés par défaut à utiliser. Par la suite, nous avons réalisé une synthèse de ces résultats (cf Annexe J). Il s'agit dès lors d'implémenter ces résultats au sein de *GenSpec* via, notamment, un module de gestion des types.

Nous avons donc procédé en deux étapes :

- Implémentation des types et énoncés par défaut dans l'outil,
- Implémentation d'un module de gestion des types.

8.1.1 Description de la solution

Implémentation des types et énoncés par défaut

Après avoir validé l'ensemble des types et des énoncés, il nous a fallu les ajouter à l'outil *GenSpec*. Pour cela, une mise à jour de la liste des types d'exigences et énoncés par défaut présents dans l'outil a été réalisée. Puis, il a fallu programmer la génération proprement dite de l'énoncé lors du choix d'un type pour une exigence. Lors de l'implémentation de cette fonctionnalité, nous avons essayé de faciliter le travail de l'utilisateur. En effet, celui-ci peut passer d'un type à l'autre et voir les énoncés correspondant lors de l'ajout d'une exigence. De ce fait, l'utilisateur peut tester ceux qui lui paraissent les plus pertinents et choisir parmi ceux-là.

La figure 8.1 fournit un exemple de génération de l'énoncé par défaut suite à la sélection d'un type d'exigences. Lorsque l'utilisateur sélectionne un type d'exigences, un énoncé par défaut est généré. Dans le cas ci-dessous, l'utilisateur a sélectionné le type d'exigences "Performances statiques", l'énoncé par défaut généré est "Le produit doit satisfaire des exigences numériques statiques". Si l'utilisateur modifie le type de l'exigence alors l'énoncé par défaut s'adaptera sauf si l'utilisateur a déjà modifié ce dernier. En effet, nous avons décidé de ne pas

remplacer l'énoncé lorsqu'il a déjà été modifié par l'utilisateur afin d'éviter tout effacement malencontreux.

FIG. 8.1 - Génération de l'énoncé suite à la sélection d'un type

De plus, les symboles "$" et "$\mu$" peuvent être utilisés par l'utilisateur pour la génération d'énoncés par défaut. "$" représente le nom du produit et "μ" représente le titre de l'exigence. Cela permet de générer un énoncé plus complet et adaptable à chaque exigence. Par exemple, si $ = *GenSpec* et μ = Exigences de performance, l'énoncé par défaut "$ doit respecter les μ" donnera l'énoncé suivant lors de la sélection du type d'exigences associé : "*GenSpec* doit respecter les Exigences de performance". Ces symboles peuvent être utilisés via le module de gestion de types (cf section 8.1.1) lors de la définition de l'énoncé par défaut d'un type d'exigences.

Implémentation d'un module de gestion des types

Afin de permettre à l'utilisateur de gérer les types et énoncés par défaut, nous avons implémenté un module de gestion de types (Fig. 8.2). Ce module, permet de réaliser diverses actions relatives aux types par défaut : visualiser, ajouter, modifier ou encore supprimer un type par défaut.

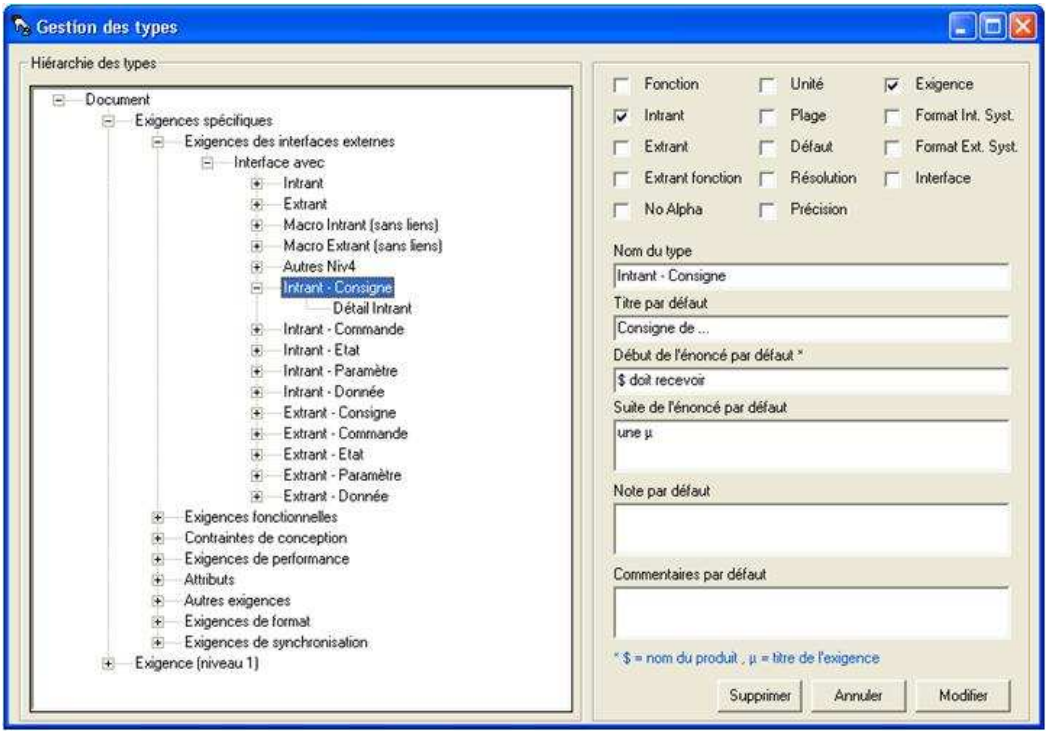

FIG. 8.2 - Module de gestion des types

Visualiser les types par défaut

Le module de gestion de types est subdivisé en deux parties. La partie de gauche contient un arbre qui permet de visualiser la hiérarchie complète des types par défaut disponibles au sein de *GenSpec*. La partie de droite du module est composée de plusieurs champs. Certains sont des options à cocher qui servent principalement aux fonctionnalités de vérification des exigences. Il faut cocher certaines options pour un type si on veut que les exigences de ce type soient soumises aux vérifications correspondantes. Les autres éléments composants cette partie du module sont : le titre par défaut, l'énoncé par défaut, la note par défaut et les commentaires par défaut. Si l'utilisateur sélectionne un type de la hiérarchie, l'ensemble des informations relatives à celui-ci sont affichées et il peut donc parcourir l'arbre pour trouver un type correspondant à l'exigence qu'il veut formuler.

Ajouter un type par défaut

Avec l'arbre représentant la hiérarchie des types, il est possible d'ajouter un type sous un autre ou au même niveau que celui-ci. Grâce à cela, l'utilisateur peut, en tout temps, ajouter des types par défaut qui lui paraissent pertinents dans le cadre d'un projet. Lorsqu'il choisit l'endroit où il ajoute le type, il doit remplir le nom de celui-ci obligatoirement tandis que les champs pour entrer le titre par défaut, énoncé par défaut, note par défaut et commentaires par défaut sont faculatifs. Si l'utilisateur complète ces champs facultatifs, ils seront générés lors du choix de ce type pour une exigence.

L'utilisateur peut également décider de copier un type déjà existant pour le placer à un endroit de la hiérarchie auquel il ne se trouve pas encore. Il s'agit en quelque sorte d'une autre forme d'ajout de types.

Modifier un type par défaut

Comme il a été dit précédemment, il est possible d'afficher l'ensemble des informations relatives à un type : énoncé associé, options cochées ou non, etc. Grâce au module de gestion de types, la possibilité est offerte de modifier ces informations, que ce soit pour les types par défaut ayant été ajoutés suite à notre recherche ou les types ajoutés par l'utilisateur lors de la réalisation du document d'exigences. Par exemple, un utilisateur peut ajouter un type, sans choisir d'énoncé momentanément et le compléter par la suite.

Supprimer un type par défaut

Tout comme pour la modification, la suppression est permise aussi bien pour les types par défaut ayant été ajoutés suite à notre recherche que pour les types ajoutés par l'utilisateur lors de la réalisation du document d'exigences. Par exemple, certains types par défaut peuvent paraître non-pertinents dans le cadre d'un projet précis, l'utilisateur peut alors les supprimer afin d'organiser la hiérarchie des types comme il le souhaite.

8.1.2 Intégration

Changements aux fonctionnalités préexistantes

L'ajout de cette nouvelle fonctionnalité n'a que peu d'impacts sur les autres fonctionnalités. Elle est en effet assez isolée et ne demande pas de modifications majeures de *GenSpec*.

L'ajout de types d'exigences par défaut ne fait qu'augmenter le nombre de types d'exigences disponibles pour l'utilisateur.

Le module de gestion des types est une fenêtre indépendante au sein de l'outil. Seule la liste des types d'exigences peut être impactée par cette nouvelle fonctionnalité puisque cette dernière peut être agrandie (ajout de types), modifiée (modification de types) ou diminuée

(suppression de types).

L'intégration d'énoncés par défaut ne modifie en rien les fonctionnalités préexistantes puisqu'il s'agit uniquement de fournir un premier aperçu à l'utilisateur de la forme attendue de l'exigence.

Changements aux interfaces

L'ajout de nouveaux types d'exigences et d'énoncés par défaut entraînent quelques modifications du point de vue des interfaces de *GenSpec*. Ces modifications sont assez légères puisqu'elles consistent uniquement en l'ajout d'une nouvelle fenêtre indépendante pour la gestion des types, la mise à jour du menu principal et la mise à jour de l'aide.

Afin de pouvoir gérer les types d'exigences et leur énoncé par défaut, il a fallu ajouter une fenêtre de gestion des types. Cette fenêtre permet l'affichage des types disponibles et de leurs énoncés par défaut ainsi que l'ajout, la modification et la suppression de ces types et énoncés. Elle est entièrement indépendante des autres fenêtres de *GenSpec*.

Afin de pouvoir gérer les types par défaut, il a fallu prévoir une nouvelle tâche dans l'un des champs existants de la barre générale de *GenSpec*. Ce dernier permet de lancer l'ouverture de la fenêtre de gestion des types.

Afin d'aider l'utilisateur dans l'utilisation de cette nouvelle fonctionnalité, il a fallu également mettre à jour l'aide disponible. Celle-ci est plus complète et l'on peut y retrouver des informations concernant la gestion des types d'exigences et de leur énoncé par défaut.

Changements envisagés mais non inclus

Différents changements ont été envisagés mais ils n'ont pas été jugés prioritaires par le concepteur de GenSpec. Ces derniers seront expliqués dans le dernier chapitre, nous ne faisons ici que les énoncer :

- Nouvelle interface d'ajout d'une exigence,
- Gestion d'une liste de types d'exigences,
- Importation de nouveaux types d'exigences.

Changements aux classes d'utilisateurs

Nous allons maintenant énoncer les droits associés à chaque classe d'utilisateurs pour cette nouvelle fonctionnalité :

- **Administrateur** : il possède tous les droits vis-à-vis de l'ajout de nouveaux types d'exigences et de la rédaction des énoncés par défaut associés. Il est le seul à pouvoir modifier les descriptions par défaut de toutes les exigences. Il peut également ajouter, modifier ou supprimer des types d'exigences.

- **Utilisateur, Testeur, Commentateur, Visiteur, Concepteur** : ces cinq dernières classes d'utilisateurs n'ont aucun droit en ce qui concerne la gestion des types d'exigences et leurs énoncés par défaut.

8.2 Génération de la SEI sur base de la SES

Afin de remédier aux différents problèmes exposés au chapitre 5, il a été décidé d'ajouter une option de génération automatique de la première mouture de la SEI sur base de la SES. De plus, la possibilité de manipuler plusieurs documents au sein d'une même instance de l'outil et un module de gestion des références ont été implémentés.

Nous avons donc agit en plusieurs étapes, les éléments suivants ont été ajoutés :

- le niveau document,
- la possibilité de générer la structure d'un document,
- un module de gestion des références,
- la génération proprement dite de la première mouture de la SEI sur base de la SES,
- les vérifications automatiques liées aux références entre SES et SEI.

8.2.1 Description de la solution

Niveau document

La première chose à implémenter fut le niveau document. En effet, sans cette amélioration, il était impossible d'envisager la suite du développement. Certaines fonctions existaient déjà afin de gérer l'aspect multi-documents mais celles-ci n'étaient pas suffisantes pour permettre l'implémentation de la génération automatique de la SEI. Il a donc fallu ajouter un niveau supplémentaire permettant de regrouper un ensemble d'exigences. Beaucoup de fonctionnalités déjà présentes dans le logiciel ont du être révisées afin de supporter ce niveau supplémentaire, entre autres, la génération des documents dans le format Microsoft Word. Grâce aux modifications que nous avons apportées, il est désormais possible d'ajouter/supprimer de nouveaux documents de manière claire et explicite. Il est également possible de générer plusieurs documents en plusieurs fichiers ou dans un grand fichier reprenant l'ensemble de ceux-ci.

Génération de la structure d'un document

Afin de respecter les normes, *GenSpec* offre la possibilité, lors de la création d'un nouveau projet, de choisir un modèle inspiré de la norme IEEE-Std-830 [Sof 98b] ou encore de la norme ISO 12207 [Int 95]. Puisque l'on peut désormais ajouter de nouveaux documents au sein d'un même projet, il a fallu permettre, lors de ces ajouts, de choisir une structure tout comme lors de la création d'un projet. La SEI peut dorénavant être générée selon un certain modèle.

Module de gestion des références

Au niveau des références, *GenSpec* offrait déjà la possibilité de lier plusieurs exigences d'un même document. Il a donc fallu adapter cela afin qu'il soit possible de lier deux documents séparés. En effet, si une SEI est générée et imprimée séparément de la SES dont elle est tirée, il faut pouvoir créer une référence vers la SES et ensuite désigner une exigence au sein de celle-ci. Pour cela, un module complet de gestion des références a été créé. Deux types de références sont désormais possibles : référence aux exigences sources et références diverses. Pour chacun des types de références, on peut créer :

- un lien au sein d'un même document,
- un lien entre des documents différents appartenant à un même projet,
- un lien entre des documents n'appartenant pas à un même projet.

Un lien peut aussi bien pointer une exigence d'un document du même projet qu'une partie d'un document totalement externe. En effet, les liens entre exigences sont intéressants mais il doit également être possible de pointer autre chose qu'une exigence. Par exemple, on peut désormais réaliser un lien entre une exigence et un paragraphe d'un livre qui a été consulté au moment de la rédaction de cette exigence.

La figure 8.3 illustre le module de gestion des références implémenté. Ce dernier permet d'entrer les informations suivantes : un numéro de référence unique, un titre en trois niveaux, une date et les auteurs ainsi que, éventuellement, un lien vers un document interne au projet.

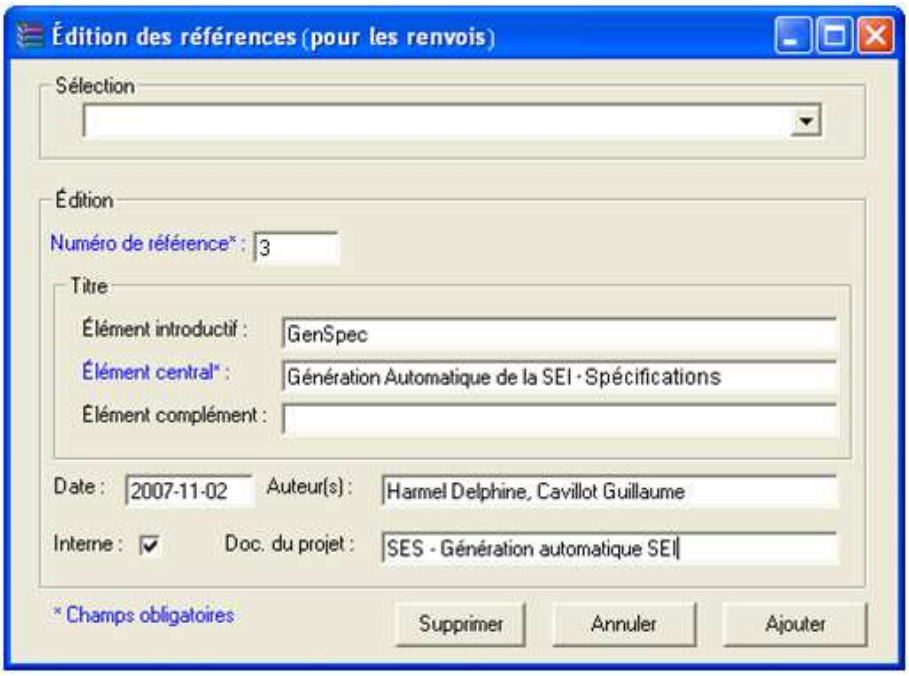

FIG. 8.3 - Module de gestion des références

Avec ce module de gestion des références, il a fallu revoir la génération du tableau des références sous format Microsoft Word. Ce tableau reprend l'ensemble des références aussi bien internes à un document qu'externes à celui-ci. Le tableau possède deux colonnes : le numéro de la référence et le titre associé à cette dernière.

Le module que nous avons implémenté pourrait également permettre de générer les références bibliographiques d'un document. En effet, celui-ci permet d'ajouter/modifier/supprimer une référence externe à un document.

Génération de la SEI sur base de la SES

L'étape suivante de notre implémentation consistait à réaliser la génération proprement dite de la SEI sur base de la SES. Pour chaque interface décrite dans la SES, une SEI peut désormais être générée. Afin de générer une SEI, l'utilisateur peut soit ne pas choisir de structure, soit opter pour une structure qui suit la norme IEEE-Std-830 ou encore la norme ISO 12207. Ensuite, une série d'exigences sont générées. Voici la liste exhaustive des règles de génération de la SEI sur base de la SES :

- Un nouveau document portant le nom de l'interface choisie est créé,
- Au sein de ce document, on retrouve une exigence pour chaque exigence décrite dans la partie concernant l'interface dans la SES,
- Pour toute exigence de type "Intrant" de la SES, une exigence de type "Format Intrant" est créée dans la SEI (et identiquement pour le type "Extrant"),
- Pour toute exigence de type "Macro Intrant" de la SES, une exigence de type "Macro Format Intrant" est créée dans la SEI (et identiquement pour le type "Macro Extrant"),
- Un lien entre chaque exigence de la SEI et l'exigence correspondante dans la SES est réalisé et inversément,
- Si l'utilisateur a choisi une structure, les autres exigences prévues par celle-ci sont générées.

La figure 8.4 fournit un exemple de résultat suite à la génération de la SEI à partir de l'exigence "Interface avec l'exploitant" de la SES. Dans ce cas précis, l'utilisateur a opté pour une génération de la SEI suivant la norme IEEE-Std-830 et cela a pour effet de géné-

rer d'autres exigences telles que les exigences de synchronisation, etc. Pour chaque exigence générée dans la SEI sur base d'une exigence de la SES, le titre est conservé. Par exemple, l'exigence 3.1.1.1.1 de la SES s'appelant "Intrant 1" et étant de type "Intrant", génère l'exigence 3.1.2.1.1 de la SEI s'appelant également "Intrant 1" mais étant de type "Format Intrant".

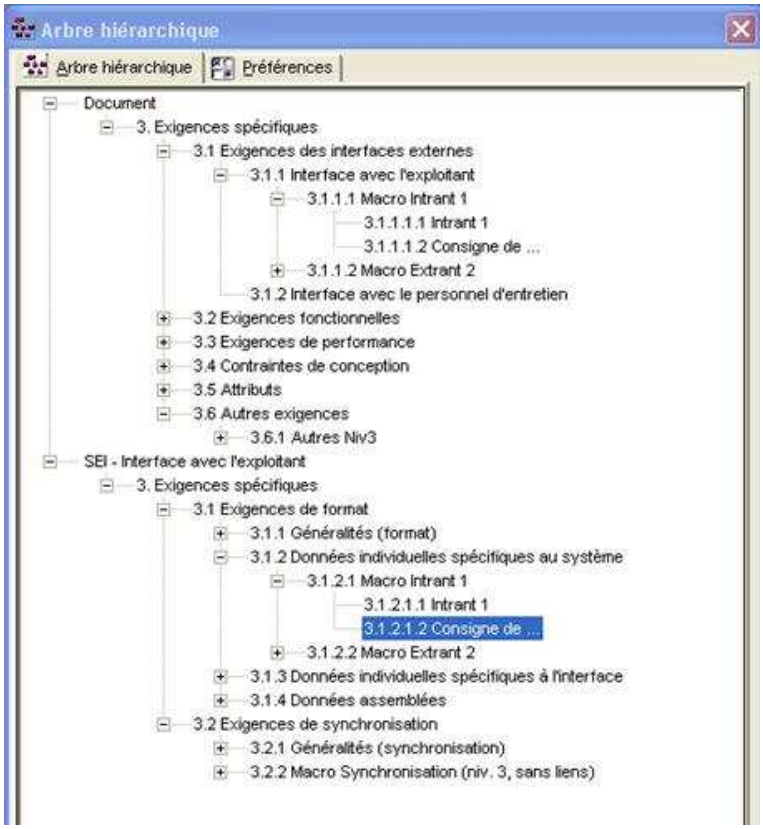

FIG. 8.4 - Résultat de la génération d'une SEI

Vérifications automatiques

La dernière partie de l'implémentation concernait les vérifications automatiques liées aux références entre SES et SEI. *GenSpec* offre déjà un bon nombre de vérifications afin de permettre à l'utilisateur d'améliorer la qualité de ses documents d'exigences. Comme il a déjà été dit, pour chaque exigence de la SEI, il doit exister un lien avec l'exigence correspondante dans la SES et vice-versa. Ces deux vérifications ont donc été implémentées.

8.2.2 Intégration

Changements aux fonctionnalités préexistantes

Cette sous-section décrit les modifications apportées aux fonctionnalités actuellement présentes au sein de *GenSpec* ainsi que les nouveaux aspects introduits au sein du logiciel.

Génération de documents Word

Auparavant, il était impossible de générer, en même temps, plusieurs documents Word d'un même projet. Dans le cadre de la nouvelle fonctionnalité, il était indispensable de pouvoir générer la SES et la SEI en même temps. C'est pourquoi la génération de documents Word a du être quelque peu modifiée pour supporter cette nouvelle option.

De nouveaux types d'exigences ont dû être ajoutés afin, notamment, de distinguer les éléments de la SES de ceux de la SEI. Par exemple, le type "Intrant" de la SES devient "Format Intrant" dans la SEI. Ces types n'étaient pas définis dans la liste des types d'exigences et il a donc fallu les inclure.

Nous avons également dû ajouter la possibilité de réaliser des vérifications suite aux liens créés entre la SES et la SEI. Par exemple, il doit être possible de repérer les intrants de la SEI pointant vers un intrant de la SES qui aurait été supprimé pour le signaler à l'utilisateur.

Changements aux interfaces

L'ajout de la fonctionnalité de génération automatique de la SEI sur base de la SES a engendré différentes modifications dans les interfaces de l'outil *GenSpec*.

Modification des menus

Il a fallu, tout d'abord, ajouter un élément pour le clic droit sur les interfaces présentes dans la partie interfaces externes de la SES afin d'offrir la possibilité de générer une SEI correspondante. De plus, il fallait implémenter l'ajout d'un document au même niveau. Le changement apporté aux interfaces était donc mineur puisqu'il s'agissait principalement d'ajouter une option à des menus.

Ajout d'une fenêtre de sélection d'une structure

Ensuite, il a fallu créer une nouvelle fenêtre afin de proposer à l'utilisateur de choisir une structure qu'il désire lors de la génération de la SEI. En effet, l'utilisateur peut créer un document qui dispose déjà d'une certaine structure d'exigences. Cette structure suit principalement les normes IEEE-Std-830 [Sof 98b] et ISO 12207 [Int 95]. Cependant, cette fenêtre est identique à celle utilisée lors de la création d'un projet.

Modification de la fenêtre "Exigences"

Il a été également nécessaire de renommer la fenêtre "Exigences" en "Propriétés" afin que celle-ci soit cohérente avec le fait que l'on peut désormais avoir à la fois des documents et des exigences dans un même projet. Il semblait en effet incohérent de garder une fenêtre nommée "Exigences" alors que cette dernière caractérisait un document.

Modification de l'onglet "Compléments"

Un autre changement à apporter aux interfaces se situait au niveau de la fenêtre "Propriétés" (remplaçante de la fenêtre "Exigences"). Il fallait modifier l'onglet "Compléments" afin de pouvoir réaliser des renvois vers des documents du projet.

Ajout d'une fenêtre de gestion des références

Afin de pouvoir gérer les références, il a fallu ajouter une fenêtre de gestion de celles-ci. Cette fenêtre permet d'afficher les différentes références ajoutées. Elle permet également l'ajout, la modification et la suppression de références. Elle est entièrement indépendante des autres fenêtres de *GenSpec*.

Mise à jour de l'aide

L'aide intégrée de l'outil est désormais plus complète. Des explications concernant le fonctionnement de la nouvelle fonctionnalité y sont consignées.

Changements aux classes d'utilisateurs

Nous allons maintenant énoncer les droits associés à chaque classe d'utilisateurs pour cette nouvelle fonctionnalité :

- **Administrateur** : un administrateur possède tous les droits sur un projet. Il peut donc générer la SEI au sein du programme ou dans un document. Il a également le droit d'ajouter des exigences à la SEI.

- **Utilisateur** : un utilisateur possède les mêmes droits qu'un administrateur au niveau de la SEI.

- **Visiteur** : un visiteur possède seulement les droits de lecture de tous les champs et de génération des documents. Il pourra donc uniquement générer la SEI au format Word. Il ne pourra pas générer celle-ci au sein du programme et y ajouter des exigences. Il faudra donc que la SEI soit générée au préalable pour que le visiteur puisse la générer dans un document Word.

- **Commentateur** : un commentateur a les mêmes droits qu'un visiteur au niveau de la SEI et il peut également commenter les exigences de celle-ci.

- **Testeur** : un testeur a les mêmes droits qu'un commentateur au niveau de la SEI et il peut accéder à l'onglet Essais des exigences de celle-ci.

- **Concepteur** : un concepteur a les mêmes droits qu'un commentateur au niveau de la SEI.

8.3 Résumé

Ce chapitre fournit une description des solutions apportées à l'outil *GenSpec*. Pour chaque fonctionnalité, nous avons explicité le cheminement suivi afin d'implémenter les deux solutions.

Nous avons ensuite parlé de l'intégration de ces solutions, tant du point de vue des changements aux fonctions préexistantes, que du point de vue des changements aux interfaces et aux droits des classes d'utilisateurs.

Le chapitre suivant consistera à évaluer notre solution technique via l'énumération des avantages et des limites de chacune des fonctionnalités. Il discute également des avantages du mode de développement que nous avons suivi et des problèmes qui ont été rencontrés.

Chapitre 9

Evaluation

Dans un premier temps, ce chapitre a pour but d'évaluer la solution technique explicitée au chapitre précédent. Nous énumérons les avantages et les limites pour chacune des fonctionnalités implémentées. Dans un deuxième temps, nous évaluons le mode de développement suivi afin d'implémenter ces deux fonctionnalités. Pour cela, nous explicitons les avantages et les problèmes rencontrés suite à l'adoption d'un tel mode de développement.

9.1 Évaluation de la solution technique

Tout d'abord, nous présentons les avantages et les limites de la solution apportée par la première fonctionnalité, à savoir l'ajout de types et d'énoncés par défaut. Ensuite, nous utilisons le même canevas afin d'évaluer la solution apportée par la deuxième fonctionnalité, la génération de la SEI sur base de la SES.

9.1.1 Ajout de types et d'énoncés par défaut

Avantages

La solution apportée est susceptible de fournir plusieurs avantages :

- Diminution du risque de redondance et d'incohérence,
- Complétude des documents d'exigences,
- Structuration des exigences,
- Amélioration continue de la qualité,
- Accélération du processus d'IE.

Diminution du risque de redondance et d'incohérence

En utilisant un OSIE, il arrivait parfois qu'un utilisateur ajoute une exigence, déjà entrée par un autre utilisateur en la formulant de manière différente. Cela arrivait car l'utilisateur en question ne retrouvait pas d'exigence correspondant à celle qu'il voulait formuler. Grâce à la hiérarchie de types mise en place, l'utilisateur sait désormais où chercher précisément pour savoir s'il existe déjà une exigence semblable à celle qu'il veut exprimer. Grâce à cette diminution du risque de redondance, les incohérences liées à la modification d'exigences peuvent également être réduites. De plus, grâce au processus d'amélioration continue des types et des énoncés, les utilisateurs s'accorderont sur la manière de formuler une exigence.

Complétude des documents d'exigences

La solution que nous avons implémentée permet d'être plus complet à deux niveaux. Premièrement, lorsque l'utilisateur consulte l'arbre des types, cela peut lui faire penser à des exigences qu'il n'avait pas envisagées. Deuxièmement, les énoncés par défaut qui ont été composés sur base de plusieurs documents et selon le respect de plusieurs règles aident

l'utilisateur à formuler des exigences de manière plus claire et plus complète.

Structuration des exigences

Grâce aux différents types par défaut présents dans *GenSpec*, il est possible de réaliser un document d'exigences bien structuré et avec plusieurs niveaux de regroupement des exigences. La hiérarchie des types d'exigences est bien équilibrée (même niveau de granularité) afin de faciliter la tâche de la personne chargée de rédiger les exigences. De plus, une bonne structure facilite la lecture et la compréhension du client et des concepteurs.

Amélioration continue de la qualité

Grâce au module de gestion de types, qui permet de modifier les informations relatives à un type dont l'énoncé par défaut, il est possible d'améliorer les énoncés grâce aux diverses remarques des utilisateurs et des clients pour arriver à des énoncés de plus en plus simples, clairs, concis et complets. De la même manière, on peut également améliorer la hiérarchie des types.

De plus, en ce qui concerne les énoncés générés, si l'utilisateur consulte les règles que nous avons regroupées, il formulera des énoncés plus clairs, plus simples, plus concis et plus complets. Ceci constitue donc un premier pas vers la gestion des connaissances puisque, petit à petit, on sélectionnera et adaptera les types d'exigences et les énoncés par défaut les plus utiles, les plus pertinents.

Accélération du processus d'IE

Grâce à la génération automatique des énoncés, du titre, des notes et des commentaires lors de la sélection d'un type, la réalisation des documents d'exigences est accélérée. Non seulement car l'utilisateur n'a plus qu'à compléter les énoncés, si cela est nécessaire, mais aussi car il lui est plus facile d'organiser les exigences grâce à la hiérarchie des types. De plus, ce processus ouvre la porte à la réutilisation des exigences. Tout comme nous l'avons mentionné ci-dessus, cela permet une première gestion des connaissances, point qui faisait défaut lors de l'analyse du processus de GL de l'unité (cf section 4.5.4).

Limites

L'ajout de types d'exigences et d'énoncés par défaut peut être néfaste si l'utilisateur commence à entrer un nombre conséquent de types par défaut ou des types inadéquats. Le document d'exigences aura une granularité trop précise qui pourrait également entraver sa qualité. Par exemple, un utilisateur ajoute les types d'exigences "Windows", "Linux", "Mac OS" en-dessous d'un type concernant la compatibilité avec les systèmes d'exploitation. Pourtant, il aurait suffit d'utiliser le type plus général "Système d'exploitation" et d'énumérer les différents systèmes d'exploitation.

Il faut veiller à ce que les nouveaux types d'exigences et/ou énoncés par défaut soient clairs, concis et précis. Si l'utilisateur entre des types incohérents et/ou des énoncés par défaut mal construits, cela annihilerait les effets bénéfiques de la solution implémentée.

De plus, la possibilité d'ajouter de nouveaux types d'exigences peut paraître rapidement superficielle pour les utilisateurs de *GenSpec* et elle deviendrait donc rapidement obsolète. En effet, certaines personnes n'utilisent que très peu de types d'exigences pour réaliser leur document d'exigences. Elles estiment qu'il n'est pas nécessaire de disposer d'un tel niveau de granularité. Dès lors, la possibilité d'ajouter ou de modifier des types d'exigences ne leur semble pas indispensable.

Cette solution ne garantit pas que l'utilisateur choisira le type d'exigences qui convient ou qu'il formulera correctement son énoncé. Il s'agit, dans un sens, de maximiser les chances

de bonne structuration et de rédaction des documents d'exigences.

L'interface d'ajout d'une exigence pourrait également être améliorée en regroupant la saisie des données en une seule fenêtre au lieu de plusieurs, comme c'est le cas actuellement.

Le logiciel n'offre pas, à l'heure actuelle, la possibilité de conserver une liste de types d'exigences propre à un utilisateur, à un projet ou encore à un domaine d'application afin de la réutiliser. Il faudrait également permettre l'importation d'une liste à partir de documents de type XML, Word, etc.

En conclusion, l'ajout de cette nouvelle fonctionnalité est majoritairement bénéfique à condition de ne pas abuser de l'ajout de types et d'utiliser à bon escient les nouvelles options implémentées. Bien évidemment, il ne s'agit pas ici de la solution parfaite et les conséquences qu'elle peut engendrer, en cas de mauvaise utilisation (inflation des types d'exigences, énoncés par défaut mal structurés, etc.), peuvent s'avérer néfastes.

9.1.2 Génération de la SEI sur base de la SES

Avantages

La solution apportée est susceptible de fournir plusieurs avantages :

- Accélération du processus d'IE,
- Cohérence entre SES et SEI,
- Complétude des documents d'exigences,
- Traçabilité entre SES et SEI.

Accélération du processus d'IE

Grâce à la génération automatique de la SEI sur base de la SES, la réalisation des documents d'exigences est accélérée. L'utilisateur n'a plus qu'à compléter les énoncés, si cela est nécessaire. Auparavant, les éléments concernant le format de chaque intrant/extrant étaient rédigés de manière manuelle. Grâce à la solution apportée, tout peut se faire au moyen de l'outil *GenSpec*. De ce fait, les délais et coûts de réalisation d'une SEI peuvent être diminués.

Cohérence entre SES et SEI

Grâce à la génération automatique de la SEI, il est plus aisé de maintenir la cohérence entre la SES et la SEI. Si certaines exigences sont modifiées dans la SES, il suffit de regénérer la SEI et de compléter la version précédemment générée. Le fait que les deux documents soient dans un même projet permet également à l'utilisateur de pouvoir consulter de manière totalement parallèle la SES et la SEI. Les erreurs de cohérence sont, de ce fait, plus facilement détectables.

Complétude des documents d'exigences

La solution implémentée permet d'être plus complet lors de la rédaction des documents d'exigences. Tout d'abord, la génération des liens permet de produire des documents où il ne manque aucun lien entre SES et SEI. Ensuite, la génération automatique des exigences de format de la SEI pour chaque exigence d'intrants/extrants de la SES permet à l'utilisateur de ne pas en oublier. Enfin, à l'aide des vérifications automatiques, l'utilisateur peut s'assurer de la complétion des documents.

Traçabilité entre SES et SEI

Un dernier avantage évident de la solution que nous avons apportée est une amélioration au niveau de la traçabilité entre SES et SEI. Les liens qui sont générés dans la SEI permettent

de retrouver l'exigence correspondante dans la SES et il en est de même dans le sens inverse. La lecture et la compréhension du lecteur sont donc également facilitées.

Limites

Premièrement, la solution de génération automatique de la SEI se limite, en tout cas pour l'instant, à la partie "Données spécifiques au système" de celle-ci. La SEI, y compris la partie concernant les exigences spécifiques, comprend d'autres parties et sous-parties qui doivent être réalisées de manière habituelle.

Deuxièmement, si la SES est mise à jour, il est possible de regénérer les SEI correspondantes, mais il faudrait ajouter un mécanisme permettant de détecter si ces SEI existent déjà et mettre à jour uniquement les exigences de la SEI correspondant à des exigences de la SES qui ont été mises à jour. Actuellement, il est uniquement possible de tout générer.

9.2 Évaluation du mode de développement

Nous allons maintenant évaluer le mode de développement suivi afin d'implémenter la solution technique. Nous allons voir ses avantages et les problèmes rencontrés.

9.2.1 Avantages du mode de développement

Le mode de développement que nous avons suivi possède plusieurs avantages. Ceux qui nous ont paru être les plus importants au cours du développement sont :

- Utilisation d'un planning précis,
- Réalisation d'un état de l'art,
- Échanges des fonctionnalités,
- Utilisation d'un mode de développement itératif et incrémental,
- Phases de tests conséquentes.

Utilisation d'un planning précis

Le planning que nous avons élaboré en début de projet fut respecté à la lettre. En effet, le projet fut terminé dans les délais qui avaient été fixés. Le planning était très précis et le suivi du projet via l'outil Microsoft Office Project nous a permis d'atteindre nos objectifs.

Réalisation d'un état de l'art

L'état de l'art que nous avons réalisé en analysant plusieurs OSIE et documents disponibles au public nous a permis de trouver des pistes concernant la problématique décrite au chapitre 5. Cela nous a permis également d'améliorer notre compréhension des besoins des différentes parties prenantes.

Échanges des fonctionnalités

Comme nous l'avons déjà dit précédemment (cf section 7.1.2), la personne chargée des phases d'analyse des besoins et de vérification d'une fonctionnalité était chargée des phases de conception et d'implémentation de l'autre fonctionnalité. Cette manière de procéder oblige la personne responsable de l'élaboration des documents d'exigences à fournir des documents de qualité afin que l'autre personne puisse comprendre et implémenter les exigences décrites dans ces documents. De plus, l'objectivité des tests lors de la phase de vérification est d'autant plus grande que la personne réalisant les tests d'une fonctionnalité n'est pas celle ayant implémenté celle-ci.

Utilisation d'un mode de développement itératif et incrémental

Comme nous l'avons dit précédemment (cf section 7.2.3), nous avons suivi un mode de développement itératif et incrémental. Cela nous a permis, entre autres, de produire de nouvelles versions stables de l'outil *GenSpec* à chacun de ces incréments ou itérations. De plus, contrairement à certains de nos prédécesseurs ayant participé au développement de *GenSpec*, nous avons inclus la phase d'analyse des besoins dans la boucle du processus. En effet, il est arrivé par le passé que la phase d'analyse des besoins soit réalisée en totalité avant de commencer les incréments ou itérations. Cela pouvait déboucher sur un nombre trop important d'exigences dès lors difficiles à réaliser si on ne définissait pas une priorité d'implémentation. Ce mode de développement permet également de pouvoir affiner notre compréhension du problème pendant le développement. Enfin, il permet de faire plus facilement face à la volatilité des exigences (cf section 2.1.2).

Phases de tests conséquentes

Comme nous l'avons dit dans la sous-section 9.2.1, il est arrivé que la phase d'analyse des besoins ne soit pas intégrée à la boucle du mode de développement. Cette pratique peut déboucher sur des phases de tests trop restreintes. En effet, le nombre d'exigences étant plus difficile à gérer, l'implémentation de celles-ci peut être très consommatrice de temps. Les phases de tests sont alors réalisées de manière peu approfondie ce qui peut déboucher sur une version instable de l'outil. Grâce à notre mode de développement et au planning que nous avons réalisé, les phases de tests constituèrent près de 40% de notre temps de développement. Grâce à ces phases de tests conséquentes, les versions successives de *GenSpec* que nous avons élaborées étaient plus stables et les bugs moins nombreux. Toutefois, les corrections occupaient environ 65% de notre temps dédié à l'intégration et aux tests.

9.2.2 Problèmes rencontrés

Lors de notre développement, deux principaux problèmes ont été rencontrés : la volatilité des exigences et le manque de documentation du code.

La volatilité des exigences

Comme il a été décrit à plusieurs reprises dans le chapitre 2 (cf section 2.1.2), les exigences peuvent changer en cours de développement. Certaines peuvent être ajoutées, modifiées ou encore supprimées. Lors de notre développement, il est arrivé à plusieurs reprises de devoir modifier les documents d'exigences alors que nous étions dans une phase ultérieure du développement.

Le changement des exigences est un problème récurrent de l'IE et il est à constater que plus les changements se produisent tard dans le développement, plus il est coûteux de les appliquer. Un changement demandé lors de l'étape de conception requiert beaucoup moins d'efforts qu'un changement demandé lors de l'implémentation et de l'intégration.

Toutefois, ce problème fut minimisé grâce à deux aspects de notre mode de développement :

- **Des exigences bien définies** : à chaque incrément ou itération, le comité *GenSpec* et plus particulièrement monsieur René Bujold était chargé de valider chaque document d'exigence et de conception. Ce processus de validation entrainait plusieurs corrections de ces documents et les exigences étaient donc clairement définies et comprises par l'ensemble des parties prenantes.

- **Un ensemble restreint d'exigences pour chaque incrément ou itération** : notre mode de développement étant itératif et incrémental, l'ensemble des exigences à définir à chaque incrément ou itération était restreint. On pouvait donc se concentrer

sur ces exigences en évitant le problème d'inflation des exigences et les conséquences qu'il peut entraîner.

Le manque de documentation du code

Le logiciel *GenSpec* est fourni avec un document d'aide très complet. Il permet à l'utilisateur et aux personnes chargées du développement de comprendre le fonctionnement des différentes fonctionnalités.

Toutefois, un problème a été rencontré au niveau de la documentation du code. En effet, celle-ci ne permettait pas toujours de comprendre la manière dont les différentes fonctionnalités ont été codées. Plusieurs procédures et variables devant être utilisées ou modifiées lors de notre développement étaient non-commentées. De ce fait, il était plus difficile d'en comprendre le fonctionnement et l'utilisation ou la modification de celles-ci était plus délicate.

Le schéma de la BD réalisé par les stagiaires précédents, Nicolas Pirmez et Olivier Pire [Pire 07], démontre la complexité de celle-ci. En effet, certains noms de champs ne sont pas explicites et certaines tables ont une utilité qu'on ne peut pas cerner sans une exploration approfondie du code.

9.3 Résumé

Nous avons exprimé les avantages et les limites de chacune des solutions. En ce qui concerne l'ajout de types d'exigences et d'énoncés par défaut, nous avons vu qu'il fallait l'utiliser à bon escient et ne pas en abuser. Si celle-ci est bien utilisée, les avantages suivants peuvent être retirés :

- Diminution du risque de redondance et d'incohérence,
- Complétude des documents d'exigences,
- Structuration des exigences,
- Amélioration continue de la qualité,
- Accélération du processus d'IE.

Au sujet de la génération de la SEI sur base de la SES, même si celle-ci est encore limitée à certaines parties du document, nous avons vu que cette fonctionnalité offrait les avantages suivants :

- Accélération du processus d'IE,
- Cohérence entre SES et SEI,
- Complétude des documents d'exigences,
- Traçabilité entre SES et SEI.

Nous avons ensuite explicité les principaux avantages du mode de développement suivi :

- Utilisation d'un planning précis,
- Réalisation d'un état de l'art,
- Échanges des fonctionnalités,
- Utilisation d'un mode de développement itératif et incrémental,
- Phases de tests conséquentes.

Enfin, nous avons décrit les problèmes que nous avons rencontrés :

- La volatilité des exigences,
- Le manque de documentation du code.

Le chapitre suivant vise à conclure les idées développées au sein de ce mémoire. Il explicite également les perspectives possibles suite au travail réalisé.

Conclusions et perspectives

L'objectif principal de ce mémoire était de décrire et justifier un ensemble de solutions qui ont été apportées à l'OSIE *GenSpec*. Ces solutions visent à améliorer la qualité des documents d'exigences rédigés avec *GenSpec* ainsi qu'accélérer le processus d'IE. Nous avons également présenté l'analyse du processus de GL que nous avons eu l'occasion de réaliser dans l'unité CA d'HQ. Cette dernière nous a notamment permis de découvrir et comprendre les problèmes que nous devions résoudre.

GenSpec offrait la possibilité d'utiliser des types d'exigences. Toutefois, les types disponibles étaient trop peu nombreux et n'offraient pas un niveau de précision assez élevé. Pour certains types, un début d'énoncé pouvait être généré mais celui-ci était incomplet et ne guidait pas assez l'utilisateur. Des problèmes dans la qualité des documents d'exigences pouvaient alors survenir. Parmi ceux-ci, l'incomplétude, l'incohérence, le manque d'uniformisation et le manque de structuration des exigences.

GenSpec ne permettait pas de rédiger clairement les SEI relatives à une SES dans le même projet que celle-ci. Généralement, des projets différents étaient créés pour la SES et les différentes SEI. Les liens entre SES et SEI étaient réalisés manuellement et cela engendrait une grande perte de temps lors de la mise à jour de l'un de ces documents. Cette pratique pouvait engendrer des problèmes de cohérence, de complétude et de traçabilité des documents.

Afin de minimiser ces problèmes, nous avons développé différentes améliorations.

Pour augmenter la qualité des documents d'exigences, nous avons ajouté un ensemble de types d'exigences et d'énoncés par défaut. Nous avons également implémenté un module de gestion de ces types et énoncés. Désormais, lorsque l'utilisateur crée une nouvelle exigence pour laquelle il choisit un type, un énoncé est généré. Les énoncés par défaut que nous avons rédigés respectent des règles de rédaction permettant d'obtenir des énoncés uniformes, simples et compréhensibles. L'ensemble des types que nous avons ajoutés est complet et facilite la structuration des exigences. L'utilisateur, est mieux guidé dans la rédaction des exigences avec *GenSpec* et le processus d'IE est accéléré.

Pour faciliter la rédaction d'une SEI, nous avons développé une fonctionnalité de génération automatique d'une première mouture de celle-ci sur base de la SES. Grâce à cette amélioration, l'utilisateur peut générer les différentes SEI relatives à une SES dans le même projet que cette dernière. Dans le cadre de cette fonctionnalité, nous avons également implémenté un module de gestion des références bibliographiques. Ces possibilités permettent notamment de réduire les erreurs de cohérence et de faciliter la traçabilité entre les documents d'exigences.

Nous avons identifié différentes possibilités d'améliorations ultérieures.

Premièrement, la création d'une nouvelle interface d'ajout d'une exigence. La création des exigences requiert actuellement l'affichage de deux fenêtres alors que l'entrée des diverses informations pourrait être réalisée en une seule fois. C'est pourquoi il faudrait sans doute résumer la création d'une exigence en une seule interface en demandant l'entrée des diverses informations nécessaires (titre de l'exigence, description, type, priorité, etc.). Cela faciliterait le travail de l'utilisateur et la présentation de *GenSpec* en serait également améliorée.

Deuxièmement, la gestion d'une liste de types d'exigences. Grâce à cela, une liste de types d'exigences personnelle à un utilisateur, relative à un projet ou encore un domaine d'application pourrait être mise en place. Cela permettrait à certains analystes de conserver leur liste des types les plus utilisés. A un plus haut niveau, une liste de types d'exigences par projet permettrait, par exemple, de mettre en évidence les types d'exigences utilisés par chacun des acteurs concernés par le projet. Via une harmonisation de ceux-ci, la cohérence du document d'exigences pourrait être améliorée.

Troisièmement, l'importation de nouveaux types d'exigences et d'énoncés par défaut. Il serait intéressant de pouvoir importer de nouveaux types d'exigences et leurs énoncés associés à partir de différents documents (XML, Word, Excel, PDF, PostScript, etc.). Cela permettrait de ne pas devoir entrer manuellement tous les types et énoncés surtout quand ceux-ci sont nombreux.

Enfin, la possibilité de générer plus simplement une SEI lors d'une mise à jour de la SES. En effet, si la SES est mise à jour, il serait idéal de pouvoir regénérer uniquement les exigences devant être mises à jour dans les SEI correspondantes.

Comme nous avons eu l'occasion de le préciser dans ce mémoire, *GenSpec* est un OSIE en constante évolution. Plusieurs améliorations sont planifiées. L'une d'elles a été proposée dans le cadre de notre analyse du processus de GL de l'unité CA : la gestion des risques. En effet, nous pensons que *GenSpec* devrait offrir des fonctionnalités permettant de gérer les risques liés à un projet. Ceux-ci devant être identifiés dès l'étape d'analyse des besoins du processus.

De plus, nous pensons que *GenSpec* devrait être adapté afin de pouvoir être utilisé sur tout système d'exploitation, ce qui n'est pas le cas actuellement.

Il est à constater que, grâce à notre contribution, *GenSpec* dispose de fonctionnalités supplémentaires permettant de gérer des types d'exigences et leurs énoncés associés, de gérer des références bibliographiques et de générer une SEI sur base d'une SES. Ces améliorations permettent d'augmenter la qualité des documents d'exigences et d'accélérer le processus d'IE.

Un article, concernant les problèmes identifiés et les solutions apportées ainsi que les avantages de ces dernières, doit paraitre dans la revue "Génie Logiciel" au Canada d'ici la fin de l'année 2008. Il a été co-écrit par les auteurs de ce mémoire, Delphine Harmel et Guillaume Cavillot, mais également par monsieur René Bujold. Cet article est disponible dans l'annexe K.

Bibliographie

[Audibert 08] Laurent Audibert. Uml 2. http ://laurent-audibert.developpez.com/Cours-UML/html/Cours-UML.html, 2007-2008.

[Berry 07] Daniel M. Berry, Sri Fatimah Tjong & Michael Hartley. *Extended Disambiguation Rules for Requirements Specifications*, Novembre 2007.

[Borderie 06] Xavier Borderie. Expliquez-moi... le CMMi. http ://www.journaldunet.com/developpeurs/tutoriel/theo/060524-model-cmmi.shtml, 2006.

[Cen 08] Centre d'Excellence en Technologies de l'Information et de la Communication. *Ingénierie des Exigences*, 2008.

[Chambers 06] Les Chambers. *Modelling Software Requirements.* Rapport technique 1.01, Chambers and Associates Pty Ltd, 2 Juillet 2006.

[Cockburn 01] Alistaire Cockburn. Writing effective usa cases. Addison-Wesley, 2001.

[Coulin 07] Chad Raymond Coulin. *A Situational Approach and Intelligent Tool for Collaborative Requirements Elicitation.* Thèse soumise pour l'obtention du grade de docteur en philosophie, Université Paul Sabatier, Laboratoire d'Analyse et d'Architecture des Systèmes, 2007. Centre National de la Recherche Scientifique.

[Crinnion 91] John Crinnion. Evolutionary systems development : A practical guide to the use of prototyping within a structured systems methodology. Plenum Press, New York, USA, 1991. Page 18.

[Davis 92] Alan M. Davis. *Operational Prototyping : A new Development Approach.* IEEE Software Institute, Septembre 19992.

[Dening 04] Anna Dening. *A Study of Concepts of Knowledge Management.* Rapport technique, University of York, York, Grande-Bretagne, Mars 2004.

[Dir 04] Règles de Structure et de Rédaction des Normes Internationales, numéro Partie 2, 2004. Directives ISO/CEI.

[Easterbrook 04a] Steve Easterbrook. *Estimation and Prioritization.* Université de Toronto, Département des Sciences, Toronto, Canada, 2004.

[Easterbrook 04b] Steve Easterbrook. *Verification and Validation.* Université de Toronto, Département des Sciences, Toronto, Canada, 2004.

[Easterbrook 05] Steve Easterbrook. *Requirements Engineering.* Université de Toronto, Département Informatique, Ontario, Canada, 2005.

[Firesmith 03] Donal G. Firesmith. *Modern Requirements Specification.* Journal of Object Technology, Mars 2003.

[Firesmith 04] Donal G. Firesmith. *Prioritizing Requirements.* Journal of Object Technology, vol. 3, no. 8, Septembre-Octobre 2004.

[Gen 92] General Accounting Office, USA. *Etude du General Accounting Office*, 1992.

[Goodrich 90] Victoria Goodrich & Lorne Olfman. *An Experimental Evaluation of Task and Methodology Variables for Requirements Definition Phase Success*. page 202. IEEE Computer Society, Janvier 1990. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences.

[Grosjean 07] Jean-Claude Grosjean. Spécifications, exigences et cahier des charges : Quoi, comment ? Ergonomie Informatique, Agilité et Qualité logiciel, 2 juillet 2007.

[Haag] Steven Haag, Maeve Cummings, Donald J. McCubbrey, Alain Pinsonneault & edition = 3ème year = 2006
Richard Donovan" address = New York, USA. *Management Information Systems for the Information Age*.

[Habra 07] Naji Habra. *Ingénierie du Logiciel : Vérification et Validation*. Facultés Universitaires Notre-Dame de la Paix, Namur, Belgique, 2006-2007.

[Hainaut 03] Jean-Luc Hainaut. *Ingénierie des Bases de Données*. Facultés Universitaires Notre-Dame de la Paix, Namur, Belgique, 2003.

[Heymans 06] Patrick Heymans. *Analyse et Modélisation des Systèmes d'Information*. Facultés Universitaires Notre-Dame de la Paix, Namur, Belgique, 2006.

[Hickey 03] Hickey & Davis. *Requirements Elicitation and Elicitation Technique Selection : A Model for Two Knowledge-Intensive Software Development Processes*. Big Island, Hawaii, 6-9 Janvier 2003. 36th Annual Hawaii International Conference on System Sciences.

[Hugues 02] Anne-Marie Hugues. La problématique du logiciel, 19 Décembre 2002.

[IEE 90] *IEEE Standard Glossary of Software Engineering Terminology*, ieee std 610.12-1990 edition, Décembre 1990.

[IEE 98] *IEEE Standard for Software Verification and Validation*, IEEE std 1012-1998 edition, 20 juillet 1998.

[IEE 04] IEEE Computer Society Professional Comittee, Los Alamitos, Californie, USA. *Guide to the SoftWare Engineering Body Of Knowledge*, 2004.

[INC 07] INCOSE Requirements Management Working Group. *INCOSE Requirements Management Tools Survey*, 2007.

[Int 95] International Standardisation Organisation, USA. *Standard for Information Technology - Software Life Cycle Processes*, iso/iec 12207.1995 edition, Août 1995.

[Int 00] International Standardisation Organisation. *Quality Management Systems - Requirements*, 2000.

[Jaton 07] Markus Jaton. *Chapitre 11 : UML, les diagrammes de composants*. Institut de Télécommunications de l'EIVD, 2007.

[Leite 87] Julio Cesar Leite. *A Survey on Requirements Analysis*. Rapport technique Advanced Software Engineering Project Technical Report RTP-071, Université de Californie, Département d'Information et Informatique, Juin 1987.

[Martin 84] James Martin. An Information Systems Manifesto. Prentice Hall, New Jersey, Janvier 1984.

[Mascaro] Gabriella Mascaro, Allan Elder, Debra Haley & al.

[Matulevicius 05] Raimundas Matulevicius. *A Requirements Engineering Tool Evaluation Approach*, Juin 2005.

[Mead 06] Nancy R. Mead. *Requirements Prioritization Introduction.* Rapport technique, Université Carnegie Mellon, 26 Septembre 2006.

[Michael 92] G. Christel Michael & Kyo C. Kang. *Issues in Requirements Elicitation.* Rapport technique, Software Engineering Institute, Carnegie Mellon, Septembre 1992.

[Mittermeir 90] Roland T. Mittermeir, Nicholas Roussopoulos, Raymond T. Yeh & Peter A. Ng. An Integrated Approach to Requirements Analysis, chapitre 5, page 121. Reinhold Van Nostrand, New York, 1990.

[Morley 97] Chantal Morley. Gestion de projets informatiques. Université de Namur, Belgique, 1997.

[NASA 08] NASA. Vérification et validation. http ://satc.gsfc.nasa.gov/assure/agbsec5.txt, Date d'accès : 13 Mars 2008.

[Pire 07] Olivier Pire & Nicolas Pirmez. Améliorations de l'outil d'Ingénierie des Exigences genspec : Glossaire, gestion multi-documents et allocations. Master's thesis, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgique, Juin 2007.

[Pro 04] Project Management Institute. *A Guide to the Project Management Body Of Knowledge (PMBOK Guide)*, 3e edition, 2004.

[Product Team 07a] CMMi Product Team. *Capability Maturity Model Integration (CMMI) Version 1.2 Overview.* Rapport technique, Software Engineering Institute, 2007.

[Product Team 07b] CMMi Product Team. *Capability Maturity Model Integration for Acquisition, Version 1.2.* Rapport technique, Software Engineering Institute, November 2007.

[Ratté 07] Harold Ratté. *Définition du contenu et du gabarit de l'artefact SEI.* Rapport technique, Unité Conception-Automatismes, Hydro-Québec, Montréal, Canada, 2007.

[Robertson 99] Suzanne Robertson & James Robertson. *Mastering the Requirements Process*, 1999.

[Robertson 06] Suzanne Robertson & James Robertson. *Volere Requirements Specification Template.* The Atlantic Systems Guild Limited, 11e edition, Février 2006.

[Royce 70] Dr. Winston W. Royce. *Managing the Development of Large Software Systems.* Proceedings, pages 1–9, Août 1970.

[Rzepka 89] William E. Rzepka. *A Requirements Engineering Testbed : Concept, Status, and First Results.* pages 339–347. IEEE Computer Society, 1989. Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences.

[Selic 03] Bran Selic. *The pragmatics of Model-Driven Development.* IEEE Software Institute, Septembre/Octobre 2003.

[Sof 98a] Software Engineering Standards Committee of the IEEE Computer Society, USA. *IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document*, IEEE std 1362-1998 edition, 1998.

[Sof 98b] Software Engineering Standards Committee of the IEEE Computer Society, USA. *IEEE Recommended Practice for Software Requirements Specifications (SRS)*, IEEE std 830-1998 edition, 20 Octobre 1998.

[Solnon 97] Christine Solnon. Ateliers de génie logiciel, 1996-1997.

[Sommerville 97] Ian Sommerville & Pete Sawyer. Requirements Engineering : A good practice guide. John Wiley & Sons, 1997.

[SPC 97] *Evolutionary Rapid Development.* SPC-97057-CMC, page 6, Herndon, Juin 1997. Software Productivity Consortium.

[Sta 95] The Standish Group. *The Standish Group Report CHAOS*, 1995.

[Sta 04] Standish Group, West Yarmouth, Massachusetts. *CHAOS Reports*, 1994-2004.

[Storey 96] Neil Storey. Safety critical computer systems. Addison-Wesley, Harlow, Angleterre, 1996.

[Tagger 05] Ben Tagger. *Software Requirements Specification for Grid 3D Application.* Rapport technique Version 1.0, National Institute for Medical Research Mill Hill, Londres, Grande-Bretagne, 4 Mars 2005.

[TCS 99] *Tactical Control System (TCS)*, 12 Février 1999. System/Subsystem Specification Version 2.0.

[Thiran 05] Philippe Thiran. *Conception des Systèmes d'Information.* Facultés Universitaires Notre-Dame de la Paix, Namur, Belgique, 2004-2005. Programme DGTIC.

[Tran 99] Eushiuan Tran. *Verification - Validation - Certification.* Université Carnegie Mellon, Printemps 1999.

[Uni 04] Unité Conception-Automatismes, Hydro-Québec, Montréal, Canada. *SEI TDST - Spécification des exigences d'interface du télé-délestage en sous-tension - Utilisateur TR*, Mai 2004.

[U.S 95] U.S Army, USA. *Interface Requirements Specification (IRS)*, di-ipsc-81434, mil-std-498 edition, 1995.

[Vincelette 06] Michel Vincelette. *Méthode de Travail, Support au Processus de Génie Logiciel.* Rapport technique MET10005a, Unité Conception-Automatismes, Hydro-Québec, Montréal, Canada, Décembre 2006.

[Vincelette 07] Michel Vincelette. *Pratiques Générales.* Rapport technique MET10200, Unité Conception-Automatismes, Hydro-Québec, Montréal, Canada, Mai 2007.

[Wallonne 04] Région Wallonne. Boîte à outils : Méthodes d'implication des utilisateurs. Wall-on-line : l'e-gouvernement wallo, 17 Décembre 2004.

[Wiegers 99] Karl E. Wiegers. Software requirements. Microsoft Press, Redmond, WA, 2ème edition, 1999.

[Wikipédia 08] Wikipédia. Software Engineering. Date d'accès : 13/03/2008, Mars 2008.

[Young 01] Ralph R. Young. Effective Requirements Practices. Addison-Wesley Information Technology Series, 2001.

[Yu 05] Eric Yu. *Strategic Actors Modeling for Requirements Engineering - the i* framework*, Avril 2005.

[Zelkowitz 95] Marvin Zelkowitz. *Advances in Computers.* Université du Maryland, 1995.