MASTER EN SCIENCES INFORMATIQUES

Amélioration des pratiques logicielles dans les contaxtes Agile

Robaeys, Antoine

Award date:
2007

Awarding institution:
Universite de Namur

FUNDP - Namur
Institut d'Informatique
Rue Grandgagnage, 21
B-5000 Namur

Amélioration des Pratiques Logicielles dans les contextes Agile

Antoine Robaeys

Mémoire présenté en vue de l'obtention du grade de
Maître en Informatique

Juin 2007

Année académique 2006 - 2007

Résumé

Afin d'améliorer la qualité de leur développement logiciel, les PME ne peuvent utiliser facilement des modèles, tels que CMMI et SPICE, dont la taille et la complexité rendent l'utilisation lourde et coûteuse. Le modèle OWPL a dès lors été développé afin de permettre à ces petites structures de réaliser une démarche d'évaluation et d'amélioration de leurs processus logiciels.

Cependant, OWPL a été initialement conçu sur base des méthodes disciplinées, mais les pratiques Agile sont de plus en plus répandues auprès des PME, et une adaptation du modèle OWPL à ces pratiques Agile s'est donc avérée nécessaire.

Dans ce mémoire, nous décrirons notre expérimentation du modèle OWPL avec deux petites entreprises québécoises. Ensuite, nous proposerons une adaptation du modèle OWPL dans les contextes Agile. Cette adaptation permet de pouvoir évaluer la proportion agilité-discipline actuelle de l'organisation afin de l'améliorer en tenant compte de son contexte et de ses objectifs.

Abstract

In order to improve their software development quality, VSE[a], can't easily use models such as, CMMI and SPICE, with a size and complexity that make utilization hard and expensive. Therefore OWPL model was developed in order to allow these VSE to perform a software process assessment and a software process improvement.

However, OWPL was initially based on plan-driven methods, but Agile practices are more and more used by VSE, and an improvement of the OWPL model with these practices is needed.

In this thesis, we'll describe our experimentation of the OWPL model with two VSE from Quebec. Then, we'll provide an improvement of the OWPL model with the Agile context. This adaptation allows to assess the current agility/plan-driven balance of the organization in order to improve it to better fit the organization's context and goals.

[a]Very Small Enterprises

Remerciements

Nous souhaitons adresser tous nos remerciements aux personnes qui nous ont apporté leur aide et qui ont ainsi contribué à la réalisation de ce mémoire.

Tout d'abord, nous remercions le Professeur Naji Habra, promoteur du mémoire, pour son soutien, son aide et ses conseils avisés, tout au long du stage et de la rédaction de ce document.

Nous remercions également le Professeur Jean-Marc Desharnais, maître de stage, et le doctorant Mohammad Zarour, pour leur dévouement, leurs conseils et leur accompagnement lors de notre stage à l'Ecole de Technologie Supérieure (ETS) de Montréal, Québec, Canada.

Merci à Christian Dubois, co-fondateur de l'entreprise Logilys, pour sa collaboration qui nous a permis de faire avancer nos recherches.

Merci à Simon Alexandre, consultant au CETIC, pour ses conseils et sa collaboration, ainsi qu'à Pascal De Vincenzo, gestionnaire de projets chez Open Engineering, pour avoir accepté de répondre à nos questions sur les pratiques logicielles de cette PME.

Enfin, nous adressons nos plus sincères remerciements à notre famille et à tous nos proches, pour leur soutien et leurs encouragements.

Table des matières

Table des figures

Acronymes

APL Amélioration des Pratiques Logicielles.

ASD Adaptive Software Development.

CETIC Centre d'Excellence en Technologies de l'Information et de la Communication - Wallonie, Belgique.

CIC Cadre des Instructeurs de Cadets.

CMMI Capability Maturity Model Integration.

DSDM Dynamic Systems Development Method.

ESA European Space Agency.

ETS Ecole de Technologie Supérieure - Québec, Canada.

FDD Feature Driven Development.

IEC International Electrotechnical Committee.

ISO International Organization for Standardization.

KPA Key Process Area.

LQL Laboratoire de Qualité Logiciel de l'Université de Namur.

MDN Ministère de la Défense Nationale.

MEMS Micro Electrical Mechanical Systems.

OOFELIE Object Oriented Finite Element Led by Interactive Executor.

OWPL Observatoire Wallon des Pratiques Logicielles.

PME Petites et Moyennes Entreprises.

RUP Rational Unified Process.

SEI Software Engineering Institute.

SPA Software Process Assessment.

SPI Software Process Improvement.

SPICE Software Process Improvement and Capability dEtermination.

SRS Software Requirement Specification.

SW-CMM Capability Maturity Model for Software.

TPE Très Petites Entreprises.

UML Unified Modelling Language.

URSC-E Unité Régionale de Soutien des Cadets de la Région de l'Est.

Introduction

Dans l'ingénierie du logiciel, les méthodes *disciplinées* sont généralement considérées comme garantes d'un bon développement pour la production d'un logiciel de qualité. A l'origine, ces méthodes approchent le développement suivant un paradigme plus ou moins linéaire et en cascade qui consiste d'abord en l'analyse des exigences suivie de la conception, du codage, des tests et de la maintenance. Lors du développement, il y a un souci de documentation à chacune des étapes qui s'accompagne de standards et de processus bien définis que l'organisation supporte et améliore continuellement. Plus récemment, des processus incrémentaux ont été adoptés par ces méthodes disciplinées mais toujours avec une forte documentation et traçabilité des exigences, de la conception et du code.

Les méthodes disciplinées constituent une base traditionnelle pour toute tentative de succès. Sans la rigueur des ingénieurs, il est possible de réussir occasionnellement grâce au talent naturel, mais la consistance professionnelle et les perspectives à long terme sont limitées. La discipline fournit une certaine force et un certain confort qui permettent de supporter le travail lorsque les choses deviennent difficiles ou qu'un événement non attendu se présente.

Alors que la discipline renforce, l'agilité libère et invente. Elle permet aux ingénieurs de s'adapter aux technologies et aux besoins changeants. Les méthodes *Agile* s'ajustent aux nouveaux environnements, réagissent et s'adaptent, tirent profit des opportunités non attendues, et mettent à jour la base d'expérience pour le futur.

L'environnement dans lequel les logiciels sont conçus a changé. Les systèmes logiciels sont de plus en plus larges et complexes, les composants commerciaux prêts à l'emploi jouent des rôles plus significatifs, et les changements dans les exigences sont de plus en plus fréquents. En outre, les logiciels sont omniprésents, ce qui rend leurs aspects de qualité et d'utilisabilité encore plus critiques.

Le monde du développement logiciel traditionnel, caractérisé par l'ingénierie et les processus, gère ce changement d'environnement en comptant sur la prévoyance des architectures à développer et à appliquer. Dès lors, ces architectures peuvent souvent gérer le changement avant qu'il n'affecte le système. Ces méthodes *disciplinées* se focalisent donc sur la qualité du logiciel et surtout sur la prévision de leurs processus.

A l'inverse, les méthodes *Agile* encouragent les développeurs à abandonner les processus lourds et à accepter le changement grâce au concept d'agilité et à un esprit plutôt adaptatif que prédictif.

La plupart des modèles et standards d'évaluation des processus logiciels, qui utilisent des méthodes disciplinées, se focalisent principalement sur les moyennes et grandes entreprises, compliquant ainsi leur application effective et efficiente dans les petites entreprises de logiciels à cause de leurs caractéristiques spécifiques et de leurs limitations. En effet, les PME ne peuvent utiliser facilement des modèles, tels que CMMI et SPICE, dont la taille et la complexité rendent l'utilisation lourde et coûteuse. La structure de ces modèles n'est pas applicable aux petites entreprises car elle définit un grand nombre d'informations, de processus et d'attributs. En outre, l'intervention d'experts en amélioration des pratiques logicielles représente un coût considérable pour les PME, en terme financier, en personnes et en ressources.

Les petites entreprises de logiciels font donc face à des problèmes similaires à ceux des grandes entreprises en ce qui concerne l'amélioration de la qualité logicielle et l'évaluation des processus. La différence principale est que ces petites entreprises manquent souvent de ressources spécialisées ou de ressources compétentes pour résoudre ces problèmes.

De par des structures légères, des équipes de petites tailles, et un mode de fonctionnement différent de celui des grandes entreprises, il est nécessaire pour les PME, que les modèles traditionnels soient adaptés à leur contexte. Le modèle OWPL a dès lors été développé afin de permettre une évaluation rapide et légère des pratiques logicielles d'une organisation, et la recommandation de bonnes pratiques qui permettront l'amélioration des processus logiciels.

OWPL ayant initialement été conçu sur base des méthodes disciplinées, et, les pratiques Agile étant de plus en plus répandues auprès des PME, une approche qualité dans ces contextes semble pertinente et une adaptation du modèle OWPL à ces pratiques Agile s'avère donc nécessaire. Ces pratiques ont, dès lors, été intégrées dans une version de la micro-évaluation du modèle et la possibilité de dresser un profil « Agile » y a été ajoutée.

Cependant, l'évaluateur réalisant généralement de manière informelle l'analyse du contexte de l'organisation évaluée (taille, maturité, objectifs), de par son expertise difficile à formaliser, un guide lui permettant de pouvoir analyser plus formellement ce contexte l'aiderait à adapter correctement la méthodologie en fonction du contexte rencontré.

L'objectif de ce mémoire est double :

- Premièrement, il s'agit d'*expérimenter* le modèle OWPL avec deux petites entreprises québécoises, afin d'améliorer leur processus logiciel. Cette expérimentation servira de base au deuxième aspect de l'objectif.

- Deuxièmement, il s'agit de proposer une *adaptation* du modèle OWPL dans les contextes Agile. Cette adaptation permet de pouvoir évaluer la proportion agilité-discipline actuelle de l'organisation afin de l'améliorer en tenant compte de son contexte et de ses objectifs. En effet, de la discipline stricte sans agilité entraîne de la bureaucratie et de la stagnation, tandis que de l'agilité sans discipline traduit un enthousiasme tranquille et prématuré. Tout bon développement de logiciel dans un environnement changeant requiert, dès lors, de l'agilité et de la discipline. Une entreprise doit donc comprendre ces deux concepts dans des proportions relatives à ses objectifs et à son environnement.

Ce mémoire se divise en deux grandes parties. La première partie, « Contexte général », constitue l'état de l'art sur la qualité des processus logiciels et sur les développements Agile. Elle permet d'assurer la compréhension du sujet et d'introduire les notions utilisées dans les parties suivantes.

La deuxième partie, « Contexte spécifique », aborde dans un premier chapitre, l'expérimentation du modèle OWPL avec deux petites entreprises québécoises, et les observations déduites de cette expérimentation. Ensuite, le second chapitre décrit l'adaptation proposée du modèle OWPL dans les contextes Agile. Enfin, le troisième chapitre décrit l'expérimentation de l'adaptation proposée auprès d'une PME wallonne et la critique du travail réalisé.

Nous terminerons par la conclusion du présent mémoire et nous aborderons des pistes de travaux futurs qui pourraient aider à améliorer la démarche OWPL.

Première partie

Contexte général

Chapitre 1

Qualité des processus logiciels

1.1 Introduction

Une approche processus est une approche qui permet à une organisation de gérer ses activités et de maîtriser les interactions entre ses processus internes. Un processus est défini comme un ensemble de ressources et d'activités corrélées qui transforment des entrées en sorties, ces ressources pouvant inclure le personnel, les équipements, la technologie et la méthodologie.

Dans l'ingénierie du logiciel, un *processus logiciel* est une séquence d'étapes réalisée dans un but précis, comme, par exemple, le développement d'un logiciel.

Afin de demeurer compétitives, les entreprises ont besoin de processus de développement logiciels efficients et rentables, leur permettant de développer des produits de qualité. Il est donc nécessaire pour ces organisations de pouvoir *évaluer* et *améliorer* leurs processus logiciels [7, 26].

En vue d'améliorer les processus logiciels de l'organisation, ceux-ci doivent avoir été évalués au préalable. L'*évaluation des processus logiciels* (Software Process Assessment (SPA)) consiste en l'évaluation d'une organisation logicielle qui permet de conseiller ses gestionnaires et professionnels sur les manières d'améliorer leurs opérations. Une telle évaluation aide à détecter les secteurs qui devraient être améliorés et ceux qui fourniraient le plus de bénéfices une fois l'amélioration réalisée. Il existe trois types d'évaluation [14] :

- *Evaluation basée sur les autres organisations* : l'organisation est comparée à d'autres, ce qui permet de tirer profit des perspectives extérieures sur les pratiques et d'assurer que les bonnes pratiques sont découvertes, analysées, adoptées et mises en application.

- *Evaluation basée sur les modèles* : l'organisation est ici comparée à un ou plusieurs modèles de bonnes pratiques, appelés « modèles de maturité ».

- *Evaluation basée sur les objectifs de l'organisation* : cette approche participative se focalise sur ce qui est unique à chaque organisation, à savoir, ses besoins spécifiques et ses objectifs. Elle prend moins de temps que l'évaluation basée sur les autres organisations et est plus appropriée que celle basée sur les modèles.

L'*amélioration des processus logiciels* (Software Process Improvement (SPI)) constitue une stratégie d'intervention qui implique des changements continus et incrémentaux à l'organisation et à ses processus. Cette démarche vise donc à améliorer les processus logiciels d'une organisation en agissant sur diverses variables [14] :

- *Visibilité* : le processus est visible.
- *Discipline* : tout le monde suit le même processus comme norme.
- *Institutionnalisation* : le processus est exigé par des politiques organisationnelles.
- *Support des gestionnaires* : le processus est supporté, à la fois, par les gestionnaires de l'organisation, et, par les gestionnaires du projet de développement.

1.2 Modèles de maturité

Un modèle de maturité contient un ensemble organisé de processus et de pratiques informatiques tirés de l'état de l'art. Ce modèle constitue une vue « idéalisée » et est utilisé comme référentiel dans le cadre d'une démarche d'amélioration (SPI) ou d'évaluation (SPA) de processus.

Le succès d'une telle démarche est favorisé si l'organisation effectue, par rapport à son contexte et à ses ressources disponibles, le bon choix de référentiel parmi les différents modèles et standards disponibles sur le marché. Remarquons que le choix peut parfois être imposé par le client ou tout autre donneur d'ordre, qui souhaite que tel modèle ou tel standard soit utilisé. Il peut également être imposé par les choix préalables de l'organisation ou d'un groupe plus grand dont elle fait partie. En effet, si un modèle ou un standard particulier a déjà été adopté, la démarche d'amélioration doit s'y conformer.

1.2.1 SW-CMM 1.1

Capability Maturity Model for Software (SW-CMM) [14], développé par le Software Engineering Institute (SEI) de la Carnegie Mellon University, est un modèle qui décrit les étapes à travers lesquelles les organisations de logiciels évoluent lorsqu'elles définissent, implantent, mesurent, contrôlent et améliorent leurs processus logiciels. Il décrit aussi bien les pratiques d'ingénierie du logiciel que les pratiques de gestion de projet et de gestion de processus.

Ce modèle définit une échelle de mesure de 5 *niveaux de maturité* pour le processus logiciel d'une organisation (Figure 1.1). Un *niveau de maturité* est un niveau bien défini qui évolue vers la réalisation d'un processus logiciel mature. Chaque *niveau de maturité* contient un ensemble limité de processus (Key Process Area (KPA)) et indique un niveau de *capacité de processus*. Plus l'organisation grimpe dans les *niveaux de maturité*, plus la *capacité de processus* augmente.

La *capacité de processus* est l'ensemble des résultats attendus qui peuvent être réalisés en suivant le processus. Pour atteindre le *niveau de maturité* supérieur, il faut satisfaire tous les objectifs de tous les KPA du niveau auquel l'organisation se trouve. Le modèle spécifie 18 KPA répartis sur les 4 derniers *niveaux de maturité*. Chaque KPA identifie un sous-ensemble d'activités connexes qui doivent permettre d'atteindre des objectifs précis pour augmenter la *capacité de processus*.

FIG. 1.1 – Niveaux de maturité

1.2.2 SPICE

ISO/IEC 15504, aussi connu sous le nom de Software Process Improvement and Capability dEtermination (SPICE) [14], est un standard international développé conjointement par l'International Organization for Standardization (ISO) et l'International Electrotechnical Committee (IEC) afin de fournir une structure pour l'évaluation des processus logiciels. Cette structure peut être utilisée par des organisations impliquées dans la planification, la gestion, la supervision, le contrôle et l'amélioration de l'acquisition, l'approvisionnement, le développement, l'opération, l'évolution et le support de logiciel.

Dans sa version de 1998, ISO/IEC 15504 est divisé en 9 parties. La première partie explique les concepts et donne un aperçu de la structure.

La deuxième partie concerne le modèle de référence qui est composé de deux dimensions : une *dimension de processus* et une *dimension de capacité*. La *dimension de processus* spécifie le contenu et les objectifs mesurables des processus ainsi que les résultats attendus attestant de leur exécution. La *dimension de capacité* contient une série d'attributs des processus représentant les caractéristiques mesurables nécessaires à leur gestion et leur amélioration. Ce modèle définit 6 *niveaux de capacité* sur lesquels sont répartis les 9 attributs de processus. Un *niveau de capacité* représente donc un ensemble d'attributs qui agissent ensemble pour fournir une amélioration majeure dans la capacité à exécuter un processus, la capacité du processus étant déterminée en fonction des résultats produits par son exécution.

Les troisième et quatrième parties fournissent un guide pour réaliser une évaluation des processus. La cinquième partie fournit un modèle d'évaluation détaillé basé sur le modèle de référence. La sixième partie décrit la compétence requise par les experts pour l'évaluation. La septième partie fournit un guide pour réaliser une amélioration des processus. La huitième partie fournit un guide pour déterminer la capacité des processus d'un fournisseur. La neuvième partie contient une liste de vocabulaire.

1.2.3 CMMI

Capability Maturity Model Integration (CMMI) [14, 25], développé par des membres de l'industrie, du gouvernement, et du SEI, consiste en une structure qui intègre plusieurs modèles, concernant des disciplines différentes telles que l'ingénierie du logiciel, l'ingénierie des systèmes, la gestion des sous-traitants, et le développement de processus et de produits intégrés. Le but de CMMI est d'améliorer l'utilisation des modèles de maturité pour l'ingénierie du logiciel ainsi que pour les autres disciplines en profitant des bonnes pratiques des différents modèles.

Il existe deux types de représentations pour la structure de CMMI : *échelonnée* et *continue*. Chacune de ces représentations va permettre à une organisation de poursuivre des objectifs d'amélioration différents. Remarquons que ces deux représentations ne diffèrent que par leur présentation et non par leur contenu.

La représentation *échelonnée*, comme pour SW-CMM, fournit une séquence bien définie d'améliorations, chacune servant de base pour la suivante. Elle permet la comparaison entre organisations par l'intermédiaire des *niveaux de maturité* qui résument les résultats de l'évaluation.

La représentation *continue*, comme pour SPICE, permet de sélectionner l'ordre des améliorations afin de correspondre au mieux avec les objectifs de l'organisation et d'éviter les domaines à risques. Elle permet les comparaisons entre organisations par *domaine de processus* et ce grâce aux *niveaux de capacité* relatifs à ce domaine de processus. Un *domaine de processus* est un sous-ensemble de pratiques connexes dans un domaine qui, réalisées collectivement, satisfont un ensemble de buts considérés comme importants pour l'amélioration dans ce domaine.

1.3 OWPL

Pour les Petites et Moyennes Entreprises (PME), l'inconvénient des modèles d'amélioration des pratiques logicielles existants, tels que ceux vus précédemment, réside dans leur complexité à les mettre en oeuvre ainsi que dans leur coût très élevé. Une démarche d'amélioration des pratiques logicielles, conçue spécifiquement pour ces entreprises, a été développée en 1998, par le Laboratoire de Qualité Logiciel de l'Université de Namur (LQL), dans le cadre du projet de l'Observatoire Wallon des Pratiques Logicielles (OWPL) [1, 13, 15, 16, 26]. Elle a été appliquée avec succès sur des dizaines d'entreprises belges et québécoises, et est toujours en amélioration, avec la collaboration du Centre d'Excellence en Technologies de l'Information et de la Communication - Wallonie, Belgique (CETIC) et de l'Ecole de Technologie Supérieure - Québec, Canada (ETS).

L'objectif du projet OWPL est de définir un modèle d'évaluation et d'amélioration des processus de production de logiciels adaptés aux petites structures, considérant l'aspect amélioration comme prépondérant. En s'inspirant des modèles existants, l'idée est de fournir un modèle simplifié qui prend en compte les spécificités des PME wallonnes, à savoir, une taille réduite, une structure peu complexe, un nombre limité d'acteurs polyvalents et un niveau modeste de maturité de processus. L'objectif ultime est de prouver la possibilité d'améliorer sensiblement le niveau de qualité du processus de production des logiciels dans les PME et Très Petites Entreprises (TPE) sans pour autant pénaliser ces entreprises en les noyant de procédures et autres tâches administratives.

1.3.1 Méthodologie

Le modèle OWPL repose sur une démarche graduelle composée de trois étapes (Figure 1.2). Remarquons que cette démarche n'est pas linéaire, les entreprises peuvent boucler au niveau le plus approprié en fonction de leur taille et de leur niveau de maturité.

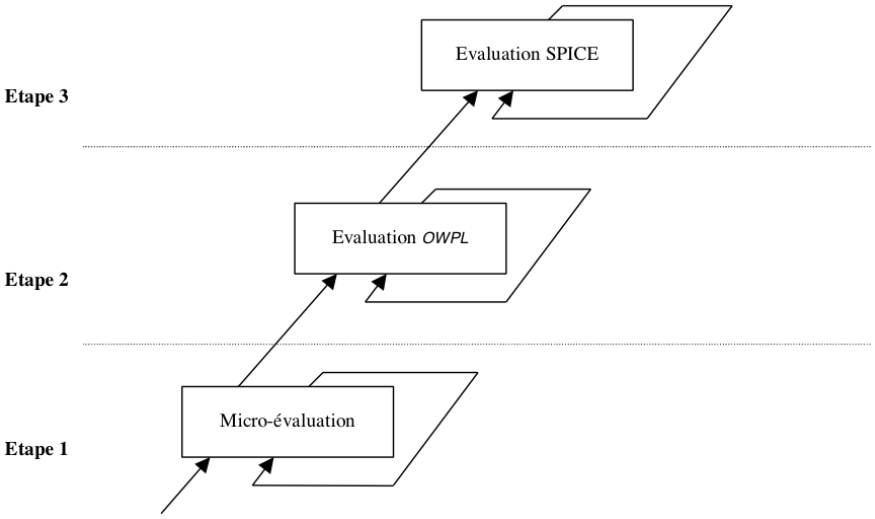

FIG. 1.2 – Démarche graduelle d'évaluation des pratiques logicielles

Etape 1 : Micro-évaluation

Lors de cette première étape, un questionnaire simplifié appelé micro-évaluation est utilisé pour collecter des informations sur les pratiques logicielles existantes dans les petites structures et pour sensibiliser les acteurs des PME aux enjeux de la qualité logicielle. Ce questionnaire couvre six axes, jugés les plus pertinents et prioritaires pour les organisations cibles, sur base d'expériences antérieures d'évaluation de processus au sein de PME et correspondant à des secteurs-clés des niveaux 2 et 3 de CMM.

Cette micro-évaluation permet d'effectuer une première analyse critique de la situation actuelle, et de définir un plan d'action très générique en fonction des axes évalués, mettant en évidence les bonnes pratiques déjà existantes ainsi que les pratiques offrant des opportunités d'amélioration. Cette étape peut être répétée à intervalles réguliers, par exemple tous les 8 mois, pour mesurer la progression éventuelle du niveau de qualité des pratiques.

Les informations collectées et les conclusions dressées suite à leur analyse peuvent également déterminer l'étendue et les objectifs d'une évaluation plus approfondie qui sera réalisée en référence au modèle OWPL qui constitue l'étape suivante de la méthodologie.

Etape 2 : Evaluation OWPL

Les résultats de la dernière micro-évaluation permettent de sélectionner le(s) processus à analyser plus en profondeur. Le modèle OWPL décompose les pratiques logicielles en 40 pratiques réparties dans 10 processus. L'analyse des pratiques logicielles d'une organisation au moyen du modèle OWPL permet d'obtenir une description très précise de la réalité, et donc de formuler une analyse et des recommandations très pertinentes.

Tout comme la micro-évaluation, l'évaluation OWPL peut être réalisée à plusieurs reprises pour mesurer l'évolution des pratiques sur une période donnée, et peut également servir de point d'entrée pour la dernière étape de notre démarche.

Etape 3 : Evaluation SPICE

Quand la taille ou le contexte d'une société justifie la nécessité d'une certification et quand la société a atteint un niveau de maturité suffisant, une évaluation en référence aux modèles CMM ou SPICE peut être réalisée. La démarche d'amélioration sera alors structurée en référence à ce modèle de processus.

1.3.2 Modèles

Modèle de micro-évaluation

La micro-évaluation couvre six axes clés : Assurance Qualité, Gestion des clients, Gestion des sous-traitants, Gestion de projet, Gestion de produits, Formation et gestion des ressources humaines.

La micro-évaluation repose essentiellement sur un questionnaire utilisé lors d'une entrevue avec un représentant de la société évaluée. Cette personne doit avoir une visibilité suffisante pour être capable de donner des informations objectives au sujet des pratiques de développement et de gestion de projet au sein de l'équipe concernée.

Le questionnaire comprend seize questions qui couvrent les axes mentionnés ci-dessus, ainsi que des sous-questions permettant à l'évaluateur de reformuler afin de raffiner les informations récoltées. Les réponses sont interprétées en référence à une grille d'évaluation figée. Deux types de questions peuvent être distingués : d'une part, les questions concernant les pratiques générales de l'organisation sont cotées sur une échelle linéaire en fonction du niveau de qualité de la pratique. D'autre part, les questions concernant les pratiques logicielles sont cotées en référence à une grille à deux entrées en fonction du niveau de qualité de la pratique ainsi que de son degré d'institutionnalisation effectif au sein de l'organisation (la pratique est présente seulement sur certains projets ou sur tous les projets).

La micro-évaluation débouche sur la rédaction d'un rapport d'une vingtaine de pages. Un rapport type présente d'abord brièvement l'approche, développe ensuite les informations collectées grâce au questionnaire et les résume par rapport aux six axes. Ensuite, ces résultats sont analysés en fonction du contexte de l'organisation évaluée (taille, objectifs, types de projets) pour formuler des recommandations qui permettent à la société de mettre en oeuvre un plan d'action susceptible de produire des résultats rapidement perceptibles, « quick wins », et donc, de s'améliorer.

Remarquons que la légèreté de la micro-évaluation repose sur la structure du questionnaire, mais également sur la possibilité de l'implémenter en n'impliquant qu'une seule personne du côté de la société évaluée.

Modèle OWPL

Le modèle OWPL constitue le composant central de la méthodologie et est composé de processus, de pratiques et de facteurs de succès (Figure 1.3).

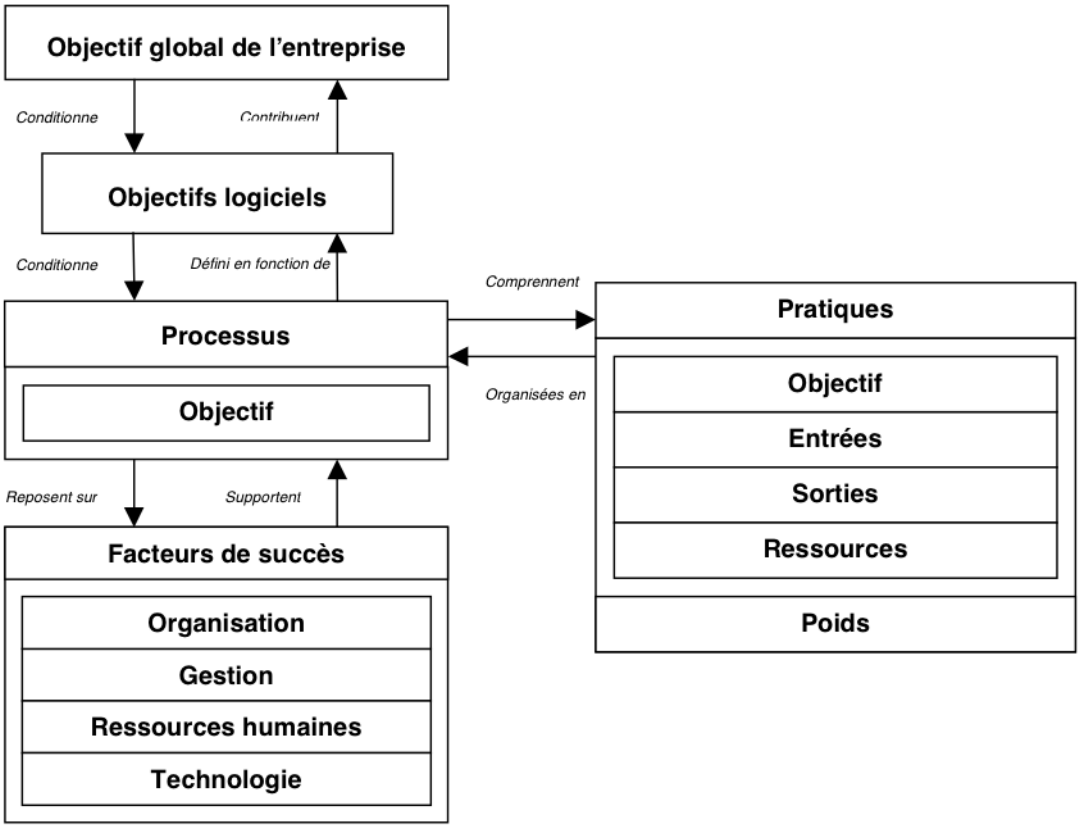

FIG. 1.3 – Structure du modèle OWPL

L'élément central du modèle OWPL est le *processus*. Celui-ci est organisé de manière à répondre à une préoccupation de l'entreprise, et défini en fonction d'un objectif qui contribue à la réalisation de l'objectif global de l'entreprise. Ces *processus* sont les suivants : Gestion des exigences, Planification, Suivi et supervision de projet, Développement, Documentation, Tests, Gestion de configuration, Gestion des sous-traitants, Gestion de la qualité, et Capitalisation des acquis. Chaque *processus* est composé de *pratiques* (entre 3 et 12) nécessaires pour sa mise en oeuvre effective et se déroule dans un environnement où des *facteurs de succès* garantissent sa performance.

Une *pratique* est une activité d'ingénierie qui contribue à la réalisation de l'objectif d'un *processus* par la création d'un délivrable ou l'amélioration de la capacité du *processus*. Elle est définie par son objectif, ses entrées, ses sorties, les ressources qui lui sont assignées et son poids qui est un indicateur de son importance dans la réalisation des objectifs du *processus*. L'évaluation de chaque *pratique*, c'est-à-dire, la mesure dans laquelle elle atteint ses objectifs, est réalisée au moyen de questionnaires plus détaillés qui utilisent un mode de cotation similaire à celui de la micro-évaluation. Chaque *pratique* est ainsi évaluée, d'une part sur une échelle de maturité à quatre niveaux, et d'autre part, par rapport à son degré d'institutionnalisation au sein de l'organisation.

Les *facteurs de succès* sont des facteurs généraux de l'environnement de l'organisation qui influencent le succès effectif des *processus*. Ils comprennent des facteurs de l'organisation, du management, des facteurs techniques et humains. Chaque *facteur de succès* est évalué sur une échelle de maturité à quatre niveaux.

1.4 Conclusion

Les bénéfices obtenus, par l'investissement d'une organisation dans l'amélioration de ses processus logiciels, sont évidents. Cependant, afin de réussir cette démarche d'amélioration, il est essentiel de choisir le modèle ou la norme qui correspond le mieux au contexte de l'organisation.

Les modèles et normes les plus connus, tels CMMI et SPICE, sont le plus souvent utilisés par des moyennes et grandes entreprises qui s'en accommodent parfaitement. Par contre, dans la plupart des PME et TPE, les pratiques logicielles sont loin d'être bien définies et le manque de ressources disponibles compromet le succès des approches d'amélioration des processus logiciels basées sur ces référentiels bien connus.

La démarche OWPL complète, comprenant la démarche graduelle et les différents modèles, a été développée pour apporter une solution aux problèmes des organisations qui ont un niveau de maturité faible et des ressources limitées, mais qui ont un besoin réel et rapide d'amélioration.

Cette démarche offre donc la possibilité aux PME d'améliorer leurs processus logiciels grâce à une approche ciblée qui permet de produire des résultats rapidement perceptibles sans investissement important. Elle repose sur l'hypothèse que toute activité d'une entreprise doit être effectuée dans l'optique de la réalisation des objectifs de l'entreprise. En outre, elle est compatible avec la plupart des modèles classiques tels CMM, dans lesquels le principe de l'approche graduelle est intégrée également. Cependant, l'utilisation d'un sous-ensemble détaillé et explicitement séparé de pratiques correspondant aux premiers niveaux CMM et la réalisation d'un modèle de référence spécifique se sont avérés efficaces, même au sein de (très) petites organisations.

Chapitre 2

Développement Agile

2.1 Introduction

Fin des années 1990, plusieurs méthodologies ont commencé à attirer l'attention du public. Chacune avait une combinaison différente d'anciennes et de nouvelles idées. Cependant, elles soulignaient toutes l'importance d'une étroite collaboration entre l'équipe de programmeurs et les experts des affaires ; une communication face à face (considérée comme plus efficace qu'une communication écrite) ; un apport fréquent de nouvelle valeur ajoutée pour les affaires ; des équipes fortes et autonomes ; des manières de gérer le code et l'équipe afin d'éviter que les changements d'exigences inévitables ne posent problème.

Ces méthodes de développement de logiciel, dites « Agile »[1] [2, 11, 14], tentent d'offrir une réponse à la communauté des affaires désireuse d'avoir des processus de développement de logiciel plus légers, plus rapides et plus adaptatifs que les processus de développement dit « disciplinés », considérés comme parfois trop lourds.

2.2 Contextes du développement Agile

Le « mouvement Agile » dans l'industrie du logiciel a vu le jour en 2001 lors de la publication, par un groupe de praticiens et de consultants, du « Manifeste pour le Développement Agile de Logiciels » [2, 6, 14]. Ce manifeste met en avant des valeurs centrales auxquelles la communauté Agile adhère :

– **les individus et les interactions** davantage que les processus et les outils.

  Le mouvement Agile met l'accent sur le rôle humain et la relation des développeurs de logiciel plutôt que sur des processus institutionnalisés et des outils de développement. Dans les pratiques Agile existantes, cela se manifeste par des collaborations étroites intra/inter équipes, des environnements de travail concentrés et divers moyens permettant de renforcer l'esprit d'équipe.

[1] Qui a de la facilité et de la rapidité dans l'exécution de ses mouvements ; prompt dans les opérations intellectuelles. (dictionnaire Robert)

– **les logiciels fonctionnels** davantage que la documentation compréhensive.

  L'objectif vital d'une équipe de développement de logiciel est de continuellement produire, à intervalles réguliers, des nouvelles versions de logiciel testées et opérationnelles. Les développeurs doivent garder le code le plus simple possible et techniquement aussi avancé que possible, limitant la charge de documentation au strict nécessaire.

– **la collaboration avec le client** davantage que la négociation de contrat.

  La relation et collaboration étroite entre les développeurs et les clients est préférée à une relation contractuelle figée. Cependant, un contrat clair est nécessaire et le processus de négociation doit être continu. Ce processus est donc vu comme un moyen de parvenir à une collaboration viable et de la maintenir. D'un point de vue des affaires, le développement Agile se focalise sur l'apport de valeur ajoutée dès le début du projet et sur la réduction des risques de non conformité par rapport aux exigences.

– **la réponse au changement** davantage que le suivi d'un plan.

  Le groupe de développement, comprenant les développeurs du logiciel et les représentants du client, doit être bien informé, compétent et autorisé à effectuer des changements, des ajustements, qui émergeraient pendant le cycle de vie du processus de développement. Cela signifie que les participants sont préparés à réaliser ces changements et que ces derniers sont prévus contractuellement.

Les méthodes Agile se focalisent sur l'efficacité et la manoeuvrabilité et considèrent les humains comme étant les premiers facteurs de réussite d'un projet. Elles utilisent donc des règles légères, mais suffisantes pour la gestion du projet, ainsi que des règles orientées sur l'humain et la communication.

2.2.1 Caractéristiques principales

Une méthode Agile doit avoir les caractéristiques suivantes [2, 14] :

– développement itératif.

  Un développement itératif est une approche utilisée pour développer du logiciel au cours de laquelle l'ensemble du cycle se compose de plusieurs itérations successives. Chaque itération est considérée comme un mini projet qui peut être composé de ses propres phases d'analyse des exigences, de conception, de programmation et de test. A la fin d'une itération, une partie du système, qui est stable, intégrée et testée, est considérée comme potentiellement livrable et représente donc une version du système.

– développement incrémental.

  Un développement incrémental signifie que l'on élabore le logiciel petit à petit, par incréments de fonctionnalités successifs. Il faut donc attribuer des priorités aux fonctionnalités, et ce, soit en donnant la priorité aux éléments du système les plus risqués à développer, soit aux éléments qui apportent le plus de valeur ajoutée au client ou soit encore, en combinant les deux s'il n'y a pas de conflit. Le client est, dès lors, activement impliqué dans la spécification, la priorité et la vérification des exigences.

– auto-organisation.

  L'équipe choisit la meilleure façon de gérer le travail.

– émergence.

  Les processus, principes et structures de travail sont mis en place durant le projet plutôt qu'avant.

2.2.2 Concepts fondamentaux

Plusieurs concepts se retrouvent dans la plupart des méthodes Agile [2, 8, 14] :

– accepter le changement.

  Considérer le changement comme un allié plutôt que comme un ennemi. Cela permet plus de créativité et de valeur ajoutée pour le client.

– cycle itératif court.

  Planifier plusieurs itérations avec un intervalle de temps assez court entre chacune, forcer l'implémentation des fonctions les plus prioritaires, délivrer rapidement de la valeur ajoutée au client.

– conception simple.

  Limiter la conception à ce qui est actuellement développé, car le changement est inévitable et la planification des fonctions futures est une perte de temps.

– restructuration.

  Restructurer le logiciel pour supprimer les éléments en double, améliorer la communication, simplifier le code, ou encore, ajouter de la flexibilité sans changer le comportement du programme.

– programmation à deux.

  Un style de programmation dans lequel deux programmeurs travaillent côte à côte sur un ordinateur, en collaborant pour le même travail de conception, d'algorithme, de code ou de test.

– révision d'itération.

  Réviser l'itération qui se termine du point de vue de l'efficacité du travail réalisé, des méthodes utilisées et des estimations. La révision permet l'amélioration de l'équipe et l'estimation des itérations futures.

– connaissance tacite.

  Etablir et mettre à jour la connaissance du projet dans la tête des participants plutôt que dans des documents.

– développement orienté par les tests.

  Les tests de modules et de méthodes sont écrits au fur et à mesure par les développeurs et les clients, avant et pendant le codage. Cela permet d'avoir des cycles d'itération courts.

2.3 Principales méthodes Agile

2.3.1 Extreme Programming (XP)

Mise au point par Kent Beck comme alternative aux cycles de développement longs des modèles traditionnels, cette méthode [2, 3, 4, 5, 14, 18] a été élaborée à partir de pratiques déjà existantes qui se sont révélées efficaces dans des processus de développement logiciel précédents. Elle fut formalisée après plusieurs utilisations dont les données ont été collectées. Le terme « Extrême » vient du fait que les pratiques et principes de bons sens sont poussés à l'extrême.

Processus

Le cycle de vie de XP (Figure 2.1) consiste en 5 phases : Exploration, Planning, Iterations to Release, Productionizing, Maintenance and Death.

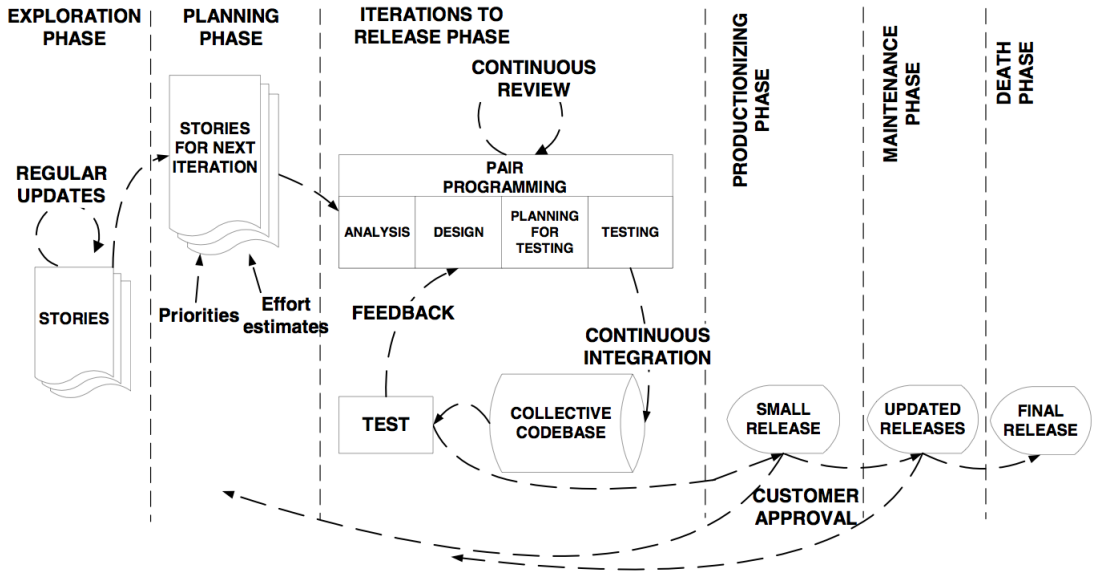

Fig. 2.1 – Cycle de vie du processus XP

– Exploration

  Le client écrit les « story cards » avec les exigences à inclure dans la première version du système. Chaque story card décrit une fonctionnalité du futur système. L'équipe de projet se familiarise avec les outils, la technologie et les pratiques qui seront utilisés. Un prototype est construit afin de tester les technologies à utiliser et d'explorer les possibilités d'architectures. La phase d'exploration varie de quelques semaines à quelques mois en fonction de la maîtrise de la technologie par les programmeurs.

– Planification

  Cette phase permet de donner des priorités aux story cards et de se mettre d'accord sur le contenu de la première version. Les programmeurs évaluent d'abord le temps de développement de chaque story card et le planning est ensuite établi de commun accord avec le client. La phase de planification dure environ deux jours.

– Iterations vers versions

  Cette phase comporte plusieurs itérations avant la première version livrable du système. Le calendrier prévu, lors de la phase de planification, est divisé en itérations qui durent de 1 à 4 semaines. La première itération crée un système incluant l'architecture de tout le système. Le client sélectionne ensuite les stories à inclure dans chaque itération et les tests fonctionnels créés par le client seront exécutés à la fin de chaque itération. A la fin de la dernière itération, le système est prêt pour la production.

– Production

  Cette phase comprend des tests supplémentaires et des vérifications sur la performance du système. Il est possible, à cette étape, que de nouveaux changements apparaissent et la décision de les inclure dans la version courante est alors à négocier avec le client. Les itérations peuvent aussi passer de 3 semaines à 1 semaine.

– Maintenance et fin

  La phase de maintenance commence dès que la première version est produite. Il s'agit dès lors d'assurer la maintenance de cette version tout en produisant la nouvelle, ce qui peut amener la productivité à diminuer. Il est également possible d'intégrer de nouvelles personnes ou de changer la structure de l'équipe.

  La phase finale commence dès que le client n'a plus de story à implémenter, à savoir, lorsque le système satisfait ses exigences fonctionnelles et non-fonctionnelles. La documentation nécessaire du système est alors écrite lorsque le système est stabilisé, c'est-à-dire, lorsqu'il n'y a plus de changement à apporter à l'architecture, à la conception ou au code.

Rôles et responsabilités

Dans le processus XP, il y a différents rôles à définir pour réaliser les tâches et les objectifs.

- Programmeur

  Le programmeur se charge d'écrire les tests et de garder le code le plus simple possible. Il se doit également de communiquer et de se coordonner avec les autres programmeurs et le reste de l'équipe.

- Client

  Le client écrit les stories, les tests fonctionnels et décide de la réalisation d'une exigence. C'est le client qui établit les priorités pour les exigences.

- Testeur

  Le testeur aide le client à écrire les tests fonctionnels qu'il exécutera par la suite. Il diffuse les résultats de ces tests et maintient les outils qu'il utilise.

- Traqueur

  Le traqueur assure le retour d'information. Pour ce faire, il confronte les estimations de l'équipe (l'effort) par rapport aux prévisions afin d'affiner les prévisions futures. Il trace également l'avancement des itérations et vérifie que l'objectif peut être atteint compte tenu des ressources disponibles et des contraintes de temps.

- Entraîneur

  L'entraîneur est responsable de l'ensemble du processus. Sa bonne compréhension de XP lui permet de guider les autres membres de l'équipe.

- Consultant

  Le consultant est un membre externe à l'équipe qui possède une connaissance technique spécifique. Il guide l'équipe dans la résolution de ses problèmes spécifiques.

- Directeur

  Le directeur est celui qui prend les décisions en communiquant avec l'équipe du projet afin de déterminer la situation courante et de distinguer les difficultés ou les déficiences du processus.

Pratiques

La méthode XP est un assemblage de pratiques existantes en génie logiciel pour réussir des projets dont les exigences changent constamment et lorsqu'il s'agit d'équipes de petite et moyenne taille.

– Jeu de planification

  L'objectif de cette pratique est de déterminer rapidement la portée de la nouvelle itération en combinant les priorités des affaires et les estimations techniques. Il est important d'avoir un dialogue permanent entre le client et les programmeurs afin d'arriver à un compromis entre les préoccupations des affaires et celles du technique. La partie des affaires décide de la portée de l'itération, des priorités, du contenu des versions et de la date des versions. La partie technique décide des estimations, du processus, de la planification détaillée et des conséquences techniques des décisions stratégiques prises par les affaires.

– Petites versions

  Un système simple est produit rapidement (tous les 2/3 mois). Les nouvelles versions sont produites parfois quotidiennement mais au moins mensuellement. Cela favorise la gestion des risques et la forte valeur ajoutée.

– Métaphore

  Le système et son fonctionnement sont définis comme une métaphore, ou un ensemble de métaphores, afin que le client comprenne clairement les éléments principaux du système.

– Conception simple

  L'accent est mis sur la conception de solutions les plus simples possibles en supprimant donc la complexité et les fonctionnalités inutiles.

– Test

  Il s'agit de vérifier le bon fonctionnement et la non régression du programme en exécutant les tests avant de coder et en parallèle au codage.

– Restructuration

  Afin de faciliter l'ajout de fonctionnalités, la restructuration de l'architecture et du code permet de supprimer les duplications, simplifier le système, et ce, tout en faisant en sorte que les tests soient satisfaits.

– Programmation à deux

  Deux programmeurs travaillent sur une machine. L'un écrit le code, tandis que l'autre, expérimenté et développeur, observe et suggère des solutions. Cela permet d'avoir un code plus propre et moins d'erreurs de syntaxe. Remarquons que les deux intervertissent régulièrement les rôles.

– Propriété collective

  Tous les membres de l'équipe sont censés connaître le code et peuvent le modifier à tout moment.

– Intégration continue

  Dès qu'un bout de code est stabilisé, il est intégré.

– Semaine de 40 heures

  Un maximum de 40 heures de travail par semaine.

– Présence du client

  Le client est présent et disponible à temps plein pour l'équipe.

– Standards de codage

  Il existe des règles de codage qui sont suivies par les programmeurs.

– Espace ouvert

  La préférence est données à une large pièce avec de petits bureaux compartimentés.

– Règles justes

  L'équipe suit ses propres règles qui peuvent évoluer en cours de projet.

2.3.2 Rational Unified Process

Rational Unified Process (RUP) [2, 14, 19] a été proposé par Philippe Kruchten et Ivar Jacobsen afin de fournir une méthodologie supportant Unified Modelling Language (UML). RUP est une méthode commerciale qui propose une approche itérative pour le développement orienté objet de systèmes et qui se base fortement sur les cas d'utilisation pour modéliser les exigences. L'approche se focalise sur ses propres méthodes et laisse peu de place à l'intégration d'autres méthodologies.

Processus

Le cycle de vie d'un projet RUP se divise en quatre *phases* : Inception, Elaboration, Construction et Transition (Figure 2.2). Chaque phase se divise en *itérations*, chacune devant produire une partie fonctionnelle du logiciel, et ce, sur une période de 2 semaines à 6 mois.

Chaque itération se divise en 9 *activités* : Modélisation des affaires, Exigences, Analyse et conception, Implémentation, Test, Gestion de la configuration et du changement, Gestion du projet et Environnement. Ces activités peuvent se dérouler en parallèle durant une même itération mais ne sont pas forcément toutes réalisées.

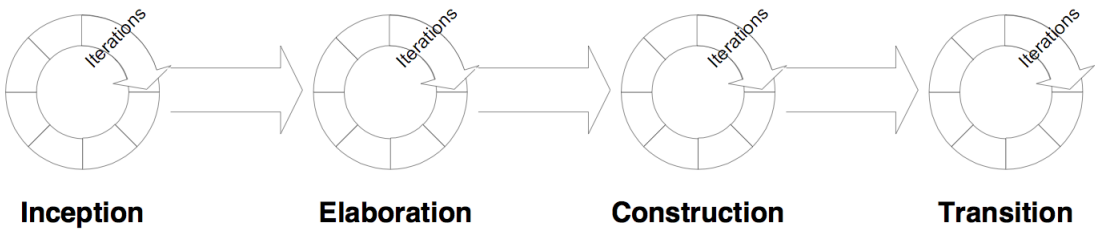

Fig. 2.2 – Cycle de vie du processus RUP

– Inception

  Lors de l'Inception, les objectifs du projet sont établis afin de considérer les besoins de chaque partie intéressée. Ceci nécessite d'établir les frontières du projet et ses critères d'acceptation. Les cas d'utilisation critiques sont identifiés et les architectures candidates du système sont conçues. Des estimations sur le planning, les coûts et la phase suivante sont également réalisées.

– Elaboration

  La base de l'architecture du logiciel est définie lors de la phase d'Elaboration. Le problème est analysé au regard du système entier à construire et le plan du projet est défini. Le processus, l'infrastructure et l'environnement de développement sont décrits de manière détaillée. Les cas d'utilisation, les acteurs ainsi que l'architecture du logiciel sont également décrits. A la fin de cette phase, une analyse est réalisée pour déterminer le risque, la stabilité architecturale ainsi que la différence entre les ressources planifiées et consommées.

– Construction

  La phase de construction procède au développement, à l'intégration et au test des composants et dispositifs d'application. Cette phase peut être comparée à un processus industriel où l'accent est mis sur la gestion des ressources, le contrôle des coûts, le contrôle du planning et la vérification de la qualité. Les résultats de cette phase sont produits rapidement et en plusieurs versions avant de pouvoir arriver à la version finale.

– Transition

  La phase de transition débute lorsque le produit est assez mature pour être distribué aux utilisateurs. Cette phase comprend du beta testing, des formations à l'utilisateur, des formations à la maintenance et de la rédaction de documentation pour l'utilisateur.

Rôles et responsabilités

Dans RUP, les rôles sont « orientés activités », c'est-à-dire, qu'il existe un rôle pour chaque activité.

- Analyste du processus des affaires

  Cet analyste est en collaboration continue avec les membres des affaires de l'organisation et dirige la définition des cas d'utilisation qui décrivent les processus et acteurs de cette organisation. Il produit une vue haut niveau des affaires (modèle de cas d'utilisation des affaires).

- Concepteur des affaires

  Le concepteur des affaires divise le modèle de cas d'utilisation des affaires en plusieurs parties afin d'identifier et de documenter les rôles et entités au sein de l'organisation.

- Réviseur du modèle des affaires

  Ce réviseur se charge de réviser les divers artefacts produits par l'analyste du processus des affaires et par le concepteur des affaires.

- Développeur de cours

  Ce développeur produit du matériel de cours pour l'utilisateur tels que des slides, des tutoriels, des exemples,...

- Toolsmith

  Cette personne se charge de développer des outils pour supporter le développement, améliorer l'automatisation des tâches répétitives et fastidieuses et pour améliorer l'intégration entre les outils.

Pratiques

RUP met en place une série importante de pratiques. On identifie 6 grandes pratiques.

- Développement itératif du logiciel

  Le logiciel est développé en petits incréments et avec des itérations courtes afin de pouvoir identifier les risques et problèmes le plus tôt possible.

- Gestion des exigences

  Identifier les exigences logicielles qui sont susceptibles de changer et qui ont un impact important sur le système. Une approche disciplinée pour gérer les exigences est donc requise afin de pouvoir leur donner une priorité, les tracer, les filtrer,...
  Cette pratique est considérée comme importante dans RUP.

– Utilisation d'architectures orientées composants

  L'architecture logicielle peut être rendue plus flexible via l'utilisation de composants. En outre, le développement de composants réutilisables peut potentiellement épargner des développements futurs.

– Modélisation visuelle du logiciel

  Les modèles sont construits car les systèmes complexes sont impossibles à comprendre dans leur totalité. En utilisant une méthode de visualisation, telle que UML, l'architecture et la conception du système peuvent être interprétées et communiquées sans ambiguïté à toutes les parties concernées.

– Vérification de la qualité du logiciel

  En faisant des tests à chaque itération, les problèmes peuvent être identifiés très tôt dans le cycle de développement et donc lorsqu'ils sont les moins coûteux à corriger.

– Contrôle des changements au logiciel

  Tout changement aux exigences doit être géré et son impact sur le logiciel doit pouvoir être tracé. La maturité du logiciel peut aussi être effectivement mesurée par la fréquence et les types de changements réalisés.

2.3.3 Autres méthodes Agile

Scrum

Scrum [2, 14, 23, 24] est une approche empirique développée pour gérer le processus de développement. Elle applique les théories de processus de contrôle industriel dans une démarche qui réintroduit les idées de *flexibilité*, *adaptabilité* et *productivité*.

Scrum ne définit pas une technique de développement particulière, mais focalise son attention sur l'interaction et le fonctionnement au sein de l'équipe afin d'assurer la flexibilité du système dans un contexte très évolutif.

Crystal family of methodologies

Crystal family [2, 14], mise au point par A. Cockburn, inclut un ensemble de méthodologies pour sélectionner la méthode la plus adaptée à chaque projet. Les méthodes Crystal ont certains points communs :

– cycle de développement incrémental (durée maximale de 4 mois) ;

– importance de la communication et de la coopération entre les personnes ;

– pas de limitation sur les pratiques de développement, les outils ou produits (utilisation de pratiques XP et Scrum) ;

– objectifs précisés pour limiter le nombre de produits.

Feature Driven Development

Feature Driven Development (FDD) [2, 14, 22] est une méthode de développement développée par Palmer et Felsing. Elle a la particularité de ne pas couvrir tous les aspects du processus de développement mais de se focaliser sur la conception. Elle a été élaborée pour être compatible avec d'autres activités liées au développement et à la gestion de projet.

Cette méthode préconise un développement itératif ainsi que l'utilisation des meilleures pratiques utilisées dans l'industrie. En outre, elle favorise une vue qualité produit ce qui nécessite des livraisons fréquentes de versions afin de vérifier continuellement le produit.

FDD se découpe en 5 étapes séquentielles menant à la réalisation d'une partie du système. Les étapes 4 et 5 constituent la partie itérative de FDD (Conception et Construction) qui supporte le développement Agile. Typiquement, une itération dure de 1 semaine à 3 semaines.

Dynamic Systems Development Method

Dynamic Systems Development Method (DSDM) [2, 14, 27] fut élaborée en 1994 par une communauté de développeurs et constituait la méthode de développement rapide numéro 1 en Grande-Bretagne.

L'idée principale de DSDM est d'estimer le temps et les ressources disponibles afin de voir ce qui peut être développé, à l'opposé du point de vue traditionnel, qui estime d'abord le nombre de fonctionnalités du système pour ensuite estimer les ressources et le temps nécessaire au développement.

Adaptive Software Development

Adaptive Software Development (ASD) [2, 14, 17] fut développé par James Highsmith et publié en 2000. ASD se concentre sur les problèmes liés aux développements de systèmes complexes et à l'E-business.

ASD préconise un développement itératif et incrémental avec un prototypage constant. ASD comporte 3 phases : *Speculate* (planning), *Collaborate* (travail en équipe et communication) et *Learn* (étude des erreurs). Ces noms de phases ont été choisis pour souligner l'aspect changeant du processus.

2.4 Conclusion

Une méthode Agile est caractérisée comme étant [2, 14] :

- **incrémentale et itérative** : des petites versions livrables de logiciel selon des cycles rapides ;
- **coopérative** : les clients et développeurs travaillent ensemble continuellement avec une forte communication ;
- **simple** : la méthode est facile à apprendre et bien documentée ;
- **adaptative** : il est possible de faire des changements de dernière minute.

L'aspect central des méthodes Agile réside donc dans la simplicité et la rapidité. L'équipe de développement se concentre d'abord sur les fonctions nécessaires permettant ainsi de délivrer, le plus vite possible, une version du système au client et d'obtenir ses réactions.

Le développement Agile est donc une vision de développement de logiciel qui est principalement centrée sur les personnes.

Deuxième partie

Contexte spécifique

Chapitre 3

Interventions dans les organisations

3.1 Introduction

Le premier objectif du mémoire est d'expérimenter le modèle OWPL avec deux petites entreprises québécoises. Cette expérimentation a pour optique de servir de base au second objectif qui consiste en une proposition d'adaptation du modèle OWPL en vue d'améliorer sa démarche.

En effet, expérimenter la démarche OWPL permet, outre une compréhension approfondie et pragmatique du modèle, de réellement comprendre les différents problèmes auxquels font face les petites entreprises, ainsi que le contexte dans lequel elles se situent.

Dans ce chapitre, nous résumerons tout d'abord les interventions que nous avons menées auprès des deux petites entreprises, à savoir, Logilys et le Quartier général de l'URSC-E. Remarquons que les détails des interventions sont repris dans les annexes de ce document. Enfin, nous conclurons avec les enseignements que nous avons pu tirer de l'expérimentation de la démarche OWPL.

3.2 Logilys

3.2.1 Présentation

Logilys est une entreprise informatique ayant une expertise reconnue dans les domaines de la consultation, l'analyse et le développement d'applications spécialisées. Elle est composée de dix employés dont huit en informatique qui oeuvrent depuis juin 2002. Leurs réalisations ont été développées pour des secteurs d'activités aussi variés que des :

- entreprises manufacturières (meuble, métal, ...) ;
- firmes de génie conseil et de consultation ;
- entreprises de services de location de plateaux, salles, terrains de loisir ;
- organismes de charité.

La mission de Logilys consiste à aider sa clientèle à demeurer performante et concurrentielle dans un environnement en constante évolution. Pour ce faire, Logilys se donne comme objectif de contribuer à l'amélioration de l'efficacité et de l'efficience des processus de gestion en concevant des logiciels apportant de réelles solutions d'affaires.

Ils commercialisent, à ce jour, trois produits :

- PRODOC : ce logiciel fait la gestion de la production et fait de la collecte de données, produit des rapports ainsi que des statistiques sur la production ;

- PROLOC : ce logiciel fait la gestion de la location pour tous les types de plateaux (salles, terrains de loisir, arénas, amphithéâtres, piscines,...) ;

- PRODON : ce logiciel fait la gestion des campagnes de sollicitation auprès du public pour des campagnes de financement d'organismes ou de fondations.

Afin d'élaborer des solutions parfaitement adaptées, ergonomiques et évolutives, Logilys privilégie une approche consultative en étroite collaboration avec les futurs utilisateurs des applications informatiques.

En effet, Logilys offre différents services à ses clients :

- CONSULTATION : Logilys permet d'améliorer et d'adapter les trois produits commercialisés. Il est en effet possible pour un client, d'ajouter de nouvelles fonctionnalités sur les logiciels en place, de créer ou d'améliorer l'interaction avec les autres logiciels, proposer de nouvelles applications répondant mieux aux besoins de l'entreprise en expansion ;

- DEVELOPPEMENT : Logilys peut, après un diagnostic des besoins et de l'environnement d'un client, développer un logiciel spécifiquement conçu pour lui ;

- FORMATION : Logilys offre des sessions de formation adaptées aux besoins des clients ;

- INSTALLATION : Logilys s'occupe de la configuration et du bon fonctionnement de ses produits dans l'environnement du client.

3.2.2 Evaluation

La démarche graduelle OWPL a été choisie comme méthodologie d'amélioration des processus logiciels. Cette méthodologie vise à rendre l'entreprise consciente des aspects qualité des logiciels et à initialiser un mécanisme SPI continu qui produira des résultats rapides et tangibles avec un minimum de ressources.

La première étape de la méthodologie propose une évaluation de l'organisation réalisée grâce à un questionnaire simplifié, appelé « micro-évaluation ». Les évaluateurs qui ont interrogé les représentants de chez Logilys, afin de réaliser la micro-évaluation, étaient des étudiants de l'ETS.

Cette micro-évaluation nous a permis d'avoir une première compréhension de l'organisation. Nous avons étudié ses résultats en fonction des six axes du modèle OWPL, la liste de leurs forces et faiblesses, et finalement, les pratiques recommandées par les étudiants pour aider l'organisation évaluée à améliorer ses pratiques et processus existants.

La seconde étape dans la démarche OWPL était de réaliser un plan d'action qui met en évidence les problèmes importants de l'organisation et des propositions de solution à ces problèmes. Afin de réaliser ce plan d'action, nous avons analysé l'information collectée par la micro-évaluation et les conclusions tirées de l'analyse de ces résultats. Nous avons donc réparti les recommandations provenant de la micro-évaluation selon des objectifs correspondants aux secteurs-clés des premiers niveaux de CMM, comme par exemple, la gestion des changements, le gestion de la configuration, etc. Ensuite, nous avons classé les objectifs les plus importants par ordre de priorité selon notre expertise et nous avons proposé une solution générale pour chacun d'eux. Pour ce faire, nous avons étudié un ensemble de bonnes pratiques existantes dans l'ingénierie du logiciel afin de dégager celles qui répondraient le mieux aux objectifs et donc aux problèmes de l'organisation. Remarquons qu'il était nécessaire de classer ces objectifs afin de concentrer le travail sur un ou deux des plus importants pour garantir que ces améliorations soient gérées correctement et qu'elles perdurent.

Le plan d'action de Logilys est repris à l'annexe A de ce document. Ci-dessous, les plus gros problèmes identifiés dans cette organisation :

1. **Développer le bon logiciel** : les exigences ne sont pas bien définies et les clients ne sont pas impliqués dans le processus de développement.

2. **Gérer les requêtes de changement** : les requêtes de changement ne sont pas bien définies.

3. **Adopter une approche de qualité** : les activités de qualité sont limitées aux tests et à la formation des futurs développeurs. En outre, aucune méthodologie de développement n'a été adoptée et chaque personne est libre d'effectuer la documentation qu'elle juge nécessaire en ce qui concerne les documents d'analyse.

4. **Délivrer le logiciel à temps** : la planification de projet n'est pas appliquée pour tous les projets (juste pour les plus importants) et il n'y a aucune façon de vérifier la progression du projet.

5. **Gérer la configuration** : aucune gestion des versions de la documentation ou du code n'est formalisée à l'aide d'un outil prévu à cet effet.

6. **Gérer les relations avec les sous-traitants** : la sélection des sous-traitants ne passe pas par un processus décisionnel formel et le suivi complet des sous-traitants est rarement réalisé.

Ci-dessous, la synthèse des solutions proposées :

1. Afin de garantir le développement du bon logiciel, à savoir, un logiciel qui répond aux exigences du client, il est impératif de pouvoir définir et gérer au mieux ces exigences. Il s'agit donc de déterminer formellement les *services* qu'un système doit offrir et les *contraintes* sous lesquelles il doit fonctionner et ce, grâce à un document de spécification des exigences logicielle (Software Requirement Specification (SRS)).

2. La traçabilité des exigences aide à gérer le changement dans les exigences car elle permet de décrire, suivre et comprendre l'évolution d'une exigence. Elle permet de lier les exigences entre elles ainsi qu'avec les documents sources. Elle aide les concepteurs à garder trace des changements et à pouvoir évaluer l'impact d'un changement sur le système avant son implémentation. Enfin, elle réalise la correspondance entre les tests et les exigences afin d'assurer une pleine couverture des objectifs du client.

3. L'approche qualité peut être améliorée en utilisant la méthode RUP qui vise à gérer et contrôler les activités de développement.

4. L'utilisation, pour chacun des projets, de techniques de mesure de taille, tel que COSMIC-FFP, et d'effort, tel que ISBSG, pour l'estimation d'un projet permettrait de connaître et de suivre la taille, l'effort, la durée et les coûts. Ces connaissances et suivis seraient plus qu'utiles dans l'optique de pouvoir délivrer un produit en temps et en heure à un client.

5. La formalisation des documents de production et du code source améliorerait la gestion de la configuration. En outre, l'utilisation d'un outil tel que « subversion » permettrait de garder trace de toutes les versions des documents et du code source.

6. Un contrat spécifique pour les sous-traitants aiderait à identifier clairement la responsabilité de chacun et à avoir un retour d'information régulier sur le travail en cours.

Considérant que cette classification ne représente qu'un point de vue basé sur la micro-évaluation, il était nécessaire de discuter de ces problèmes avec l'organisation et d'avoir une meilleure compréhension de la manière dont ils font actuellement face à ces problèmes afin d'être capable d'améliorer leur processus logiciel. La première réunion avec un des représentants de l'entreprise s'est donc déroulée à la fin de l'étape d'évaluation.

Lors de cette réunion, les objectifs étaient de comprendre quels sont les problèmes les plus important pour l'organisation et vérifier la pertinence de notre classement en fonction du contexte actuel de l'entreprise. Une fois d'accord avec le représentant, nous avons planifié de les aider à résoudre certains problèmes en tenant compte des trois critères suivants :

- le court terme ;

- la réalisabilité pour l'entreprise en terme de coûts, de ressources disponibles, etc.

- l'utilité pour l'entreprise.

3.2.3 Problématique

Nous avons choisi de les aider à améliorer les deux premiers problèmes, à savoir, le développement du bon logiciel et la gestion des requêtes de changement. Ces deux problèmes ont été choisis car les représentants de l'organisation ont une motivation très importante en ce qui concerne la gestion des changements et car nous avons argumenté qu'il est difficile de gérer les requêtes de changement sans une gestion des exigences.

Leur motivation vient du fait qu'actuellement, le représentant de l'organisation, qui est également le chef de projet, reçoit énormément de requêtes de changement, et ce, de manière informelle. En effet, Logilys permet à ses clients de réclamer du changement dans les fonctionnalités du logiciel qu'ils utilisent. Ainsi, l'organisation est dotée de points de contact qui reçoivent ces requêtes par téléphone ou par courriel et qui les transmettent, sous forme manuscrite, au chef de projet avec la description de la requête de changement. Le problème vient alors de la difficulté, pour le chef de projet, de gérer ces requêtes de cette manière car il n'existe aucun processus formel ni aucun modèle formel pour la gestion des requêtes et la gestion des exigences.

3.2.4 Amélioration

La solution qualité apportée à Logilys est reprise aux annexes C et D de ce document. A l'annexe C, en ce qui concerne la gestion des exigences, et, à l'annexe D, en ce qui concerne la gestion des changements. Ci-dessous, la synthèse des améliorations apportées en ce qui concerne la gestion des exigences et la gestion des changements pour Logilys.

Gestion des exigences

Il est important d'utiliser un bon document de spécification des exigences logicielles (SRS) afin de pouvoir gérer les exigences au mieux. Ainsi, nous avons présenté, en détail, le modèle IEEE 830-1998 à Logilys.

En plus de cela, il était nécessaire d'utiliser un outil spécifique qui permet de gérer les *exigences spécifiques* qui constituent la partie la plus importante du document SRS. Après plusieurs recherches et analyses, nous avons décidé d'utiliser GenSpec qui est un outil développé par Hydro-Québec.

GenSpec est basé sur des normes internationales et permet de résoudre les problèmes bien connus en ingénierie des exigences. GenSpec organise les exigences de manière hiérarchique afin de faciliter la compréhension d'une vue générale à une vue plus spécifique. Il permet de :

- définir les exigences ;
- caractériser une exigence ; priorité, commentaire, fichier attaché, documents sources ;
- lier les exigences qui sont logiquement dépendantes ;
- lier les procédures de test aux exigences auxquelles elles s'appliquent ;
- gérer le changement dans les exigences en fournissant un historique des changements par exigence ;
- générer les documents formels, avec les exigences détaillées, selon le modèle IEEE 830-1998.

La réunion suivante avec le représentant de l'organisation s'est déroulée à la fin de l'étape d'amélioration de la gestion des exigences. Nous avons présenté au chef de projet la solution proposée, nous avons réalisé une démonstration de cette solution et répondu à ses questions. Nous avons basé notre démonstration sur un de leurs projets, à savoir, ProDon, afin de faciliter sa compréhension de GenSpec.

En outre, nous avons discuté du deuxième problème qui concerne la gestion des changements, afin d'avoir une meilleure compréhension du contexte de leur organisation et de leur motivation, expliqué dans la section précédente sur la *problématique*. Nous avons également planifié de nous rencontrer quelques jours plus tard dans leur entreprise afin de leur fournir des informations supplémentaires sur GenSpec et la solution d'amélioration pour leur gestion des changements.

Gestion des changements

En ce qui concerne la gestion des changements, nous avons fourni un processus de gestion des changements adapté au contexte de l'organisation qui permet de formaliser la manière dont les requêtes de changement sont enregistrées, gérées et résolues.

Afin de formaliser la description d'une requête, nous avons fourni un modèle formel pour enregistrer chacune des requêtes des clients. Ce modèle est composé de trois parties complémentaires : l'identification, la description et l'analyse. Les deux premières seront utilisées par les points de contact de l'organisation avec les clients. Les trois parties pourront ensuite être mises à jour par le chef de projet au fur et à mesure que la requête évolue dans les différentes étapes du processus.

Lors de la réunion suivante, dans leur entreprise, nous avons procédé à la formation d'un de leurs employés qui sera en charge d'essayer GenSpec avec un de leurs projets. Après cette formation, nous avons présenté au représentant de l'organisation la solution qualité pour l'amélioration de la gestion des changements, à savoir, le processus et le modèle présenté précédemment, et nous avons discuté avec lui pour les ajuster aux besoins de Logilys.

Pour supporter ce processus et ce modèle, nous avons développé pour l'organisation un programme, appelé ModeX, qui permet d'enregistrer et de gérer les requêtes de changement et de faire des liens entre les requêtes et les exigences. Une fonctionnalité intéressante est la possibilité de voir si une requête de changement est concernée par une autre grâce aux liens avec les exigences. Cela permet au chef de projet de fusionner plusieurs requêtes qui concernent la même exigence.

3.2.5 Résultats

L'organisation semble vouloir intégrer les solutions proposées, et une ressource, en plus du chef de projet, sera en charge d'essayer d'intégrer ces améliorations. Ils semblent donc être plus conscients des aspects qualité du logiciel et veulent appliquer et maintenir cette démarche pour améliorer leurs processus à court et long terme.

3.2.6 Travaux futurs

Il sera très important de réaliser une autre évaluation de l'organisation quelques mois plus tard afin d'évaluer l'amélioration réalisée par cette organisation et d'analyser s'ils ont réussi à appliquer et maintenir la solution qualité.

3.3 Quartier général de l'URSC-E

3.3.1 Présentation

Le département IT de l'Unité Régionale de Soutien des Cadets de la Région de l'Est (URSC-E) est composé d'un total de trois employés dont deux qui travaillent comme programmeurs, et un comme chef de projet. Ce département est en charge du développement de logiciel pour la gestion des employés et des cadets.

Les fonctionnalités principales du logiciel concernent l'instruction des employés et des cadets, et facilitent les tâches pour ceux qui gèrent les ressources humaines. Le client principal est l'URSC-E mais le logiciel est déployé de plus en plus dans d'autres régions du Canada.

Le Quartier Général de l'URSC-E est situé à Saint-Jean-sur-Richelieu et a un effectif de 137 membres permanents de la Force Régulière, de la Force de Réserve et des employés civils du Ministère de la Défense Nationale (MDN). Avec le support de plus de 1500 officiers du Cadre des Instructeurs de Cadets (CIC), l'URSC-E supervise plus de 17000 cadets distribués dans 260 unités à travers le Québec et la vallée de l'Outaouais. Son budget annuel de 34 milliards $ est principalement alloué aux activités annuelles des cadets, en ce compris, le salaire du personnel.

3.3.2 Evaluation

La démarche graduelle OWPL a été choisie comme méthodologie d'amélioration des processus logiciels. Cette méthodologie vise à rendre l'entreprise consciente des aspects qualité des logiciels et à initialiser un mécanisme SPI continu qui produira des résultats rapides et tangibles avec un minimum de ressources.

La première étape de la méthodologie propose une évaluation de l'organisation réalisée grâce à un questionnaire simplifié, appelé « micro-évaluation ». Les évaluateurs qui ont interrogé les représentants du département IT de l'URSC-E, afin de réaliser la micro-évaluation, étaient des étudiants de l'ETS.

Cette micro-évaluation nous a permis d'avoir une première compréhension de l'organisation. Nous avons étudié ses résultats en fonction des six axes du modèle OWPL, la liste de leurs forces et faiblesses, et finalement, les pratiques recommandées par les étudiants pour aider l'organisation évaluée à améliorer ses pratiques et processus existants.

La seconde étape dans la démarche OWPL était de réaliser un plan d'action qui met en évidence les problèmes importants de l'organisation et des propositions de solution à ces problèmes. Afin de réaliser ce plan d'action, nous avons analysé l'information collectée par la micro-évaluation et les conclusions tirées de l'analyse de ces résultats. Nous avons donc réparti les recommandations provenant de la micro-évaluation selon des objectifs correspondant aux secteurs-clés des premiers niveaux de CMM, comme, par exemple, la gestion des changements, la gestion de la configuration, etc. Ensuite, nous avons classé les objectifs les plus importants par ordre de priorité selon notre expertise et nous avons proposé une solution générale pour chacun d'eux. Pour ce faire, nous avons étudié un ensemble de bonnes pratiques existantes dans l'ingénierie du logiciel afin de dégager celles qui répondraient le mieux aux objectifs et donc aux problèmes de l'organisation. Remarquons qu'il était nécessaire de classer ces objectifs afin de concentrer le travail sur un ou deux des plus importants pour garantir que ces améliorations soient gérées correctement et qu'elles perdurent.

Le plan d'action du quartier général de l'URSC-E est repris à l'annexe B de ce document. Ci-dessous, les plus gros problèmes identifiés dans cette organisation :

1. **Estimer le projet** : aucune connaissance ni suivi de l'effort, de la durée et des coûts d'un projet.

2. **Développer le bon logiciel** : aucun document formel pour la spécification des exigences (SRS) et aucune traçabilité entre les tests et ces exigences.

3. **Fournir de la bonne documentation logicielle** : le processus de documentation du code est laissé à la seule discrétion des développeurs.

4. **Maintenir la qualité** : aucune méthodologie pour le contrôle de la qualité ni aucune pratique de gestion de la qualité qui permet d'éviter de dépendre des individualités.

Ci-dessous, la synthèse des solutions proposées :

1. L'utilisation, pour chacun des projets, de techniques de mesure de taille, tel que COSMIC-FFP, et d'effort, tel que ISBSG, pour l'estimation d'un projet, permettrait de connaître et de suivre la taille, l'effort, la durée et les coûts. Ces connaissances et suivis seraient plus qu'utiles dans l'optique de pouvoir délivrer un produit en temps et en heure à un client.

2. Afin de garantir le développement du bon logiciel, à savoir, un logiciel qui répond aux exigences du client, il est impératif de pouvoir définir et gérer au mieux ces exigences. Il s'agit donc de déterminer formellement les *services* qu'un système doit offrir et les *contraintes* sous lesquelles il doit fonctionner et ce, grâce à un document de spécification des exigences logicielles (SRS).

3. La meilleure manière de produire de la bonne documentation logicielle pour un projet est de le documenter progressivement. Cela permet d'améliorer sa propre compréhension du projet ainsi que celle des autres développeurs. De plus, cela aide à comprendre ce que l'on veut écrire avant de l'écrire.

4. L'approche qualité peut être améliorée en utilisant la méthode RUP qui vise à gérer et contrôler les activités de développement.

Considérant que cette classification ne représente qu'un point de vue basé sur la micro-évaluation, il était nécessaire de discuter de ces problèmes avec l'organisation et d'avoir une meilleure compréhension de la manière dont ils font actuellement face à ces problèmes afin d'être capable d'améliorer leur processus logiciel. La première réunion avec un des représentants de l'entreprise s'est donc déroulée à la fin de l'étape d'évaluation.

Lors de cette réunion, les objectifs étaient de comprendre quels sont les problèmes les plus important pour l'organisation et de vérifier la pertinence de notre classement en fonction du contexte actuel de l'entreprise. Une fois d'accord avec le représentant, nous avons planifié de les aider à résoudre certains problèmes en tenant compte des trois critères suivants :

- le court terme ;

- la réalisabilité pour l'entreprise en terme de coûts, de ressources disponibles, etc.

- l'utilité pour l'entreprise.

3.3.3 Problématique

Nous avons choisi de les aider à améliorer le premier problème, à savoir, l'estimation de projet. Nous avons choisi ce problème car le représentant de l'organisation, qui est le chef de projet, a une motivation importante en ce qui concerne l'estimation de projet. En effet, il réalise actuellement son propre planning avec ses propres tables, schémas, et autres, afin d'estimer un projet et donc il ne possède pas de manière formelle, ni de technique qui lui permettraient de connaître et suivre son projet.

Nous avons étudié des techniques telles que COSMIC-FFP et ISBSG, ainsi que différentes manières d'instaurer un programme de mesure afin d'aider le chef de projet à établir un tel programme, qui lui permettrait de connaître et de suivre l'effort, la durée et les coûts de son projet.

Malheureusement, il fut impossible de les aider à ce moment car l'équipe de développement définissait seulement les fonctionnalités du projet. En effet, une technique telle que COSMIC-FFP a besoin des fonctionnalités détaillées du programme pour s'appliquer correctement.

3.3.4 Travaux futurs

Nous avons décidé de postposer le travail d'amélioration afin d'être capable de les aider, quelques mois plus tard, lorsqu'il serait possible de travailler avec l'ensemble complet des fonctionnalités du programme.

3.4 Conclusion

Ces interventions nous ont permis de constater la réelle limite de certaines petites entreprises, tant au niveau de leur structure, qu'au niveau de leurs processus logiciels. Avec une structure restreinte en terme de ressources et de capacité ainsi que des processus logiciels non formalisés, les petites entreprises font face à d'énormes difficultés lorsqu'elles tentent d'appliquer des démarches d'amélioration des processus logiciels non adaptées. En effet, si les recommandations ne sont pas proportionnelles au contexte de l'entreprise, cette dernière risque fortement de ne pas réussir à les appliquer correctement. Cette première constatation souligne d'ores et déjà l'importance du contexte de l'organisation qui permet d'éviter d'adopter une démarche inadéquate à la situation.

Nous avons également pu constater la présence de certaines caractéristiques Agile dans le fonctionnement de ces organisations. En effet, nous avons pu observer bon nombre de comportements Agile tels que, l'utilisation d'une communication tacite au sein d'un même espace de travail, une forte autonomie dans la gestion du travail, des attitudes positives face aux changements fréquents dans les exigences, etc. Ces comportements dénotent l'intérêt d'une démarche qualité dans les contextes Agile et dès lors une adaptation du modèle OWPL à ces contextes.

A posteriori, nous avons également le sentiment que la solution qualité proposée à Logilys, en ce qui concerne la gestion des exigences, est un peu trop formelle par rapport à la structure assez limitée de l'organisation. Nous pensons que l'analyse réalisée d'après l'adaptation du chapitre suivant, nous aurait été fortement bénéfique et nous aurait permis de détecter l'éventuelle inadéquation de la pratique proposée.

Ces observations renforcent l'importance de l'impact du contexte d'une organisation sur une démarche d'amélioration des processus logiciels tel que OWPL. L'adaptation qui suit prend dès lors tout son sens puisqu'elle fournit un guide qui permettra à l'évaluateur de prendre en compte le contexte de l'organisation et d'évaluer dans quelle mesure des caractéristiques Agile ou plus disciplinées sont présentes.

Chapitre 4

Adaptation du modèle OWPL

4.1 Introduction

Le « Standish Group International, Inc », lors de son étude du troisième trimestre 2004, a découvert que 71% des projets de développement logiciel échouaient, soit, dans la réalisation de leurs objectifs, soit, de par une annulation directe avant achèvement.

Nous pensons que ces échecs proviennent principalement d'une inadéquation de la méthodologie choisie. En effet, le contexte dans lequel le développement du projet se déroule est rarement pris en considération. Or, en tenant compte de ce contexte, on favorise l'efficacité du développement du logiciel en choisissant des pratiques qui sont adaptées aux circonstances dans lesquelles elles seront adoptées.

Certaines idées préconçues ont longtemps persisté dans l'ingénierie du logiciel. Il était de coutume de croire que différents types de méthodes ne pouvaient être combinés au sein d'un même projet et qu'il n'existait donc qu'un seul type de méthode possible pour un projet donné. Au contraire, le fait d'extraire le meilleur des diverses méthodes, et ce, en fonction du contexte du projet, permet d'améliorer sensiblement la qualité du logiciel ainsi développé.

Dans le modèle SW-CMM, « l'assurance qualité » est définie comme étant la conformité avec les spécifications et les processus, tandis que les méthodes Agile perçoivent essentiellement la qualité comme étant la satisfaction du client.

Le terme « discipliné » inclut dans sa définition, à la fois, la conformité avec les processus établis, qui est utilisée par les bureaucrates CMM, et le contrôle autonome, qui est utilisé par les esprits Agile. Une partie de la différence entre les approches disciplinées et les approches Agile, provient donc d'une prépondérance d'une des deux significations du terme « discipliné ». Il est difficile d'argumenter contre l'une ou l'autre et cela nous mène alors à penser que les deux types de disciplines sont nécessaires, et ce, dans différentes mesures selon le contexte. Le défi est alors de trouver, pour chaque type de projet, le juste équilibre entre l'agilité et la discipline.

Au vu de l'importance du contexte dans une démarche d'amélioration des processus logiciels d'une organisation et au vu de la présence de caractéristiques Agile dans ce contexte, l'adaptation du modèle OWPL que nous proposons, réside dans l'insertion d'une nouvelle phase d' « analyse du contexte » dans la démarche existante, plus précisément, dans la micro-évaluation. En effet, c'est lors de cette première étape de la démarche OWPL, que les premières recommandations doivent être réalisées avec une bonne connaissance du contexte de l'organisation, considérant que la seconde étape de la démarche, l'évaluation OWPL, se base sur ces résultats.

Cette analyse a pour objectif d'évaluer la proportion agilité-discipline de l'organisation, c'est-à-dire, évaluer quels sont les *critères du contexte* de l'organisation qui présentent des caractéristiques de type Agile et/ou de type discipliné pour un type de projet particulier. Une fois cette évaluation réalisée, elle permettra à l'évaluateur de recommander à l'organisation des pratiques qui sont adaptées à son contexte.

Dans la démarche existante, les résultats de la micro-évaluation sont analysés de manière informelle en fonction du contexte de l'organisation, juste avant de formuler les recommandations qui aboutiront à la rédaction d'un plan d'action. La nouvelle démarche ayant pour but de formaliser cette analyse du contexte, nous avons choisi d'insérer cette nouvelle phase juste avant la formulation des recommandations, à savoir, en la substituant à l'analyse informelle réalisée dans la démarche existante. En outre, le besoin d'information sur l'organisation étant nécessaire à l'évaluation des critères du contexte, cette analyse utilise les informations résultant du questionnaire de la micro-évaluation, réalisé dans la phase précédente.

Cette nouvelle phase propose donc un guide d'analyse du contexte qui permet d'assurer que l'évaluateur passe par toutes les étapes de l'étude du contexte de l'organisation afin de garantir une évaluation qui soit la plus juste possible.

Remarquons que cette analyse du contexte se base sur les critères (Home Grounds) de Boehm et Turner [9], qui permettent de mettre en évidence les différences de contexte selon qu'on utilise une méthode Agile ou disciplinée. Cependant, alors que Boehm et Turner proposent une méthode d'évaluation basée sur les risques, sans toutefois expliciter comment mesurer ces risques, la méthode proposée ci-après fournit un guide qui aide l'évaluateur à mesurer les *critères du contexte* et qui permet d'obtenir une vision complète du contexte.

Dans la section suivante, nous présenterons les différents *critères du contexte* qui seront utilisés par le guide afin d'évaluer si le contexte présente des caractéristiques plus Agile ou plus disciplinées. La section qui suit nous indiquera la méthodologie à suivre pour évaluer chacun de ces critères et pour évaluer le contexte général de l'organisation par rapport à tous ces critères.

4.2 Critères du contexte

Chacune des approches, Agile ou disciplinée, est plus adéquate dans son environnement habituel. En effet, les méthodes disciplinées sont généralement utilisées pour des systèmes larges et complexes avec des attributs de sécurité et de haute fiabilité. En outre, leurs exigences et leur environnement doivent être relativement stables. Les méthodes Agile sont, à l'inverse, généralement plutôt utilisées pour des systèmes plus petits avec des équipes de développement réduites. Leurs exigences et leur environnement sont assez volatiles, mais le client et les utilisateurs sont facilement disponibles.

La nature complexe du développement logiciel et la variété des méthodes rendent la comparaison entre les approches disciplinées et Agile difficile et imprécise. Cependant, nous pouvons énoncer plusieurs critères importants du contexte des projets logiciels pour lesquels il y a des différences selon qu'on utilise une méthode disciplinée ou Agile.

Les critères du contexte qui suivent vont donc permettre à l'évaluateur de mettre en balance les caractéristiques que présente le contexte de l'organisation pour un type de projet, et ce, par rapport à chacun de ces critères, afin d'avoir une évaluation de la proportion agilité-discipline pour chacun d'eux. Nous développerons la manière de mesurer ces critères dans la section suivante.

Illustrons, dès à présent, les caractéristiques des différents critères selon qu'on se situe dans un contexte Agile ou discipliné.

4.2.1 Critères généraux

Objectifs principaux

Ce critère concerne les objectifs principaux de ce type de projet, qui définissent les grandes lignes de ce que ce dernier va apporter à l'organisation ou au client, et la manière dont il va s'agencer pour supporter la stratégie choisie.

- Agile : *« Apport de valeur rapide et acceptation du changement »*

  La plus grande priorité des pratiques Agile est de satisfaire le client, le plus tôt possible, en lui distribuant progressivement du logiciel de valeur. En outre, ces pratiques prônent plutôt une certaine réactivité qui permet d'inclure rapidement les demandes de changement dans les exigences.

- Discipliné : *« Haute fiabilité et stabilité »*

  Les méthodes disciplinées supportent ces objectifs via une forte planification et des stratégies de vérification et de validation des exigences. La concentration est dès lors portée sur la prédiction et la stabilité à travers la standardisation, la mesure et le contrôle des activités.

Taille

Ce critère concerne le nombre de personnes qui participent à ce type de projet, la complexité et la taille estimée du logiciel à développer.

– Agile : *« Petite équipe et petit projet »*

  Les processus Agile fonctionnent beaucoup mieux avec des équipes de petite ou de moyenne taille, qui travaillent sur des applications relativement petites. En général, une équipe d'une dizaine de personnes est recommandée et il est plutôt risqué de considérer une équipe plus importante.

– Discipliné : *« Grande équipe et gros projet »*

  Les méthodes traditionnelles sont plus adaptées aux gros projets. Les plans, la documentation et les processus fournissent une meilleure communication et une meilleure coordination entre les membres de cette équipe importante.

Environnement

Ce critère concerne le type de variables externes qui ont une influence sur ce type de projet.

– Agile : *« Changeant »*

  Les approches Agile s'appliquent plus aux environnements fortement changeants. Elles considèrent les organisations comme des systèmes adaptables dans lesquels les exigences sont émergentes plutôt que spécifiables à priori.

– Discipliné : *« Stable »*

  Les méthodes disciplinées fonctionnent beaucoup mieux lorsque les exigences sont déterminées à l'avance et lorsqu'elles restent relativement stables.

4.2.2 Critères de gestion

Relation client

Ce critère concerne le type de relation que l'organisation entretient avec son client dans le cadre de ce type de projet.

– Agile : *« Relation étroite »*

  Le client est en étroite relation avec l'équipe de développement et avec les utilisateurs du système qu'il représente afin de permettre au projet de fournir de la valeur le plus rapidement possible à l'organisation. Cette forme de relation requiert donc une présence permanente du client auprès des développeurs.

– Discipliné : *« Relation contractuelle »*

  La base de la relation entre l'équipe de développement et le client est fondée sur une forme de contrat accepté mutuellement. Les problèmes sont analysés au préalable et les solutions sont formalisées dans ce document qui reprend l'ensemble des exigences du client. Dès lors, le client sait ce qu'il est en droit d'attendre et les développeurs savent ce qu'ils ont à faire. Cette forme de relation ne requiert donc qu'une présence occasionnelle du client auprès des développeurs.

Planification et contrôle

Ce critère concerne la manière dont la planification et le contrôle sont effectués pour ce type de projet.

– Agile : *« Planification intériorisée et contrôle qualitatif »*

  La vitesse et l'agilité des projets Agile proviennent principalement de la planification délibérée de l'équipe qui instaure les opérations sur base d'une connaissance interpersonnelle tacite. Le contrôle s'effectue alors sur base de cette vision commune et sur la progression réelle du travail.

– Discipliné : *« Planification documentée et contrôle quantitatif »*

  Les plans représentent une grande partie de la documentation requise et servent de base pour une connaissance documentée explicite qui soutient les processus et fournit un moyen de communication assez large. Les méthodes disciplinées se basent fortement sur ces plans qui documentent le processus (planning, procédures) et le produit (exigences, architecture, standards) afin que les membres de l'équipe soient coordonnés. Le contrôle du progrès est alors évalué selon les plans établis.

Communication

Ce critère concerne la manière dont les membres de l'organisation collaborent et communiquent entre eux pour ce type de projet.

– Agile : *« Interpersonnelle et tacite »*

  Les méthodes Agile se basent sur la communication de personne à personne. L'accent est mis sur les individus et les interactions et dénote une claire préférence pour la collaboration entre les membres de l'équipe. L'investissement dans ce style de communication rend possible le partage d'une connaissance tacite commune à l'équipe.

– Discipliné : *« Documentée et explicite »*

  Les méthodes disciplinées se basent sur une communication à sens unique. La connaissance est documentée, en décrivant le processus, les rapports de progression, etc., et communiquée explicitement d'une entité à une autre.

4.2.3 Critères techniques

Exigences

Ce critère concerne la manière dont les exigences sont exprimées et gérées pour un projet de ce type.

– Agile : *« Informelles »*

  Les exigences sont exprimées de manière informelle et sont adaptables. De par leurs cycles itératifs courts, les pratiques Agile déterminent les changements nécessaires à apporter et les solutionnent dans l'itération suivante. Le fait de déterminer l'ensemble des exigences qui sera inclus dans l'itération suivante, est réalisé en concertation avec les clients qui expriment leurs priorités.

– Discipliné : *« Formelles »*

  Les exigences sont formalisées en une spécification complète qui forme la base pour assurer la cohérence, la traçabilité, le développement et les tests.

Développement

Ce critère concerne la manière dont l'organisation développe du logiciel pour un projet de ce type.

– Agile : *« Conception simple »*

  Les Agilistes encouragent les développeurs à rendre la conception la plus simple possible à chaque opportunité. Cela sous-entend de développer uniquement ce qui est nécessaire à l'itération en cours et de ne pas anticiper de nouvelles fonctionnalités.

– Discipliné : *« Conception architecturale »*

  Une quantité importante d'effort est consacrée à l'étude et à la définition d'une architecture robuste qui supportera la vision du système telle que prédéfinie dans les phases d'analyse.

Tests

Ce critère concerne la manière dont l'organisation gère les tests relatifs à ce type de projet.

– Agile : *« Tests continus »*

  Les méthodes Agile organisent le développement en petits incréments, elles utilisent des techniques, telles que la revue de code ou encore la programmation à deux, afin d'éviter les erreurs et ce, au fur et à mesure du développement. Elles développent également des tests exécutables qui remplacent les exigences et qui s'appliquent dès le départ et de manière continue durant le développement.

– Discipliné : *« Tests de spécification »*

  Les méthodes disciplinées s'assurent de la correction du système en vérifiant les exigences et l'architecture spécifiées plus tôt dans le processus de développement. Elles investissent également dans des procédures de test automatisées.

4.2.4 Critères du personnel

Développeurs

Ce critère concerne le type de qualification des individus de l'organisation pour un projet de ce type.

– Agile : *« Hautement qualifiés »*

  Les méthodes Agile nécessitent d'avoir des personnes hautement qualifiées avec de l'expérience. Il est important pour ces méthodes d'inclure des développeurs talentueux, avec le sens de la communication et avec des compétences élevées.

– Discipliné : *« Qualifiés »*

  Les méthodes disciplinées fonctionnent évidemment mieux avec des personnes hautement qualifiées mais peuvent néanmoins très bien fonctionner avec des personnes un peu moins qualifiées ou expérimentées grâce à la planification du projet et l'architecture prédéfinie du système.

Culture

Ce critère concerne le type de culture qui règne au sein de l'organisation pour un projet de ce type.

– Agile : *« Liberté et confiance »*

  Dans la culture Agile, les personnes disposent d'un degré important de liberté et de confiance pour leur permettre de définir et de résoudre elles-mêmes les différents problèmes. Il s'agit d'un environnement ouvert, où l'on attend de chacun qu'il réalise le travail nécessaire pour le succès du projet. Cela comprend donc, entre autres choses, la prise en charge spontanée de tâches non notifiées.

– Discipliné : *« Politiques et procédures »*

  Dans la culture disciplinée, les personnes suivent des politiques et des procédures claires qui définissent leur rôle dans l'équipe. Les tâches de chacun sont donc bien définies et l'on attend des personnes qu'elles accomplissent le travail qui leur a été défini afin de l'intégrer facilement dans l'ensemble du système.

4.3 Méthodologie

Lors de l'établissement du plan d'action de la micro-évaluation formulant les recommandations qui permettront à la société d'améliorer son processus logiciel, l'évaluateur analyse, de manière informelle, les résultats de la micro-évaluation en fonction du contexte de l'organisation évaluée.

La méthodologie abordée dans cette section propose un guide, à utiliser juste avant la rédaction du plan d'action de la micro-évaluation, qui permet d'aider l'évaluateur à déterminer s'il est plus approprié de recommander des pratiques de type Agile et/ou des pratiques plus disciplinées pour un projet particulier. En effet, il permet d'évaluer la proportion agilité-discipline de l'entreprise sur un type de projet afin de pouvoir suggérer l'application de pratiques en conséquence.

Ce guide est un moyen pragmatique de réconcilier les forces et les faiblesses des méthodes Agile et disciplinées, qui incite à utiliser les avantages de chacune des deux méthodes lorsque la situation l'exige, et qui aide donc les développeurs à réaliser le bon équilibre pour leurs projets.

Le guide proposé se base sur les critères précédents qui représentent donc un ensemble de conditions sous lesquelles l'utilisation de l'un ou l'autre type de pratique augmente la probabilité de succès du projet. En effet, plus les conditions particulières du projet diffèrent des conditions génériques d'une méthode, plus l'utilisation de pratiques de ce type de méthode met en péril la réussite du projet.

4.3.1 Nouvelle démarche de la micro-évaluation

Comme l'illustre le schéma de la figure 4.1, notre méthodologie propose de formaliser l'*analyse du contexte* en une phase, située entre les phases d'analyse des informations et de rédaction du plan d'action de la micro-évaluation existante. La nouvelle micro-évaluation comprend donc les quatre phases suivantes, dont la troisième qui a été introduite :

1. questionnaire : Le questionnaire de la micro-évaluation existante permet de collecter des informations sur les pratiques logicielles dans l'organisation.

2. analyse des informations : Les informations collectées sont analysées et résumées par rapport aux six axes de la micro-évaluation existante.

3. *analyse du contexte* : Les résultats sont étudiés en fonction de l'analyse du contexte qui est formalisée au moyen du guide décrit dans la sous-section suivante et représenté à la figure 4.1.

4. plan d'action : Des recommandations sont formulées pour permettre la rédaction d'un plan d'action avec les responsables de l'organisation.

Nous remarquerons que, pour ne pas alourdir la structure de la démarche OWPL, et parce que nous jugeons l'information suffisante, l'analyse du contexte se réalise sur base des résultats qui proviennent de la phase précédente, à savoir, l'analyse des informations.

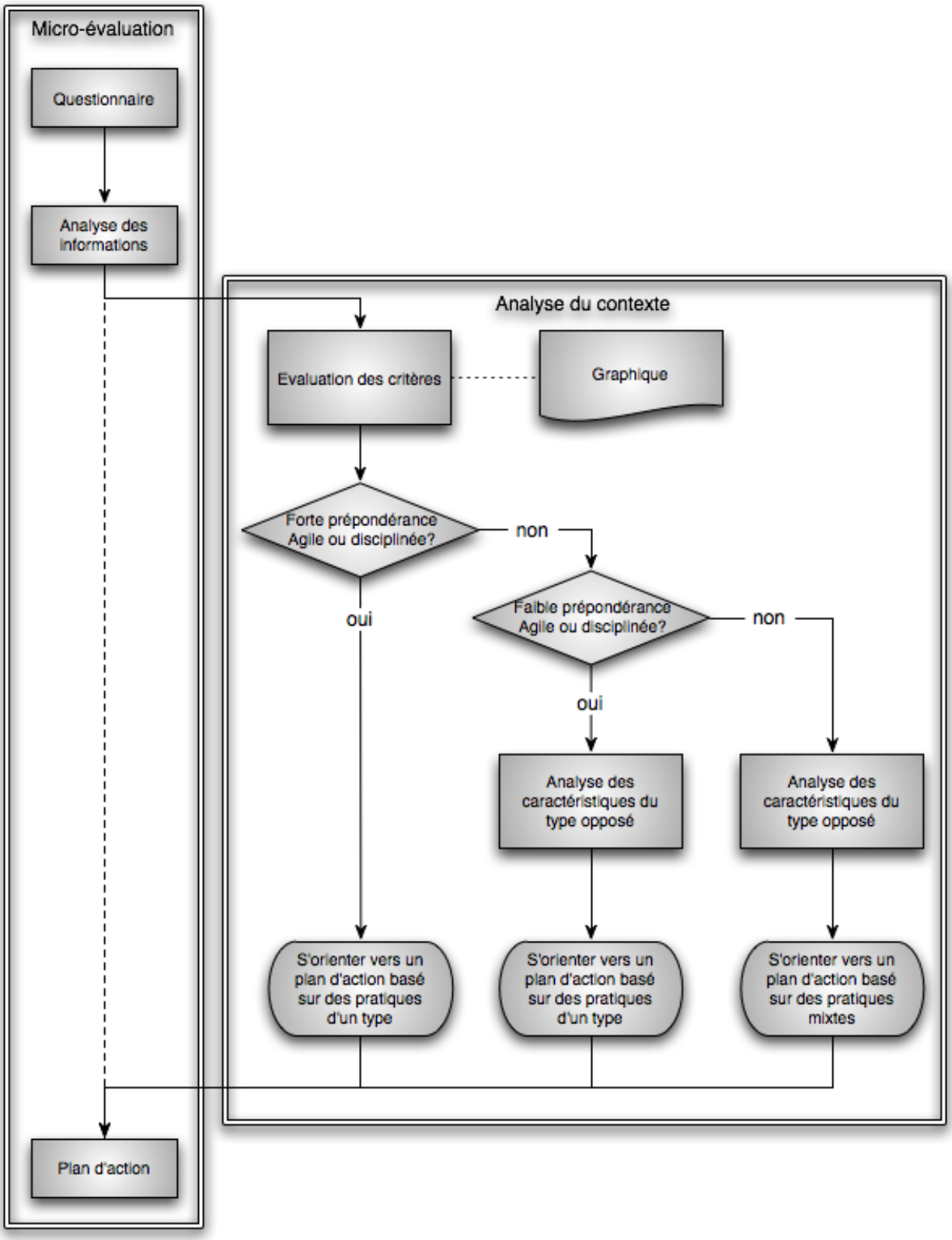

FIG. 4.1 – Guide d'analyse du contexte

4.3.2 Analyse du contexte

Evaluation des critères

L'évaluation du contexte de l'organisation est basée sur l'évaluation des critères précédents. Il s'agit d'évaluer, pour chaque critère, dans quelle proportion le type de projet d'une organisation présente des caractéristiques plus Agile ou plus disciplinées.

A cette fin, notre méthodologie utilise l'outil graphique représenté à la figure 4.2, où chaque axe représente un critère. Plus les valeurs des axes se situent près du centre, plus les chances de succès augmentent avec des pratiques de type Agile. A l'inverse, si les valeurs des axes se situent près du pourtour, les chances de succès augmentent alors avec des pratiques de type discipliné.

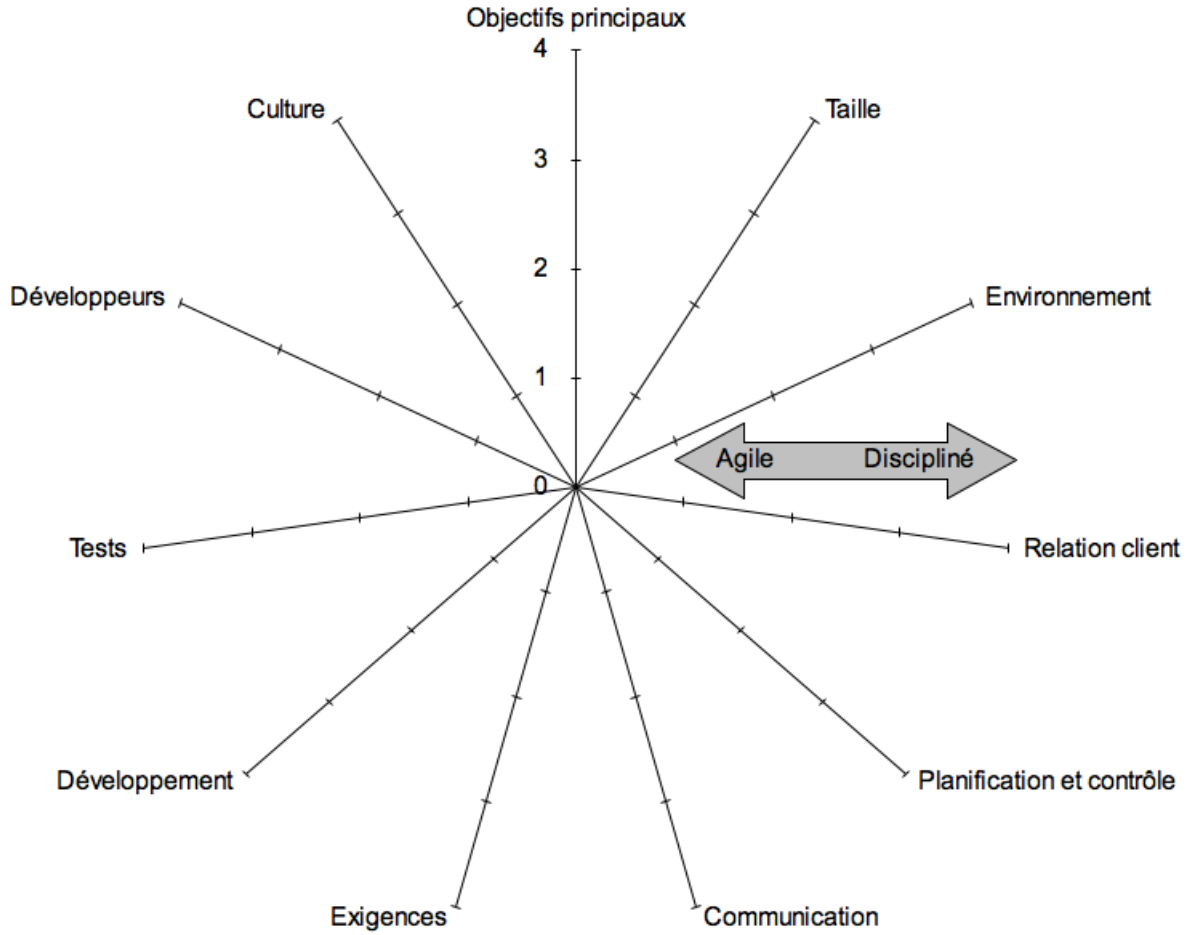

FIG. 4.2 – Graphique d'évaluation des critères

La graduation d'un axe va de 0 à 4 et la sémantique suivante est à considérer par l'évaluateur afin de mesurer chacun des critères :

- **0 & 4** : ces deux valeurs ont des significations diamétralement opposées. La valeur **0** indique clairement que le contexte de l'organisation, par rapport au critère considéré, présente exactement et uniquement des caractéristiques de type Agile. A l'inverse, la valeur **4** indique des caractéristiques de type discipliné.
- **2** : cette valeur indique que le contexte de l'organisation, par rapport au critère considéré, présente des caractéristiques, à la fois de type Agile et de type discipliné, de manière équitable.
- **1 & 3** : ces deux valeurs sont les valeurs intermédiaires entre les deux extrêmes (0 & 4) et le juste milieu (2). La valeur **1** indique que le contexte de l'organisation, par rapport au critère considéré, présente des caractéristiques de type Agile, à quelques exceptions près. La valeur **3** indique des caractéristiques de type discipliné, à quelques exceptions près.

Forte prépondérance Agile ou disciplinée ?

Le graphique résultant de l'évaluation précédente fournit une vision complète du contexte de l'organisation par rapport à tous les critères. Cette vision va permettre d'analyser globalement quelle est la proportion agilité-discipline de l'organisation, pour un type de projet, afin de savoir quel(s) type(s) d'approche (Agile ou discipliné) est (sont) le(s) plus approprié(s).

Lors de ce premier point de décision, il s'agit de déterminer s'il y a prépondérance forte de l'un des deux types d'approche dans le contexte de l'organisation. Une prépondérance forte signifie que le contexte présente des caractéristiques qui sont catégoriquement d'un seul type d'approche, pour l'ensemble des critères considérés.

Dans le cas où l'évaluateur constate une telle prépondérance, il va pouvoir étudier les résultats de la phase d'analyse des informations de la micro-évaluation en fonction de ce contexte. Dès lors, durant la phase de rédaction du plan d'action, l'évaluateur pourra formuler des recommandations sur base des pratiques du type d'approche prépondérant.

Dans le cas où aucune prépondérance forte ne peut être déterminée, l'évaluateur considérera l'étape suivante.

Faible prépondérance Agile ou disciplinée ?

Lors de ce deuxième point de décision, il s'agit de déterminer s'il y a prépondérance faible de l'un des deux types d'approche dans le contexte de l'organisation. Une prépondérance faible signifie que l'évaluateur ne peut ni déterminer une prépondérance forte, ni déterminer une équivalence des types d'approche. Une équivalence des types d'approches signifie que le contexte présente autant de caractéristiques d'un type d'approche que de l'autre type, pour l'ensemble des critères considérés.

Dans le cas où l'évaluateur détermine une équivalence des types d'approches, il va pouvoir étudier les résultats de la phase d'analyse des informations de la micro-évaluation en fonction de ce contexte. Dès lors, durant la phase de rédaction du plan d'action, l'évaluateur pourra formuler des recommandations sur base des pratiques des deux types d'approches. Chaque pratique sera alors recommandée par rapport aux critères où son type est prépondérant.

Dans le cas où une prépondérance faible peut être déterminée, il va pouvoir étudier les résultats de la phase d'analyse des informations, de la micro-évaluation, en fonction de ce contexte. Dès lors, durant la phase de rédaction du plan d'action, l'évaluateur pourra formuler des recommandations sur base des pratiques du type d'approche prépondérant.

Analyse des caractéristiques du type opposé

Dans le cas d'une prépondérance faible ou d'une équivalence, il s'agit d'analyser les caractéristiques du type d'approche opposé, présentes dans le contexte de l'organisation. La considération de ces caractéristiques permettra alors, durant la phase de rédaction du plan d'action avec les responsables de l'organisation, de recommander des pratiques qui prendront en compte ces éléments du contexte. En effet, cela permettra d'éviter de recommander des pratiques contradictoires ou conflictuelles avec celles présentent dans l'organisation.

Dans le cas d'une prépondérance faible, il s'agit d'analyser les caractéristiques du type d'approche non prépondérant.

Dans le cas d'une équivalence, il s'agit d'analyser les caractéristiques des deux types d'approches.

4.4 Conclusion

Le changement étant de plus en plus présent dans les projets informatiques, la nécessité d'incorporer de l'agilité dans les projets de grande envergure se fait de plus en plus ressentir, mais le besoin de discipline pour ces larges systèmes logiciels à forte complexité reste néanmoins présent.

Le succès et la viabilité d'un développement logiciel requiert, dès lors, à la fois de la discipline et de l'agilité, car cela nécessite de l'adaptation au changement, à l'environnement ainsi qu'à un ensemble d'exigences. Il est donc crucial de pouvoir comprendre quand un développement plus Agile ou plus discipliné est approprié. Le fait de ne pas se limiter à une seule méthode et de s'adapter au projet, souligne l'importance de l'analyse du contexte.

Différentes combinaisons de discipline et d'agilité sont donc possibles d'un projet de développement à l'autre. Trouver la bonne quantité de rigueur ou de processus est un problème auquel l'évaluateur doit faire face en prenant en compte différents critères qui vont l'aider à évaluer le contexte du projet dans l'optique d'atteindre le bon équilibre entre agilité et discipline pour ce projet.

Dans le chapitre suivant, nous décrirons l'expérimentation auprès d'une entreprise wallonne de notre proposition d'adaptation de la démarche OWPL, afin d'illustrer l'utilisation de la méthode proposée, de la valider et de critiquer le travail réalisé.

Chapitre 5

Validation

5.1 Introduction

Afin d'illustrer l'application de l'adaptation de la démarche OWPL sur une petite entreprise, nous avons décidé d'expérimenter la nouvelle démarche de la micro-évaluation avec une PME wallonne, à savoir, Open Engineering S.A..

Open Engineering est une société de développement de logiciel, qui fait partie du groupe Samtech, et qui est composée de huit ingénieurs. Samtech est une spin-off, créée il y a 20 ans et Open Engineering est une spin-off, créée en 2001. Elles sortent toutes deux du même laboratoire de l'université de Liège, et traitent des thématiques similaires.

Leur spécificité réside dans le développement d'un logiciel de simulation numérique qui permet de faire du prototypage virtuel. Dans ce dessein, Open Engineering dispose d'une application centrale unique, appelée Object Oriented Finite Element Led by Interactive Executor (OOFELIE), qui représente le logiciel de simulation afin de réaliser les calculs. La raison d'être de Open Engineering est donc de développer OOFELIE, de l'industrialiser, après dix années de recherche à l'université, en générant des produits qui répondent à des besoins du marché et de continuer à la faire évoluer en la mettant à disposition de partenaires universitaires et dans des centres de recherche.

Pour faire évoluer OOFELIE, Open Engineering industrialise les fonctionnalités existantes, en industrialisant les fonctionnalités récupérées de l'université et des centres de recherche, ou alors ajoute des fonctionnalités spécifiques. Des contrats sont établis notamment avec la European Space Agency (ESA), avec la commission européenne, avec les centres de recherche, ou encore, avec tout autre client, tels que des bureaux d'étude, intéressés par un logiciel de simulation de contraintes multiphysiques.

Open Engineering base l'ensemble de ses projets sur son application centrale, OOFELIE. Selon les besoins du client, ils adaptent alors l'interface graphique et réduisent les fonctionnalités au strict nécessaire. En général, Open Engineering essaye autant que possible d'anticiper l'avenir en développant de nouveaux produits.

Leurs plus gros projets se situent dans le développement de logiciels de simulation pour les Micro Electrical Mechanical Systems (MEMS), qui représentent pour eux un marché très important puisqu'ils en sont déjà à cinq produits développés, plus un en cours. Leur application étant multidisciplinaire et multichamps, cela leur permet de se spécialiser dans le couplage, c'est-à-dire, l'analyse de plusieurs problèmes simultanément. Cela est fondamental dans le monde des MEMS, car il faut alors prendre en compte un ensemble d'effets.

Dans la section suivante, nous allons décrire les trois premières phases de la nouvelle démarche de la micro-évaluation appliquées à Open Engineering, à savoir, (1) la collection d'informations sur les pratiques logicielles de Open Engineering, (2) l'analyse de ces informations selon les six axes de la micro-évaluation existante, et enfin, (3) l'analyse du contexte sur base de ces résultats.

5.2 Application de la nouvelle démarche

5.2.1 Questionnaire

Lors d'une entrevue avec Monsieur Pascal De Vincenzo, développeur et gestionnaire de projets chez Open Engineering, nous avons pu récolter des informations sur les pratiques logicielles de Open Engineering. Cet entretien s'est déroulé sous forme d'interview réalisée avec le questionnaire de la micro-évaluation existante, repris à l'annexe E de ce document.

5.2.2 Analyse des informations

Sur base des réponses fournies par Monsieur De Vincenzo au questionnaire de la micro-évaluation existante, nous avons pu résumer les informations collectées par rapport aux six axes de la micro-évaluation existante. Ensuite, afin de réaliser une vue synoptique de l'évaluation des pratiques logicielles de Open Engineering, nous avons interprété ces résultats en référence à la grille d'évaluation figée de la micro-évaluation existante.

Gestion de la qualité

Open Engineering a commencé la certification ISO 9001[1] en mettant en place certaines procédures organisationnelles, qui l'ont aidée à structurer la société. Cependant, Open Engineering n'a pas continué la phase de certification car elle n'a pas de demande en la matière. En effet, ses procédures sont acceptées par l'ESA, sans être pour autant certifiées, qui impose certaines contraintes supplémentaires, certaines normes de qualité, qui sont cependant adaptées pour s'appliquer de manière plus réaliste au développement des logiciels de simulation de Open Engineering, compte tenu de sa petite structure.

Open Engineering pratique des tests d'intégration et des tests industriels, c'est-à-dire, des tests acceptés dont on connaît les bons résultats. Les tests unitaires ne sont pas encore réalisés, mais des outils ont été trouvés qui permettraient éventuellement de les exécuter à moindres coûts.

Open Engineering cherche toujours à s'améliorer en terme de qualité. Lorsqu'ils constatent une possibilité d'amélioration, ils essayent de trouver des solutions quand le problème est réellement présent et s'ils ont les ressources pour le faire.

[1] La norme internationale ISO 9001 spécifie les exigences organisationnelles relatives au système de management de la qualité.

Relation Clients

Open Engineering établit un cahier des charges avec ses clients. Ce document officiel, rédigé en début de projet, représente le contrat entre les deux parties, et contient le planning des fonctionnalités demandées en fonction des besoins du client. Les exigences du client y sont donc inscrites, de commun accord avec ce dernier, afin d'expliciter les fonctionnalités désirées.

Open Engineering gère les demandes de changement grâce à des réunions organisées régulièrement avec le client, en les actant dans un PV, qui permet de garder trace des conventions. Certaines demandes sont cependant parfois plus formalisées, lorsque le développement est plus spécifique ou s'il y a de la recherche, c'est-à-dire, lorsque l'équipe de développement sait ce qu'il faudrait changer mais pas comment il faudrait le changer.

Open Engineering planifie un ensemble de réunions avec le client, mais des réunions supplémentaires peuvent également s'organiser à la demande du client. Lors de ces réunions, l'équipe de développement peut valider les exigences du client en lui présentant la version en cours d'itération.

Gestion des Sous-Traitants

Open Engineering est en contact avec un centre de recherche en Argentine, mais ce centre est plutôt un partenaire privilégié qu'un sous-traitant. Ce centre participe depuis le début au développement d'OOFELIE. Les interactions sont nombreuses avec ce centre et il arrive qu'il participe à des développements avec Open Engineering. Le centre a un accès direct aux sources, ce qui lui permet de réaliser le développement et de prévenir Open Engineering, une fois le développement terminé, qui se charge de réaliser les tests.

Développement et gestion de projet

Open Engineering se base sur certains concepts clés tels que le développement itératif par incréments, avec des temps de cycles très courts, en fonction des fonctionnalités à développer. Les itérations se basent sur des petits cycles de vie en « V ». Selon les projets, les phases sont délimitées par la production d'un livrable, tel que des documents, des notes techniques,... En général, les projets internes ne font pas l'objet de réels livrables, seul le tutorial sera modifié avec l'ajout des fonctionnalités.

L'équipe de développement utilise plutôt des techniques informelles pour représenter ce que le client veut, comme des « story cards », mais ne s'encombre pas de formalisme, tel qu'UML. Open Engineering utilise également parfois le refactoring et la relecture de code croisée, mais pas pour tous les projets car tous les développeurs ne sont pas capables de comprendre ce qu'un autre a fait dans un domaine plus poussé. Pour certains projets, il peut leur être nécessaire en cas de problèmes, de revoir le code ou de faire appel à des experts de l'université pour réfléchir sur la problématique. En outre, l'équipe a convenu d'un ensemble de règles de programmation adaptées d'un guide de bonnes pratiques selon les besoins.

Open Engineering divise ses projets en séquences, grâce à une ligne du temps définie à priori, où les parties à réaliser, en séquentiel ou en parallèle, sont délimitées par des « milestones », qui aboutissent à la rédaction d'un rapport, qui décrit les fonctionnalités introduites, de manière plus ou moins détaillée en fonction du projet. Lorsqu'un non-sens ou une dérive est détectés, Open Engineering est capable d'ajuster le planning initial et ce, avec l'accord du client, en lui proposant des solutions alternatives.

Open Engineering organise des réunions mensuelles d'avancement des projets, qui lui permettent d'avoir une vision directe, tous les mois, de la façon dont les projets évoluent, afin de pouvoir mieux anticiper les éventuelles dérives. Des rapports de clôture des milestones peuvent également être consultés afin de connaître l'état d'avancement. Si besoin est, une réunion spécifique peut être organisée afin d'analyser un problème particulier et de lui trouver une solution. En outre, des interactions permanentes ont lieu entre les membres de l'équipe, ce qui permet de résoudre certains problèmes « sur le tas ».

Le besoin en communication est donc très fort, car il y a des personnes avec beaucoup d'expérience, qui connaissent parfaitement l'architecture de l'application, mais également des personnes moins expérimentées qui doivent pouvoir se référer aux personnes plus expérimentées.

Open Engineering utilise l'interface graphique du groupe Samtech, qui a développé une interface utilisateur dédiée au calcul numérique et à l'interfaçage d'algorithmes de résolution. Une personne de chez Open Engineering se charge d'adapter cette interface pour qu'elle fonctionne avec OOFELIE. Les deux sont, dès lors, totalement découplées. Vu qu'il s'agit de méthodes de résolution et d'algorithmes, et vu le découplage avec l'interface graphique, Open Engineering dispose de scripts qui représentent les tests d'intégration et les tests industriels, et qui permettent de savoir si la version en cours d'un projet est en bon état ou pas. Un système de compilation et de test nocturne permet de qualifier OOFELIE toutes les nuits sur plusieurs plates-formes et permet de générer des rapports qui informent de la bonne exécution ou non de ces tests. Cela permet donc de savoir si les développements réalisés sont portables et si certaines modifications apportées endéans les 24h ont provoqué un problème ou non.

Gestion des produits

Open Engineering possède un système de gestion de version très facile qui permet de créer des branches de développement, c'est-à-dire CVS, qui permet de travailler à plusieurs sur les mêmes fichiers. La flexibilité fournie par cet outil leur est nécessaire étant donné que OOFELIE est partagée avec les centres de recherche qui ont accès aux sources et qui ont, eux aussi, leur propre développement.

Open Engineering utilise également une documentation simplifiée qui est embarquée dans le code et qui est extraite à partir d'un outil, tel que Doxygen, qui permet d'extraire de l'information statique sur le comportement de l'application grâce à l'intégration, par le développeur, de balises dans le code source. En outre, certains rapports sont rédigés en clôture de certains milestones pour décrire les fonctionnalités de ce qui a été fait, et ce, de manière plus ou moins détaillée en fonction du projet.

Cette documentation n'est pas toujours suffisante pour des projets, tels que ceux qui sont établis avec l'ESA, qui impose des contraintes assez fortes. En effet, l'ESA exige une description complète de ce qui va être fait, de la manière dont cela va être fait, de l'implémentation, des tests réalisés,... Ce qui alourdit fortement les processus de développement de la spin-off.

Formation et gestion des ressources humaines

Open Engineering a une politique de formation pour ses ingénieurs. En effet, il est préférable, selon eux, d'avoir un bon ingénieur, qui connaît bien sa théorie scientifique et qui va être formé à la programmation, plutôt qu'un informaticien, qui connaît bien la programmation mais qui a des difficultés au niveau de certaines connaissances scientifiques, comme par exemple, les connaissances physiques.

Les employés étant des ingénieurs, ceux-ci ne savent pas nécessairement programmer. Les formations proposées en interne sont donc réalisées pour leur permettre d'apprendre à programmer et de s'accoutumer à l'environnement de travail de Open Engineering. En outre, des formations externes en langue et en gestion sont également réalisées si nécessaire.

Synoptiques

Le synoptique détaillé donne une représentation graphique de l'interprétation de chaque question de l'évaluation par l'évaluateur. Le synoptique résumé propose une représentation plus générale en fonction des six axes définis précédemment. Plus la zone de couverture est étendue, meilleure est la couverture de la pratique. Le synoptique détaillé et le synoptique résumé sont représentés respectivement aux figures 5.1 et 5.2. Remarquons l'absence de l'axe, « Gestion des sous-traitants », vu l'absence de sous-traitant chez Open engineering.

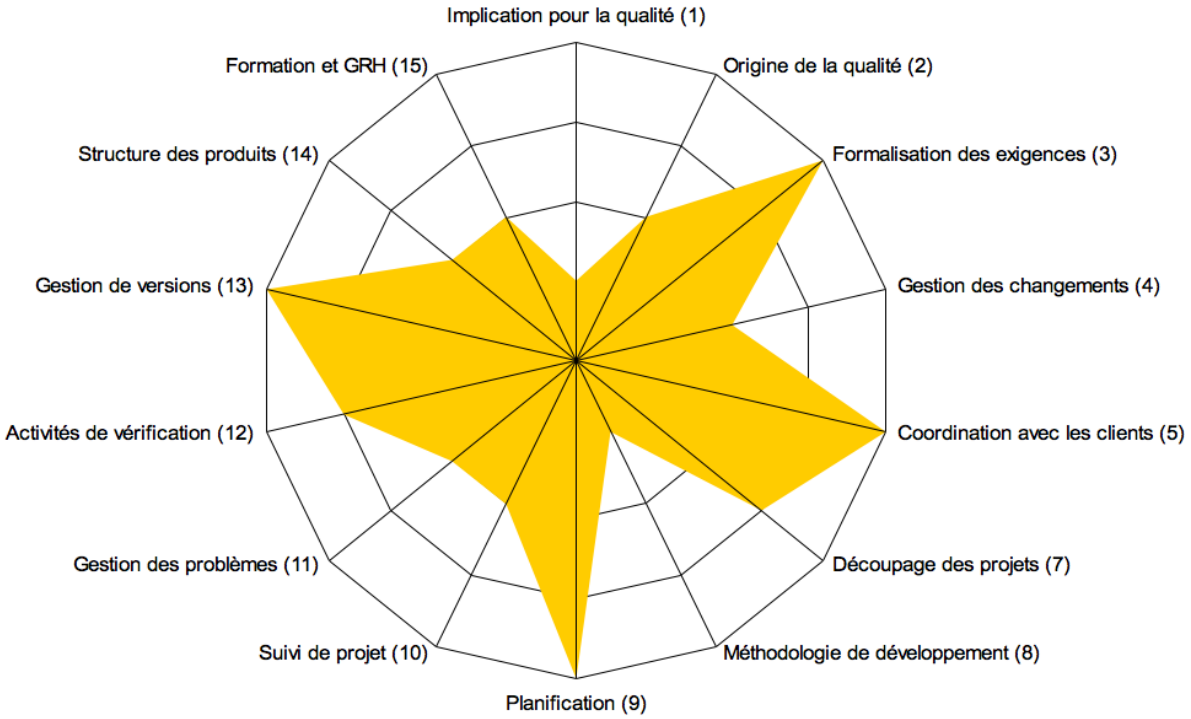

Fig. 5.1 – Synoptique détaillé - Open Engineering

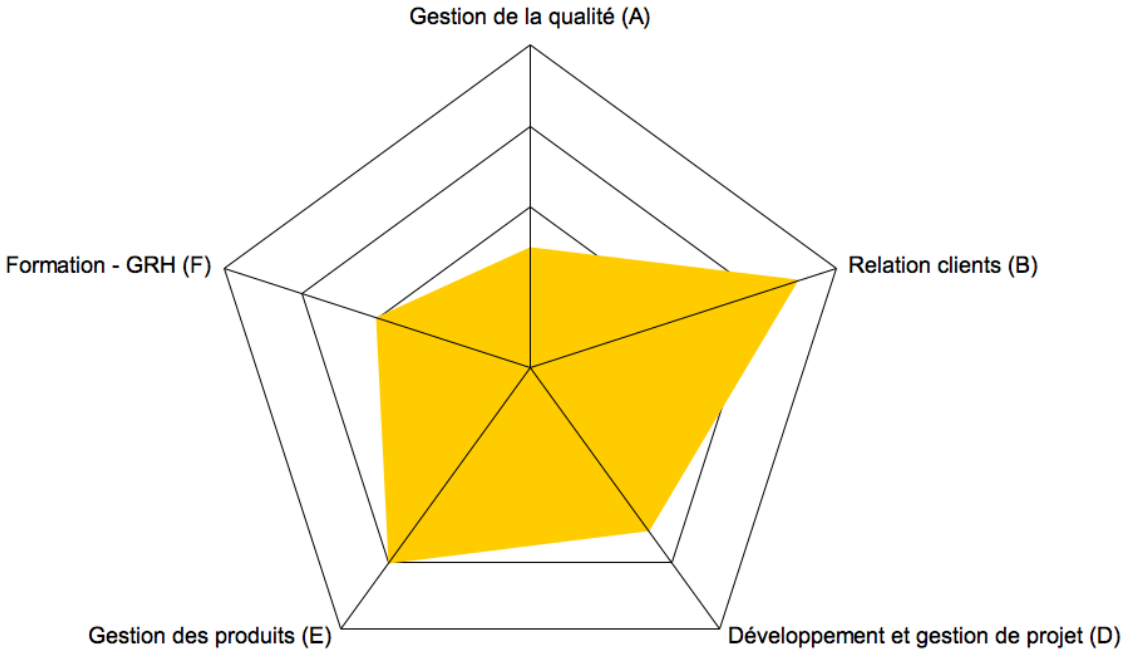

FIG. 5.2 – Synoptique résumé - Open Engineering

5.2.3 Analyse du contexte

Afin de réaliser l'analyse du contexte de Open Engineering, la nouvelle démarche de la micro-évaluation propose d'utiliser le guide présenté dans le chapitre précédent. Les résultats que nous venons d'analyser, dans les sections précédentes, vont désormais nous servir de base d'information pour l'évaluation des critères.

Evaluation des critères

Nous avons pu remarquer que Open Engineering base l'ensemble de ses projets sur son application centrale, OOFELIE. En outre, les processus de développement logiciels varient très peu d'un projet à l'autre, au vu des degrés d'institutionnalisation effectifs des pratiques au sein de la PME, représentés dans le questionnaire de la micro-évaluation existante.

Dès lors, les plus gros projets de Open Engineering se situant dans le développement de logiciels de simulation pour les MEMS, nous pouvons considérer ce « type de projet », comme représentatif de l'ensemble des projets de développement de l'organisation. Evaluons dès à présent, chacun des critères du contexte de Open Engineering.

1. *Objectifs principaux*

   Open Engineering tente de satisfaire le client le plus tôt possible, en organisant un ensemble de réunions avec ce dernier, lors desquelles, l'équipe peut présenter la version en cours d'itération. C'est également lors de ces réunions, que les demandes de changements éventuelles du client peuvent être traitées. Toutefois, Open Engineering vérifie les exigences grâce à son système de test nocturne, et, planifie sérieusement ce type de projet, en le divisant en séquences sur une ligne du temps définie à priori.

Le contexte de l'organisation, par rapport aux objectifs principaux de ce type de projet, présente donc des caractéristiques de type Agile, à quelques exceptions près. Dès lors, nous évaluerons la valeur de ce critère à **1**.

2. *Taille*

   Open Engineering dispose d'une petite équipe, puisque composée de huit personnes, qui travaille sur des applications relativement petites, comme le développement de logiciels de simulation pour les MEMS, basés sur leur application centrale OOFELIE.

   Le contexte de l'organisation, par rapport à la taille de ce type de projet, présente donc exactement et uniquement des caractéristiques de type Agile. Dès lors, nous évaluerons la valeur de ce critère à **0**.

3. *Environnement*

   Open Engineering détermine les exigences du client à l'avance en établissant un cahier des charges qui explicite les fonctionnalités désirées. Toutefois, ces exigences ne restent pas forcément stables, et des exigences supplémentaires ou des modifications d'exigences peuvent survenir à tout moment, grâce aux réunions organisées régulièrement avec le client.

   Le contexte de l'organisation, par rapport à l'environnement de ce type de projet, présente donc des caractéristiques, à la fois, de type Agile et de type discipliné, de manière équitable. Dès lors, nous évaluerons la valeur de ce critère à **2**.

4. *Relation client*

   Open Engineering établit un contrat avec le client. Ce document ne formalise toutefois pas les solutions mais explicite simplement les fonctionnalités à développer. Une présence régulière du client est donc requise pour lui permettre de fournir des informations utiles au développement.

   Le contexte de l'organisation, par rapport à la relation avec le client de ce type de projet, présente donc des caractéristiques, à la fois, de type Agile et de type discipliné, de manière équitable. Dès lors, nous évaluerons la valeur de ce critère à **2**.

5. *Planification et contrôle*

   Open Engineering planifie le projet en séquences, sur base d'une ligne du temps, où les parties à réaliser sont délimitées. En outre, des réunions mensuelles permettent d'avoir une idée de l'avancement du projet. Cette planification et ce contrôle sont donc communiqués entre les membres de l'équipe et ne nécessitent pas une communication explicite. Toutefois, des rapports de clôture des milestones sont rédigés afin de documenter légèrement les fonctionnalités introduites.

   Le contexte de l'organisation, par rapport à la planification et au contrôle de ce type de projet, présente donc des caractéristiques de type Agile, à quelques exceptions près. Dès lors, nous évaluerons la valeur de ce critère à **1**.

6. *Communication*

   Open Engineering privilégie la communication, notamment pour que les personnes moins expérimentées puissent se référer à celles qui sont plus expérimentées. Des interactions permanentes ont lieu entre les personnes, et des réunions sont souvent organisées afin, notammment, de résoudre les problèmes rencontrés. Toutefois, le développement est légèrement documenté, par les rapports de clôture des milestones et par la documentation statique de l'application développée.

   Le contexte de l'organisation, par rapport à la communication de ce type de projet, présente donc des caractéristiques de type Agile, à quelques exceptions près. Dès lors, nous évaluerons la valeur de ce critère à **1**.

7. *Exigences*

   Open Engineering exprime de manière informelle les fonctionnalités à développer, suivant les priorités du client. Ces exigences sont tout à fait adaptables, grâce aux cycles de développement itératifs courts, qui permettent d'apporter les modifications éventuelles demandées dans l'itération suivante.

   Le contexte de l'organisation, par rapport aux exigences de ce type de projet, présente donc exactement et uniquement des caractéristiques de type Agile. Dès lors, nous évaluerons la valeur de ce critère à **0**.

8. *Développement*

   Open Engineering ne réalise pas de phase d'analyse qui permettrait de prédéfinir formellement le système dans l'optique d'assurer la robustesse de l'architecture de développement. En effet, l'équipe se charge de développer les fonctionnalités de l'itération en cours et n'anticipe pas de nouvelles fonctionnalités.

   Le contexte de l'organisation, par rapport au développement de ce type de projet, présente donc exactement et uniquement des caractéristiques de type Agile. Dès lors, nous évaluerons la valeur de ce critère à **0**.

9. *Tests*

   Open Engineering organise le développement en petits incréments et utilise parfois des techniques, telle que la relecture de code croisée. En outre, Open Engineering développe des tests exécutables qui sont appliqués dès le départ et de manière continue durant le développement, grâce au système de compilation et de test nocturne, qui donne des informations sur l'exécution des tests.

   Le contexte de l'organisation, par rapport aux tests de ce type de projet, présente donc exactement et uniquement des caractéristiques de type Agile. Dès lors, nous évaluerons la valeur de ce critère à **0**.

10. *Développeurs*

    Open Engineering dispose d'une équipe de personnes qualifiées mais dont les expériences varient. En effet, certains possèdent beaucoup d'expérience et connaissent parfaitement l'architecture de l'application, tandis que d'autres sont moins expérimentés.

    Le contexte de l'organisation, par rapport aux développeurs de ce type de projet, présente donc des caractéristiques, à la fois, de type Agile et de type discipliné, de manière équitable. Dès lors, nous évaluerons la valeur de ce critère à **2**.

11. *Culture*

    Open Engineering offre la possibilité aux développeurs de résoudre eux-mêmes les problèmes, disposant d'un environnement ouvert où la collaboration et la spontanéité prévalent. Toutefois, les tâches de chacun sont relativement bien définies et il importe que leur travail soit accompli afin de l'intégrer dans le système.

    Le contexte de l'organisation, par rapport à la culture dans ce type de projet, présente donc des caractéristiques de type Agile, à quelques exceptions près. Dès lors, nous évaluerons la valeur de ce critère à **1**.

L'évaluation de l'ensemble des critères peut désormais être représentée grâce à l'outil graphique (Figure 5.3) qui fournit une vision complète du contexte de l'organisation.

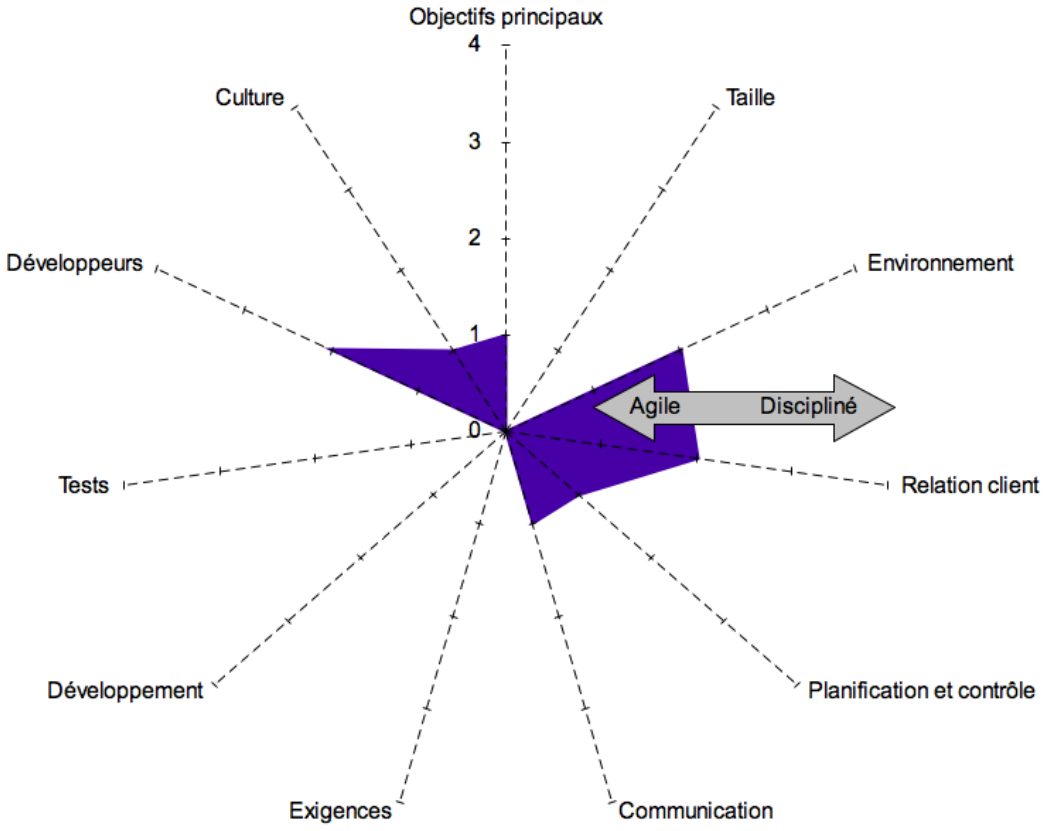

FIG. 5.3 – Graphique d'évaluation des critères - Open Engineering

Forte prépondérance Agile ou disciplinée ?

Le graphique précédant, repris à la figure 5.3, va à présent nous permettre d'identifier la proportion agilité-discipline de l'organisation, afin de savoir quel(s) type(s) de pratiques est (sont) à recommander à Open Engineering, dans le cadre d'une démarche d'amélioration de ses processus logiciels.

Lors de ce premier point de décision, il s'agit de déterminer s'il y a une prépondérance forte, à savoir, si le contexte de Open Engineering présente des caractéristiques qui sont catégoriquement d'un seul type d'approche, pour l'ensemble des critères considérés.

En analysant la figure 5.3, il apparaît clairement qu'aucune des deux approches ne prévaut sur l'autre, pour l'ensemble des critères, et qu'il n'y a, dès lors, aucune prépondérance forte. Nous allons donc considérer l'étape suivante.

Faible prépondérance Agile ou disciplinée ?

Lors de ce second point de décision, il s'agit de déterminer s'il y a une équivalence, à savoir, si le contexte de Open Engineering présente autant de caractéristiques d'un type d'approche que de l'autre, pour l'ensemble des critères considérés.

En analysant la figure 5.3, il apparaît clairement qu'aucune équivalence entre les deux approches ne peut être déterminée, pour l'ensemble des critères, et qu'il y a, dès lors, une prépondérance faible de l'approche Agile.

Les résultats de la phase d'analyse des informations, de la micro-évaluation, vont donc pouvoir être étudiés en fonction de ce contexte, et des recommandations sur base des pratiques Agile vont pouvoir être formulées dans le cadre d'une démarche d'amélioration des processus logiciels de Open Engineering.

Analyse des caractéristiques du type opposé

Dans le cas de Open Engineering, il s'agira lors de cette phase, d'analyser les caractéristiques disciplinées du contexte de l'organisation. La considération de ces caractéristiques permettra alors, durant la phase de rédaction du plan d'action avec les responsables de Open Engineering, de recommander des pratiques Agile qui prendront en compte ces éléments disciplinés du contexte.

Le graphique de la figure 5.3 et l'évaluation de chacun des critères du contexte, nous renseignent sur ces caractéristiques disciplinées :

- *Objectifs principaux* : il y a planification des projets via une ligne du temps définie à priori.
- *Environnement* : un cahier des charges explicite les fonctionnalités désirées et détermine les exigences du client à l'avance.
- *Relation client* : un contrat est établi avec le client.
- *Planification et contrôle* : des rapports de clôture des milestones documentent légèrement les fonctionnalités introduites.

– *Communication* : le développement est légèrement documenté par les rapports de clôture des milestones et par la documentation statique du développement.

– *Développeurs* : l'équipe est composée des personnes moins expérimentées.

– *Culture* : les tâches de chacun sont relativement bien définies.

5.3 Conclusion

Outre l'illustration de l'utilisation de la méthode proposée, cette intervention nous a permis de valider notre formalisation de l'analyse du contexte d'une organisation.

Tout d'abord, nous avons pu confirmer qu'il était possible d'analyser le contexte d'une organisation, uniquement sur base des résultats obtenus par les deux premières phases de la micro-évaluation existante. Ces résultats constituent, en effet, une base d'information suffisante pour évaluer les critères du contexte de l'organisation. Cela permet alors, à la nouvelle démarche de la micro-évaluation, de ne pas alourdir la structure de la démarche OWPL, qui se veut applicable pour les petites structures.

Ensuite, nous avons pu constater que, même si les critères du contexte sont évalués sur base d'un type de projet particulier, ce type de projet est assez représentatif de l'ensemble des projets de développement de l'organisation. Cela signifie qu'il est alors possible de généraliser les conclusions et les recommandations pour l'ensemble de l'organisation. Cependant, dans le cas où plusieurs types de projet différents devaient être constatés dans l'organisation évaluée, une analyse du contexte indépendante pour chacun de ces types de projets serait plutôt indiquée.

Nous avons également pu vérifier que la méthode proposée, pour mesurer la proportion agilité-discipline de chacun des critères du contexte, s'avère adéquate et relativement simple d'utilisation. En effet, la sémantique qui est à considérer par l'évaluateur, et la mise en évidence, pour chacun de ces critères, des différences entre les caractéristiques d'un contexte Agile et d'un contexte discipliné, favorisent cette mesure.

L'outil graphique, proposé par le guide, s'est aussi avéré utile puisqu'il a aisément permis d'évaluer le contexte global de tous les critères, et donc de l'organisation. Cet outil et les différentes étapes du guide ont dès lors effectivement permis de mesurer la proportion agilité-discipline présente dans le contexte de l'organisation.

Enfin, ayant identifié une prépondérance faible pour la recommandation de pratiques Agile dans l'organisation, il est intéressant de constater que la proportion agilité-discipline identifiée nous a également renseigné sur les caractéristiques disciplinées présentes dans l'organisation. Cette connaissance permet, en effet, de recommander des pratiques Agile qui auront le plus de chance de s'intégrer dans le contexte actuel de la PME.

Conclusion et travaux futurs

Conclusion

La démarche OWPL est une démarche d'amélioration des processus logiciels, adaptée aux PME dotées d'un niveau de maturité faible et de ressources limitées, qui permet d'améliorer rapidement et efficacement les pratiques logicielles de l'organisation.

En expérimentant cette démarche auprès de petites entreprises, nous avons pu constater les réelles difficultés auxquelles celles-ci font face lorsqu'elles désirent améliorer leurs processus de développement logiciels. Dans ce contexte, il importe de pouvoir leur recommander des pratiques logicielles qui tiennent compte de leur situation et de leur environnement.

Nous avons également pu constater auprès des PME avec lesquelles nous avons travaillé, que certains concepts ou pratiques Agile étaient appliqués. Cela confirme l'utilisation grandissante des pratiques Agile dans les PME. Dès lors, une adaptation du modèle OWPL, prenant en compte le contexte d'une organisation et les pratiques Agile, s'est avérée nécessaire.

L'adaptation proposée permet donc de guider l'évaluateur dans l'évaluation du contexte de l'organisation, c'est-à-dire, dans l'évaluation de la proportion de caractéristiques Agile ou disciplinées, présentes dans les pratiques de la PME. Cette évaluation lui permettra alors de pouvoir recommander des pratiques de développement plus ou moins disciplinées dans l'optique d'atteindre des processus de développement logiciels de qualité, par rapport au contexte de la PME.

Pour conclure, nous remarquerons que la mesure des critères, formalisée avec la méthodologie présentée, laisse la possibilité à différents évaluateurs d'avoir des mesures légèrement différentes. Cependant, de par le cadre formel de cette méthode, ces différences ne sauraient être réellement majeures. L'analyse du contexte proposée fournit, dès lors, un avantage certain sur une évaluation réalisée de manière informelle, et donc encore plus dépendante de l'évaluateur.

Travaux futurs

Il serait nécessaire de continuer la validation de l'adaptation de la démarche OWPL proposée. Il faudrait donc expérimenter la nouvelle démarche avec un nombre plus important de PME. Cela permettrait d'étudier des contextes différents, en passant par les différents chemins du guide d'analyse, et permettrait dès lors de pouvoir généraliser les conclusions.

Nous pensons qu'il serait également intéressant d'étendre la nouvelle démarche OWPL aux grandes entreprises désireuses d'appliquer une méthodologie moins lourde et moins coûteuse que celle proposée traditionnellement pour l'évaluation et l'amélioration des processus de développement logiciels.

En outre, il serait pertinent, particulièrement dans le cas où la nouvelle démarche serait étendue et appliquée à de grandes entreprises, d'adapter la méthodologie pour permettre de gérer une multitude de types de projets différents au sein d'une même organisation.

Bibliographie

[1] **Abouelfattah M. M., Bamba J-C., Desharnais J-M., Habra N., Laporte C. Y., Renault A.**. *Initiating Software Process Improvement in Small Enterprises : Experiments with Micro-Evaluation Framework*. SWEDC-REK, International Conference on Software Development, Reykjavik, Iceland, University of Iceland, pp.153-164, Décembre 2001.

[2] **Abrahamsson P., et al.**. *Agile software development methods : Review and analysis*. Finland, VTT Publications 478, 2002.

[3] **Ambler S.**. *Agile Modeling : Effective Practices for Extreme Programming and the Unified Process*. New York, John Wiley & Sons, Inc.

[4] **Beck K.**. *Embracing Change With Extreme Programming*. IEEE Computer 32(10) : 70-77, 1999.

[5] **Beck K.**. *Extreme programming explained : Embrace change*. Reading, Mass., Addison-Wesley, 1999.

[6] **Beck K., Beedle M., van Bennekum A., Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R., Mellor S., Schwaber K., Sutherland J., Thomas D.**. *Manifesto for Agile Software Development*. `http://www.AgileManifesto.org`, 2001.

[7] **Beitz A., El-Emam K., Järvinen J.**. *A Business Focus to Assessments*. NRC/ERB-1070, NRC 43615, Décembre 1999.

[8] **Boehm B.**. *Get Ready For The Agile Methods, With Care*. Computer 35(1) : 64-69, 2002.

[9] **Boehm B., Turner R.**. *Balancing Agility and Discipline : A Guide for the Perplexed*. Addison-Wesley, Boston, 2003.

[10] **CETIC**. *Micro-évaluation Agile*. Version 1.1, 2004.

[11] **Cockburn A.**. *Agile Software Development*. Boston, Addison-Wesley, 2002.

[12] **Cohn M.**. *Agile Estimating and Planning*. Prentice Hall, Novembre 2005.

[13] **Desharnais J-M., Habra N., Laporte C. Y., Renault A., Stambollian A.**. *OWPL : A Light Model & Methodology for Initiating Software Process Improvement*. Proceedings of SPICE Conference, May 4-5, 2006, Luxembourg 2006.

[14] **Habra N.**. *Ingénierie du logiciel*. INFO2204, F.U.N.D.P., Institut d'Informatique Université de Namur, Année académique 2005-2006.

[15] **Habra N., Renault A.**. *Modèle OWPL. Evaluation et amélioration des pratiques logicielles dans les PME wallonnes*. Version 1.2.2 b, F.U.N.D.P., Institut d'Informatique Université de Namur, Janvier 2001.

[16] **Habra N., Renault A.**. *OWPL. Une Méthodologie et des Modèles Légers pour Initier une Démarche d'Amélioration des Pratiques Logicielles APL*. Revue ISI « Ingénierie des Systèmes d'Information », numéro spécial « Qualité des Systèmes d'information », vol. 9, Hermes, 2004.

[17] **Highsmith J. A.**. *Adaptive Software Development : A Collaborative Approach to Managing Complex Systems*. New York, NY, Dorset House Publishing, 2000.

[18] **Jeffries R., Anderson A., Hendrickson C.**. *Extreme Programming Installed*. Upper Saddle River, NJ, Addison-Wesley, 2001.

[19] **Kruchten P.**. *The Rational Unified Process : an Introduction*. Addison-Wesley, 2000.

[20] **McCabe R., Polen M.**. *Should You Be More Agile ?* STSC CrossTalk, The Journal of Defense Software Engineering, Octobre 2002.

[21] **Nicolette D.**. *When to be Agile*. The IT Metrics and Productivity Journal, Décembre 2005.

[22] **Palmer S. R., Felsing J. M.**. *A Practical Guide to Feature-Driven Development*. Upper Saddle River, NJ, Prentice-Hall, 2002.

[23] **Rising L., Janoff N. S.**. *The Scrum software development process for small teams*. IEEE Software 17(4) : 26-32, 2000.

[24] **Schwaber K., Beedle M.**. *Agile Software Development With Scrum*. Upper Saddle River, NJ, Prentice-Hall, 2002.

[25] **Software Engineering Institute**. *Capability Maturity Model ® Integration (CMMI)*. Version 1.1, Carnegie Melon University, 2002.

[26] **Stambollian A.**. *Projet de synthèse de la maîtrise en génie logiciel sur l'amélioration de la performance des processus et pratiques logiciels dans des petites entreprises françaises*. Université de Québec : Ecole de Technologie Supérieure, Montréal, Québec, Canada, Avril 2006.

[27] **Stapleton J.**. *Dynamic systems development method - The method in practice*. Addison-Wesley, 1997.

[28] **Szalvay V.**. *An Introduction to Agile Software Development*. Danube Technologies, Inc., Novembre 2004.

Troisième partie

Annexes

Annexe A

Plan d'action - Logilys

Département Génie Logiciel

Logilys - Plan d'action

Antoine Robaeys
antoine.robaeys@student.fundp.ac.be

27 Septembre 2006

École de Technologie Supérieure - 1100 rue Notre-Dame Ouest - Montréal
Tél. : (514) 396-8800 - Fax : (514) 396-8950

Table des matières

Introduction

Ce document consiste en une synthèse détaillée du plan d'action élaboré pour Logilys. Ce rapport se base principalement sur la micro-évaluation réalisée le 20 juin 2006 et permet de faire ressortir les recommandations tirées de cette évaluation.

Selon notre perception des problèmes nous suggérons l'ordre de priorité suivant :

1. **Développer le bon logiciel** : Les exigences ne sont pas bien définies et les clients ne sont pas impliqués dans le processus de développement.

2. **Gérer les requêtes de changement** : Les requêtes de changement ne sont pas bien définies.

3. **Adopter une approche de qualité** : Les activités de qualité sont limitées aux tests et à la formation des futurs développeurs. En outre, aucune méthodologie de développement n'a été adoptée et chaque personne est libre d'effectuer la documentation qu'elle juge nécessaire en ce qui concerne les documents d'analyse.

4. **Délivrer le logiciel à temps** : La planification de projet n'est pas appliquée pour tous les projets (juste pour les plus importants) et il n'y a aucune façon de vérifier la progression du projet.

5. **Gérer la configuration** : Aucune gestion des versions de la documentation ou du code n'est formalisée à l'aide d'un outil prévu à cet effet.

6. **Gérer les relations avec les sous-traitants** : La sélection des sous-traitants ne passe pas par un processus décisionnel formel et le suivi complet des sous-traitants est rarement réalisé.

Nous nous chargerons de solutionner et d'implanter, pour la fin novembre, un problème que nous aurons choisi par consensus avec l'équipe de Logilys en prenant en compte les trois critères de choix suivants :

- Le court terme.
- La réalisabilité pour l'entreprise en terme de coûts, de ressources disponibles, etc.
- L'utilité pour l'entreprise.

Chapitre 1

Développer le bon logiciel

En ingénierie des exigences, le processus d'analyse des objectifs du client peut être représenté comme suit :

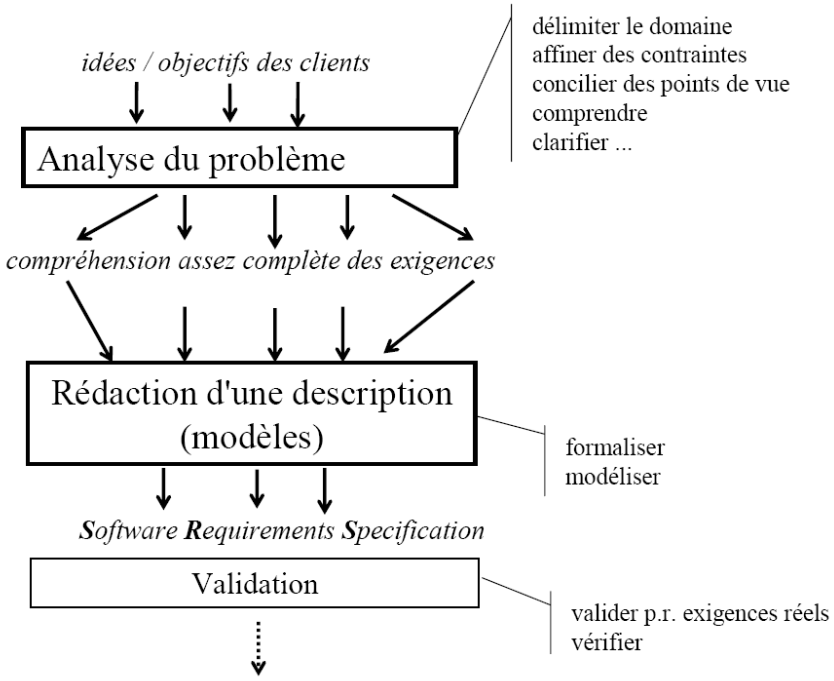

FIG. 1.1 – Processus d'analyse

Afin de garantir le développement du bon logiciel, à savoir un logiciel qui répond aux exigences du client, il est impératif de pouvoir formaliser au mieux ces exigences. Il s'agit donc de déterminer les *services* qu'un système doit offrir et les *contraintes* sous lesquelles il doit fonctionner.

1.1 Software Requirements Specification

Le document SRS (Software Requirements Specification) se charge de spécifier les exigences. Il fournit des bénéfices tels que :

- Établir un accord entre les clients et les développeurs.
- Réduire le temps de développement de l'effort.
- Fournir une base pour estimer les coûts et les échéanciers.
- Fournir une base pour la validation et la vérification.

Volere est un modèle de document SRS qui peut être adapté selon les besoins. Volere est le résultat de plusieurs années de pratique, de consultance et de recherche dans l'ingénierie des exigences. Ce modèle fournit des sections pour chaque type d'exigences appropriées aux systèmes logiciels actuels permettant ainsi de structurer le cahier des charges.

Les chapitres du plan de cahier des charges sont les suivants :

1.1.1 FONDEMENTS DU PROJET

But du projet

- Problème de l'utilisateur ou contexte du projet
- Objectifs du projet

Personnes et organismes impliqués dans les enjeux du projet

- Client
- Acheteur
- Autre parties prenantes

Utilisateurs du produit

- Utilisateurs directs du produit
- Priorité assignée aux utilisateurs
- Implication nécessaire de la part des utilisateurs dans le projet
- Utilisateurs concernés par les opérations de maintenance du produit

1.1.2 CONTRAINTES SUR LE PROJET

Contraintes imposées (non négociables)

- Contraintes sur la conception de la solution
- Environnement de fonctionnement du système actuel
- Applications « partenaires » (avec lesquelles le produit doit collaborer)
- Progiciels ou composants commerciaux
- Lieux de fonctionnement prévus
- De combien de temps les développeurs disposent-ils pour le projet ?
- Quel est le budget affecté au projet ?

Glossaire et conventions de dénomination

Faits et hypothèses utiles

- Facteurs influençant le produit, mais qui ne sont pas des contraintes imposées sur les exigences
- Hypothèses que l'équipe fait sur le projet

1.1.3 EXIGENCES FONCTIONNELLES

Portée du travail

- La situation actuelle
- Contexte du travail
- Division du travail en événements métier

Portée du produit : cas d'utilisation

- Limites du produit : diagramme de cas d'utilisation
- Description sommaire des cas d'utilisation

Exigences fonctionnelles et exigences sur les données

- Exigences fonctionnelles
- Exigences sur les données

1.1.4 EXIGENCES NON FONCTIONNELLES

Ergonomie et convivialité du produit

- L'interface
- Le style du produit

Facilité d'utilisation et facteurs humains

- Facilité d'utilisation
- Personnalisation et internationalisation
- Facilité d'apprentissage
- Facilité de compréhension et politesse
- Exigences d'accessibilité

Performance du produit

- Rapidité d'exécution et temps de latence
- Exigences critiques de sûreté
- Précision et exactitude
- Fiabilité et disponibilité
- Robustesse ou tolérance à un emploi erroné
- Capacité de stockage et montée en charge
- Adaptation du produit à une augmentation de volume à traiter
- Longévité

Exigences opérationnelles et environnementales

- Environnement physique prévu
- Environnement technologique prévu
- Applications « partenaires » (avec lesquelles le produit doit collaborer)

– Approche « produit » prêt à être commercialisé

Maintenance, support, portabilité, installation du produit

– Maintenance du produit
– Conditions spéciales concernant la maintenance du produit
– Exigences en matière de support
– Exigences de portabilité
– Installation du système

Sécurité

– Accès au système
– Intégrité
– Protection des données à caractère personnel
– Audit et traçabilité
– Protection contre les infections

Exigences culturelles et politiques

– Exigences culturelles
– Exigence politiques

Lois et standards influençant le produit

– Conformité avec la loi
– Conformité avec des standards

1.1.5 AUTRES ASPECTS DU PROJET

Questions sans réponse

Progiciels et composants commerciaux

Nouveaux problèmes, créés par le nouveau système

Tâches à faire pour livrer le système

Contrôle final de qualité sur site

Risques liés au projet

Estimation des coûts du projet

Manuel utilisateur et formations à envisager

Salle d'attente : idées pour les futures versions

Idées de solutions

Chapitre 2

Gérer les requêtes de changement

Les deux sources majeures de changement dans un système sont principalement :

- Le besoin de régler une déficience dans le système.
- Des changements dans les besoins organisationnels.

Afin de pouvoir gérer ces changements, le processus d'analyse des objectifs du client peut être adapté comme suit :

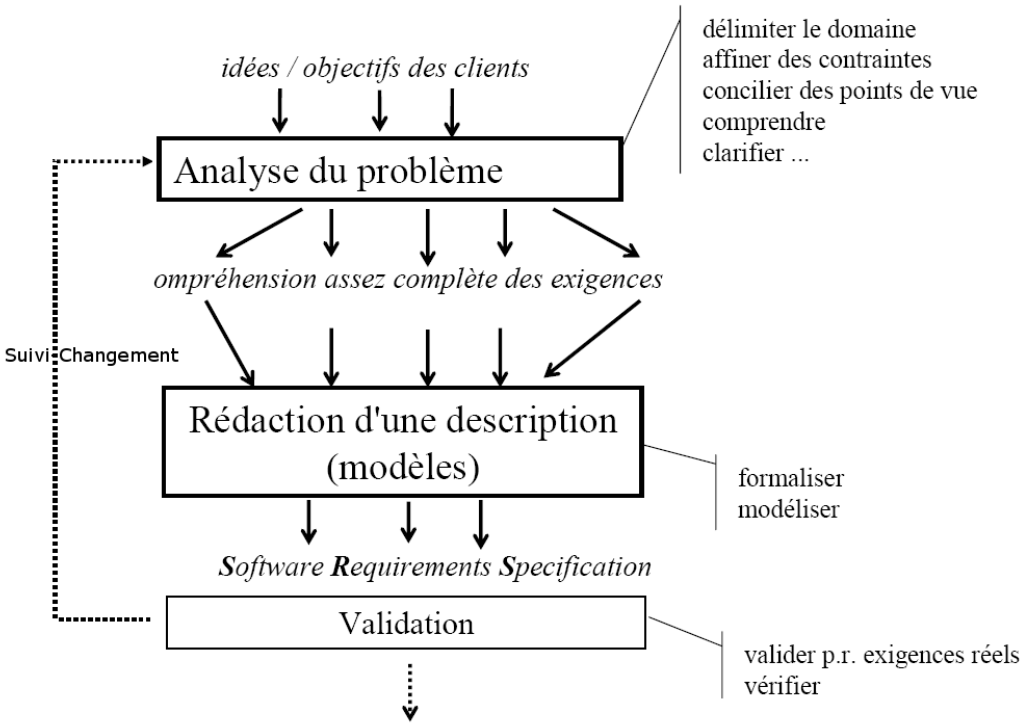

FIG. 2.1 – Processus d'analyse

2.1 La traçabilité des exigences

La traçabilité des exigences est la capacité de décrire, de suivre et de comprendre l'évolution d'une exigence. Cela permet de :

- Relier les exigences, les sources (Documents, Cas d'utilisation,...) et les justifications.
- Aider les concepteurs à garder trace des changements et à pouvoir évaluer l'impact d'une exigence sur le système avant son implémentation.
- Faire la correspondance entre les tests et les exigences afin d'assurer une pleine couverture des objectifs des clients.

Le modèle de spécification des exigences, Volere, propose d'utiliser cette structure comme un guide pour écrire chacune des exigences :

<table>
<tr><td>Numéro de l'exigence : identifiant unique</td><td>Type d'exigence : référence au modèle</td><td>Evénements/Cas d'utilisation : Liste d'événements/de use cases qui nécessitent cette exigence</td></tr>
<tr><td colspan="3">Description : objectif de l'exigence, en une phrase, généralement sous la forme « le produit devrait faire ceci pour telle personne »</td></tr>
<tr><td colspan="3">Justification : raison pour laquelle cette exigence est présente</td></tr>
<tr><td colspan="3">Origine : qui a émis cette exigence ?</td></tr>
<tr><td colspan="3">Critère de satisfaction : une mesure de l'exigence qui permet de tester si la solution proposée remplit l'exigence initiale.</td></tr>
<tr><td colspan="3">Numéros des tests vérifiant l'exigence : Liste de tests assurant que le logiciel vérifie l'exigence courante.</td></tr>
<tr><td colspan="2">Contentement de l'acheteur : degré de contentement de l'acheteur si le produit final satisfait cette exigence. De 1 (pas intéressé) à 5 (très content).</td><td>Mécontentement du maître d'ouvrage : degré de mécontentement de l'acheteur si le produit final ne satisfait pas cette exigence. De 1 (quasi indifférent) à 5 (très mécontent)</td></tr>
<tr><td colspan="2">Exigences dépendantes : liste d'exigences dont l'implémentation dépend de l'implémentation de celle-ci.</td><td>Exigences conflictuelles : exigences qui ne peuvent pas être implémentées si celle-ci l'est.</td></tr>
<tr><td colspan="3">Priorité : une estimation de la valeur de l'acheteur.</td></tr>
<tr><td colspan="3">Documents relatifs : référence à des documents qui illustrent et expliquent cette exigence.</td></tr>
<tr><td colspan="3">Historique : création, modifications, destruction apportées à l'exigence.</td></tr>
</table>

Fig. 2.2 – Structure Volere

Chapitre 3

Adopter une approche de qualité

Outre les tests et les formations des futurs développeurs, une amélioration de la qualité du processus logiciel chez Logilys serait de tester la qualité des exigences dès que l'on commence à les écrire. Pour ce faire, la structure Volere (figure 2.2) inclut un critère de satisfaction. Ce critère permet de spécifier une exigence de manière claire, signifiant que l'exigence n'est pas ambiguë ou mal comprise. S'il n'y a aucun critère de satisfaction, il n'y a aucune façon de savoir si une solution correspond à l'exigence.

Il s'agit donc d'évaluer quantitativement l'exigence en spécifiant son critère de satisfaction. Ce dernier représente une mesure objective de la signification de l'exigence et c'est ce critère qui va permettre d'évaluer si une solution donnée satisfait vraiment l'exigence.

Afin d'assurer une pleine couverture des objectifs des clients, à savoir, assurer que le logiciel vérifie bel et bien toutes les exigences, il nous serait possible de faire la correspondance entre les tests et les exigences en associant à une exigence une liste de tests. Bien que cette possibilité ne soit pas inclue dans la structure du modèle de Volere, il nous paraîtrait intéressant de l'y incorporer.

3.1 Rational Unified Process

Dans l'optique de gérer et contrôler la qualité, il serait intéressant d'utiliser la méthodologie RUP afin d'améliorer la qualité du processus de développement.

RUP est une approche itérative pour le développement OO de systèmes et possède trois caractéristiques :

1. **Piloté par les cas d'utilisation**, et ce de la phase d'inception à la phase de déploiement.

2. **Centré sur l'architecture** : Le processus cherche à comprendre les aspects statiques et dynamiques les plus significatifs du système. L'architecture permet de réaliser les besoins exprimés par les utilisateurs à travers les cas d'utilisation en tenant compte d'autres facteurs tels que la plate-forme d'exécution, les interfaces utilisateurs, les besoins non fonctionnels.

3. **Itératif et incrémental** : Les itérations désignent des étapes d'enchaînement d'activité et les incréments correspondent à des stades de développement du produit. Chaque itération est considérée comme un mini-projet qui donne lieu à un incrément. Les itérations se succèdent dans un ordre logique pour prendre en compte les cas d'utilisation et traiter en priorité les risques majeurs et les problèmes imprévus.

Remarquons que ces trois notions sont parfaitement complémentaires :

- Les cas d'utilisation doivent trouver leur place dans l'architecture.
- L'architecture doit prévoir la réalisation de tous les cas d'utilisation présents et à venir.
- L'architecture et les cas d'utilisation évoluent donc conjointement.
- L'architecture fournit la structure qui servira de cadre au travail effectué au cours des itérations tandis que les cas d'utilisation définissent les objectifs et orientent le travail de chaque itération.

En outre, cette démarche présente plusieurs avantages :

1. **Risque financier limité** : S'il faut reprendre une itération, la valeur du système n'est pas engagée dans son entier.
2. **Risque de retard limité** : Les risques, identifiés et résolus dès les premiers stades de développement, ne remettent pas en cause le travail déjà effectué.
3. **Accélération du rythme de développement** : Travail plus efficace vers des objectifs clairs à court terme.
4. **Adaptation à l'évolution des besoins** : Les besoins des utilisateurs ne peuvent être intégralement définis à l'avance, ils se dégagent peu à peu des itérations successives.

3.2 Les quatre étapes du processus

Le processus RUP consiste en des cycles qui peuvent se répéter tout au long de la vie d'un système. Un cycle contient les quatre étapes suivantes :

1. **Inception** : Objectifs du projet - Besoins des stakeholders - Estimations des planning et coûts - Frontière du projet.
2. **Elaboration** : Analyse du problème - Plan du projet - Description architecturale et détaillée - Analyse de risque et de stabilité architecturale.
3. **Construction** : Développement - Tests - Gestion des ressources, coûts et planning - Vérification de la qualité.
4. **Transition** : Distribution aux utilisateurs - Formations - Documentation.

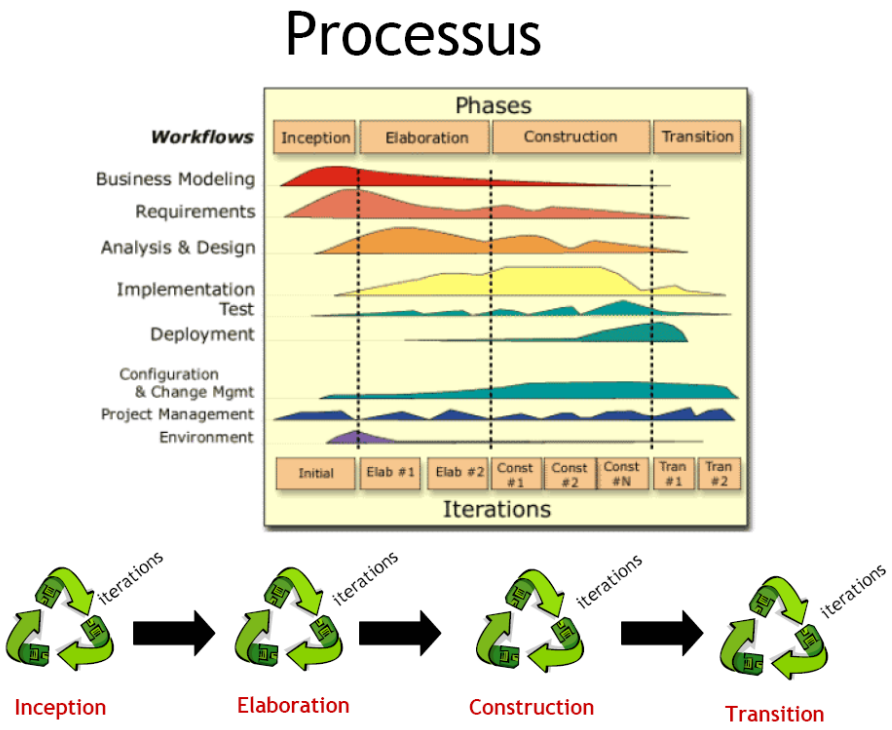

Fig. 3.1 – Rational Unified Process

Chapitre 4

Délivrer le logiciel à temps

L'utilisation, pour chacun des projets, de techniques de mesure de taille, tel que COSMIC-FFP, et d'effort, pour l'estimation d'un projet permettrait de connaître et de suivre :

- *La taille*
- *L'effort*
- *La durée*
- *Les coûts*

Ces connaissances et suivis seraient plus qu'utiles dans l'optique de pouvoir délivrer un produit en temps et en heure à un client.

COSMIC-FFP est une méthode, internationalement reconnue et des plus moderne, de mesure de la taille fonctionnelle d'un logiciel. Elle fut développée par COSMIC, the Common Software Measurement International Consortium, un consortium regroupant les meilleurs experts dans le domaine de la mesure logicielle.

Les mesures de taille fonctionnelle sont basées sur les exigences fonctionnelles des utilisateurs du logiciel. De telles mesures de taille sont dès lors valables dès les premières étapes de l'estimation du projet logiciel ainsi que pour comparer les mesures de performance de projets (e.g. Productivité = Taille fonctionnelle / Effort) au travers de projets logiciels utilisant des technologies différentes.

COSMIC-FFP est la seule méthode moderne de mesure de taille fonctionnelle conçue à la fois pour les systèmes informatiques de gestion et pour les systèmes à temps réel.

4.1 Les étapes de l'estimation

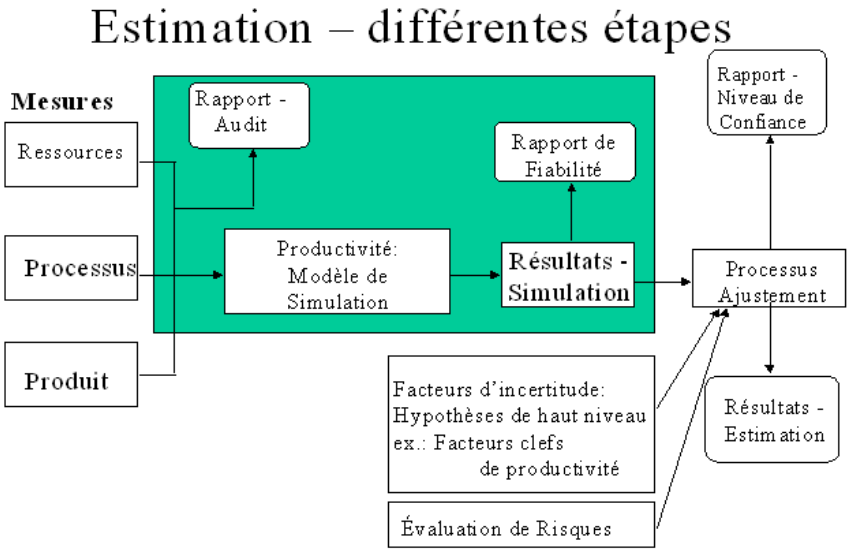

FIG. 4.1 – Processus de l'estimation

4.1.1 Les mesures

On distingue trois classes de mesures :

- Ressources : L'ensemble des objets qui contribuent au processus.
- Processus : Les activités liées au développement logiciel.
- Produits : Les livrables qui sont le résultat du processus.

4.1.2 Le rapport d'audit

Le rapport d'audit devrait faire ressortir les problèmes potentiels liés à la précision et l'exhaustivité de la mesure, que ce soit une mesure de ressource, de processus ou de produit.

4.1.3 Le modèle de simulation (modèle de productivité)

Les modèles de qualité, de budgétisation, de productivité et d'estimation. Par exemple, un modèle d'estimation fait état des efforts sur la taille alors qu'un modèle de qualité fait état des anomalies sur la taille.

4.1.4 Le rapport de fiabilité

Par exemple, le rapport de fiabilité du modèle de simulation de la qualité doit fournir des preuves documentées de degré de fiabilité de chaque modèle (ou gabarit utilisé), et ceci doit être documenté dans le modèle initial, pour le processus de calibrage et le processus de validation. Par exemple, il devrait fournir des informations sur le périmètre de sa fiabilité et son étendue à l'extérieur de ce périmètre.

4.1.5 Les facteurs d'incertitude, de risque et les hypothèses de haut niveau

Tout projet a ses facteurs d'incertitudes (technologie utilisée, qualité du personnel), ses risques (liés à la technologie, les facteurs organisationnels et la taille du produit) et ses considérations de haut niveau (Exemple : on ne peut demander plus qu'un certain prix et la livraison doit se faire avant une certaine date). Il y a donc un processus d'ajustement plus ou moins défini que l'on retrouve dans certaines approches ou techniques relevant plus d'un art que de la science

4.1.6 Le processus d'ajustement

Ce processus est subjectif et basé sur l'expérience des individus qui font ces ajustements.

4.1.7 Le rapport du niveau de confiance

Le rapport de confiance doit essayer de quantifier le degré de confiance que l'on peut avoir dans les résultats de l'estimation. Le degré de confiance sera, entre autre, dépendant de la façon dont sont présentés les résultats.

Le rapport du niveau de confiance doit fournir tous les ajustements faits aux résultats provenant du modèle de simulation en plus de fournir une justification pour ces ajustements. Ces ajustements sont réalisés le plus souvent sur la base du jugement des praticiens et des gestionnaires, dans le contexte de leur expérience passée et actuelle de livraison de projets ainsi que leur engagement actuel.

4.1.8 Les résultats de l'estimation

Les résultats d'une estimation doivent aussi tenir compte de l'évolution des projets dans le temps, ce qui entraîne un certain nombre de préoccupations telles que, l'évolution de l'envergure du projet en cours de réalisation, l'échéance des projets et l'évolution des estimés en cours de réalisation.

Chapitre 5

Gérer la configuration

L'amélioration de la gestion de configuration passe par une formalisation de la production des documents et des codes sources. Cette formalisation instaurerait ainsi un certain gabarit pour les différents types de documents.

En outre, l'implantation d'un outil de gestion de versionnement, tel que subversion (parfois abrégé SVN), et une formation éventuelle des employés à cet outil permettrait de conserver les versions de l'ensemble des documents et code source, formant ainsi l'historique des changements apportés à ces derniers.

Chapitre 6

Gérer les relations avec les sous-traitants

Etant parfois dans l'obligation de sous-traiter, suite au manque de travail dans certains domaines comme le développement Web, il serait intéressant de remplacer les ententes verbales avec les sous-traitants par un contrat spécifique avec ceux-ci.

Ce contrat spécifierait clairement les responsabilités du sous-traitant et permettrait de procéder à un suivi régulier de la progression du travail en cours.

Bibliographie

[1] **Volere**, *Requirements Specification Template*, Edition 11, Atlantic Systems Guild.

[2] **Projet SPINOV WP5.3**, *Plan de cahier des charges et spécification des exigences non fonctionnelles avec Volere*, CRP Henri Tudor - CITI, The Atlantic Systems Guild Inc.

[3] **Traçabilité**, *Toward Reference Models for Requirements Traceability*, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 27, NO. 1, JANUARY 2001.

[4] **Processus de l'estimation**, *Modèle d'estimation*, Jean-Marc Desharnais, Alain Abran.

[5] **Génie logiciel**, *Ingenierie des logiciels*, Naji Habra, F.U.N.D.P.

[6] **Génie logiciel**, *Ingenierie des exigences*, Patrick Heymans, F.U.N.D.P.

Annexe B

Plan d'action - Quartier général de l'URSC-E

Université du Québec
École de technologie supérieure

Département Génie Logiciel

Le Quartier général de l'Unité Régionale de Soutien des Cadets de la Région de l'Est - Plan d'action

Antoine Robaeys
antoine.robaeys@student.fundp.ac.be

4 Octobre 2006

École de Technologie Supérieure - 1100 rue Notre-Dame Ouest - Montréal
Tél. : (514) 396-8800 - Fax : (514) 396-8950

Table des matières

Introduction

Ce document consiste en une synthèse détaillée du plan d'action élaboré pour le Quartier général de l'Unité Régionale de Soutien des Cadets de la Région de l'Est. Ce rapport se base principalement sur la micro-évaluation réalisée le 01 juin 2006 et permet de faire ressortir les recommandations tirées de cette évaluation.

Selon notre perception des problèmes nous suggérons l'ordre de priorité suivant :

1. **Estimation de projet** : Aucune connaissance ni suivi de l'effort, de la durée et des coûts d'un projet.

2. **Développer le bon logiciel** : Aucun document formel pour la spécification des exigences (SRS) et aucune traçabilité entre les tests et ces exigences.

3. **Fournir de la bonne documentation** : Le processus de documentation du code est laissé à la seule discrétion des développeurs.

4. **Assurance qualité** : Aucune méthodologie pour le contrôle de la qualité ni aucune pratique de gestion de la qualité qui permet d'éviter de dépendre des individualités.

Nous nous chargerons de solutionner et d'implanter, pour la fin novembre, un problème que nous aurons choisi par consensus avec l'équipe de l'Unité Régionale en prenant en compte les trois critères de choix suivants :

- Le court terme.
- La réalisabilité pour l'entreprise en terme de coûts, de ressources disponibles, etc.
- L'utilité pour l'entreprise.

Chapitre 1

Estimation de projet

L'utilisation, pour chacun des projets, de techniques de mesure de taille, tel que COSMIC-FFP, et d'effort, pour l'estimation d'un projet permettrait de connaître et de suivre :

- *La taille*
- *L'effort*
- *La durée*
- *Les coûts*

COSMIC-FFP est une méthode, internationalement reconnue et des plus moderne, de mesure de la taille fonctionnelle d'un logiciel. Elle fut développée par COSMIC, the Common Software Measurement International Consortium, un consortium regroupant les meilleurs experts dans le domaine de la mesure logicielle.

Les mesures de taille fonctionnelle sont basées sur les exigences fonctionnelles des utilisateurs du logiciel, à savoir, la quantité de fonctionnalités requises par ces utilisateurs. De telles mesures de taille sont dès lors valables dès les premières étapes de l'estimation du projet logiciel ainsi que pour comparer les mesures de performance de projets (e.g. Productivité = Taille fonctionnelle / Effort) au travers de projets logiciels utilisant des technologies différentes.

COSMIC-FFP est la seule méthode moderne de mesure de taille fonctionnelle conçue à la fois pour les systèmes d'information de gestion et pour les systèmes embarqués ou à temps réel.

1.1 Les étapes de l'estimation

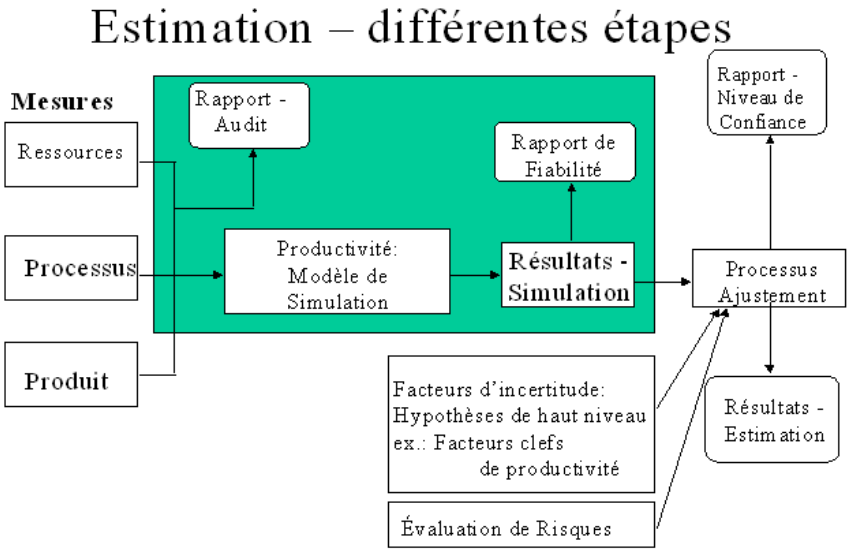

FIG. 1.1 – Processus de l'estimation

1.1.1 Les mesures

On distingue trois classes de mesures :

- Ressources : L'ensemble des objets qui contribuent au processus.
- Processus : Les activités liées au développement logiciel.
- Produits : Les livrables qui sont le résultat du processus.

1.1.2 Le rapport d'audit

Le rapport d'audit devrait faire ressortir les problèmes potentiels liés à la précision et l'exhaustivité de la mesure, que ce soit une mesure de ressource, de processus ou de produit.

1.1.3 Le modèle de simulation (modèle de productivité)

Les modèles de qualité, de budgétisation, de productivité et d'estimation. Par exemple, un modèle d'estimation fait état des efforts sur la taille alors qu'un modèle de qualité fait état des anomalies sur la taille.

1.1.4 Le rapport de fiabilité

Par exemple, le rapport de fiabilité du modèle de simulation de la qualité doit fournir des preuves documentées de degré de fiabilité de chaque modèle (ou gabarit utilisé), et ceci doit être documenté dans le modèle initial, pour le processus de calibrage et le processus de validation. Par exemple, il devrait fournir des informations sur le périmètre de sa fiabilité et son étendue à l'extérieur de ce périmètre.

1.1.5 Les facteurs d'incertitude, les hypothèses de haut niveau et les facteurs de risque

Tout projet a ses facteurs d'incertitudes (technologie utilisée, qualité du personnel), ses risques (liés à la technologie, les facteurs organisationnels et la taille du produit) et ses considérations de haut niveau (Exemple : on ne peut demander plus qu'un certain prix et la livraison doit se faire avant une certaine date). Il y a donc un processus d'ajustement plus ou moins défini que l'on retrouve dans certaines approches ou techniques relevant plus d'un art que de la science

1.1.6 Le processus d'ajustement

Ce processus est subjectif et basé sur l'expérience des individus qui font ces ajustements.

1.1.7 Le rapport du niveau de confiance

Le rapport de confiance doit essayer de quantifier le degré de confiance que l'on peut avoir dans les résultats de l'estimation. Le degré de confiance sera, entre autre, dépendant de la façon dont sont présentés les résultats.

Le rapport du niveau de confiance doit fournir tous les ajustements faits aux résultats provenant du modèle de simulation en plus de fournir une justification pour ces ajustements. Ces ajustements sont réalisés le plus souvent sur la base du jugement des praticiens et des gestionnaires, dans le contexte de leur expérience passée et actuelle de livraison de projets ainsi que leur engagement actuel.

1.1.8 Les résultats de l'estimation

Les résultats d'une estimation doivent aussi tenir compte de l'évolution des projets dans le temps, ce qui entraîne un certain nombre de préoccupations telles que, l'évolution de l'envergure du projet en cours de réalisation, l'échéance des projets et l'évolution des estimés en cours de réalisation.

Chapitre 2

Développer le bon logiciel

En ingénierie des exigences, le processus d'analyse des objectifs du client peut être représenté comme suit :

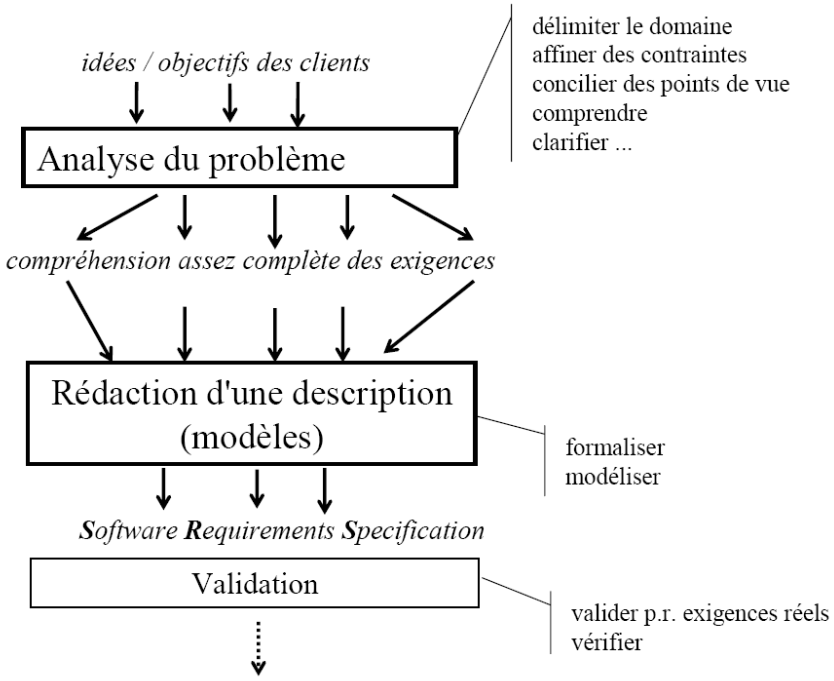

FIG. 2.1 – Processus d'analyse

Afin de garantir le développement du bon logiciel, à savoir un logiciel qui répond aux exigences du client, il est impératif de pouvoir formaliser au mieux ces exigences. Il s'agit donc de déterminer les *services* qu'un système doit offrir et les *contraintes* sous lesquelles il doit fonctionner.

2.1 Software Requirements Specification

Le document SRS (Software Requirements Specification) se charge de spécifier les exigences. Il fournit des bénéfices tels que :

- Établir un accord entre les clients et les développeurs.
- Réduire le temps de développement de l'effort.
- Fournir une base pour estimer les coûts et les échéanciers.
- Fournir une base pour la validation et la vérification.

Volere est un modèle de document SRS qui peut être adapté selon les besoins. Volere est le résultat de plusieurs années de pratique, de consultance et de recherche dans l'ingénierie des exigences. Ce modèle fournit des sections pour chaque type d'exigences appropriées aux systèmes logiciels actuels permettant ainsi de structurer le cahier des charges.

Les chapitres du plan de cahier des charges sont les suivants :

2.1.1 FONDEMENTS DU PROJET

But du projet

- Problème de l'utilisateur ou contexte du projet
- Objectifs du projet

Personnes et organismes impliqués dans les enjeux du projet

- Client
- Acheteur
- Autre parties prenantes

Utilisateurs du produit

- Utilisateurs directs du produit
- Priorité assignée aux utilisateurs
- Implication nécessaire de la part des utilisateurs dans le projet
- Utilisateurs concernés par les opérations de maintenance du produit

2.1.2 CONTRAINTES SUR LE PROJET

Contraintes imposées (non négociables)

- Contraintes sur la conception de la solution
- Environnement de fonctionnement du système actuel
- Applications « partenaires » (avec lesquelles le produit doit collaborer)
- Progiciels ou composants commerciaux
- Lieux de fonctionnement prévus
- De combien de temps les développeurs disposent-ils pour le projet ?
- Quel est le budget affecté au projet ?

Glossaire et conventions de dénomination

Faits et hypothèses utiles

- Facteurs influençant le produit, mais qui ne sont pas des contraintes imposées sur les exigences
- Hypothèses que l'équipe fait sur le projet

2.1.3 EXIGENCES FONCTIONNELLES

Portée du travail

- La situation actuelle
- Contexte du travail
- Division du travail en événements métier

Portée du produit : cas d'utilisation

- Limites du produit : diagramme de cas d'utilisation
- Description sommaire des cas d'utilisation

Exigences fonctionnelles et exigences sur les données

- Exigences fonctionnelles
- Exigences sur les données

2.1.4 EXIGENCES NON FONCTIONNELLES

Ergonomie et convivialité du produit

- L'interface
- Le style du produit

Facilité d'utilisation et facteurs humains

- Facilité d'utilisation
- Personnalisation et internationalisation
- Facilité d'apprentissage
- Facilité de compréhension et politesse
- Exigences d'accessibilité

Performance du produit

- Rapidité d'exécution et temps de latence
- Exigences critiques de sûreté
- Précision et exactitude
- Fiabilité et disponibilité
- Robustesse ou tolérance à un emploi erroné
- Capacité de stockage et montée en charge
- Adaptation du produit à une augmentation de volume à traiter
- Longévité

Exigences opérationnelles et environnementales

- Environnement physique prévu
- Environnement technologique prévu
- Applications « partenaires » (avec lesquelles le produit doit collaborer)

– Approche « produit » prêt à être commercialisé

Maintenance, support, portabilité, installation du produit

– Maintenance du produit
– Conditions spéciales concernant la maintenance du produit
– Exigences en matière de support
– Exigences de portabilité
– Installation du système

Sécurité

– Accès au système
– Intégrité
– Protection des données à caractère personnel
– Audit et traçabilité
– Protection contre les infections

Exigences culturelles et politiques

– Exigences culturelles
– Exigence politiques

Lois et standards influençant le produit

– Conformité avec la loi
– Conformité avec des standards

2.1.5 AUTRES ASPECTS DU PROJET

Questions sans réponse

Progiciels et composants commerciaux

Nouveaux problèmes, créés par le nouveau système

Tâches à faire pour livrer le système

Contrôle final de qualité sur site

Risques liés au projet

Estimation des coûts du projet

Manuel utilisateur et formations à envisager

Salle d'attente : idées pour les futures versions

Idées de solutions

2.2 La traçabilité des exigences

La traçabilité des exigences est la capacité de décrire, de suivre et de comprendre l'évolution d'une exigence. Cela permet de :

- Relier les exigences, les sources (Documents, Cas d'utilisation,...) et les justifications.
- Aider les concepteurs à garder trace des changements et à pouvoir évaluer l'impact d'une exigence sur le système avant son implémentation.
- Faire la correspondance entre les tests et les exigences afin d'assurer une pleine couverture des objectifs des clients.

Le modèle de spécification des exigences, Volere, propose d'utiliser cette structure comme un guide pour écrire chacune des exigences :

<table>
<tr><td>Numéro de l'exigence : identifiant unique</td><td>Type d'exigence : référence au modèle</td><td>Evénements/Cas d'utilisation : Liste d'événements/de use cases qui nécessitent cette exigence</td></tr>
<tr><td colspan="3">Description : objectif de l'exigence, en une phrase, généralement sous la forme « le produit devrait faire ceci pour telle personne »</td></tr>
<tr><td colspan="3">Justification : raison pour laquelle cette exigence est présente</td></tr>
<tr><td colspan="3">Origine : qui a émis cette exigence ?</td></tr>
<tr><td colspan="3">Critère de satisfaction : une mesure de l'exigence qui permet de tester si la solution proposée remplit l'exigence initiale.</td></tr>
<tr><td colspan="3">Numéros des tests vérifiant l'exigence : Liste de tests assurant que le logiciel vérifie l'exigence courante.</td></tr>
<tr><td colspan="2">Contentement de l'acheteur : degré de contentement de l'acheteur si le produit final satisfait cette exigence. De 1 (pas intéressé) à 5 (très content).</td><td>Mécontentement du maître d'ouvrage : degré de mécontentement de l'acheteur si le produit final ne satisfait pas cette exigence. De 1 (quasi indifférent) à 5 (très mécontent)</td></tr>
<tr><td colspan="2">Exigences dépendantes : liste d'exigences dont l'implémentation dépend de l'implémentation de celle-ci.</td><td>Exigences conflictuelles : exigences qui ne peuvent pas être implémentées si celle-ci l'est.</td></tr>
<tr><td colspan="3">Priorité : une estimation de la valeur de l'acheteur.</td></tr>
<tr><td colspan="3">Documents relatifs : référence à des documents qui illustrent et expliquent cette exigence.</td></tr>
<tr><td colspan="3">Historique : création, modifications, destruction apportées à l'exigence.</td></tr>
</table>

Fig. 2.2 – Structure Volere

Chapitre 3

Fournir de la bonne documentation

3.1 Documentation du code

La meilleure façon de produire une documentation de grande qualité pour votre projet est de le documenter au fur et à mesure. Ce procédé favorise :

- La compréhension de ce que l'on veut écrire et ce avant de l'écrire.
- La compréhension du code pour soi.
- La compréhension du code pour les autres.

Remarquons que trop de commentaires pourraient bien avoir l'effet inverse de celui attendu et qu'il est important de respecter et suivre des règles bien définies telles que des commentaires structurés mis avant une classe, une méthode ou un attribut et qui seront extraits automatiquement pour une mise en page lisible à l'aide d'un outil tel que « msdn ».

Pour Visual Basic par exemple, il est donc possible d'insérer un commentaire XML dans des applications VB.NET avant une définition de classe, une propriété, des variables de membre ou une méthode afin de permettre un génération automatique de documentation et ce dans plusieurs formats tels que aide HTML de type MSDN (.chm), aide Visual Studio .NET (HTML Help 2) et pages Web de type MSDN-Online.

Chapitre 4

Assurance qualité

Outre les tests et les revues de code, une amélioration de la qualité du processus logiciel de l'Unité Régionale serait de tester la qualité des exigences dès que l'on commence à les écrire. Pour ce faire, la structure Volere (figure 2.2) inclut un critère de satisfaction. Ce critère permet de spécifier une exigence de manière claire, signifiant que l'exigence n'est pas ambiguë ou mal comprise. S'il n'y a aucun critère de satisfaction, il n'y a aucune façon de savoir si une solution correspond à l'exigence.

Il s'agit donc d'évaluer quantitativement l'exigence en spécifiant son critère de satisfaction. Ce dernier représente une mesure objective de la signification de l'exigence et c'est ce critère qui va permettre d'évaluer si une solution donnée satisfait vraiment l'exigence.

Afin d'assurer une pleine couverture des objectifs des clients, à savoir, assurer que le logiciel vérifie bel et bien toutes les exigences, il nous serait possible de faire la correspondance entre les tests et les exigences en associant à une exigence une liste de tests. Bien que cette possibilité ne soit pas inclue dans la structure du modèle de Volere, il nous paraîtrait intéressant de l'y incorporer.

4.1 Rational Unified Process

Dans l'optique de gérer et contrôler la qualité, et au vu de l'intérêt porté par le Capitaine Coulombe au processus UP, il serait intéressant d'utiliser la méthodologie RUP afin d'améliorer la qualité du processus de développement.

RUP est une approche itérative pour le développement OO de systèmes et possède trois caractéristiques :

1. **Piloté par les cas d'utilisation**, et ce de la phase d'inception à la phase de déploiement.

2. **Centré sur l'architecture** : Le processus cherche à comprendre les aspects statiques et dynamiques les plus significatifs du système. L'architecture permet de réaliser les besoins exprimés par les utilisateurs à travers les cas d'utilisation en tenant compte d'autres facteurs tels que la plate-forme d'exécution, les interfaces utilisateurs, les besoins non fonctionnels.

3. **Itératif et incrémental** : Les itérations désignent des étapes d'enchaînement d'activité et les incréments correspondent à des stades de développement du produit. Chaque itération est considérée comme un mini-projet qui donne lieu à un incrément. Les itérations se succèdent dans un ordre logique pour prendre en compte les cas d'utilisation et traiter en priorité les risques majeurs et les problèmes imprévus.

Remarquons que ces trois notions sont parfaitement complémentaires :

- Les cas d'utilisation doivent trouver leur place dans l'architecture.
- L'architecture doit prévoir la réalisation de tous les cas d'utilisation présents et à venir.
- L'architecture et les cas d'utilisation évoluent donc conjointement.
- L'architecture fournit la structure qui servira de cadre au travail effectué au cours des itérations tandis que les cas d'utilisation définissent les objectifs et orientent le travail de chaque itération.

En outre, cette démarche présente plusieurs avantages :

1. **Risque financier limité** : S'il faut reprendre une itération, la valeur du système n'est pas engagée dans son entier.
2. **Risque de retard limité** : Les risques, identifiés et résolus dès les premiers stades de développement, ne remettent pas en cause le travail déjà effectué.
3. **Accélération du rythme de développement** : Travail plus efficace vers des objectifs clairs à court terme.
4. **Adaptation à l'évolution des besoins** : Les besoins des utilisateurs ne peuvent être intégralement définis à l'avance, ils se dégagent peu à peu des itérations successives.

4.2 Les quatre étapes du processus

Le processus RUP consiste en des cycles qui peuvent se répéter tout au long de la vie d'un système. Un cycle contient les quatre étapes suivantes :

1. **Inception** : Objectifs du projet - Besoins des stakeholders - Estimations des planning et coûts - Frontière du projet.
2. **Elaboration** : Analyse du problème - Plan du projet - Description architecturale et détaillée - Analyse de risque et de stabilité architecturale.
3. **Construction** : Développement - Tests - Gestion des ressources, coûts et planning - Vérification de la qualité.
4. **Transition** : Distribution aux utilisateurs - Formations - Documentation.

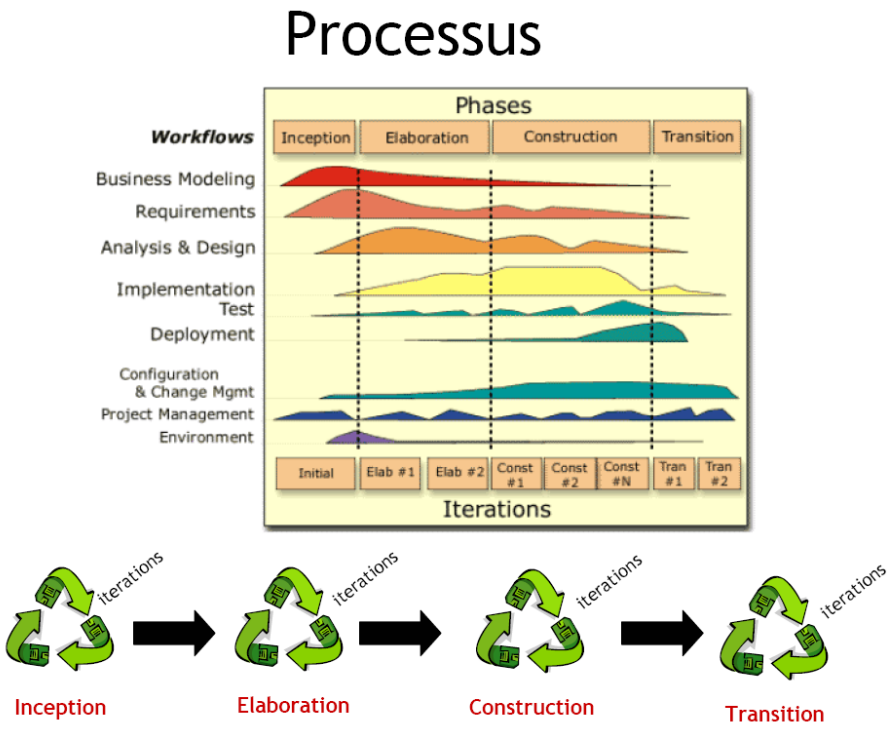

Fig. 4.1 – Rational Unified Process

Bibliographie

[1] **Processus de l'estimation**, *Modèle d'estimation*, Jean-Marc Desharnais, Alain Abran.

[2] **Volere**, *Requirements Specification Template*, Edition 11, Atlantic Systems Guild.

[3] **Projet SPINOV WP5.3**, *Plan de cahier des charges et spécification des exigences non fonctionnelles avec Volere*, CRP Henri Tudor - CITI, The Atlantic Systems Guild Inc.

[4] **Traçabilité**, *Toward Reference Models for Requirements Traceability*, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 27, NO. 1, JANUARY 2001.

[5] **Génie logiciel**, *Ingenierie des logiciels*, Naji Habra, F.U.N.D.P.

[6] **Génie logiciel**, *Ingenierie des exigences*, Patrick Heymans, F.U.N.D.P.

Annexe C

Solution qualité - Logilys

Université du Québec
École de technologie supérieure

Département Génie Logiciel

Logilys - Solution qualité

Antoine Robaeys
antoine.robaeys@student.fundp.ac.be

7 Novembre 2006

École de Technologie Supérieure - 1100 rue Notre-Dame Ouest - Montréal
Tél. : (514) 396-8800 - Fax : (514) 396-8950

Table des matières

Introduction

Ce document présente une solution qualité qui répond aux deux premiers problèmes énoncés dans le plan d'action pour Logilys du 27 septembre 2006.

Tout d'abord, nous présenterons un modèle de spécification des exigences logicielles (SRS) basé sur la norme IEEE 830-1998. Ce modèle suggère un ensemble d'éléments à préciser pour les exigences d'un système logiciel.

Dans un second chapitre, nous verrons l'outil de support aux exigences « GenSpec ». Cet outil permet de maintenir de manière structurée les « exigences spécifiques » du document SRS.

Enfin, nous illustrerons l'utilisation de GenSpec avec une partie des exigences spécifiques du logiciel « ProDon ».

Chapitre 1

Spécification des exigences logicielles

Par exigence, l'IEEE 610.12 entend :

- Une condition ou une capacité nécessaire à un utilisateur pour résoudre un problème ou atteindre un objectif ;
- Une condition ou une capacité que doit posséder un système afin de satisfaire aux termes d'un contrat, d'une norme ou d'une spécification formellement imposée ;
- La représentation documentée de cette condition ou capacité.

Le document SRS (Software Requirements Specification) se charge de spécifier les exigences. Il fournit des bénéfices tels que :

- *Établir un accord entre les clients et les développeurs :*

  La description complète des fonctions à réaliser par le logiciel spécifié dans le SRS, va aider l'utilisateur potentiel à déterminer si le logiciel répond à ses besoins.

- *Réduire le temps de développement de l'effort :*

  La préparation du SRS oblige les différents groupes concernés à considérer rigoureusement toutes les exigences avant que la conception ne commence.
  Une révision approfondie des exigences du SRS peut révéler des oublis, des incompréhensions et des inconsistances assez tôt dans le cycle de développement, à savoir, lorsque ces erreurs sont les plus faciles à corriger.

- *Fournir une base pour estimer les coûts et les échéanciers :*

  La description du produit à développer, tel que spécifier dans le SRS, est une base réaliste pour l'estimation du coût d'un projet.

- *Fournir une base pour la validation et la vérification :*

  Il est aisé de développer des plans de validation et de vérification à partir d'un bon SRS, permettant ainsi de fournir une base à la mesure.

Le point suivant présente un modèle SRS, basé sur la norme IEEE 830-1998, en décrivant chacune des sections qui le structure. Il est important de remarquer que ce modèle est adaptable en fonction du type de projet et des besoins pour ce projet car un modèle organisationnel optimal pour toutes les situations n'existe pas.

1.1 Modèle SRS

1.1.1 INTRODUCTION

L'introduction d'un SRS doit fournir une vue d'ensemble de son contenu.

Objectif

Cette sous-section doit :

- Définir les objectifs du SRS.
- Spécifier le public-cible du SRS.

Portée

Cette sous-section doit :

- Identifier par nom le système logiciel à produire.
- Expliquer brièvement ce qu'il doit faire et, si nécessaire, ce qu'il ne doit pas faire.
- Décrire le contexte de l'application dans lequel le produit s'intègre, incluant les objectifs, les bénéfices et retombées du projet.

Définitions, acronymes et abréviations

Cette sous-section doit fournir les définitions de tous les termes, acronymes et abréviations requises pour interpréter correctement le document SRS.

Remarquons que cette information peut être fournie par référence, via des annexes au SRS.

Références

Cette sous-section doit lister tous les documents, normes, rapports, etc., pertinents en identifiant leur bibliographie. Cela peut s'ajouter aux annexes si nécessaire.

Aperçu

Cette sous-section doit :

- Décrire ce que le reste du SRS contient.
- Expliquer comment le SRS est organisé.

1.1.2 DESCRIPTION GÉNÉRALE

La description générale d'un SRS doit fournir une vue d'ensemble du produit logiciel dont les détails seront spécifiés dans la section 3 du document SRS.

Fonctions du produit

Cette sous-section doit fournir un résumé des fonctions majeures du logiciel (à détailler en section 3) pour donner une vue d'ensemble au lecteur.

Description des utilisateurs

Cette sous-section doit décrire les caractéristiques générales des utilisateurs potentiels du logiciel incluant le niveau d'éducation, d'expérience et de connaissance technique.

Contraintes

Cette sous-section doit fournir une description générale des éléments qui limitent les options de développement, tels que les :

- Règlements des organismes gouvernementaux, syndicaux,...
- Limitations du matériel (timing des signaux,...).
- Interfaces à d'autres applications.
- Opérations en parallèle.
- Fonctions d'audit.
- Fonctions de contrôle.
- Exigences de langages de programmation.
- Protocoles de communication.
- Exigences de fiabilité.
- Aspects critiques de l'application.
- Considérations de sûreté et de sécurité.

Hypothèses et dépendances

Cette sous-section doit lister les prémisses de base qui affectent les exigences reprises dans le SRS. Remarquons qu'il ne s'agit pas ici de contraintes de conception mais plutôt de changements qui pourraient affecter les exigences du SRS.

Par exemple, une hypothèse serait qu'un système d'exploitation particulier soit disponible sur le matériel désigné pour le logiciel. Si dans les faits, le système d'exploitation n'est pas disponible, le SRS devrait alors changer en conséquence.

1.1.3 EXIGENCES SPÉCIFIQUES

Les exigences spécifiques d'un SRS, à savoir, les exigences logicielles, représentent la section la plus importante du document.

Ces exigences doivent être listées à un niveau suffisamment détaillé pour :

1. Permettre aux concepteurs de concevoir un système qui satisfait ces exigences.
2. Permettre aux testeurs de vérifier que le système satisfait bien ces exigences.

Nous verrons, dans le chapitre suivant, l'outil de support aux exigences, « GenSpec », qui permet d'organiser et de maintenir, de manière structurée, ces exigences spécifiques.

Les sous-sections suivantes présentent les exigences à inclure dans les exigences spécifiques :

Exigences fonctionnelles

Les exigences fonctionnelles définissent les actions fondamentales pour accepter les entrées, effectuer les traitements et générer les sorties. Ces exigences sont généralement listées en commençant par « Le système doit... ».

Dans cette sous-section, les traitements (fonctions) effectués par le système doivent être décrits en détail. Il peut également être approprié de partitionner les exigences fonctionnelles en sous-fonctions ou sous-processus. Remarquons que cela n'implique pas que la conception du logiciel sera aussi partitionnée de la sorte.

Contraintes de performance

Cette sous-section doit spécifier les exigences numériques statiques et dynamiques du logiciel et/ou des interactions avec l'utilisateur.

Les exigences numériques statiques sont, par exemple, le nombre de terminaux à supporter, le nombre d'utilisateurs simultanés à supporter, etc.

Les exigences numériques dynamiques sont, par exemple, le nombre de transactions, la quantité de données à gérer endéans une certaine période, etc.

Il est important que toutes ces exigences soient spécifiées en termes mesurables.

Contraintes de conception

Cette sous-section doit spécifier les contraintes qui peuvent être imposées par :

- Certains standards, normes et règlements : Format de rapport, nomenclature des données,...
- Des limites matérielles.
- Des limites logicielles.
- Politiques d'intégrité des données : Type d'information utilisé par les différentes fonctions, contraintes d'intégrité,...

Contraintes d'attributs

Certains attributs logiciels peuvent être considérés comme des exigences. Il est important que ces attributs, nécessaires au logiciel, puissent être spécifiés afin de pouvoir objectivement vérifier leur réalisation.

Ces attributs sont les suivants :

- Disponibilité : Robustesse du système, heures pleines,...
- Sécurité : Protection contre intrusions, modification, destruction,...
- Maintenabilité : Exigences de modularité, complexité,...
- Portabilité : Utilisation d'un langage portable, d'un système d'exploitation particulier,...

Autres exigences

De manière optionnelle, il est possible d'ajouter, dans cette sous-section, des exigences qui ne trouveraient pas leur place dans une des sous-sections précédentes.

1.1.4 INFORMATIONS COMPLÉMENTAIRES

Les informations complémentaires rendent l'utilisation du SRS plus aisée.

Annexes

Les annexes ne sont pas toujours considérés comme faisant partie du SRS et ne sont pas toujours nécessaires. Ils peuvent inclure une description des problèmes à résoudre par le logiciel, des informations qui pourraient aider le lecteur du SRS, etc.

Lorsque des annexes sont inclus, le SRS doit explicitement préciser si les annexes font ou non partie des exigences.

Index

La table des matières et index sont très important dans un soucis de clarté.

Chapitre 2

Outil de support aux exigences

Hydro-Québec a développé une solution en 2001, rigoureusement basée sur des normes internationales, qui lui permet de résoudre les problèmes les plus souvent rencontrés en ingénierie des exigences. Cette solution est simple et systématique mais cependant, son application peut être ardue sans le support d'un outil logiciel.

Hydro-Québec a donc développé son propre outil d'ingénierie des exigences, à savoir, GenSpec. Ce dernier n'offre aucune facilité de conception ni de génération de code logiciel. Par contre, il est simple, gratuit, offre beaucoup de flexibilité quant au format des documents générés et est spécifiquement axé sur la méthode qui consiste à suivre les règles suivantes pour la gestion des exigences spécifiques d'un document SRS :

2.1 Méthode

2.1.1 Structurer les exigences de façon hiérarchique

- Spécifier l'ensemble des exigences sous la forme d'un arbre hiérarchique : les exigences « enfant » sous les exigences « parent », les unes découlant des autres.
- Spécifier les exigences parent de façon à ce que chacune d'elles soit la synthèse de ses enfants. Cela aide à présenter les exigences de façon graduelle, systématiquement de la vue d'ensemble à la vue détaillée.
- Ordonner les exigences de façon à ce que les informations nécessaires à leur compréhension soient contenues dans l'exigence ou dans celles qui la précèdent.

2.1.2 Limiter chacune des exigences à un paragraphe

- Utiliser un seul paragraphe par exigence, si l'exigence peut être spécifiée dans un seul paragraphe simple, clair et concis.
- Sinon, diviser l'exigence en plusieurs exigences « validables », d'un seul paragraphe, si elle peut être ainsi divisée.
- Sinon, référer à une annexe décrivant l'exigence.

2.1.3 Référer aux exigences sources

Lorsqu'une exigence découle directement d'une autre spécifiée dans un autre document, y référer. Référer non seulement au document mais aussi au paragraphe spécifiant l'exigence. Cet autre document peut être une spécification, un courriel, un compte-rendu de réunion, etc.

2.1.4 Relier les exigences par des renvois

Par des renvois, relier les exigences ayant des liens logiques, les exigences devant potentiellement être modifiées si l'une d'elles est modifiée. Cela exclut les liens parent-enfant, puisqu'ils sont déjà présents dans la structure des exigences. Dans tous les cas, spécifier la raison du renvoi.

Ainsi structurées, les exigences peuvent être vues comme un simple empilement de briques interreliées. Assurément, cela facilite l'ajout, le retrait et la modification d'exigences. De surcroît, cela réduit le risque d'incohérences. En effet, une incohérence est souvent introduite dans un texte lors d'une modification d'une information redondante : la modification n'est pas effectuée partout où l'information apparaît.

2.1.5 Respecter les règles de rédaction

Respecter les règles fondamentales de rédaction technique, principalement les suivantes :

- Utiliser le langage courant et n'utiliser des mots techniques que si la précision l'exige.
- Faire des phrases simples, claires et concises.
- Uniformiser les exigences : rédiger de façon analogue des exigences analogues et de façon identique des exigences identiques.

Cela implique notamment de (1) ne pas utiliser de synonymes, même pour les mots de liaison et (2) utiliser systématiquement le même verbe pour spécifier une exigence, tel « devoir ».

2.2 GenSpec

La figure 2.1 présente l'interface homme-machine de GenSpec : à droite apparaît l'arbre d'exigences ; à gauche, le formulaire d'entrée de l'exigence sélectionnée.

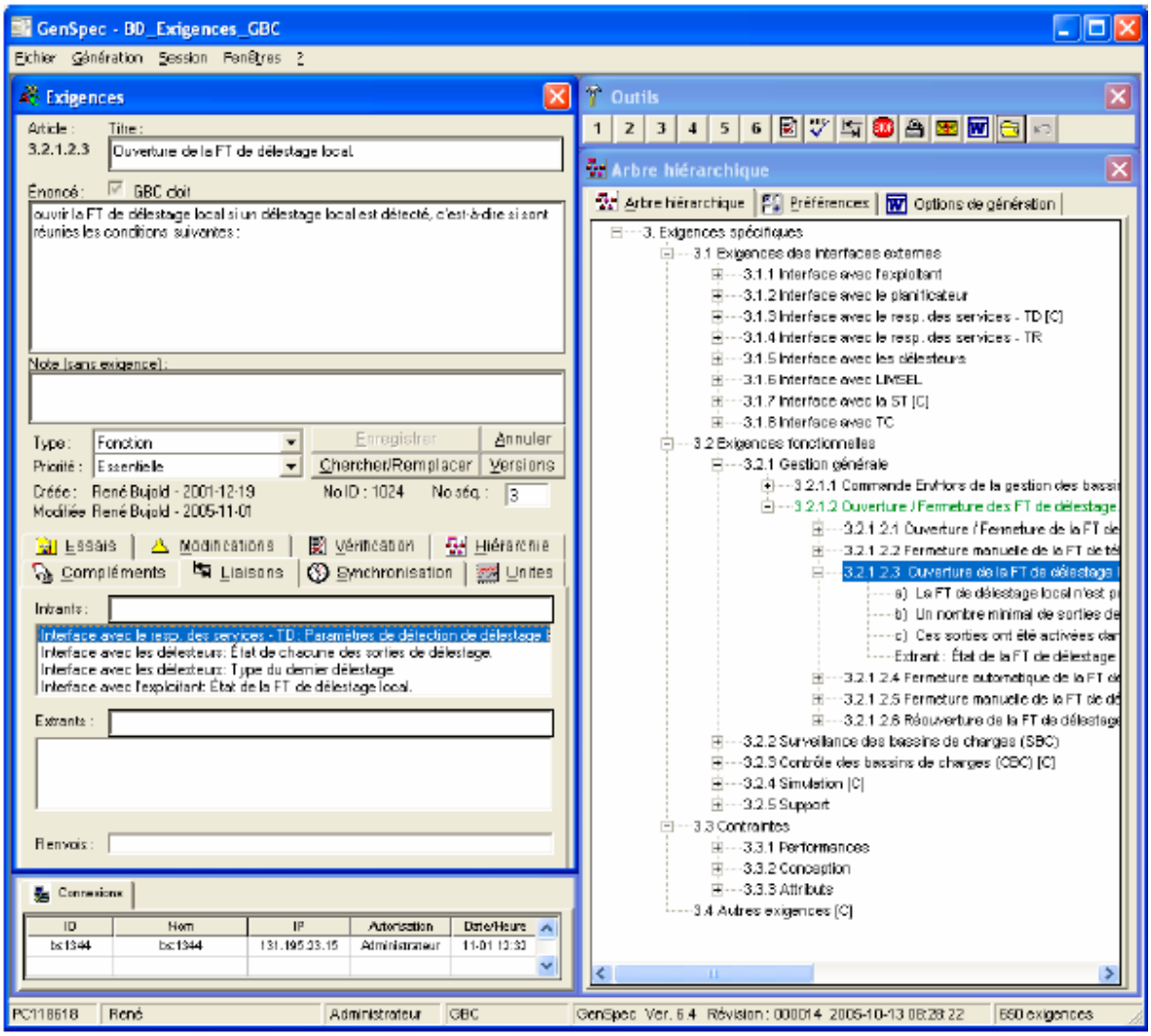

FIG. 2.1 – GenSpec

Les fonctions principales de GenSpec sont les suivantes :

2.2.1 Définition des exigences

Cette fonction permet à plusieurs utilisateurs en même temps (multi-utilisateur) d'entrer ou de modifier des exigences ; elle supporte notamment les commandes Chercher et Remplacer (figure 2.2).

FIG. 2.2 – Recherche / Remplacement

2.2.2 Caractérisation des exigences

Par exigence, cette fonction génère un numéro de référence unique et permet d'entrer l'identification de la source (référence à un paragraphe d'un autre document), la priorité, une note, un commentaire et un fichier joint, tous pouvant être générés dans le document d'exigences (figure 2.3). Remarquons que les exigences commentées apparaissent en vert dans l'arbre d'exigences.

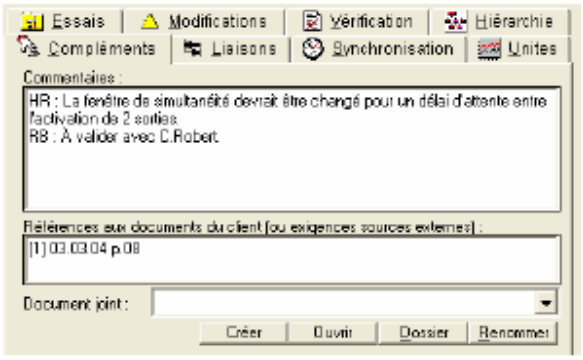

FIG. 2.3 – Données complémentaires

2.2.3 Structuration et liaison des exigences

Cette fonction permet de structurer et de lier les exigences (renvois) par de simples commandes clic et glisse. De plus, elle offre des facilités de navigation telle une commande d'aller-retour rapide entre l'origine et la destination d'un lien. Remarquons qu'à la demande, les exigences reliées peuvent apparaître en bleu dans l'arbre d'exigences.

2.2.4 Évaluation de conformité aux exigences

Par exigence, cette fonction permet d'entrer des procédures d'évaluation de conformité, et le résultat de cette évaluation (figure 2.4). De plus, elle permet de générer un rapport d'évaluation contenant les exigences, leurs procédures d'évaluation et les résultats de cette évaluation.

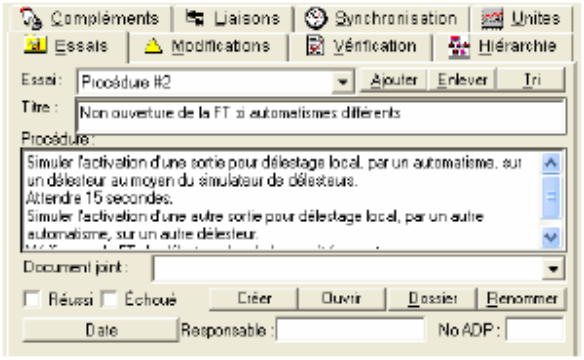

FIG. 2.4 – Procédures d'essais

2.2.5 Contrôle et analyse des exigences

Cette fonction empêche d'introduire des incohérences de hiérarchie ou de liaison d'exigences – règles de hiérarchie et de liaison paramétrables – et, en particulier, de supprimer une exigence à laquelle d'autres renvoient. De plus, elle offre un vérificateur d'exigences (figure 2.5), y compris un vérificateur d'orthographe et de grammaire. De surcroît, elle permet de générer un tableau Sources Vs Exigences, facilitant la vérification de l'exactitude des exigences. Remarquons qu'à la demande, les exigences en erreur peuvent apparaître en rouge dans l'arbre d'exigences.

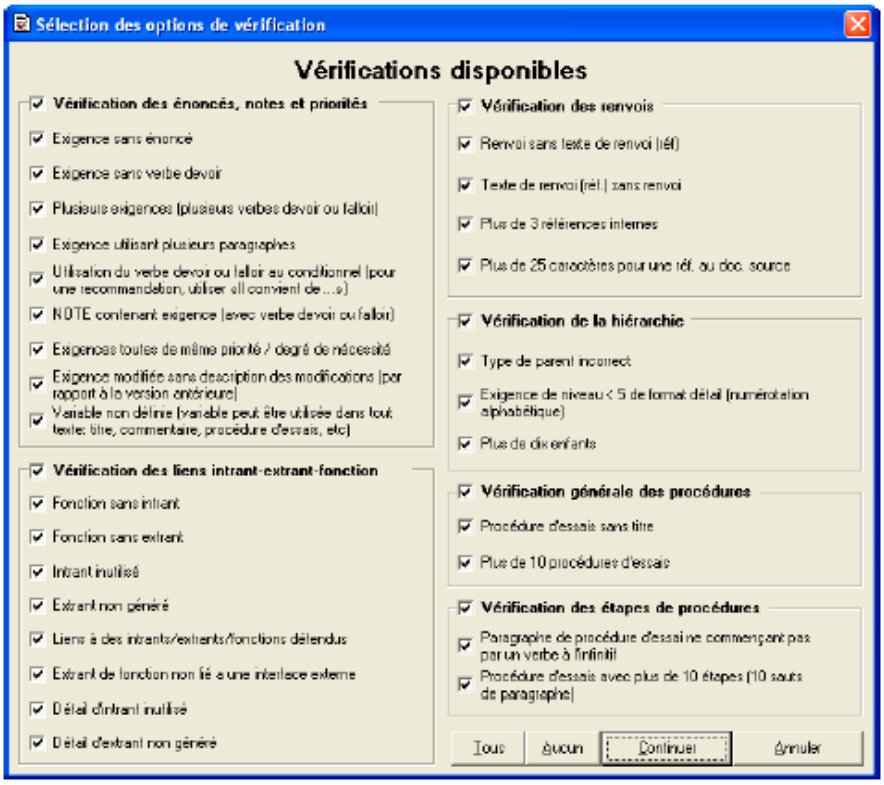

FIG. 2.5 – Vérification des exigences

2.2.6 Normalisation des exigences

Cette fonction permet de générer automatiquement des textes de début d'exigence selon le type d'exigence – texte et type paramétrables. De plus, elle permet de définir et d'utiliser des variables dans les textes d'exigences.

2.2.7 Configuration des documents d'exigences

Cette fonction offre une grande quantité d'options de formatage des documents générés ; elle permet notamment d'en exclure des exigences et d'en inclure d'autres avec la mention « Non applicable ».

2.2.8 Gestion de l'historique des exigences

Cette fonction permet d'entrer et de visualiser la raison de modification d'une exigence par rapport à la version antérieure, et d'enregistrer une version formelle de l'ensemble des exigences. De plus, elle permet de comparer la version actuelle avec une version antérieure, et de ramener une ou toutes les exigences telles qu'elles étaient à une version antérieure. Enfin, elle permet de générer un tableau Historique des modifications d'exigence. En complément, elle permet d'enregistrer de simples copies de sécurité des exigences.

2.3 Avantages

2.3.1 Réduction des coûts

GenSpec réduit les coûts de l'ingénierie des exigences :

1. Concentre les efforts sur les exigences plutôt que sur le formatage, les documents étant générés automatiquement.
2. Permet de structurer facilement les exigences.
3. Permet de générer automatiquement des textes de début d'exigence, aidant, de surcroît, à la normalisation.
4. Facilite la liaison des exigences.
5. Génère automatiquement le rapport d'évaluation.

2.3.2 Facilitation de la lecture

GenSpec facilite la lecture des exigences :

1. Oriente à bien structurer les exigences, un paragraphe par exigence, graduellement, de la vue d'ensemble à la vue détaillée, basé sur une notion d'arbre d'exigences : exigences parents sous lesquelles se retrouvent des exigences enfants, les exigences parents étant la synthèse (vue d'ensemble) de leurs enfants.
2. Permet de lier chacune des exigences aux besoins du client ou exigences sources.
3. Offre une grande quantité d'options de formatage des documents générés.
4. Uniformise les documents d'exigences, parce que générés automatiquement.

2.3.3 Réduction de la quantité d'erreurs

GenSpec réduit la quantité d'erreurs d'exigences :

1. Génère un tableau de vérification : Sources Vs Exigences.
2. Facilite la couverture de l'ensemble des exigences : définition de tous les intrants et extrants sur les interfaces externes et liaison de chacun d'eux aux fonctions, et inversement.
3. Empêche d'introduire des incohérences de hiérarchie ou de liaison d'exigences.
4. Simplifie les corrections et mises à jour des exigences et des procédures d'évaluation, ces procédures étant définies avec les exigences.
5. Offre un vérificateur d'exigences.
6. Permet d'utiliser des variables dans les textes d'exigences.
7. Gère un historique des modifications d'exigence.

2.3.4 Respect des normes internationales

GenSpec respecte des normes internationales, dont les normes pertinentes de IEEE et de ISO/CEI. En effet, en plus des avantages ci-dessus, GenSpec :

1. Fixe un numéro de référence unique par exigence.
2. Oriente à énoncer des exigences validables, faisant abstraction des moyens de réalisation, les intrants et extrants d'exigences ne pouvant être que ceux des interfaces externes.
3. Facilite la modification des exigences, étant bien structurées et les liens entre exigences étant clairement définis : liens parent-enfant, liens intrant-extrant-fonction et autres liens (renvois).
4. Définit la priorité de chacune des exigences.

Chapitre 3

Utilisation de GenSpec

Ce chapitre va permettre d'illustrer l'utilisation de GenSpec pour la spécification d'une partie des exigences spécifiques du logiciel « ProDon ».

3.1 Création d'un nouveau projet

GenSpec permet de créer un projet en se basant sur différents modèles qui diffèrent surtout par leur structure. Nous avons opté pour le modèle de la figure 3.1 qui correspond principalement à la structure de la norme 830 de IEEE et qui permet de structurer les exigences spécifiques d'un document SRS basé sur la même norme.

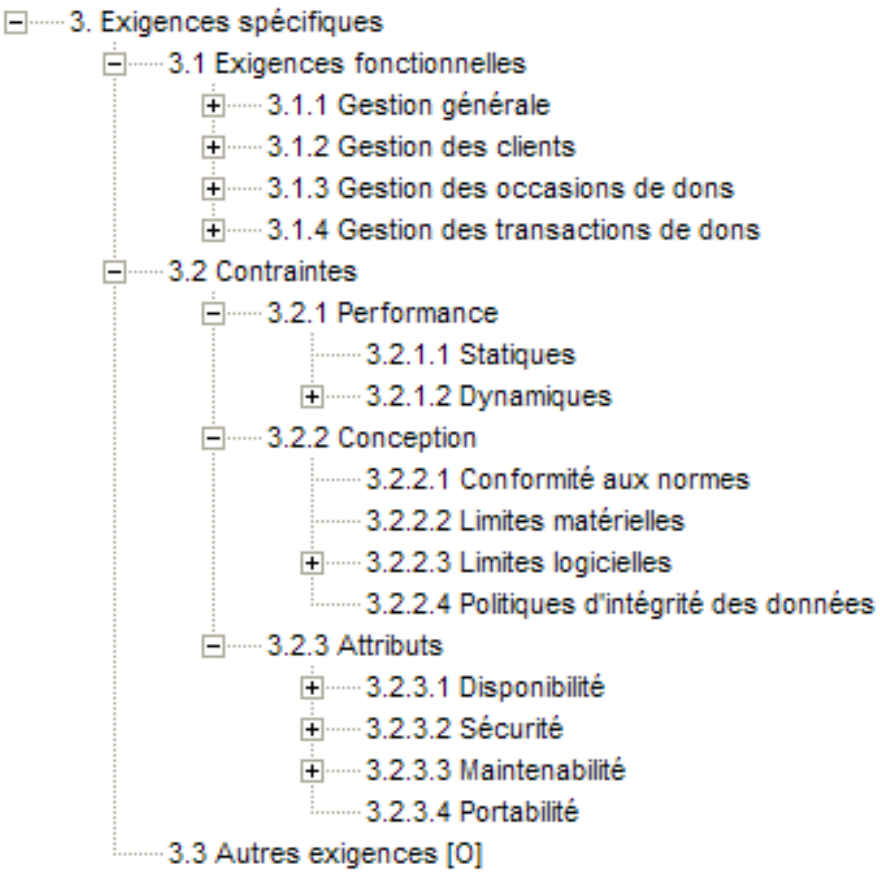

Fig. 3.1 – Modèle IEEE 830

3.2 Gestion des usagers

La gestion des usagers se fait à partir du menu Session et est relative au projet ouvert, c'est-à-dire qu'un usager ayant un compte dans le projet X n'a pas automatiquement un compte dans le projet Y.

Il y a cinq types d'usager dans GenSpec :

1. **Administrateur** : Possède l'accès à toutes les fonctionnalités de GenSpec.

2. **Utilisateur** : Possède l'accès à toutes les fonctionnalités de GenSpec excepté la gestion des usagers et la génération de versions officielles.

3. **Visiteur** : Possède seulement des droits de lecture de tous les champs et des droits de génération des documents.

4. **Commentateur** : Possède les mêmes restrictions qu'un Visiteur excepté qu'il a les droits d'écriture sur les champs Commentaires.

5. **Testeur** : Possède les mêmes restrictions qu'un Visiteur excepté qu'il a les droits d'écriture sur les champs Commentaires et les champs de l'onglet Essais.

Par défaut, l'usager qui a créé un nouveau projet se voit accorder le droit d'accès Administrateur.

3.3 Manipulation des exigences

Afin d'ajouter une exigence, il suffit de cliquer dans la fenêtre Arbre hiérarchique, avec le bouton droit de la souris, sur l'exigence parent de l'exigence à ajouter et de sélectionner l'option « Ajouter sous cette exigence... ». Ces exigences peuvent ensuite être aisément modifiées, déplacées, supprimées, etc.

3.3.1 Types d'exigences

Chaque exigence de l'arbre hiérarchique est d'un *type* bien précis. Afin de structurer au mieux les exigences, il convient de respecter la hiérarchie de types d'exigences de la figure 3.2 où sous chaque type d'exigence sont présentés les types possibles.

Le niveau maximum de l'arbre hiérarchique est paramétrable (entre 1 et 6) mais il est toutefois recommandé, pour les exigences fonctionnelles, d'utiliser les 6 niveaux afin d'obtenir un arbre équilibré.

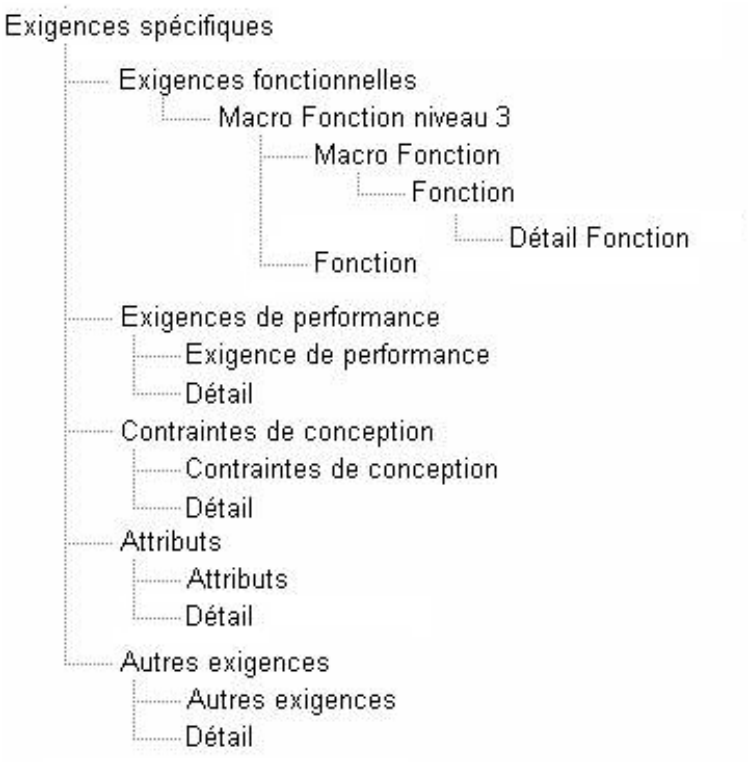

FIG. 3.2 – Types d'exigences

- **Macro Fonction** est une fonction de très haut niveau, composée de fonctions. Une macro ne peut être argument d'une fonction.
- **Fonction** est une tâche, action ou activité qui doit être exécutée pour parvenir à un résultat.
- **Détail** est un niveau de définition supplémentaire. Le détail est utilisé lorsqu'il est nécessaire de préciser une exigence sans augmenter inutilement la définition de cette dernière. Un détail est numéroté de façon alphabétique, et non numérique, dans l'arbre hiérarchique.
- **Autres exigences** est un type qui concerne les exigences qui n'entrent pas dans les catégories déjà définies.

Remarquons que les différents types proposés par GenSpec sont paramétrables et l'utilisateur peut donc définir lui-même ses propres types d'exigences.

En spécifiant les exigences du logiciel « ProDon », selon la hiérarchie précédente via GenSpec, on obtient la hiérarchie fournie dans l'annexe 1.

3.3.2 Caractérisation d'une exigence

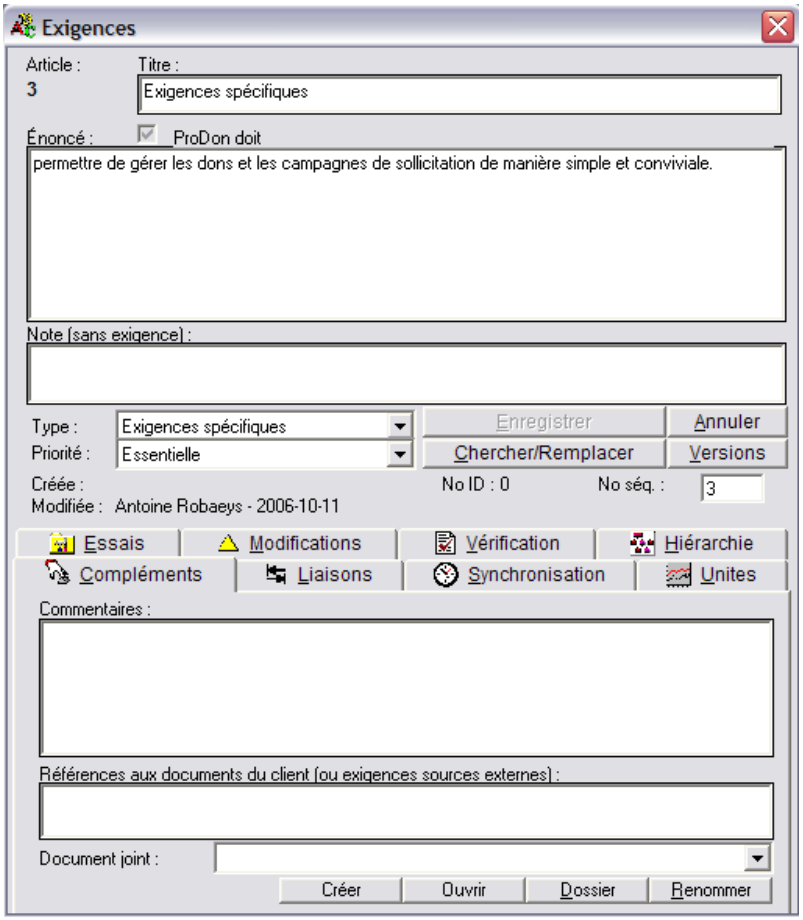

FIG. 3.3 – Caractérisation d'une exigence

Titre

Le titre d'une exigence est le nom de cette exigence dans l'arbre hiérarchique et est fonction de son type :

- **Macro Fonction** : Son titre doit, généralement, être de la forme : « Gestion de... ».
- **Fonction** : Son titre doit correspondre au but de l'exigence. Par exemple, « Ajouter un client ».
- **Détail** : Son titre doit correspondre au nom du détail. Par exemple, « Nom », « adresse », etc.
- **Contraintes** : Leur titre doit mentionner ce qui fait l'objet de la contrainte. Par exemple, « Recherche dans la BD », « Système d'exploitation », etc.

Énoncé

L'énoncé consiste à mentionner ce que l'exigence en cours d'édition amène de plus dans la spécification. L'énoncé d'une exigence est automatiquement précédé d'un bout de phrase prédéterminé dans le cas où l'utilisateur entre un énoncé commençant par une minuscule.

Par exemple, pour l'exigence « Ajouter un client », l'énoncé est : « ProDon doit permettre d'ajouter un client en spécifiant son nom ou prénom ainsi qu'éventuellement ses autres informations ». Remarquons que les deux premiers mots (bout de phrase prédéterminé) ont été ajouté automatiquement.

Liaisons

Un renvoi est un lien direct à implanter entre deux exigences qui ont des liens logiques. Le logiciel nécessite donc de lui spécifier, au moyen de la souris, vers quelle exigence effectuer le renvoi.

Par exemple, l'exigence « Chercher un client » dans « Gestion des clients », est identique, excepté au niveau de leur priorité, à l'exigence « Chercher un client » dans « Ajouter une transaction de dons ».

Note

Il est possible d'ajouter une note complétant l'énoncé de l'exigence, qui ne doit introduire **aucune autre exigence** car elle serait alors dite cachée, et les développeurs risqueraient de ne pas en tenir compte en étudiant le document d'exigences. Une note peut également être utilisée afin de justifier la présence d'un renvoi.

Par exemple, la note de l'exigence « Chercher un client » dans « Gestion des clients », justifie la présence du renvoi à l'exigence « Chercher un client » dans « Ajouter une transaction de dons », en stipulant : « Cette exigence est identique à celle qu'elle référence dans le renvoi excepté sa priorité. ».

Type

Il est possible de modifier le type d'une exigence ajoutée précédemment, en fonction de sa position dans l'arbre hiérarchique, afin de ne pas devoir la supprimer puis la rajouter.

Priorité

Le degré de priorité doit être choisi entre trois valeurs différentes : Essentielle, Complémentaire et Optionnelle. Remarquons que le degré de priorité d'une exigence enfant d'une exigence complémentaire peut très bien être essentielle. En effet, ce degré de nécessité est relatif à l'existence même du parent. Une exigence enfant est essentielle SI son exigence parent complémentaire ou optionnelle devient un évènement vérifié.

Par exemple, l'exigence « Chercher une transaction de dons » dans « Gestion des transactions de dons », a un degré de priorité équivalent à Complémentaire. Dès lors, il s'avère que nous considérons alors comme complémentaire (non essentiel) de retrouver une transaction via une action de recherche. En effet, les transactions peuvent être consultées via une simple liste.

Commentaires

Sous l'onglet Compléments de la fenêtre Exigences, un champ est réservé aux commentaires sur l'exigence spécifiée. Ce champ comprend donc des points sur lesquels porter son attention lors du développement.

En outre, il serait intéressant d'utiliser cet espace afin d'y insérer toute requête de changement relative à l'exigence spécifiée. Rappelons nous qu'une exigence commentée apparaît en vert dans l'arbre hiérarchique et qu'il est possible d'inclure ou non ces commentaires lors de la génération de documents *Word*.

Par exemple, nous pourrions désirer enregistrer une requête de changement concernant le temps de réponse d'une recherche dans une base de données restreinte (dont le nombre d'objets est inférieur ou égal à 2500). Nous pourrions également désirer mentionner le nom de la personne qui devra se charger de valider cette requête ainsi que lier des documents qui illustrent cette requête.

Pour ce faire, il suffirait donc d'insérer, dans le champ commentaires de l'exigence « BD restreinte », dans la contrainte de performance dynamique, « Recherche dans la BD » : « RC : Le temps de recherche maximum devrait être réduit à 2 secondes. (À valider par C. Dubois) ».
Il suffirait aussi de joindre des documents *Word* illustrant cette requête, via la fonction « Document joint » qui se situe en dessous du champ commentaire.

Modification

Dans le cas où l'exigence est modifiée après sa création, l'onglet Modifications permet à l'utilisateur d'entrer un court texte spécifiant les différences entre la version antérieure et la nouvelle mise à jour.

Comme il est possible avec le logiciel de revenir vers une version antérieure, les modifications correspondantes sont conservées dans le temps. Le texte entré dans l'onglet Modifications des exigences se retrouvera dans les documents générés « Listes des modifications courantes » et/ou dans « Historique des modifications ».

- Dans le document « Liste des modifications courantes », on retrouve un tableau contenant, pour chaque exigence modifiée, la description de la modification effectuée depuis la dernière Version Officielle, le nom du responsable et la date où la modification a été effectuée.

- Dans le document « Historique des modifications », on retrouve un tableau contenant, par version officielle, les modifications qui ont été effectuées depuis la création du projet.

Par exemple, nous désirons appliquer la requête de changement vue dans une des sections précédentes, à savoir, « RC : Le temps de recherche maximum devrait être réduit à 2 secondes. (À valider par C. Dubois) ».

Nous modifions donc l'exigence « BD restreinte » en conséquence et nous explicitons les modifications par rapport à la version antérieure. Ensuite, nous supprimons la requête de changement du champ Commentaires, ainsi que les éventuels documents joints y afférents, puisque cette requête à été concrétisée en exigence.

Ci-dessous, la liste des modifications courantes telle que générée par GenSpec :

Exigence	#	Modifications	Resp.	Date
3.2.1.2.1.a BD restreinte	3552	Le temps de recherche maximum est passé de 3 à 2 secondes.	Antoine Robaeys	06-11-02

FIG. 3.4 – Liste des modifications courantes

Essais

Sous l'onglet Essais, la spécification des différents essais effectués et l'entrée de leurs résultats sont possibles. Afin de préciser de quelle manière les essais doivent être effectués, des procédures doivent être entrées.

Le premier champ sert à donner un titre à la procédure. Le champ sous ce dernier, plus volumineux, permet de spécifier la procédure en soit (les étapes à effectuer, les points à relever, les avertissements, etc.). Chaque paragraphe d'une procédure doit commencer par un verbe à l'infinitif. Tout schéma ou figure pouvant aider à effectuer la procédure peut être joint.

Deux boîtes pouvant être cochées servent à indiquer si, une fois la procédure effectuée, l'essai a Réussi ou Échoué. Il est également possible de spécifier la date où le test a été effectué, le responsable qui a supervisé les tests et le numéro d'Avis De Problème (ADP) éventuel.

Par exemple, une procédure serait de mesurer le temps de recherche d'un client dans une BD restreinte :

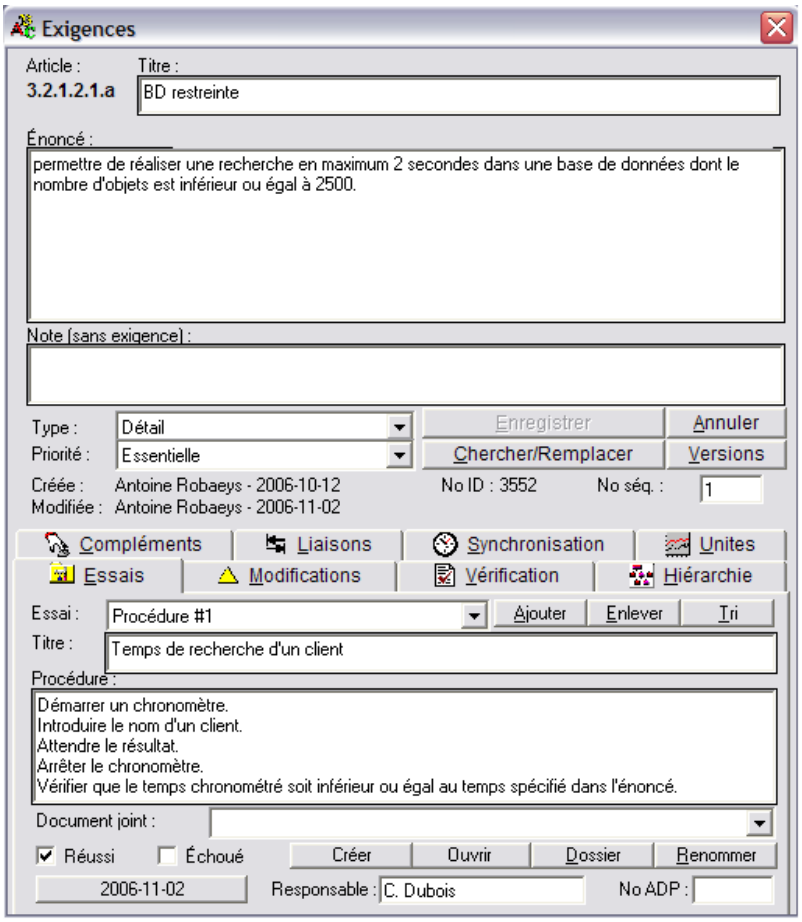

FIG. 3.5 – Procédure d'essai

Hiérarchie

L'onglet hiérarchie permet d'avoir un clair aperçu de la situation hiérarchique de l'exigence spécifiée. Remarquons qu'aucun champ ne peut être modifié via cet onglet.

3.4 Génération de documents *Word*

GenSpec permet de générer, à partir de la spécification d'exigences effectuée, 6 types de documents sous format *Word*, complets, qui incluent les paramètres voulus :

- Arbre hiérarchique des exigences. (Annexe 1)
- Spécification des exigences. (Annexe 2)
- Références aux exigences sources externes.
- Liste des modifications courantes.
- Historique des modifications.
- Cahier des essais.

L'utilisateur a le contenu complet des documents générés en son contrôle. Il peut ainsi choisir d'inclure les commentaires, les procédures, les titres d'exigences, de ne pas inclure les cases Réussi/Échoué, etc. Ce contrôle peut s'exercer sur tous les niveaux séparément et pour chacun des documents offerts en génération.

Annexe A

Arbre hiérarchique des exigences de ProDon

3. EXIGENCES SPÉCIFIQUES
  3.1. Exigences fonctionnelles
    3.1.1. Gestion générale
      3.1.1.1. Gestion des projets
        3.1.1.1.1. Consulter la liste des projets
          3.1.1.1.1.a) Navigation
          3.1.1.1.1.b) Code des projets
          3.1.1.1.1.c) Titre des projets
          3.1.1.1.1.d) Description des projets
          3.1.1.1.1.e) Client des projets
          3.1.1.1.1.f) Individu des projets
          3.1.1.1.1.g) Total de chaque projet
        3.1.1.1.2. Chercher un projet
          3.1.1.1.2.a) Code du projet
          3.1.1.1.2.b) Titre du projet
          3.1.1.1.2.c) Description du projet
          3.1.1.1.2.d) Client du projet
          3.1.1.1.2.e) Individu du projet
          3.1.1.1.2.f) Total du projet
        3.1.1.1.3. Consulter un projet
          3.1.1.1.3.a) Code du projet
          3.1.1.1.3.b) Titre du projet
          3.1.1.1.3.c) Début du projet
          3.1.1.1.3.d) Fin du projet
          3.1.1.1.3.e) Client du projet
          3.1.1.1.3.f) Description du projet
          3.1.1.1.3.g) Total du projet
          3.1.1.1.3.h) Dons reçus du projet
          3.1.1.1.3.i) Dons émis du projet
        3.1.1.1.4. Ajouter un projet
          3.1.1.1.4.a) Code du projet
          3.1.1.1.4.b) Titre du projet
          3.1.1.1.4.c) Début du projet
          3.1.1.1.4.d) Fin du projet
          3.1.1.1.4.e) Client du projet
          3.1.1.1.4.f) Description du projet
        3.1.1.1.5. Modifier un projet
          3.1.1.1.5.a) Code du projet
          3.1.1.1.5.b) Titre du projet
          3.1.1.1.5.c) Début du projet
          3.1.1.1.5.d) Fin du projet
          3.1.1.1.5.e) Client du projet
          3.1.1.1.5.f) Description du projet
        3.1.1.1.6. Retirer un projet
          3.1.1.1.6.a) Confirmation
    3.1.2. Gestion des clients
      3.1.2.1. Consulter la liste des clients
        3.1.2.1.1. Navigation
        3.1.2.1.2. Informations générales
          3.1.2.1.2.a) NUC
          3.1.2.1.2.b) Prénom
          3.1.2.1.2.c) Nom
          3.1.2.1.2.d) Appellation
          3.1.2.1.2.e) Titre
          3.1.2.1.2.f) Compagnie
          3.1.2.1.2.g) Adresse
          3.1.2.1.2.h) Langue
        3.1.2.1.3. Informations complémentaires
          3.1.2.1.3.a) Téléphones
          3.1.2.1.3.b) Courriels
          3.1.2.1.3.c) Site Internet
          3.1.2.1.3.d) Solliciteur
          3.1.2.1.3.e) Secteur
          3.1.2.1.3.f) Référence
          3.1.2.1.3.g) Commentaires
        3.1.2.1.4. Informations spécifiques
          3.1.2.1.4.a) Date de naissance
          3.1.2.1.4.b) Date d'inscription

3.1.2.1.4.c) Anonymat
3.1.2.1.4.d) Bénéficiaire
3.1.2.1.4.e) Reçus et appels

3.1.2.2. Chercher un client
- 3.1.2.2.1. Informations générales
  - 3.1.2.2.1.a) NUC
  - 3.1.2.2.1.b) Prénom
  - 3.1.2.2.1.c) Nom
  - 3.1.2.2.1.d) Appellation
  - 3.1.2.2.1.e) Titre
  - 3.1.2.2.1.f) Compagnie
  - 3.1.2.2.1.g) Adresse
  - 3.1.2.2.1.h) Langue
- 3.1.2.2.2. Informations complémentaires
  - 3.1.2.2.2.a) Téléphones
  - 3.1.2.2.2.b) Courriels
  - 3.1.2.2.2.c) Site Internet
  - 3.1.2.2.2.d) Solliciteur
  - 3.1.2.2.2.e) Secteur
  - 3.1.2.2.2.f) Référence
  - 3.1.2.2.2.g) Commentaires
- 3.1.2.2.3. Informations spécifiques
  - 3.1.2.2.3.a) Date de naissance
  - 3.1.2.2.3.b) Date d'inscription
  - 3.1.2.2.3.c) Anonymat
  - 3.1.2.2.3.d) Bénéficiaire
  - 3.1.2.2.3.e) Reçus et appels

3.1.2.3. Consulter un client
- 3.1.2.3.1. Informations générales
  - 3.1.2.3.1.a) Appellation
  - 3.1.2.3.1.b) Genre
  - 3.1.2.3.1.c) Prénom
  - 3.1.2.3.1.d) Nom
  - 3.1.2.3.1.e) Titre
  - 3.1.2.3.1.f) Compagnie
  - 3.1.2.3.1.g) Adresse
  - 3.1.2.3.1.h) Langue
- 3.1.2.3.2. Informations complémentaires
  - 3.1.2.3.2.a) Téléphones
  - 3.1.2.3.2.b) Courriels
  - 3.1.2.3.2.c) Site Internet
  - 3.1.2.3.2.d) Solliciteur
  - 3.1.2.3.2.e) Secteur
  - 3.1.2.3.2.f) Référence
  - 3.1.2.3.2.g) Commentaires
- 3.1.2.3.3. Informations spécifiques
  - 3.1.2.3.3.a) Date de naissance
  - 3.1.2.3.3.b) Date d'inscription
  - 3.1.2.3.3.c) Anonymat
  - 3.1.2.3.3.d) Activité
  - 3.1.2.3.3.e) Bénéficiaire
  - 3.1.2.3.3.f) Reçus et appels
  - 3.1.2.3.3.g) Catégories de clients
  - 3.1.2.3.3.h) Liens
  - 3.1.2.3.3.i) Caractéristiques

3.1.2.4. Ajouter un client
- 3.1.2.4.1. Informations générales
  - 3.1.2.4.1.a) Appellation
  - 3.1.2.4.1.b) Genre
  - 3.1.2.4.1.c) Prénom
  - 3.1.2.4.1.d) Nom
  - 3.1.2.4.1.e) Titre
  - 3.1.2.4.1.f) Compagnie
  - 3.1.2.4.1.g) Adresse
  - 3.1.2.4.1.h) Langue
- 3.1.2.4.2. Informations complémentaires
  - 3.1.2.4.2.a) Téléphones
  - 3.1.2.4.2.b) Courriels
  - 3.1.2.4.2.c) Site Internet

3.1.2.4.2.d) Solliciteur
3.1.2.4.2.e) Secteur
3.1.2.4.2.f) Référence
3.1.2.4.2.g) Commentaires
3.1.2.4.3. Informations spécifiques
3.1.2.4.3.a) Date de naissance
3.1.2.4.3.b) Date d'inscription
3.1.2.4.3.c) Anonymat
3.1.2.4.3.d) Activité
3.1.2.4.3.e) Bénéficiaire
3.1.2.4.3.f) Reçus et appels
3.1.2.4.3.g) Catégories de clients
3.1.2.4.3.h) Liens
3.1.2.4.3.i) Caractéristiques
3.1.2.5. Modifier un client
3.1.2.5.1. Informations générales
3.1.2.5.1.a) Appellation
3.1.2.5.1.b) Genre
3.1.2.5.1.c) Prénom
3.1.2.5.1.d) Nom
3.1.2.5.1.e) Titre
3.1.2.5.1.f) Compagnie
3.1.2.5.1.g) Adresse
3.1.2.5.1.h) Langue
3.1.2.5.2. Informations complémentaires
3.1.2.5.2.a) Téléphones
3.1.2.5.2.b) Courriels
3.1.2.5.2.c) Site Internet
3.1.2.5.2.d) Solliciteur
3.1.2.5.2.e) Secteur
3.1.2.5.2.f) Référence
3.1.2.5.2.g) Commentaires
3.1.2.5.3. Informations spécifiques
3.1.2.5.3.a) Date de naissance
3.1.2.5.3.b) Date d'inscription
3.1.2.5.3.c) Anonymat
3.1.2.5.3.d) Activité
3.1.2.5.3.e) Bénéficiaire
3.1.2.5.3.f) Reçus et appels
3.1.2.5.3.g) Catégories de clients
3.1.2.5.3.h) Liens
3.1.2.5.3.i) Caractéristiques
3.1.2.6. Retirer un client
3.1.2.6.1. Confirmation
3.1.3. Gestion des occasions de dons
3.1.3.1. Consulter la liste des occasions de dons
3.1.3.1.1. Navigation
3.1.3.1.2. Code
3.1.3.1.3. Description
3.1.3.1.4. Code organisation
3.1.3.1.5. Nom organisation
3.1.3.2. Chercher une occasion de dons
3.1.3.2.1. Code
3.1.3.2.2. Description
3.1.3.2.3. Code organisation
3.1.3.2.4. Nom organisation
3.1.3.3. Consulter une occasion de dons
3.1.3.3.1. Informations générales
3.1.3.3.1.a) Code
3.1.3.3.1.b) Description
3.1.3.3.1.c) Présence d'une activité
3.1.3.3.1.d) Visibilité lors de l'ajout d'une transaction
3.1.3.3.1.e) Documents à émettre
3.1.3.3.2. Consulter une activité
3.1.3.3.2.a) Code
3.1.3.3.2.b) Description
3.1.3.3.2.c) Type
3.1.3.3.2.d) Documents à émettre
3.1.3.3.2.e) Provenance

3.1.3.3.3. Consulter une dépense
3.1.3.3.3.a) Code
3.1.3.3.3.b) Nom
3.1.3.4. Ajouter une occasion de dons
3.1.3.4.1. Informations générales
3.1.3.4.1.a) Code
3.1.3.4.1.b) Description
3.1.3.4.1.c) Présence d'une activité
3.1.3.4.1.d) Visibilité lors de l'ajout d'une transaction
3.1.3.4.1.e) Documents à émettre
3.1.3.4.2. Ajouter une activité
3.1.3.4.2.a) Code
3.1.3.4.2.b) Description
3.1.3.4.2.c) Type
3.1.3.4.2.d) Documents à émettre
3.1.3.4.2.e) Provenance
3.1.3.4.3. Ajouter une dépense
3.1.3.4.3.a) Code
3.1.3.4.3.b) Nom
3.1.3.5. Modifier une occasion de dons
3.1.3.5.1. Informations générales
3.1.3.5.1.a) Code
3.1.3.5.1.b) Description
3.1.3.5.1.c) Présence d'une activité
3.1.3.5.1.d) Visibilité lors de l'ajout d'une transaction
3.1.3.5.1.e) Documents à émettre
3.1.3.5.2. Modifier une activité
3.1.3.5.2.a) Code
3.1.3.5.2.b) Description
3.1.3.5.2.c) Type
3.1.3.5.2.d) Documents à émettre
3.1.3.5.2.e) Provenance
3.1.3.5.3. Modifier une dépense
3.1.3.5.3.a) Code
3.1.3.5.3.b) Nom
3.1.3.6. Retirer une occasion de dons
3.1.3.6.1. Confirmation
3.1.4. Gestion des transactions de dons
3.1.4.1. Consulter la liste des transactions de dons
3.1.4.1.1. Navigation
3.1.4.1.2. Client
3.1.4.1.3. Projet
3.1.4.1.4. Date
3.1.4.1.5. Occasion de dons
3.1.4.1.6. Montant
3.1.4.1.7. Mode de paiement
3.1.4.1.8. Documents
3.1.4.1.9. Notes
3.1.4.1.10. Solliciteur
3.1.4.2. Chercher une transaction de dons
3.1.4.2.1. Client
3.1.4.2.2. Projet
3.1.4.2.3. Date
3.1.4.2.4. Occasion de dons
3.1.4.2.5. Montant
3.1.4.2.6. Mode de paiement
3.1.4.2.7. Documents
3.1.4.2.8. Notes
3.1.4.2.9. Solliciteur
3.1.4.3. Consulter une transaction de dons
3.1.4.3.1. Client
3.1.4.3.2. Projet
3.1.4.3.3. Date
3.1.4.3.4. Occasion de dons
3.1.4.3.5. Montant
3.1.4.3.6. Mode de paiement
3.1.4.3.7. Documents
3.1.4.3.8. Notes
3.1.4.3.9. Solliciteur

- 3.1.4.4. Ajouter une transaction de dons
  - 3.1.4.4.1. Chercher un client
    - 3.1.4.4.1.a) Informations générales
    - 3.1.4.4.1.b) Informations complémentaires
    - 3.1.4.4.1.c) Informations spécifiques
  - 3.1.4.4.2. Informations générales
    - 3.1.4.4.2.a) Projet
    - 3.1.4.4.2.b) Date
    - 3.1.4.4.2.c) Occasion de dons
    - 3.1.4.4.2.d) Montant
    - 3.1.4.4.2.e) Mode de paiement
    - 3.1.4.4.2.f) Documents
    - 3.1.4.4.2.g) Notes
    - 3.1.4.4.2.h) Solliciteur
- 3.1.4.5. Modifier une transaction de dons
  - 3.1.4.5.1. Modifier un client
    - 3.1.4.5.1.a) Informations générales
    - 3.1.4.5.1.b) Informations complémentaires
    - 3.1.4.5.1.c) Informations spécifiques
  - 3.1.4.5.2. Informations générales
    - 3.1.4.5.2.a) Projet
    - 3.1.4.5.2.b) Date
    - 3.1.4.5.2.c) Occasion de dons
    - 3.1.4.5.2.d) Montant
    - 3.1.4.5.2.e) Mode de paiement
    - 3.1.4.5.2.f) Documents
    - 3.1.4.5.2.g) Notes
    - 3.1.4.5.2.h) Solliciteur
- 3.1.4.6. Retirer une transaction de dons
  - 3.1.4.6.1. Confirmation

3.2. Contraintes

- 3.2.1. Performance
  - 3.2.1.1. Statiques
  - 3.2.1.2. Dynamiques
    - 3.2.1.2.1. Recherche dans la BD
      - 3.2.1.2.1.a) BD restreinte
      - 3.2.1.2.1.b) BD volumineuse
- 3.2.2. Conception
  - 3.2.2.1. Conformité aux normes
  - 3.2.2.2. Limites matérielles
  - 3.2.2.3. Limites logicielles
    - 3.2.2.3.1. Système d'exploitation
    - 3.2.2.3.2. Gestionnaire de BD
  - 3.2.2.4. Politiques d'intégrité des données
- 3.2.3. Attributs
  - 3.2.3.1. Disponibilité
    - 3.2.3.1.1. Accès à distance
  - 3.2.3.2. Sécurité
    - 3.2.3.2.1. Gestion des droits d'accès
      - 3.2.3.2.1.a) Définir les accès
  - 3.2.3.3. Maintenabilité
    - 3.2.3.3.1. Documentation disponible
      - 3.2.3.3.1.a) Spécification d'exigences de système
    - 3.2.3.3.2. Implémentation de la traçabilité
    - 3.2.3.3.3. Ajout-modification-retrait d'une fonctionnalité
  - 3.2.3.4. Portabilité

3.3. Autres exigences

Annexe B

Spécification des exigences de ProDon

1. EXIGENCES SPÉCIFIQUES

ProDon doit permettre de gérer les dons et les campagnes de sollicitation de manière simple et conviviale.

Il en découle les exigences suivantes :

1) Exigences fonctionnelles
2) Contraintes
3) Autres exigences

Info. complémentaires :
- No. de l'exigence : 0
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1. Exigences fonctionnelles

ProDon doit exécuter les fonctions requises pour permettre de gérer les dons et les campagnes de sollicitation.

Il en découle les exigences suivantes :

1) Gestion générale
2) Gestion des clients
3) Gestion des occasions de dons
4) Gestion des transactions de dons

Info. complémentaires :
- No. de l'exigence : 146
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.1. Gestion générale

ProDon doit offrir des fonctions de gestion générale telle la gestion de projets, la gestion de la configuration, etc.

Il en découle les exigences suivantes :

1) Gestion des projets

Info. complémentaires :
- No. de l'exigence : 1868
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.1.1. Gestion des projets

ProDon doit offrir des fonctions de gestion spécifiques aux projets telle la consultation, la création, etc.

Info. complémentaires :
- No. de l'exigence : 1870
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion générale
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.1.1.1. Consulter la liste des projets

ProDon doit permettre de consulter la liste des projets déjà créés :

a) Navigation. Permettre la navigation dans la liste des projets ;

b) Code des projets. Permettre d'avoir un aperçu des codes des projets ;

c) Titre des projets. Permettre d'avoir un aperçu des titres des projets ;

d) Description des projets. Permettre d'avoir un aperçu des descriptions des projets ;

e) Client des projets. Permettre d'avoir un aperçu du nom des clients des projets ;

f) Individu des projets. Permettre d'avoir un aperçu du nom des individus des projets ;

g) Total de chaque projet. Permettre d'avoir un aperçu des montants totaux de chaque projet.

1.1.1.1.2. Chercher un projet [C]

ProDon doit permettre de rechercher un projet en introduisant certaines de ses informations, en tout ou en partie :

a) Code du projet [C]. Permettre d'introduire le code ou une partie du code du projet ;

b) Titre du projet [C]. Permettre d'introduire le titre ou une partie du titre du projet ;

c) Description du projet [C]. Permettre d'introduire la description ou une partie de la description du projet ;

d) Client du projet [C]. Permettre d'introduire le nom du client ou une partie du nom du client du projet ;

e) Individu du projet [C]. Permettre d'introduire le nom de l'individu ou une partie du nom de l'individu du projet ;

f) Total du projet [C]. Permettre d'introduire le montant total ou une partie du montant total du projet.

1.1.1.1.3. Consulter un projet

ProDon doit permettre de consulter en détail les différentes informations d'un projet déjà créé :

a) Code du projet. Permettre de consulter le code du projet ;

b) Titre du projet. Permettre de consulter le titre du projet ;

c) Début du projet. Permettre de consulter la date de début du projet ;

d) Fin du projet. Permettre de consulter la date de fin du projet ;

e) Client du projet. Permettre de consulter le nom du client du projet ;

f) Description du projet. Permettre de consulter la description du projet ;

g) Total du projet. Permettre de consulter le montant total du projet ;

h) Dons reçus du projet. Permettre de consulter en détail l'ensemble des dons reçus du projet ;

i) Dons émis du projet. Permettre de consulter en détail l'ensemble des dons émis du projet.

1.1.1.1.4. Ajouter un projet

ProDon doit permettre d'ajouter un projet en spécifiant son code ainsi qu'éventuellement ses autres informations :

a) Code du projet. Demander de spécifier le code du projet ;

b) Titre du projet. Permettre de spécifier le titre du projet ;

c) Début du projet. Permettre de spécifier la date de début du projet ;

d) Fin du projet. Permettre de spécifier la date de fin du projet ;

e) Client du projet. Permettre de spécifier le nom du client du projet ;

f) Description du projet. Permettre de spécifier la description du projet.

1.1.1.1.5. Modifier un projet

ProDon doit permettre de modifier les différentes informations d'un projet déjà créé :

a) Code du projet. Permettre de modifier le code du projet ;

b) Titre du projet. Permettre de modifier le titre du projet ;

c) Début du projet. Permettre de modifier la date de début du projet ;

d) Fin du projet. Permettre de modifier la date de fin du projet ;

e) Client du projet. Permettre de modifier le nom du client du projet ;

f) Description du projet. Permettre de modifier la description du projet.

1.1.1.1.6. Retirer un projet

ProDon doit permettre de supprimer un projet ainsi que toutes ses informations :

a) Confirmation. Demander confirmation avant suppression du projet et de ses caractéristiques.

1.1.2. Gestion des clients

ProDon doit offrir des fonctions de gestion spécifiques aux clients telle la consultation, la création, etc.

Il en découle les exigences suivantes :

1) Consulter la liste des clients
2) Chercher un client
3) Consulter un client
4) Ajouter un client
5) Modifier un client
6) Retirer un client

Info. complémentaires :
- No. de l'exigence : 5110
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.2.1. Consulter la liste des clients

ProDon doit permettre de consulter la liste des clients déjà créés.

Info. complémentaires :
- No. de l'exigence : 5196
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des clients
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.2.1.1. Navigation

ProDon doit permettre la navigation dans la liste des clients.

Info. complémentaires :
- No. de l'exigence : 5453
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des clients
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.2.1.2. Informations générales

ProDon doit permettre d'avoir un aperçu des informations générales du client tel son nom, son prénom, son numero de client, etc :

a) NUC. Permettre d'avoir un aperçu du numero du client ;

b) Prénom. Permettre d'avoir un aperçu du prénom du client ;

c) Nom. Permettre d'avoir un aperçu du nom du client ;

d) Appellation. Permettre d'avoir un aperçu de l'appellation du client ;

e) Titre. Permettre d'avoir un aperçu du titre du client ;

f) Compagnie. Permettre d'avoir un aperçu de la compagnie du client ;

g) Adresse. Permettre d'avoir un aperçu de l'adresse du client ;

h) Langue. Permettre d'avoir un aperçu de la langue du client.

1.1.2.1.3. Informations complémentaires

ProDon doit permettre d'avoir un aperçu des informations complémentaires du client tels ses numéros de téléphone, ses adresses courriel, etc :

a) Téléphones. Permettre d'avoir un aperçu des numéros de téléphone du client ;

b) Courriels. Permettre d'avoir un aperçu des adresses courriel du client ;

c) Site Internet. Permettre d'avoir un aperçu du site Internet du client ;

d) Solliciteur. Permettre d'avoir un aperçu du solliciteur du client ;

e) Secteur. Permettre d'avoir un aperçu du secteur du client ;

f) Référence. Permettre d'avoir un aperçu de la référence du client ;

g) Commentaires. Permettre d'avoir un aperçu des commentaires sur le client.

1.1.2.1.4. Informations spécifiques

ProDon doit permettre d'avoir un aperçu des informations spécifiques du client telle sa date de naissance, sa date d'inscription, etc :

a) Date de naissance. Permettre d'avoir un aperçu de la date de naissance du client ;

b) Date d'inscription. Permettre d'avoir un aperçu de la date d'inscription du client ;

c) Anonymat. Permettre d'afficher si le client veut rester anonyme ou non ;

d) Bénéficiaire. Permettre d'afficher si le client est un bénéficiaire ou non ;

e) Reçus et appels. Permettre d'afficher la façon dont les reçus et appels seront envoyés au client.

1.1.2.2. Chercher un client [C]

ProDon doit permettre de rechercher un client en introduisant certaines de ses caractéristiques, en tout ou en partie.

NOTE – Cette exigence est identique à celle qu'elle référence dans le renvoi excepté sa priorité.

Info. complémentaires :
- No. de l'exigence : 5206
- Réf. interne : 1.1.4.4.1
- Exig. parent niv. 3 : Gestion des clients
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.2.2.1. Informations générales [C]

ProDon doit permettre d'introduire, en tout ou en partie, des informations générales du client tel son nom, son prénom, son numero de client, etc :

a) NUC [C]. Permettre d'introduire le numéro ou une partie du numero du client ;

b) Prénom [C]. Permettre d'introduire le prénom ou une partie du prénom du client ;

c) Nom [C]. Permettre d'introduire le nom ou une partie du nom du client ;

d) Appellation [C]. Permettre d'introduire l'appellation ou une partie de l'appellation du client ;

e) Titre [C]. Permettre d'introduire le titre ou une partie du titre du client ;

f) Compagnie [C]. Permettre d'introduire la compagnie ou une partie de la compagnie du client ;

g) Adresse [C]. Permettre d'introduire l'adresse ou une partie de l'adresse du client ;

h) Langue [C]. Permettre d'introduire la langue ou une partie de la langue du client.

1.1.2.2.2. Informations complémentaires [C]

ProDon doit permettre d'introduire, en tout ou en partie, des informations complémentaires du client tels ses numéros de téléphone, ses adresses courriel, etc :

a) Téléphones [C]. Permettre d'introduire un numéro de téléphone ou une partie d'un numéro de téléphone du client ;

b) Courriels [C]. Permettre d'introduire une adresse courriel ou une partie d'une adresse courriel du client ;

c) Site Internet [C]. Permettre d'introduire le site Internet ou une partie du site Internet du client ;

d) Solliciteur [C]. Permettre d'introduire le solliciteur ou une partie du solliciteur du client ;

e) Secteur [C]. Permettre d'introduire le secteur ou une partie du secteur du client ;

f) Référence [C]. Permettre d'introduire la référence ou une partie de la référence du client ;

g) Commentaires [C]. Permettre d'introduire les commentaires ou une partie des commentaires sur le client.

1.1.2.2.3. Informations spécifiques [C]

ProDon doit permettre d'introduire, en tout ou en partie, des informations spécifiques du client telle sa date de naissance, sa date d'inscription, etc :

a) Date de naissance [C]. Permettre d'introduire la date de naissance ou une partie de la date de naissance du client ;

b) Date d'inscription [C]. Permettre d'introduire la date d'inscription ou une partie de la date d'inscription du client ;

c) Anonymat [C]. Permettre de spécifier si le client veut rester anonyme ou non ;

d) Bénéficiaire [C]. Permettre de spécifier si le client est un bénéficiaire ou non ;

e) Reçus et appels [C]. Permettre de spécifier la façon dont les reçus et appels sont envoyés au client.

1.1.2.3. Consulter un client

ProDon doit permettre de consulter en détail les différentes informations d'un client déjà créé.

Info. complémentaires :
- No. de l'exigence : 5398
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des clients
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.2.3.1. Informations générales

ProDon doit permettre de consulter en détail les informations générales du client tel son nom, son prénom, son adresse, etc :

a) Appellation. Permettre de consulter l'appellation du client ;

b) Genre. Permettre de consulter le genre du client ;

c) Prénom. Permettre de consulter le prénom du client ;

d) Nom. Permettre de consulter le nom du client ;

e) Titre. Permettre de consulter le titre du client ;

f) Compagnie. Permettre de consulter la compagnie du client ;

g) Adresse. Permettre de consulter l'adresse du client ;

h) Langue. Permettre de consulter la langue du client.

1.1.2.3.2. Informations complémentaires

ProDon doit permettre de consulter en détail les informations complémentaires du client tels ses numéros de téléphone, ses adresses courriel, etc :

a) Téléphones. Permettre de consulter les numéros de téléphone du client ;

b) Courriels. Permettre de consulter les adresses courriel du client ;

c) Site Internet. Permettre de consulter le site Internet du client ;

d) Solliciteur. Permettre de consulter le solliciteur du client ;

e) Secteur. Permettre de consulter le secteur du client ;

f) Référence. Permettre de consulter la référence du client ;

g) Commentaires. Permettre de consulter les commentaires sur le client.

1.1.2.3.3. Informations spécifiques

ProDon doit permettre de consulter en détail les informations spécifiques du client telle sa date de naissance, sa date d'inscription, etc :

a) Date de naissance. Permettre de consulter la date de naissance du client ;

b) Date d'inscription. Permettre de consulter la date d'inscription du client ;

c) Anonymat. Permettre de consulter si le client désire demeurer anonyme ou non ;

d) Activité. Permettre de consulter si le client est inactif ou non ;

e) Bénéficiaire. Permettre de consulter si le client est un bénéficiaire ou non ;

f) Reçus et appels. Permettre de consulter la façon dont les reçus et appels sont envoyés au client ;

g) Catégories de clients. Permettre de consulter si le client est associé à des catégories de clients ;

h) Liens. Permettre de consulter si ce client est lié à d'autres clients ;

i) Caractéristiques. Permettre de consulter les caractéristiques du client.

1.1.2.4. Ajouter un client

ProDon doit permettre d'ajouter un client en spécifiant son nom ou prénom ainsi qu'éventuellement ses autres informations.

Info. complémentaires :
- No. de l'exigence : 5236
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des clients
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.2.4.1. Informations générales

ProDon doit permettre d'ajouter les informations générales du client tel son nom, son prénom, son adresse, etc :

a) Appellation. Permettre de spécifier l'appellation du client ;

b) Genre. Permettre de spécifier le genre du client ;

c) Prénom. Demander de spécifier le prénom du client ;

d) Nom. Demander de spécifier le nom du client ;

e) Titre. Permettre de spécifier le titre du client ;

f) Compagnie. Permettre de spécifier la compagnie du client ;

g) Adresse. Permettre de spécifier l'adresse du client ;

h) Langue. Permettre de spécifier la langue du client.

1.1.2.4.2. Informations complémentaires

ProDon doit permettre d'ajouter les informations complémentaires du client tels ses numéros de téléphone, ses adresses courriel, etc :

a) Téléphones. Permettre de spécifier les numéros de téléphone du client ;

b) Courriels. Permettre de spécifier les adresses courriel du client ;

c) Site Internet. Permettre de spécifier le site Internet du client ;

d) Solliciteur. Permettre de spécifier le solliciteur du client ;

e) Secteur. Permettre de spécifier le secteur du client ;

f) Référence. Permettre de spécifier la référence du client ;

g) Commentaires. Permettre de spécifier des commentaires sur le client.

1.1.2.4.3. Informations spécifiques

ProDon doit permettre d'ajouter les informations spécifiques du client telle sa date de naissance, sa date d'inscription, etc :

a) Date de naissance. Permettre de spécifier la date de naissance du client ;

b) Date d'inscription. Permettre de spécifier la date d'inscription du client ;

c) Anonymat. Permettre de spécifier si le client désire demeurer anonyme ou non ;

d) Activité. Permettre de spécifier si le client est inactif ou non ;

e) Bénéficiaire. Permettre de spécifier si le client est un bénéficiaire ou non ;

f) Reçus et appels. Permettre de spécifier la façon dont les reçus et appels seront envoyés au client ;

g) Catégories de clients. Permettre d'associer le client à des catégories de clients ;

h) Liens. Permettre de créer des liens avec d'autres clients ;

i) Caractéristiques. Permettre d'ajouter des caractéristiques au client.

1.1.2.5. Modifier un client

ProDon doit permettre de modifier les différentes informations d'un client déjà créé.

NOTE – Cette exigence est identique à celle qu'elle référence dans le renvoi.

Info. complémentaires :
- No. de l'exigence : 5283
- Réf. interne : 1.1.4.5.1
- Exig. parent niv. 3 : Gestion des clients
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.2.5.1. Informations générales

ProDon doit permettre de modifier les informations générales du client tel son nom, son prénom, son adresse, etc :

a) Appellation. Permettre de modifier l'appellation du client ;

b) Genre. Permettre de modifier le genre du client ;

c) Prénom. Permettre de modifier le prénom du client ;

d) Nom. Permettre de modifier le nom du client ;

e) Titre. Permettre de modifier le titre du client ;

f) Compagnie. Permettre de modifier la compagnie du client ;

g) Adresse. Permettre de modifier l'adresse du client ;

h) Langue. Permettre de modifier la langue du client.

1.1.2.5.2. Informations complémentaires

ProDon doit permettre de modifier les informations complémentaires du client tels ses numéros de téléphone, ses adresses courriel, etc :

a) Téléphones. Permettre de modifier les numéros de téléphone du client ;

b) Courriels. Permettre de modifier les adresses courriel du client ;

c) Site Internet. Permettre de modifier le site Internet du client ;

d) Solliciteur. Permettre de modifier le solliciteur du client ;

e) Secteur. Permettre de modifier le secteur du client ;

f) Référence. Permettre de modifier la référence du client ;

g) Commentaires. Permettre de modifier les commentaires sur le client.

1.1.2.5.3. Informations spécifiques

ProDon doit permettre de modifier les informations spécifiques du client telle sa date de naissance, sa date d'inscription, etc :

a) Date de naissance. Permettre de modifier la date de naissance du client ;

b) Date d'inscription. Permettre de modifier la date d'inscription du client ;

c) Anonymat. Permettre de modifier le fait que le client désire demeurer anonyme ou non ;

d) Activité. Permettre de modifier le fait qu'un client est inactif ou non ;

e) Bénéficiaire. Permettre de modifier le fait qu'un client est bénéficiaire ou non ;

f) Reçus et appels. Permettre de modifier la façon dont les reçus et appels seront envoyés au client ;

g) Catégories de clients. Permettre de modifier les catégories du client ;

h) Liens. Permettre de modifier les liens du client avec d'autres clients ;

i) Caractéristiques. Permettre de modifier les caractéristiques du client.

1.1.2.6. Retirer un client

ProDon doit permettre de supprimer un client ainsi que toutes ses informations.

Info. complémentaires :
- No. de l'exigence : 5341
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des clients
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.2.6.1. Confirmation

ProDon doit demander confirmation avant suppression du client et de ses informations.

Info. complémentaires :
- No. de l'exigence : 5397
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des clients
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3. Gestion des occasions de dons

ProDon doit offrir des fonctions de gestion spécifiques aux occasions de dons telle la consultation, la création, etc.

Il en découle les exigences suivantes :

1) Consulter la liste des occasions de dons
2) Chercher une occasion de dons
3) Consulter une occasion de dons
4) Ajouter une occasion de dons
5) Modifier une occasion de dons
6) Retirer une occasion de dons

Info. complémentaires :
- No. de l'exigence : 5614
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.1. Consulter la liste des occasions de dons

ProDon doit permettre de consulter la liste des occasions de dons déjà créées.

Info. complémentaires :
- No. de l'exigence : 5615
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.1.1. Navigation

ProDon doit permettre la navigation dans la liste des occasions de dons.

Info. complémentaires :
- No. de l'exigence : 5618
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.1.2. Code

ProDon doit permettre d'avoir un aperçu du code de l'occasion de dons.

Info. complémentaires :
- No. de l'exigence : 6424
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.1.3. Description

ProDon doit permettre d'avoir un aperçu de la description de l'occasion de dons.

Info. complémentaires :
- No. de l'exigence : 6425
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.1.4. Code organisation

ProDon doit permettre d'avoir un aperçu du code de l'organisation de l'occasion de dons.

Info. complémentaires :
- No. de l'exigence : 6427
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.1.5. Nom organisation

ProDon doit permettre d'avoir un aperçu du nom de l'organisation de l'occasion de dons.

Info. complémentaires :
- No. de l'exigence : 6426
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.2. Chercher une occasion de dons [C]

ProDon doit permettre de rechercher une occasion de dons en introduisant certaines de ses informations, en tout ou en partie.

Info. complémentaires :
- No. de l'exigence : 5625
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.3.2.1. Code [C]

ProDon doit permettre d'introduire le code ou une partie du code de l'occasion de dons.

Info. complémentaires :
- No. de l'exigence : 6444
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.3.2.2. Description [C]

ProDon doit permettre d'introduire la description ou une partie de la description de l'occasion de dons.

Info. complémentaires :
- No. de l'exigence : 6445
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.3.2.3. Code organisation [C]

ProDon doit permettre d'introduire le code de l'organisation ou une partie du code de l'organisation de l'occasion de dons.

Info. complémentaires :
- No. de l'exigence : 6446
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.3.2.4. Nom organisation [C]

ProDon doit permettre d'introduire le nom de l'organisation ou une partie du nom de l'organisation de l'occasion de dons.

Info. complémentaires :
- No. de l'exigence : 6447
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.3.3. Consulter une occasion de dons

ProDon doit permettre de consulter en détail les différentes informations d'une occasion de dons déjà créée.

Info. complémentaires :
- No. de l'exigence : 5640
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.3.1. Informations générales

ProDon doit permettre de consulter les informations générales de l'occasion de dons tel son code, sa description, etc :

a) Code. Permettre de consulter le code de l'occasion de dons ;

b) Description. Permettre de consulter la description de l'occasion de dons ;

c) Présence d'une activité. Permettre de vérifier si une activité doit être ajoutée ou non lorsque l'on effectue un don dans cette occasion ;

d) Visibilité lors de l'ajout d'une transaction. Permettre de vérifier si l'occasion de dons doit être visible lors de l'ajout d'une transaction ou non ;

e) Documents à émettre. Permettre de consulter les documents par défaut à émettre.

1.1.3.3.2. Consulter une activité

ProDon doit permettre de consulter une activité d'une occasion de dons :

a) Code. Permettre de consulter le code de l'activité ;

b) Description. Permettre de consulter la description de l'activité ;

c) Type. Permettre de consulter le type de l'activité ;

d) Documents à émettre. Permettre de consulter les documents par défaut à émettre ;

e) Provenance. Permettre de consulter la provenance de l'activité.

1.1.3.3.3. Consulter une dépense

ProDon doit permettre de consulter une dépense d'une occasion de dons :

a) Code. Permettre de consulter le code de la dépense ;

b) Nom. Permettre de consulter le nom de la dépense.

1.1.3.4. Ajouter une occasion de dons

ProDon doit permettre d'ajouter une occasion de dons en spécifiant son code ainsi qu'éventuellement ses autres informations.

Info. complémentaires :
- No. de l'exigence : 6207
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.4.1. Informations générales

ProDon doit permettre d'ajouter les informations générales de l'occasion de dons tel son code, sa description, etc :

a) Code. Demander de spécifier le code de l'occasion de dons ;

b) Description. Permettre de spécifier la description de l'occasion de dons ;

c) Présence d'une activité. Permettre de spécifier si une activité doit être ajoutée ou non lorsque l'on effectue un don dans cette occasion ;

d) Visibilité lors de l'ajout d'une transaction. Permettre de spécifier si l'occasion de dons doit être visible lors de l'ajout d'une transaction ou non ;

e) Documents à émettre. Permettre de spécifier les documents par défaut à émettre.

1.1.3.4.2. Ajouter une activité

ProDon doit permettre d'ajouter une activité à une occasion de dons :

a) Code. Permettre de spécifier le code de l'activité ;

b) Description. Demander de spécifier la description de l'activité ;

c) Type. Permettre de spécifier le type de l'activité ;

d) Documents à émettre. Permettre de spécifier les documents par défaut à émettre ;

e) Provenance. Demander de spécifier la provenance de l'activité.

1.1.3.4.3. Ajouter une dépense

ProDon doit permettre d'ajouter une dépense à une occasion de dons :

a) Code. Permettre de spécifier le code de la dépense ;

b) Nom. Demander de spécifier le nom de la dépense.

1.1.3.5. Modifier une occasion de dons

ProDon doit permettre de modifier les différentes informations d'une occasion de dons déjà créée.

Info. complémentaires :
- No. de l'exigence : 5821
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.5.1. Informations générales

ProDon doit permettre de modifier les informations générales de l'occasion de dons tel son code, sa description, etc :

a) Code. Permettre de modifier le code de l'occasion de dons ;

b) Description. Permettre de modifier la description de l'occasion de dons ;

c) Présence d'une activité. Permettre de modifier le fait qu'une activité doit être ajoutée ou non lorsque l'on effectue un don dans cette occasion ;

d) Visibilité lors de l'ajout d'une transaction. Permettre de modifier le fait qu'une occasion de dons doit être visible lors de l'ajout d'une transaction ou non ;

e) Documents à émettre. Permettre de modifier les documents par défaut à émettre.

1.1.3.5.2. Modifier une activité

ProDon doit permettre de modifier une activité d'une occasion de dons :

a) Code. Permettre de modifier le code de l'activité ;

b) Description. Permettre de modifier la description de l'activité ;

c) Type. Permettre de modifier le type de l'activité ;

d) Documents à émettre. Permettre de modifier les documents par défaut à émettre ;

e) Provenance. Permettre de modifier la provenance de l'activité.

1.1.3.5.3. Modifier une dépense

ProDon doit permettre de modifier une dépense d'une occasion de dons :

a) Code. Permettre de modifier le code de la dépense ;

b) Nom. Permettre de modifier le nom de la dépense.

1.1.3.6. Retirer une occasion de dons

ProDon doit permettre de supprimer une occasion de dons ainsi que toutes ses informations.

Info. complémentaires :
- No. de l'exigence : 6150
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.3.6.1. Confirmation

ProDon doit demander confirmation avant suppression de l'occasion de dons et de ses informations.

Info. complémentaires :
- No. de l'exigence : 6206
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des occasions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4. Gestion des transactions de dons

ProDon doit offrir des fonctions de gestion spécifiques aux transactions de dons telle la consultation, la création, etc.

Il en découle les exigences suivantes :

1) Consulter la liste des transactions de dons
2) Chercher une transaction de dons
3) Consulter une transaction de dons
4) Ajouter une transaction de dons
5) Modifier une transaction de dons
6) Retirer une transaction de dons

Info. complémentaires :
- No. de l'exigence : 1861
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1. Consulter la liste des transactions de dons

ProDon doit permettre de consulter la liste des transactions de dons déjà créées.

Info. complémentaires :
- No. de l'exigence : 6625
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.1. Navigation

ProDon doit permettre la navigation dans la liste des transactions de dons.

Info. complémentaires :
- No. de l'exigence : 6628
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.2. Client

ProDon doit permettre d'avoir un aperçu du client lié à la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10619
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.3. Projet

ProDon doit permettre d'avoir un aperçu du projet lié à la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10603
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.4. Date

ProDon doit permettre d'avoir un aperçu de la date de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10604
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.5. Occasion de dons

ProDon doit permettre d'avoir un aperçu de l'occasion de dons liée à la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10605
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.6. Montant

ProDon doit permettre d'avoir un aperçu du montant de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10606
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.7. Mode de paiement

ProDon doit permettre d'avoir un aperçu du mode de paiement de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10607
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.8. Documents

ProDon doit permettre d'avoir un aperçu des documents liés à la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10608
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.9. Notes

ProDon doit permettre d'avoir un aperçu des notes sur la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10609
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.1.10. Solliciteur

ProDon doit permettre d'avoir un aperçu du solliciteur de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10610
- Degré de nécessité : Essentielle

- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.2. Chercher une transaction de dons [C]

ProDon doit permettre de rechercher une transaction de dons en introduisant certaines de ses informations, en tout ou en partie.

Info. complémentaires :
- No. de l'exigence : 7438
- Degré de nécessité : Complémentaire
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.2.1. Client [C]

ProDon doit permettre d'introduire le client ou une partie du client de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10611
- Degré de nécessité : Complémentaire
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.2.2. Projet [C]

ProDon doit permettre d'introduire le projet ou une partie du projet de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10620
- Degré de nécessité : Complémentaire
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.2.3. Date [C]

ProDon doit permettre d'introduire la date ou une partie de la date de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10612
- Degré de nécessité : Complémentaire
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.2.4. Occasion de dons [C]

ProDon doit permettre d'introduire l'occasion de dons, ou une partie de l'occasion de dons, de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10613
- Degré de nécessité : Complémentaire
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.2.5. Montant [C]

ProDon doit permettre d'introduire le montant ou une partie du montant de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10614
- Degré de nécessité : Complémentaire
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.2.6. Mode de paiement [C]

ProDon doit permettre d'introduire le mode de paiement ou une partie du mode de paiement de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10615
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.4.2.7. Documents [C]

ProDon doit permettre d'introduire un document ou une partie d'un document de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10616
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.4.2.8. Notes [C]

ProDon doit permettre d'introduire une note ou une partie d'une note de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10617
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.4.2.9. Solliciteur [C]

ProDon doit permettre d'introduire le solliciteur ou une partie du solliciteur de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10618
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Complémentaire
- Réf. besoin client : s.o.

1.1.4.3. Consulter une transaction de dons

ProDon doit permettre de consulter en détail les différentes informations d'une transaction de dons déjà créée.

Info. complémentaires :
- No. de l'exigence : 8261
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.3.1. Client

ProDon doit permettre de consulter le client lié à la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10622
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.3.2. Projet

ProDon doit permettre de consulter le projet lié à la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10623
- Degré de nécessité : Essentielle
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.3.3. Date

ProDon doit permettre de consulter la date de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10624
- Degré de nécessité : Essentielle
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.3.4. Occasion de dons

ProDon doit permettre de consulter l'occasion de dons liée à la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10625
- Degré de nécessité : Essentielle
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.3.5. Montant

ProDon doit permettre de consulter le montant de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10627
- Degré de nécessité : Essentielle
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.3.6. Mode de paiement

ProDon doit permettre de consulter le mode de paiement de la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10621
- Degré de nécessité : Essentielle
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.3.7. Documents

ProDon doit permettre de consulter les documents liés à la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10628
- Degré de nécessité : Essentielle
- Réf. interne : s.o.
- Réf. besoin client : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons

1.1.4.3.8. Notes

ProDon doit permettre de consulter les notes liées à la transaction de dons.

Info. complémentaires :
- No. de l'exigence : 10629
- Degré de nécessité : Essentielle

– Réf. interne : s.o.
– Exig. parent niv. 3 : Gestion des transactions de dons
– Réf. besoin client : s.o.

1.1.4.3.9. Solliciteur

ProDon doit permettre de consulter le solliciteur lié à la transaction de dons.

Info. complémentaires :
– No. de l'exigence : 10626
– Réf. interne : s.o.
– Exig. parent niv. 3 : Gestion des transactions de dons
– Degré de nécessité : Essentielle
– Réf. besoin client : s.o.

1.1.4.4. Ajouter une transaction de dons

ProDon doit permettre d'ajouter une transaction de dons en spécifiant le client, l'occasion de dons, le montant et le mode de paiement ainsi qu'éventuellement ses autres informations.

Info. complémentaires :
– No. de l'exigence : 9157
– Réf. interne : s.o.
– Exig. parent niv. 3 : Gestion des transactions de dons
– Degré de nécessité : Essentielle
– Réf. besoin client : s.o.

1.1.4.4.1. Chercher un client

ProDon doit permettre de rechercher un client en introduisant certaines de ses caractéristiques, en tout ou en partie.

NOTE – Cette exigence est identique à celle qu'elle référence dans le renvoi excepté sa priorité.

a) Informations générales. ProDon doit permettre d'introduire, en tout ou en partie, des informations générales du client tel son nom, son prénom, son numero de client, etc ;

b) Informations complémentaires. ProDon doit permettre d'introduire, en tout ou en partie, des informations complémentaires du client tels ses numéros de téléphone, ses adresses courriel, etc ;

c) Informations spécifiques. ProDon doit permettre d'introduire, en tout ou en partie, des informations spécifiques du client telle sa date de naissance, sa date d'inscription, etc.

Info. complémentaires :
– No. de l'exigence : 10255
– Réf. interne : 1.1.2.2
– Exig. parent niv. 3 : Gestion des transactions de dons
– Degré de nécessité : Essentielle
– Réf. besoin client : s.o.

1.1.4.4.2. Informations générales

ProDon doit permettre de spécifier les informations générales de la transaction de dons tel son montant, son occasion de dons, son mode de paiement, etc :

a) Projet. Permettre de spécifier le projet lié à la transaction de dons ;

b) Date. Permettre de spécifier la date de la transaction de dons ;

c) Occasion de dons. Demander de spécifier l'occasion de dons de la transaction de dons ;

d) Montant. Demander de spécifier le montant de la transaction de dons ;

e) Mode de paiement. Demander de spécifier le mode de paiement de la transaction de dons ;

f) Documents. Permettre de spécifier les documents liés à la transaction de dons ;

g) Notes. Permettre de spécifier des notes sur la transaction de dons ;

h) Solliciteur. Permettre de spécifier le solliciteur de la transaction de dons.

1.1.4.5. Modifier une transaction de dons

ProDon doit permettre de modifier les différentes informations d'une transaction de dons déjà créée.

Info. complémentaires :
- No. de l'exigence : 9394
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.5.1. Modifier un client

ProDon doit permettre de modifier les différentes informations d'un client déjà créé.

NOTE – Cette exigence est identique à celle qu'elle référence dans le renvoi.

a) Informations générales. ProDon doit permettre de modifier les informations générales du client tel son nom, son prénom, son adresse, etc ;

b) Informations complémentaires. ProDon doit permettre de modifier les informations complémentaires du client tels ses numéros de téléphone, ses adresses courriel, etc ;

c) Informations spécifiques. ProDon doit permettre de modifier les informations spécifiques du client telle sa date de naissance, sa date d'inscription, etc.

Info. complémentaires :
- No. de l'exigence : 10630
- Réf. interne : 1.1.2.5
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.5.2. Informations générales

ProDon doit permettre de modifier les informations générales de la transaction de dons tel son montant, son occasion de dons, son mode de paiement, etc :

a) Projet. Permettre de modifier le projet lié à la transaction de dons ;

b) Date. Permettre de modifier la date de la transaction de dons ;

c) Occasion de dons. Permettre de modifier l'occasion de dons de la transaction de dons ;

d) Montant. Permettre de modifier le montant de la transaction de dons ;

e) Mode de paiement. Permettre de modifier le mode de paiement de la transaction de dons ;

f) Documents. Permettre de modifier les documents liés à la transaction de dons ;

g) Notes. Permettre de modifier les notes sur la transaction de dons ;

h) Solliciteur. Permettre de modifier le solliciteur de la transaction de dons.

1.1.4.6. Retirer une transaction de dons

ProDon doit permettre de supprimer une transaction de dons ainsi que toutes ses informations.

Info. complémentaires :
- No. de l'exigence : 10198
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.1.4.6.1. Confirmation

ProDon doit demander confirmation avant suppression de la transaction de dons et de ses informations.

Info. complémentaires :
- No. de l'exigence : 10254
- Réf. interne : s.o.
- Exig. parent niv. 3 : Gestion des transactions de dons
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2. Contraintes

ProDon doit respecter des contraintes de performance, de conception et d'attribut (disponibilité, sécurité, maintenabilité, etc).

Il en découle les exigences suivantes :

1) Performance
2) Conception
3) Attributs

Info. complémentaires :
- No. de l'exigence : 2178
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.1. Performance

ProDon doit satisfaire des exigences numériques.

Il en découle les exigences suivantes :

1) Statiques
2) Dynamiques

Info. complémentaires :
- No. de l'exigence : 2179
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.1.1. Statiques

ProDon doit satisfaire des exigences numériques statiques (ne variant pas selon la situation).

Info. complémentaires :
- No. de l'exigence : 2689
- Réf. interne : s.o.
- Exig. parent niv. 3 : Performance
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.1.2. Dynamiques

ProDon doit satisfaire des exigences numériques dynamiques (variant selon la situation), telle une recherche dans la base de données, dépendant du nombre d'objets présents.

Info. complémentaires :
- No. de l'exigence : 2690
- Réf. interne : s.o.
- Exig. parent niv. 3 : Performance
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.1.2.1. Recherche dans la BD

ProDon doit permettre de réaliser des recherches de manière performante dans la base de données :

a) BD restreinte. permettre de réaliser une recherche en maximum 3 secondes dans une base de données dont le nombre d'objets est inférieur ou égal à 2500 ;

Commentaire : RC : Le temps de recherche maximum devrait être réduit à 2 secondes. (À valider par C. Dubois)

b) BD volumineuse. permettre de réaliser une recherche en maximum 5 secondes dans une base de données dont le nombre d'objets est supérieur à 2500.

Info. complémentaires :
- No. de l'exigence : 3551
- Réf. interne : s.o.
- Exig. parent niv. 3 : Performance
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.2. Conception

ProDon doit satisfaire des contraintes de conception imposées par des normes, limites matérielles, etc.

Il en découle les exigences suivantes :

1) Conformité aux normes
2) Limites matérielles
3) Limites logicielles
4) Politiques d'intégrité des données

Info. complémentaires :
- No. de l'exigence : 3554
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.2.1. Conformité aux normes

Info. complémentaires :
- No. de l'exigence : 4061
- Réf. interne : s.o.
- Exig. parent niv. 3 : Conception
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.2.2. Limites matérielles

Info. complémentaires :
- No. de l'exigence : 4062
- Réf. interne : s.o.
- Exig. parent niv. 3 : Conception
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.2.3. Limites logicielles

ProDon doit satisfaire des contraintes logicielles telle l'utilisation d'un système d'exploitation spécifique et l'utilisation d'un gestionnaire de base de données spécifique.

Info. complémentaires :
- No. de l'exigence : 4588
- Réf. interne : s.o.
- Exig. parent niv. 3 : Conception
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.2.3.1. Système d'exploitation

ProDon doit permettre son installation sous Windows 98 ou supérieur.

Info. complémentaires :
- No. de l'exigence : 5098
- Réf. interne : s.o.
- Exig. parent niv. 3 : Conception
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.2.3.2. Gestionnaire de BD

ProDon doit permettre l'utilisation de Microsoft SQL Server comme gestionnaire de la base de données.

Info. complémentaires :
- No. de l'exigence : 5099
- Réf. interne : s.o.
- Exig. parent niv. 3 : Conception
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.2.4. Politiques d'intégrité des données

Info. complémentaires :
- No. de l'exigence : 4592
- Réf. interne : s.o.
- Exig. parent niv. 3 : Conception
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.3. Attributs

ProDon doit satisfaire des exigences de Disponibilité, Sécurité, Maintenabilité et Transférabilité.

Il en découle les exigences suivantes :

1) Disponibilité
2) Sécurité
3) Maintenabilité
4) Portabilité

Info. complémentaires :
- No. de l'exigence : 4593
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.3.1. Disponibilité

ProDon doit être disponible à distance et offrir une haute disponibilité sans toutefois nécessiter de la redondance.

Info. complémentaires :
- No. de l'exigence : 5022
- Réf. interne : s.o.
- Exig. parent niv. 3 : Attributs
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.3.1.1. Accès à distance

ProDon doit être disponible à distance à travers un réseau tel un intranet.

Info. complémentaires :
- No. de l'exigence : 5109
- Réf. interne : s.o.
- Exig. parent niv. 3 : Attributs
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.3.2. Sécurité

ProDon doit gérer sa sécurité d'utilisation et de fonctionnement, tel exiger un mot de passe pour son utilisation.

Info. complémentaires :
- No. de l'exigence : 4608
- Réf. interne : s.o.
- Exig. parent niv. 3 : Attributs
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

1.2.3.2.1. Gestion des droits d'accès

ProDon doit ne permettre l'accès à son interface utilisateur qu'au moyen d'un mot de passe, d'au moins cinq caractères, par utilisateur :

a) Définir les accès. Permettre de spécifier les accès autorisés d'un utilisateur pour chacune des fonctionnalités de ProDon.

Info. complémentaires :
- No. de l'exigence : 5100
- Réf. interne : s.o.
- Degré de nécessité : Essentielle
- Réf. besoin client : s.o.

– Exig. parent niv. 3 : Attributs

1.2.3.3. Maintenabilité

ProDon doit faciliter son entretien et son évolution, notamment en offrant une documentation minimale et en étant conçu de façon à faciliter l'ajout, la modification ou le retrait d'une fonctionnalité.

Info. complémentaires :
– No. de l'exigence : 5095
– Réf. interne : s.o.
– Exig. parent niv. 3 : Attributs
– Degré de nécessité : Essentielle
– Réf. besoin client : s.o.

1.2.3.3.1. Documentation disponible

ProDon doit offrir une documentation de son développement :

a) Spécification d'exigences de système. Offrir la spécification de ses exigences.

Info. complémentaires :
– No. de l'exigence : 5102
– Réf. interne : s.o.
– Exig. parent niv. 3 : Attributs
– Degré de nécessité : Essentielle
– Réf. besoin client : s.o.

1.2.3.3.2. Implémentation de la traçabilité

ProDon doit offrir un moyen de retracer, par son numéro, toute exigence de système.

Info. complémentaires :
– No. de l'exigence : 5107
– Réf. interne : s.o.
– Exig. parent niv. 3 : Attributs
– Degré de nécessité : Essentielle
– Réf. besoin client : s.o.

1.2.3.3.3. Ajout-modification-retrait d'une fonctionnalité

ProDon doit offrir la possibilité d'ajouter, modifier ou retirer une fonctionnalité.

Info. complémentaires :
– No. de l'exigence : 5108
– Réf. interne : s.o.
– Exig. parent niv. 3 : Attributs
– Degré de nécessité : Essentielle
– Réf. besoin client : s.o.

1.2.3.4. Portabilité

Info. complémentaires :
– No. de l'exigence : 5096
– Réf. interne : s.o.
– Exig. parent niv. 3 : Attributs
– Degré de nécessité : Essentielle
– Réf. besoin client : s.o.

1.3. Autres exigences [O]

Info. complémentaires :

- No. de l'exigence : 805
- Réf. interne : s.o.
- Exig. parent niv. 3 : s.o.
- Degré de nécessité : Optionnelle
- Réf. besoin client : s.o.

Bibliographie

[1] **Génie logiciel**, *Ingenierie des exigences*, Patrick Heymans, F.U.N.D.P.

[2] **GenSpec**, *Ingénierie des Exigences - Une méthode simple et systématique*, IEEE Canadian Review - Fall / Automne 2004, René Bujold, Hydro-Québec, Montréal, QC.

[3] **GenSpec**, *Ingénierie des Exigences - L'outil de support « GenSpec »*, Revue Canadienne IEEE, Automne 2005, René Bujold, Hydro-Québec, Montréal, QC.

[4] **SRS**, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE Std 830-1998(Revision of IEEE Std 830-1993).

[5] **Traçabilité**, *Toward Reference Models for Requirements Traceability*, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 27, NO. 1, JANUARY 2001.

[6] **Volere**, *Requirements Specification Template*, Edition 11, Atlantic Systems Guild.

Annexe D

Gestion des changements - Logilys

Département Génie Logiciel

Gestion des changements

Antoine Robaeys
antoine.robaeys@student.fundp.ac.be

15 Novembre 2006

École de Technologie Supérieure - 1100 rue Notre-Dame Ouest - Montréal
Tél. : (514) 396-8800 - Fax : (514) 396-8950

Table des matières

Introduction

Ce document a pour but de présenter un processus de gestion des changements qui permet de formaliser la manière dont les requêtes de changement, provenant des clients de Logilys, sont enregistrées, gérées et solutionnées par l'entreprise.

La première partie illustre chacune des étapes du processus, ainsi que le modèle du formulaire utilisé pour représenter une requête de changement d'un client.

La seconde partie représente l'analyse des besoins, en terme de gestion des changements, de l'entreprise Logilys. Cette analyse servira de base pour le développement de l'outil « modeX », qui facilitera la gestion des requêtes de changement chez Logilys, selon le processus formel présenté précédemment.

Première partie

Analyse du processus

Chapitre 1

Processus de gestion des changements

Le processus de gestion des changements, représenté à la figure 1.1, a pour but d'aider l'entreprise à gérer formellement les requêtes de changement provenant de leurs clients.

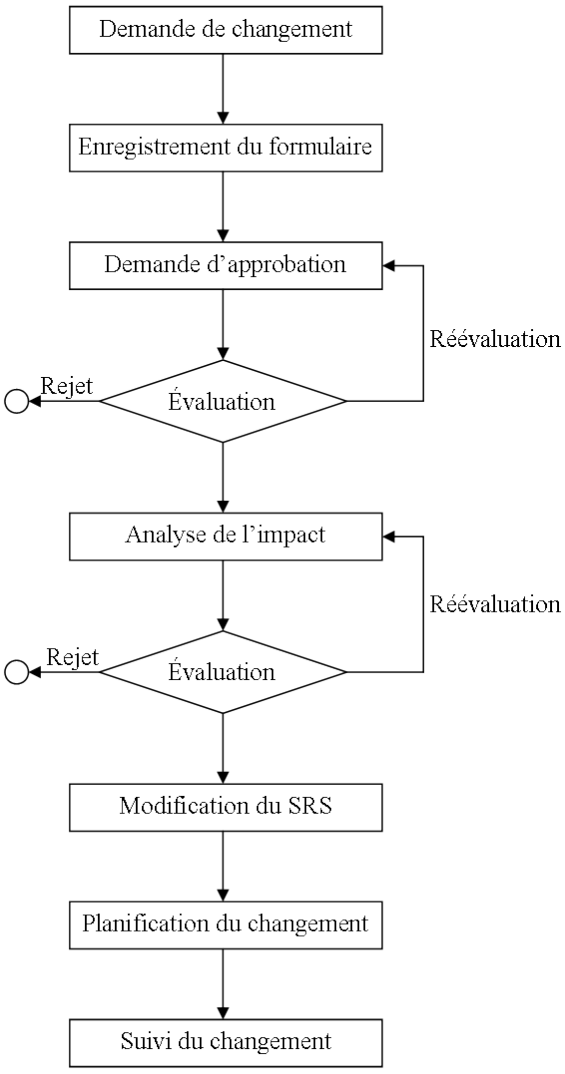

FIG. 1.1 – Processus de gestion des changements

Ci-dessous, la description de chacune des étapes du processus :

1.1 Demande de changement

Cette première étape consiste en une requête de la part du client qui souhaite voir un changement dans le logiciel qu'il utilise. Pour ce faire, le client contacte l'entreprise via l'un des points de contact, préalablement établi par cette entreprise pour permettre aux clients d'émettre leurs requêtes. Ces points de contact peuvent être un numéro de téléphone, un courriel, une adresse, etc.

1.2 Enregistrement du formulaire

Cette étape représente l'enregistrement de la requête de changement via un formulaire bien défini. La personne en charge du point de contact va réaliser et enregistrer une description détaillée et formelle du changement voulu par le client. Remarquons que le modèle de ce formulaire est décrit dans le chapitre suivant.

À la fin de cette étape, une requête de changement est enregistrée et est dans l'état « ENREGISTRÉE ».

1.3 Demande de validation

Cette première évaluation va permettre de décider s'il y a lieu ou non de réaliser la requête de changement. C'est le chef de projet qui est responsable d'évaluer la faisabilité du changement et de recontacter éventuellement le client pour de plus amples informations.

À la fin de cette étape, la requête de changement est dans l'état « VALIDÉE » ou « REJETÉE ».

1.4 Analyse de l'impact

Cette seconde évaluation va permettre de décider s'il y a lieu ou non de réaliser la requête de changement, préalablement validée, en fonction de son impact sur le système existant. Le chef de projet va donc étudier l'impact que le changement aurait sur les autres exigences et prendre en considération toutes les alternatives possibles à ce changement.

À la fin de cette étape, la requête de changement est dans l'état « APPROUVÉE » ou « REJETÉE ».

1.5 Modification du SRS

À cette étape, il s'agit de traduire la requête de changement, préalablement approuvée, dans la spécification des exigences existantes et ce, via l'outil GenSpec. Un changement dans le système peut impliquer soit, une nouvelle exigence pour le système, soit une modification d'une ou plusieurs exigences existantes.

À la fin de cette étape, la requête de changement est dans l'état « SPÉCIFIÉE ».

1.6 Planification du changement

Une fois le changement spécifié, il s'agit de planifier la réalisation de celui-ci. Le chef de projet est donc responsable de répartir, dans le temps, les tâches nécessaires à la réalisation du changement entre les membres de son équipe.

À la fin de cette étape, la requête de changement est dans l'état « PLANIFIÉE ».

1.7 Réalisation du changement

Lors de cette étape, les ressources allouées au développement du changement vont réaliser ce dernier en respectant la planification préétablie.

À la fin de cette étape, la requête de changement est dans l'état « RÉALISÉE ».

1.8 Suivi du changement

Cette dernière étape consiste en un suivi de la réalisation du changement qui a été planifié. Pour ce faire, le chef de projet consulte régulièrement les ressources allouées afin de connaitre l'avancement du développement et constitue, ensuite, une équipe de testeurs qui se chargera de réaliser des plans de tests bien définis afin de vérifier la modification apportée au système.

À la fin de cette étape, la requête de changement est dans l'état « TESTÉE ».

Chapitre 2

Formulaire de changement

Ce chapitre décrit le modèle de formulaire utilisé pour l'enregistrement des requêtes de changement provenant des clients. Ce modèle est composé de trois parties complémentaires, à savoir, l'*identification*, la *description* et l'*analyse*.

Les parties concernant l'*identification* et la *description* de la demande, seront enregistrées par la personne en charge du point de contact avec le client. Les trois parties pourront ensuite être mises à jour par le chef de projet, au fur et à mesure de la progression de la demande à travers les étapes du processus. Remarquons que les éléments dotés d'une astérisque sont obligatoires.

2.1 Identification

Nom du projet* : *Le nom du produit à développer.*
Numéro de la requête* : *Le numéro de la requête dans le projet.*
Titre* : *Le titre résumant le but de la requête.*
Émise par* : *Le client qui émet la requête.*
Enregistrée par* : *Le point de contact du client, celui qui enregistre la requête.*
Priorité* : *Un niveau de priorité de la requête basé sur trois niveaux.*
État* : *L'état de la requête, tel que précisé dans le processus de gestion des changements.*
Date de demande* : *La date à laquelle la requête a été émise par le client.*
Date d'échéance : *La date à laquelle la requête devient inutile pour le client.*

2.2 Description

Type* : *Le type de la requête, tel que : Ajout, Modification, Retrait,...*
Description* : *Une description de la requête, de ce que le client voudrait.*
Description originale : *Une description de l'existant, i.e, comment cela est géré pour le moment.*
Justification* : *La raison qui justifie la réalisation de la requête.*
Commentaire technique : *Un commentaire technique sur la requête.*
Documents relatifs : *Des documents annexes permettant d'illustrer la requête.*

2.3 Analyse

2.3.1 Évaluation

Aperçu de l'impact* : *Description de l'impact du changement sur les autres exigences.*
Exigences concernées : *La liste des numéros d'exigences qui sont concernées par le changement demandé.*
Alternatives : *Les alternatives possibles relatives au changement demandé.*
Action* : *Ce qu'il convient d'entreprendre pour réaliser le changement : Temps, Ressources,...*

2.3.2 Planification

Analyste* : *La personne responsable d'évaluer et de planifier le changement.*
Approbateur* : *La personne qui donne son accord pour réaliser le changement.*
Implémenteur* : *La personne responsable d'implémenter le changement.*
Date d'approbation : *La date à laquelle l'approbateur a donné son accord.*
Date de début des travaux : *La date planifiée de début des travaux.*
Date de fin des travaux : *La date planifiée de fin des travaux.*
Date finale : *La date de réalisation finale du changement.*

Deuxième partie

Analyse des besoins

Chapitre 3

Classes d'utilisateurs

3.1 Chef de projet

3.1.1 Caractéristiques

1. **Attributs physiques :** La personne a l'âge légal pour exercer la fonction de chef de projet. Il n'y a à priori aucun âge spécifique ni de sexe typique. Rien n'est prévu au sein du logiciel pour la prise en charge de chef de projet à capacités réduites. Des incapacités physiques et handicaps modérés sont cependant tolérés.
2. **Attributs mentaux :** Le chef de projet est familiarisé aux concepts modernes des interfaces graphiques. Notre application n'est pas adaptée aux handicaps lourds. Du fait de sa qualification et de son intérêt évident pour la gestion des changements, le chef de projet adopte une attitude positive vis-à-vis de la tâche.
3. **Qualifications et connaissances :** La langue maternelle du chef de projet est le français. Les périphériques d'entrée utilisés sont le clavier et la souris.
4. **Caractéristiques du travail :** Le but du travail du chef de projet est de mettre à jour le système en gérant les demandes de changement ainsi qu'en gérant l'ajout et la suppression d'utilisateurs. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel.

3.1.2 Environnement

1. **Localisation du produit :** Le logiciel est utilisé principalement dans les locaux de l'entreprise.
2. **Position :** Le chef de projet est généralement assis pour utiliser l'ordinateur et adopte vraisemblablement cette même position pour utiliser le logiciel.
3. **Matériel :** L'ordinateur du chef de projet est suffisamment puissant pour faire fonctionner la machine virtuelle java et notre application.
4. **Logiciel et Système d'exploitation :** Le système d'exploitation du chef de projet est Windows XP. Notre application ne requiert pas de système d'exploitation particulier mais une machine virtuelle Java doit être installée.
5. **Structure :**
   - *Travail en groupe :* Le logiciel est utilisé individuellement.
   - *Assistance :* Une aide incorporée au logiciel est disponible pour le chef de projet.
   - *Interruptions :* Ce facteur dépend fortement du contexte dans lequel le chef de projet se trouve. Dans le cas le plus probable, celui-ci n'est pas interrompu.
   - *Communications :* Le réseau de l'entreprise permet de véhiculer l'information liée à l'utilisation du logiciel.

3.2 Utilisateur

3.2.1 Caractéristiques

1. **Attributs physiques :** La personne a l'âge légal pour exercer la fonction d'utilisateur. Il n'y a à priori aucun âge spécifique ni de sexe typique. Rien n'est prévu au sein du logiciel pour la prise en charge d'utilisateurs à capacités réduites. Des incapacités physiques et handicaps modérés sont cependant tolérés.
2. **Attributs mentaux :** L'utilisateur est familiarisé aux concepts modernes des interfaces graphiques. Notre application n'est pas adaptée aux handicaps lourds. Du fait de sa qualification et de son intérêt évident pour la gestion des changements, l'utilisateur adopte une attitude positive vis-à-vis de la tâche.
3. **Qualifications et connaissances :** La langue maternelle de l'utilisateur est le français. Les périphériques d'entrée utilisés sont le clavier et la souris.
4. **Caractéristiques du travail :** Le but du travail de l'utilisateur est de mettre à jour le système en y incluant les nouvelles demandes de changement. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel.

3.2.2 Environnement

1. **Localisation du produit :** Le logiciel est utilisé principalement dans les locaux de l'entreprise.
2. **Position :** L'utilisateur est généralement assis pour utiliser l'ordinateur et adopte vraisemblablement cette même position pour utiliser le logiciel.
3. **Matériel :** L'ordinateur de l'utilisateur est suffisamment puissant pour faire fonctionner la machine virtuelle java et notre application.
4. **Logiciel et Système d'exploitation :** Le système d'exploitation du chef de projet est Windows XP. Notre application ne requiert pas de système d'exploitation particulier mais une machine virtuelle Java doit être installée.
5. **Structure :**
   - *Travail en groupe :* Le logiciel est utilisé individuellement.
   - *Assistance :* Une aide incorporée au logiciel est disponible pour l'utilisateur.
   - *Interruptions :* Ce facteur dépend fortement du contexte dans lequel l'utilisateur se trouve. Dans le cas le plus probable, celui-ci n'est pas interrompu.
   - *Communications :* Le réseau de l'entreprise permet de véhiculer l'information liée à l'utilisation du logiciel.

Chapitre 4

Objectifs des utilisateurs

Les cas d'utilisation représentent le comportement affiché par le système sous certaines conditions de manière à satisfaire un objectif de l'un des acteurs. Les diagrammes de cas d'utilisation montrent quant à eux les acteurs, les limites du système, les relations entre acteurs et le système ainsi que les relations entre les cas d'utilisation eux-mêmes. Ceux-ci permettent, aux travers de scénarii, de capturer, représenter et valider les fonctions d'un domaine, les spécifier et donner un aperçu de la dynamique et des interactions. Du fait de leur simplicité, les cas d'utilisation ont pour avantage notable d'être compréhensibles par tous.

Notons que, afin d'avoir une vision plus précise et plus simple du système, nous l'avons divisé en deux parties : le système d'information du chef de projet (SI Chef de projet) et le système d'information de l'utilisateur (SI Utilisateur). De plus, toutes les fonctionnalités disponibles pour l'utilisateur le sont aussi pour le chef de projet.

4.1 SI du chef de projet

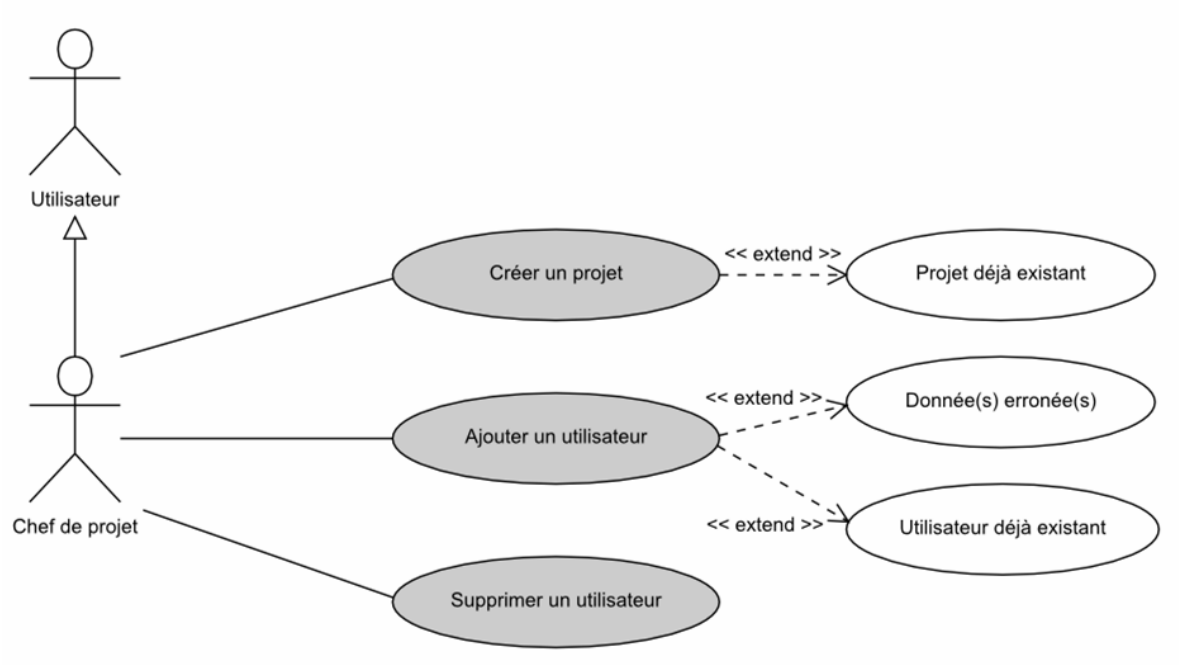

Fig. 4.1 – Diagramme de cas d'utilisation du chef de projet

4.2 SI de l'utilisateur

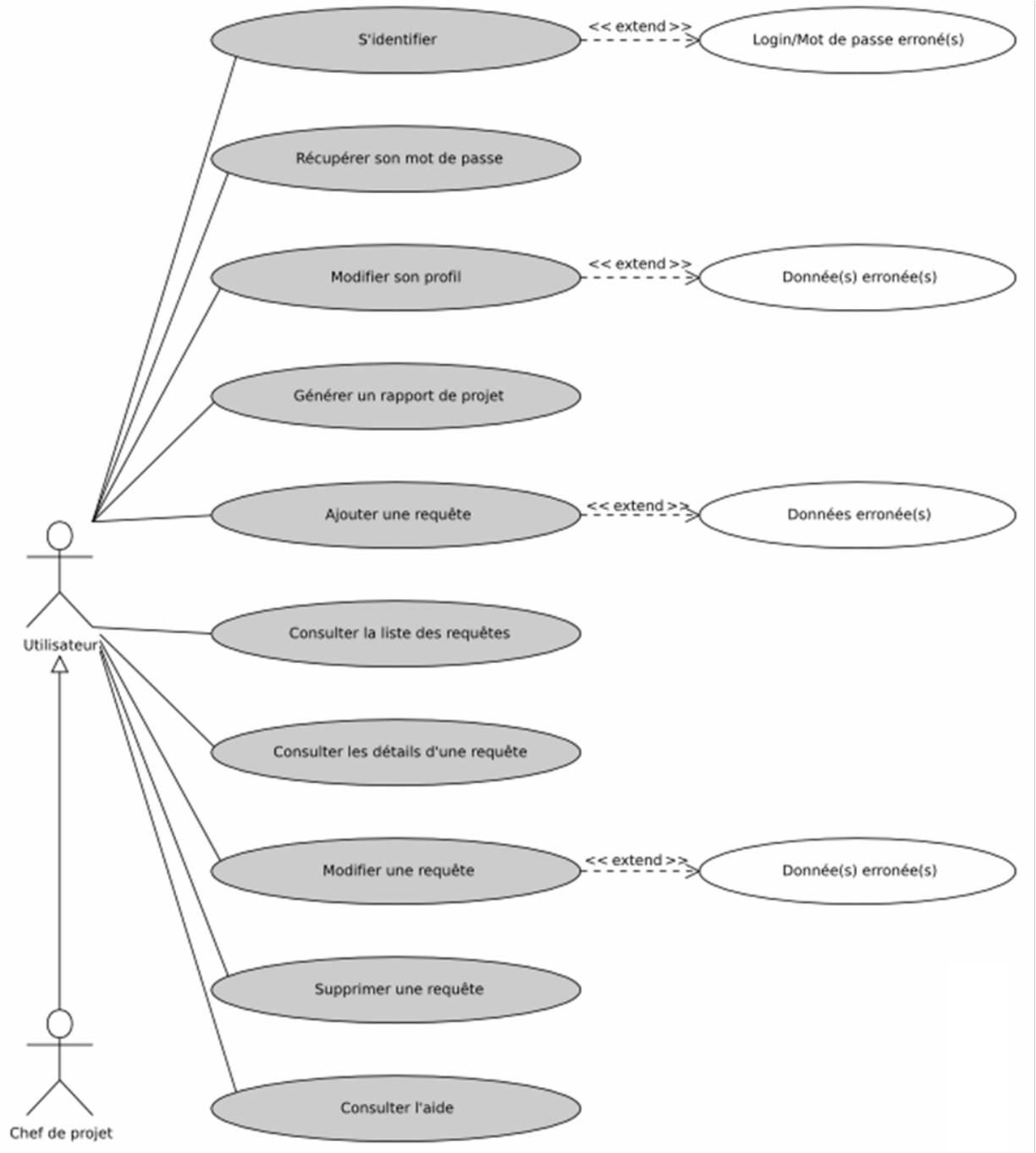

FIG. 4.2 – Diagramme de cas d'utilisation de l'utilisateur

Chapitre 5

Scénarii d'utilisation du chef de projet

5.1 Créer un projet

Résumé :

L'acteur crée un nouveau projet.

Acteur :

Le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouveau projet est enregistré dans le système.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre les caractéristiques du nouveau projet.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que le nouveau projet a bien été enregistré dans le système.
5. L'acteur reçoit l'information.	

5.1.1 Extension : Projet déjà existant

Résumé :

L'acteur crée un nouveau projet qui est déjà enregistré dans le système.

Acteur :

Le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouveau projet est identique à un projet déjà enregistré dans le système.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouveau projet n'est pas enregistré dans le système.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre les caractéristiques d'un projet existant.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que le projet existe déjà et lui demande de réintroduire les caractéristiques d'un nouveau projet.
5. Retour au point 1 du cas normal.	

5.2 Ajouter un utilisateur

Résumé :

L'acteur ajoute un nouvel utilisateur dans le système.

Acteur :

Le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouvel utilisateur est enregistré dans le système.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre les caractéristiques du nouvel utilisateur.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que le nouvel utilisateur a bien été enregistré dans le système avec un login déterminé.
5. L'acteur reçoit l'information.	

5.2.1 Extension : Donnée(s) erronée(s)

Résumé :

L'acteur ajoute un nouvel utilisateur dans le système avec une/des donnée(s) erronée(s).

Acteur :

Le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Une/des donnée(s) est/sont erronée(s).

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouvel utilisateur n'est pas enregistré dans le système.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre les caractéristiques du nouvel utilisateur dont certaines sont erronées.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que certaines données sont erronées et lui demande de réintroduire correctement les caractéristiques d'un nouvel utilisateur.
5. Retour au point 1 du cas normal.	

5.2.2 Extension : Utilisateur déjà existant

Résumé :

L'acteur ajoute un nouvel utilisateur qui est déjà enregistré dans le système.

Acteur :

Le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouvel utilisateur est identique à un utilisateur déjà enregistré dans le système.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouvel utilisateur n'est pas enregistré dans le système.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre les caractéristiques d'un utilisateur déjà existant.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que l'utilisateur existe déjà et lui demande de réintroduire les caractéristiques d'un nouvel utilisateur.
5. Retour au point 1 du cas normal.	

5.3 Supprimer un utilisateur

Résumé :

L'acteur supprime un des utilisateurs du système.

Acteur :

Le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

L'utilisateur est enregistré dans le système.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

L'utilisateur n'est plus enregistré dans le système.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur sélectionne l'utilisateur à supprimer.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que l'utilisateur est bien supprimé.
5. L'acteur reçoit l'information.	

Chapitre 6

Scénarii d'utilisation de l'utilisateur

6.1 S'identifier

Résumé :

L'acteur s'identifie afin d'accéder au programme de gestion des changements.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur lance le système.	
2. L'acteur entre son login et son mot de passe.	
	3. Le système reçoit les données.
	4. Le système traite et vérifie les données.
5. L'acteur entre dans le programme.	

6.1.1 Extension : Login et/ou mot de passe erroné(s)

Résumé :

L'acteur s'identifie, afin d'accéder au programme de gestion des changements, avec un login et/ou un mot de passe erroné(s).

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur entre un login et/ou un mot de passe erroné(s).

Postcondition :

Le système est opérationnel.

L'acteur n'est pas identifié.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur lance le système. 2. L'acteur entre son login et son mot de passe de façon erronée.	
	3. Le système reçoit les données. 4. Le système traite et vérifie les données. 5. Le système signale à l'acteur qu'il a entré un login et/ou un mot de passe erroné(s) et lui demande de s'identifier à nouveau.
6. Retour au point 2 du cas normal.	

6.2 Récupérer son mot de passe

Résumé :

L'acteur tente de récupérer son mot de passe.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

Postcondition :

Le système est opérationnel.

Le mot de passe est envoyé par mail à l'utilisateur.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur lance le système. 2. L'acteur demande son mot de passe au système.	
	3. Le système reçoit la demande. 4. Le système traite et vérifie la demande. 5. Le système envoie à l'utilisateur son mot de passe.
6. L'utilisateur reçoit son mot de passe.	

6.3 Modifier son profil

Résumé :

L'acteur modifie son profil.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le profil de l'acteur est modifié.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre les donnés afin de modifier son profil.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que son profil a été modifié avec succès.
5. L'acteur reçoit l'information.	

6.3.1 Extension : Donnée(s) erronée(s).

Résumé :

L'acteur modifie son profil avec une/des donnée(s) erronée(s).

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Certaines données entrées par l'acteur sont erronées.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le profil de l'acteur est inchangé.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre une/des donnés erronée(s) afin de modifier son profil.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système demande à l'acteur de rentrer des données correctes.
5. Retour au point 1 du cas normal.	

6.4 Générer un rapport de projet

Résumé :

L'acteur génère, au format pdf, le rapport des requêtes de changement relatives à un projet.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le rapport est généré.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur demande au système de générer le rapport relatif à un projet.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que le rapport a été créé et placé dans le répertoire relatif au projet concerné.
5. L'acteur reçoit l'information.	

6.5 Ajouter une requête

Résumé :

L'acteur ajoute une nouvelle requête de changement dans le système.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

La nouvele requête de changement est enregistrée dans le système.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre les caractéristiques concernant la nouvelle requête de changement.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que la nouvelle requête de changement a bien été enregistrée dans le système.
5. L'acteur reçoit l'information.	

6.5.1 Extension : Donnée(s) erronée(s)

Résumé :

L'acteur ajoute une nouvelle requête de changement dans le système avec une/des donnée(s) erronée(s).

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Une/des donnée(s) est/sont erronée(s).

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

La nouvele requête de changement n'est pas enregistrée dans le système.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre les caractéristiques de la nouvelle requête de changement avec une/des donnée(s) erronée(s).	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système demande à l'acteur de rentrer des données correctes.
5. Retour au point 1 du cas normal.	

6.6 Consulter la liste des requêtes

Résumé :

L'utilisateur consulte la liste des requêtes de changement d'un projet particulier et selon un ordre spécifié.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

L'utilisateur a accès à la liste des requêtes de changement d'un projet particulier et selon un ordre spécifié.

Flux d'événement :

Acteur	*Réponse du système*
1. L'utilisateur demande pour consulter la liste des requêtes de changement d'un projet particulier et selon un ordre bien précis.	
	2. Le système reçoit la demande. 3. Le système traite la demande. 4. Le système envoie la liste des requêtes du projet, classée selon l'ordre spécifié.
5. L'acteur a accès à la liste des requêtes du projet selon l'ordre spécifié.	

6.7 Consulter les détails d'une requête

Résumé :

L'acteur

L'acteur consulte les détails d'une requête, à savoir, l'ensemble de ses caractéristiques ainsi que la liste des requêtes liées (requêtes qui référencent au moins une des exigences référencées par la requête consultée).

Acteur :

Le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

L'acteur reçoit les détails de la requête.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur spécifie la requête pour laquelle il veut obtenir les détails, et ce, en consultant la liste des requêtes.	
	2. Le système reçoit la demande. 3. Le système traite la demande. 4. Le système envoie les détails de la requête spécifiée.
5. L'acteur a accès aux détails de la requête spécifiée.	

6.8 Modifier une requête

Résumé :

L'acteur modifie certaines caractéristiques d'une requête de changement enregistrée dans le système.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

La requête de changement est enregistrée dans le système.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Certaines caractéristiques de la requête de changement sont changées.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre les nouvelles caractéristiques de la requête de changement.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que les changements ont été effectués.
5. L'acteur reçoit l'information.	

6.8.1 Extension : Donnée(s) erronée(s)

Résumé :

L'acteur modifie, de façon erronée, certaines caractéristiques d'une requête de changement enregistrée dans le système.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

La requête de changement est enregistrée dans le système.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Les caractéristiques de la requête de changement sont inchangées.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur entre, de façon erronée, les nouvelles caractéristiques de la requête de changement.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système demande à l'acteur de rentrer des données correctes.
5. Retour au point 1 du cas normal.	

6.9 Supprimer une requête

Résumé :

L'acteur supprime une des requête de changement enregistrée dans système.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

L'acteur est identifié.

La requête de changement est enregistrée dans le système.

Postcondition :

Le système est opérationnel.

L'acteur est identifié.

La requête de changement n'est plus enregistrée dans le système.

Flux d'événement :

Acteur	*Réponse du système*
1. L'acteur sélectionne la requête de changement à supprimer.	
	2. Le système reçoit les données. 3. Le système traite et vérifie les données. 4. Le système avertit l'acteur que la requête de changement est supprimée.
5. L'acteur reçoit l'information.	

6.10 Consulter l'aide

Résumé :

L'utilisateur demande de l'aide sur la manière dont fonctionne le programme.

Acteur :

L'utilisateur et le chef de projet.

Précondition :

Le système est opérationnel.

Postcondition :

Le système est opérationnel.

L'utilisateur reçoit l'aide demandée.

Flux d'événement :

Acteur	*Réponse du système*
1. L'utilisateur demande de l'aide sur la manière dont fonctionne le programme.	
	2. Le système reçoit la demande.
	3. Le système traite la demande.
	4. Le système envoie l'aide à l'acteur.
5. L'acteur reçoit l'aide demandée.	

Chapitre 7

Diagramme de la statique

7.1 Schéma de la statique

Le modèle Entité-Relation-Attribut propose des concepts (principalement les entités, les associations et les attributs) permettant de décrire un ensemble de données relatives à un domaine défini.

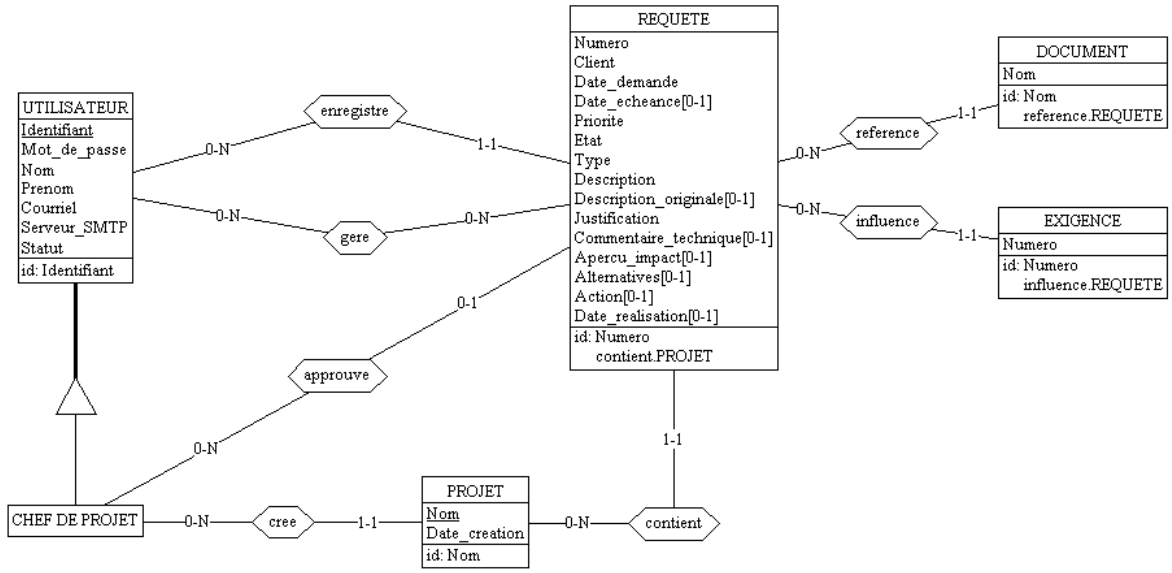

Fig. 7.1 – Modèle de la statique

Chapitre 8

Diagramme de composants

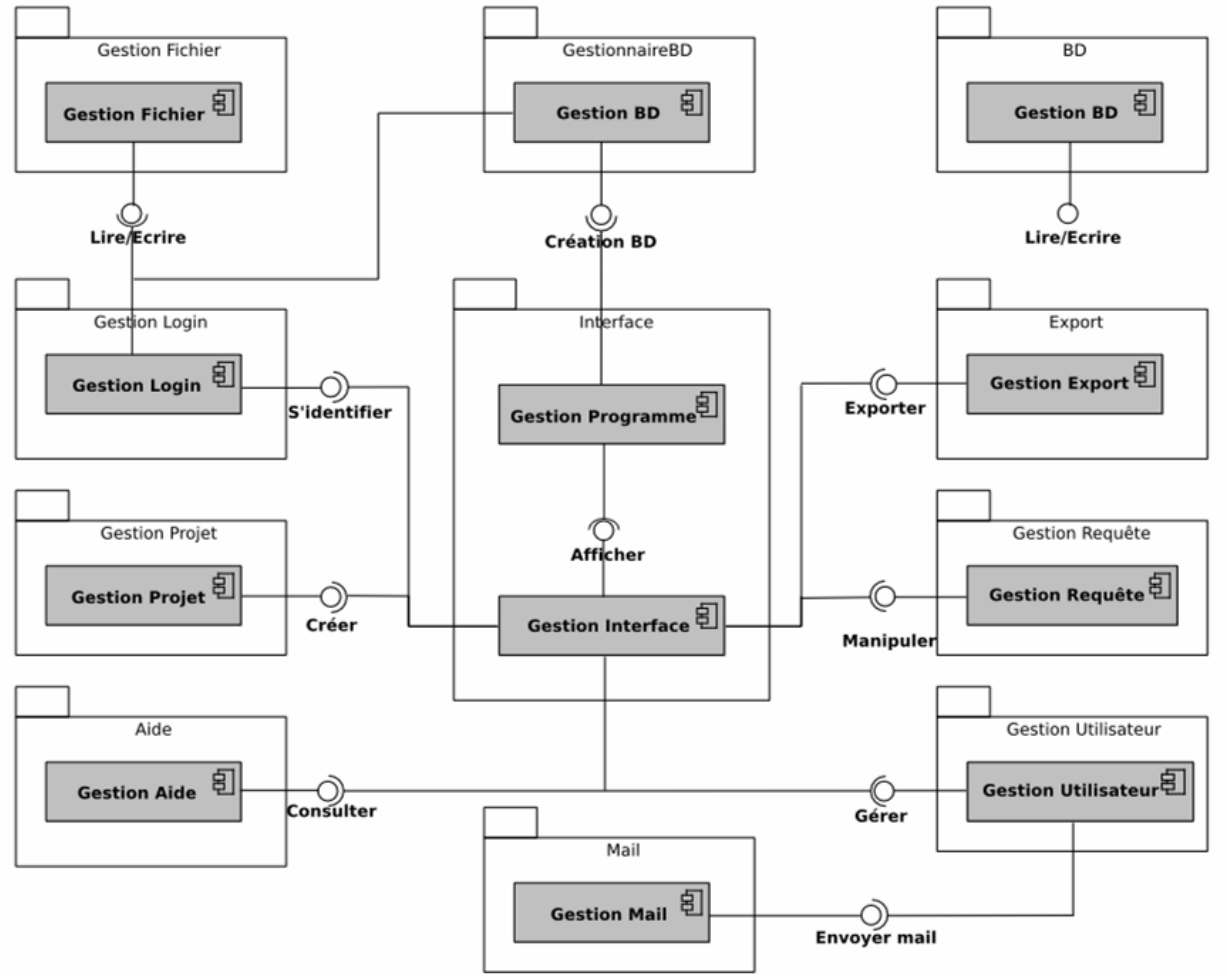

FIG. 8.1 – Diagramme de composants

Chapitre 9

Diagramme de classes

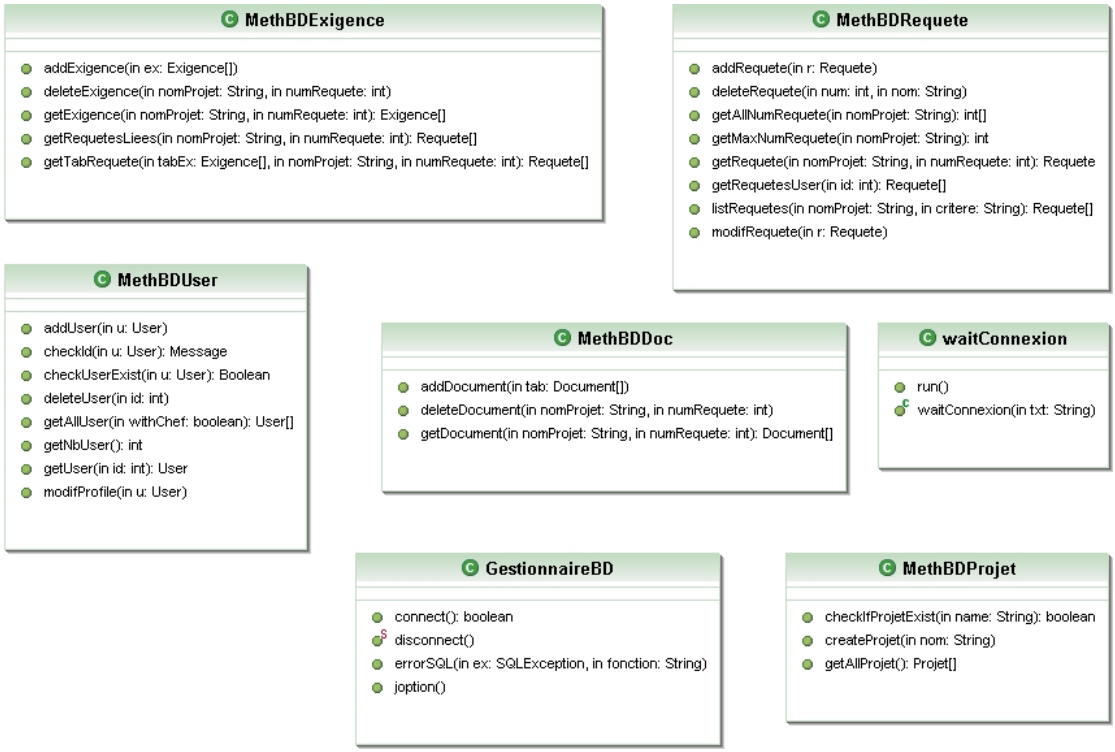

FIG. 9.1 – Package BD

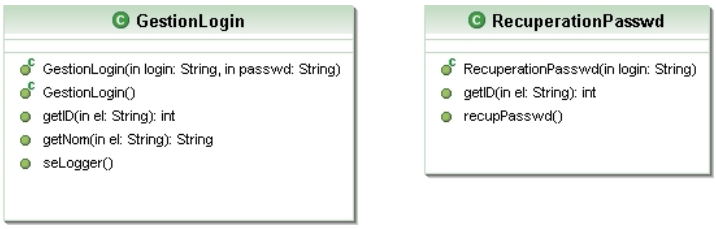

FIG. 9.2 – Package GestionLogin

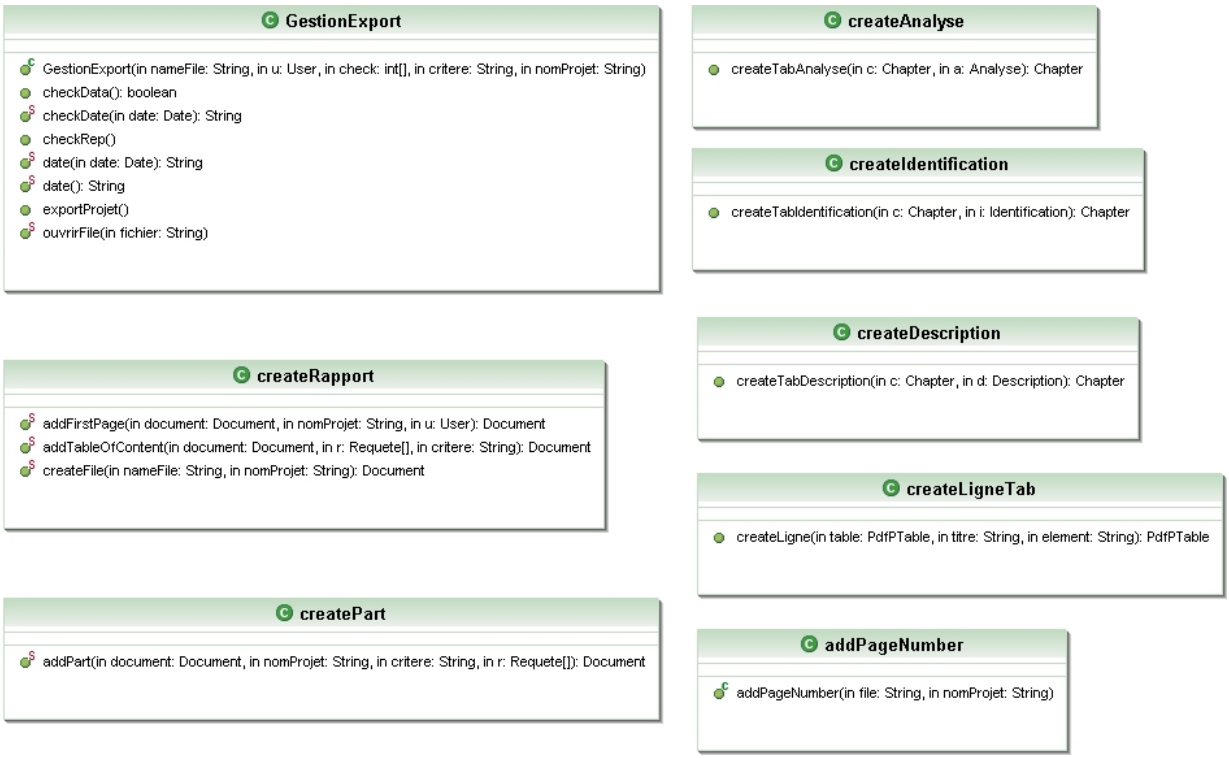

FIG. 9.3 – Package Export

FIG. 9.4 – Package GestionAide

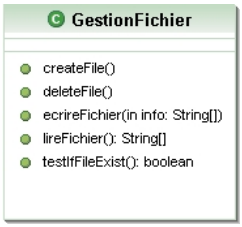

FIG. 9.5 – Package GestionFichier

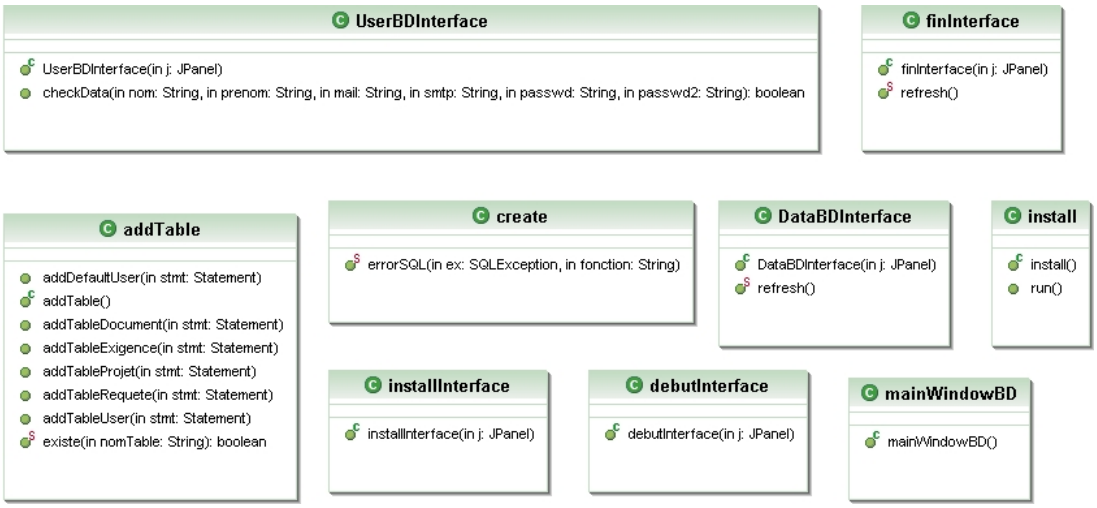

FIG. 9.6 – Package GestionnaireBD

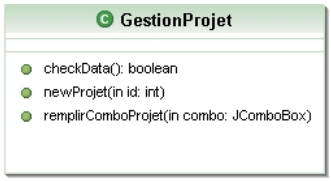

FIG. 9.7 – Package GestionProjet

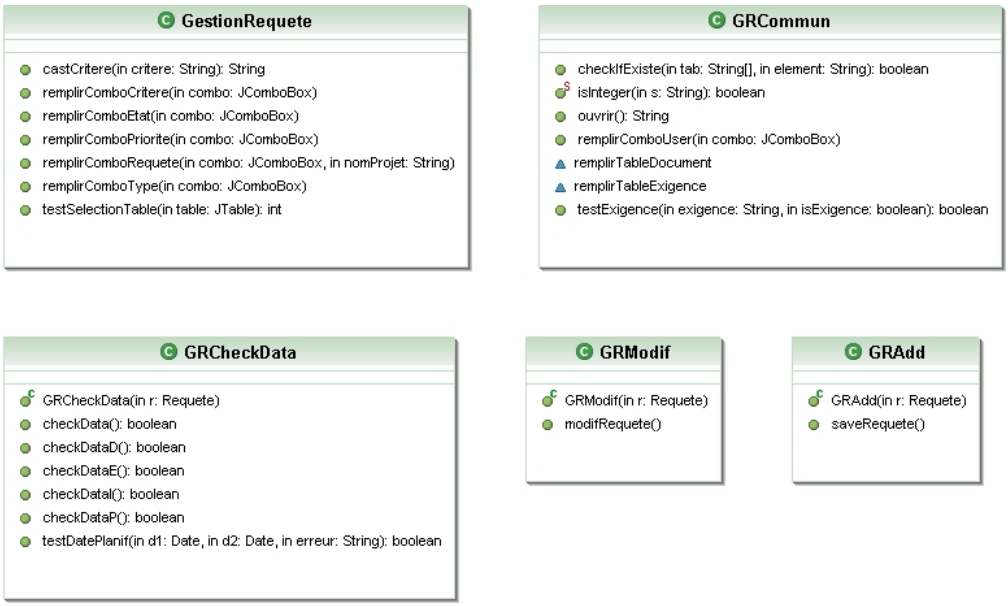

FIG. 9.8 – Package GestionRequete

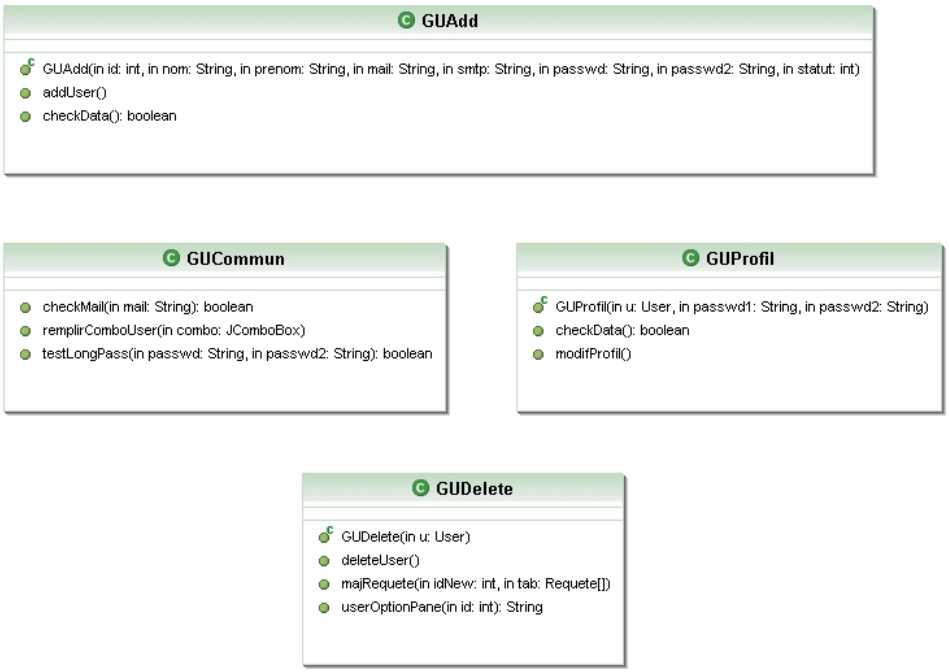

FIG. 9.9 – Package GestionUtilisateurs

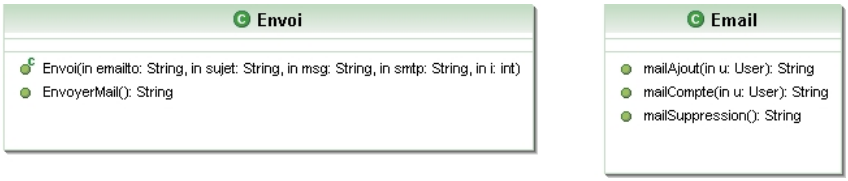

FIG. 9.10 – Package Mail

Fig. 9.11 – Package Interface

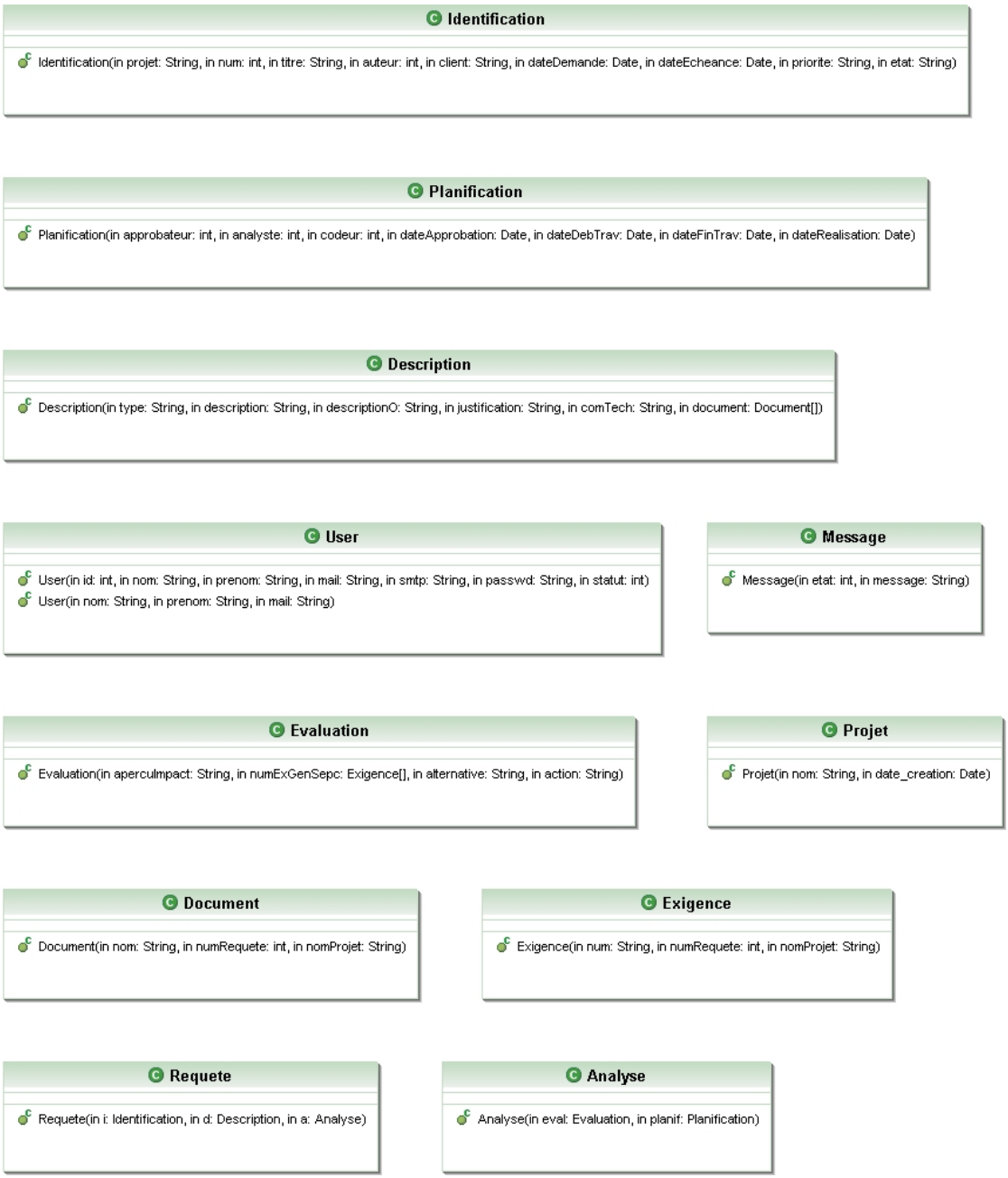

Fig. 9.12 – Package Objet

Troisième partie

Explication des interfaces

Chapitre 10

Gestion de la base de données

10.1 Initialisation de modeX

ModeX utilisant une base de données, la phase d'initialisation permet, lors du premier démarrage de modeX sur un poste de travail, soit, d'initialiser une base de données, soit, de se connecter à une base de données existante.

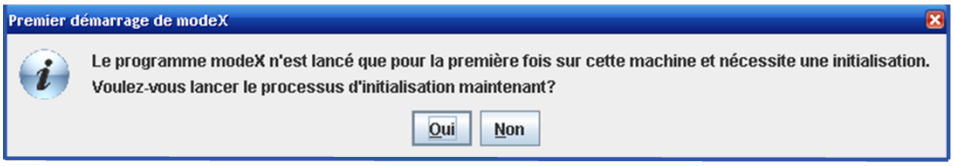

FIG. 10.1 – Initialisation de modeX

ModeX détectera automatiquement si la base de données spécifiée ci-après est déjà initialisée sur le serveur. Le cas échéant, il permettra à l'utilisateur de connecter modeX à cette base de données. Dans le cas contraire, il permettra au chef de projet de créer les tables de la base de données de modeX.

FIG. 10.2 – Démarrage de l'initialisation

10.2 Gestion de la base de données

Afin de pouvoir se connecter à la base de données, il est nécessaire de spécifier l'adresse IP du serveur de base de données de l'entreprise.

Les champs en dessous de l'adresse IP - nom de la BD, login et mot de passe - permettent de spécifier la base de données à laquelle modeX doit se connecter.

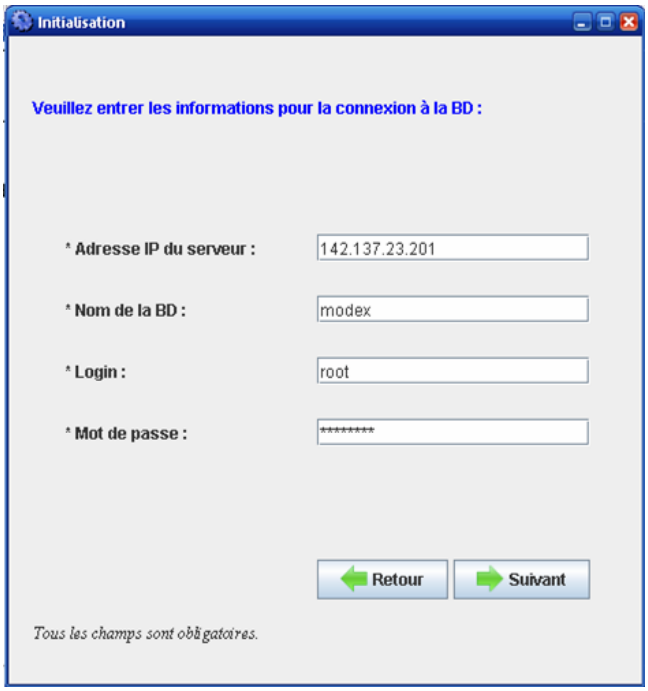

FIG. 10.3 – Connexion à la base de données

10.3 Gestion de votre compte

Les informations à introduire à cette étape concernent soit, l'utilisateur potentiel du programme modeX, soit, le chef de projet qui initialise la base de données.

Dans le premier cas, ces informations seront envoyées par mail au chef de projet qui donnera son approbation en ajoutant, ultérieurement, l'utilisateur dans le système.

Dans le second cas, ces informations permettront la création du compte « Chef de projet »dans le système.

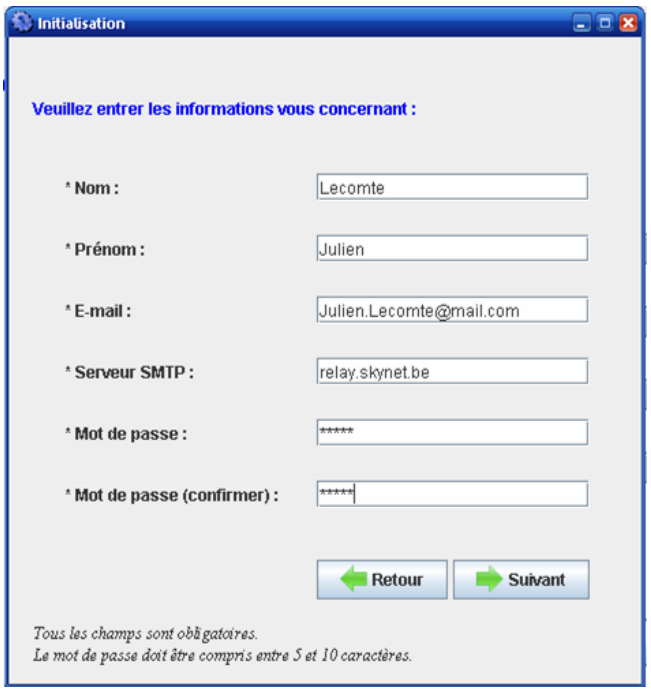

FIG. 10.4 – Informations sur le compte à créer

Remarquons que le serveur smtp étant généralement le serveur de l'entreprise, il est toutefois possible de le personnaliser pour une éventuelle utilisation hors des locaux de l'entreprise.

10.4 Installation

L'étape d'installation permet de :

- Tester la connexion à la base de données.
- Détecter la base de données existante ou créer les tables de la nouvelle base de données.
- Ajouter éventuellement le compte « Chef de projet » au système.
- Envoyer par mail les informations de l'utilisateur au chef de projet ou envoyer les informations du compte « Chef de projet ».

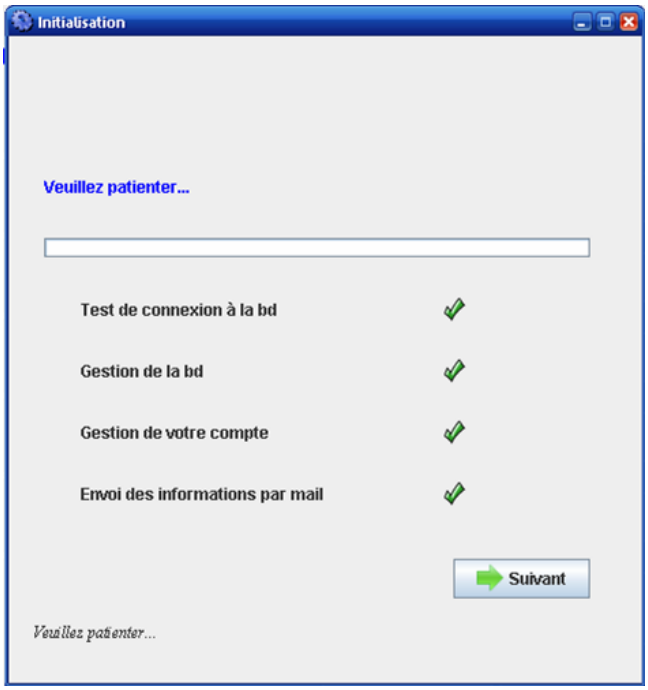

FIG. 10.5 – Installation

10.5 Fin

La phase d'initialisation se termine en récapitulant les informations. En cliquant sur terminer, le programme se lancera directement et permettra l'identification de l'utilisateur ou du chef de projet.

Remarquons que l'utilisateur ne pourra se connecter à modeX que lorsque le chef de projet l'aura explicitement accepté et introduit dans le système.

FIG. 10.6 – Fin de l'installation

Chapitre 11

Identification

11.1 S'identifier

Lors du démarrage de modeX, la première fenêtre qui s'affiche permet de s'identifier dans le système. Il vous faudra donc posséder un compte dans ce dernier, à savoir, un login et un mot de passe.

Le login est composé d'un numéro d'identification unique, généré automatiquement lors de la création du compte, ainsi que du nom et prénom de la personne. Le mot de passe doit, quant à lui, être composé de minimum 5 et maximum 10 caractères.

Suivant votre type de compte - « Chef de projet » ou « Utilisateur » - vous pourrez accéder ou non à certaines fonctionnalités de modeX. Nous verrons ci-après les fonctionnalités disponibles pour chacun des deux types de compte.

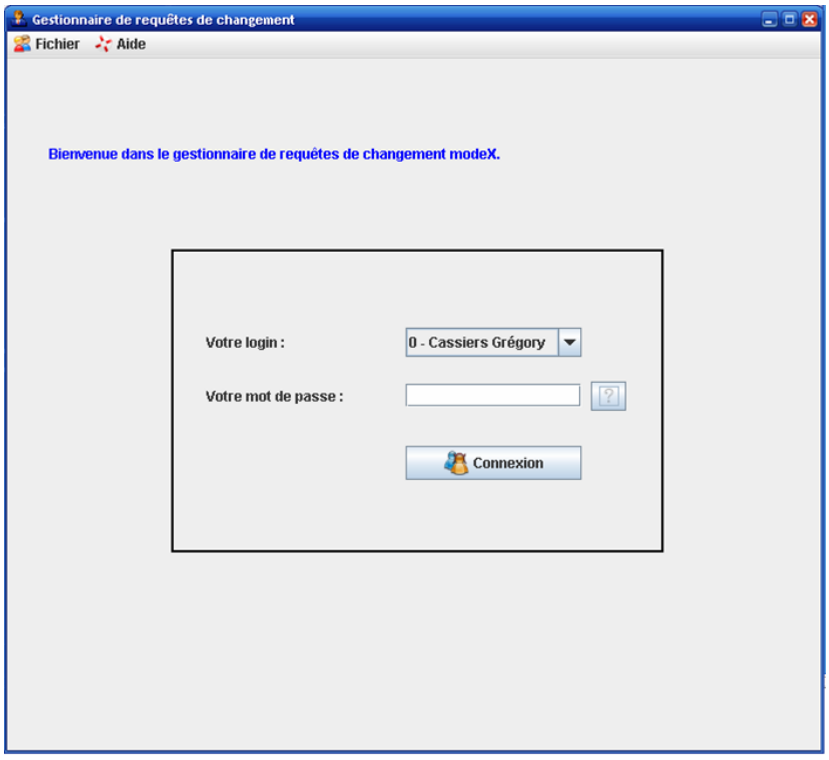

FIG. 11.1 – Identification

11.2 Récupérer son mot de passe

La récupération de votre mot de passe peut se réaliser via le bouton « ? » qui se situe à droite du champ de saisie du mot de passe. En cliquant dessus, votre mot de passe sera envoyé à l'adresse mail que vous avez spécifié dans votre compte.

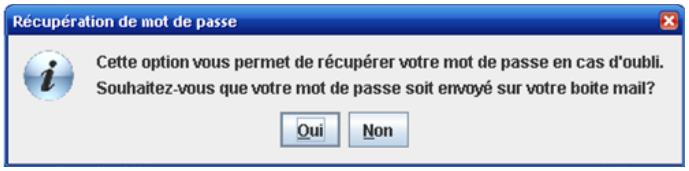

FIG. 11.2 – Récupération de mot de passe

Chapitre 12

Menu "Fichier"

12.1 Modifier le profil

Cette fonctionnalité permet au chef de projet et à l'utilisateur de modifier les caractéristiques de leur compte.

La modification de ces informations nécessite la réintroduction du mot de passe courant du compte. Il est également possible de modifier ce mot de passe en spécifiant dans les deux derniers champs le nouveau mot de passe désiré.

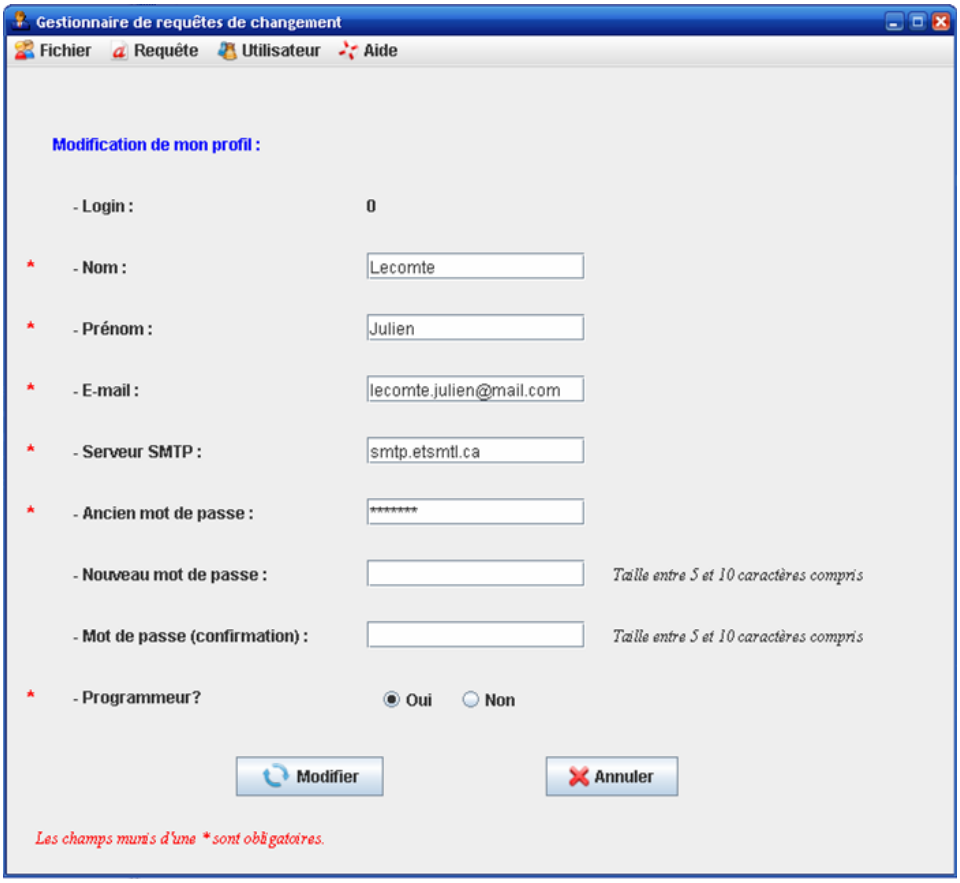

FIG. 12.1 – Modification du profil

12.2 Nouveau projet

Le chef de projet peut créer des projets afin de pouvoir gérer les requêtes de changement pour chacun de ces projets en particulier. Ceci entraîne la création d'un dossier avec le nom du projet dans le répertoire courant de l'application.

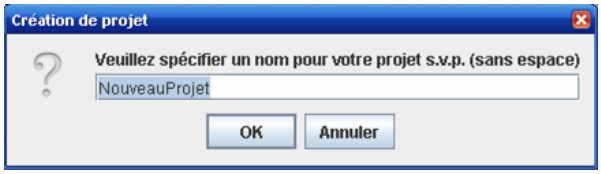

FIG. 12.2 – Nouveau projet

12.3 Exporter le projet

Le chef de projet et l'utilisateur peuvent générer au format pdf un rapport contenant l'ensemble des requêtes d'un projet particulier. Ce document sera placé dans un dossier « Rapport » situé dans le dossier du projet.

Pour chaque requête, il est permis de choisir les parties d'information que l'on veut générer telles que l'identification, la description et l'analyse.

Remarquons que, dans un but de clarté, le choix d'une partie implique automatiquement le choix des parties précédentes. Par exemple, le choix de la partie description, inclura également la partie identification.

En outre, il est possible d'organiser les requêtes par critère. Par exemple, on souhaitera voir apparaître les requêtes d'un projet par ordre de priorité.

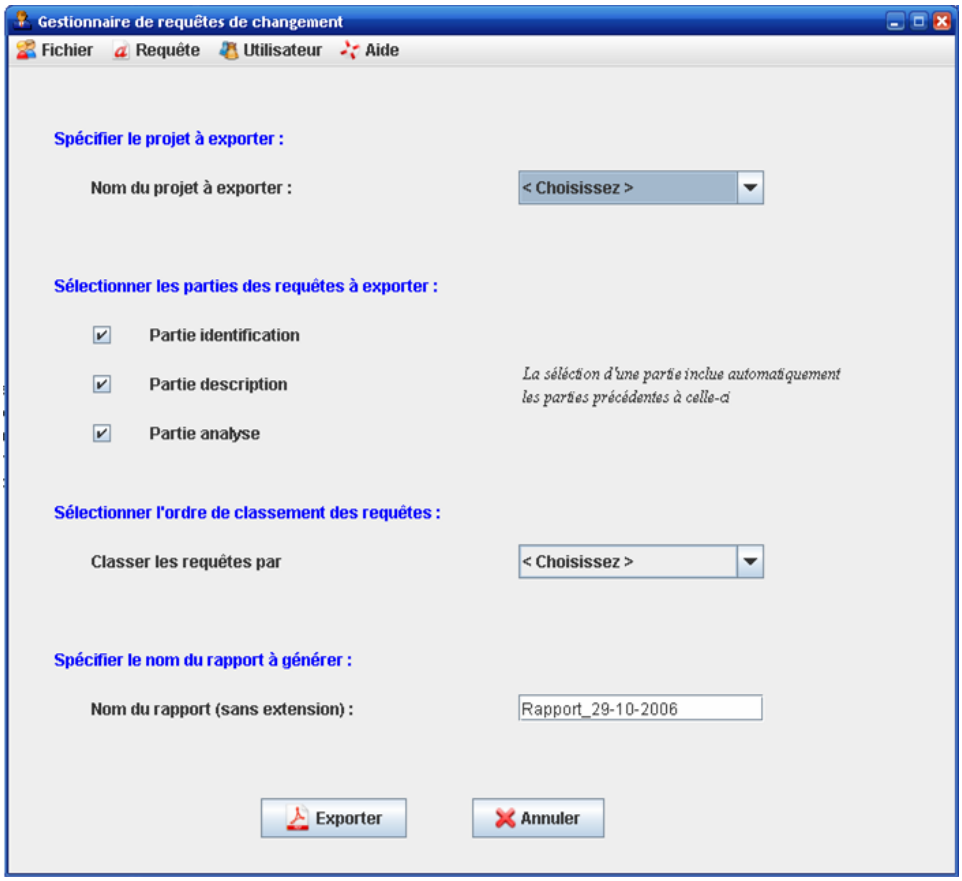

FIG. 12.3 – Exporter le projet

12.4 Déconnexion

En cliquant sur déconnexion, vous revenez à l'interface d'identification de modeX.

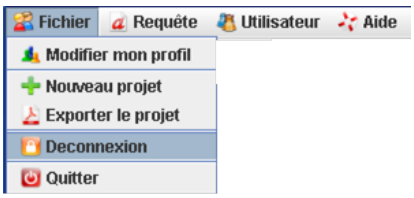

FIG. 12.4 – Déconnexion

12.5 Quitter

En cliquant sur quitter, vous fermez le programme modeX.

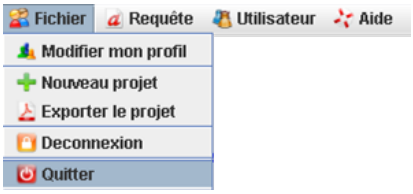

FIG. 12.5 – Quitter

Chapitre 13

Menu "Requête"

13.1 Ajouter une requête

L'ajout d'une requête dans le système peut être réalisée par le chef de projet et par l'utilisateur. Cependant, l'utilisateur ne pourra introduire que certaines données relatives à la requête, à savoir, les informations contenues dans la partie Identification et la partie Description.

13.1.1 Identification

Cette partie d'information permet l'identification de la requête. Une requête est identifiée par le projet qu'elle concerne et par son numéro, généré automatiquement et unique dans le projet. Les détails de ces informations sont explicités dans le chapitre 2 de ce document.

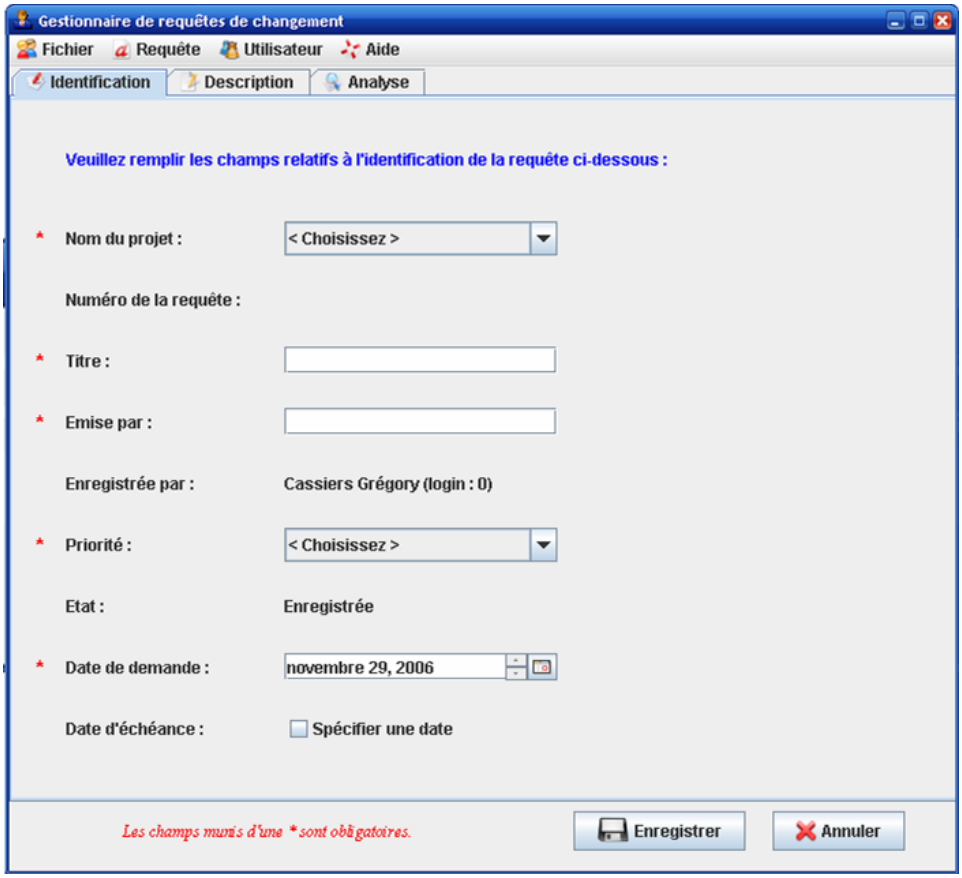

Fig. 13.1 – Ajouter une requête - Identification

13.1.2 Description

Cette partie d'information permet la description du changement désiré.

Les détails de ces informations sont explicités dans le chapitre 2 de ce document. Remarquons que les noms des documents relatifs peuvent être ajoutés ou retirés aisément via un navigateur de documents en cliquant sur « Ajouter » ou « Retirer ».

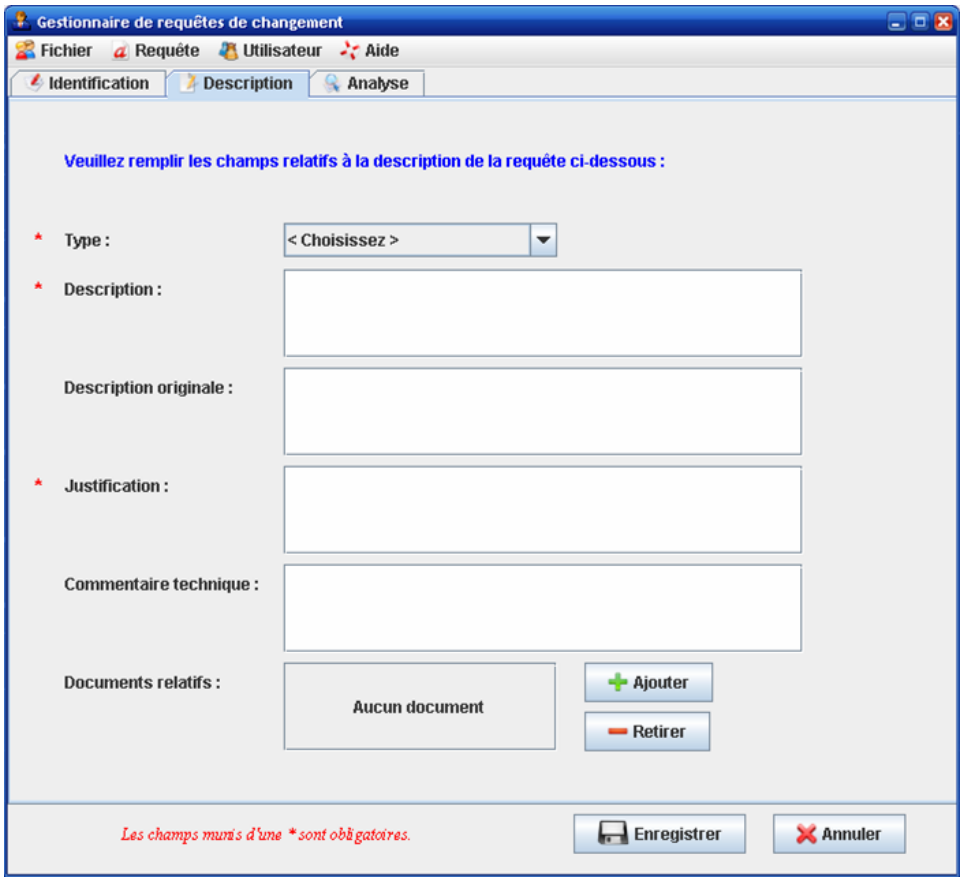

FIG. 13.2 – Ajouter une requête - Description

13.1.3 Analyse - Evaluation

Cette partie d'information permet au chef de projet, et au chef de projet uniquement, de spécifier les informations d'évaluation de la requête.

Les détails de ces informations sont explicités dans le chapitre 2 de ce document. Remarquons que les numéros d'exigences peuvent être ajoutés en respectant la syntaxe suivante : x.x.x.x.x.x où x représente un niveau de l'arbre hiérarchique des exigences.

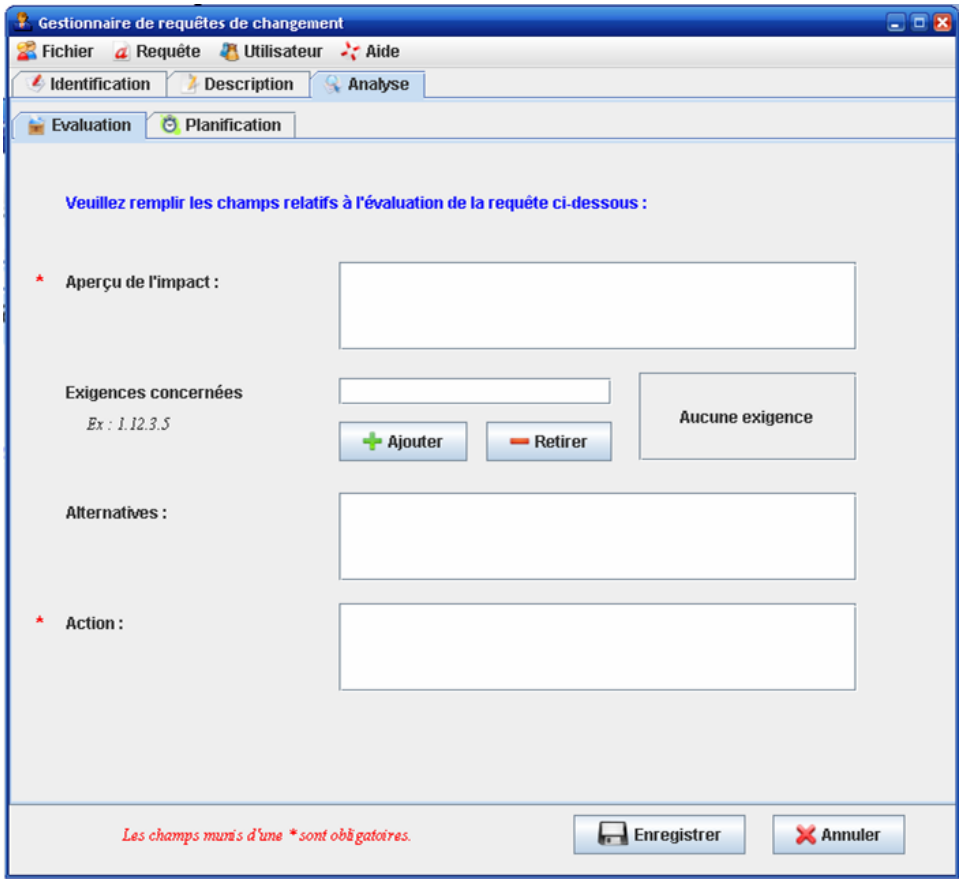

FIG. 13.3 – Ajouter une requête - Evaluation

13.1.4 Analyse - Planification

Cette partie d'information permet au chef de projet, et au chef de projet uniquement, de spécifier les informations de planification du changement.

Les détails de ces informations sont explicités dans le chapitre 2 de ce document. Remarquons que l'analyste sera toujours le chef de projet.

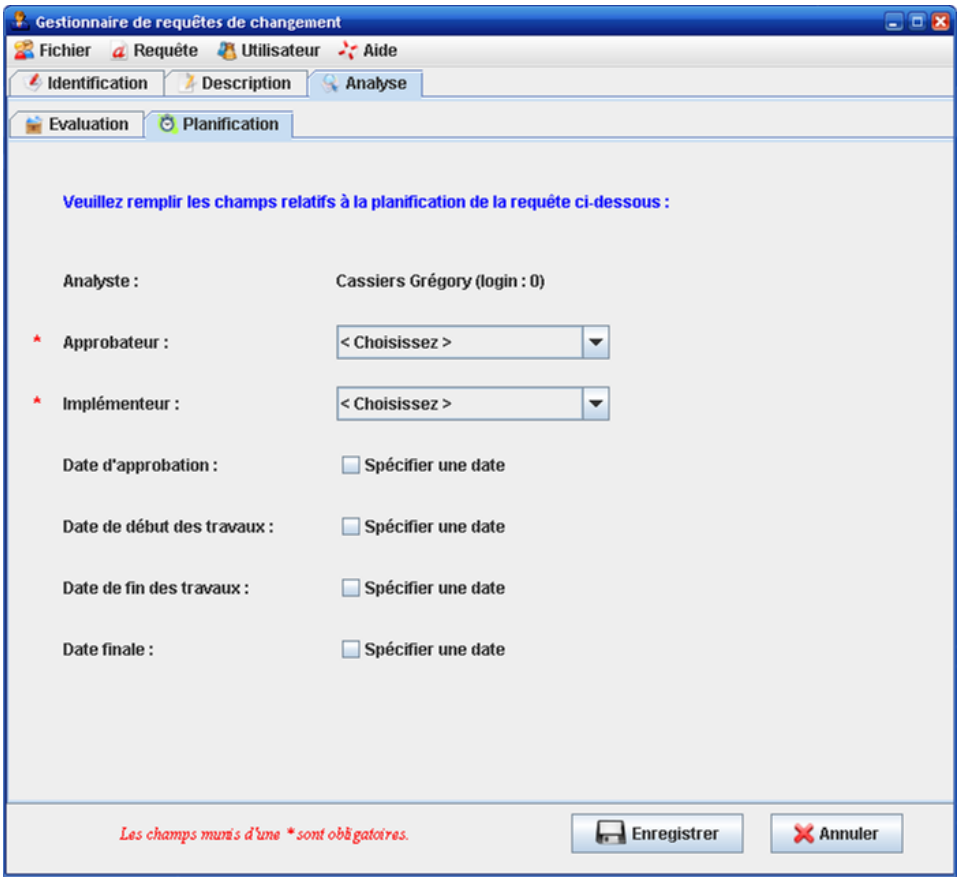

FIG. 13.4 – Ajouter une requête - Planification

13.2 Modifier une requête

La modification d'une requête dans le système peut être réalisée par le chef de projet et par l'utilisateur. Cependant, comme pour l'ajout d'une requête, l'utilisateur ne pourra modifier que certaines données relatives à la requête, à savoir, les informations contenues dans la partie Identification et la partie Description.

13.3 Supprimer une requête

La suppression d'une requête dans le système peut être réalisée par le chef de projet et par l'utilisateur.

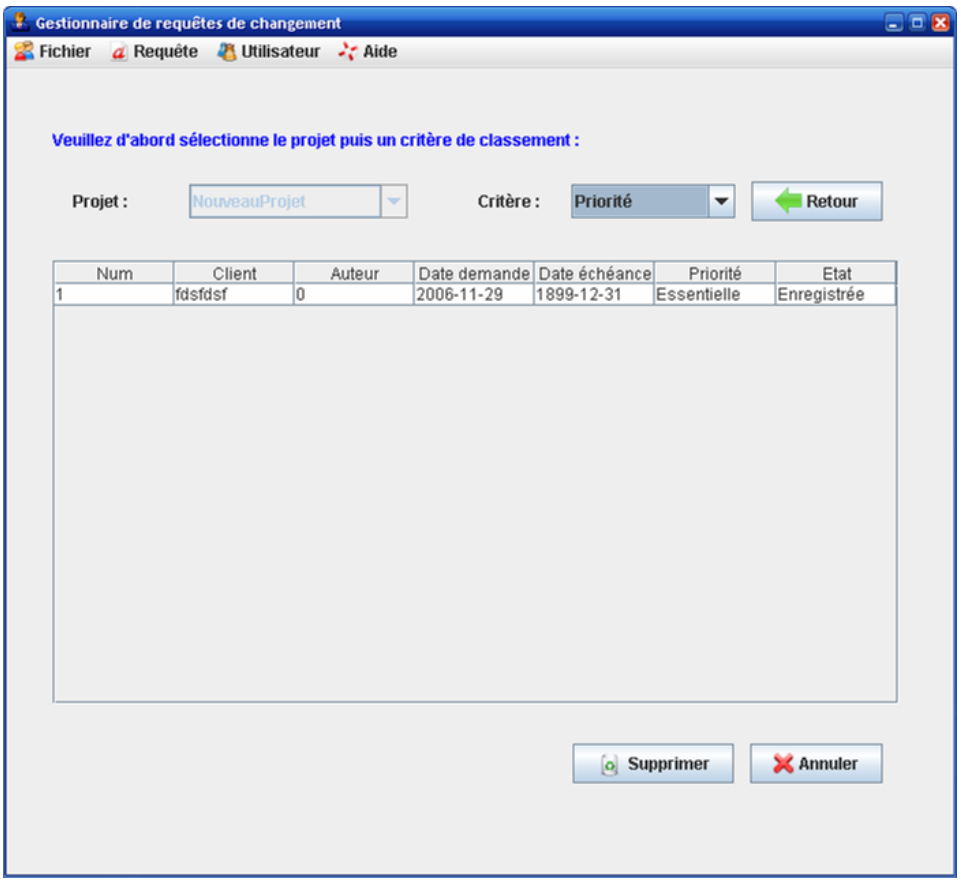

FIG. 13.5 – Supprimer une requête

13.4 Consulter les détails d'une requête

Cette fonctionnalité permet au chef de projet et à l'utilisateur de consulter les caractéristiques d'une requête sans pouvoir modifier ces informations.

En outre, il est possible de lister les requêtes dépendantes de la requête consultée. Les requêtes dépendantes sont des requêtes qui référencent au moins une exigence référencée par la requête consultée.

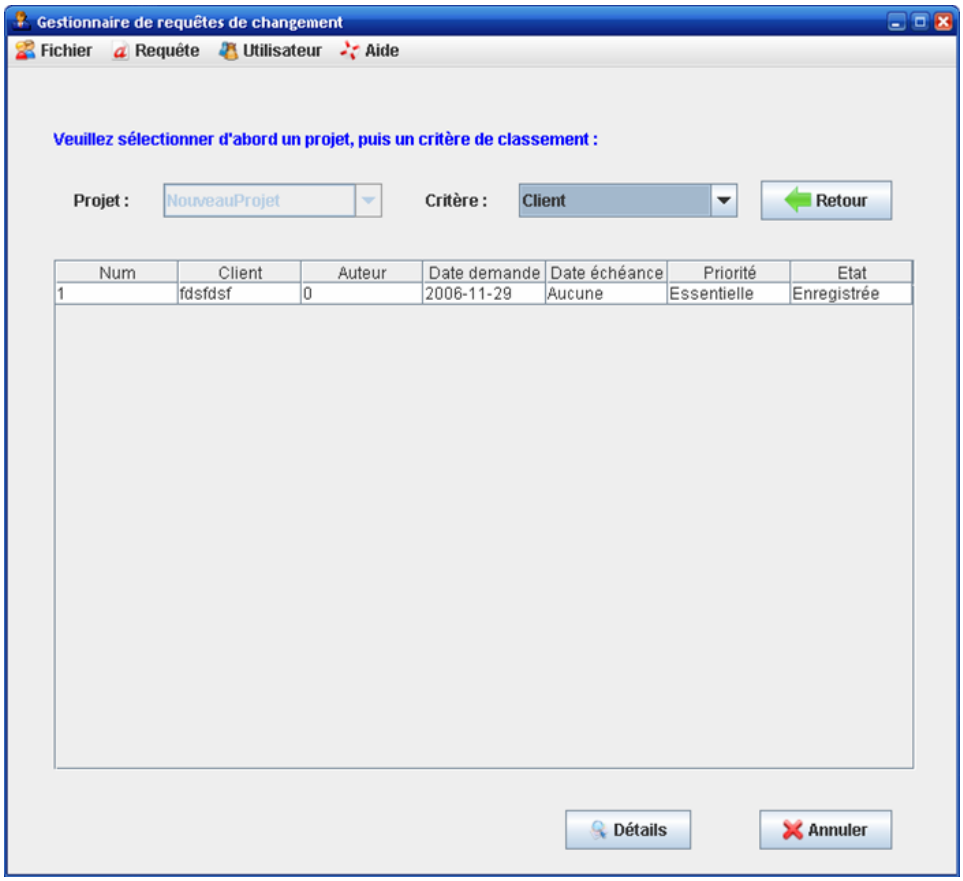

FIG. 13.6 – Détails d'une requête

13.5 Lister les requêtes

Pour modifier, supprimer ou consulter une requête, l'utilisateur et le chef de projet pourront lister toutes les requêtes d'un projet, et ce, par critère de classement, afin de sélectionner la requête désirée.

Le bouton retour permet de sélectionner un autre projet et un autre critère de classement.

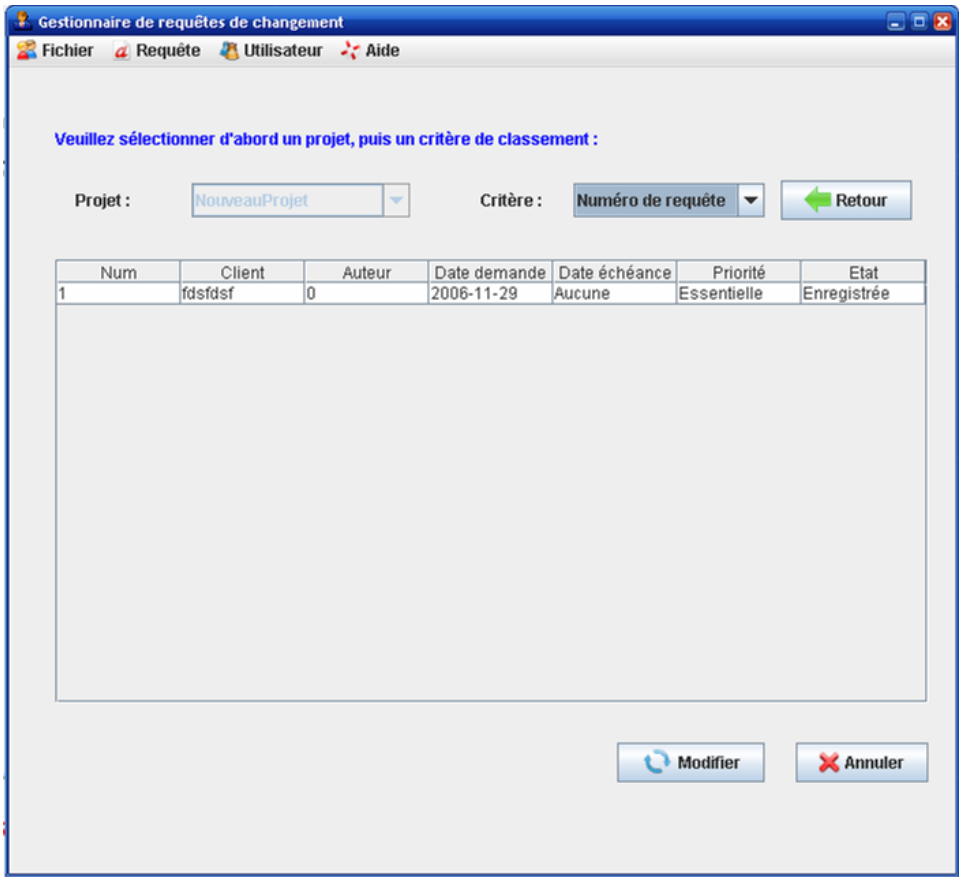

FIG. 13.7 – Lister les requêtes

Chapitre 14

Menu "Utilisateur"

14.1 Ajouter un utilisateur

Seul le chef de projet a la possibilité d'ajouter des utilisateurs au système.

Le chef de projet reçoit les demandes d'inscription au système par mail lorsqu'un utilisateur a, en spécifiant ses informations d'identification, initialisé modeX sur un nouveau poste de travail. Le chef de projet décidera ensuite d'ajouter ou non l'utilisateur concerné en spécifiant les informations de son compte.

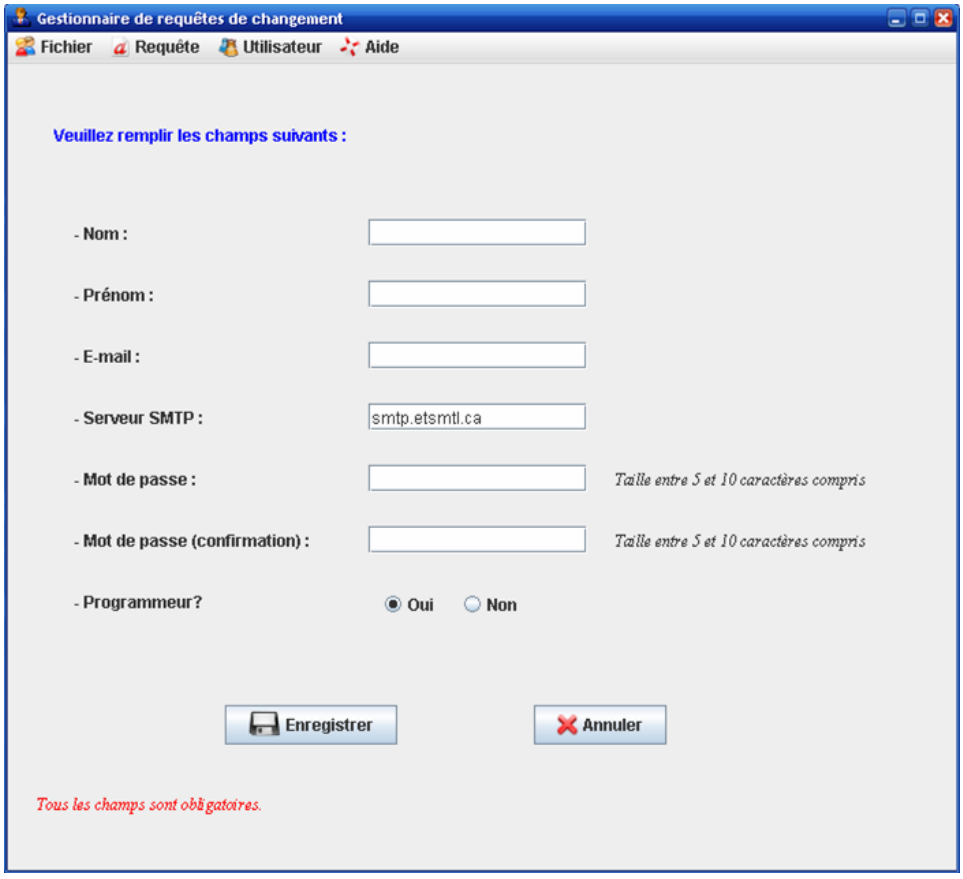

FIG. 14.1 – Ajouter un utilisateur

14.2 Supprimer un utilisateur

Seul le chef de projet a la possibilité de supprimer des utilisateurs du système.

Lors de la suppression d'un utilisateur par le chef de projet, l'utilisateur est averti par mail de la suppression de son compte.

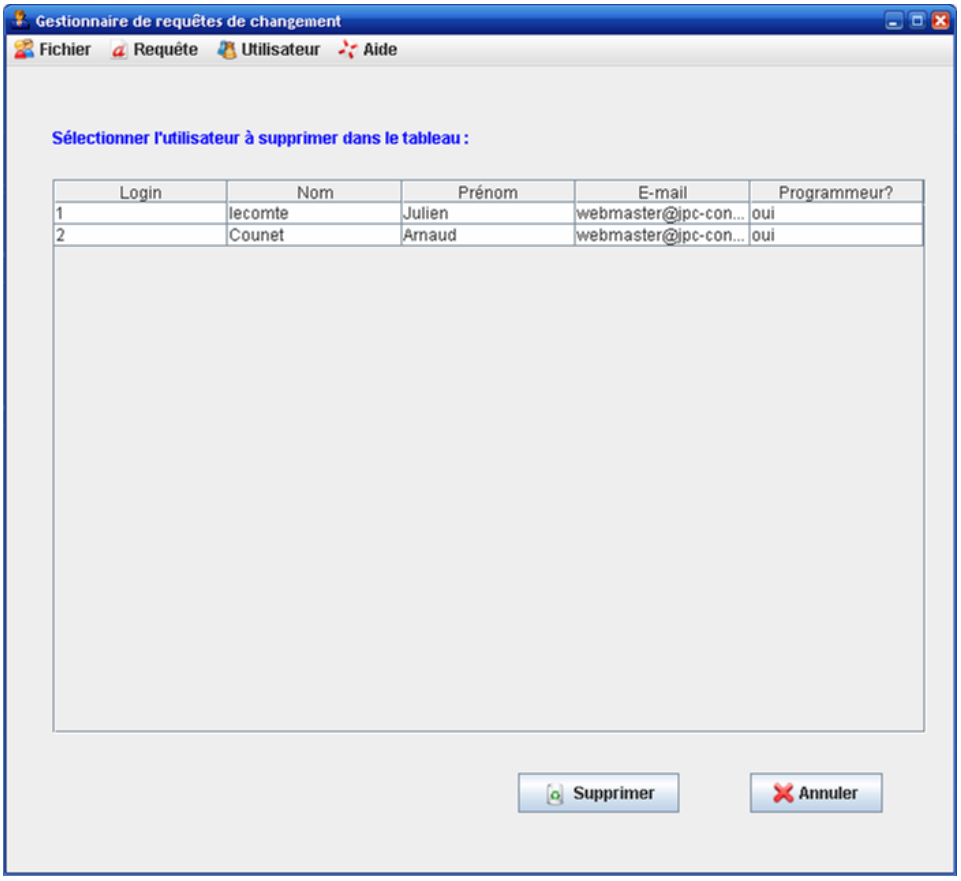

FIG. 14.2 – Supprimer un utilisateur

Si l'utilisateur était impliqué dans une ou plusieurs requêtes de changement, le chef de projet doit remplacer cet utilisateur par un autre, ceci afin d'éviter qu'une requête référence un utilisateur non présent dans le système.

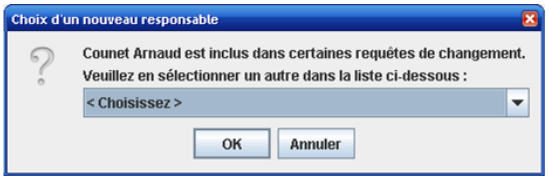

FIG. 14.3 – Utilisateur inclu dans une requête

Annexe E

Questionnaire de la micro-évaluation - Open Engineering

Réf. de l'entrevue	/
Personne interrogée	Pascal De Vincenzo
Position dans la société	Développeur et Gestionnaire de projets
Evaluateur	Antoine Robaeys
Date de l'entrevue	3 mai 2007
Type d'entrevue	Entrevue

Introduction

a. **Pouvez-vous me présenter brièvement votre société, ses activités, ses principaux acteurs (employés, clients, fournisseurs…) ?**

Open Engineering est une société de développement de logiciel, qui fait partie du groupe Samtech, et qui est composée de huit ingénieurs. Samtech est une spin-off, créée il y a 20 ans et Open Engineering est une spin-off, créée en 2001. Elles sortent toutes deux, du même laboratoire de l'université de Liège, et traitent des thématiques similaires.

Leur spécificité réside dans le développement d'un logiciel de simulation numérique qui permet de faire du prototypage virtuel. Dans ce dessein, Open Engineering dispose d'une application centrale unique, appelée OOFELIE, qui représente le logiciel de simulation afin de réaliser les calculs. La raison d'être d'Open Engineering est donc de développer OOFELIE, de l'industrialiser, après dix années de recherche à l'université, en générant des produits qui répondent à des besoins du marché et de continuer à la faire évoluer en la mettant à disposition de partenaires universitaires et dans des centres de recherche.

Pour faire évoluer OOFELIE, Open Engineering industrialise soit, les fonctionnalités existantes, en industrialisant les fonctionnalités récupérées de l'université et des centres de recherche, soit, en ajoutant des fonctionnalités spécifiques. Des contrats sont établis notamment avec l'ESA, avec la commission européenne, avec les centres de recherche, ou encore, avec tout autre client, tels que des bureaux d'étude, intéressés par un logiciel de simulation de contraintes multiphysiques.

b. **Pouvez-vous me décrire le contexte dans lequel évolue votre service à l'heure actuelle ? Quels sont les principaux projets en cours, quelles sont les priorités, etc. ?**

Open Engineering base l'ensemble de ses projets sur son application centrale, OOFELIE. Selon les besoins du client, ils adaptent alors l'interface graphique et réduisent les fonctionnalités au strict nécessaire. En général, Open Engineering essaye autant que possible d'anticiper l'avenir en développant de nouveaux produits.

Leurs plus gros projets se situent dans le développement de logiciels de simulation pour les MEMS, qui représentent pour eux un marché très important puisqu'ils en sont déjà à cinq produits développés plus un en cours. Leur application étant multidisciplinaire et multichamps, cela leur permet de se spécialiser dans le couplage, c'est-à-dire, lorsqu'il faut regarder tous les problèmes simultanément. Cela est fondamental dans le monde des MEMS car il faut alors prendre en compte un ensemble d'effet simultanément.

A. Gestion de la qualité

1. **Est-ce que votre entreprise a déjà entamé des activités qualités ?**
   - De quel type ?
   - Qu'est ce que cela vous a apporté ?

Open Engineering a commencé la certification ISO-9001, en mettant en place certaines procédures organisationnelles qui ont aidé à structurer la société, mais ils n'ont pas continué la phase de certification car ils n'ont pas de demande en la matière.

Leurs procédures sont acceptées par l'ESA, sans être pour autant certifié, qui impose cependant certaines contraintes supplémentaires, certaines normes de qualité, comme l'ECSS-E40 et l'ECSS-Q80, qui sont cependant adaptées pour s'appliquer de manière plus réaliste au développement des logiciels de simulation d'Open Engineering, compte tenu de sa petite structure.

Proposition	Rép.	Points
Non		0
Actions qualité ponctuelles	x	1
Procédures suivies par toutes les équipes		2
Procédures utilisées et adaptées en permanence		4

2. **Quels moyens mettez-vous en œuvre pour garantir la qualité des produits logiciels que vous développez ?**

Open Engineering pratique des tests d'intégration et des tests industriels, c'est-à-dire, des tests acceptés dont on connaît les bons résultats.

Les tests unitaires ne sont pas encore réalisés, mais des outils ont été trouvés qui permettraient éventuellement de les exécuter à moindres coûts.

Proposition	Rép.	Points
Cela dépend des développeurs		0
Cela dépend des chefs de projets		1
Méthode de validation disponible	x	2
Coordination par un Responsable A.Q.		4

En résumé, au niveau "Gestion de la qualité (A)", estimez-vous que ce qui est fait est efficace et permet d'atteindre les résultats attendus ?

Oui. Cependant, Open Engineering cherche toujours à s'améliorer en la matière. Lorsqu'ils constatent une possibilité d'amélioration, ils essayent de trouver des solutions quand le problème est réellement présent et s'ils ont les ressources pour le faire.

B. Relation Clients

3. **Formalisez-vous les exigences de vos clients ?**
   - Etablissez-vous un cahier de charges ?
   - Y a-t-il un document écrit reprenant les fonctionnalités demandées par le client ?
   - Ce document est-il revu par le client pour accord ?

Open Engineering établit un cahier des charges avec ses clients. Ce document officiel, rédigé en début de projet, représente le contrat entre les deux parties, et contient le planning des fonctionnalités demandées en fonction des besoins du client. Les exigences du client y sont donc inscrites, de commun accord avec ce dernier, afin d'expliciter les fonctionnalités désirées.

	Projets	
Proposition	Certains	Tous
Oui, en interne (unilatéral)		
Oui, avec approbation du client		x

4. **Comment les clients vous font-ils part des modifications à apporter aux fonctionnalités attendues ?**
   - Y a-t-il une trace écrite des demandes de modifications ?
   - Vos clients utilisent-ils un formulaire ou une procédure pour les demandes de modification ?

Open Engineering gère les demandes de changement grâce à des réunions organisées régulièrement avec le client, en les actant dans un PV qui permet de garder trace des conventions.

Certaines demandes sont cependant parfois plus formalisées, lorsque le développement est plus spécifique ou s'il y a de la recherche, c'est-à-dire, lorsqu'ils savent ce qu'il faudrait changer mais pas comment il faudrait le changer.

	Projets	
Proposition	Certains	Tous
Simple trace écrite		x
Formulaire et/ou procédure standard		

5. **Organisez-vous des réunions de coordination avec vos clients en cours de projet ?**
   - S'agit-il de réunions régulières ou organisées à la demande ?

Open Engineering planifie un ensemble de réunions avec le client, mais des réunions supplémentaires peuvent également s'organiser à la demande du client.

Lors de ces réunions, l'équipe de développement peut valider les exigences du client en lui présentant la version en cours d'itération.

	Projets	
Proposition	Certains	Tous
A la demande seulement		
Systématiquement, régulièrement		x

En résumé, au niveau "Relation clients (B)", estimez-vous que ce qui est fait est efficace et permet d'atteindre les résultats attendus ?

Oui.

C. Gestion des Sous-Traitants

6a. **Comment assurez-vous la sélection de vos sous-traitants ?**
   - Avez-vous des sous-traitants attitrés ou remettez-vous parfois ce choix en cause ?
   - Avez-vous une procédure bien définie pour la sélection de vos sous-traitants ?

Open Engineering est en contact avec un centre de recherche en Argentine, mais ce centre est plutôt un partenaire privilégié qu'un sous-traitant. Ce centre participe depuis le début au développement d'OOFELIE.
Les interactions sont nombreuses avec ce centre et il arrive qu'ils participent à des développements.

	Projets	
Proposition	Certains	Tous
Suivant des critères isolés comme le prix, la renommée …		x
Suivant une procédure de sélection rigoureuse (appel d'offre, critères de sélection)		

6b. **Comment assurez-vous le suivi du travail de vos sous-traitants ?**

- Avez-vous des contacts avec vos sous-traitants durant le déroulement des contrats ?
- Avez-vous des réunions régulières avec vos sous-traitants ?

Les contacts avec ce partenaire argentin sont rares car ceux-ci ont un accès direct aux sources, ils font le développement et ils préviennent Open Engineering une fois le développement terminé, qui se charge de réaliser les tests et de les contacter en cas de problème.

	Projets	
Proposition	Certains	Tous
Contacts occasionnels, à la demande		x
Réunions régulières		

En résumé, au niveau "Relation sous-traitants (C)", estimez-vous que ce qui est fait est efficace et permet d'atteindre les résultats attendus ?

Oui

D. Développement et gestion de projet

7. **Découpez-vous vos projets en phases/étapes ?**

- Adoptez-vous un cycle de vie formalisé pour vos projets ? Lequel ?
- Les phases sont-elles délimitées par la production d'un livrable, la validation d'un document ?

Les itérations se basent sur des petits cycles de vie en « V ». Selon les projets, les phases sont délimitées par la production d'un livrable, tel que des documents, des notes techniques,...En général, les projets internes ne font pas l'objet de réels livrables, seul le tutorial sera modifié avec l'ajout des fonctionnalités.

	Projets	
Proposition	Certains	Tous
Cycle de vie sans livrable		x
Cycle de vie avec phases délimitées par des livrables	x	

8. **Suivez-vous une méthodologie de développement pour vos projets ?**

- Avez-vous un guide décrivant la façon d'aborder un projet dans votre société ?
- Avez-vous une façon standard d'aborder certaines étapes du développement ?

Open Engineering se base sur certains concepts clés, tels que, le développement itératif par incréments, avec des temps de cycles très courts, en fonction des fonctionnalités à développer.

Ils utilisent plutôt des techniques informelles pour représenter ce que le client veut, comme des story cards, mais ne s'encombre pas de formalisme poussé, tel qu'UML.

Open Engineering utilise également parfois le refactoring et la relecture de code croisée, mais pas pour tous les projets car tous les développeurs ne sont pas capables de comprendre ce qu'un autre à fait dans un domaine plus poussé.

L'équipe a convenu d'un ensemble de règles de programmation adaptées d'un guide de bonnes pratiques selon les besoins.

	Projets	
Proposition	Certains	Tous
Méthodologie pour certaines phases du projet (documentée ou non)	x	
Méthodologie documentée pour l'ensemble du développement		

9. **Vos développements font-ils l'objet de plannings ?**
   - Faites-vous, en début de projet, une estimation du timing et des ressources à affecter aux différentes phases ?
   - Ajustez-vous le planning initial en cours de projet en cas de dérive ?

Open Engineering divise ses projets en séquences grâce à une ligne du temps définie à priori, où les parties à réaliser, en séquentiel ou en parallèle, sont délimitées par des milestones qui aboutissent à la rédaction d'un rapport, qui décrit les fonctionnalités introduites, de manière plus ou moins détaillée en fonction du projet.

Lorsqu'un non-sens ou une dérive est détecté, Open Engineering est capable d'ajuster le planning initial, et ce, avec l'accord du client, en lui proposant des solutions alternatives. Le but étant que le projet aboutisse.

	Projets	
Proposition	Certains	Tous
Planning établi mais jamais mis à jour		
Planning établi et ajusté en cours de projet		x

10. **Pouvez-vous connaître à tout moment l'état d'avancement et les ressources consommées par chaque projet en cours ?**

   - Les membres de l'équipe tracent-ils leurs activités ?
   - Utilisez-vous un outil de gestion de projet ?
   - Avez-vous à tout moment un état d'avancement ou devez-vous compiler des données éparses pour le constituer ?

Open Engineering organise des réunions mensuelles d'avancement des projets, qui leur permettent d'avoir une vision directe, tous les mois, de la façon dont les projets évoluent, afin de pouvoir mieux anticiper les éventuelles dérives.

Il leur est également possible de consulter les rapports de clôture des milestones afin de connaître l'état d'avancement.

	Projets	
Proposition	Certains	Tous
Suivi approximatif (état à constituer à la demande)		x
Gestion de projet effective (tableau de bord permanent)		

11. **Les membres de l'équipe se réunissent-ils régulièrement pour analyser les problèmes survenant en cours de projet ?**

   - Lors de ces réunions, essaie-t-on de déterminer la cause des problèmes détectés et d'anticiper les problèmes potentiels ?

Si besoin est, une réunion spécifique peut-être organisée afin d'analyser un problème particulier et de lui trouver une solution. En outre, des interactions permanentes ont lieu entre les membres de l'équipe, ce qui permet de résoudre certains problèmes « sur le tas ».

Le besoin en communication est donc très fort car il y a des personnes avec beaucoup d'expérience, qui connaissent parfaitement l'architecture de l'application, mais également des personnes moins expérimentées qui doivent pouvoir se référer aux personnes plus expérimentées.

	Projets	
Proposition	Certains	Tous
Réunions régulières assurant un traitement curatif des problèmes		x
idem + anticipation des problèmes		

12. **Comment faites-vous pour détecter et corriger les erreurs en cours de projet ?**
    - Avez-vous une phase de tests en cours ou en fin de projet ?
    - Effectuez-vous des séances de relecture pour le code, les documents d'analyse ?
    - Etablissez-vous des plans de tests en cours de projet ?

Open Engineering utilise l'interface graphique du groupe Samtech, qui a développé une interface utilisateur dédiée au calcul numérique et à l'interfaçage d'algorithmes de résolution. Une personne de chez Open Engineering se charge d'adapter cette interface pour qu'elle fonctionne avec OOFELIE. Les deux sont, dès lors, totalement découplés.

Vu qu'il s'agit de méthodes de résolution et d'algorithmes, et vu le découplage avec l'interface graphique, Open Engineering dispose de scripts qui représentent les tests d'intégration et les tests industriels, et qui permettent de savoir si la version en cours d'un projet est en bon état ou pas. Ils disposent d'un système de compilation et de test nocturne qui leur permet de qualifier OOFELIE toutes les nuits sur plusieurs plateformes et qui permet de générer des rapports qui informent de la bonne exécution ou non de ces tests. Cela leur permet donc de savoir si les développements réalisés sont portables et si certaines modifications apportées endéans les 24h ont provoqué un problème ou non.

Pour certains projets, il peut leur être nécessaire en cas de problèmes, de revoir le code ou de faire appel à des experts de l'université pour réfléchir sur la problématique.

	Projets	
Proposition	Certains	Tous
Vérification en fin de projet (test)		x
Vérification tout au long du projet (revues, inspections, plans de tests)	x	

En résumé, au niveau "Développement et gestion de projet (D)", estimez-vous que ce qui est fait est efficace et permet d'atteindre les résultats attendus ?

E. Gestion des produits

13. **Les produits de votre travail sont-ils identifiés de manière systématique ? Les différentes versions sont-elles gérées et archivées ?**
    - Pouvez-vous retrouver ou reconstruire facilement une ancienne version de vos logiciels et autres produits livrés au client ?
    - Existe-t-il un historique pour les documents intermédiaires (rapports d'analyse, plans de tests) ?
    - Utilisez-vous un outil de gestion de configuration ? Pour quels produits l'utilisez-vous ?

Open Engineering possède un système de gestion de version très facile qui permet de créer des branches de développement, c'est-à-dire, CVS, qui permet de travailler à plusieurs sur les mêmes fichiers. La flexibilité fournie par cet outil leur est nécessaire étant donné que OOFELIE est partagée avec les centres de recherche qui ont accès aux sources et qui ont, eux aussi, leur propre développement.

	Projets	
Proposition	Certains	Tous
Gestion de versions pour certains produits		
Gestion de versions pour tous les produits		x

14. **Utilisez-vous des documents type, des structures pré-définies pour les produits de votre travail ?**
    - Utilisez-vous un canevas de cahier des charges ?
    - Avez-vous une structure type pour vos documents d'analyse ?
    - Utilisez-vous des templates de programmation ?

Open Engineering utilise une documentation simplifiée qui est embarquée dans le code et qui est extraite à partir d'un outil, tel que Doxygen, qui permet d'extraire de l'information statique sur le comportement de l'application grâce à l'intégration, par le développeur, de balises dans le code source. En outre, certains rapports sont rédigés en clôture de certains milestones pour décrire les fonctionnalités de ce qui a été fait, et ce, de manière plus ou moins détaillée en fonction du projet.

Cette documentation n'est pas toujours suffisante pour des projets, tels que ceux qui sont établis avec l'ESA, qui leur impose des contraintes assez fortes. En effet, l'ESA exige une description complète de ce qui va être fait, comment cela va être fait, de l'implémentation, des tests réalisés,...Ce qui alourdit fortement les processus de développement de la spin-off.

	Projets	
Proposition	Certains	Tous
Certains produits structurés pour certaines phases		x
Structures ou documents type pour la plupart des produits		

En résumé, au niveau "Gestion des produits (E)", estimez-vous que ce qui est fait est efficace et permet d'atteindre les résultats attendus ?

Oui

F. Formation et gestion des ressources humaines

15. **Existe-t-il une politique de formation ?**
    - Planifiez-vous les formations que suivront vos informaticiens ?
    - Les formations sont-elles choisies en fonction des besoins actuels ou en anticipant les besoins futurs ?

Open Engineering a une politique de formation pour ses ingénieurs. En effet, il est préférable, selon eux, d'avoir un bon ingénieur, qui connaît bien sa théorie scientifique et qui va être formé à la programmation, plutôt qu'un informaticien, qui connaît bien la programmation mais qui a des difficultés au niveau de certaines connaissances scientifiques, comme par exemple, les connaissances physiques.

Les employés étant des ingénieurs, ceux-ci ne savent pas nécessairement programmer. Les formations proposées en interne sont donc réalisées pour leur permettre d'apprendre à programmer et de s'accoutumer à l'environnement de travail d'Open Engineering.

Des formations externes en langue et en gestion sont également réalisées si nécessaire.

Proposition	Rép.	Points
Pas de formation		0
Formations libres		1
Plan de formation suivant les besoins	x	2
Plan de formation anticipant les besoins, gestion des compétences		4

En résumé, au niveau "Formation - GRH (F)", estimez-vous que ce qui est fait est efficace et permet d'atteindre les résultats attendus ?

Globalement oui.

Remarques sur le questionnaire

c. **Avez-vous de remarques concernant ce questionnaire ?**
    - Certaines questions vous ont-elles paru confuses ou ambiguës ?
    - Y a-t-il des termes pour lesquels vous avez eu des problèmes de compréhension ?
    - Pensez-vous que ce questionnaire a fait un panorama suffisamment large ou y a-t-il d'autres sujets que vous auriez aimé voir traiter ?

Aucun problème de compréhension. Parfois l'ordre des questions a semblé un peu étonnant.

d. **Remarques de l'évaluateur**