UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
FRÉDÉRIC PAQUET

LES PROCESSUS D'INGÉNIERIE LOGICIELLE APPLIQUÉS DANS LE CONTEXTE DU DÉVELOPPEMENT DE JEUX VIDÉO

30 avril 2007

Bibliothèque
Paul-Émile-Boulet

UQAC
Université du Québec
à Chicoutimi

Mise en garde/Advice

Afin de rendre accessible au plus grand nombre le résultat des travaux de recherche menés par ses étudiants gradués et dans l'esprit des règles qui régissent le dépôt et la diffusion des mémoires et thèses produits dans cette Institution, **l'Université du Québec à Chicoutimi (UQAC)** est fière de rendre accessible une version complète et gratuite de cette œuvre.

Motivated by a desire to make the results of its graduate students' research accessible to all, and in accordance with the rules governing the acceptation and diffusion of dissertations and theses in this Institution, the **Université du Québec à Chicoutimi (UQAC)** is proud to make a complete version of this work available at no cost to the reader.

L'auteur conserve néanmoins la propriété du droit d'auteur qui protège ce mémoire ou cette thèse. Ni le mémoire ou la thèse ni des extraits substantiels de ceux-ci ne peuvent être imprimés ou autrement reproduits sans son autorisation.

The author retains ownership of the copyright of this dissertation or thesis. Neither the dissertation or thesis, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

RÉSUMÉ

Le développement de logiciels est un univers complexe. Plusieurs projets se réalisent avec la collaboration d'équipes multidisciplinaires. La compétition y est très forte, les changements technologiques et de personnels y sont fréquents.

Parallèlement à cela, il est difficile d'implémenter ou de supporter une structure de projet informatique lourde, telle que celles proposées par les méthodologies/normes/modèles de gestion usuelles (CMM, ITIL, ISO, SWEBOK, etc.). Néanmoins, les équipes doivent se munir d'un cadre de gestion couvrant les aspects essentiels de la gestion de projet informatique si elles ne veulent pas être la proie des problèmes récurrents de l'ingénierie logicielle.

Cette étude reprend la majorité des processus communs aux grandes méthodologies, les intègre dans un cadre de référence léger et établit une stratégie de mise en oeuvre pour le fonctionnement de chacun des processus et de la méthodologie qui en découle. Cette dernière devra être suffisamment simple pour être comprise et appliquée par tous les membres des équipes de développement, même ceux qui ne sont pas des informaticiens.

La conception d'outils n'est pas le but de ce travail. Il y est davantage question d'une approche légère basée sur une infrastructure technique simple, permettant à une équipe multidisciplinaire d'obtenir une base efficace pour amener rapidement les processus fondamentaux du développement logiciel à un niveau acceptable et compatible avec les différentes méthodologies plus lourdes.

REMERCIEMENTS

Tout d'abord, je tiens à remercier mes deux codirecteurs de recherche, MM. Sylvain Boivin et Jean Rouette pour leur implication, leur dévouement et leur encouragement tout au long du projet. De plus, sans leur complémentarité, il m'aurait été difficile de livrer un contenu que je voulais de qualité. Ils ont su comprendre l'intérêt que je portais au domaine, ainsi que les motivations qui m'ont amené à rédiger ce document et cela, afin de mieux m'encadrer et orienter mes recherches dans une direction qui m'était parfois inconnue.

Je tiens également à remercier, au nom de toute l'équipe de LyB, la direction du Département d'informatique et de mathématique de l'Université du Québec à Chicoutimi pour nous avoir permis de nous lancer dans ce projet. Grâce à l'ouverture, le support et la grande liberté démontrés par tous les membres de la direction, ce fut assurément l'une des expériences les plus enrichissantes que nous avons vécue tout au long du parcours scolaire et professionnel.

De plus, je tiens à exprimer ma reconnaissance à M. Noel Llopis, de chez High Moon Studios, pour avoir voulu partager ses recherches et répondre à mes questions, au studio Ubisoft Québec pour sa participation dans ce projet, et à M. Yvon Joly pour avoir grandement amélioré la qualité du français que l'on retrouve dans cet ouvrage.

Finalement, j'adresse mes remerciements les plus sincères à ma famille et à mes amis pour leur soutien inconditionnel tout au long de cette démarche.

TABLES DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1

INTRODUCTION

La complexité des projets de développement logiciels continue de croître. Il est devenu fréquent que des équipes multidisciplinaires doivent collaborer pour la réalisation de logiciels hautement spécialisés. Pensons simplement au domaine des jeux vidéo, qui demandent maintenant plusieurs années de développement et requièrent la participation de professionnels plus spécialisés et qui proviennent de différents horizons. Différents problèmes continuent d'affecter le développement de logiciels. Selon une étude effectuée par le *Standish Group International* [Standish G, 2001] en l'an 2000, portant sur plus de 200 000 projets informatiques aux Etats-Unis, 23% de ceux-ci furent des échecs et la moitié des 77% qui furent livrés présentaient soit un dépassement de budget ou un retard de livraison, ou encore avaient été livrés sans toutes les fonctionnalités planifiées au départ.

Malgré tout, ces résultats apparemment désastreux constituent une amélioration considérable sur le rapport précédent portant sur une étude faite en 1994. Le taux de succès a augmenté de 11,8% et le taux d'échec à diminué de 8,1%. L'étude a recensé les 10 facteurs qui ont le plus d'importance dans le succès d'un projet. Nous y retrouvons entre autres une infrastructure logicielle standardisée, une méthodologie formelle et surtout l'expérience du gestionnaire de projet. Cette expérience est décrite en termes de compétences dans l'élaboration des processus, dans l'utilisation des méthodologies et dans la capacité de bien structurer le travail.

Sans cadre ni méthode, les projets risquent de tomber dans plusieurs pièges souvent répétés en gestion de l'informatique. L'avancement des tâches n'est suivi qu'à travers la perception des personnes impliquées, ce qui peut entraîner un glissement dans les tâches et

donc reporter les échéances. Les membres de l'équipe ne savent pas trop quelle est la ligne directrice à suivre. Le gestionnaire ne peut avoir une image claire de l'avancement du projet à un moment précis dans le temps. Des parcelles de code sont perdues, des versions sont écrasées, des erreurs qui avaient été corrigées refont surface, ou alors la migration des ressources clés, interne ou externe, est mal gérée et l'information qu'elles détenaient est perdue ou difficile à retrouver.

Il existe plusieurs solutions sous forme de méthodologies, de normes ou de modèles telles que CMM, SWEBOK, ITIL, ou ISO. Cependant, en raison des lourdeurs de celles-ci, ainsi qu'au tempérament rebelle de plusieurs informaticiens qui les incitent à ne pas s'investir dans la production des artefacts nécessaires au bon fonctionnement de ces méthodologies. Plusieurs processus de développement furent forgés autour des valeurs de la philosophie agile, qui encourage entre autres, la simplicité et l'interaction entre les individus.

Il est toutefois nécessaire de trouver un moyen qui permet d'inclure les processus fondamentaux des méthodologies tout en favorisant la participation de l'ensemble des acteurs d'un projet. Ces derniers peuvent ainsi saisir davantage l'importance de leur coopération dans ces processus et le bénéfice qu'ils peuvent en retirer.

En 2004-2005, l'auteur a eu à gérer une équipe de conception de jeu vidéo d'une dizaine de personnes dans un projet de grande envergure avec un niveau de risque élevé. Plusieurs problèmes, tels que ceux mentionnés précédemment, sont survenus pendant cette

expérience, principalement en raison d'une structure trop souple, d'une infrastructure inadéquate et des méthodes de travail qui auraient pu être mieux supportées. Le contexte, une petite équipe multidisciplinaire nouvellement formée et devant être efficace rapidement, a rendu l'établissement d'une structure plus rigide très pénible. Bien que le projet fut un succès, les lacunes rencontrées nous ont poussé à approfondir davantage le sujet, à chercher des alternatives et à mieux comprendre l'univers de la gestion de projet informatique.

Les grands cadres de référence tels que CMM, ITIL, ISO ou SWEBOK, que nous verrons dans les chapitres suivants, proposent des processus qui doivent être considérés dans le développement ou l'entretien de logiciels.

Plusieurs firmes offrent une formation en entreprise sur ces méthodologies en expliquant le « quoi » et le « pourquoi » de tels processus. C'est d'ailleurs lors d'une formation sur les normes ITIL chez Sogique[1], que nous avons compris l'utilité de ce genre de bibliothèque d'infrastructure en TI, mais aussi les difficultés d'implantation de ce type de cadre de référence et leur lourdeur administrative.

Une étude parue dans le *Journal of Systems and Software* soulève que :

> Le problème présentement avec l'amélioration du processus logiciel SPI Software Process Improvement n'est pas le manque de standard ou de modèles, mais plutôt **un manque de stratégie efficiente pour implémenter ces modèles et standards efficacement** ... Dans la littérature, beaucoup d'attention est portée sur la question des activités à implémenter **plutôt que sur la manière de les implémenter.** Nous

[1] Sogique, société de gestion informatique : www.sogique.qc.ca

> croyons que l'identification des activités à implémenter n'est pas suffisante et que la connaissance du « **comment** » arriver à les implémenter est tout aussi nécessaire pour le succès de l'implantation d'un programme d'amélioration.
>
> [Niazi 2003] – *Notre emphase – Notre traduction*

Ce travail présente des moyens d'aborder la majorité des processus liés au développement de logiciel et se préoccupe des questions suivantes : Quelle stratégie de mise en oeuvre utiliser pour mettre en place ces processus ? Comment rendre leur utilisation simple et efficace ? Comment rendre une équipe de conception logicielle performante? Comment faire suivre les principes des méthodologies usuelles, sans pour autant faire crouler les principaux acteurs du développement sous la documentation et les tâches qui s'éloignent de leur expertise et qui réduisent leur intérêt?

L'un des objectifs de ce mémoire est d'offrir une somme suffisante d'information à toute nouvelle équipe voulant se lancer dans la conception de logiciel, et en particulier de jeux vidéo, permettant d'éviter plusieurs pièges fréquents en informatique, mais aussi à des équipes existantes de pouvoir remettre en question certains de leurs processus pour améliorer leur infrastructure de développement. Les stratégies de mise en œuvre présentées dans ce mémoire sont suffisamment souples pour être implémentées au sein d'une équipe de développement pilotée autant par une approche de programmation extrême que par le processus unifié. Les standards atteints grâce à l'implémentation de ces stratégies peuvent aussi être utilisés comme base pour une éventuelle intégration d'un cadre de référence plus rigide comme ISO, ITIL ou CMM.

Sans vouloir remplacer les cadres de référence usuels, la solution suggérée, une fois implémentée et utilisée en conformité avec sa procédure, pourra toutefois servir d'approche simplifiée. Ainsi, en appliquant les concepts présentés dans les prochains chapitres, une équipe devrait être en mesure d'atteindre une bonne partie des standards de base, sans y investir trop de ressources et de temps. Elle pourra alors augmenter sa productivité et améliorer la communication au sein des différents groupes d'intervenants. Elle réduira ses risques de pertes – défauts trouvés, versions, code, procédure, connaissances, et autres – et sera en mesure de mieux établir la ligne à suivre dans le projet. Ces objectifs sont atteignables grâce à notre méthodologie légère, qui implique beaucoup de relations entre les processus afin de mieux partager l'information et de la rendre plus efficace.

En résumé, le problème consiste à concevoir et intégrer une infrastructure de développement pour atteindre la plupart des standards des grandes méthodologies de génie logiciel au sein d'une équipe multidisciplinaire qui se doit d'être efficace et ne peut investir beaucoup temps dans la réalisation des artefacts de documentation. Les objectifs du travail de recherche quant à eux, sont :

- Cibler les processus de développement logiciel problématiques, touchant plusieurs équipes œuvrant dans la conception de jeux vidéo.
- Étudier les différentes pistes de solutions proposées par les méthodologies de génie logiciel usuelles pour répondre à ces processus défaillants.
- Trouver les solutions qui répondent le mieux au contexte du développement de jeux vidéo.
- Incorporer ces solutions dans la méthodologie de développement proposée avec une approche la plus systémique possible.
- Établir des stratégies de mises en œuvre pour l'intégration de chacun des processus.

Le chapitre 2 est un bref retour sur l'évolution du développement logiciel et compare deux méthodes très populaires depuis quelques années, soit les processus unifiés et la programmation extrême. Les grands cadres de référence seront analysés en second lieu en fonction des lacunes perçues par les chercheurs ou les membres de l'industrie.

Le chapitre 3 expose les problématiques relatives au contexte des entreprises de conception de jeux vidéo et les défis qu'ont à surmonter la majeure partie des nouvelles organisations de développement logiciel.

Le chapitre 4 porte sur les différents processus tirés des cadres de référence, présentés au chapitre 2, qui sont abordés tout au long de ce travail. L'impact de ces processus sur le développement logiciel, les difficultés reliées à leur implantation dans le contexte de l'industrie du jeu vidéo et les relations entre les différents processus, sont explorés en fonction de la littérature et des observations cumulées lors du projet de conception de jeu auquel nous avons participé.

Le chapitre 5 suggère une stratégie de mise en œuvre d'un cadre de référence léger et souple pour répondre aux processus décrits précédemment. Ensuite, on décrit et critique plusieurs types d'outils génériques disponibles qui pourraient être utilisés pour la mise en application d'une infrastructure de développement.

Le chapitre 6 conclut cette étude en présentant le potentiel futur du cadre de référence que nous présentons et les améliorations qui pourraient y être apportées. Le tout,

dans le but de continuer à améliorer la performance et la qualité du développement logiciel, tout en facilitant la cohésion des équipes multidisciplinaires.

Un guide d'utilisation, très simple, d'une vingtaine de pages, apparaît en annexe pour les concepteurs, les développeurs et les gestionnaires de projet, afin de supporter la stratégie de mise en œuvre et mieux exploiter les relations entre les outils.

CHAPITRE 2

GESTION DU DÉVELOPPEMENT LOGICIEL

Ce chapitre a pour objectif de faire une revue des différents courants du développement logiciel en mettant de l'avant les deux méthodes les plus répandues ces dernières années, soit le processus unifié et la programmation extrême. Suite à cela, les principales méthodologies et normes de gestion de projets logiciels seront abordées. Ces dernières serviront de références pour les chapitres suivants.

2.1 Méthodes de développement

Au milieu des années 90, tandis que les paradigmes de programmation se raffinaient et entraînaient une diminution considérable des erreurs de code, les chances qu'un projet informatique soit livré à temps, sans dépasser le budget, variaient encore entre 9 et 28%, en fonction de la taille et de la complexité des projets [*Standish G. 1999*]. Plusieurs méthodes de développement virent alors le jour. D'un côté, les processus unifiés, centrés sur l'architecture, faits de grandes itérations et pilotés par les diagrammes UML. De l'autre, les pratiques dites « agiles », où les programmeurs et les clients sont mis au centre du développement, comme dans le cas de la programmation extrême, la planification prend moins de place et se fait parallèlement aux livrables, qui doivent être très rapprochés les uns des autres. Ces deux familles furent développées principalement pour les mêmes objectifs, augmenter la communication entre les acteurs du développement logiciel et ainsi accroître la qualité globale. À la suite de cela, bien des méthodes hybrides sont apparues, s'adaptant à l'environnement des organisations et des conditions avec lesquelles ils devaient travailler. Des livres comme « *Balancing Agility and Discipline: A Guide for the Perplexed* » [Boehm-Turner 2003] ou « *Agile and Iterative Development : A Manager's Guide* » [Larman 2003], émettent des nuances et font des rapprochements entre les deux mondes.

2.1.1 L'approche agile

Bien que l'on voit fréquemment dans la littérature l'expression « méthode agile », il n'en demeure pas moins que l'approche agile est davantage une philosophie qu'un processus de développement. Cette philosophie est fondée principalement sur quatre valeurs qui se divisent en 12 principes. Voici ces quatre valeurs et quelques principes tirés du manifeste sur le développement agile : [Agile 2001]

Valeurs : *(Notre traduction)*

- L'équipe et les interactions plutôt que les processus et les outils;
- Une application fonctionnelle plutôt qu'une documentation compréhensible;
- La collaboration du client plutôt que la négociation d'un contrat;
- Réagir au changement plutôt que suivre une planification.

Principes: (*Traduction de Wikipédia*[2] *, que nous avons comparée au texte original*)

- Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles.
- Le changement est bienvenu, même tardivement dans le développement. Les processus agiles exploitent le changement comme avantage compétitif pour le client.
- Livrer fréquemment une application fonctionnelle, toutes les deux semaines à deux mois, avec une tendance pour la période la plus courte.
- Les artistes et les développeurs doivent collaborer quotidiennement au projet.
- Bâtissez le projet autour de personnes motivées. Donnez leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail.
- Une attention continue à l'excellence technique et à la qualité de la conception améliore l'agilité.
- La simplicité - l'art de maximiser la quantité de travail à ne pas faire est essentiel.
- Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'organisent par elles-mêmes.
- À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son comportement dans ce sens.

---

2 Traduction des différents principes agiles : http://fr.wikipedia.org/wiki/M%C3%A9thode_agile

Plusieurs processus de développement logiciel tels que la programmation extrême, Crystal Clear[3], Agile Unified Process[4] (AUP), SCRUM[6], ont adhéré aux valeurs agiles. Le contexte particulier de l'industrie du jeu vidéo que nous verrons dans le chapitre suivant et les problèmes recensés par leurs artisans. Ces problèmes, qui sont en relation avec les différents processus de développement logiciel que nous aborderons dans le chapitre 4, révèlent certaines failles de la philosophie agile. L'emphase de cette approche sur l'interaction humaine, plutôt que sur les outils et les processus, et le peu d'intérêt envers la documentation et la planification sont des facteurs pouvant causer certaines complications dans ce contexte. Par ailleurs, la philosophie agile est plutôt appréciée des programmeurs, qui ont mis au point cette approche en réaction au formalisme trop poussé des méthodes classiques [BI 2001]. L'un des objectifs de ce travail est d'établir une infrastructure de développement suffisamment souple pour leur plaire, tout en insérant des concepts d'ingénierie logicielle qui apporteront des solutions aux problèmes notés par les gens de l'industrie.

2.1.2 Programmation extrême (*Extreme Programming* ou *XP*)

En 1996, Kent Beck débute un projet de développement logiciel chez DaimlerChrylser[7] en implantant les nouveaux concepts qu'il avait élaborés auparavant avec la collaboration de Ward Cunningham, l'inventeur des systèmes wiki. À la suite de cette expérience, il définit quatre valeurs fondamentales de ce qui allait devenir la

---

3 Crystal Clear : http://www.developerdotstar.com/mag/bookreviews/cockburn_crystal_clear.html
4 AUP : http://www.ambysoft.com/unifiedprocess/agileUP.html
6 SCRUM : http://www.scrumalliance.org
7 Historique de la programmation extrême : http://www.extremeprogramming.org/Kent.html

programmation extrême, soit la communication, la simplicité, le retour d'information (feed-back) et le courage.

La communication est un point central dans la programmation extrême, tant auprès des développeurs que des clients. Dans le premier cas, la communication se fait en grande partie grâce à la programmation par paire. Cette technique consiste à placer deux programmeurs par ordinateur, l'un écrivant le code et l'autre validant le travail effectué en même temps. Dans un projet réalisé avec la programmation extrême, chaque matin commence par une courte rencontre où, debout, tous les membres de l'équipe sont placés en cercle. Cette façon de faire permet au gestionnaire du projet de discuter des problèmes, des solutions et de s'assurer que tout le monde suit la même ligne. De plus, la conception du système se fait au moyen des cartes CRC. L'utilisation des cartes CRC, l'acronyme de Classes, Responsabilités et Collaborations permet aux concepteurs de discuter du design du projet autour d'une table tout en schématisant les différents objets qui composeront le logiciel avec leurs responsabilités et leurs collaborations. Pour terminer, la communication entre les développeurs est facilitée grâce aux standards de programmation, aux conventions définissant les noms et aux règles de formatage du code qui doivent être suivies. Ces techniques permettent à tous de comprendre rapidement les fichiers sources des autres membres de l'équipe, puisque chaque fichier est construit de la même façon, avec les mêmes standards. Dans le second cas, la communication avec les clients est l'une des règles de la programmation extrême, puisqu'ils doivent être disponibles pour participer au projet à toutes les étapes, non pas uniquement à la conception, mais aussi au développement. L'implication du client dans le projet se fait avec les scénarios

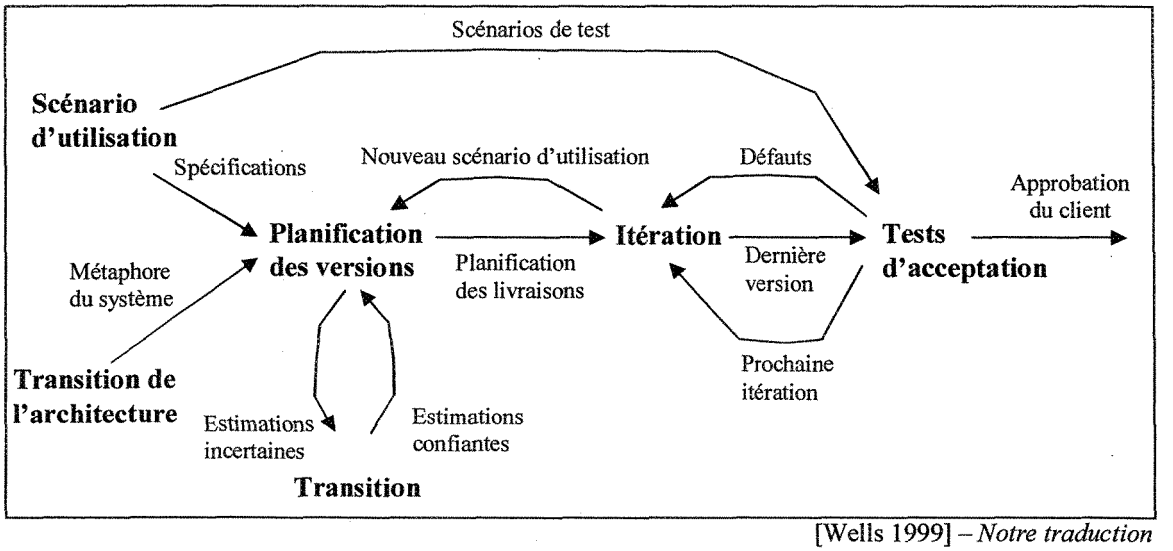
d'utilisation, comme de petites histoires qu'il écrit avec l'aide des développeurs pour définir les différents livrables, et sa participation est requise aux tests fonctionnels de chacune des versions qu'il reçoit régulièrement.

Contrairement aux méthodologies plus lourdes, la programmation extrême n'encourage pas les grandes analyses et les planifications à long terme. Selon ses créateurs, le changement est inévitable, ces étapes s'avèrent inutiles, car elles devront être recommencées « Seulement 10% de ces extras seront utilisés, vous aurez alors perdu 90% de votre temps » [Wells 1999] – *notre traduction.* La simplicité est l'une des clés de cette méthode. Le code doit être simplifié au maximum et les fonctionnalités développées doivent se limiter à celles qui étaient planifiées. Cette planification est elle aussi simplifiée, puisqu'elle repose uniquement sur des itérations d'une à trois semaines.

Le retour d'information, entre les développeurs et avec les clients, est une autre valeur fondamentale de la programmation extrême. Lorsqu'un défaut est détecté, il peut être corrigé par n'importe quel membre de l'équipe, puisque tout le monde est propriétaire de tous les fichiers sources, et un test unitaire est ensuite créé pour s'assurer que ce défaut ne réapparaisse pas plus tard dans le développement. Les tests unitaires permettent aux programmeurs d'altérer le code de leurs coéquipiers en s'assurant qu'il demeure fidèle aux anciens standards et limitent les chances que leurs changements fassent ressurgir des défauts qui étaient déjà corrigés dans les anciennes versions. Des tests d'acceptations, ou des tests fonctionnels, sont ensuite créés à partir des scénarios d'utilisation pour valider que l'itération à livrer réponde aux besoins du client.

La dernière valeur de la programmation extrême, le courage, se définit dans la disposition des membres de l'équipe à accepter et à encourager les changements qui, par ailleurs, sont inévitables. Les clients peuvent changer d'idée ou trouver des améliorations possibles tout au long du cycle de développement du projet. Le courage signifie ici la disposition de tous les acteurs impliqués à évaluer les impacts des modifications suggérées et de rapidement planifier le développement en conséquence. La figure 2.1 démontre le cycle de développement d'un projet réalisé en programmation extrême. On peut remarquer les différentes itérations et la planification de chacune d'elle faite à partir des scénarios d'utilisation. Le cycle démontre aussi que très peu de documentation est générée lors d'un cycle de programmation et l'analyse se fait tout au long du développement et non massivement en phase de démarrage du projet.

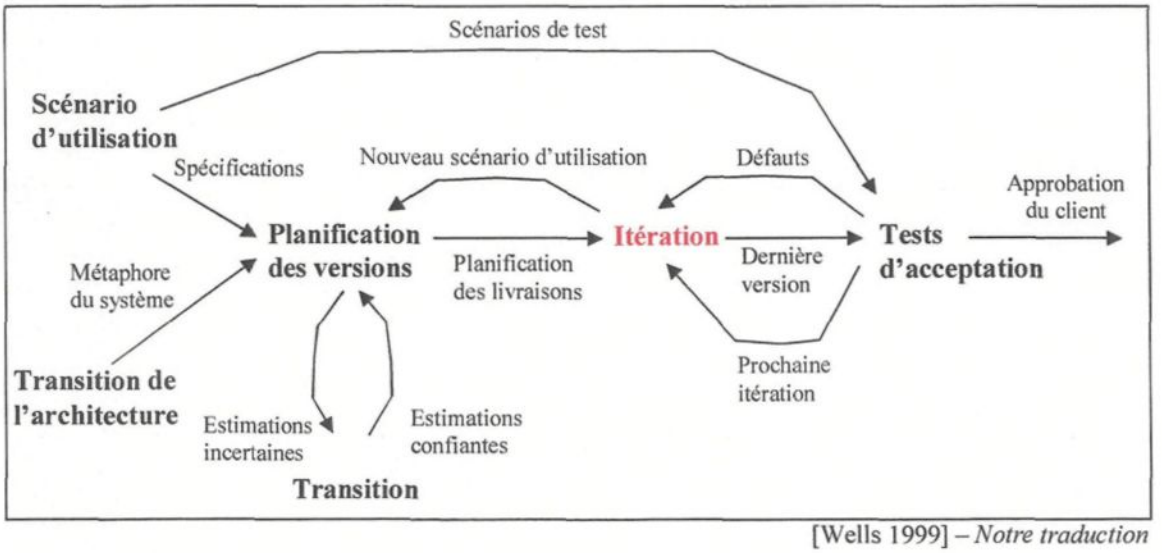
Figure 2.1 : Cycle de la programmation extrême



[Wells 1999] – *Notre traduction*

La dernière valeur de la programmation extrême, le courage, se définit dans la disposition des membres de l'équipe à accepter et à encourager les changements qui, par ailleurs, sont inévitables. Les clients peuvent changer d'idée ou trouver des améliorations possibles tout au long du cycle de développement du projet. Le courage signifie ici la disposition de tous les acteurs impliqués à évaluer les impacts des modifications suggérées et de rapidement planifier le développement en conséquence. La figure 2.1 démontre le cycle de développement d'un projet réalisé en programmation extrême. On peut remarquer les différentes itérations et la planification de chacune d'elle faite à partir des scénarios d'utilisation. Le cycle démontre aussi que très peu de documentation est générée lors d'un cycle de programmation et l'analyse se fait tout au long du développement et non massivement en phase de démarrage du projet.

Figure 2.1 : Cycle de la programmation extrême



[Wells 1999] – *Notre traduction*

2.1.3 Processus unifié (*Rational*)

Le processus unifié UP, de l'anglais : *Unified Process*, est une évolution de l'*Objectory Process*, l'une des premières approches objets développées en 1987 par Ivar Jacobson. RUP (*Rational Unified Process*) est probablement l'adaptation la plus répandue de la méthode UP, bien qu'il en existe plusieurs autres, telle que Enterprise Unified Process (EUP), Extreme Unified Process (XUP), Agile Unified Process (AUP), Two Tracks Unified Process (2TUP). Cette méthodologie, créée par *Rational Software Corporation*, qui est maintenant une division d'IBM, est une fusion entre le *Rational Process* et le processus unifié. Comme la programmation extrême, la méthode UP est itérative et incrémentale. Par contre, bien que plusieurs personnes sur Internet tentent de rapprocher le processus unifié et la philosophie agile, le AUT « *Agile Unified Process* », ou le XUP « *Extreme Unified Processus* », deux méthodes hybrides entre les deux mondes, certains fondements sont toutefois aux antipodes. Le processus unifié est tout d'abord centré sur l'architecture logicielle, se basant principalement sur les composants et l'utilisation du langage UML, un langage de modélisation unifié. L'élaboration des artéfacts, comme la documentation technique et l'évolution des modèles UML, doit être mise à jour tout au long d'un projet, ce qui diffère de la programmation extrême qui est plus orientée autour du code. Les itérations sont aussi habituellement plus volumineuses dans le processus unifié, puisqu'on suggère des durées entre deux semaines et six mois. [Smith 2003]

La planification d'un projet piloté par une méthodologie RUP est réalisée sur deux niveaux. Premièrement, une planification globale du projet, comprenant sa structure, les

différentes itérations qui seront développées avec leurs livrables, leur échéance, les ressources humaines et matérielles nécessaires à la réalisation de chaque itération et les risques qui y sont associés. En second lieu, une planification d'itération, faite pour chacune d'elles, est composée d'une description détaillée du travail à accomplir, des rôles de chaque membre, des activités et des artéfacts à produire avec leurs dates de début et de fin, et des critères mesurables pour évaluer les risques du projet, son avancement et son succès ou son échec.

On peut retrouver dans un document sur la gestion de projet logiciel avec RUP, publié sur le site web d'IBM, un échantillon des différents artéfacts qui doivent être produits tout au long d'un projet piloté avec RUP : [West 2003] – *Notre traduction*

- **Occasion d'affaires** – Décrit l'estimation des bénéfices et des coûts du logiciel, ainsi que les critères de succès du projet.
- **Vision** – Définit le projet, le marché qu'il vise et les fonctionnalités principales qui seront développées.
- **Cas d'utilisation** – Définit un service ou une fonction du système.
- **Prototype d'interface utilisateur** – Simule les interfaces utilisateurs, telles que définies par le client et pouvant être testées par celui-ci.
- **Liste des risques** – Décrit les risques à contrôler pendant chaque itération.
- **Modélisation** – Identifie les principales abstractions qui composent le modèle.
- **Test fonctionnel** – Examine la fonctionnalité requise pour répondre à une exigence particulière.
- **Version** – Collection d'activités qui définissent une version particulière.
- **Défaut/Amélioration** – Décrit le processus pour résoudre un défaut ou effectuer une amélioration en réponse à la rétroaction du client pour les prochaines versions.

Une méthodologie telle que le RUP est plus lourde à supporter, puisqu'elle couvre l'ensemble du cycle de développement et génère beaucoup de documentation. A priori, ces adaptations des processus unifiés sont mieux adaptées aux projets de moyenne et grande envergure que les méthodes dites « extrêmes ». Éric Germain et Pierre Robillard avancent à ce sujet : « Les avantages des activités de conception obligatoires et des artéfacts sont les plus évidents dans de grands projets. » [Germain-Robillard 2004] *(Notre traduction)*.

2.2 Cadres de référence de la gestion de projet

Peu importe le choix du processus de développement logiciel sélectionné, les gestionnaires de projet devraient se munir de méthodes ou de normes pour augmenter leurs chances de succès. Plusieurs auteurs s'entendent pour affirmer que la majeure partie des problèmes en développement logiciel, ne se trouvent plus dans la programmation, mais plutôt dans l'organisation et le contrôle du travail. Face au dépassement de budgets et d'échéances des projets informatiques, Pankaj Jalote de la firme indienne Infosys se demande [Jalote 2002] : « Quelle est la cause de tous ces échecs ? Bien qu'il y ait de multiples raisons, l'une des plus importantes est un management de projet inadéquat. »

Parallèlement au changement du côté des processus de développement, plusieurs cadres de référence, basés sur les meilleures pratiques de l'industrie, ont fait leur apparition. Certains furent initiés et élaborés directement dans le monde de l'informatique, tels que l'intégration du modèle d'évolution de la capacité (CMM), la bibliothèque d'infrastructure en technologie de l'information (ITIL), et plus récemment le guide du corpus des connaissances en génie logiciel (SWEBOK – *Software Engineering Body of Knowledge*),

tandis que les autres sont plus généraux et plus près de la gestion de projet, comme les normes de qualité ISO 9001-2000 ou le corpus des connaissances en management de projet (PMBOK – *Project Management Body of Knowledge*) du PMI (*Project Management Institute*). Nous survolerons ici les cadres de référence mentionnés ci-dessus, puisque nous en tirerons plusieurs concepts et processus.

2.2.1 CMM

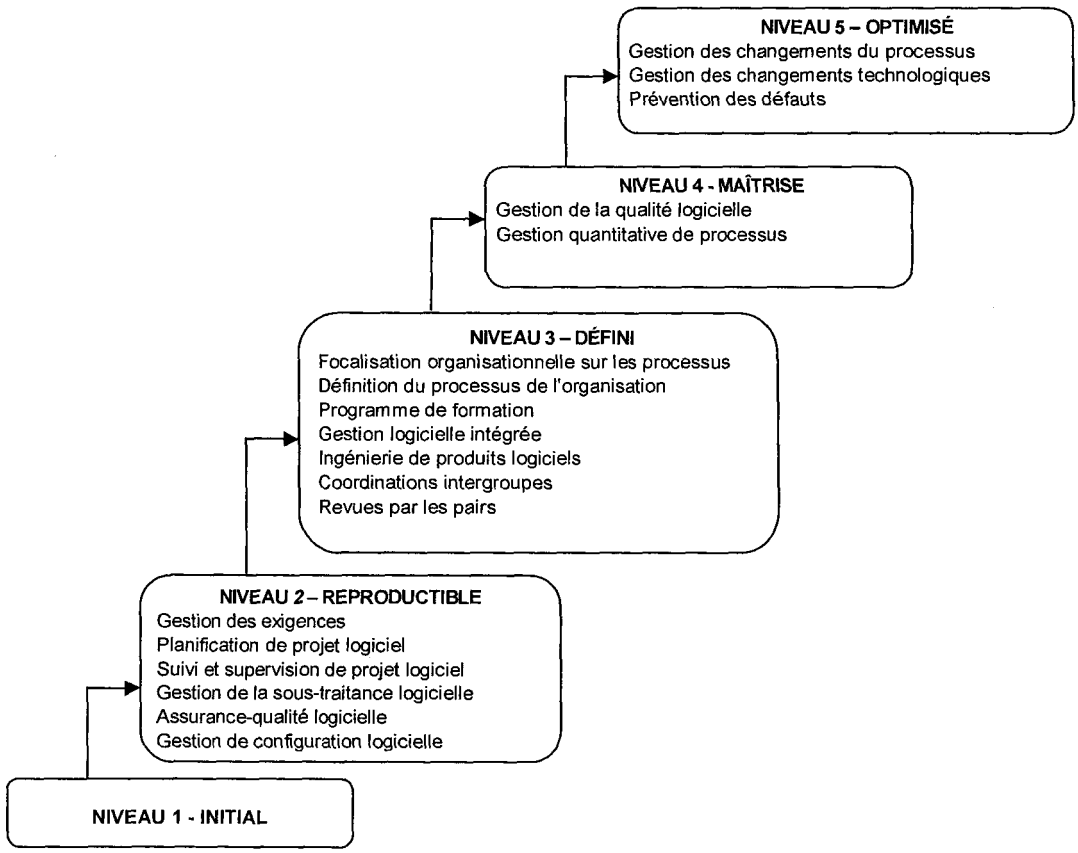
Le modèle d'évolution de la capacité (*Capability Maturity Model)* a été initié par le département de la défense américaine. Celui-ci a aussi financé la mise sur pied du SEI (*Software Engineering Institute*), propriété de la Carnegie Mellon University de Pittsburgh, afin de développer le modèle. Le CMM consiste à gérer la maturité des processus dans le développement logiciel. Une fois que tous les processus du développement logiciel sont établis, on doit améliorer leur capacité pour les rendre plus matures, donc plus performants. «/ .../ avec des processus matures, un projet est réalisé en suivant des processus bien définis. Dans ce cas, l'issue du projet dépend moins des personnes et plus des processus. » [Jalote 2002].

À l'inverse, James Bach, ayant utilisé CMM chez Borland, écrit dans la revue *American Programmer* [Bach 1994], que l'un des problèmes de ce modèle est qu'il repose entièrement sur les processus et ignore les gens qui les utilisent. De plus, il affirme que « Le modèle CMM encourage le transfert des vrais objectifs de l'amélioration des processus en une mission artificielle, soit celle de l'atteinte d'un plus haut niveau de maturité » (*notre traduction*). Il termine même son post-scriptum en affirmant que, selon

lui, la raison du succès du CMM est que le modèle offre « l'espoir et l'illusion d'une gestion, dans l'environnement chaotique du développement logiciel » *(Notre traduction)*.

Il existe cinq niveaux de maturité dans le modèle CMM qui déterminent le cheminement à suivre pour que les processus de développement logiciel gagnent en maturité. On peut distinguer les cinq niveaux de maturité représentés dans la figure 2.2.

Figure 2.2 : Niveaux de maturité dans le modèle CMM



[Jalote 2002]

On retrouve dans la littérature des critiques par rapport à l'étendu du modèle. Bien qu'on suggère de l'implémenter progressivement, le modèle CMM implique souvent trop

de ressources et d'effort pour les besoins réels des projets, ce qui a entraîné le développement de méthodes plus légères [Schwaber 2004].

Puisque ce travail porte sur l'encadrement de certains processus dans un environnement de base, nous nous intéresserons à plusieurs aspects des niveaux 2 et 3 du modèle de CMM. Bien que le SEI insiste sur le fait que le CMM ne soit pas une méthodologie, ni une norme, mais plutôt un modèle devant être adapté aux besoins de l'organisation, il n'en demeure pas moins que sa raison d'être est la même que les autres normes décrites dans ce chapitre, soit d'accroître la performance des équipes de développement logiciel en améliorant leurs processus.

2.2.2 ITIL

La détection des incidents et des problèmes est un point crucial pour plusieurs organisations et peut même être le point central de leur mission. Les centres d'appels en sont d'ailleurs un bel exemple. Voilà pourquoi dans les années 1980, l'agence britannique CCTA (*Central Computer & Telecommunications Agency*), maintenant l'OGC (*Office of Government Commerce*), créa ITIL (*Information Technology Infrastructure Library*), un cadre de travail commun, tiré des meilleures pratiques reconnues en matière de gestion des services des technologies de l'information.

Chez Sogique, l'implantation du premier processus d'ITIL, la gestion des incidents, a pris environ deux ans et coûté quelques centaines de milliers de dollars. Bien que ses dirigeants apprécient grandement la bibliothèque d'infrastructure ITIL, il n'en demeure pas

moins, que leurs plus petits partenaires, faute de moyens, n'ont pu poursuivre au même rythme leur implantation.

Bien que le sujet de cette étude ne porte pas sur les centres de service, nous nous intéresserons tout de même à certains aspects d'ITIL relatifs à la gestion des problèmes et aux relations entre ses processus.

2.2.3 ISO

Les normes ISO (*International Organization for Standardization*) sont probablement les normes sur les systèmes qualités les plus connues au monde. On peut lire sur le site Internet de l'organisation par rapport à ISO 9000 : « Elles sont mises en œuvre par quelque 760 900 organismes dans 154 pays » [ISO 2005]. Dans le livre *In Pursuit of Quality: The Case Against ISO 9000*, on retrouve 10 arguments pour ne pas utiliser ISO-9000 dont : « ISO 9000 n'a pas réussi à renforcer les bonnes relations client-fournisseur » et « ISO 9000 n'a pas encouragé les responsables à réfléchir différemment » [Seddon 1997] *(Notre traduction)*.

Pour les fins de ce travail, nous nous pencherons davantage sur les systèmes de management de qualité définis dans l'ISO 9001:2000. Cette norme établit huit principes de management de la qualité, qui dressent une ligne directrice pour l'amélioration des performances, dont *l'approche processus* et le *management par approche système*.

2.2.4 PMBOK

Le guide du corpus des connaissances en management de projet (Guide PMBOK) est le guide de référence produit par le PMI (*Project Management Institute*), à l'intention de ses membres, plus de 200 000 professionnels dispersés dans 125 pays.

> Le principal objectif du *Guide PMBOK* est de définir le sous-ensemble du Corpus des connaissances en management de projet qui est généralement reconnu de bonne pratique. « Définir » signifie proposer une présentation générale plutôt qu'une description exhaustive. « Généralement reconnu » signifie que la connaissance et les pratiques présentées sont le plus souvent applicables à la majorité des projets et que leur valeur et leur utilité font l'objet d'un large consensus. « Bonne pratique » signifie qu'il est généralement admis que la mise en œuvre de ces compétences, outils et techniques peut améliorer les chances de succès d'une large gamme de projets différents. Cette notion de bonne pratique ne signifie pas que la connaissance décrite doit être uniformément appliquée à tous les projets ; il appartient à l'équipe de management de projet de déterminer ce qui est approprié pour un projet spécifique. [PMI 2004]

Le PMBOK est davantage un guide de connaissances sur la gestion de projet qu'une méthodologie de gestion de projet. Puisqu'il est d'un niveau d'abstraction supérieur aux trois autres, on peut alors dire qu'il peut être davantage un complément qu'un concurrent. Nous nous intéresserons principalement aux chapitres portant sur la planification et le contenu et nous porterons un intérêt particulier sur le « management des communications », qui est l'un des objectifs que nous poursuivons dans l'élaboration de notre infrastructure.

2.2.5 SWEBOK

Le guide du corpus des connaissances en génie logiciel (SWEBOK – Software Engineering Body of Knowledge), est l'une des plus récentes références en génie logiciel. Le guide du SWEBOK, géré par l'Université du Québec, et plus spécifiquement par l'École de technologie supérieure et l'Université du Québec à Montréal, a été approuvé par le *IEEE Computer Society Board of Governors* en 2004. Voici les 10 domaines de connaissances du SWEBOK [IEEE 2004], ainsi que les chapitres dans lesquels ils sont développés :

- **Exigences du logiciel (*Chapitre 2*)**
- Conception du logiciel (*Chapitre 3*)
- Construction du logiciel (*Chapitre 4*)
- Essai du logiciel (*Chapitre 5*)
- Maintenance du logiciel (*Chapitre 6*)
- **Gestion de la configuration du logiciel (*Chapitre 7*)**
- **Management du génie logiciel (*Chapitre 8*)**
- Processus du génie logiciel (*Chapitre 9*)
- **Outils et méthodes du génie logiciel (*Chapitre 10*)**
- **Qualité du logiciel (*Chapitre 11*)**

L'emphase sur certains domaines de la liste précédente spécifie que ces derniers contiennent des points abordés dans ce travail. La correspondance entre ces domaines et les

processus proposés dans ce mémoire est présentée au chapitre 4 dans le tableau 4.1 (*Relations entre les processus*).

L'objectif de ce guide est d'orienter l'ingénieur logiciel dans les limites des connaissances de sa discipline. La préface du SWEBOK met d'ailleurs en garde les ingénieurs logiciels puisque dans la réalité, et plus spécifiquement lorsqu'ils sont en charge d'une équipe ou d'un projet, les connaissances qu'ils doivent acquérir et maîtriser vont au-delà du SWEBOK. « Les ingénieurs logiciels de pratique devront également savoir beaucoup de choses au sujet de l'informatique, du management de projet, et de l'ingénierie de système -- pour en nommer quelques-uns. » [IEEE 2004]. C'est d'ailleurs précisément dans cette optique de complémentarité que ce mémoire chevauche l'ingénierie logicielle et la gestion de projet et emprunte des concepts aux deux disciplines. Le SWEBOK demeure toutefois fort intéressant puisqu'il est le seul cadre parmi ceux présentés dans ce travail à s'aventurer vers des moyens, des méthodes et des outils pour supporter ses processus et il réunit une somme considérable de connaissances essentielles au développement de logiciels.

CHAPITRE 3

CONTEXTE DU DÉVELOPPEMENT DE JEUX VIDÉO

L'industrie du jeu vidéo évolue dans un contexte stimulant, tant au niveau créatif que financier, mais qui apporte un bon nombre de contraintes et de défis à surmonter. Loin d'être un nouveau phénomène, en 1971, le jeu Pong était vendu à plus de 100 000 exemplaires, il n'en demeure pas moins que l'industrie du jeu vidéo continue sa forte croissance. On peut d'ailleurs lire dans le rapport sur l'analyse du positionnement de l'industrie du jeu interactif au Québec, fait pour le compte d'Alliance NumeriQC :

> Les jeux interactifs se classent d'ailleurs au premier rang de tous les « médias » lorsqu'il s'agit de mesurer la croissance en termes de dollars dépensés par les consommateurs, avec un taux de croissance annuel composé de 9,3%, prévu sur la période 2000-2005 ... Par ailleurs, l'évolution démographique favorise aussi la demande de jeux, puisqu'on constate que cette forme de loisir et de divertissement recrute de plus en plus d'adeptes chez les adultes et les femmes.
>
> [Secor 2003]

3.1.1 Complexité du développement et de la gestion

Parallèlement avec la croissance du marché et l'avancement de la technologie, la complexité du développement de jeux, elle aussi, n'a cessé de croître. David Perry, président de Shiny Entertainment Inc., déclare par rapport au marché : « Autour du monde, les joueurs exigent des jeux immersifs, plus complexes et plus passionnants. Quand ils ont obtenu ce qu'ils veulent, ils récompensent l'équipe de développement avec des ventes massives. » [Irish 2005] - *Notre traduction.*

Toujours selon M. Perry, la croissance rapide du marché, ainsi que l'exigence accrue des joueurs, ont fait augmenter dramatiquement la taille des équipes de travail.

> Les équipes ont commencé à s'accroître d'une à deux personnes, puis de deux à quatre, puis de quatre à 10, et ainsi de suite. Maintenant les équipes

> comportent 30 à 60 personnes ou même 100 à 200 personnes. Cela crée beaucoup plus de complications que lorsque l'industrie est née.
> [Irish 2005] - *Notre traduction.*

En ce sens, le rapport commandé par Alliance NumeriQC ajoute que la «*sophistication accrue des technologies* » et les *« attentes de plus en plus élevées des consommateurs* » ont fait augmenter la taille des projets. La durée plus longue du développement et les budgets plus importants, rendent la tâche plus ardue pour les gestionnaires de projets et les risques d'échecs plus élevés.

De plus, en référant au rôle de développeur : « Il s'agit d'un [métier] très complexe qui exige à la fois une grande créativité et d'excellentes compétences technologiques et de gestion. » [Secor 2003]

3.1.2 Multidisciplinarité des équipes

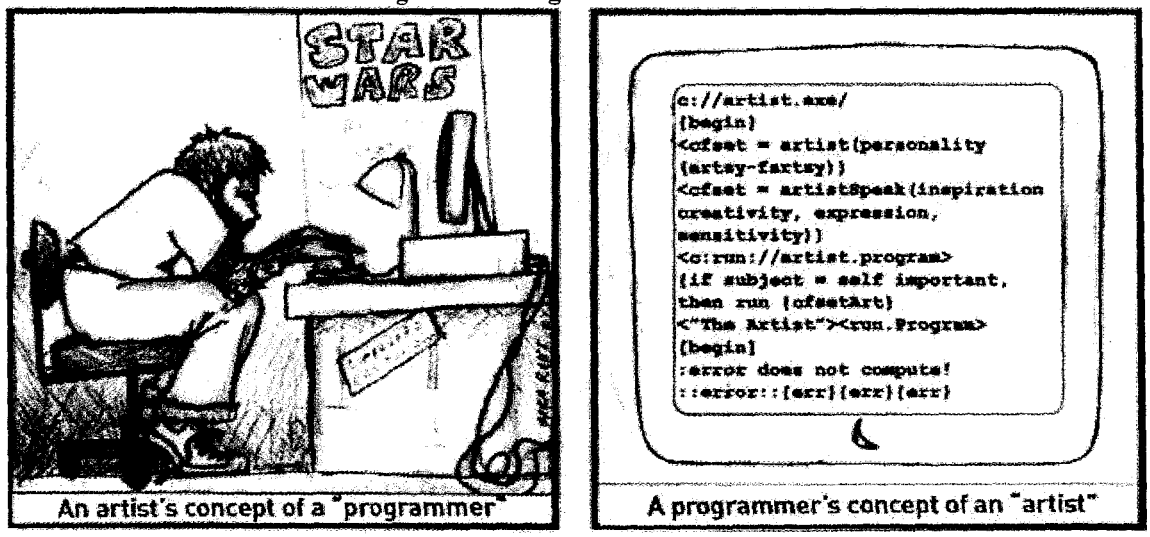
Bien que le développement d'applications logicielles en général, soit une activité relativement complexe, l'un des points qui ajoute un niveau supplémentaire de complexité au développement de jeux vidéo est la multidisciplinarité des équipes qui doivent collaborer. La création d'un jeu vidéo demande des informaticiens avec un haut niveau de compétences théoriques et pratiques, dont les caractéristiques personnelles recherchées tournent davantage autour de l'analyse rationnelle, la réflexion avec différents niveaux d'abstraction et la compréhension des interactions entre des systèmes complexes. Parallèlement à ce type de ressources, pour concevoir un « bon » jeu, il est impératif de

réunir une gamme diversifiée d'artistes de talent qui font preuve d'originalité, de créativité et ayant une vision artistique.

Malgré le fait qu'elle soit essentielle, cette collaboration amène un certain nombre de complications dues aux divergences des profils typiques de personnalité. Les deux principaux groupes ne partagent pas nécessairement les mêmes zones de confort et d'inconfort au travail, n'ont pas les mêmes modes d'apprentissage et ne perçoivent pas leur travail de la même façon. « Les programmeurs et les artistes, ce qui n'est pas une surprise, sont deux groupes littéralement opposés. » [Theodore 2005] - *Notre traduction.*

On peut d'ailleurs trouver dans la revue *Game Developper*, de janvier 2005, une illustration (figure 3.1) résumant très bien les perceptions qu'ont les artistes des programmeurs et vice et versa.

Figure 3.1 : Programmeur vs artiste



[Theodore 2005]

Alors qu'un groupe peut passer des systèmes Windows aux fenêtres de commandes Unix, l'autre préfère la simplicité et l'immersion des Macintosh. Il faut donc être conscient de ces divergences lors de l'établissement de l'infrastructure de travail, afin que cette dernière soit compatible avec ces environnements et ces modes de fonctionnement différents. Il est difficile d'envisager l'implantation d'un système quelconque pour un groupe d'artistes en exigeant d'eux l'apprentissage de plusieurs lignes de commandes. Une certaine réticence et de l'insécurité risquent de freiner leur implication dans ce changement plutôt brutal. D'un autre côté, on ne peut demander aux programmeurs d'utiliser uniquement les Macintosh pour uniformiser l'ensemble des machines de l'organisation, pour des raisons financières, techniques et pour éviter un mouvement de protestation généralisé.

Il faut aussi tenir compte d'un troisième groupe d'utilisateurs ayant à utiliser parfois l'infrastructure en place, soit les gestionnaires, les ressources humaines, les gens du marketing, etc. Il est vrai que la multidisciplinarité des équipes en ingénierie logicielle n'est pas un phénomène nouveau, ni exclusif au développement de jeux vidéo. Cependant, la particularité de cette industrie réside dans le fait que la proportion des artistes et des programmeurs est approximativement la même au sein des projets.

3.1.3 Pénurie de main-d'œuvre et fluctuation des équipes de travail

Au-delà des différences entre les groupes, un autre élément des ressources humaines vient compliquer les choses dans ce contexte particulier. On peut lire dans le cahier de

Jobboom « *Les carrières d'avenir* » du 16 janvier 2006, que certaines industries québécoises, dont celle de la conception de jeux vidéo, doivent freiner leur expansion en raison d'un manque de main-d'oeuvre qualifiée.

Cette rareté amène les gestionnaires à prendre en considération plusieurs facteurs importants. La pénurie de gens compétents alimente la chasse aux talents et rend parfois les équipes de projet vulnérables au départ de personnes essentielles, parties offrir leurs services aux plus offrants. Ces départs provoquent alors des remaniements des équipes en place qui retardent et complexifient la poursuite des activités. De plus, à l'intérieur des gros studios pouvant piloter plusieurs projets en même temps, il arrive que les équipes fluctuent fréquemment en raison des besoins de chacun des projets, ainsi que des décisions de la direction, comme l'abandon d'un jeu, l'accélération du développement en fonction d'une exposition ou d'une date de sortie. Certaines ressources sont retirées ou alors injectées dans les équipes pour atteindre ces objectifs et peuvent être utilisées au sein de plusieurs projets dans la même année, d'où l'intérêt de bien gérer les connaissances de l'organisation.

Où se trouvent les compétences clés dans l'organisation ? Y a-t-il des conflits d'interdépendance de projets pour le partage des ressources humaines ? Comment sont gérées les connaissances acquises ? Seront-elles perdues lorsque les personnes qui les possèdent partiront ? Ceci ne représente qu'un échantillon des nombreuses questions relatives à ce problème.

3.1.4 Changement fréquent de plate-forme technologique

Contrairement à plusieurs applications de gestion tournant depuis plusieurs décennies sur des environnements technologiques éprouvés sinon pratiquement inchangés, les logiciels de jeux interactifs doivent s'adapter à des plate-formes multiples et qui changent fréquemment. Les jeux sur PC doivent être testés avec plusieurs machines présentant des assemblages de composantes et des périphériques les plus disparates possible. Les jeux développés pour les téléphones cellulaires, quant à eux, doivent aussi être testés sur le plus grand nombre de modèles possible.

Pour ce qui est du développement de jeux sur les consoles, les tests sont un peu plus faciles puisque toutes les consoles d'un type sont à peu près identiques. Par contre, elles ne durent que quelques années pour ensuite être remplacés par des consoles de nouvelle génération souvent très différentes de leurs ancêtres, il faut alors recommencer la formation et refaire une suite d'outils de développement.

3.1.5 Méthodes de développement

Puisque les environnements technologiques changent fréquemment et que les projets ont un très faible taux de redondance, si ce n'est des suites et des ajouts, il est difficile, voire peu recommandé, pour les organisations dans l'industrie de la conception de jeux vidéo, d'intégrer un cadre de référence de développement dans son ensemble.

Suite au *Game Developers Conference* de 2004 en Californie, 150 personnes de l'industrie, dont 30-35% de gestionnaires de projets, se sont rencontrées pour une table ronde sur les méthodes d'ingénierie à améliorer dans le contexte particulier du développement de jeux interactifs « L'intérêt pour l'ingénierie logicielle continue de croître puisque les projets deviennent plus compliqués, les équipes grossissent et les budgets augmentent » [Llopis 2004] – *Notre traduction.*

On peut lire dans le sommaire de cette rencontre, que la majorité des intervenants n'utilisent pas réellement de méthodologie de développement, si ce n'est une approche de « *coder et corriger* » *(code and fix)*. De fait, 20% d'entre eux disent utiliser une approche plus contrôlée itérative et seulement 6% une méthode agile. L'absence ou le peu de méthodes de développement dans ce contexte particulier, augmente l'importance et démontre la nécessité de mettre en place une infrastructure et des processus détaillés pour éviter les pièges courants des projets en ingénierie logicielle.

CHAPITRE 4

PROCESSUS DE DÉVELOPPEMENT

Ce chapitre présente différents processus et discute de la problématique qui est relative à leur exploitation. Les processus furent choisis en fonction des difficultés rencontrées lors du projet de conception de jeu, mentionné dans le premier chapitre, mais surtout en fonction des besoins exprimés par les professionnels de l'industrie. Une revue de ces besoins, ainsi que leur source, débute ce chapitre. Ensuite, quatre processus, dont il est possible de trouver l'équivalent dans les différents cadres de références vus précédemment, sont présentés pour répondre à ces besoins. Pour terminer, tous les processus sont détaillés dans le but d'en préciser l'importance et d'identifier les éléments pouvant nuire à leur mise en œuvre informatique.

Chaque année entre 2002 et 2004, plus d'une centaine de représentants de l'industrie du jeu vidéo se sont réunies lors du *Game Developers Conference* pour discuter des principaux problèmes qu'ils perçoivent dans leurs processus de développement. Noel Llopis, architecte technique en chef chez *High Moon Studios*, fut l'organisateur de ces tables rondes et l'on peut lire sur son site Web[8], le sommaire des échanges et les résultats des sondages qu'il a effectués lors de ces rencontres.

On retrouve parmi les discussions des dernières tables rondes des sujets tels que le contrôle des documents, l'importance des processus, la gestion des défauts, le partage de l'information et la documentation.

---

[8] Site Web de Noel Llopis : http://www.convexhull.com/

En 2004, la discussion a débuté avec les avantages que l'ingénierie logicielle pourrait apporter au développement de jeux vidéo. Selon les participants, voici plusieurs raisons qui poussent les gens de l'industrie à s'intéresser davantage aux différents processus d'ingénierie:

- Réduire les risques.
- Réduire les défauts.
- Favoriser la réutilisation.
- Faciliter le développement multiplate-formes.
- Réduire les pertes de temps.
- Faciliter le transfert entre les projets.
- Améliorer le support pour leurs équipes de développement.

[Llopis 2004] – *Notre traduction*

Afin de trouver des pistes de solutions à ces problématiques, nous avons observé comment les cadres de références abordaient ces sujets. Cela nous a amené à regrouper ces préoccupations en un ensemble de quatre grands processus, soit la « ***gestion des documents et des modifications*** », la « ***gestion des défauts*** », la « ***gestion de la connaissance*** » et la « ***planification de projet*** ». Une fois mis en œuvre, ils pourront couvrir la majorité des points soulevés lors des *GameDeveloper Conferences*. Ces processus étant communs au développement logiciel en général, on les retrouve dans les principaux cadres de référence.

4.1 Relations entre les processus

Puisque ces processus sont tous traités par ISO, CMM, SWEBOK et ITIL, le tableau 4.1 identifie les relations entre les processus et les chapitres ou les points qui abordent chacun des sujets au sein des cadres de référence identifiés précédemment. Les

termes français des normes ISO sont ceux tirés directement de *l'Organisation internationale de normalisation*, ceux du SWEBOK proviennent du site officiel[9], tandis que la traduction des processus clés de CMM est tirée du lexique du *Centre de génie logiciel appliqué* (CGLA) du *Centre de Recherche Informatique de Montréal* (CRIM) [CGLA 1996] et celle d'ITIL à partir du *dictionnaire de la Gestion des Services des TI* du itSMF (*The IT Service Management Forum*) [Stay 2003].

Tableau 4.1 : Relations entre les processus

	ISO 9001:2000	CMM	SWEBOK	ITIL
Gestion des documents et des modifications	4.2.3 Maîtrise des documents 4.2.4 Maîtrise des enregistrements 7.5.5 Préservation du produit	N.2 - Gestion des besoins N.2 - Gestion de la configuration logicielle N.3 - Système d'ingénierie N.3 - Coordination intergroupe N.5 - Gestion des processus	C. 7 - Gestion de la configuration du logiciel C. 10 - Outils et méthodes du génie logiciel	• Gestion des changements • Historique des changements • Gestion des mises à jour
Gestion des défauts	4.1 Exigences générales 7.2 Processus client 7.5.3 Identification et traçabilité 8.2 Surveillance et mesure	N.2 - Système d'assurance qualité N.3 - Revue des pairs N.4 - Gestion des besoins N.4 - Gestion de la qualité logicielle N.5 - Prévention des défauts	C. 5 - Essai du logiciel C. 7 - Gestion de la configuration du logiciel C. 10 - Outils et méthodes du génie logiciel C. 11 - Qualité du logiciel	• Gestion des incidents • Gestion des problèmes • Gestion des niveaux de services
Gestion de la connaissance	4.2.1 Généralités 4.2.2 Manuel qualité 5.5.3 Communication interne	N.2 - Gestion de la configuration logicielle N.3 - Programme d'entraînement N.5 - Processus de gestion des changements	C. 2 - Exigences du logiciel C. 10 - Outils et méthodes du génie logiciel	• Historique des changements
Planification de projet	7.1 Planification 7.3 Conception - développement 7.5.1 Maîtrise de la production	N.2 - Gestion des besoins N.2 - Gestion de projet logiciel N.2 - Suivi et supervision de projet logiciel	C. 8 - Management du génie logiciel C. 10 - Outils et méthodes du génie logiciel	• Gestion des risques • Gestion des niveaux de services • Gestion de la performance

4.2 Gestion des documents et des modifications

L'élaboration de mécanismes de gestion des documents et des modifications efficaces est particulièrement importante dans la planification d'un projet de conception

[9] Site officiel du SWEBOK : http://www.swebok.org

logicielle, puisqu'elle touche directement au cœur de la production, soit le code source et les fichiers nécessaires au développement du produit/service.

Ce processus ne concerne pas les demandes et approbations de changements, mais plutôt la conservation de toutes les versions et modifications qu'a connues un fichier. Dans la littérature, on retrouve parfois le terme « gestion de la configuration ». Tout au long de ce document, nous utiliserons « gestion des documents et des modifications », que nous trouvons plus approprié puisque nous ne traiterons pas de l'ensemble de ce qui est couvert par la gestion de la configuration. De fait, la gestion de la configuration inclue l'identification des ressources matériels, progiciels ou logiciels à tout moment du cycle de développement selon des procédures spécifiques (*voir figure 5.1, gestion de la configuration selon le SWEBOK*). Dans ce travail, nous nous concentrerons principalement sur la gestion des versions des items de création de logiciel, ainsi que des documents reliés au cycle de développement. Mise à part le SWEBOK, cette portion de gestion de projet n'apparaît pas clairement dans les autres méthodologies. Toutefois, elle n'en demeure pas moins cruciale dans le domaine de l'informatique. D'ailleurs, suite à quelques échanges, lors de conversations ou via l'administration d'un questionnaire sur les processus de l'ingénierie logicielle, dans le contexte des jeux vidéo, avec les gens de l'industrie, l'amélioration du processus de gestion des documents et des modifications semble être directement au cœur des préoccupations des gestionnaires de projets. Plusieurs affirment même que ce processus est celui qu'ils désirent améliorer le plus rapidement possible.

4.2.1 Gestion efficace du contenu et des modifications

Une bonne maîtrise de la gestion des documents et des modifications permettra de :

- ✓ Suivre la trace des modifications d'un fichier (acteurs et espaces temporels).
- ✓ Revenir facilement sur une version précédente.
- ✓ Faciliter les sauvegardes globales du contenu.
- ✓ Mieux connaître l'avancement du développement (pour le gestionnaire).
- ✓ Éviter la perte de code source ou l'écrasement d'un mauvais fichier.
- ✓ Centraliser l'information de façon sécuritaire.

De plus, le contrôle des documents, et plus spécifiquement de toutes les données relatives au contenu et aux modifications, est l'un des points importants des normes ISO 9001, introduit depuis l'an 2000 :

> **4.2.4 Maîtrise des enregistrements** - Les enregistrements doivent être établis pour apporter la preuve de la conformité aux exigences du SMQ. Les enregistrements doivent rester lisibles, faciles à identifier et accessibles. Une procédure documentée doit être établie pour assurer l'identification, le stockage, la protection, l'accessibilité, la durée de conservation et l'élimination des enregistrements.
>
> [ISO 9001:2000]

La perte ou l'altération de code source est l'un des éléments qui peut nuire considérablement à la rencontre des délais en développement logiciel. Les données, stockées uniquement sur un poste client, peuvent être perdues lorsque ce poste fait défaillance, dans le cas de virus ou de bris par exemple. Une modification d'un fichier entraîne un problème majeur insoupçonné. Deux personnes travaillent en même temps sur

un même module sans nécessairement le savoir et créer alors des conflits. Un client préfère une ancienne version ou trouve que ses demandes ne correspondent plus à ses besoins. Un gestionnaire de projet veut savoir qui a fait un changement et quand il l'a fait.

Il existe encore beaucoup d'autres scénarios comme ceux-ci. D'ailleurs, plusieurs furent vécus lors du projet de conception de jeu vidéo auquel nous avons déjà fait référence, ce qui justifie amplement le processus de gestion des documents et des modifications.

4.2.2 Sauvegarde des fichiers

L'enregistrement de copies de sécurité est aussi un problème relié à ce processus. Un artisan, qu'il soit programmeur, artiste ou gestionnaire, peut travailler sur un ou des fichiers pendant un certain temps sans faire nécessairement une copie de sécurité. Dans ce cas-ci, une défaillance sur son poste de travail peut rapidement compromettre tous les efforts investis depuis sa dernière sauvegarde sur une unité externe. Bien que plusieurs organisations exigent la copie des dossiers de travail sur un serveur de fichiers, il est difficile de le valider et d'en mesurer l'habitude des membres de l'équipe. Pour des raisons fonctionnelles, plusieurs doivent travailler en mode local et ainsi, courir le risque de ne pas prendre le temps ou d'oublier de sauvegarder sur le serveur.

4.2.3 Environnement multiplates-formes

Puisque nous avons établi dans le chapitre précédent que les différentes équipes de conception de jeux vidéo utilisent diverses plates-formes de développement telles que Unix, Windows et Mac, la solution retenue devra être en mesure de supporter tous ces

environnements. La connexion d'un lecteur réseau sur un dossier partagé d'un serveur de fichiers apparaît donc plus problématique puisqu'il devient nécessaire d'installer un protocole serveur et des clients sur tous les postes, ftp ou ssh par exemple. Cela a donc pour effet d'ajouter des étapes de manipulation et des complications pouvant décourager certaines personnes.

4.2.4 Sécurité et confidentialité des données

Puisque l'industrie du jeu vidéo est un domaine hautement compétitif, les projets doivent être maintenus dans le plus grand secret. Parfois, dans le cas des entreprises de conversion de plates-formes technologiques, les projets ne peuvent être dévoilés que lorsqu'ils ont reçu l'approbation de l'organisation qui produit la version originale. La sécurité des données confidentielles est donc un aspect à prendre en considération dans l'élaboration d'un mécanisme de gestion de contenu et de modifications. Des niveaux de sécurité devront pouvoir être mis en place pour définir quels utilisateurs ou quels groupes d'utilisateurs pourront lire, modifier ou supprimer chacun des fichiers ou dossiers. La solution choisie devra être en mesure de garantir la protection des données lors du transport entre le serveur et un client distant par l'entremise d'un canal sécurisé, *ssh* par exemple.

4.2.5 Navigation entre les versions

En tout temps, les membres de l'équipe de conception devraient avoir accès à la fois à la dernière version stable de l'ensemble du projet, mais aussi à la version en cours de développement. L'objectif est d'optimiser le temps de recherche lorsqu'un nouveau problème survient. En ayant la possibilité de naviguer dans l'historique des changements,

on peut donc rapidement savoir si le défaut a été créé à partir des dernières modifications et isoler ainsi les variables de causes possibles, dans le cas, bien sûr, où il n'aurait pas été décelé avant.

De plus, lorsqu'une nouvelle version est dite stable, il est préférable de mettre en place des mécanismes qui engendrent rapidement les démarches nécessaires pour commencer les différents tests exigés par l'organisation, en fonction de l'endroit où se situe le projet dans son cycle de vie.

4.2.6 Solution client

Bien sûr, dans certains cas, pour des projets isolés ou des modules spécifiques, il se peut qu'une seule personne ait à travailler sur des fichiers. La procédure d'utilisation devrait être suffisamment simple pour inciter un développeur solitaire à employer les mécanismes mis en place; il doit par contre en voir l'intérêt, chose qui devra lui être présentée lors de sa formation sur le cadre de gestion.

> Bien que beaucoup de développeurs solos aient des difficultés pour s'ajuster à l'utilisation d'un logiciel de gestion de versions, une fois que vous commencez à l'employer fréquemment, non seulement cela deviendra une partie normale de votre processus de développement, mais vous pouvez commencer à vous sentir nerveux en travaillant sans lui.
>
> [OSA 2003] – *Notre traduction*

4.3 Gestion des défauts.

Avec la généralisation sur le marché des normes et des standards de qualité tel ISO 9001, il est pratiquement indispensable d'élaborer des mécanismes pour contrôler les défauts et les corriger rapidement, sans en perdre la trace. « Établir un mécanisme ou un

ensemble de mécanismes pour détecter le besoin d'effectuer une correction d'un processus ou d'un produit. » [SCE-CMM 2003] – *Notre traduction*

Un système efficace de gestion des défauts centralise et conserve l'ensemble des incidents et des problèmes, ce qui rapproche l'organisation encore un peu plus du contrôle des données préconisé par les normes ISO [ISO 9001:2000] présentées précédemment, qui indique que tout document doit être conservé. Cette mesure existe non seulement pour s'assurer qu'aucun défaut ne soit oublié, mais aussi pour alimenter la base de connaissance de l'organisation. Grâce à ce type de système, il devient alors possible d'analyser les tendances et les répétitions des erreurs dans chacune des étapes de conception. On peut alors trouver des failles ou des forces dans nos équipes et disposer d'informations nécessaires à la prévention des défauts, tel que suggéré dans le cinquième niveau du CMM.

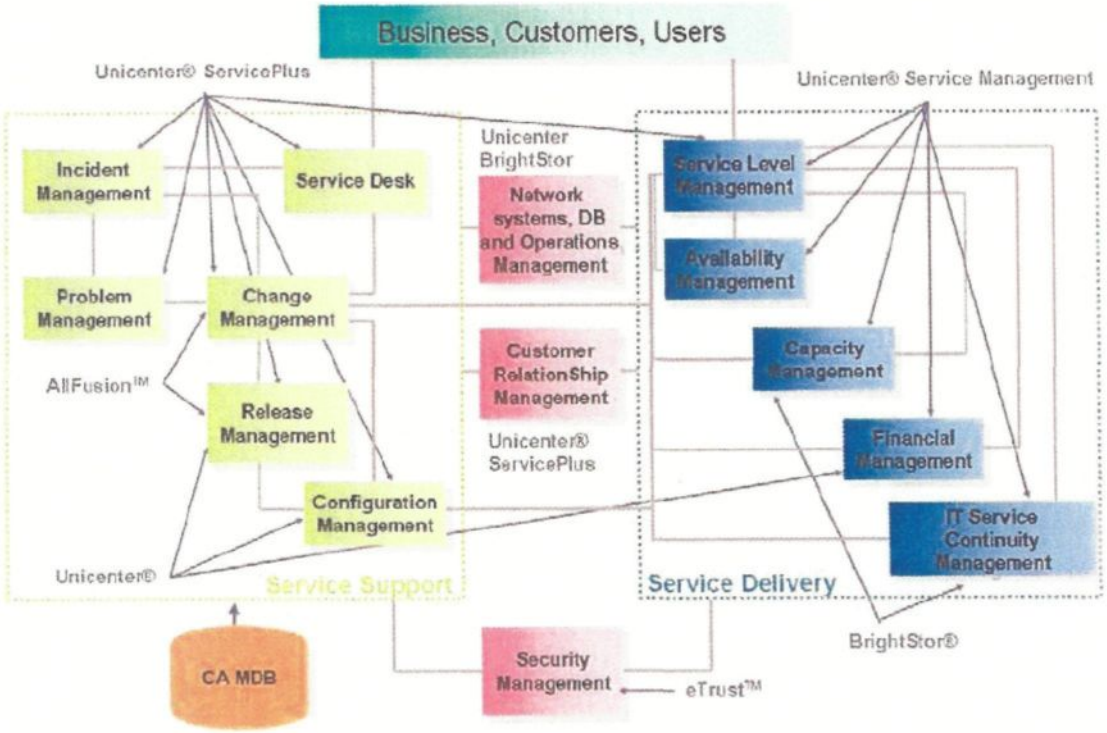
Une bonne maîtrise de la gestion des défauts permet :

- ✓ D'améliorer la qualité globale du produit/service.
- ✓ D'augmenter la communication des membres de l'équipe et celle avec les clients.
- ✓ D'impliquer davantage tous les acteurs du projet dans sa réussite.
- ✓ D'accroître la satisfaction du client.
- ✓ D'éliminer les défauts perdus et redondants.
- ✓ D'obtenir un meilleur portrait du travail effectué par les membres de l'équipe.

Les processus ITIL nous donnent une bonne représentation des interactions entre les différents aspects de la gestion des défauts. La figure 4.1 représente l'ensemble des processus décrits dans ITIL. Aux fins de ce travail, nous nous intéresserons uniquement au

côté gauche de la figure sur le « *Service Support* ». Nous pouvons constater les relations nombreuses entre le centre de services, la gestion des problèmes, des changements et des versions.

Figure 4.1 : Modèle des processus ITIL [10]



[Stephenson 2004]

Nous pouvons remarquer dans cette figure qu'il existe une nuance entre la gestion des incidents et des problèmes dans le modèle ITIL. Voici les définitions des deux termes selon le dictionnaire de la gestion des services des TI de l'itSMF (*The IT Service Management Forum*) [Stay 2003] (*Notre traduction*) :

---

10 Les outils présentés dans la figure 1, sont des outils pour les grandes organisations, aucune version pour fin de tests ou charte de prix n'est disponible. Des outils de substitution seront donc présentés dans les chapitres suivants.

Incident : *Un événement ne faisant pas partie de l'opération normale d'un service et qui cause ou peut causer une perturbation ou une réduction de la qualité des services et de la productivité du client.*

Problème : *Origine inconnue d'un ou plusieurs incidents existants ou potentiels.*

Dans le cas d'un service, la distinction est importante puisque plusieurs éléments externes au fournisseur peuvent être la cause de l'incident : un lien Internet, le routeur, configuration réseau, etc. Pour les produits, le terme *défaut* sera utilisé dans tous les cas.

4.3.1 Difficultés liées au processus

La détection d'un défaut doit pouvoir se faire par tous les acteurs à partir des principaux artisans, c'est-à-dire les programmeurs et les artistes, jusqu'aux clients, certains jouant le rôle de bêta-testeurs et les autres espérant la correction d'un défaut gênant. Peu importe la phase du cycle de vie où est rendu un jeu, on doit être en mesure d'assurer la collecte des défauts détectés et ce, même lorsque le jeu est terminé et sur les tablettes.

Les méthodes pour signaler un défaut doivent bien sûr être différentes selon qu'il est trouvé par un membre de l'équipe en phase de développement ou par un joueur suite à la sortie du jeu. Les joueurs ou les testeurs de type « boîte noire », soit ceux qui ignorent le fonctionnement interne du jeu, ont à décrire le défaut, celui-ci devra être classé de façon à ce que chacun d'eux soit connu de l'équipe de développement et qu'aucun ne soit perdu. Pour ce faire, le système doit permettre de classer les défauts et de les assigner aux personnes qui peuvent les prendre en charge, tout en donnant l'opportunité de connaître à

tout moment, l'état du défaut, qui a effectué des interventions pour le corriger, quelles furent ces interventions et à quel moment.

Les défauts doivent être représentés de façon unique dans le système, mais il faut se garder la possibilité de signaler le nombre de répétitions; si plusieurs personnes différentes rapportent le même problème, cela signifie qu'il a sûrement une certaine importance. De plus, la communication des défauts aux membres de l'équipe de développement est un point à ne pas négliger. Des mécanismes efficaces permettant d'alerter la personne ou le groupe à qui l'on vient d'assigner un défaut sont primordiaux.

4.3.2 Détection interne et externe

Des cellules indépendantes sont créées dans plusieurs organisations, afin de tester les versions produites par les équipes de développement. Les tests qu'ils ont à exécuter sont habituellement élaborés par leur supérieur ou le gestionnaire du projet. Les tâches doivent être claires et précises. Plusieurs équipes utilisent des listes à cocher pour s'assurer que tous les points sont passés en revue.

Il n'y a pas que les équipes de tests qui ont pour mandat la détection de défauts. Tous les membres d'une équipe doivent partager le même souci de qualité et donc partager les fonctions d'assurance-qualité. La détection d'un défaut ne doit pas être perçue dans l'organisation comme la « recherche de poux » entre collègues, mais plutôt une préoccupation de qualité globale partagée par tous et chacun. « Un élément important au sujet des non-conformités est de donner à tout le personnel approprié la liberté pour

identifier les items, activités ou processus non conformes et de les encourager à suggérer des améliorations. » [ISO 9001 :2000] – *Notre traduction*

Puisqu'il est difficile de pouvoir tester un logiciel sur tous les systèmes possibles et dans tous ses recoins, il est de plus en plus commun de voir des versions *Bêta* offertes gratuitement sur Internet pour permettre aux utilisateurs de tester les logiciels, ce qui constitue une pratique courante dans le monde des jeux vidéo.

À l'intérieur du SWEBOK et des normes de l'IEEE, antérieures au SWEBOK (1997), voici les objectifs que l'on décrit de ces tests ou évaluations, internes ou externes à l'organisation, désignés sous le terme de « *walkthroughs* » : [IEEE 2004]

- ✓ Trouver des anomalies.
- ✓ Améliorer le produit logiciel.
- ✓ Considérer des implantations alternatives.
- ✓ Évaluer la conformité aux normes et exigences.

Le système doit aussi être suffisamment adaptable pour être en mesure de créer lui-même des billets. Ainsi, par exemple, lorsqu'une nouvelle version apparaît dans le processus de gestion des documents et des modifications, on peut automatiser la création d'un billet afin que cette version soit testée dès sa sortie. Le billet est ce que nous considérons l'occurrence créée dans le système lorsqu'un nouveau défaut est détecté. Il contient les informations relatives à ce qui a causé le défaut, aux étapes franchies pour le régler et à son état de résolution. Une description plus exhaustive du billet apparaît dans le chapitre suivant.

4.3.3 Contrôle du temps investi dans la gestion des défauts

Il est aussi à noter que les gestionnaires aiment habituellement connaître la répartition du temps alloué à chaque tâche par les membres de leur équipe. Ils sont ainsi en mesure de voir le nombre d'interventions faites auprès de la clientèle par un employé, le nombre de corrections qu'il a effectué, le nombre de défauts qu'il a trouvés versus ceux qu'il a créés. Un membre de l'équipe provoquant plusieurs défauts, mais qui en découvre peu, pourra alors être rencontré pour qu'on lui suggère de s'impliquer davantage dans l'assurance-qualité, ou de voir avec lui des solutions pour qu'il améliore certains points, principalement par la formation et l'obligation de livrer des tests fonctionnels et par la diminution de la rapidité de développement au profit de la qualité.

D'un autre côté, un employé que l'on blâme de performances ordinaires pourrait se justifier en démontrant le faible taux de défauts associés à son travail, des corrections qu'il a apportées pour en régler certains, ou des erreurs qu'il a signalées. Encore faut-il que ce soit dans l'exercice de ses fonctions. Ces derniers points, bien qu'ils n'apparaissent souvent pas dans l'échéancier, sont néanmoins très importants, puisqu'ils impliquent les membres de l'équipe dans le contrôle de la qualité du produit à tous les niveaux de conception.

4.3.4 Base de données des défauts

Peu importe la solution retenue, la base de données des défauts en est le point fondamental. Les recherches doivent y être rapides et efficaces, afin d'inciter les membres de l'équipe et les clients à la consulter lorsqu'une anomalie survient. Elle doit être suffisamment malléable pour pouvoir y insérer directement les procédures nécessaires à la

solution, afin qu'un membre non informé du défaut ou de l'incident puisse accompagner rapidement et efficacement un client, ou mieux, lui donner la possibilité de le faire seul.

Cette base de données doit aussi être accessible via le système de gestion de la connaissance, car elle contient une bonne partie de la mémoire de l'organisation sur les erreurs du passé.

4.4 Gestion de la connaissance

La gestion de la connaissance (*Knowledge Management*) est le processus le plus complexe à aborder puisqu'il est intimement lié à l'organisation et à la façon dont l'information y est contrôlée. Comme l'explique Tom Tobin, dans son manifeste sur les causes de succès de la gestion de la connaissance, paru dans le *Government Computer News*, « La gestion de la connaissance n'est pas une technologie ou une série de méthodologies ... elle est plutôt une pratique ou une discipline qui implique des personnes, des processus et des technologies » [Tobin 2005] – *Notre traduction.*

Dans un marché où l'information et les compétences sont capitales, la gestion des connaissances occupe un rôle de premier plan. L'une des célèbres citations d'Andrew Carnegie[11] résume bien cette importance : « Le seul capital irremplaçable d'une organisation est la connaissance et les habiletés de ses membres. La productivité de ce capital dépend de l'efficacité de ces personnes à partager leurs compétences avec ceux qui peuvent les utiliser. »[12] *(Notre traduction)*

---

[11] Andrew Carnegie : Industriel et philanthrope américain, ses dons ont permis la création des *Carnegie librairies* (2500 bibliothèques au États-unis) et de la *Carnegie Mellon University*

[12] http://www.providersedge.com/kma/in-line_frames/quotes_inline.htm

4.4.1 Mémoire organisationnelle

Tel que mentionné auparavant, les normes ISO indiquent qu'il est avantageux de garder une trace électronique de tout ce qui a été décidé et fait, mais encore faut-il être en mesure de retrouver ces informations efficacement. La gestion de la connaissance consiste à conserver une mémoire organisationnelle pouvant résister aux changements à l'intérieur des équipes de travail, mais surtout, offrant la possibilité aux membres des équipes d'hériter de l'apprentissage technique et stratégique de l'entreprise. Cette gestion des connaissances et de la mémoire est d'autant plus importante dans le contexte des jeux vidéo en raison des changements de plate-forme et des générations d'un jeu.

Une bonne maîtrise de gestion de la connaissance permet de:

- ✓ Faciliter l'intégration d'un nouvel arrivant dans un projet.
- ✓ Accélérer les phases de démarrage d'un projet.
- ✓ Minimiser la perte de connaissance occasionnée par le départ de personnes critiques.
- ✓ Utiliser les connaissances explicites codifiées dans le passé pour augmenter l'efficacité dans l'avenir.

Pour atteindre ces objectifs, l'élaboration du processus de gestion de la connaissance doit s'établir en trois phases, qui seront développées dans la section « difficultés liées au processus » :

1. Uniformisation de l'information.
2. Implantation d'une infrastructure informatique adéquate
3. Instauration de mécanismes de recherche.

4.4.2 Connaissances tacites et explicites

Depuis les travaux d'Ikuko Tanaka en 1985, on distingue dans la littérature deux types de connaissances, les connaissances « tacites » et les connaissances « explicites ». Les connaissances tacites représentent le « savoir-faire » des individus. Elles sont difficilement transférables puisqu'elles proviennent de l'expérience, des aptitudes et des raisonnements d'une personne. Ikujiro Nonaka, dans un article publié dans le Harvard Business Review, caractérise les « connaissances tacites » via une analogie : « Un maître artisan, après des années, peut développer une expertise au bout de ses doigts, mais il arrive souvent qu'il ne puisse articuler les principes scientifiques ou techniques derrière ce qu'il sait. » [Nonaka 1991] – *Notre traduction*

Les connaissances explicites sont celles pouvant être codifiées. Elles proviennent d'informations brutes comme la documentation, le fichier de code source, la liste de caractéristiques du jeu, ou plus élaborées comme la synthèse de réunion, la procédure, les méthodologies. La gestion de la connaissance consiste à conserver le plus efficacement possible les connaissances explicites de l'organisation et d'inciter les personnes qui y travaillent à convertir une partie de leurs connaissances tacites en connaissances explicites.

Par exemple, un chef pâtissier ayant 30 ans d'expérience peut, grâce à ses connaissances, ses compétences et son intuition, développer un nouveau gâteau qui pourrait faire sensation. Un nouveau venu dans le métier ne pourrait probablement pas arriver à un tel résultat aussi rapidement. Par contre, le chef pâtissier pourrait décider d'écrire la recette du gâteau, ce qui permettrait à l'apprenti de le reproduire de façon quasi identique, mais il

pourrait aussi écrire la démarche et les raisons qui l'ont poussé à tenter ces combinaisons. Voilà deux exemples de codifications de connaissances tacites en connaissances explicites, qui auraient probablement un effet bénéfique sur la formation et la performance de l'apprenti. Face à cette distinction entre les deux catégories de connaissances, Nonaka énumère les quatre modèles d'acquisition de connaissances dans une organisation :

Tacite à tacite : Lorsque deux personnes discutent entre elles d'un sujet quelconque.

Explicite à tacite : Le gestionnaire qui regarde des données financières de l'organisation pour baser ses décisions.

Explicite à explicite : Une personne utilise des données de l'organisation pour en faire un rapport.

Tacite à explicite : Le maître pâtissier qui rédige sa recette.

[Nonaka 1991]

Le processus de gestion des connaissances vise à faire tendre l'acquisition de connaissances vers les deux derniers modèles, donc de converger vers des connaissances explicites.

4.4.3 Difficultés liées au processus

Où se trouve l'information que l'on cherche ? Voilà la principale question à laquelle nous devons répondre. La recherche doit pouvoir se faire en parcourant le plus grand nombre de sources possibles, sans pour autant franchir les règles de confidentialité établies par l'organisation. Pour chacune des phases de l'élaboration du processus de gestion de la connaissance, plusieurs points doivent être abordés.

4.4.4 Uniformisation de l'information

Avant même de penser à l'informatisation de l'information, il est primordial d'en connaître la structure et d'en uniformiser les méthodes de classement et de stockage. Il ne faut pas attendre l'installation de l'infrastructure pour établir des procédures d'utilisation et implémenter des standards pour collecter les données. Sinon, les membres de l'équipe pourraient développer de mauvaises habitudes, mais surtout, l'information enregistrée dans la période entre l'installation des outils et celle de la formation des membres aux procédures, risque d'être difficilement réutilisable. Cette étape est ardue puisqu'il faut tenter de prévoir les besoins d'information futurs de l'organisation et des différentes éventualités de changements qui peuvent chambarder les structures en place.

J'en prends pour preuve le cas Sogique que j'ai vécu. La formation sur ITIL et les méthodes pour classer les billets d'incidents sont arrivées un an après que l'application fut mise en place. Tous les billets ouverts depuis le début de l'opération, plus de 8 000 billets, durent être reclassés pour uniformiser la classification.

Cette uniformisation doit se répercuter sur tous les projets, afin qu'ils aient tous la même structure. Ainsi, un membre devant se joindre à une équipe peut se familiariser rapidement avec l'environnement du projet puisque l'infrastructure supportant les différents processus, entre autres les spécifications du projet, le concept de haut niveau, le code source, les défauts non résolus, est la même d'un projet à l'autre. De plus, lorsque les membres sont à l'aise avec une structure, le lancement d'un nouveau projet peut se faire plus rapidement.

4.4.5 Implantation d'une infrastructure informatique adéquate

Bien que la gestion de la connaissance ne repose pas uniquement sur l'informatique, il n'en demeure pas moins que l'infrastructure technologique mise en place pour les autres processus aura un impact direct sur l'efficacité de la recherche d'information. Tout d'abord, il est inapproprié de penser à une solution qui devrait chercher localement l'information sur les postes des membres de l'équipe de développement. Les recherches monopoliseraient le réseau et la sécurité serait difficile à gérer. Les connaissances devront être puisées sur le ou les serveurs qui hébergeront les différentes sources d'information comme les serveurs de courriels, de gestion des défauts, de gestion des documents et des modifications, de planification ou de documentation.

Les courriels font partie de l'information d'une organisation. Si on exige que toutes les décisions ou les argumentations soient électroniques ou confirmées de façon électronique, le courriel devient alors une pièce justificative de premier ordre. Par exemple, un assistant-producteur veut connaître les raisons qui ont poussé un designer de niveaux à faire un choix esthétique, ce dernier lui répondra probablement verbalement, l'assistant producteur devrait alors lui demander de lui énumérer brièvement ses raisons par courriel, pour en garder la trace ou s'il vient à l'oublier.

Un membre de l'équipe doit pouvoir chercher rapidement dans ces courriels envoyés ou reçus, mais aussi hériter des courriels que les autres ont jugé utiles de rendre publics et accessibles aux membres de l'équipe ou de l'organisation.

4.4.6 Instauration de mécanismes de recherche

Même avec une infrastructure informatique adéquate et des données bien enregistrées au bon endroit, sans mécanisme de recherche performant la gestion des connaissances ne peut être efficace. Cette étude ne se penche pas sur les algorithmes qui pourraient être mis en place pour effectuer cette prospection d'information au cœur des données que contient une organisation, mais plutôt sur l'implantation d'une solution, ou d'un élément de solution, qui puisse permettre d'atteindre rapidement des résultats satisfaisants.

L'efficacité du processus de gestion de la connaissance sera liée à ceux des mécanismes de recherche. L'efficacité des mécanismes de recherche doit être mesurée en fonction de trois facteurs :

- La rapidité à trouver l'information.
- La qualité de l'information trouvée.
- Le respect des règles de confidentialité établies.

La recherche de l'information doit se faire suffisamment rapidement pour inciter les usagers à l'utiliser. Si ce n'est pas le cas, plusieurs utilisateurs prendront ce temps pour chercher à tâtons en prenant le risque de ne rien trouver. Thomas B. Riley, président et directeur général du *Commonwealth Centre for Electronic Governance*, dans son second rapport sur la gestion des connaissances et la technologie, dit à propos de la recherche d'information : « Une des principales difficultés de la gestion des connaissances consiste à trouver l'information nécessaire rapidement et à peu de frais ». [Riley 2003]

Au-delà de la rapidité de recherche, il faut tout de même que l'information trouvée corresponde à l'information recherchée. Comme le dit le proverbe de René Thom[13], « trop d'information tue l'information », il faut réussir à trouver l'information pertinente recherchée et se limiter le plus possible à celle-ci. L'utilisateur doit donc être en mesure de configurer les bornes à l'intérieur desquelles il désire effectuer ses recherches afin de ne pas être submergé par des données inutiles.

Dans l'industrie du jeu vidéo, il n'est pas rare que plusieurs projets soient menés de front dans la plus grande confidentialité pour tenter d'éviter l'espionnage industriel et le vol d'idées. Les mécanismes de recherche mis en place doivent prendre en considération les droits d'accès à l'information des personnes qui utilisent le système, afin qu'un utilisateur ne puisse recevoir des données pour lesquelles il n'a pas le droit d'accès.

4.4.7 Connaissances : coût et pouvoir

Implémenter un processus de gestion des connaissances implique une modification de la structure de communication au sein des équipes. Il est probable que certains membres résistent à ce changement puisqu'il est possible qu'ils perdent une partie du pouvoir organisationnel, qu'ils détiennent en fonction de leurs connaissances ou des informations dont ils disposent. Kathleen Kelley Reardon citée dans le *Havard Business Review* à propos de la politique et de la recherche de pouvoir à l'intérieur d'une organisation : « La politique est aussi présente dans les bureaux que l'éclairage fluorescent » [Reardon 2005]. Thomas B.

---

[13] René Thom : Mathématicien et philosophe, fondateur d'une branche des mathématiques modernes : « la théorie du chaos ».

Riley énumère différents types de coûts sociaux pour construire, gérer et mettre à niveau un système de gestion de la connaissance.

L'utilisation des connaissances comporte trois types de coûts : [Riley 2003]

- Les coûts de disponibilité pour mettre les connaissances à la disposition des utilisateurs.
- Les coûts d'accessibilité pour généraliser l'accès aux connaissances.
- Les coûts d'applicabilité pour permettre d'utiliser efficacement les connaissances.

La personne en charge de la mise en place de ce processus doit être consciente des coûts, mais surtout de cette résistance, afin de s'assurer que tous les niveaux hiérarchiques y participent pour le bien de l'équipe et de l'organisation.

4.5 Planification de projet

La planification de projet est un problème depuis les débuts de l'ingénierie logicielle. Contrairement à la production d'autres biens, l'élaboration des échéances d'un développement de logiciel ne peut être calculée uniquement en fonction d'un nombre d'unités. Par exemple, avec 100 000 voitures à produire, il est possible d'estimer le temps nécessaire à la fabrication d'une voiture et la multiplier, calculer les périodes pour les changements dans la chaîne de montage comme la peinture, les différents modèles, les options, et prévoir une marge d'erreur dans le cas d'un bris, ou d'une réparation. De plus, en tout temps, le gestionnaire peut s'informer du nombre de voitures produites pour ainsi mettre à jour sa planification et réévaluer ses échéances.

Cette précision est pratiquement impossible en informatique puisque les projets reposent sur des livrables souvent intangibles et les tâches nécessaires à la réalisation du produit ne sont pas répétitives et souvent faites à partir de nouvelles technologies, ce qui limite les références pour déterminer les échéances et la gestion des risques, puisqu'on ne connaît pas toujours le niveau de complexité des tâches lors de la planification, ni le temps qui sera nécessaire à sa réalisation. L'estimation des échéances, l'avancement des tâches et l'évaluation des risques sont habituellement déformés par les perceptions des différents acteurs y participant. Frank Padberg résume bien la planification de projet dans un article publié dans le magazine *Software Process Improvement and Pratice* de mars 2006 :

> La planification de projets logiciels a lieu sous une incertitude considérable. Pour différentes raisons, il est difficile d'estimer le temps nécessaire pour compléter certaines tâches de développement dans un projet logiciel. En outre, la rétroaction entre différentes activités cause souvent des erreurs et des retards.
>
> [Padberg 2006]- *Notre traduction.*

Auparavant, la planification de projet n'était pas incluse à l'intérieur d'une telle infrastructure puisqu'elle se réalisait uniquement par les gestionnaires, travaillant seul sur leur ordinateur. Par contre, les nouvelles méthodologies de gestion telles que celles présentées dans le chapitre 2, insistent sur l'importance d'impliquer les membres de l'équipe de développement dans l'élaboration des échéances, mais aussi dans le suivi de l'avancement des tâches. Leur participation dans la planification et dans le suivi du projet doit donc être supportée par l'infrastructure de développement et les processus en place.

4.5.1 Définition du contenu

La première étape de planification consiste à décrire le projet de façon suffisamment précise afin que tous les acteurs qui y seront impliqués puissent avoir une bonne idée des objectifs du projet, des résultats attendus et du travail à réaliser. La description du contenu permet de vendre un projet à un client, ou aux gestionnaires (en cas de projet produit à l'interne), mais aussi d'en expliquer les grandes lignes aux différentes équipes qui auront à y participer.

Dans le contexte de l'industrie du jeu vidéo, la définition du contenu, réalisée par le studio de développement, se retrouve dans le premier jet présenté à l'éditeur comme Ubisoft, Activision, Vivendi Universal, etc. et ce, même lorsque les projets proviennent de leurs propres studios de développement. Cette ébauche permet, entre autres, à l'éditeur de choisir les projets dans lesquels il investira.

Dans la conception d'un jeu vidéo, cette étape est primordiale puisqu'elle sert à préciser le jeu que l'on veut développer. De quel type de jeu s'agira-t-il ? Pour quelle(s) plate-forme(s) sera-t-il développé ? Quels seront les personnages et les environnements dans lesquels ils évolueront ? Quelles seront les fonctionnalités ou les particularités qui rendront le jeu attrayant ? Voici un éventail des questions devant être discutées avant même d'enclencher le reste du processus de planification. Parmi les éléments à aborder dans la définition du contenu d'un projet selon la troisième édition du corpus des connaissances en management de projet [PMI 2004], on retrouve :

- **Les objectifs du projet :** critères mesurables de succès du projet.
- **La description du contenu du produit** : caractéristiques et raisons pour lesquelles le projet a été entrepris.
- **Les exigences du projet** : conditions que le projet doit satisfaire pour atteindre les normes imposées.
- **Les limites du projet** : ce qui est inclus et exclu du projet.
- **Les livrables du projet** : artéfacts exigés qui constituent le projet et les résultats auxiliaires (rapports, documentation, etc.).
- **Les contraintes du projet** : contraintes spécifiques du projet telles qu'un budget prédéfini ou des dates imposées à certains livrables.
- **Les risques initiaux du projet** : risques connus ou anticipés.
- **L'estimation du coût** : estimation du coût global du projet avec une indication sur le degré d'exactitude du montant.

L'infrastructure mise en œuvre doit être en mesure de capter l'imagination des membres de l'organisation et de la rendre disponible afin d'alimenter les gestionnaires de projet dans la description du jeu. Des techniques seront proposées pour atteindre cet objectif dans le chapitre suivant, à l'intérieur du processus de gestion de la connaissance.

4.5.2 Méthodes d'estimation de la charge de travail

Bien qu'elle soit faite sous une incertitude considérable, tel que M. Padberg le souligne, il existe toutefois des méthodes pour améliorer l'estimation de l'effort nécessaire à la réalisation des projets logiciels. Le modèle de COCOMO, acronyme de l'anglais *COnstructive COst MOdel* fut élaboré à cet effet par Barry Boehm au début des années 1980. Ce modèle, basé sur des statistiques, s'est amélioré au cours des années pour tenter d'estimer les coûts et les délais d'un projet informatique. Au départ, cette méthode

consistait davantage à prédire l'effort et le temps de production des différentes phases d'un projet, en fonction de la complexité du projet, avec trois niveaux possibles, et du nombre de milliers de lignes de code. Le modèle s'est ensuite améliorer (COCOMO 2) pour prendre en considération la réutilisation des composantes existantes, les facteurs de productivité et les technologies utilisées. Puisque COCOMO 1 et 2 se base beaucoup sur le nombre de lignes de code et que cet indicateur n'est plus vraiment représentatif de la complexité ou de l'envergure d'un projet, ce modèle tend à disparaître dans l'industrie.

Dans les mêmes années que COCOMO, Allan Albrecht a conçu l'approche par points de fonctions. Ce modèle consiste à décomposer le système à réaliser en plusieurs fonctions et ainsi estimer chacune d'elles, en se basant sur les données, c'est-à-dire les entrées et les sorties, et les transactions : les requêtes externes, les fichiers logiques internes et les fichiers interfaces externes. Cette méthode est maintenant soutenue par l'IFPUG[14] (*The International Function Point Users' Group*), une organisation gouvernementale à but non lucratif dont la mission est de promouvoir et de supporter la gestion de développement logiciel, via l'approche par points de fonctions. Cette méthode est populaire, mais optimisée davantage pour des projets d'informatique de gestion que pour des projets plus techniques, comme le développement de jeux vidéo.

Plus récemment, suivant les principes des points de fonctions et grâce à la popularisation d'UML, l'approche des points de cas d'utilisation UCP, provenant de l'acronyme de l'anglais *Use Case Point*, s'est taillée une place considérable dans

---

[14] Site de l'IFPUG : http://www.ifpug.org

l'estimation de l'effort et du temps nécessaires au développement logiciel. Cette méthode consiste à prendre chacun des cas d'utilisation ou de scénarios, à déterminer les différents facteurs pouvant avoir un impact sur le développement tels que les facteurs techniques ou environnementaux, et à analyser l'importance de ces facteurs dans le projet pour ensuite procéder à l'estimation de l'effort nécessaire. Puisque les cas d'utilisation et le langage UML ne sont pas très utilisés dans l'industrie du développement de jeux vidéo, ce modèle est, lui aussi, rarement employé dans ce contexte.

Depuis, plusieurs modèles ont vu le jour tels que SEER-SEM[15], acronyme de l'anglais : System *Evaluation and Estimation of Resources - Software Estimating Model*, PROBE[16], acronyme de l'anglais : *PROxy Based Estimation* et plus récemment, le jeu de planification (*planning game*[17], provenant de la méthodologie de développement eX*treme Programming*). Contrairement aux autres méthodes décrites précédemment, le jeu de planification n'est pas construit autour de calculs fondés sur des statistiques, mais plutôt autour de discussions ou d'histoires, puisque cette technique est présentée sous forme de jeu) entre les gestionnaires et les développeurs. Les gestionnaires peuvent classer les scénarios, ou les livrables à réaliser, par priorités, par marchés ou par clients tandis que de leurs côtés, les développeurs classent ces scénarios par risque. Ensuite, les deux groupes estiment la durée de chacun des scénarios, qui varient entre quelques jours et trois semaines, et l'ordonnancement des scénarios pour planifier les différentes itérations.

---

15 SEER-SEM : http://www.galorath.com

16 PROBE : http://www.sei.cmu.edu/publications/documents/06.reports/06tn017.html

17 Planning game : http://www.xprogramming.com/xpmag/whatisxp.htm#planning

4.5.3 Structure de découpage du projet (Planification des tâches)

L'élaboration de la structure de découpage, ou la planification des tâches, est l'une des étapes les plus importantes de ce processus puisqu'elle spécifie clairement les mandats et les responsabilités des différents acteurs impliqués dans le projet. Cette activité consiste à définir, le plus précisément possible, l'ensemble des tâches nécessaires à la réalisation complète du projet, leur ordonnancement, leur durée et les personnes qui y travailleront. De plus, la structure de découpage définit l'utilisation des ressources humaines et matérielles qui devront être mobilisées pour l'accomplissement des tâches, ainsi que l'impact d'un temps excédentaire à celui planifié initialement pour la réalisation d'une activité, cet impact se mesurant autant sur le plan financier que des échéances. Ce dernier point permet de situer dans le temps la mobilisation des ressources critiques, comme un spécialiste d'une technologie particulière, un serveur de calcul ou une licence exclusive.

Il est recommandé dans la dernière édition du corpus des connaissances en management de projet [PMI 2004], dans le domaine du management des délais du projet (chapitre 6), de faire participer les parties prenantes au projet dans la planification et l'avancement des activités qu'elles ont à réaliser. « Le jugement d'experts, inspirés par l'information historique, peut être utilisé aussi souvent que possible. Chaque membre de l'équipe de projet devrait aussi fournir des informations sur l'estimation de la durée ou la durée maximale recommandée des activités ... ». Cette pratique permet non seulement d'avoir plusieurs avis et de diminuer les risques d'erreurs dans l'évaluation, mais aussi de responsabiliser l'équipe de projet dans l'atteinte des objectifs. Lorsqu'ils ont accès à la

planification et à l'avancement des activités, les personnes impliquées dans l'accomplissement des tâches peuvent rapidement informer le gestionnaire du projet du niveau d'avancement de chacune d'elles et l'avertir d'un éventuel retard possible. Les normes ISO vont d'ailleurs en ce sens avec le point 7.3.1 « Planification de la conception et du développement » qui invite les organisations à communiquer leur planification de projet aux personnes qui auront à réaliser le produit et donc l'intégrer dans leur infrastructure de développement.

> L'organisme doit gérer les interfaces entre les différents groupes impliqués dans la conception et le développement pour assurer une communication efficace et une attribution claire des responsabilités. Les éléments de sortie de la planification doivent être mis à jour autant que nécessaire au cours du déroulement de la conception et du développement.
>
> [ISO 9001:2000]

Plusieurs organisations fonctionnent avec plus d'un niveau de découpage (deux, dans la majorité des cas). Le premier niveau est simplifié et couvre l'ensemble du projet avec les étapes les plus importantes (*milestone*). Cette planification est destinée à être publiée dans l'ensemble des couches administratives de l'organisation (équipe de développement, marketing, gestionnaires, etc.), ainsi qu'à certaines entités externes (éditeur, institution financière, etc.). Le second niveau se colle à la planification simplifiée pour en détailler les tâches. L'usage est strictement interne et évolue tout au long du projet, contrairement au premier niveau qui ne change pratiquement pas (dans le meilleur des mondes).

4.5.4 Mise à jour de la planification

Le suivi continu des tâches planifiées par le chargé de projet et les membres de l'équipe permet au gestionnaire d'être plus efficace dans sa gestion des risques et de réagir plus rapidement. Lorsqu'un dépassement important des échéances se produit, ou semble s'annoncer, le gestionnaire doit refaire sa planification ou puiser dans une zone tampon qu'il avait prévue à cet effet. Il est à sa discrétion de publier ou non les zones tampons et les rendre disponibles aux membres de l'équipe.

Selon l'article « *Critical Chain Project Scheduling: Do Not Oversimplify* », paru dans le *Project Management Journal* : « les zones tampons (*buffers*) devraient offrir une protection contre les variations probables et devraient agir comme garantie en fournissant un mécanisme proactif de protection » [Herroelen et al. 2002] – *Notre traduction.* Par contre, ils affirment l'importance de ne pas exagérer l'utilisation de ces zones tampons : « l'expérience de Herroelen et Leus (2000) révèle que la règle du 50% pour l'évaluation des zones tampons peut entraîner une sérieuse surestimation des zones tampons requises et, par conséquent, de la durée du projet. » [Herroelen et al. 2002] – *Notre traduction.* Les auteurs du livre : *Applied Sofware Project Management* [Stellman-Greene 2005] renchérissent sur cette idée en soulignant que l'utilisation abusive des zones tampons procure un certain confort aux membres de l'équipe; s'ils le savent, ils peuvent s'y fier et prendre en considération ce temps dans leurs échéances. Puisque le temps est élastique, ils peuvent sous-entendre ces périodes tampons dans leurs échéances et les considérer comme acquises, ce qui rend alors la gestion des risques plus difficile.

4.5.5 Gestion des risques

Lors de la table ronde du *Game Developers Conferences* de 2004, la première session a commencé par une discussion pour déterminer comment le génie logiciel pouvait aider le développement de jeux vidéo. La réduction des risques fait partie des principaux points soulevés par cette question. Le contexte particulier de ce domaine, tel que décrit dans le chapitre 3, fait en sorte que les projets réalisés dans cette industrie ont un niveau de risque assez élevé. Selon N.R. Narayan Murthy, Directeur général d'Infosys, « Tout ce qui vaut la peine d'être entrepris contient des risques. Le défi d'un entrepreneur n'est pas de les éviter, mais de mettre en œuvre une stratégie pour limiter ces risques ». [Jalote 2002]

La gestion des risques consiste à évaluer les risques possibles en identifiant les sources de problèmes envisageables, les chances ou les probabilités qu'elles le soient et l'impact qu'elles pourraient avoir sur le projet. Une fois cette analyse effectuée, un plan d'action doit être mis en place pour surveiller ces risques, limiter leurs dommages s'ils s'avèrent et suivre leur évolution tout au long du projet. Neal S. Gray mentionne dans le PM Network [Gray 2001] que planifier sans faire d'évaluation des risques est peu utile en affirmant que « par l'estimation des risques associés à une partie du projet, le gestionnaire du projet peut montrer aux décideurs les résultats et les conséquences éventuels de leurs actions ». Voici les éléments de sortie proposés par l'une des activités du CMM (treizième activité du processus : « répétable ») concernant la gestion des risques : [SEI 1993] – *Notre traduction*

> Les risques associés au coût, aux ressources, aux échéances et aux aspects techniques du projet logiciel sont identifiés, estimés et documentés.

1. Les risques sont analysés et placés en priorité, sur la base de leur impact potentiel.
2. Les contingences de ces risques sont identifiées (zones tampons, alternative pour les ressources humaines et matérielles).

CHAPITRE 5

STRATÉGIE DE MISE EN OEUVRE

Une fois les processus assimilés, il est impératif de mettre en place une infrastructure pour les supporter et les encadrer. Ce chapitre présente une solution de mise en œuvre capable de répondre aux difficultés exposées dans les chapitres précédents et ainsi, mieux supporter le développement de logiciels dans un contexte semblable à celui de l'industrie du jeu vidéo. Les choix réalisés lors de l'élaboration de cette infrastructure sont capitaux puisqu'ils font la différence entre l'acceptation du changement par l'équipe de projet et le refus total de sa part. Les outils et les procédures doivent être clairs et simples pour minimiser l'effet de résistance au changement. Des formations doivent aussi être données, afin que tous les membres de l'équipe utilisent les applications de la même façon et qu'ils comprennent l'importance de les utiliser, pour eux et pour l'organisation.

Tout d'abord, la pierre angulaire de l'architecture repose sur l'utilisation de technologies indépendantes des différentes plates-formes telles qu'Unix, Windows et Mac, basé sur des protocoles standard. Ainsi, en se libérant des outils propriétaires et des langages fermés, il est possible d'éviter un bon nombre de problèmes récurents en informatique. Par exemple, pour la saisie des défauts trouvés dans le projet en cours, il est possible d'utiliser Microsoft InfoPath pour créer un formulaire, qui par la suite, envoie les résultats dans une base de données. Cette infrastructure nécessite par contre que toutes les personnes voulant publier un défaut aient une version récente de Microsoft Office. Le système de gestion des défauts deviendra alors dépendant de ce facteur. Le changement de version de la suite Office, la migration vers un autre progiciel tel que OpenOffice.org, ou l'utilisation d'une autre plate-forme telle que Linux ou Macintosh, pourraient rendre le

système instable, ou tout simplement inutilisable, à moins d'un investissement considérable d'efforts et de ressources.

Autant que possible, nous favorisons donc l'utilisation du protocole *http* avec toute la gamme de protocoles reconnus dans le développement Internet (comme les SSH, SMTP, IMAP, etc.) et des applications pouvant s'installer sur un serveur Web standard. Ces précautions apportent une indépendance technique permettant de moderniser toute une infrastructure de façon transparente pour les utilisateurs. L'installation est plus rapide uniquement sur le serveur, et non sur tous les postes, et la maintenance est simplifiée puisque l'architecture est indépendante des systèmes d'exploitation, chacun disposant d'un navigateur Web. De plus, l'utilisation d'une infrastructure entièrement Web offre la possibilité d'intégrer assez aisément une interface centrale, servant de portail au système tout entier. Le portail donne ainsi l'impression d'un tout et non la juxtaposition de plusieurs applications différentes. Ce dernier point augmente considérablement l'ergonomie du système, mais nécessite un effort de programmation supplémentaire. L'objectif de ces spécifications est d'inciter les parties prenantes à l'implantation de la stratégie de mise en œuvre à s'investir pour s'approprier les changements et facilitant ainsi la transition.

5.1 Gestion des documents et des modifications

Plusieurs outils informatiques existent sur le marché présentement pour répondre aux besoins de ce processus. Peu importe la solution retenue, il est préférable de miser sur une architecture de type client-serveur où le serveur entreposera l'ensemble des documents et des modifications de tous les postes clients. Ceci facilitant grandement la sauvegarde du

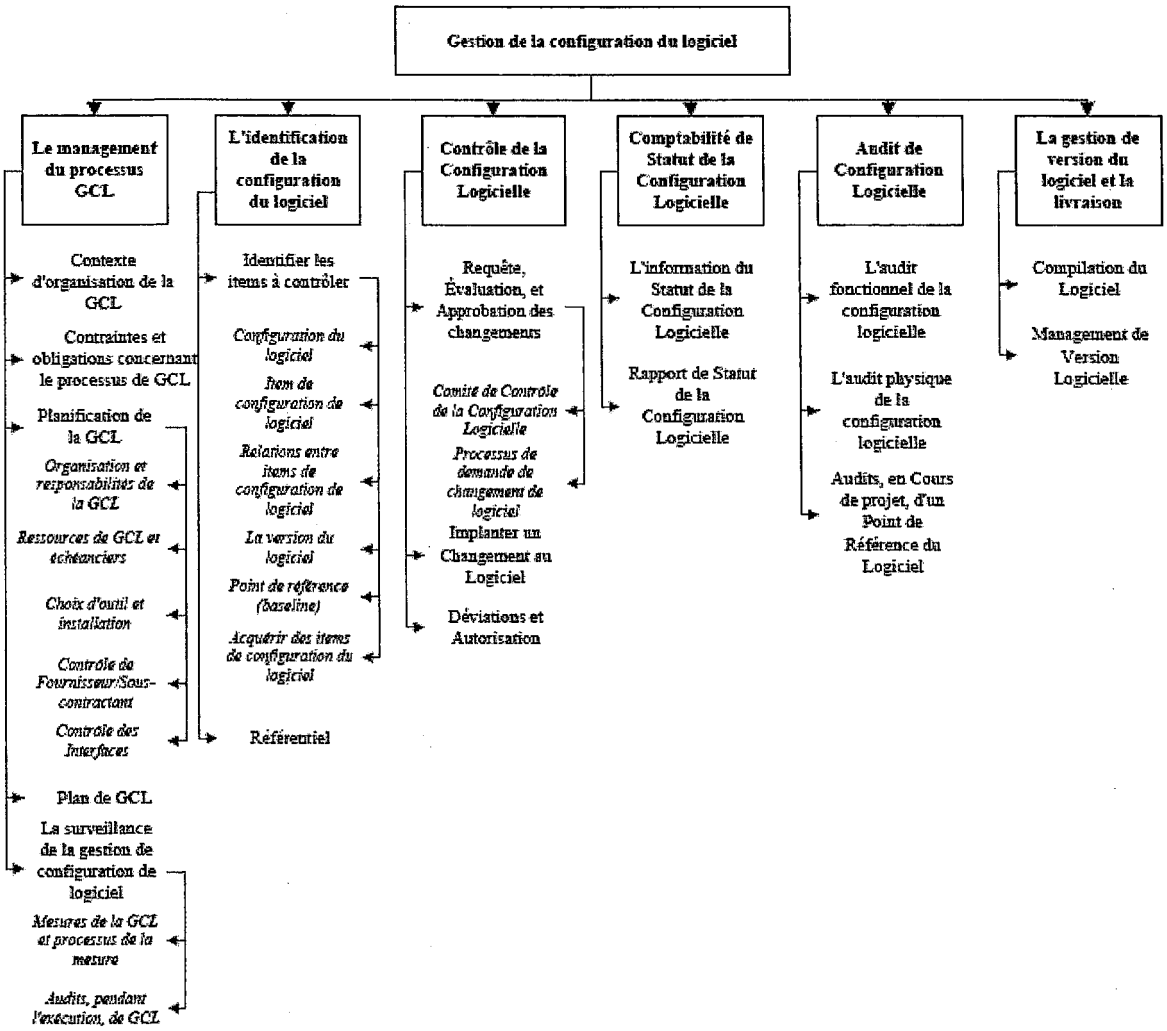
travail de tous les membres de l'équipe, puisqu'il n'y a qu'un seul lieu de stockage et une seule sauvegarde à prendre. De plus, cela assure l'intégrité des fichiers puisque l'information est stockée en un seul endroit, à moins que deux personnes travaillent sur un fichier au même moment. Mais encore là, des mécanismes, que nous identifions dans le guide d'utilisation mis en annexe (A-1), peuvent être mis en œuvre pour établir la fusion entre deux fichiers de code source différents. Seules des raisons de confidentialité pourraient favoriser la solution client. Encore là, une bonne solution devrait être en mesure de séparer les dossiers accessibles de ceux qui ne le sont pas, pour chaque utilisateur ou groupe d'utilisateurs, en fonction des besoins de sécurité et de confidentialité exigés par l'organisation.

Le guide du SWEBOK énumère une liste exhaustive des activités nécessaires à la réalisation d'un processus mature de gestion des documents et des modifications (gestion de la configuration du logiciel) qui est présentée dans la figure 5.1. Afin de conserver l'aspect léger de notre méthodologie, nous nous contenterons d'élaborer les activités essentielles au processus, les points nécessitant des éclaircissements ou qui peuvent générer des difficultés, les solutions de mise en œuvre ainsi que l'une des problématiques associées au processus qu'aucun outil ne permet de résoudre pour le moment.

À la suite de la mise en œuvre de la méthodologie et de l'infrastructure suggérées, une équipe de développement de logiciels, désirant améliorer son processus de gestion des documents et des nouveautés, pourrait utiliser les notions présentées dans le guide du SWEBOK. Cependant, nous considérons difficile la mise en place d'un tel processus dans

sa globalité, dans un contexte d'une nouvelle équipe ou dans un contexte semblable à celui du jeu vidéo.

Figure 5.1 : Gestion de la configuration (des documents et des modifications) selon le guide du SWEBOK



[IEEE 2004]

5.1.1 Identification des documents à contrôler

Puisqu'il est impossible de penser contrôler tous les fichiers de tous les membres de toutes les équipes sans engorger le réseau et les serveurs de l'organisation, un tri est donc nécessaire pour identifier les documents qui seront contrôlés. Une hiérarchie des différents dossiers, sous-dossiers et fichiers doit d'abord être créée sur un SCV (Systèmes de Contrôle de Version) par le gestionnaire du projet ou par quelqu'un mandaté à cet effet, afin que les membres de l'équipe puissent synchroniser leur ordinateur au SCV. Cette étape leur permet d'accélérer la prise de connaissance de la structure des dossiers et des fichiers et donne une idée assez précise de l'endroit où devront être insérés les fichiers futurs.

Il est aussi recommandé d'inclure un espace particulier pour chacun des membres des équipes dans le SCV en dehors de la structure du projet, leur offrant ainsi la possibilité de contrôler leurs propres documents comme les courriels personnels et les fichiers inachevés. Cette précaution permet d'épurer les fichiers spécifiques au projet, de limiter les alertes de modification et d'accélérer le système (dans le cas où le SCV du projet est sur un serveur différent du SCV des employés).

5.1.2 Fréquence des publications

Le contrôle de version ne signifie pas qu'il est nécessaire de publier les fichiers, c'est-à-dire de les envoyer sur le répertoire de travail en cours du serveur à toutes les fois qu'il y a un changement. Il est préférable d'attendre quelques heures, ou même quelques jours afin que les modifications aient été complétées, ou partiellement complétées, sur un fichier pour les publier. Ainsi, l'historique des changements d'un fichier ne sera pas pollué

de multiples versions inutiles, mais surtout on limite la publication de fichiers non testés, remplis d'erreurs. Dans le cas contraire, ces fichiers inaboutis peuvent se retrouver sur d'autres postes et entraîner des erreurs collatérales faisant perdre du temps. Il est important de faire comprendre ces détails aux membres de l'équipe, afin de les conscientiser à cette problématique et les rendre suffisamment à l'aise avec le processus pour qu'ils puissent connaître le bon moment et les raisons pour publier leurs fichiers.

5.1.3 Contrôle des publications

Le gestionnaire de projet doit aussi contrôler les publications des membres de son équipe pour deux raisons particulières. Premièrement, elles lui permettent de connaître l'avancement du projet. En suivant les descriptions des publications, le gestionnaire peut avoir une idée assez précise des tâches qui sont terminées, celles qui ne le sont pas et qui les ont réalisées. De plus, le nombre de publications (trop ou trop peu) d'un membre peut signifier que ce dernier n'utilise pas convenablement le SCV. Cette raison est primordiale lors de l'implantation du processus au sein des équipes, afin de s'assurer de la compréhension des procédures d'utilisation.

5.1.4 Version stable

Le développement de jeux nécessite plusieurs équipes de disciplines différentes qui ne peuvent utiliser les mêmes versions au même moment. Les artistes ont besoin d'une version suffisamment stable du code pour intégrer et tester leurs modèles 3D, leurs textures, leurs sons ambiants, etc. Les programmeurs doivent habituellement avoir les dernières versions en développement pour s'assurer qu'aucun changement ne vient altérer le reste du

code. Les gens en communication, de leur côté, ont besoin d'une version légère intégrant assez de fonctionnalités et de qualités artistiques pour débuter les phases de mises en marché (démo jouable, vidéo, affiche, etc.).

Pour remédier à ce problème, la plupart des SCV que nous verrons plus loin dans ce chapitre, offrent la possibilité de créer des branches et ainsi de séparer un projet en différentes phases. Par exemple, lorsqu'une version en développement atteint une certaine maturité et est considérée stable, il est alors possible d'écraser l'ancienne version du code utilisée par les artistes, par cette dernière tout en conservant l'ancienne version pour revenir en arrière rapidement, au cas où la nouvelle causerait trop de problèmes. Ces branches sont une copie indépendante d'un fichier ou d'un répertoire dans le système de contrôle de version. Les programmeurs peuvent alors continuer de modifier des fichiers sans courir le risque de nuire au travail des artistes qui ont maintenant la possibilité de tester leur matériel sur une version stable et plus à jour.

Figure 5.2 : Publication d'une nouvelle branche ou mise à jour d'une version stable

L'icône du fichier et ceux des dossiers racines, alertent qu'une version plus récente que celle sur leur ordinateur est disponible sur le SCV

Un courriel est envoyé aux personnes en question pour les alerter d'un changement sur le SCV

La figure 5.2 démontre la publication d'une nouvelle branche ou la mise à jour d'une version stable. Chacun des membres de l'équipe de développement qui possède une version locale du dossier contenant la nouvelle branche sera avisé lors de la publication. Le système de contrôle de version informe aussi l'équipe de test et les gestionnaires de l'ajout d'une nouvelle branche. Puisque ces opérations sont commentées (en fonction des procédures de l'organisation), les gestionnaires peuvent rapidement être mis au courant de l'avancement du projet et de la publication dans le SCV des livrables. L'équipe de test, de son côté, peut planifier et effectuer les tests sur la nouvelle version stable et détecter rapidement ses défauts. Nous verrons ultérieurement dans ce chapitre les mécanismes pour déclencher la vérification de cette nouvelle branche.

La figure 5.3 illustre un exemple de publication d'une nouvelle version d'un fichier dans le système de contrôle de version. Dans ce cas-ci, les programmeurs travaillent sur une version du code en développement, tandis que les artistes eux, élaborent des modèles 3D, des niveaux, des textures, et d'autres à l'aide d'une version stable. Lors de la modification dans le dossier de code, les programmeurs auront alors une alerte et les artistes ne verront aucun changement puisqu'ils n'ont pas accès à ce dossier (ils n'ont besoin que des versions considérées stables). On peut distinguer dans la figure que le ou les gestionnaires du projet peuvent aussi être informés de la mise à jour des versions. Ils peuvent avoir une version locale synchronisée avec le système de contrôle de version ou bien être alerté par des messages ou des courriels, dépendant des fonctions disponibles du SCV.

Figure 5.3 : Publication d'un fichier dans une version en développement

L'icône du fichier et ceux des dossiers racines, alertent qu'une version plus récente que celle sur leur ordinateur est disponible sur le SCV

Un courriel est envoyé aux personnes en question pour les alerter d'un changement sur le SCV

5.1.5 Systèmes de contrôle de versions

Les SCV (Systèmes de Contrôle de Version) sont arrivés dans le monde du développement logiciel en 1972 avec le SCCS (*Source Code Control System*), conçu par les gens de *Bell Telephone Laboratories*. Dans les années 1980, le *RCS (Revision Control System*), est apparu pour ensuite inspirer neuf ans plus tard le *CVS* (*Concurrent Version System*), qui est sans aucun doute le plus répandu des SCV en ce moment.

Dans les mêmes années, *BitKeeper* faisait son apparition, utilisé pour supporter plusieurs gros projets libres, tel que *Linux* et plus tard, *MySql*. Par la suite, *Microsoft* a sorti *VSS* (*Visual Source Safe*). Plusieurs alternatives libres à *CVS* sont apparues telles que *Aegis*, *Arch*, *Darcs*, *OpenCM* et bien d'autres; ces derniers sont tous disponibles à partir de

SourceForge[18]. Pour terminer, *Subversion*, une amélioration de *CVS*, qui commence à se tailler une place de premier plan. Pour ce qui est des moyennes et grandes entreprises, des outils plus performants, plus complexes et plutôt dispendieux, tels que *Perforce* (variant entre 500$ US et 800$ US par utilisateur !) et *Rationnal ClearCase* d'*IBM* (5 400 $) sont disponibles sur le marché. Par contre, des besoins de performance très spécifiques devraient être à l'origine du choix de ces outils.

Lors du *Game Developers Conference* de 2004, Noel Llopis [Llopis 2004] a réalisé un sondage auprès des différents participants (oeuvrant dans l'industrie du développement de jeux vidéo) sur les outils qu'ils utilisent pour effectuer le contrôle des documents et du code source (car ce ne sont pas toujours les mêmes dans les deux cas). Voici les résultats et les commentaires sur chacun des outils recueillis par monsieur Llopis :

Gestion des documents

Visual SourceSafe: *40%.*	Histoires d'horreurs de bases de données constamment corrompues, mais cela semble plus ou moins fonctionner.
Alienbrain: *40%.*	Les gens ne semblent pas enthousiastes envers ce système. Trop lourd pour l'utilisation. Seulement bon si vous employez les fonctionnalités plus avancées.
CVS: *8%.*	Ils n'utilisent que peu de données.
Perforce: *5%.*	Les artistes trouvent agréables les GUI par défaut et les utilisent sans problème.
Aucun SCV: 5%.	Ils considèrent la possibilité d'utiliser un système de gestion de documents.

Gestion du code source

Perforce: *40%.*	La plupart des gens utilisent les branches.
Visual SourceSafe: *36%.*	Pratiquement personne n'utilise les branches (pas surprenant).

[18] Sources pour télécharges les différents SCV libres: http://sourceforge.net

CVS: *20%.*	Le GUI de Tortoise est apparemment très bon, bien que plusieurs ne l'utilisent qu'en mode « ligne de commandes ».
Accurev: 4%.	Une intéressante validation en deux étapes est la principale différence de cet outil.

Llopis souligne que personne n'utilisait Subversion lors du sondage, mais que certains s'y intéressaient. Il est à noter qu'en 2004, la version 1.0 de Subversion venait tout juste d'être rendue disponible.

5.1.6 Comparaison entre les SCV

Afin de faire un choix éclairé quant au SCV à utiliser, une comparaison (tableau 5.1) d'outils génériques distribués gratuitement (*CVS*, *BitKeeper*, *Arch* et *Subversion*) sera faite en fonction des critères suivants : le niveau de documentation et de support, les plates-formes supportées, la gestion des permissions, c'est-à-dire la sécurité, le support réseau, la possibilité de travailler sur un fichier ou un répertoire, de le copier, le renommer, le déplacer, de publier de façon atomique, alors que chaque fichier a sa version, et pour terminer, la facilité d'utilisation générale, car il n'y a pas que le code source à gérer, mais aussi les documents des gestionnaires et des membres des disciplines autres que l'informatique.

Les critères d'évaluation sont choisis en fonction de ceux que l'on retrouve dans la littérature, mis à part le dernier, *Facilité d'utilisation*, que nous avons déterminé pour répondre à la problématique du contexte particulier et plus précisément, de la multidisciplinarité des équipes de développement de jeux vidéo. Ce critère est répondu en fonction de l'expérimentation de l'auteur vis-à-vis l'utilisation de chacun des outils. La

couleur de fond pâle de certaines cellules, indique une force d'un SCV en comparaison des autres et la couleur foncée représente une faiblesse.

Tableau 5.1 : Comparaison entre les SCV

	CVS	BitKeeper	Arch	Subversion
Documentation	Excellente, beaucoup de documentation, de support disponible sur le Web et de livres.	Très bon, site Web bien fait avec beaucoup d'information, aucun livre. (*http://www.bitkeeper.com*)	Pauvre, plutôt des wiki, pas forcément à jour avec les dernières versions.	Excellente, beaucoup sur le Web et quelques livres déjà disponibles.
Plates-formes supportées	Unix, Win, MacOs X	Supporte plus d'une vingtaine de plates-formes différentes	Unix (~cygwin)	Unix, Win, MacOs X
Gestion des Permissions	Limité, gestion par groupe, très peu efficace et devient rapidement encombrant.	Oui, mais comporte quelques failles de sécurités	Oui, se fait lors de la connexion avec les différents protocoles.	Le contrôle des permissions se fait lors de la connexion par ssh, http (avec Apache).
Support réseau (client-serveur)	Moyen, utilise son propre protocole et peut passer par un tunnel ssh	Bon, possède son propre protocole et peut utiliser http.	Excellent, peut utiliser plusieurs protocoles (FTP, SFTP, WebDAV) ainsi que les protocoles de système de fichier à distance (NFS, SMB).	Très bon, utilise WebDav ou un protocole svn par tunnel ssh.
Travail sur un répertoire	Oui	Non, les changements se font sur tous les répertoires	Possible mais le *check-out* se fait sur tous les répertoires.	Oui
Renommer, copier, déplacer un fichier/répertoire	Non	Oui	Oui (sauf copie)	Oui, Complexité O (1) [19]
Publication atomique	Non	Oui	Oui	Oui
Facilité d'utilisation	Très simple, devenu un standard. Peut être difficile à configurer.	Commandes semblables au CVS, mais avec quelques nuances.	Plusieurs commandes, peu d'outils disponibles, formation nécessaire.	Pratiquement identique au CVS.

Tel qu'on peut le déduire en observant le tableau 5.1, les différents SCV ont chacun leurs avantages, qui devraient être pris en considération en fonction des besoins du projet, de ses équipes et de son envergure. Par contre, on peut rapidement constater que chacun des

[19] Dans la théorie de la complexité informatique, O (1) détermine un temps constant (dans notre contexte, cela indique que peu importe le nombre de fichiers dans le répertoire, le temps nécessaire à la réalisation de ces actions sera approximativement toujours le même).

systèmes présentés a une longueur d'avance sur le CVS qui, malgré sa popularité, comporte certains désagréments tels que son architecture non flexible où il est impossible de copier, déplacer et renommer un fichier, la publication du répertoire entier et non de façon atomique, puis sa difficulté à gérer les permissions telles que les répertoires, droits d'accès et utilisateurs.

Dans un contexte d'équipe multidisciplinaire, on ne peut se permettre de devoir former les gens à apprendre un grand nombre de commandes pour utiliser notre SCV. Il doit donc être le plus transparent possible au système d'exploitation et demander un minimum d'effort de la part des membres de l'équipe pour faciliter l'immersion et réduire la résistance à son utilisation.

Pour ces raisons, Subversion sera le SCV retenu pour le guide d'utilisation de l'environnement que nous proposons. Il existe plusieurs clients graphiques pour l'utilisation de Subversion. Plusieurs applications en mode client pour différentes plates-formes sont déjà disponibles sur des sites tels que *www.sourceforge.net, http://subversion.tigris.org, http://tmate.org/svn, http://www.smartcvs.com/* et bien d'autres. Dans le guide d'utilisation, TortoiseSVN (Windows) sera le client graphique, puisqu'il est très visuel et en français pour nos exemples, mais sinon, Rapidsvn serait un bon choix puisqu'il est déjà offert pour les principales plates-formes Windows, Linux et MacOS.

5.1.7 Le problème de la numérotation des versions

Dès que l'on sort un document du SCV, soit en format imprimé ou lorsqu'on l'envoie de façon électronique, une question persiste. Le document en question est-il dépassé, une version plus récente est-elle disponible ? C'est aussi le cas pour les manuels de procédures. Plusieurs personnes préfèrent avoir ce genre de document en version papier plutôt qu'électronique. Chez Sogique, par exemple, dès qu'une procédure est imprimée, une inscription apparaît sur les feuilles pour indiquer qu'elle est déjà dépassée. Ce mécanisme rappelle aux utilisateurs de ne pas se fier uniquement à la version imprimée, mais ne peut avertir ce dernier de la version dont il dispose, à moins qu'un numéro soit mis à jour manuellement.

Pour le moment, les principaux SCV ne permettent pas d'ajouter à l'intérieur des fichiers, le numéro de la version d'un document. Il serait fort pratique de pouvoir configurer notre SCV, afin que ce dernier puisse mettre en pied de page d'un fichier texte, par exemple, le numéro de la révision au moment de l'impression. La personne qui utilise ce document en version papier n'aurait qu'à regarder le numéro actuel de la version dans le SCV pour savoir si son document est à jour ou non. Bien sûr, ce genre de fonctionnalité ne peut être implémenté que sur les SCV qui font les mises à jour de façon atomique. Puisque CVS met à jour l'ensemble de l'arborescence du projet à chacune des publications, le numéro n'est plus en fonction d'un fichier, mais du projet. L'ajout d'une telle fonctionnalité est donc impossible avec CVS, mais demeure néanmoins envisageable avec Subversion.

La gestion des utilisateurs avec leurs niveaux de sécurité est un autre petit problème des SCV présentés ici. Bien que ces derniers, Subversion par exemple, permettent de créer des groupes et des utilisateurs avec leur mot de passe, il n'en demeure pas moins que cet exercice est plutôt laborieux. Tout est géré à l'intérieur d'un fichier texte pour le projet, où l'on doit écrire les autorisations et les restrictions pour chaque fichier ou dossier et sous-dossier. Un exemple est fourni dans le guide d'utilisation en annexe. Cela peut devenir lourd à gérer dans une organisation ayant plusieurs dizaines d'employés. Par contre, une petite application locale peut rapidement être développée pour accélérer le processus d'édition de ce fichier.

5.2 Gestion des défauts.

Tout au long du développement et même une fois le produit ou service livré, l'organisation doit se doter d'une infrastructure capable de répertorier efficacement les défauts et les incidents notés par les acteurs du processus de conception, du développeur au client. Cette action devra ensuite avoir pour effet d'alarmer des gens responsables qui devront prendre conscience de l'existence de ce défaut/incident, par exemple un défaut pour un produit, un incident pour un service, en déterminer l'impact et mettre en œuvre les mesures nécessaires pour le corriger ou s'assurer de limiter les conséquences sur le développement futur. L'infrastructure devra aussi être suffisamment souple pour contenir les tests planifiés lors de la conception et leur état d'avancement.

Lorsqu'un service est interrompu à cause d'un incident ou d'un problème, par exemple un site Web, il est donc essentiel d'avoir un système très efficace pour répondre le

plus rapidement possible aux utilisateurs qui en sont privés. Leur demande de support doit être prise en charge sans délai. Ils doivent avoir l'impression qu'une intervention est entreprise dès l'ouverture du billet, et donc avoir un retour d'information très rapide. Dans le cas d'un incident ou problème connu, il est recommandé que le membre de l'équipe en charge de la demande puisse avoir accès à une procédure pour répondre à l'utilisateur et effectuer les premiers tests associés à ce type d'erreur. Les membres du projet doivent connaître l'existence d'un défaut, mais aussi être au courant de son degré de résolution. Le gestionnaire de projet est ainsi plus en mesure d'en calculer l'impact et de voir s'il doit injecter davantage de ressources pour l'investigation ou la correction. En ayant une image assez nette de l'ensemble des défauts du projet, il peut mettre à jour sa gestion des risques et se servir de ce système pour orienter ses décisions.

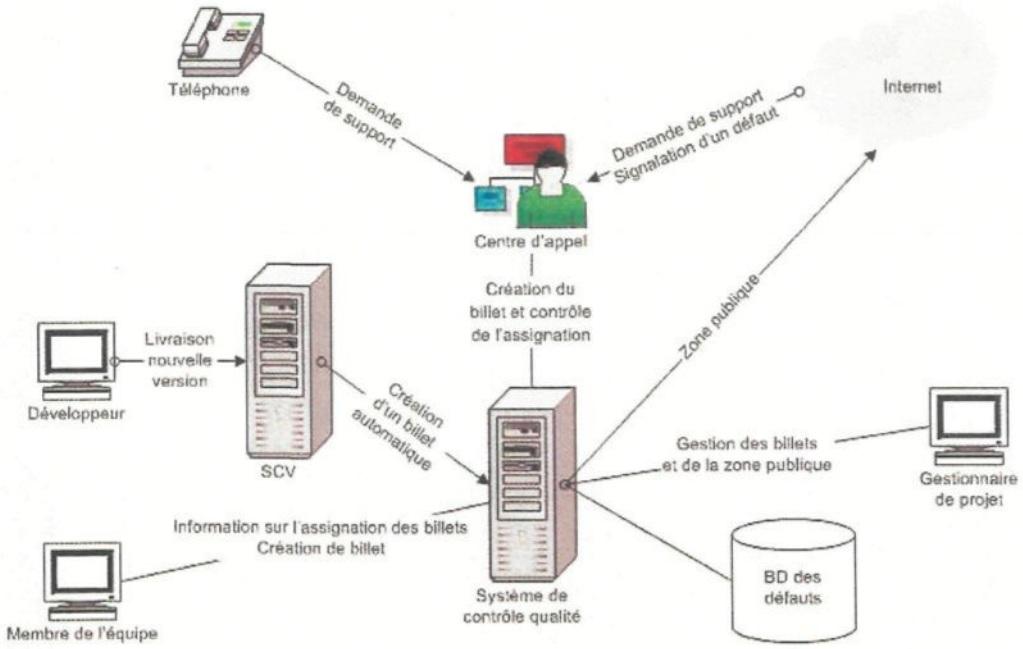
Le système qualité peut aussi être utilisé pour répondre aux défauts ou aux incidents les plus fréquemment rapportés. Un client qui est bloqué pour une raison quelconque peut trouver rapidement une solution à son problème en naviguant dans la zone publique du système. Certains aspects tels que la précision des solutions, la bonne utilisation de la langue et la pertinence des problèmes de cette zone doivent toutefois être contrôlés, afin de conserver un nombre assez restreint de problèmes et ne pas nuire à l'image de l'organisation.

5.2.1 Infrastructure de gestion de défauts proposée

La figure 5.4 est une représentation de l'infrastructure suggérée dans ce travail. On peut remarquer trois points d'entrée différents dans le système de contrôle de qualité, soit le

centre d'appel, le système de contrôle de version (SCV) ou encore un membre de l'équipe, (contrôle qualité, développeur ou autres). La personne qui prend l'appel ou la demande via le Web a pour mandat de vérifier l'existence d'un billet et de le mettre à jour s'il existe. Dans le cas contraire, elle doit créer un nouveau billet avec le plus d'information possible et ensuite l'assigner à la personne ou au groupe qui est en charge de la correction de ce type de défaut au meilleur de ses connaissances.

Figure 5.4 : Infrastructure de système de gestion de défauts



5.2.2 Détection interne

Une fois un défaut détecté, il faut vérifier s'il est déjà connu dans le système par l'entremise de la base de données des défauts. Si oui, une note doit apparaître pour en signaler la répétition par la personne X à la date Y. Si non, un billet de défaut doit être créé et adressé à la personne, ou au groupe, concernée avec les informations suivantes :

- Catégorie de défaut (réseau, interface, modèle 3D, etc.).
- Département.
- Projet (jeu).
- Brève description.
- Endroit spécifique où le défaut est apparu et l'événement qui l'a déclenché.
- Description complète avec fichier joint si possible (ex : copie d'écran).
- Informations relatives à la découverte (qui et quand).

5.2.3 Détection externe

Les informations d'un billet qui sont entrées par les utilisateurs externes apparaissent sensiblement les mêmes que lors de la détection interne à l'exception du destinataire qui doit être général, puisque les personnes externes ne peuvent savoir qui prendra en charge un billet en fonction de sa nature. Le centre d'appel fait alors la recherche pour savoir si le défaut signalé est déjà connu dans le système. S'il ne l'est pas, un nouveau billet est alors créé et assigné par le centre d'appel. Dans le cas d'un défaut déjà répertorié, si le billet est résolu, une décision doit être prise à savoir s'il doit être ouvert à nouveau, en raison d'un ancien défaut qui refait surface par exemple, et les personnes qui avaient travaillé pour le résoudre doivent en être informées. Si le billet n'est pas résolu, le système doit permettre de noter rapidement la répétition, permettant ainsi de calculer l'impact d'un défaut. Lorsqu'une centaine de clients prennent le temps de signaler un défaut dans un court laps de temps, on peut décider de classer le défaut en urgence et tomber en gestion de problème.

5.2.4 Assignation des billets

Une fois le rapport de défaut complété, de façon interne ou externe, il reste à assigner le billet à la personne qui sera en charge de sa correction ou de son suivi. Pour ce faire, une procédure d'assignation doit être mise en place. Cette procédure établit qui doit faire l'assignation et quels sont les critères de qualité exigés par l'organisation, quels champs doivent être remplis et comment ils doivent l'être en fonction de la nature des différents types de défauts. L'établissement de ces critères facilite l'uniformisation du processus et la réalisation de rapport. Il faut donc désigner une personne ou un groupe comme répartiteur des billets.

Voici les rôles qu'un répartiteur doit tenir :

- ✓ Vérifier la classification des billets (bonne catégorie, bon projet).
- ✓ Estimer sommairement l'impact d'un défaut.
- ✓ Assurer la non-redondance des billets (fermer un billet redondant, le faire pointer vers l'original et insérer une note dans ce dernier pour démontrer la fréquence et l'impact).
- ✓ Assigner le billet à la personne la plus en mesure de répondre rapidement au meilleur de ses connaissances en fonction :
    - de la catégorie de défaut;
    - des disponibilités des membres de l'équipe pour éviter d'assigner un billet urgent à un technicien absent, par exemple;
    - de la répartition des charges (éviter d'envoyer tous les billets de défauts à la même personne, à moins d'avis contraire).
- ✓ S'assurer que les personnes assignées ont pris rapidement connaissance du billet.
- ✓ Réassigner les billets mal assignés.

5.2.5 Prise en charge du défaut

De leur côté, les personnes assignées aux billets doivent rapidement prendre connaissance du défaut ou de l'incident et ouvrir le billet. Cette opération permet donc à la personne assignée de valider qu'elle est la bonne ressource, que le billet est bien classé et dans la bonne catégorie, de mesurer la sévérité et la priorité du défaut ou de l'incident. Lorsqu'un membre prend conscience d'un billet et qu'il constate qu'il est mal classé, il doit retourner le billet au répartiteur en indiquant sa recommandation pour le classement. Ceci dans le but que le répartiteur comprenne et apprenne de ses erreurs. La personne du centre d'appel qui assigne le billet, doit aussi s'assurer qu'il est pris en charge selon les standards de qualité de l'organisation, entre autres le temps maximum pour prendre conscience du billet, pour établir un premier contact, pour résoudre le billet et sinon, escalader le billet. La rigueur de cette surveillance permet de conserver l'attention des membres de l'équipe envers le système de gestion des défauts. Un chef d'équipe se voyant attribuer plusieurs billets, puisque ses subordonnés n'en prennent pas conscience ou qu'ils n'y répondent pas, pourra alors entreprendre les démarches nécessaires pour corriger ces lacunes.

L'escalade, selon ITIL, signifie le transfert du billet à une autre personne. Elle peut se faire de façon horizontale (fonctionnelle), ce qui implique le transfert du billet à un autre membre de l'équipe du même niveau hiérarchique. Ceci arrive lorsque la personne à qui le billet était assigné n'est pas en mesure de répondre par manque de temps, de compétence ou de connaissance. Sinon, l'escalade se fait verticalement, c'est-à-dire hiérarchiquement. Le billet est alors transféré à un membre d'un niveau hiérarchique supérieur. Ce dernier sera

alors plus en mesure d'appliquer de la pression auprès des intervenants ou d'accéder aux ressources nécessaires pour la correction du défaut.

Toutes les interventions effectuées pour un défaut, de l'ouverture du billet, jusqu'à sa fermeture, doivent être archivées dans le système selon les normes ISO. « Les enregistrements de la nature des non-conformités et de toutes les actions ultérieures entreprises, y compris les dérogations obtenues, doivent être conservés ». [ISO 9001 :2000]

Le gestionnaire de projet, ou toute autre personne qu'il a mandatée à cet effet, doit prendre le temps de vérifier les billets ouverts, afin de s'assurer que les clients obtiennent une réponse dans des délais acceptables et cela lui permet d'être conscient des défauts majeurs de son produit ou service. Un contrôle des billets mis dans la zone publique et accessible aux clients, tel que présenté précédemment, est nécessaire afin de s'assurer du niveau de langage utilisé et de la qualité des solutions présentées.

5.2.6 Comparaison des outils

Il existe un très grand nombre d'outils disponible sur Internet pour la correction et la gestion des défauts. Deux types de logiciels génériques peuvent servir à remplir les fonctions nécessaires au processus de contrôle qualité. On les retrouve principalement sous les noms de traqueur de défauts (*bug trackers*) et de centre d'assistance (*helpdesk*). Seulement sur le site DMOZ open directory project (*http://dmoz.org*), on peut trouver une cinquantaine de logiciels traqueurs de défauts, dont 27 sont gratuits, de même qu'une centaine de systèmes de centre d'assistance différents.

Les outils de type traqueur de défauts sont utilisés principalement pour les produits et les centres d'assistance pour les services. Par contre, puisque leurs fonctions sont a priori les mêmes, un bon système devrait être suffisamment adaptable pour convenir autant à un type d'utilisation qu'à un autre. L'analyse des outils génériques qui sera faite prend en considération cet élément important. Le prix de ces outils varie énormément passant de zéro à plus de 100 000 $, particulièrement les outils de type centre d'assistance qui sont un peu plus spécialisés et dont certains intègrent les normes ITIL.

Parmi les logiciels et les types qui seront comparés dans le tableau 5.2, on retrouve *Mantis*, le meilleur traqueur de défauts selon Micheal Flanakin [Flanakin 2005], *dotproject*, un « outil de gestion de projet » présentement en développement sur http://sourceforge.net, *One or Zero*, l'un des centres d'assistance libre les plus appropriés via le site www.helpdesks.com et *Web Help Desk*, l'un des nombreux centres d'assistance sur le marché, suggéré par le site www.capterra.com et utilisé par plusieurs grandes organisations[20]. Les critères utilisés proviennent de différents sites Internet[21], mis à part « adaptable » et « facilité » qui sont intégrés pour répondre à la problématique apportée par le contexte et où on retrouve des utilisateurs ayant des cultures informatiques différentes tels que les artistes, les informaticiens et les gestionnaires.

---

20 www.webhelpdesk.com/clients.html

21 http://geekswithblogs.net/flanakin/articles/CompareWebTrackers.aspx, http://www.websina.com/bugzero/bug-defect-tracking.html, http://www.helpdesks.com/Completely_Web_Based/index.htm

Les cellules pâles représentent des forces de l'outil en comparaison des autres et les foncées, des faiblesses notables. Les différents outils qui ont été choisis dans le cadre ce travail sont comparés selon les critères suivants :

- ✓ Type : Type d'outils.
- ✓ Prix : Catégorie de prix.
- ✓ Adaptable : Capacité à être façonné selon les besoins du projet.
- ✓ Assignation : Personne et/ou groupe.
- ✓ Recherche : Billets ouverts et fermés (gestion de la connaissance).
- ✓ Zone publique : Possibilité de rendre un billet public.
- ✓ Fichier : Possibilité de joindre des fichiers.
- ✓ Rapport : Impression de rapport pour le gestionnaire.
- ✓ Temps : Gestion du temps des billets.
- ✓ Serveur : Fonctionne sous quelle plate-forme.
- ✓ Facilité : Facilité pour l'utilisateur et le gestionnaire.

5.2.7 Observations

Il est nécessaire de faire une bonne étude des besoins avant de se lancer dans le choix d'un système de contrôle qualité. Dans un contexte où le contact avec le client est fréquent et les délais de réponses doivent être rapides et contrôlés, par exemple le centre d'appel, la décision devrait tendre davantage vers une solution de centre d'assistance comme *Web Help Desk*, qui assurera la gestion du temps dans le contrôle des billets. Dans un cas comme un centre d'appel, le logiciel qui sera choisi deviendra le principal outil de travail des membres de l'équipe. Il est donc facilement justifiable de débourser quelques milliers ou dizaines de milliers de dollars pour un bon système qualité répondant aux besoins de l'organisation en fonction des services offerts.

Tableau 5.2 : Comparaison entre les outils de contrôle qualité

	Mantis	dotproject	One or Zero	Web Help Desk
Type	« Bug tracker »	Outil de gestion de projet	« Helpdesk »	« Helpdesk »
Prix	Gratuit (GPL)[22]	Gratuit (GPL)	Gratuit (GPL)	**Privé** : entre 3,000 $[23] et 15,000 $ + 650 $ à 3000 $ par année. **Hébergé** : entre 115 et 220$ / mois
Adaptable	Code source fourni Ajout de champs adaptés	Code source fourni	Code source fourni	Champs malléables, mais uniquement texte et liste. Code source non fourni
Assignation	Assignation à une personne et des projets	Très simple et bien fait (plusieurs personnes)	Billet ouvert à des groupes. Assignation à une personne	Assignation à une personne et prend en considération les absences. Procédure d'escalade
Recherche	Plusieurs critères et recherche rapide, profil utilisateur	Petite recherche simple pour les billets, mais aucune pour le reste (recherche difficile pour l'archivage)	Plusieurs critères et recherche rapide	Plusieurs critères et recherche rapide, profil utilisateur
Zone publique	Oui, très simple	Pas encore (en développement)	Oui, mais davantage pour les bases de connaissances	Très bien, simple et efficace
Fichier	Plusieurs	Plusieurs, mais difficile à trouver (parfois inaperçu)	Plusieurs	Plusieurs
Rapport	Très bien, plusieurs statistiques sur les ouvertures et les résolutions	Très bien, plusieurs formes de rapport, selon des critères établis. Diagramme de Gantt pour les tâches à faire	Très bien, plusieurs statistiques sur les utilisateurs et les billets	Excellent, plusieurs types de rapports sous plusieurs formats. Rapport client, techniciens, groupes, etc.
Temps	Temps inscrit à chaque intervention	Oui (tâches et billets) Calcul du temps global pour facturation	Oui, temps calculé pour chaque intervention	Très bien, alerte lorsqu'un billet est ouvert trop longtemps sans intervention
Serveur	Indépendant (langage interprété), serveur Web nécessaire	Indépendant (langage interprété), serveur Web nécessaire	Indépendant (langage interprété), serveur Web nécessaire	MacOS X Server, Windows Server, RedHat Linux, Sun Solaris 8
Facilité	Très simple	Très complexe, difficilement adaptable aux besoins mentionnés dans ce travail	Très simple, une fois les quelques subtilités comprises (supporteur et groupe de supporteurs)	Très simple, mais beaucoup d'options pour les membres de l'équipe

Par exemple, lors de mon passage chez Sogique, le centre de service provincial pour le réseau sociosanitaire, l'un de mes mandats était la programmation d'un outil de rapports puisant les informations dans leur centre d'assistance, C2, un outil conçu par CPL

22 The GNU General Public License (GPL) : http://www.gnu.org/copyleft/gpl.html

23 Tous les prix sont en dollars US.

Technologies[24], qui vaut près de 100 000 $. Pour une organisation comme Sogique, un outil comme *Mantis*, n'est pas du tout approprié, vu les besoins spécifiques et le niveau de performance dont ils ont besoin.

Pour plusieurs petites et moyennes équipes dont le principal mandat n'est pas le service à la clientèle, mais bien le développement, un logiciel comme *Mantis* peut très bien être envisagé. De plus, puisqu'il est libre, il est possible d'ajouter ou de modifier certaines parties, par exemple l'ajout de procédures, pour mieux l'adapter aux besoins de l'organisation.

Puisque nous nous penchons davantage sur le développement logiciel des équipes multidisciplinaires et que la possibilité d'adapter la solution est nécessaire pour l'élaboration de l'ensemble de l'infrastructure, *Mantis*, version 1.0, sera l'outil retenu dans la procédure d'utilisation.

5.3 Gestion de la connaissance

Tel que mentionné dans le chapitre précédent, la gestion de la connaissance est le processus le plus difficile à mettre en œuvre, puisqu'il dépend de l'infrastructure organisationnelle et informatique déjà en place et de la manière dont l'information et la communication sont gérées au sein de l'organisation. De plus, il faut savoir inciter les membres d'une équipe à partager leurs connaissances.

---

24 CPL Technologies : http://www.cpl-inc.com

L'American Productivity and Quality Center (APQC), qui publie plusieurs études sur la gestion de la connaissance, tire comme conclusion dans le livre *Creating a Knowledge-Sharing Culture* [APQC 1999], six facteurs pour inciter les membres d'une équipe à partager leurs connaissances :

- Le partage des connaissances doit appuyer la mission de l'organisme.
- Les réseaux informels constituent les moyens nécessaires au partage des connaissances.
- Les dirigeants et les gestionnaires doivent donner l'exemple dans le partage des connaissances.
- Le partage des connaissances doit s'harmoniser avec la culture générale.
- Les formules de travail doivent encourager le partage des connaissances.
- Il faut reconnaître et récompenser le partage des connaissances.

Parallèlement à l'implantation d'une culture organisationnelle sur le partage de l'information, il est aussi nécessaire d'établir une infrastructure capable de la supporter et de la rendre efficace. On peut lire dans le magazine Direction Informatique, l'analyse de Sebastien Ruest, vice-président du groupe de recherche en service chez IDC Canada, sur le retard des entreprises canadiennes en gestion de la connaissance :

> La plupart des organisations n'ont pas de stratégie fondée pour procéder à une gestion de façon correcte. D'un côté, elles ont établi des portails d'information et transféré des contenus de la paperasse et des classeurs vers le numérique et les bases de données, mais elles n'ont pas de systèmes d'accès logique ou continu à ces contenus ... [Ferland 2006]

Afin de mieux séparer les différents aspects du processus de gestion de la connaissance, nous séparerons la mise en œuvre de ce processus en trois éléments distincts :

1. La gestion des courriels.
2. La structure d'un projet.
3. Les mécanismes de recherche.

5.3.1 Gestion des courriels

Les courriels sont devenus, dans bien des cas, le moyen de communication de prédilection des organisations. L'information contenue dans ces courriels est souvent primordiale, unique et difficile à retrouver. Selon une enquête effectuée par FileNet Corporation auprès de 200 entreprises européennes ayant un chiffre d'affaires de plus de 500 millions de dollars :

> La majorité des entreprises ne sont pas capables d'archiver, de localiser et d'accéder aux emails, à la messagerie instantanée ou vocale, en dépit du fait que 58% des CIO reconnaissent que l'email est parfois leur seul enregistrement d'une information essentielle.
>
> [CompanynewsGroup 2006]

De plus, le courriel peut devenir un moyen efficace d'archiver électroniquement toutes les discussions au sein d'une entreprise. Les gestionnaires devraient encourager l'enregistrement des discussions concernant le projet; pensons au téléphone, au SMS, aux réunions informelles et 5 à 7, en demandant un bref courriel à leur intention ou adressé à un autre membre de l'équipe. Cette opération est rapide et permet de conserver une mémoire de ce qui est dit. Puisque le développement des jeux vidéo repose sur l'innovation technologique et sur l'imagination des concepteurs, ce principe de conservation des idées

est d'autant plus important que l'inspiration peut arriver à tout moment et son origine provient souvent de discussions informelles.

Changer les infrastructures de distribution de courriels déjà en place peut être une tâche difficile et peut demander beaucoup d'effort. Par contre, il est nécessaire d'incorporer des mécanismes pour archiver, rechercher et accéder aux courriels. Ces mécanismes diffèrent principalement en fonction du paradigme utilisé et donc, si les courriels sont stockés sur un serveur ou sur les postes du côté client. Les deux principaux paradigmes de distributions de courriels employés dans l'industrie sont le mode enligne qui consiste à se connecter sur un serveur pour accéder à distance aux courriels et le mode hors-ligne, qui se résume à utiliser un poste client pour télécharger les nouveaux courriels.

Avec le mode hors ligne, une fois les courriels transférés, ils sont ensuite supprimés du serveur ou archivés, en fonction du protocole utilisé, POP ou IMAP. Nous encourageons le protocole IMAP puisque ce dernier permet de conserver une version électronique de tous les courriels de l'entreprise. Ainsi, si un utilisateur supprime par mégarde un message important, il peut toutefois en retrouver une copie dans les archives du serveur. Ce mode est pratique puisqu'il libère considérablement le réseau si on le compare au mode enligne. Toutes les recherches dans les historiques de courriels se font de façon locale, donc ne surchargent pas le serveur. Maintenant, avec la puissance des serveurs et la robustesse des réseaux, cet argument est davantage à considérer pour les grandes organisations avec beaucoup de comptes utilisateur. Par contre, la recherche des courriels déjà téléchargés peut se faire sans avoir accès au réseau, ni à Internet, et elle fonctionne même si le serveur ou le

réseau est défectueux. Lorsqu'un membre de l'équipe utilise plusieurs postes clients, par exemple un ordinateur portable, Palm ou BlackBerry, il doit toutefois les synchroniser à un endroit pour organiser ses archives personnelles. Lorsqu'elles sont consolidées, il devrait alors les publier dans le SCV, dans un répertoire personnel.

Pour ce qui est du partage des informations pouvant être utiles aux autres membres de l'équipe, un dossier d'archives devrait être disponible dans le SCV commun du projet. Ainsi, toute personne recevant un courriel intéressant, une idée originale, des détails techniques, des informations sur la concurrence, peut classer ce message dans ce dossier, et ainsi le rendre visible à tous, une fois publié sur le SCV. La recherche se fera donc uniquement sur ce dossier et non sur l'ensemble des courriels de l'ensemble des membres de l'équipe, ce qui entraînerait une trop grande quantité d'information et rendrait la recherche inefficace.

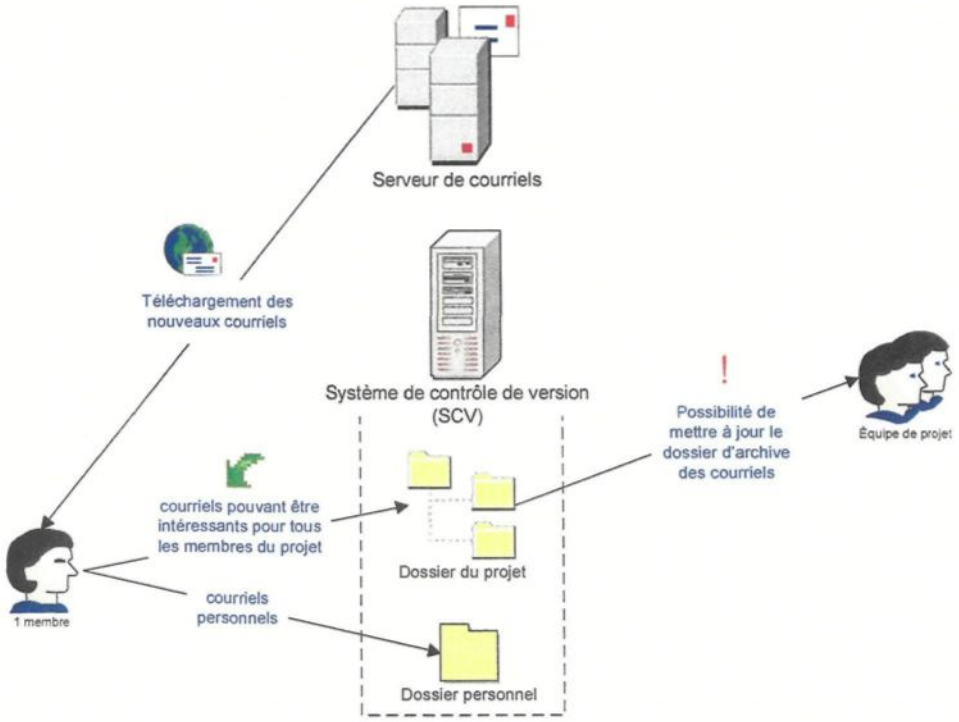
Lorsque l'approche enligne est déployée, l'archivage des courriels et la recherche se fait uniquement via le serveur. La visualisation et la gestion des courriels se fait grâce à une interface Web qui permet de rendre accessibles les messages sur n'importe quel poste disposant d'une connexion Internet et d'un navigateur, sans avoir à configurer quoi que ce soit. La plupart de ces interfaces permettent de télécharger une version locale des courriels et des contacts, pour offrir la possibilité de fonctionner en mode hors ligne. Cette fonctionnalité peut toutefois être désactivée pour augmenter le contrôle de l'information dans une industrie où le niveau de rotation de la main-d'œuvre est élevé. On n'aura qu'à

désactiver le compte d'un employé qui part ou qui est remercié, afin que ce dernier ne puisse partir avec les données contenues dans les courriels autant personnels que publics.

Dans le mode enligne, il est souhaitable de prendre en sauvegarde les courriels sur le serveur de façon incrémentale et complète, par exemple prendre une copie des nouveaux courriels à tous les jours et une copie complète de l'ensemble des courriels, une fois par semaine, car tous les registres se trouvent au même endroit. Par contre, ces copies ne servent qu'en cas de défaillance du serveur ou lorsqu'un courriel, effacé par inadvertance, doit être retrouvé.

La figure 5.5 illustre le partage des courriels lorsqu'un membre de l'équipe publie une nouvelle version du fichier d'archives de courriels. Bien sûr, certains outils de navigation de courriels permettent son partage. Par contre, le moyen présenté ici, est à un niveau d'abstraction supérieur et ne dépend plus de l'outil. Le changement d'outil peut donc survenir sans compromettre le système en place. La sauvegarde des courriels intéressants pour le projet est aussi simplifiée. Dans ce cas-ci, uniquement le SCV est à prendre en considération puisqu'il contient les courriels publiés, tous les dossiers personnels et les dossiers des différents projets.

Figure 5.5 : Partage des courriels dans un mode hors ligne



L'icône du fichier et ceux des dossiers racines, alertent qu'une version plus récente que celle sur leur ordinateur est disponible sur le SCV

Mise à jour et publication du ou des fichiers

Une interface Web efficace doit permettre à tous les propriétaires d'un compte utilisateur d'utiliser des dossiers publics. Ainsi, chaque courriel jugé intéressant peut être classé dans ce dossier et devenir accessible aux autres membres, et donc se greffer de manière transparente à leurs courriels ce qui leur permettrait d'interroger ces fichiers lors des recherches. Ces dossiers peuvent être regroupés selon les différents départements de l'organisation, comme la gestion, la programmation, la modélisation et d'autres. Ce mécanisme est donc instantané et ne demande aucun effort de la part des utilisateurs, contrairement au mode hors-ligne qui oblige les utilisateurs à mettre à jour leur dossier

d'archives sur le SCV. Il est cependant possible qu'un effort de développement soit nécessaire pour atteindre cet objectif, dépendamment de l'infrastructure de distribution de courriels enligne mis en place.

5.3.2 Structure d'un projet

Définir une structure précise de l'information permet aux nouveaux arrivants au sein d'un projet qui débute ou qui est déjà en cours d'être plus rapidement fonctionnels. Cette structure apporte une certaine homogénéité entre les différents projets qui facilitent les points de repère, puisque l'information se trouve toujours au même endroit. Ainsi, à l'aide d'un portail qui lui est propre, accessible uniquement aux participants du projet via une interface Web sécurisée, chacun des projets peut être présenté avec la même structure d'information et mis à la disposition de l'équipe qui y travaille.

Le portail peut être visible ou non à partir d'Internet, en fonction de l'existence de distances géographiques entre certains membres d'une équipe, mais aussi des besoins de sécurité, que nous verrons ultérieurement. Voici les différents aspects qui devraient être pris en considération pour la réalisation du portail :

- La vision et les particularités du projet;
- la structure de découpage des tâches;
- les membres de l'équipe et leurs fonctions;
- la base des défauts du projet;
- l'accès au système de contrôle de version (SCV);
- les discussions entourant le projet.

Il est important que tous les membres d'une équipe comprennent la vision et les particularités du projet, afin qu'ils travaillent dans le même sens. Les particularités du projet ici réfèrent aux fonctionnalités qui devront être développées et les spécifications qui seront nécessaires pour y arriver. Dans un contexte de jeux vidéo, les particularités peuvent représenter les modes de jeux disponibles, une description sommaire d'un ou de plusieurs environnements de jeu, des types de personnages à créer, des options qu'ils pourront effectuer. Andrew Stellman et Jennifer Greene, deux gestionnaires de projets logiciels et auteurs du livre : *Applied Software Project Management*, affirment par rapport à l'intérêt de créer un document sur la vision et les particularités d'un projet : « Quelques-uns des problèmes les plus communs et dispendieux, qu'un projet puisse éprouver sont dus à un manque de communication au niveau des objectifs de base du projet et des aspects particuliers à accomplir. » [Stellman-Greene 2005] – *Notre traduction.* Dans la méthode que nous proposons, une grande partie de ces objectifs apparaissent dans la définition du contenu du projet, décrit précédemment (4.5.1).

Nous suggérons que la structure de découpage des tâches apparaisse dans l'architecture afin que les membres de l'équipe en soient informés, mais aussi qu'ils y participent. La mise en œuvre de cette activité est abordée dans le processus de planification de projet (5.4). Un inventaire des ressources humaines de toutes les personnes participantes au projet avec leurs fonctions, devrait être établi, tenu à jour et rendu disponible aux autres. Ce mécanisme clarifie les rôles et identifie les personnes qui prennent les décisions sur chacun des aspects. Le point 5.5.1 (*Responsabilité et*

autorité) des normes ISO 9001:2000, indique très clairement cet aspect : « La direction doit assurer que les **responsabilités et 'autorité** sont définies et communiquées au sein de l'organisme » [ISO 9001 :2000]. Certains outils de planification de projet permettent de rédiger automatiquement cette liste en format Web. Sinon, cet inventaire peut être rédigé à l'aide d'un éditeur *html* quelconque et apparaître sous forme de liste ou d'organigramme. Il est aussi préférable d'inclure directement les coordonnées telles que le numéro de poste téléphonique et l'adresse de courriel de chacun des intervenants. Ce dernier point est particulièrement important lorsque les équipes sont séparées géographiquement ou que des contractants sont employés comme les musiciens et les acteurs.

Un lien vers la base de données des défauts du projet peut apparaître dans le portail pour améliorer l'homogénéité du processus de gestion de connaissance. Puisque nous avons suggéré précédemment un système de gestion des défauts basé sur une architecture Web, l'insertion d'un lien menant l'utilisateur directement aux problèmes non résolus du projet en question, peut se faire sans trop d'effort de programmation. Ainsi, toutes les parties prenantes peuvent accéder rapidement à la liste des défauts encore présents pour se confronter aux *objectifs qualité et aux exigences relatives au produit*, tel que mentionné dans les normes ISO.

Lorsque le SCV permet la navigation dans les versions via le protocole http, comme Subversion, il devient alors simple d'ajouter un lien sur le portail du projet. Cet ajout offre la possibilité d'incorporer beaucoup de contenu, sans avoir à faire de développement ou de maintenance sur le portail. Via l'interface Web du SCV, la documentation du projet peut

être accessible, ou encore la dernière version stable, en fonction des besoins. Dans le cas où le portail est accessible à partir d'Internet, même si le site est sécurisé il n'est peut-être pas recommandé d'insérer les fichiers de code source, pour ne pas être victime de vol ou d'espionnage industriel. En octobre 2003, le studio Valve annonçait sur son site Web[25], s'être fait voler le code source du jeu Half Life 2, à cause d'une faille dans Outlook, pour ensuite le voir distribué sur Internet. Cet incident a obligé le studio à retarder de plusieurs mois la sortie du jeu, mais aussi tous les jeux qui allaient utiliser le nouveau moteur graphique. Puisqu'une mésaventure de ce genre peut être dispendieuse pour une organisation, il est donc important de prendre les précautions nécessaires à ce niveau.

Afin d'augmenter la communication au sein des équipes de projet, il peut être intéressant de mettre en place un lieu de discussion où chacun peut s'exprimer et publier l'information qu'il trouve pertinente au projet. Dans un contexte de développement de jeux vidéo, cet endroit peut être utilisé pour discuter des particularités tant aux niveaux technique qu'esthétique, des personnages, des environnements graphiques, des trames sonores, du script, etc. Un forum électronique de discussion est encore plus pratique qu'une simple conservation de courriels puisque les sujets peuvent être classés. Un membre peut suivre l'évolution d'un thème et être alerté par un courriel, lorsqu'un commentaire vient d'être publié sur un sujet précis. L'utilisation d'un forum peut parfois s'avérer plus efficace qu'une réunion de type « brainstorming », puisque les participants peuvent répondre lorsqu'ils ont le temps ou l'inspiration et ce, sans la pression de devoir s'exprimer devant

---

25 http://www.halflife2.net/forums/showthread.php?threadid=10692

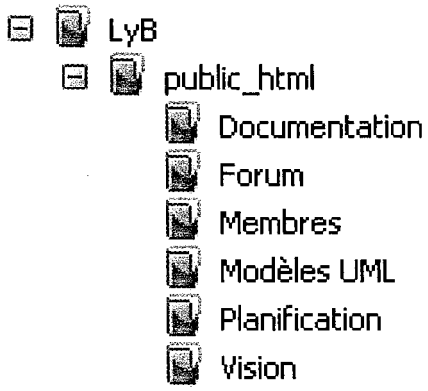
un groupe, ce qui freine parfois certaines personnes plus timides. On peut lire dans le *Knowledge Management Review* (mai-juin 2005), un article de J. H. Derick Sohn, qui décrit l'implantation d'un forum électronique à l'Institut des technologies avancées de Samsung, pour encourager les chercheurs provenant de différentes disciplines à interagir et échanger des idées.

Voici les avantages clés que Samsung a vus dans ce forum interdisciplinaire: [Sohn 2005]

- **Faciliter l'échange d'idées** : ce genre de forum représente un moyen intéressant pour échanger des perspectives différentes et rafraîchissantes parmi des chercheurs provenant de différents milieux et ayant des expertises diverses.
- **Stimuler la création de connaissance** : les idées exposées par des chercheurs provenant de milieux différents peuvent créer des opportunités intéressantes pour confronter des perspectives extrêmement différentes.
- **Accélérer le développement des idées** : les nouvelles percées rendent souvent nécessaires de nouvelles approches pour traduire des idées innovatrices en réalité. Dans un rendez-vous ouvert, les commentaires et les suggestions des experts de tous les secteurs technologiques peuvent être fixés facilement et rapidement.
- **Augmenter l'attribution de ressources en R&D** : en agissant l'un avec l'autre, les chercheurs sont exposés aux connaissances et aux expériences des autres experts et peuvent éviter de gaspiller du temps à reproduire des recherches qui ont déjà été faites.

La figure 5.6 représente une arborescence simplifiée que pourrait avoir le portail d'un projet avec les spécifications présentées précédemment. L'ensemble des pages de cette arborescence est bien sûr privé, protégé et n'est accessible qu'aux membres de l'équipe de projet. On ne voit aucun dossier pour l'accès à la dernière version stable à partir du SCV ni pour les défauts à partir du système de gestion des défauts, puisque ce sont deux systèmes indépendants. Leur accès se ferait donc uniquement via un lien *URL* standard. Le dossier « Forum » ne sera présent que lorsque le forum électronique est indépendant des autres projets. Dans le cas où le forum est partagé par tous les membres de l'organisation, même ceux qui ne travaillent pas sur le projet, le forum ne sera installé qu'une seule fois et pourra être accessible directement à partir du portail de tous les projets. Le dossier « Forum » ne comportera qu'un fichier redirigeant l'utilisateur vers la section du forum global réservé au projet. Le contenu des autres dossiers dépendra des solutions retenues pour créer la documentation et la planification, que nous verrons dans le point 5.3.4 « Outils génériques ».

Figure 5.6 : Arborescence du portail d'un projet



5.3.3 Les mécanismes de recherche

Pour l'implantation d'un mécanisme de recherche efficace sur une architecture basée sur le Web, on ne peut passer à côté des moteurs de recherche. Le principe de ces moteurs est que l'usager interroge une base de données contenant l'indexation des sites Web qui y sont inscrits. Selon David Green, dans un article paru dans le *Online Information Review*, les différents moteurs de recherche ont trois composantes primaires : [Green 2000]

- Des robots d'indexation (*spiders*) qui examinent les sites Web.
- Une base de données *ou index*, des sites web inscrits.
- Un logiciel d'interrogation et de récupération.

Le rôle du robot est d'alimenter la base de données du moteur de recherche en explorant les mots et les liens vers de nouvelles pages pour les indexer avec l'URL de la page où ils apparaissent. La plupart de ces moteurs de recherche (*Google, Yahoo, MSN Search, etc.*) sont d'abord fait pour indexer des pages de partout sur Internet. Pour les rendre accessibles à tous les utilisateurs, leurs robots iront donc chercher de l'information sur tous les sites qui y sont inscrits, pour ainsi mettre à jour leur propre base de données.

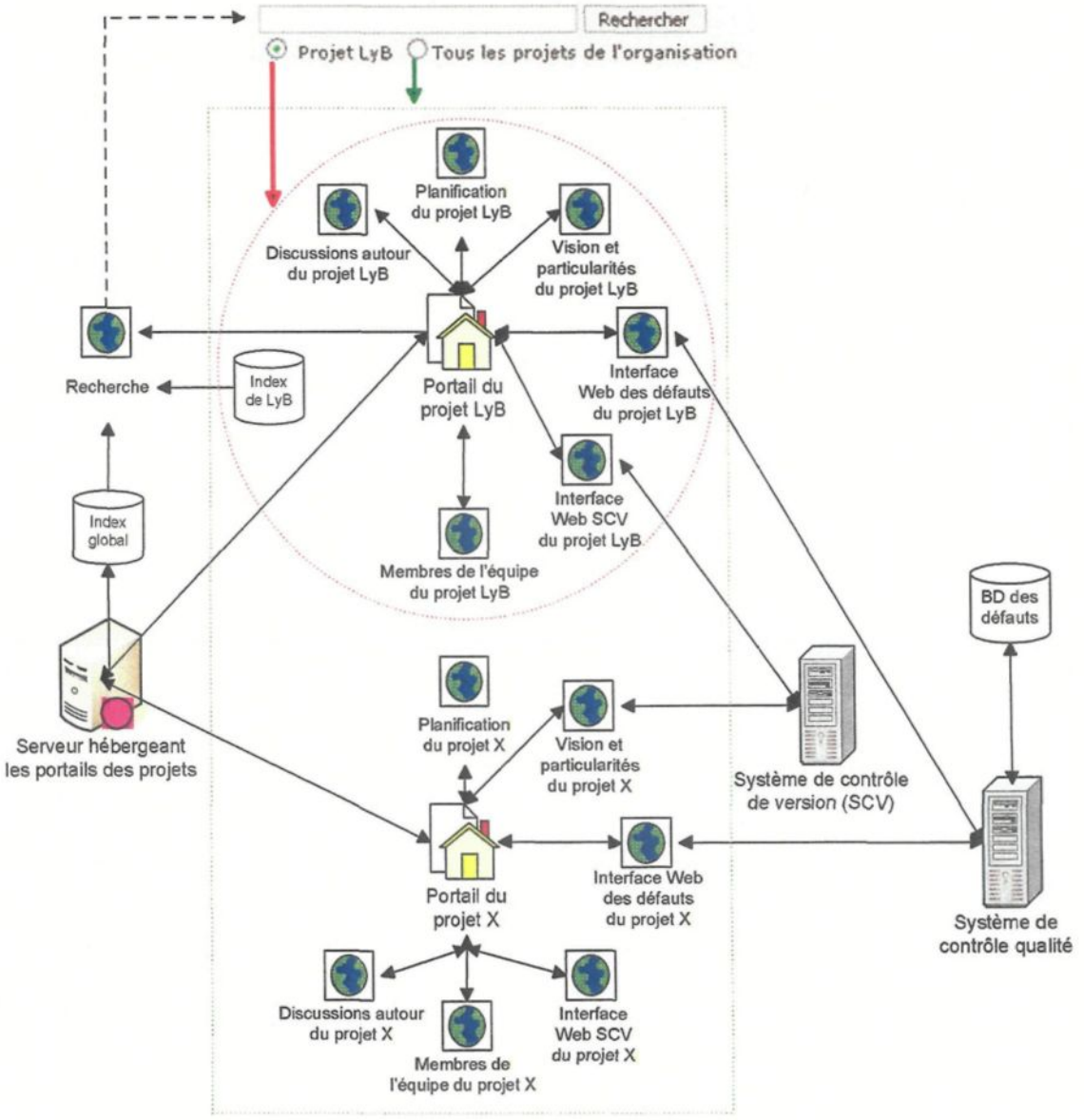
Puisque le portail que nous suggérons n'est pas accessible aux utilisateurs qui ne sont pas impliqués dans le développement, il est donc fortement déconseillé d'utiliser un moteur de recherche dont l'indexation se fait via un site Internet extérieur. La plupart des robots ne trouveraient pratiquement rien sur le portail, car ils n'indexent pas les pages protégées par un mot de passe. Par contre, quelques-uns d'entre eux permettent de les indexer grâce à un nom d'utilisateur et un mot de passe. Certaines informations peuvent

alors être visibles via la description des pages et créer quelques fuites. Il est donc préférable d'utiliser un outil qui permet d'héberger la base de données directement sur notre serveur, et de limiter les robots aux frontières de notre système. Pour augmenter l'efficacité du processus, il est possible de créer un niveau d'abstraction dans la recherche et de diviser les différents index. De cette façon, chaque projet possède sa propre base de données et une base globale indexe l'ensemble des projets de l'organisation.

L'efficacité de la récupération des données est un point qui est difficilement calculable lorsque vient le temps de choisir une solution de recherche. Pourtant, ce point est important puisqu'il déterminera la qualité des réponses en fonction des questions posées au moteur de recherche. Chacun des outils devrait être testé sur le même site puis être questionné avec les mêmes critères pour établir lequel filtre le mieux les données pour en afficher l'information recherchée. Cette étape est cependant difficilement réalisable puisque chacune des solutions, que nous verrons plus loin dans ce chapitre, vaut plusieurs milliers de dollars.

La figure 5.7 démontre un exemple de portail d'un projet (*LyB*), avec une structure telle que proposée précédemment. La recherche peut se faire autant sur le projet en cours que sur l'ensemble des projets de l'organisation. Il est important de noter que le SCV, le système de gestion des défauts, la base de données des défauts, ainsi que les différents index peuvent tous être réunis sur le même serveur physique. Les représentations sont purement schématisées pour expliquer les concepts. La recherche dans les courriels n'apparaît pas dans cette figure puisqu'elle dépend du mode, enligne ou hors-ligne, établi.

Figure 5.7 : Structure d'un projet avec son portail et mécanisme de recherche



5.3.4 Outils génériques

Les interfaces Web des systèmes de planification, de gestion des défauts et de gestion des versions disponibles des documents sont générées par leurs outils respectifs.

Dans le cas du document sur la vision et les particularités, ainsi que la liste des participants au projet, un simple générateur de pages *html* pourrait être utilisé. Que ce soit une suite bureautique pouvant exporter des documents Web telle que *Microsoft Office, Sun StarOffice* ou même *OpenOffice.org* [26], ou un générateur *html* plus spécialisé tel que *Macromedia Dreamweaver, Adobe GoLive, Microsoft FrontPage*, cela ne fait que très peu de différence. Par contre, l'utilisation de wiki pour rédiger ce genre d'information commence à s'implémenter de plus en plus dans l'industrie des technologies de l'information.

Wiki

Le système wiki, inventé par Ward Cunningham en 1995, permet de visualiser et d'éditer des pages de façon dynamique à partir d'un navigateur Web. Tout individu possédant une autorisation peut alors créer des pages ou modifier les pages existantes. Des restrictions peuvent toutefois être instaurées pour limiter les droits des différents groupes d'utilisateurs. Chacune des pages possède son historique, qui permet aux créateurs des pages de revenir aux versions antérieures ou de voir les modifications, et l'identité de leur auteur. Le wiki est pratique lorsque plusieurs personnes ont à travailler sur les différents documents, sans les obliger à connaître le langage html ou à posséder un éditeur.

---

[26] OpenOffice.org : suite bureautique libre, élaborée à partir de *StarOffice* et distribuée gratuitement sur Internet.

Il existe plus d'une centaine de systèmes de wiki; la plupart sont gratuits et libres. Les principales caractéristiques à prendre en considération lors du choix d'un système wiki sont :

- Configuration nécessaire qui doit être compatible avec l'architecture en place.
- Éditeur de balise de texte ou *WYSIWYG*[28]. Il permet aux usagers d'éditer des pages wiki sans connaître nécessairement la syntaxe, facilite l'utilisation.
- Possibilité d'attacher / télécharger des fichiers.
- Licence d'utilisation dont quelques-uns sont payants.
- Facilité d'installation.
- Gestion des droits d'utilisateurs.

Les critères sélectionnés sont déterminés en fonction de ceux que l'on retrouve principalement sur Internet et les comparaisons sont établies grâce aux sites *Wikipedia*[29] et *Wikimatrix*[30]. Ce dernier permet de comparer des systèmes wiki, en les sélectionnant dans une liste, en fonction de critères déterminés.

Les différents systèmes wiki présentés dans le tableau 5.3 sont choisis pour leur intégration d'un éditeur de balise de texte, ou mieux encore, *WYSIWYG)*, ou la possibilité d'ajouter un. Ce critère est important dans le contexte où les utilisateurs ne sont pas tous à l'aise avec l'apprentissage d'un langage de balise de texte, qui pourrait être un facteur suffisant pour créer une résistance à l'utilisation du système. De plus, tous les systèmes comparés permettent la gestion des droits d'utilisateurs pour des raisons de sécurité.

---

28 *WYSIWYG*: De l'anglais : « What You See Is What You Get ».

29 http://en.wikipedia.org/wiki/Comparison_of_wiki_software

30 http://www.wikimatrix.org

Tableau 5.3 : Comparaison des systèmes wiki

	Configuration nécessaire	**Éditeur *WYSIWYG***	**Attacher / télécharger un fichier**	**Coût de la licence**	**Facilité d'installation**
Confluence (v. 2.2)	Apache, JDK 1.4+, J2EE 1.3	Oui	Oui	1 200 $ US	Simple, assistant de configuration
Corendal Wiki (v. 2.0)	Java, MySql ou Oracle, Tomcat ou Oracle Application Server	Oui	Oui	0 $	Modérément simple
MediaWiki **(v. 1.6.7)**	Serveur Apache ou IIS, PHP, MySql	Possibilité d'en ajouter	Oui	0 $	Modérément simple
MoinMoin (v. 1.5.3)	Apache, ou IIS, ou quelques autres	Oui	Oui	0 $	Modérément simple
Perspective (v. 0.906)	IIS, .Net 1.1	Oui	Oui	0 $	Modérément simple
PmWiki (v. 1.2.10)	Serveur Apache ou IIS	Possibilité d'en ajouter	Optionnel	0 $	Simple
WackoWiki (R. 4.2)	Serveur Apache ou IIS, PHP et MySql	Oui (*WikiEdit*)	Oui	0 $	Rapide et simple
XWiki (v. 0.9.840)	Tomcat, Java, MySql ou HSQLDB	Oui	Oui	0 $	Simple

La configuration nécessaire est le critère le plus important à prendre en considération parmi les outils présentés dans le tableau 5.3, puisque les différents systèmes sont similaires en plusieurs points. Il est donc préférable de choisir un outil fait pour l'architecture informatique en place. *MediaWiki*, le système derrière l'encyclopédie libre

Wikipédia[33], sera utilisé dans notre guide d'utilisation, pour sa configuration nécessaire, qui est celle dont nous disposons et son éditeur de texte capable d'insérer du *LaTeX*[34], donc des formules mathématiques.

Bien que les discussions autour d'un projet pourraient être gérées à l'aide du système de wiki, les forums électroniques sont mieux adaptés à ce genre d'activités. Ils permettent de classer les différents sujets, de publier une affirmation et de gérer les réponses ou les réactions face à celle-ci.

La plupart des forums électroniques intègrent directement un moteur de recherche. Ce moteur peut alors être utilisé pour raffiner les recherches afin de chercher uniquement dans les forums et non dans le système en entier.

Forum électronique

Les principaux critères de sélection d'un forum électronique sont sensiblement les mêmes que ceux des systèmes wiki. La configuration nécessaire à l'installation du forum, la présence d'un éditeur de balise de texte ou *WYSIWYG*, la possibilité d'attacher et de télécharger un fichier à un message sont, tout comme dans le choix des systèmes wiki, des facteurs à prendre en considération. L'indexation des messages pour améliorer l'efficacité des recherches, la division ou la fusion des réponses, lorsqu'elles ne répondent pas à la bonne question, ou deux réponses se révélant quasi identiques, la recherche automatique des sujets similaires et l'alerte par courriel lorsqu'une réponse est ajoutée, sont des caractéristiques pouvant aussi influencer le choix d'un forum électronique. Le tableau 5.4

---

33 *Wikipédia* : http://fr.wikipedia.org/wiki/MediaWiki

34 *LaTeX* : Langage de composition de documents, couramment utilisé pour la rédaction scientifique.

établit une comparaison entre plusieurs forums suggérés sur Internet, chacun d'eux dispose d'au moins un éditeur de balise de texte, de la possibilité d'attacher et télécharger un fichier et d'activer l'alerte par courriel lors d'une notification. vBulletin, est un forum électronique très efficace et chaudement recommandé par ses utilisateurs[37] et qui répond à tous nos critères de sélection. Nous utiliserons toutefois *MyBB* dans notre guide d'utilisation, puisque sa dernière version, sortie récemment (septembre 2006), répond à la majorité de nos critères, mais surtout est gratuite sous licence *GPL*.

Tableau 5.4 : Comparaison des forums électroniques

	Configuration nécessaire	**Éditeur WYSIWYG**	**Indexation des messages**	**Division et fusion**	**Recherche automatique de sujets similaires**	**Coût de la licence**
vBulletin *(V 3.5.4)*	PHP, MySql	Oui (optionnel)	Oui	Oui	Oui	160 $ US 300 $ US par année pour le support
IPB *(V 2.1.3)*	PHP *(avec MySql ou MS SQL Server ou Oracle 9i)*	Non	Oui	Oui	Non	185 $ US
FuseTalk *(Enterprise)*	ColdFusion Server *(avec Oracle 9i, MySqL ou MS SQL Server)* ouMicrosoft .Net Framework *(avec MS SQL Server)*	Oui	Uniquement avec *ColdFusion* pour le moment.	Non	Oui	5000 $ US
FUDforum *(2.7.5)*	PHP *(avec MySql ou PostgreSQL)*	Non	Non	Non	Non	0 $ *(GPL)*
WowBB *(V 1.65)*	PHP, MySql	Oui	Oui	Oui	Non	99 $ US
WoltlabBB *(V 2.3)*	PHP, MySql	Oui	Oui	Oui	Non	205 $ US
E-Blah *(Platinum 9.71B)*	Perl	Non	Non	Oui	Non	0 $ *(GPL)*

37 http://www.vbulletin.com/customers.php

Moteur de recherche

Comme il a été mentionné précédemment dans ce chapitre, nous ne conseillons pas d'utiliser un moteur de recherche externe au système via Internet pour des raisons de sécurité évidentes. Puisque l'infrastructure proposée est utilisée à l'interne (Intranet), l'index des pages doit donc se retrouver à l'intérieur des bastions de sécurité de l'entreprise. Il existe principalement deux types d'outils génériques pouvant indexer les pages provenant de l'Intranet. Premièrement, des serveurs dédiés, dont les tâches se résument à indexer les documents disponibles à l'intérieur d'un réseau ou d'un serveur et d'afficher les résultats des recherches des utilisateurs. Ces solutions sont très spécialisées, performantes, mais assez dispendieuses, en fonction du nombre de fichiers que l'on désire indexer. Deuxièmement, des logiciels que l'on peut installer et configurer directement sur nos serveurs. Ce type est le moins coûteux, mais dépend énormément du serveur sur lequel il est installé et semble, en général, moins performant que les serveurs dédiés. Le tableau 5.5 présente les différents moteurs de recherche trouvés sur Internet.

Parmi les différents critères de comparaison que nous avons retenus :

- Requêtes par minute : moyenne de requêtes maximum auxquelles le système est capable de répondre en une minute).
- Formats de fichier indexé : formats que les robots peuvent indexer et dans lesquels le moteur peut rechercher.
- Type d'outil : logiciel ou serveur.
- Limite par jour : liens cliqués à la suite d'une recherche.
- Nombre maximum de fichiers indexés : en fonction des licences ou du matériel.
- Prix approximatif : les devises utilisées sur les sites n'ont pas été converties.

La taille des projets, des équipes, des équipements informatiques en place et les moyens financiers de l'organisation doivent être pris en considération pour le choix d'un outil de recherche, puisque les montants sont plus importants que pour les autres types d'outils présentés dans ce travail. Pour une petite équipe, des logiciels comme *Webglimpse* ou *Webinator* peuvent faire le travail. Une organisation ayant plusieurs équipes, plus d'une centaine d'employés et pilotant plusieurs projets, peut penser à se munir d'un appareil comme ceux proposés par *Google*, *Thunderstone* ou *Novel* (*Index Engines*).

Tableau 5.5 : Comparaison des moteurs de recherche

	Requêtes par minute	**Formats de fichier indexé**	**Type d'outil**	**Limite par jour**	**Nombre maximum de fichiers indexés**	**Prix approximatif**
Google Mini	50	220 formats différents	Serveur	Aucune	Entre 50 K et 300 K	Entre 2000 € et 9000 €
Google Search Appliance *(GB-1001)*	300	220 formats différents	Serveur	Aucune	1,5 M	30000 € et plus
Google Search Appliance *(GB-8008)*	1000	220 formats différents	Serveur	Aucune	15 M	Environs 200 000 €
Webglimpse	60 [39]	Html, Pdf, Word, et plusieurs autres	Logiciel	Aucune	9 MM	Entre 400 $ US et 4000 $ US
Thunderstone SBE	1000	Html, Pdf, Word, Flash, Javascript et plusieurs autres	Serveur	Aucune	50 K	5000 $ US
Thunderstone Search Appliance *(APP-250)*	2500	Html, Pdf, Word, Flash, Javascript et plusieurs autres	Serveur	Aucune	6 M	10000 $ US et plus
Index Engines *(ES-100)*	3000	Office, Pdf, courriel, texte compressé, images et autres	Serveur	Aucune	4 M	25000 $ US
Webinator *(Commercial)*	Non applicable[40]	Html, Pdf, Word, Flash, Javascript et plusieurs autres	Logiciel	10 K	20 K	700 $ US

39 Dépendant du nombre de fois que les mots recherchés sont indexés et de la puissance du serveur.

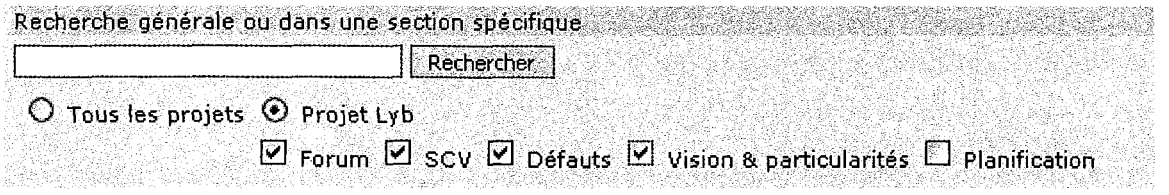
40 En fonction du serveur sur lequel le logiciel est installé

Une autre avenue est aussi possible pour effectuer la recherche au sein des différents systèmes de l'infrastructure. Elle consiste à utiliser et fusionner les différents outils de recherche intégrés dans chacun des systèmes de l'infrastructure tels que la gestion des défauts, le contrôle de version, wiki, les forums de discussion, pour en faire une seule recherche. Cette méthode demande un certain effort de développement, mais peut s'avérer efficace et moins dispendieuse que l'achat d'un serveur dédié.

Lorsque l'infrastructure est séparée sur plusieurs serveurs, cette technique de recherche sera améliorée puisque les recherches seront partagées par les différentes machines, et non par une seule comme les deux premiers types d'outils. Avec cette alternative, il est aussi possible de spécifier les sections et les limites de la recherche.

La figure 5.8 représente le genre de critères que l'on peut insérer dans nos recherches. Dans cet exemple, on suppose que la recherche se fait à partir du portail du projet *Lyb*. Il est alors possible de chercher à l'intérieur de ce projet, dans des sections spécifiques, ou dans l'ensemble des projets de l'organisation.

Figure 5.8 : Critères de recherche



L'utilisation de logiciels libres facilite cette alternative, puisque le code source est disponible. Dans le cas d'outils propriétaires, l'effort de développement nécessaire est plus

important, à moins que les compagnies conceptrices de ces logiciels collaborent. Puisque tous les outils présentés dans cette architecture sont libres et pour des raisons économiques, cette dernière méthode sera celle retenue pour le guide d'utilisation.

5.4 Planification de projet

Nous avons observé au point précédent sur la gestion de connaissance que les documents simples, réalisés auparavant par un logiciel de traitement texte, peuvent être distribués et mis à jour directement par un navigateur web via un système wiki. L'utilisation de ce genre d'outil est aussi envisageable pour la définition du contenu et l'estimation des charges. De plus, un forum électronique peut être mis en place pour alimenter les discussions entourant le contenu d'un projet et pour faire jaillir de nouvelles idées.

En ce qui a trait à la gestion des risques, nous considérons qu'il est peut-être hasardeux de la rendre disponible à l'interne d'une équipe, puisque certaines sources d'inquiétude peuvent provenir des capacités ou des aptitudes de certaines personnes. Il devient alors politiquement inconfortable d'identifier ces risques et de les contrôler dans une architecture ouverte et accessible à tous les membres d'un projet. Pour le moment, aucune solution ne semble s'imposer dans l'industrie afin d'améliorer la gestion des risques en développement logiciel. La plupart des personnes que nous avons interrogées à ce sujet affirment n'utiliser qu'un logiciel de traitement de texte, ou une solution maison faite à partir d'un tableur ou d'une base de données. Il est à noter qu'il serait possible de réaliser la gestion des risques grâce à un système wiki sécurisé, disponible uniquement aux

gestionnaires qui ont un niveau d'accès suffisant. Un exemple de solution maison développée avec *MS Access* pour piloter les risques d'un portefeuille de projet est présenté en annexe (*Annexe 2 : Gestion des risques*).

5.4.1 Structure de découpage des tâches

La structure de découpage des tâches est l'aspect de la planification de projet ayant les besoins les plus spécifiques et nécessite une solution plus spécialisée. Lors de nos différentes discussions avec les gens de l'industrie, nous avons pu noter que la majorité des gestionnaires de projet utilisent les diagrammes de Gantt pour élaborer leur structure de découpage. La plupart s'en servent à deux niveaux, l'un global et simplifié pour présenter les principaux livrables et les différentes versions à développer, et un autre plus détaillé pour gérer l'ensemble des activités. Le corpus de connaissances du PMI [PMI 2004] précise qu'il faut toutefois y aller avec modération quant au niveau de détail, et adapter le degré de décomposition en fonction de la taille et de la complexité du projet ou du livrable : « À mesure que le travail est décomposé vers des niveaux inférieurs de détail, sa planification, son management et sa maîtrise s'améliorent. Cependant, une décomposition excessive peut mener à un effort de management non productif… ».

5.4.2 Outils de planification

Il est difficile d'aborder les solutions de planification de projet sans prendre en considération *Microsoft Project (MS Project)* qui est, à ce jour, l'outil le plus populaire auprès des gestionnaires de projet. Ce logiciel permet la gestion de portefeuille de projets, l'élaboration de diagrammes de Gantt en mode graphique et peut envoyer les tâches aux

personnes concernées en se synchronisant sur Outlook. De plus, la version EPM *(Enterprise Project Management)* permet d'installer un serveur Web et de rendre l'application enligne et accessible via un navigateur Internet. Par contre, cette solution ne fonctionne que sous une infrastructure Microsoft. Le système d'exploitation du serveur doit être *Windows Server 2000,* ou une version plus récente, *MS Project* ne peut être installé que sur des postes *Windows*, chacun des postes devant être muni d'une licence d'utilisation de *MS Project*, les tâches ne sont synchronisées que sur *Outlook* et l'application Web n'est accessible que via le navigateur de *Windows*, *Internet Explorer*. Bref, seules les personnes ayant des ordinateurs munis de Windows peuvent utiliser *MS Project* et ses composantes sans soucis d'incompatibilité. Puisque plusieurs personnes de l'industrie du développement de jeux vidéo fonctionnent dans un environnement autre que ceux offerts par *Microsoft*, nous nous intéresserons aux autres solutions basées sur une architecture Web, trouvées à partir du volume de Stellman et Greene [Stellman-Greene 2005] et sur Internet.

Voici les critères que nous avons établis pour comparer les différents outils de planification du tableau 5.6 :

Outils :

- Licence : type de licence et prix approximatif d'utilisation (tous les montants sont en dollars américains).
- Version : version testée.
- Facilité d'utilisation : échelle de 1 à 5 (1 étant très compliqué et 5 très facile d'utilisation).
- Support : support par courriel ou téléphone.
- Recherche : intégration d'un système de recherche dans l'outil.

Gestion des tâches :

- Assignation des tâches : assignation ou réassignation des ressources à des tâches.
- Avancement : niveau d'accomplissement des tâches et l'effort réel effectué (temps / personne).
- Retour d'information : ajout de commentaires sur la réalisation d'une tâche (raison d'un retard ou d'une avance, problèmes envisagés ou rencontrés, etc.).
- Alertes : message d'alerte lors d'une assignation, d'un dépassement, ou d'un problème avec une tâche.
- Récurrentes : gestion des tâches récurrentes.

Planification :

- Diagramme de Gantt : possibilité d'afficher un diagramme de Gantt avec les dépendances fonctionnelles.
- Rapports et statistiques : présentation des statistiques sur le projet (temps utilisé, montant total, etc.).
- Livrables : affichage particulier pour bien visualiser les divers livrables.
- Chemin critique : visualisation des tâches cruciales selon la méthode du chemin critique.
- Budget : capacité de l'outil à gérer les coûts en fonction des ressources utilisées.

Une étude spécifique des besoins est nécessaire avant de choisir une solution de planification de projet. Certains outils présentés dans le tableau 5.6 offrent beaucoup plus de fonctionnalités que celles énumérées dans nos critères de sélection. Il est toutefois important de noter que certaines de ces fonctionnalités peuvent engendrer de la confusion chez les utilisateurs lorsqu'elles se retrouvent à plusieurs endroits dans l'infrastructure. Par exemple, *dotProject* possède son propre forum électronique et son système de gestion des défauts. Certaines personnes peuvent alors être tentées d'utiliser ces fonctionnalités pour entrer des discussions ou des défauts, ce qui a comme effet de disperser l'information entre plusieurs systèmes alors que l'objectif premier est de la centraliser à un seul endroit.

Tableau 5.6 : Comparaison des outils de gestion de projet basés sur le Web

	dotProject	Ganttproject	@Task	eStudio	AceProject	WebAsyst	Vertabase
Licence	GPL 0 $	GPL 0 $	Entre 200$ et 395$ par utilisateur / année	495$ / mois (*hébergé*). 10 000$ (*réseau interne*)[41]	100$ / mois (*hébergé*). Entre 1 295$ et 3 000$ (*réseau interne*)[42]	Environ 500$ / mois (hébergé). 1750$ (réseau interne)	4 800$[43] + 5 000$ / année
Version	2.0.4	1.11.1	4	6	4.2	*ND*	pro
Facilité d'utilisation	3.5	5	3.5	4	4	4	2
Support			✓	✓	✓	✓	✓
Recherche	✓		✓		✓		
Assignation des tâches	✓	✓	✓	✓	✓	✓	✓
Avancement	✓	✓	✓	✓	✓		✓
Retour d'information	✓	✓	✓	✓	✓		✓
Alertes	✓		✓	✓	✓		
Récurrente							
Diagramme de Gantt	✓	✓	✓[44]	✓	✓	✓[45]	✓
Rapports et statistiques	✓		✓	✓	✓	✓	✓
Livrables	✓	✓	✓	✓	✓	✓	✓
Chemin critique			✓				✓
Budget	✓		✓	✓	✓		✓

Un gestionnaire, désirant uniquement élaborer et publier des diagrammes de Gantt, pourrait très bien utiliser *Ganttprojet* puisqu'il est rapide, efficace et simple d'utilisation.

---

41 Fonctionne avec Windows Server 2000 et SQL Serveur, prix pour moins de 250 utilisateurs + 1500 $ US pour les frais d'installation.

42 Fonctionne avec une base de données SQL Serveur ou Access, les prix varient en fonction du nombre d'utilisateurs.

43 Montant pour 5 gestionnaires et 20 utilisateurs

44 Diagramme de Gantt, mais sans l'affichage des dépendances fonctionnelles

45 Idem

L'application fonctionne en mode client, mais peut aussi être utilisée à l'aide d'un navigateur. L'enregistrement des projets se fait sur un fichier, qui peut par la suite être inséré dans le SCV. Par contre, cet outil assez limité ne possède aucune autre fonction ce qui est peu pratique pour un gestionnaire de projet voulant approfondir sa planification.

@task est la solution qui compte le plus de fonctionnalités intéressantes à notre avis. Le système peut être installé sur un serveur Intranet sur la plupart des architectures courantes, *Windows*, *Linux*, *Mac OS*, avec plusieurs systèmes de base de données tels que *Microsoft SQL Serveur 2000*, *Oracle* et *MySql*, et l'on peut y accéder grâce aux navigateurs les plus populaires tels qu'*Internet Explorer*, *Firefox* et *Safari*. De plus, l'importation et l'exportation des données dans *MS Project* permet aux gestionnaires habitués à ce logiciel de continuer à l'utiliser sans problème; la version française de *MS Project* crée par contre certains conflits. Cet outil n'est pas le plus simple à utiliser lors d'un premier contact, mais les grandes lignes en seront expliquées dans le guide d'utilisation de notre infrastructure présentée en annexe.

5.4.3 Observations

La plupart des solutions de planification de projet présentées dans ce travail furent développées ou sont encore en amélioration en 2006, ce qui nous laisse croire que plusieurs innovations devraient voir le jour chez les concurrents de *MS Project*. Ce produit de *Microsoft* demeure toutefois efficace, comme le prouve la palme d'or selon le

TopTenREVIEWS 2006 [46], dans la mesure où l'infrastructure de l'organisation est centrée uniquement autour des technologies *Windows*.

Malgré les nouvelles fonctionnalités qui s'ajoutent sans cesse aux outils existants, il est important de ne pas perdre de vue les principaux objectifs du processus de planification qui sont de définir le contenu du projet, d'en estimer la charge de travail et les risques potentiels, de le décomposer en tâches qui seront assignées à des ressources humaines et matérielles et, en dernier lieu, que les personnes qui en ont la charge alimentent la planification en détaillant leurs interventions, et en donnant leurs opinions sur les échéances et l'avancement des tâches, ainsi que sur les risques qui y sont associés.

5.5 Synthèse des résultats

Ce chapitre a fait un survol de plusieurs éléments de mise en œuvre pour répondre aux différents besoins énumérés précédemment. La stratégie proposée dans ce travail à chacun des processus, respecte les objectifs présentés en début de chapitre, soit l'indépendance de l'architecture informatique nécessaire à l'implémentation des outils, la possibilité d'établir une communication entre chacun d'eux et qui est principalement basée autour des technologies Web. Bien que les outils abordés dans ce chapitre sont assez flexibles pour être installés dans la majorité des environnements de développement de logiciels, il n'en demeure pas moins qu'un effort considérable doit être déployé pour les intégrer dans l'architecture déjà en place, pour former les utilisateurs et s'assurer que chacun des intervenants les utilisent correctement selon les exigences de l'organisation

---

46 TopTenREVIEWS 2006 : http://project-management-software-review.toptenreviews.com/

CHAPITRE 6

CONCLUSION

Ce mémoire a étudié des stratégies à employer pour contrer les difficultés liées aux processus ciblés comme problématique par l'industrie du développement de logiciels, et plus spécifiquement, dans un contexte comme celui des jeux vidéo. Les méthodes de développement peuvent différer d'une équipe à l'autre et le niveau de formalisme utilisé dans la gestion de projet informatique dépend souvent de la volonté de ses gestionnaires à suivre un cadre de référence ou non. Ces cadres de référence provenant du monde de la gestion ou de l'ingénierie logicielle, souvent critiqués et laissés de côté par les développeurs en raison de leur lourdeur et des efforts nécessaires à la satisfaction de leurs exigences, apportent pourtant des éléments de réponse aux problèmes concernant la diminution des défauts, le transfert d'information entre les équipes, la réutilisation des éléments existants, etc.

L'un des problèmes soulevés face aux méthodes de développement et aux cadres de référence est qu'ils présentent des concepts souvent abstraits devant être pris en considération lors de la réalisation de projets, sans pour autant indiquer de moyens pour y arriver, ni de stratégies pour les mettre en œuvre. Nous avons donc réuni les problématiques énumérées par les professionnelles de l'industrie lors de discussions, à l'aide d'un questionnaire à leur intention, par l'information recueillie lors des *Game Developers Conference*, ainsi que lors de notre expérimentation d'un an avec une équipe d'étudiants impliqués dans la conception d'un jeu. Cet exercice nous a permis de mieux comprendre les réalités de l'industrie et les difficultés liées à la mise en oeuvre des concepts au sein d'équipes déjà en fonction et de mieux saisir les contraintes qui amènent

les artisans du jeu vidéo à mettre de côté les grands cadres de référence. Le contexte du développement de jeux vidéo est particulier puisque les équipes sont multidisciplinaires (principalement de programmeurs et d'artistes) dont les comportements et les habitudes technologiques sont souvent opposés. La complexité des projets, la rapidité avec laquelle ils doivent être réalisés, les changements fréquents de main d'œuvre ainsi que de plateformes technologiques apportent aussi leurs lots de complications devant être prises en considération lors de l'élaboration de l'infrastructure de développement. Les solutions proposées dans ce travail furent donc pensées à partir de ce contexte pour se coller aux réalités des gens de l'industrie.

À partir de ces recherches, il nous fut possible d'élaborer les balises qui allaient limiter notre champ d'études pour focaliser essentiellement sur les problématiques énumérées par les professionnels du domaine des jeux vidéo et non sur celles de l'ingénierie logicielle en général. Cette étape nous a permis de regrouper les difficultés trouvées en quatre processus distincts. Ces processus sont la gestion des documents et des modifications, la gestion des défauts, la gestion de la connaissance et la planification de projet. Ensuite, nous avons identifié les concepts des différentes méthodologies en relation avec les processus ciblés afin de déterminer les objectifs que nous allions établir pour valider l'efficacité des différentes solutions considérées. Nous avons alors cherché des moyens d'atteindre ces objectifs à l'aide de solutions déjà existantes sur le marché et dans la communauté des logiciels libres.

Le guide d'utilisation, placé en annexe de ce document, établi une procédure technique et pratique pour mettre en place une infrastructure capable d'atteindre les standards de base discutés dans les chapitres précédents pour chacun des processus. Il est simple et court (une trentaine de pages) afin d'inciter les membres des équipes à le lire et comprendre l'intérêt des procédures qui y sont incluses. Il est certain que chaque organisation doit se l'approprier et le façonner à son image, en fonction des subtilités de la situation qui lui est propre. Il est important de noter que l'installation d'une telle infrastructure ne garantit pas l'atteinte des objectifs sans exiger un effort considérable dans la formation aux outils mis en place et dans l'établissement de procédures claires quant à son utilisation. La période de transition nécessite aussi une surveillance rigoureuse, afin de ne pas laisser de mauvaises habitudes s'introduire au cœur des nouvelles méthodes des équipes de projet.

L'expérimentation de la conception d'un jeu vidéo avec les étudiants nous a démontré l'importance d'utiliser des méthodes et une infrastructure adéquate pour supporter le développement logiciel. Par contre, lorsque nous avons tenté d'incorporer certains outils pour essayer d'y remédier, nous avons négligé des éléments comme la formation et l'élaboration de procédures d'utilisation. Les changements furent alors pénibles et les processus n'ont pu être exploités à leur plein potentiel. À la suite de cette expérience, l'auteur a fondé une compagnie dans le domaine du développement logiciel et il utilise la méthodologie présentée dans ce travail avec la plupart des solutions de mise en œuvre proposées. En élaborant des procédures d'utilisation et en formant les personnes qui

utilisent l'infrastructure de développement, les problèmes rencontrés lors de l'expérimentation précédente ne se sont pas reproduits.

6.1 Ouverture sur l'avenir

Les recherches effectuées pour ce mémoire ne sont que le commencement d'un long cheminement vers l'amélioration des processus. Bien qu'elles ciblent le contexte du jeu vidéo, il n'en demeure pas moins qu'elles s'appliquent tout autant à de nouvelles entreprises ou à des organisations qui n'ont pas encore formalisés leurs processus de développement de logiciels. Elles offrent à une équipe la possibilité d'installer rapidement un environnement et des méthodes de travail qui augmenteront la communication entre les membres des équipes et leur efficacité tout en diminuant les risques d'erreurs et de pertes. Cette formalisation de la gestion des documents et des changements, de la gestion des défauts, de la connaissance et de la planification de projet permet à tous les membres d'une équipe de travailler de la même façon, sans pour autant nuire à leur productivité, ni à les forcer à effectuer des activités pour lesquelles ils n'ont aucun intérêt.

Comme il a été mentionné auparavant, plusieurs outils sont présentement en développement et risquent d'être plus efficaces que ceux présentés dans ce mémoire. Par contre, il est essentiel d'introduire les concepts derrière ces outils, afin que l'équipe en perçoive l'intérêt et s'investisse dans l'amélioration des procédures d'utilisation et des nouveaux outils plus performants. Une fois qu'une infrastructure semblable à celle présentée dans ce travail est mise en place, on ne peut qu'aller vers l'avant puisque les avantages qu'elle offre sont souvent intéressants pour tous les niveaux de l'organisation.

Nous croyons qu'il serait intéressant d'effectuer des recherches et de mener des expériences sur le temps nécessaire aux membres d'une équipe de développement logiciel pour l'accomplissement des tâches requises par la méthode présentée dans ce mémoire, à l'intérieur de l'architecture proposée versus le temps perdu par les membres d'une équipe sans méthode, ni architecture de développement particulière. Nous sommes toutefois conscients qu'il est difficile de mesurer ces différences de temps sur une période restreinte, puisque certains processus tels que la gestion de la connaissance, apportent des bénéfices à moyen et à long terme. Plusieurs variables comme la compétence des sujets, la chimie des groupes créés, l'expérience et l'aptitude des gestionnaires de projet peuvent aussi avoir un impact considérable sur les résultats. Un nombre important de groupes ayant à réaliser un projet identique serait nécessaire pour atténuer l'influence de ces variables sur les résultats.

Un sondage pourrait aussi être réalisé auprès des sujets afin de faire ressortir les irritants survenus tout au long du projet, en fonction des différents groupes d'utilisateurs tels que les programmeurs, artistes, gestionnaires et autres, autant pour les équipes ayant employé la méthode et l'infrastructure que celles qui n'y ont pas eu recours.

BIBLIOGRAPHIE

[Agile 2001] Cunningham W. and all, *Principles behind the Agile Manifesto*, 10/2001, http://agilemanifesto.org/.

[APQC 1999] American Productivity & Quality Center, *Creating a knowledge-sharing culture*, APQC, ISBN: 1928593178, 1999, 92p.

[Bach 1994] Bach J., *The immaturity of CMM*, American Programmer, pp 13-18, 1994.

[BI 2001] Business Interactif, *Extreme Programming Méthodes Agiles – Tour d'Horizon*, Business Interactif, 2001, 72p.

[Boehm-Turner 2003] Boehm B., Turner R., *Balancing Agility and Discipline: A Guide for the Perplexed,* Addison-Wesley Professional, ISBN: 0321186125, 2003, 304 p.

[Collins-Sussman 2003] Collins-Sussman B. and all, *Subversion The Definitive Guide*, 6685M, 09/2003, http://roederberg.dyndns.org/~arnold/file_share/doc/svn-book.html

[CompanynewsGroup 2006] Davin L., *Une gestion de contenu insuffisante rend les initiatives de mise en conformité risquées*, CompanyNews, 25/04/2006, http://www.companynewsgroup.com/communique.asp?co_id=108963

[Ferland 2006] Ferland J.F., *Partager les connaissances demande un effort*, Direction Informatique, 05/2006, http://www.directioninformatique.com/di/client/fr/DirectionsStrategiques/Nouvelles.asp?id=39391.

[Flanakin 2005] Flanakin M., Comparison: Web-based Tracker, Micheal Flanakin's Web Log, 08/2005, http://geekswithblogs.net/flanakin/articles/CompareWebTrackers.aspx.

[Germain-Robillard 2004] Germain E., Robillard P., Engineering-*based processes and agile methodologies for software development: a comparative case study*, The Journal of Systems and Software, 02/2005 v75 i1-2 p17(11).

[Gray 2001] Gray N., *In project estimating, hope for the best but plan for what is most likely to occur*, Pm network, 08/2001, pp 56-57.

[Hass 2002] Hass A.M.J., *Configuration Management Principles and Practice*, Addison Wesley Professional, ISBN: 0321117662, 2002, 432 p.

[IEEE 2004] IEEE Computer Society, *Guide to the Software Engineering - Body of Knowledge*, ISBN 0-7695-2330-7, 2004, 204 p.

[Irish 2005], Irish, *The game producer's handbook*, Course Technology, 1-59200-617-5, 2005, 350 p.

[ISO 9001:2000] International Organization for standardization, *ISO 9001:2000: Quality management systems – Requirements,* 12/2004.

[ISO 2005], ISO, ISO *9000 et ISO 14000 - en bref*, 10/2005, http://www.iso.org/iso/fr/iso9000-14000/understand/inbrief.html.

[Jalote 2002] Jalote P., *Gestion de projet informatique en pratique*, ISBN : 2-7440-1432-X, Campus Press, 2002.

[Larman 2002], C. Larman, *UML et les Design Patterns*, ISBN: 2744070904, Campus Press, 02/2002.

[Larman 2003], C. Larman, *Agile and Iterative Development: A Manager's Guide*, Addison-Wesley Professional, ISBN: 0131111558, 08/2003, 368 p.

[Llopis 2004] Noel Llopis, *By the Books: Solid Software Engineering for Games*, GDC 2004 roundtable, 03/2004, http://www.convexhull.com/.

[MSF 2002] Microsoft, *Livre Blanc: Modèle de processus de la structure Microsoft Solutions Framework (MSF)*, 2002.

[Iiazi 2003] Niazi, M., Wilson, D., *A Maturity model for the implementation of software process improvement*[Nonaka 1991], Ikujiro Nonaka, *The knowledge-creating company*, Harvard Business Review, 00178012, Nov/Dec 91, Vol. 69, Num. 6.

[OSA 2003] Open Source Armenia, *Version Control and Project Management*, 12/2003, http://opensourcearmenia.com/education/technical/version-control_html.

[Padberg 2006], Frank Padberg, *A study on optimal scheduling for software projects,* Software Process: Improvement and Practice, mars 2006, p 77-91.

[PMI 2004], *Guide du corpus des connaissances en management de projet.* American National Standard, ANSI/PMI 99-001-2004.

[Reardon 2005], Reardon K. K., *it's All Politics: Winning in a World Where Hard Work and Talent Aren't Enough,* Harvard Business Review, 00178012, Sep2005, Vol. 83, Num. 9.

[Riley 2003], T. B. Riley, *International tracking survey report '03': Number two: Knowledge management and technology*, CCEG (The Commonwealth Centre for e-Governance), mars 2003, 27p.

[Schwaber 2004] Ken Schwaber, *Agile Project Management with Scrum*, Microsoft Press, 02/2004, ISBN: 0-7356-1993-X.

[Secor 2003], Secor Conseil, *Analyse du positionnement de l'industrie du jeu interactif au Québec*, Alliance NumeriQC, janvier 2003.

[Seddon 1994] John Seddon, *In Pursuit of Quality: The Case Against ISO 9000*, Oak Tree Press, 05/1997, ISBN: 1860760422.

[SEI 1993] Software Engineering Institute, *Key Practices of the Capability Maturity Model, Version 1.1*, février 1993, 479p.

[Smith 2003] John Smith, *A Comparison of the IBM Rational Unified Process and eXtreme Programming*, IBM, 2003.

[Sohn 2005], J.H. Derick Sohn, *Facilitating "knowledge clashes" at Samsung*, KM Review, 1369-7633, Mai/Juin 2005, Vol. 8 Num. 2, p6-7, 2p.

[Standish G. 1999], Standish Group International inc., *Chaos: Recipe for success,* http://www.standishgroup.com/sample_research/PDFpages/chaos1999.pdf.

[Standish G. 2001], Standish Group International, *Extreme Chaos,* http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf.

[Stellman-Greene 2005], Stellman A., Greene J., *Applied Software Project Management*, O'Reilly, 11/2005, ISBN: 0-596-00948-8.

[Stephenson 2004] Stephenson M., Kampman J., *Livre Blanc: Développer les meilleures pratiques dans un environnement informatique complexe*, Computer Associates, 2004.

[Theodore 2005], Theodore S., *How the other half lives*, Game Developer Magazine, Jan 2005 p. 49 a 51.

[Tobin 2005] Tobin T., *Ten Principles for Knowledge Management Success*, Government Computer news, 03/2005. 7p.

[Wells 1999] Wells D., Never Add Functionality Early, 1999, http://www.extremeprogramming.org/rules/early.html

[West 2003] David West, Planning a project with the IBM Rational Unified Process, IBM, 2003.

ANNEXE 1 : GUIDE D'UTILISATION

Table des matières

1 - Le but d'une procédure

Le choix d'un outil est une chose, mais l'utilisation que l'on en fait en est une autre. La mise en œuvre d'une infrastructure de développement, telle que présentée dans ce travail, apporte des avantages considérables tout au long de la durée de vie des projets. Par contre, son implantation au sein des différentes équipes n'est pas sans risque. Effectivement, le changement parfois significatif des méthodes de travail peut être laborieux pour plusieurs. Le succès de cette délicate intervention réside principalement dans les moyens employés pour motiver les membres des équipes à s'investir dans ces changements, de les former convenablement aux différents systèmes ainsi qu'aux processus instaurés et de leur fournir les outils nécessaires pour comprendre le fonctionnement de l'infrastructure, mais aussi ce qu'ils peuvent en retirer.

Ce guide est donc essentiel dans l'intégration et l'efficacité de l'infrastructure de développement, puisqu'il établit une méthode pour commencer du bon pied. Tous les outils suggérés dans ce mémoire se retrouvent expliqués sommairement dans ce guide d'utilisation avec une petite procédure pour les utiliser de façon efficace. Le guide d'utilisation doit être suffisamment bref et précis afin d'inciter les membres de l'équipe à le lire et à l'employer convenablement. Il contient uniquement les aspects techniques et pratiques de l'utilisation des outils nécessaires à la mise en œuvre des concepts plus abstraits présentés dans ce mémoire et à l'atteinte de leurs objectifs. Ce guide pourrait être séparé en deux parties distinctes, soit l'une pour les utilisateurs et l'une pour les personnes ayant à configurer l'infrastructure, afin de séparer les aspects techniques de ceux uniquement nécessaires à l'utilisation. Puisque ce guide est un élément de départ à l'infrastructure, il fusionne les deux visions et doit ensuite être adapté à l'organisation selon ses réalités. Il est important de noter que le guide d'utilisation doit être adapté à l'organisation et ne renferme que les bases de l'infrastructure. Il est donc nécessaire de l'enrichir en fonction des différents outils et des spécifications de l'entreprise pour le rendre le plus concret possible.

2 - Gestion des documents et des modifications

La solution suggérée dans ce mémoire pour ce processus est le système de contrôle de version *Subversion*. Cet outil se divise en deux parties distinctes, soit la partie serveur, qui héberge les documents, contrôle les versions et les distribue, ainsi que la partie client, devant souvent être installée sur les postes des différents membres; certaines distributions de Linux intègrent déjà *Subversion* et permettent l'utilisation via un mode de commande.

L'installation du serveur nécessite trois étapes. Premièrement, installer *Subversion* sur la machine en fonction du système d'exploitation; les différentes versions sont téléchargeables à partir du site officiel de *Subversion*[47]. Une fois que l'application du côté serveur fonctionne et est accessible à partir du réseau interne de l'organisation, les deux autres étapes seront nécessaires pour le démarrage de tous les nouveaux projets.

2.1 - Nouveau projet

Après l'acceptation d'un nouveau projet, la première étape devrait être la création du dossier racine du projet sur le serveur et la création ou la réutilisation des comptes utilisateurs, avec leurs droits d'accès sur les différents répertoires qui évolueront tout au long du projet. Ainsi, le gestionnaire du projet pourra déjà déposer ses artéfacts, entre autres les documents de haut niveau, les échéanciers, l'analyse fonctionnelle, sur le serveur et les rendre disponibles aux gens concernés.

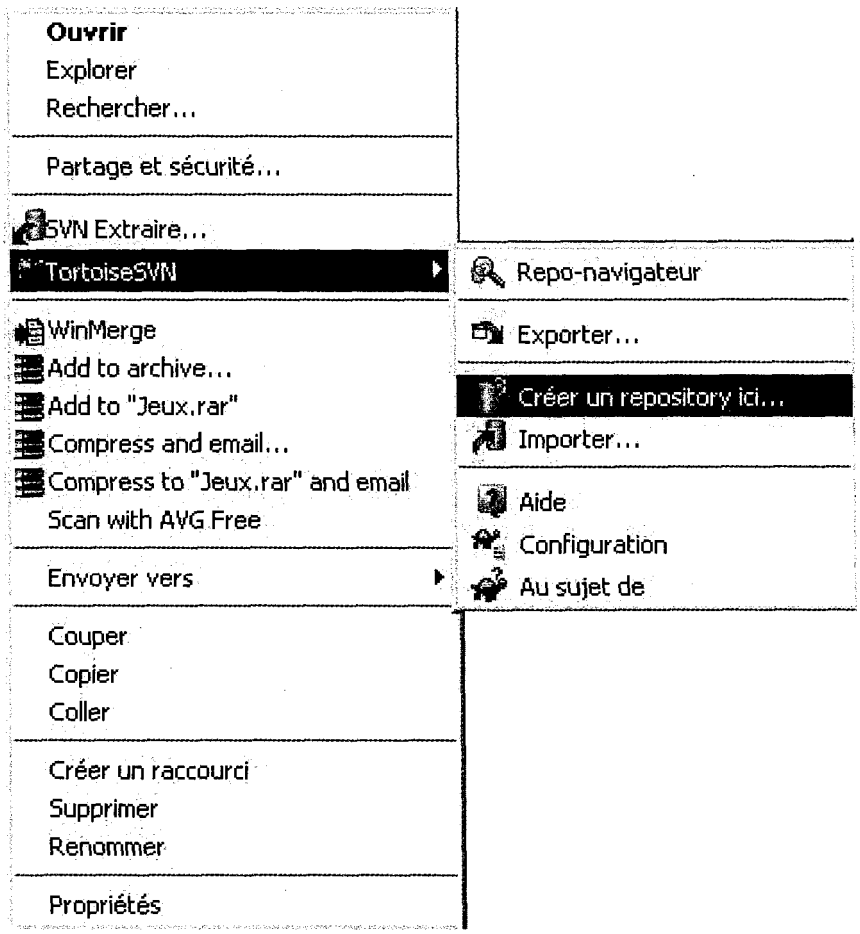
Tout d'abord, il faut créer le répertoire racine et lui associer un type de base de données pour conserver les versions. La figure A-1 démontre cette étape en considérant que *TortoiseSVN* est installé sur le serveur; sinon, il est aussi possible de le faire en mode de commandes, plusieurs tutoriaux traitent de ce sujet et sont disponibles sur Internet. Subversion offre deux types de base de données, Berkeley DB et FSFS. Bien que chacun ait ses avantages, nous proposons FSFS, qui a été développé spécialement pour Subversion,

---

47 Site officiel de *Subversion* : http://subversion.tigris.org/

quelques années après Berkeley DB et nous semble mieux adapté au contexte du développement de jeux vidéo, parce qu'il est plus rapide dans les publications imposantes en taille, moins sensible aux interruptions, et capable d'une sauvegarde de fichiers indépendante de la plate-forme.

Figure A-1 : Création du répertoire racine à partir de Windows



2.2 - Utilisateur et groupes

Dans le cas de l'installation de l'infrastructure, lors de la première utilisation de *Subversion*, il est certain que le gestionnaire, ou toute personne désignée à cet effet, aura à inscrire les utilisateurs et les groupes d'utilisateurs. Une fois cette opération terminée, il

pourra conserver ce fichier et l'associer aux autres projets. Pour *Subversion*, les comptes d'utilisateurs et les mots de passe sont créés avec l'outil « htpasswd » fourni par Apache. On utilise alors ces comptes pour former les groupes qui seront utiles pour accélérer l'association des droits d'accès sur les répertoires (voir figure A-2).

Figure A-2 : Gestion des groupes et des permissions

```
# Ce fichier défnit les groupes d'utilisateurs et les dossiers du projet LyB
# définition des groupes

[groups]
programmeurs: Fred, Alex, Sam
modélisateurs: Maxime, Mathieu, Sam
equipe: Fred, Alex, Sam, Maxime, Mathieu

# définition des droits d'accès sur les répertoires (/ = racine)

 [/]
* = r

[LyB : /repPourTous]
@equipe = r
Fred = rw

[LyB : /repCodeSource]
@programmeurs = rw
@modelisateurs =
Fred = r

[LyB : /modele3D]
@modelisateurs = rw
@programmeurs = r
```

Ici, dans l'exemple de la figure A-2, tout le monde a accès au répertoire racine du projet *LyB* en lecture, l'équipe peut lire le répertoire « *repPourTous* », mais seul *Fred* peut l'éditer. Tous les programmeurs peuvent écrire et lire le code source « *repCodeSource* », à l'exception de *Fred* qui, même s'il fait partie du groupe, ne peut que lire les fichiers du répertoire. Les *modélisateurs* n'ont pas accès au répertoire, à l'exception de *Sam*, qui fait partie du groupe *programmeurs*. Le groupe *modélisateurs* peut écrire dans le répertoire « *modele3D* », et *Sam* aussi, même s'il fait également partie du groupe *programmeurs*. Il est important de noter que puisque le système peut être utilisé par l'entremise de plusieurs

plates-formes technologiques différentes, il est préférable de ne pas utiliser d'accents, d'espaces ou de caractères spécifiques à une langue pour l'appellation des utilisateurs, des groupes et des répertoires.

2.3 - Connexion

Une fois la création des comptes utilisateur et des permissions terminée, les membres de l'équipe peuvent se connecter sur le répertoire du projet, ou sur plusieurs projets, et avoir accès aux fichiers rendus disponibles. À partir de là, ils seront en mesure de mettre à jour leurs documents sur le serveur, d'être informés lorsque leurs versions ne sont plus à jour et de suivre les changements qui leur seront apportés tout au long du projet.

Pour créer la connexion au serveur *Subversion*, il faut tout d'abord installer un logiciel client, à moins d'utiliser une version de *Linux* qui l'intègre. L'outil utilisé dans ce guide est *TortoiseSVN* pour *Windows*. Ces outils permettent de rendre l'utilisation du SCV presque invisible à l'utilisateur en se fondant dans le système de gestion des fichiers du système d'exploitation. Lorsqu'il est installé sur le poste, il ne reste qu'à importer le dossier et ses sous-dossiers. Pour y arriver, il suffit de cliquer avec le bouton droit de la souris sur le dossier qui deviendra le dossier contenant les fichiers du projet et ensuite cliquer sur « importer », que l'on peut voir dans la figure A-1. Ensuite, il faut entrer le *URL* du répertoire source du projet. Le tableau A-1, pris à partir du site officiel de *TortoiseSVN*, démontre les schémas des adresses URL qui doivent être utilisés pour accéder à un répertoire en fonction des protocoles réseau désirés.

Tableau A-1 : URL d'accès aux référentiels [48]

Schéma	Méthode d'accès
`file://`	Accès direct au référentiel sur disque local ou réseau.
`http://`	Accès via le protocole *WebDAV* à un serveur *Apache* avec *Subversion*.
`https://`	Même chose que `http://`, mais avec cryptage *SSL*.
`svn://`	Accès *TCP/IP* non authentifié via un protocole personnalisé à un serveur

[48] Référence du tableau A-1 : http://tortoisesvn.net/docs/nightly/TortoiseSVN_fr/tsvn-basics-svn.html

Schéma	Méthode d'accès
	svnserve.
svn+ssh://	Accès *TCP/IP* authentifié, crypté via un protocole personnalisé à un serveur *svnserve*

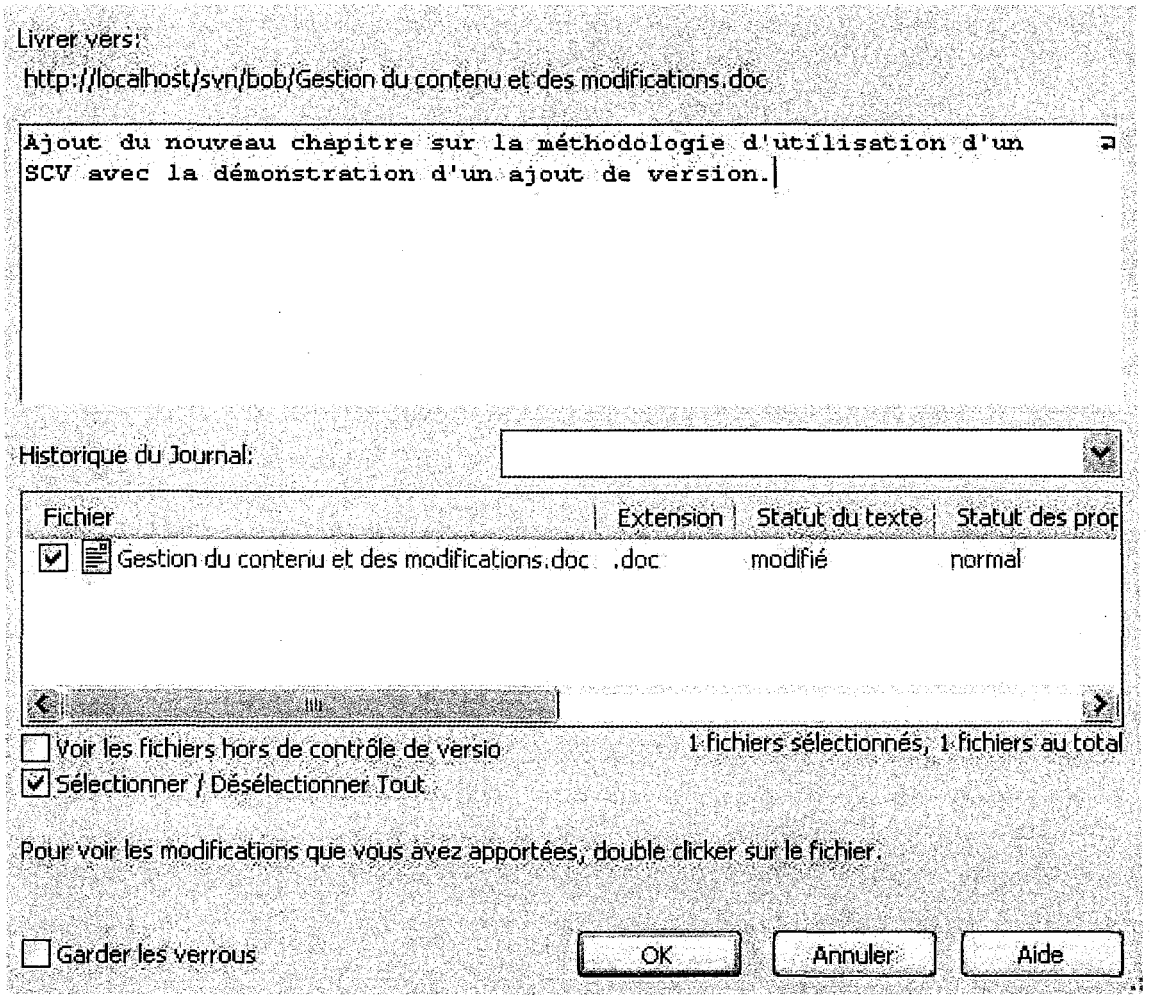
2.4 - Mise à jour

Lors de la formation donnée aux utilisateurs, il est bon de rappeler de ne pas livrer des versions à tous les petits changements que l'on apporte à un fichier lorsque l'on sait qu'il nous reste des modifications à faire. Dans le cas contraire, le serveur devient pollué par une grande quantité de versions quasi identiques. La livraison devrait plutôt se faire lorsque le travail sur le fichier est terminé, ou qu'une partie, par exemple une fonction, est complétée et utilisable.

Les mises à jour bien documentées (voir figure A-3) deviennent significatives pour ceux qui les importent. À partir de l'implantation du SCV, le gestionnaire devra rapidement avertir les membres qui ne documentent pas leur version, afin que l'ajout de commentaire devienne rapidement un réflexe et non une tâche. Dans le cas d'une mise à jour permettant de corriger un défaut, la personne publiant la nouvelle version devrait intégrer la référence du défaut, comme le numéro de billet distribué par *Mantis* ou l'adresse *URL* offrant la possibilité d'accéder directement au défaut dans le système. Ainsi, cela facilite la relation entre la découverte du problème et sa correction, tout en permettant de revenir rapidement à l'ancienne version si les modifications ont créé des effets de bord non désirés.

L'utilisation d'un SCV local comme ceux intégrés dans les outils de développement, par exemple *Visual SourceSafe* à l'intérieur de *Visual Studio.NET*, peut permettre la sauvegarde des modifications temporaires, pour ensuite livrer uniquement sur le serveur les fichiers suffisamment avancés pour être utiles aux autres.

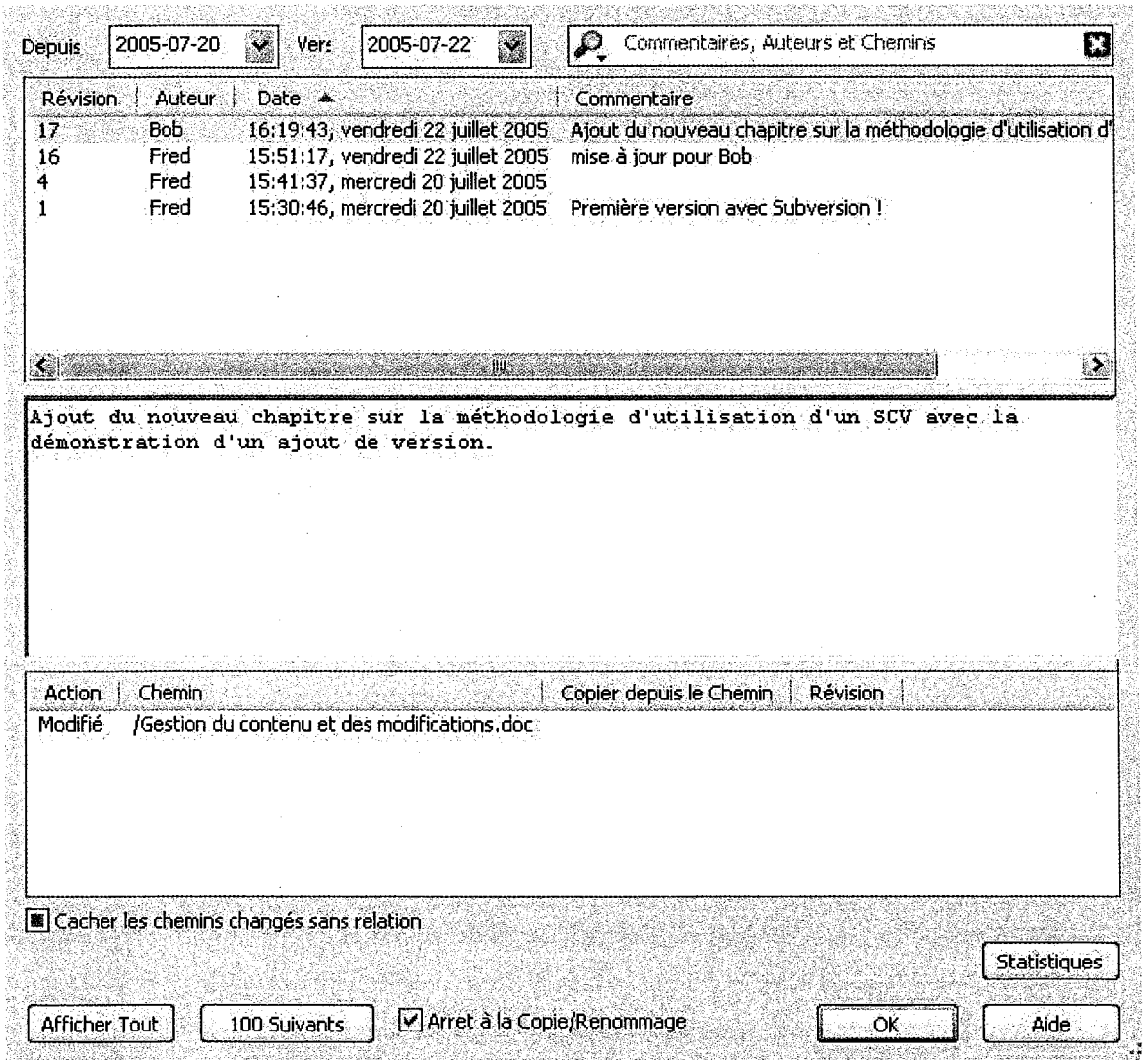
Figure A-3 : Documentation des livraisons



Après l'implantation de ces mécanismes, le gestionnaire de projet devra prendre l'habitude d'examiner l'historique des changements effectués sur le répertoire racine. Bien qu'anodine, lorsque la mise à jour est bien accomplie, comme nous l'avons dis précédemment, cette opération lui permettra de connaître l'avancement global et le respect des objectifs du projet en tout temps, même à distance. La figure A-4 démontre les informations qu'un gestionnaire peut obtenir simplement en regardant l'historique des

changements d'un fichier ou d'un répertoire, soit les dates de modifications, les personnes impliquées, les fichiers ajoutés, supprimés ou modifiés.

Figure A-4 : Historique des changements

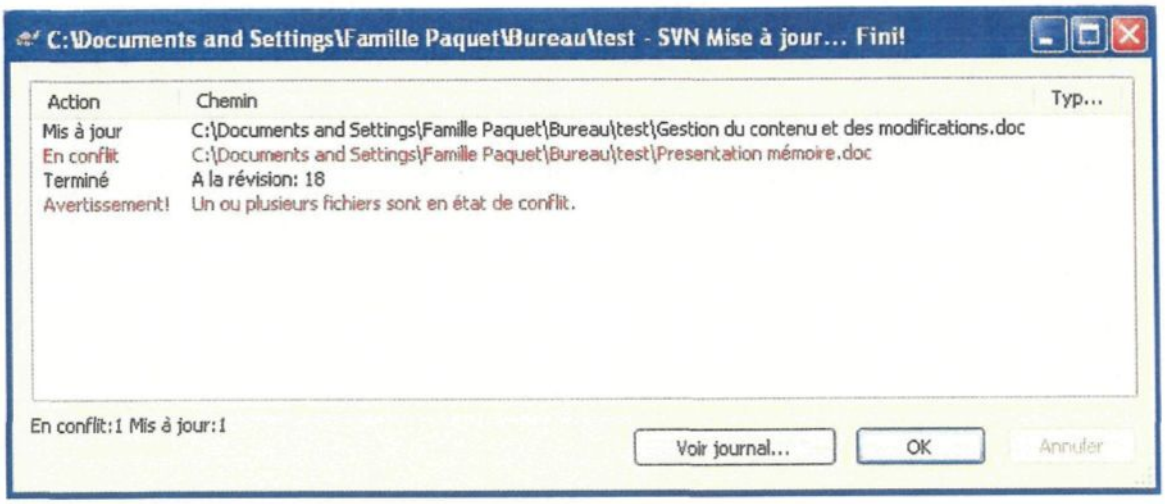


2.5 - Fusion entre deux fichiers conflictuels

Tel que mentionné auparavant, il est possible que deux personnes travaillent en même temps sur le même fichier. Lorsque le dernier des deux voudra livrer sa nouvelle version, le SCV lui indiquera que le fichier est en conflit (figure A-5). Une fois que le SCV

a détecté le conflit, il reste alors à fusionner les deux fichiers pour en faire une version unique contenant toutes les modifications apportées par les deux utilisateurs.

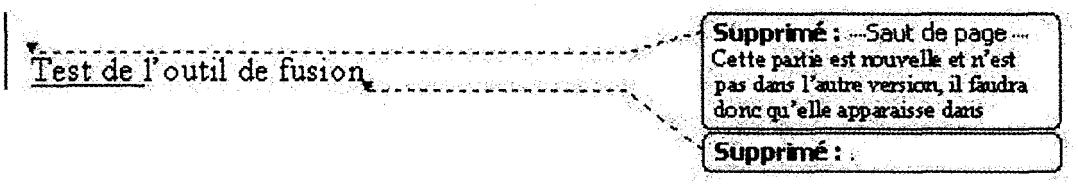
Figure A-5 : Détection d'un conflit à l'aide de Subversion (TortoiseSVN)



Certains SCV intègrent les outils de fusion directement dans leur système. La plupart des SCV libres offrent, quant à eux, la possibilité d'intégrer d'autres logiciels qui feront la fusion à leur place. Il existe un bon nombre de ces outils de fusion « merge tools » gratuitement offerts sur Internet.

Il est préférable de choisir un outil de fusion intégrant un grand nombre de types de fichiers et qui peut être utilisé directement par le SCV. La plupart des SCV offrent la possibilité de configurer un outil de fusion externe. Pour ce qui est de celui que nous avons choisi, *TortoiseSVN*, il possède son propre outil, *Tortoise Merge*, pour lequel on peut associer un bon nombre de modules d'extensions (*plug-in*) tels que *Microsoft Visual Studio*, *Rational Rose*, *Visual Age*, et d'autres. Ces options permettent donc de voir les différences entre deux fichiers directement dans notre environnement de développement (voir l'exemple de la figure A-6). Le choix de l'outil de fusion devrait donc être en fonction de sa capacité à cohabiter avec l'environnement et les spécifications du projet.

Figure A-6 : Différence entre deux versions directement dans Word



2.6 - Améliorations

Lorsque *Subversion* est choisi pour être le système de contrôle des versions d'une organisation, il peut être intéressant de créer un outil pour faciliter la gestion des utilisateurs et des droits d'accès au lieu de dupliquer le même fichier de configuration. Une application maison permettant d'éditer les fichiers de configuration en utilisant une base de données peut être conçue pour centraliser les informations concernant les utilisateurs, dont le nom d'utilisateur et le mot de passe. Cette amélioration permet d'accélérer le processus de création de projet dans le SCV et d'uniformiser les comptes et les mots de passe pour l'ensemble des projets.

Afin d'augmenter l'efficacité et la synergie entre les processus, une solution efficiente devrait permettre d'alerter automatiquement l'équipe de test lorsqu'il y publication d'une modification ou ajout d'une nouvelle branche dans le SCV. Cet ajout offre aux testeurs l'opportunité d'effectuer le plus tôt possible le contrôle qualité sur les changements qui ont été apportés. Un exemple d'automatisation d'ouverture de billet sera présenté dans la gestion des défauts au point 3.4 « Création de billet via le SCV », à la figure A-11.

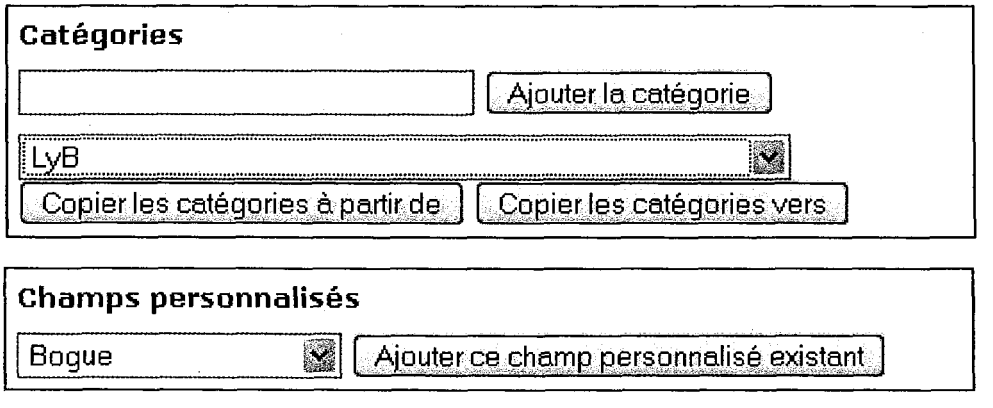
3 - Gestion des défauts

L'outil qui a été retenu dans ce mémoire pour soutenir la gestion des défauts est *Mantis*. L'installation de l'application ne nécessite qu'un serveur Internet avec *PHP* avec une instance de la base de données *MySql* et se fait très rapidement. Certaines décisions

doivent être prises lors de la configuration de *Mantis* en fonction des réalités de l'organisation. Une structure doit être déterminée pour représenter les étapes de production d'un projet et les différentes équipes qui y collaboreront. Dans un projet de conception de jeu vidéo, on peut créer plusieurs sous-projets qui correspondent aux différentes étapes de production, ou aux différents livrables. Cette distinction de projet et de sous-projets permet d'avoir une vue sur l'ensemble du projet et une autre sur la phase qui est présentement en développement.

De plus, *Mantis* permet de créer des catégories et des champs personnalisés. Dans notre cas, les catégories seront utilisées pour classer les défauts parmi les différents départements de l'entreprise. Ainsi, les programmeurs d'intelligence artificielle ou les concepteurs de personnages 3D, pourront voir uniquement les défauts qui les concernent s'ils le désirent et alléger l'interface, pour ne pas, par exemple, avoir à chercher dans les 300 défauts présents pour tous les départements. Les champs personnalisés peuvent servir à classifier les types de défauts comme les bogues, les améliorations demandées, les nouveautés à ajouter, etc. Les catégories et les champs personnalisés méritent d'être bien pensés dès l'introduction de *Mantis* comme outil de gestion des défauts puisqu'ils deviendront des repères et que leur configuration peut être copiée d'un projet à l'autre d'un simple clic comme le montre la figure A-7.

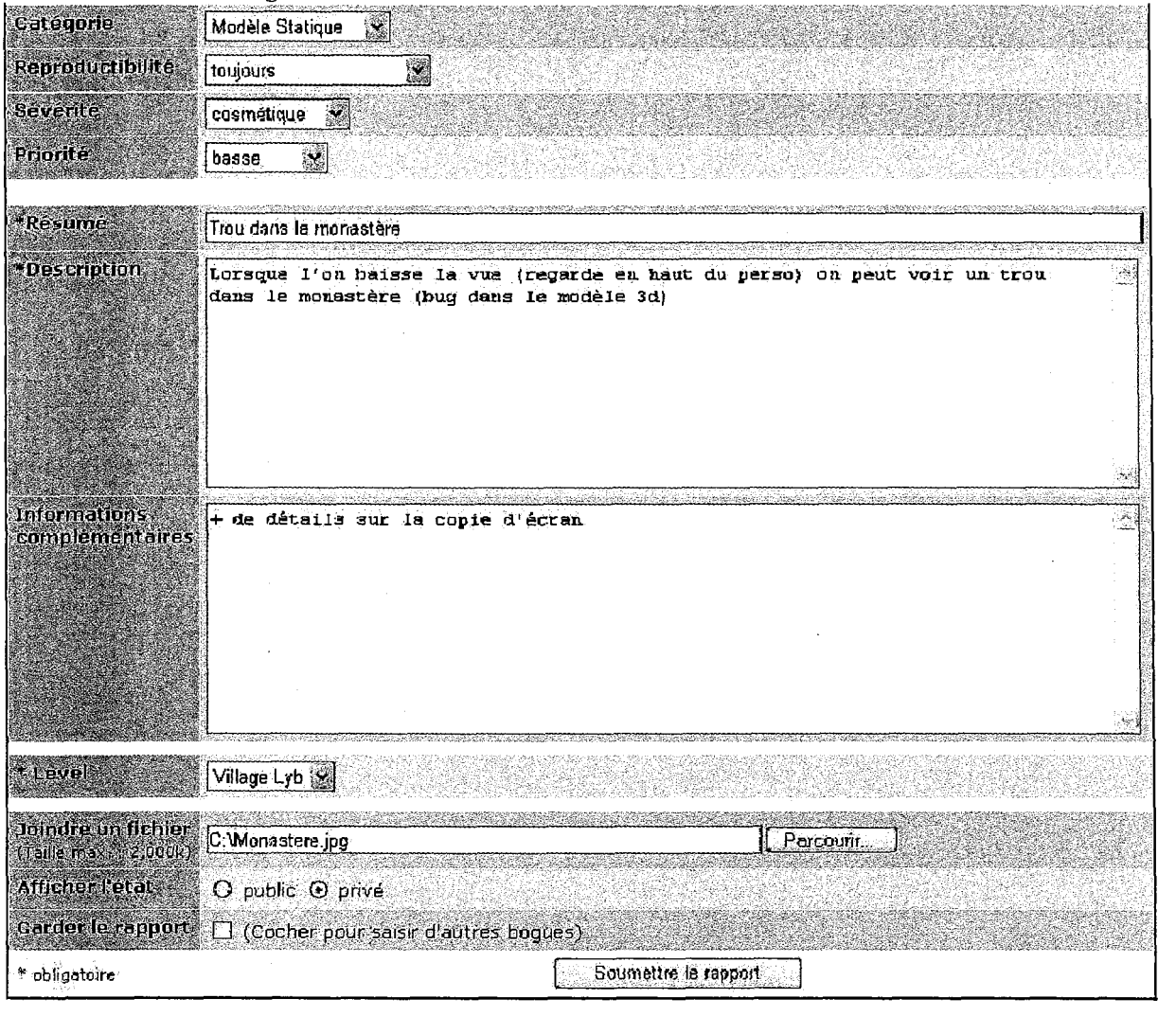
Figure A-7 : Configuration des catégories et des champs personnalisés



3.1 - Ouverture de billet

La figure A-8 démontre l'ouverture d'un billet de défaut dans *Mantis* lors d'une détection interne ou externe. On peut remarquer les changements apportés pour adapter le système au besoin d'un projet, par exemple l'équipe de conception de jeux vidéo.

Figure A-8 : Ouverture d'un billet de défaut dans Mantis



3.2 - Prise de conscience des billets

Les personnes assignées aux billets doivent rapidement prendre connaissance du défaut et ouvrir le billet. Cette opération lui permet donc de valider qu'elle est la bonne ressource, que le billet est bien classé et dans la bonne catégorie, et de mesurer la sévérité/priorité du défaut. Lorsqu'un membre prend conscience d'un billet et qu'il constate qu'il est mal classé, il doit donc retourner le billet au répartiteur en indiquant une note avec sa recommandation, dans le but qu'il apprenne et comprenne ses erreurs.

La figure A-9 est la légende des couleurs par défaut dans *Mantis*, qui permet au répartiteur et aux membres de l'équipe de savoir s'ils ont de nouveaux billets assignés; la figure A-10 est un exemple de billets assignés à une personne. On peut constater qu'elle n'a pas pris conscience encore du billet # 8 qui lui a été affecté.

Figure A-9 : Légende des couleurs par défaut dans Mantis

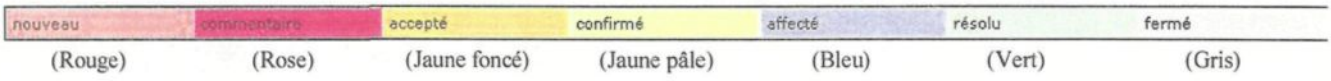
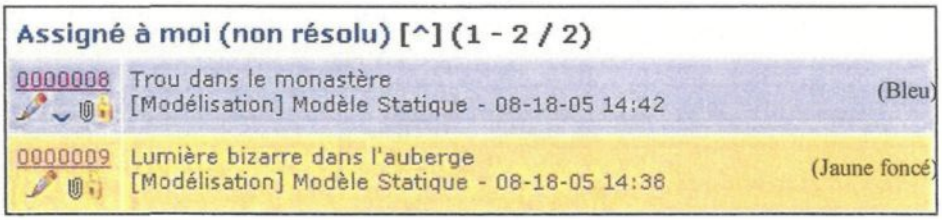


Figure A-10 : Exemple de billets assignés à un membre d'une équipe



De plus, lorsque le répartiteur constate qu'un certain temps – à déterminer selon le type de produit ou de service et selon l'organisation – s'est écoulé suite à l'assignation sans que le billet ne soit ouvert, c'est-à-dire sans que la personne n'en ait pris connaissance, il devra par conséquent réassigner le billet à un autre membre de l'équipe ou l'escalader, autrement dit faire suivre à son supérieur hiérarchique, ou à une personne particulièrement compétente pour ce type de défaut. Cette action permettra donc au gestionnaire de projet de

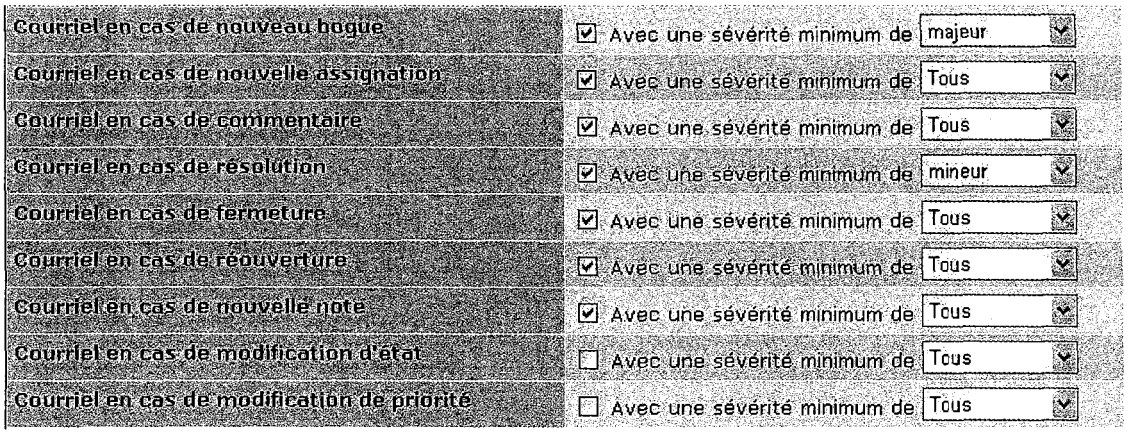
mesurer le temps écoulé par les membres de l'équipe avant qu'ils ne prennent connaissance des billets (temps d'ouverture – temps d'assignation) et les mauvaises assignations de billets.

3.3 - Portail et gestion des alertes

Puisqu'on ne peut s'attendre à ce que les membres de l'équipe de développement, mis à part les membres de l'équipe de test, attendent constamment l'arriver de billets, il est possible de mettre en place des mécanismes pour les avertir qu'un billet vient de leur être assigné ou qu'un de leurs billets a été modifié. La relève à intervalles réguliers des courriels est un réflexe déjà bien en place dans bien des organisations. Le développement d'une telle habitude ne demande donc pas vraiment de conditionnement supplémentaire.

La figure A-11 démontre le genre d'alertes par courriel qu'il est possible de gérer dans *Mantis* pour chaque utilisateur. Il ne reste qu'à déterminer les événements qui déclencheront les alertes pour chacun des utilisateurs ou groupe d'utilisateurs.

Figure A-11 : Alerte par courriel possible dans *Mantis*



3.4 - Création de billet via le SCV

Lorsqu'une nouvelle version est publiée dans le SCV par un artiste ou un programmeur, il est possible, avec quelques lignes de code, de créer un billet

automatiquement, qui sera assigné aux membres de l'équipe de tests. Ces derniers pourront alors rapidement tester et évaluer l'impact des changements.

Un fichier *Perl* (\tools\hook-scripts\commit-email.pl.in) peut être appelé dans *Subversion* lorsqu'une version est livrée pour envoyer un courriel. À partir de ce fichier, il est donc possible d'ajouter des instructions *Perl* pour créer un nouveau billet dans *Mantis*, ou plutôt dans sa base de données *MySql*. La figure A-12 est un exemple de modification apportée à ce fichier pour atteindre cet objectif.

Figure A-12 : Exemple de code à ajouter dans *Subversion*

```
// Ceci est un exemple, une adaptation doit être faite afin de l'intégrer dans Mantis selon les
// Variables du serveur, de MySql et des configurations de Mantis (utilisateur, projet, etc.)

use DBI;

$dbh = DBI->connect("DBI:mysql:database=$db;host=$host",
                    $user, $password, {RaiseError => 1});

my $query = sprintf("INSERT INTO `mantis_bug_table` ( `project_id` , `reporter_id` , `handler_id` ,
`duplicate_id` , `priority` , `severity` , `reproducibility` , `status` , `resolution` , `projection` , `category`
, `date_submitted` , `last_updated` , `eta` , `bug_text_id` , `os` , `os_build` , `platform` , `version` ,
`fixed_in_version` , `build` , `profile_id` , `view_state` , `summary` , `sponsorship_total` , `sticky` )

VALUES ( '3', '5', '0', '0', '30', '10', '100', '10', '10', '10', 'Choisir catégorie', '2005-08-20 15:52:59', '2005-
08-20 15:52:59', '10', '10', '', '', '', '', '', '0', '10', 'Nouvelle version \n  Auteur: $author\n
Numero de version: $rev\n ', '0', '0') ", $number, $dbh->quote("name"));

$dbh->do($query);
```

Tous ces mécanismes sont mis en place dans le seul but de permettre l'assignation rapide et efficace des billets, afin qu'aucun défaut ne soit perdu et que les membres concernés soient rapidement mis au courant des défauts qui se rattachent à eux.

Cette procédure d'utilisation d'un outil de détection et de correction des défauts tel que *Mantis* permet ainsi d'atteindre en très grande partie les objectifs décrits précédemment, sans pour autant demander énormément de ressources financières et humaines.

4 - Gestion de la connaissance

Le processus de gestion de la connaissance est celui qui nécessite le plus d'ajustement en raison des différences entre l'infrastructure présentée dans ce travail et celle de l'organisation. Les trois outils qui seront présentés dans ce guide d'utilisation seront *MediaWiki*, un système wiki pour rédiger les documents devant être publiés directement sur l'Intranet et pouvant être modifiés à partir de là (comme la définition du contenu), *MyBB*, un forum électronique permettant de gérer les discussions entre les membres d'un projet en conservant un historique de ce qui a été énoncé, et *Webglimpse*, un moteur de recherche très simple permettant d'annexer les pages web des différents systèmes à l'intérieur du réseau de l'entreprise.

4.1 - Système wiki

L'outil de type wiki qui a été retenu dans ce travail est *MediaWiki*, un système à l'origine conçu pour soutenir la populaire encyclopédie libre *Wikipédia*. L'installation de base de *MediaWiki* est très simple; par contre, puisque cet outil est destiné principalement à la rédaction d'une encyclopédie universelle, il est donc fait pour le grand public. Cet aspect est intéressant puisque *MediaWiki* est construit pour être suffisamment simple à l'utilisation, cependant, la gestion des droits d'utilisateurs n'est pas une option par défaut, il faut alors la configurer.

4.1.1 - Installation et configuration

Pour l'installation de *MediaWiki*, il suffit de créer une base de données *MySql* sur le serveur (*Apache* ou *IIS*) et d'y ajouter les fichiers téléchargés à partir d'Internet[49]. Une fois cette étape accomplie, il suffit d'appeler le fichier de configuration (*config/*) et de suivre les étapes apparaissant dans la fenêtre du navigateur. Comme il a été mentionné auparavant, *MediaWiki* nécessite certains ajustements pour permettre la gestion des droits d'utilisation. Dans le cas contraire, l'unique configuration possible est la permission de modification des

[49] Site pour télécharger *MediaWiki* : http://www.mediawiki.org/wiki/MediaWiki

pages, ce qui est largement insuffisant dans un contexte aussi compétitif que le domaine des jeux vidéo et où même le nom des projets doit rester secret.

Les configurations nécessaires à l'implantation d'une sécurité adéquate peuvent se trouver dans le manuel d'utilisation (*meta-wiki*[50]) de l'outil. La plupart des modifications à faire se trouvent à l'intérieur du fichier de configuration de MediaWiki (LocalSettings.php) une fois l'installation faite.

Voici les principales lignes à ajouter dans ce fichier :

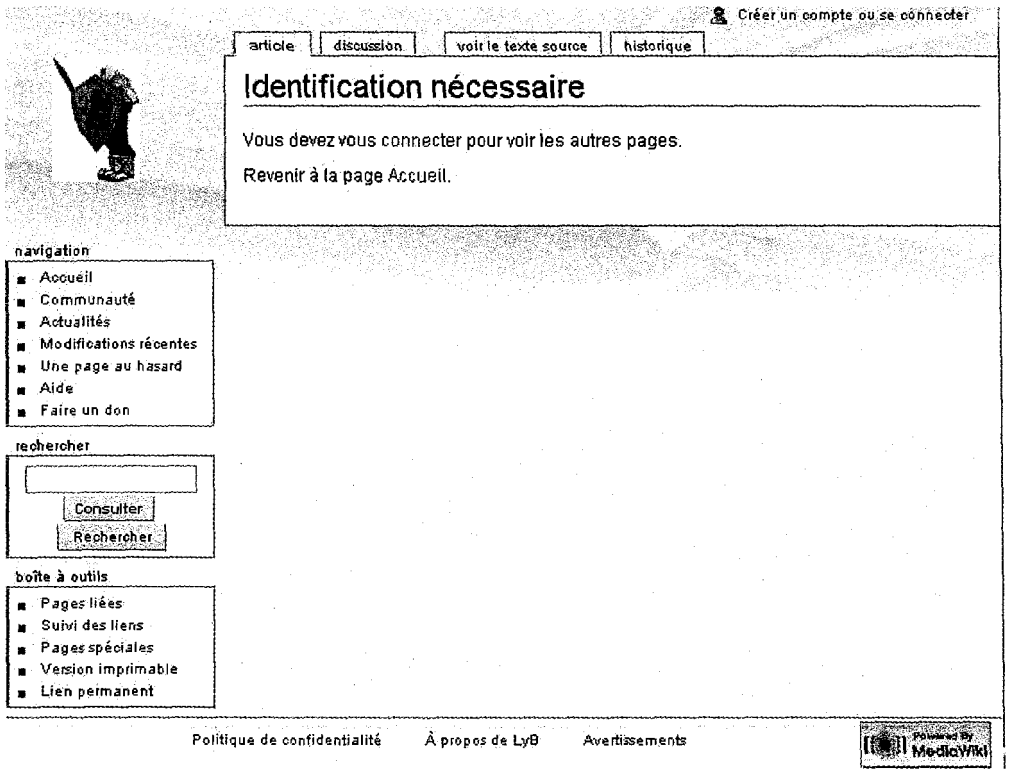
```
// Désactive la création d'un nouveau compte par un utilisateur anonyme.
$wgGroupPermissions['*']['createaccount'] = false;

// Pour rendre l'édition de page accessible uniquement aux utilisateurs connectés.
$wgGroupPermissions['*']['edit'] = false;

// Enlève l'adresse IP dans le menu du haut et l'option « discussion avec IP».
$wgShowIPinHeader = false;

// Enlève les permissions de lecture au groupe « anonyme ».
$wgWhitelistRead = array("Main Page", "Special:Userlogin", "-", "MediaWiki:Monobook.css");
$wgGroupPermissions['*']['read'] = false;
```

La figure A-13 démontre un exemple de l'interface de *MediaWiki* lorsqu'un utilisateur n'est pas connecté et qu'il accède à la page d'accueil, ou à n'importe quelle autre page du système, une fois que l'outil est correctement configuré.

---

50 Manuel d'utilisation : http://meta.wikimedia.org/wiki/Help:User_rights

Figure A-13 : Configuration sécuritaire de *MediaWiki*



On peut remarquer que l'option de création des comptes semble toujours visible dans le haut à droite de l'écran, mais la création est toutefois désactivée, donc un utilisateur anonyme peut uniquement se connecter et non créer un nouveau compte.

4.1.2 - Gestion des droits d'utilisation

MediaWiki permet aussi la création de comptes individuels et de groupe pour faciliter la gestion des droits d'utilisation. La syntaxe pour l'attribution des permissions à un groupe est plutôt simple et se fait toujours de la même façon. Dans le fichier de configuration (*LocalSettings.php*), ajouter les différents droits avec le nom du groupe que vous désirez créer de la manière suivante :

```
$wgGroupPermissions['SuperUtilisateur']['createaccount'] = true;
```

Les personnes ayant les droits d'administrateurs peuvent ainsi associer les utilisateurs au groupe « SuperUtilisateur » qui a comme privilège, selon l'exemple ci-dessus, de créer des nouveaux comptes. L'ajout de permission aux groupes fonctionne toujours selon la même syntaxe (**$wgGroupPermissions['Groupe']['Privilège']** = true ou false;). Lorsque le groupe inséré n'existe pas, il est alors créé. Le groupe « * » désigne tous les utilisateurs y compris les utilisateurs anonymes.

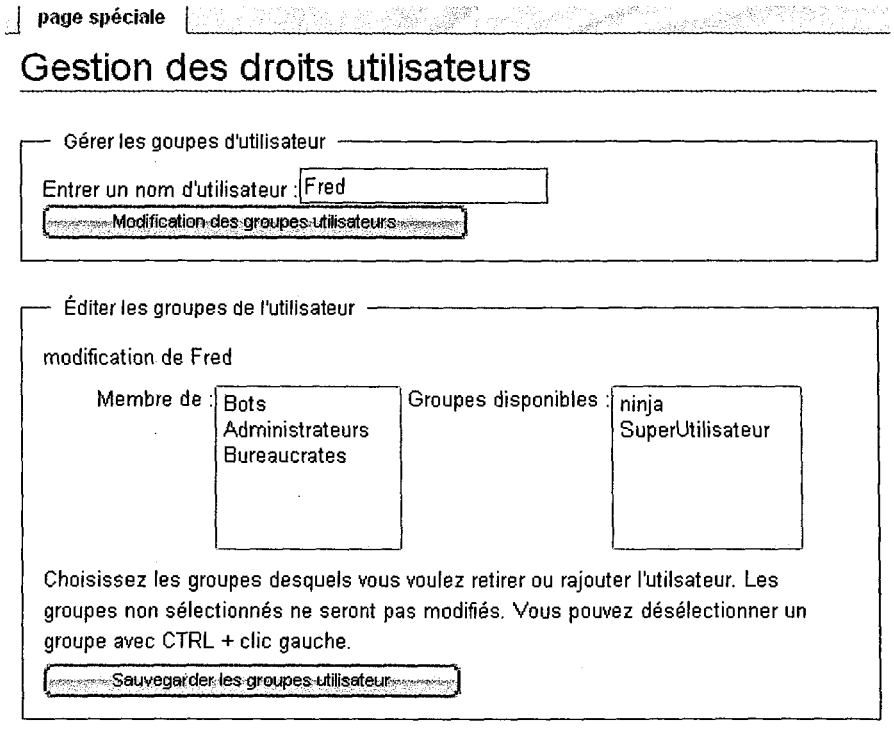
Voici les différents privilèges que l'on peut associer à un groupe, et qui doivent être écrits en anglais : *'createaccount', 'read', 'edit', 'delete', 'createpage', 'createtalk', 'move', 'upload', 'reupload', 'reupload-shared', 'minoredit', 'autoconfirmed', 'bot', 'block', 'deletedhistory', 'editinterface', 'import', 'importupload', 'patrol', 'protect', 'rollback', 'unwatchedpages', 'userrights'.*

La figure A-14 présente la gestion des droits des utilisateurs, qui se résume en fait à l'association d'un utilisateur à un ou plusieurs groupes. On peut remarquer dans la figure que le groupe « SuperUtilisateur » est maintenant disponible pour être associé à un utilisateur. Il est aussi possible de sécuriser une page afin d'empêcher les utilisateurs anonymes, les personnes n'ayant pas les droits d'administrateurs ou le privilège *'protect'* de la modifier. Sinon il reste l'éventualité de protéger les pages contre une ou des adresses IP spécifiques, ce qui n'est pas pratique dans une organisation où les adresses sont distribuées de façon dynamique et non de façon fixe.

4.1.3 - Nouveau projet

Par défaut, la page d'accueil de *MediaWiki* propose un lien vers le guide d'utilisation de l'outil. Nous conseillons de conserver ce lien sur cette page, afin qu'elle demeure facilement accessible aux utilisateurs voulant se renseigner sur l'outil et sa syntaxe, puisque la plupart des logiciels de type wiki ont leur propre syntaxe.

Figure A-14 : Gestion des droits aux utilisateurs dans *MediaWiki*



Lors du démarrage d'un nouveau projet, nous conseillons simplement de copier la table « user » et « user_groups » d'un ancien projet, ainsi que les configurations des utilisateurs dans le fichier *LocalSettings.php*. Une fois ces étapes terminées, il ne restera qu'à modifier les différences de privilèges entre les deux projets. Pour changer les options dans le menu de navigation, il suffit d'entrer « MediaWiki:Sidebar » dans la bar de recherche et ensuite modifier la page.

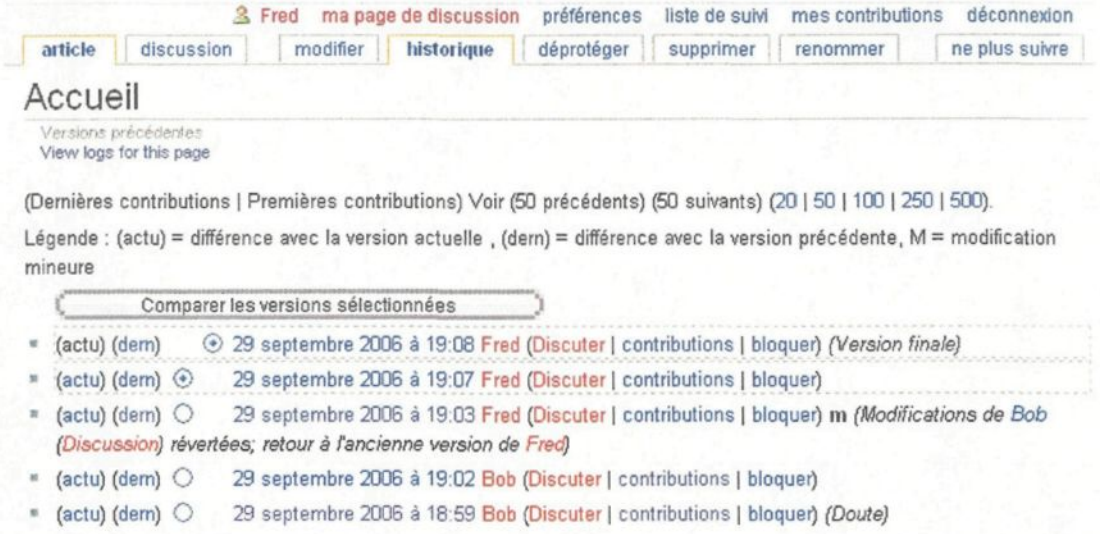
4.1.4 - Utilisation

Une fois l'installation et la configuration terminées, les utilisateurs peuvent commencer à publier leurs textes dans l'outil. Une légère marche à suivre et une petite formation peuvent être utiles pour faciliter l'acceptation de l'outil et les familiariser avec les concepts d'un système wiki tels que la création des pages, les modifications possibles par les autres, les discussions autour des pages, la surveillance des modifications.

D'ailleurs, pour ce dernier point, il vaut la peine de bien informer les utilisateurs sur l'utilité de « surveiller » les pages que l'on crée et bien configurer les préférences des utilisateurs afin que ces derniers reçoivent un courriel lors d'une modification ou d'une discussion, par exemple. La surveillance d'une page s'active ou se désactive simplement en cliquant sur le lien « suivre » ou « ne pas suivre » dans le haut de la page en question.

La figure A-15 est un exemple d'historique d'une page. À partir de cet historique, un utilisateur ayant les droits nécessaires peut comparer les versions entre elles, voir les modifications qu'une personne a apportées et revenir à une version précédente (*'rollback'*).

Figure A-15 : Configuration sécuritaire de *MediaWiki*



4.2 - Forum électronique

L'outil retenu pour permettre aux membres d'une équipe de discuter de différents sujets en relation avec un projet est *MyBB*. L'installation est très simple et se fait comme la plupart des logiciels libres sur le marché. On doit simplement télécharger les fichiers trouvés sur le site Web officiel sur le serveur, créer une base *MySql* et configurer le système à partir du fichier d'installation (*MyBB/install/*). Bien qu'il soit possible de traduire tout le système à partir des écrans de traduction offerts dans *MyBB*, des traductions dans plusieurs

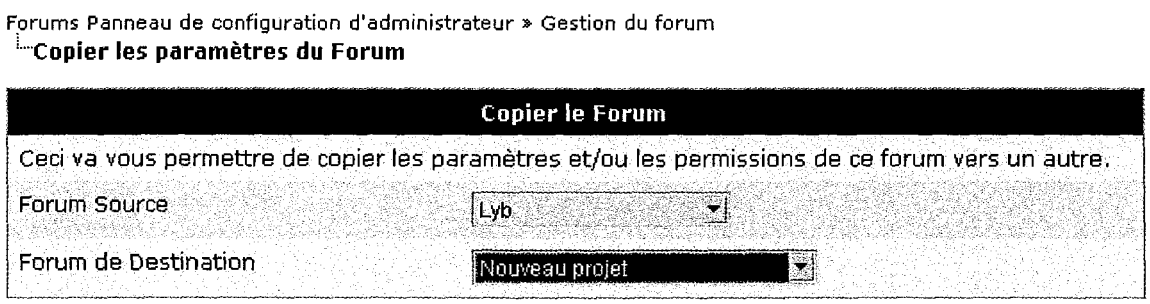
langues sont déjà disponibles sur Internet et pouvant être utilisées gratuitement, ou servir de base.

4.2.1 – Configurations nécessaires

La configuration de *MyBB* demande toutefois un certain temps à l'administrateur. La création des différents sujets de conversation ainsi que des comptes utilisateurs, qui sont les étapes les plus longues, peut être considérablement accélérée si certaines mesures sont prises lors de la première installation. Dans le cas où *MyBB* doit être réinstallé pour un nouveau projet, il suffit de copier les données des tables (« *mybb_users* », « *mybb_usergroups* », « *mybb_forum* », « *mybb_forumpermissions* ») et de mettre à jour les différents forums pour les adapter au nouveau projet, en supposant que « *mybb_* » est le préfixe des tables de la base de données.

Sinon, lorsqu'un nouveau projet implique simplement un nouveau sujet, une option qui facilite la gestion de la connaissance, puisque le même moteur cherche dans la même table, quoique cela puisse ralentir le système avec le temps, l'administrateur du forum doit reproduire les différents sujets pour le nouveau projet. Cette étape peut être accélérée en travaillant directement sur la table « *mybb_forums* », mais uniquement par une personne ayant de bonnes connaissances en base de données et connaissant bien le système. Pour les permissions d'utilisation aux différents sujets, elle peut les copier directement à partir des sujets déjà existants (voir figure A-16 qui démontre cette opération) pour ensuite apporter les changements nécessaires, ce qui est beaucoup plus rapide que de tout recommencer.

Figure A-16 : Copie des permissions à partir d'un autre forum dans *MyBB*



4.2.2 – Groupes utilisateurs

Les trois groupes *Administrateur*, *Modérateur* et *Enregistré* sont les principaux groupes utilisateurs qui devraient être utilisés au départ dans le forum. Puisque le forum électronique est l'outil employé pour débattre les opinions et les idées innovatrices des membres des différentes équipes, il est nécessaire d'enlever toutes les permissions au groupe anonyme (« *unregistered* »), afin que les informations ne puissent être volées par des personnes externes qui réussiraient à entrer dans le réseau.

Le ou les administrateurs sont ceux qui gèrent les différents forums, les configurations, les permissions et les utilisateurs. Les modérateurs, quant à eux, peuvent gérer les sujets, par exemple verrouiller, ouvrir, éditer, supprimer, diviser, copier et déplacer, et gérer également les réponses. Les personnes enregistrées sont en fait le reste des membres de l'organisation ou de l'équipe. Par défaut, les comptes du groupe *Enregistré* ont les droits nécessaires pour gérer leurs propres messages. Cependant, dans l'optique de conserver tous les enregistrements, tel que mentionné dans les normes ISO [ISO 9001:2000], nous conseillons de retirer les privilèges de ce groupe afin qu'il ne puisse supprimer ou altérer ce qu'il a publié; si les membres du groupe désirent rectifier une idée, ils peuvent toutefois se citer dans un nouveau message.

La figure A-17 présente un exemple des permissions pouvant être accordées au groupe : *Enregistré*, pour un forum spécifique ou sur l'ensemble des forums, selon la volonté de l'administrateur.

Figure A-17 : Permissions pour le groupe : *Enregistré*, sur le forum Lyb

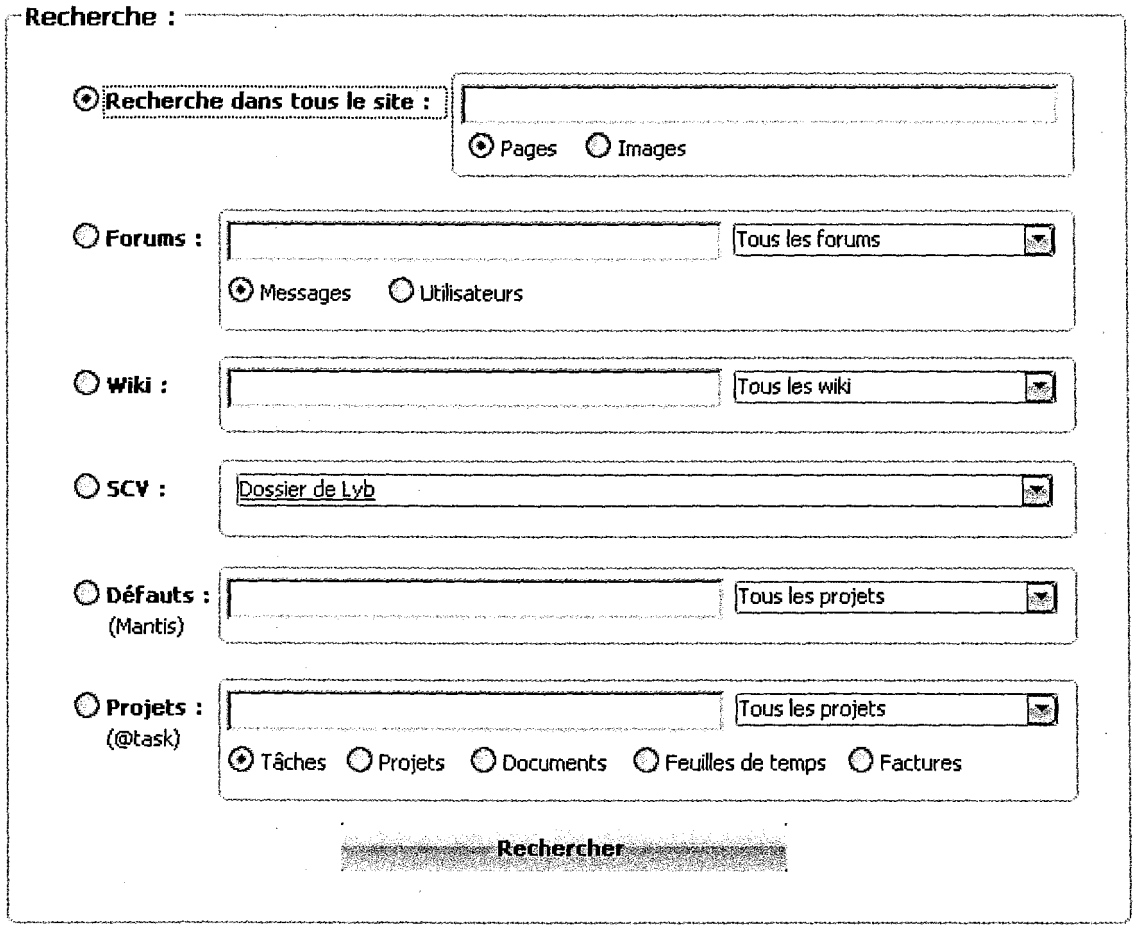
Forums Panneau de configuration d'administrateur
Modifier les permissions

Modifier les Permissions de forum pour Registered dans Lyb	
○ Utiliser les paramètres du groupe / Hériter depuis les forums parents (supprimera les paramètres personnalisés)	
◉ Utiliser les paramètres personnalisés ci-dessous	
Permissions : Voir	
Peut voir le forum	◉ Oui ○ Non
Peut voir les discussions du forum	◉ Oui ○ Non
Peut télécharger les pièces jointes	◉ Oui ○ Non
Permissions : Poster / Noter	
Peut poster des sujets	◉ Oui ○ Non
Peut poster des réponses	◉ Oui ○ Non
Peut poster des pièces jointes	◉ Oui ○ Non
Peut noter les sujets	◉ Oui ○ Non
Permissions : Edition	
Peut éditer ses propres messages	○ Oui ◉ Non
Peut supprimer ses propres messages	○ Oui ◉ Non
Peut supprimer ses propres sujets	○ Oui ◉ Non
Peut modifier ses propres pièces jointes	○ Oui ◉ Non
Permissions : Sondages	
Poster créer des sondages	◉ Oui ○ Non
Peut voter dans les sondages	◉ Oui ○ Non
Permissions : Divers	
Peut effectuer une recherche sur le forum	◉ Oui ○ Non

4.3 - Recherche

L'installation et la configuration du moteur de recherche, logiciel ou matériel, dépendent fortement de l'outil choisi. Toutefois, la plupart des outils ont cette particularité en commun, ils sont disponibles à partir d'une adresse URL, qui sert de portail à la recherche. Puisque la majeure partie des logiciels présentés dans ce travail possèdent leur propre moteur de recherche, il peut être approprié d'inclure des outils qui peuvent spécifier la recherche avec des critères plus précis. La figure A-18 illustre ces propos avec un exemple d'interface Web permettant de rechercher à l'intérieur de tous les outils de l'infrastructure.

Figure A-18 : Portail unique pour la recherche



Il est certain que l'interface présentée dans la figure précédente nécessite un temps de développement et d'entretien, principalement pour l'ajout des nouveaux dossiers contrôlés par le SCV. Par contre, l'utilisation unique d'un moteur de recherche (premier élément du haut dans la figure A-18) ne peut garantir à lui seul un résultat souhaitable à tout coup, aussi efficace soit-il, lorsque les mots clés sont fréquemment employés et se retrouvent à plusieurs endroits. L'efficacité, en terme de qualité de réponse, se trouve donc grandement améliorée grâce à ce genre de technique.

L'intégration de la majorité des outils dans un portail de recherche commun est plutôt simple puisqu'ils sont, en grande partie, des logiciels libres développés pour le web, principalement en PHP. Le SCV (*Subversion*) peut être plus complexe à insérer dans ce portail puisqu'il n'est pas du tout fait à partir des mêmes technologies. Cependant, un simple URL peut accéder à un dossier contrôlé par le SCV. Avec *Subversion*, lorsque la personne qui tente d'accéder, par l'entremise de cette adresse URL, à un client (*Subversion*) installé sur son poste comme *TortoiseSVN*, ce client s'ouvre et la recherche peut s'effectuer directement à cet endroit si l'utilisateur a les droits d'accès à ce dossier, bien sûr. Dans ce cas-ci, l'administrateur Web devra mettre à jour le portail de recherche pour ajouter les adresses URL des dossiers des différents projets dans la liste, comme le « Dossier de Lyb » dans la figure A-18).

5 - Planification de projet

Le dernier outil présenté dans ce travail permet à un gestionnaire de planifier ses projets, mais surtout de partager sa planification avec les différents acteurs qui auront à intervenir dans les processus de réalisation des tâches. Le logiciel sélectionné est *@task*, lequel peut aussi utiliser les fichiers de format *MS-Project*. Il peut s'installer sur les principaux systèmes d'exploitation capables de supporter Java version 1.4.2 ou mieux, les bases de données courantes telles que MySQL, Oracle 8i et plus, Microsoft SQL Server 2000 et plus. Cependant, les deux dernières sont recommandées pour une installation par

nœuds groupés « Cluster Node », qui peut supporter plus de cinq cents utilisateurs. Il peut enfin s'installer sur plusieurs modes clients tels que la plupart des navigateurs Web communs, WAP, Java, SOAP et Flash.

Puisque *@task* est fait pour supporter plusieurs projets, plusieurs équipes et même plusieurs studios, l'installation ne devrait se faire qu'une seule fois, à moins d'être dans l'incapacité de sécuriser le réseau entre deux endroits géographiquement éloignés. Avant de commencer à gérer les projets, il est nécessaire d'introduire toutes les personnes susceptibles de travailler au sein des différents projets dans le système et d'entrer les données concernant leur utilisation, et le coût approximatif de chaque ressource, afin de calculer les totaux des tâches et des projets. Cette étape ne doit être réalisée qu'une seule fois, mais doit être mise à jour lors de l'ajout de nouveau personnel et quelques fois pour s'assurer que les montants et les équipes sont encore cohérents. Bien qu'un outil tel que *@task* possède un bon nombre de fonctionnalités, ce guide s'intéresse principalement aux points suivants :

Gestionnaires :

- Gestion des équipes et des membres;
- création d'un projet;
- structure de découpage des tâches;
    - élaboration des différentes tâches du projet;
    - attribution des contraintes de précédence;
    - estimation des dates d'échéance pour chacune d'elles;
    - association des ressources aux tâches;
    - planification des livrables (*milestones*) et confirmation.
- surveillance des projets.

Membres de l'équipe :

- Avancement des tâches;
    - heures travaillées et niveau d'accomplissement;
    - notes d'explications.

5.1 - Gestionnaire

Il est certain que le ou les gestionnaires seront ceux qui utiliseront davantage l'outil de gestion de projet. *@task.* Il leur permet d'ailleurs de gérer les équipes et facilite le partage des ressources clés. Il est aussi possible d'établir un coût approximatif de chacune des tâches, ainsi que de l'ensemble du projet avec un descriptif de l'utilisation du temps de chaque personne sur chacune des tâches.

Dans une organisation où les chargés de projet travaillent bien ensemble et ne sont pas en compétition, nous recommandons de leurs attribuer les droits d'accès « Multi-Group Admnistrator » et non « Project Manager », car ce dernier est plutôt frustrant puisqu'il ne dispose pas des privilèges nécessaires à l'attribution d'une ressource dont il n'est pas le superviseur. Dans un cadre rigide de projet, cela pourrait être souhaitable, mais dans un contexte où les ressources passent constamment d'un projet à l'autre, comme celui des jeux vidéo, ce niveau d'accès est trop restrictif. Le gestionnaire avec le niveau d'accès « Multi-Group Admnistrator » peut donc créer des projets, des équipes, des tâches, et assigner des ressources.

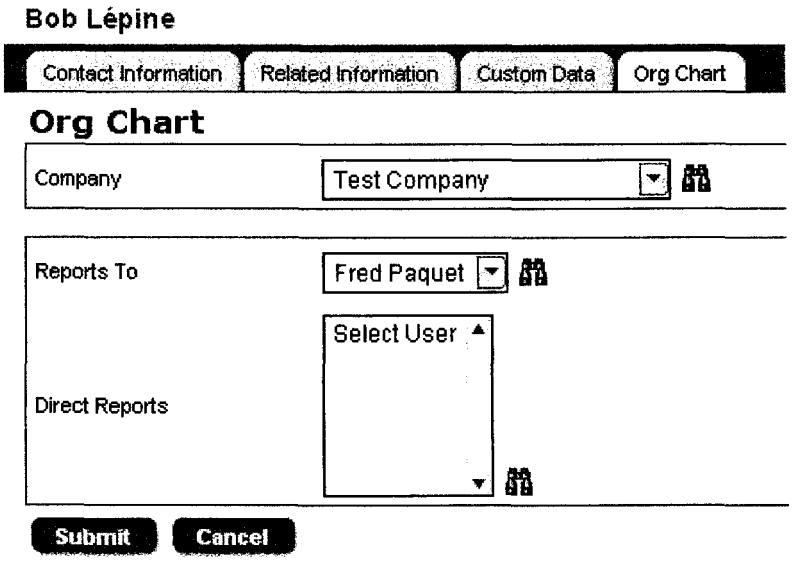
5.1.1 – Gestion des membres et des équipes

Dans la création des utilisateurs ou des membres, il est possible de sélectionner leurs supérieurs. Cette configuration permet aux gestionnaires de projets ayant des subordonnés et un niveau d'accès « Project Manager », de leur attribuer des tâches ou de les inclure dans leurs projets. Tel que mentionné auparavant, sans cet organigramme, le gestionnaire ne pourrait distribuer les tâches aux collaborateurs du projet.

La figure A-19 démontre l'assignation d'un supérieur (*Fred Paquet*) à un membre de l'équipe (*Bob Lépine*) dans *@task.* Il est aussi possible de créer des équipes selon les professions ou les **rôles** des membres. Par exemple, nous pouvons insérer toutes les personnes qui font de la programmation d'intelligence ou qui sont capables de le faire dans une équipe que nous appellerions « développeur AI ». Ainsi, si un gestionnaire a besoin

rapidement d'un tel spécialiste, il pourrait chercher directement dans @task et voir qui est disponible et connaître les tâches sur lesquelles chacun d'eux travaille à ce moment-là, si évidemment ce dernier détient les droits d'accès suffisants.

Figure A-19 : Assignation d'un supérieur à un membre dans *@task*

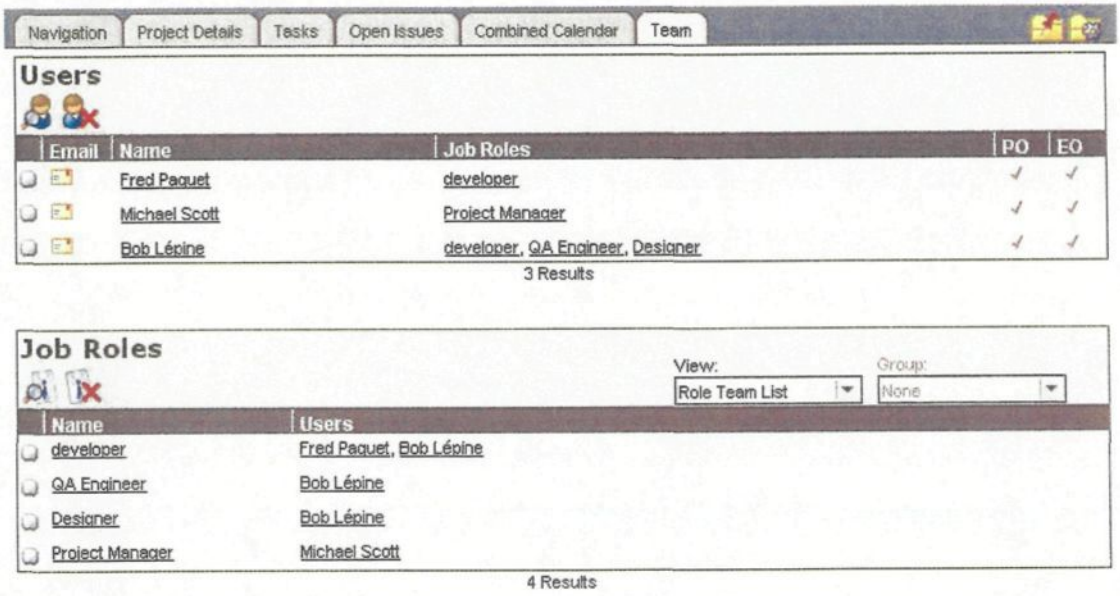


Une fois l'organigramme achevé et les équipes créées, le gestionnaire peut commencer à concevoir son projet et les tâches qui seront nécessaires à sa réalisation, que nous verrons au point suivant « 5.1.2 : Création d'un projet ». À l'intérieur des projets, il est maintenant possible pour un gestionnaire d'ajouter les membres qui lui sont subordonnés, aux tâches auxquelles il veut les assigner. La figure A-20 est un exemple d'équipe supervisée par un gestionnaire de projet, avec les rôles que chacun peut tenir au sein du projet.

Avec ce type d'interface, le gestionnaire peut rapidement connaître les ressources dont il dispose et les forces de chacun. Il peut aussi rechercher et affecter de nouvelles ressources dans le projet à partir de cette interface ou définir de nouveaux rôles qui sont

nécessaires à la réalisation du projet et ainsi connaître les ressources de l'organisation pouvant remplir ces fonctions, ainsi que leur disponibilité.

Figure A-20 : Membres assignés et rôles nécessaires à un projet dans *@task*



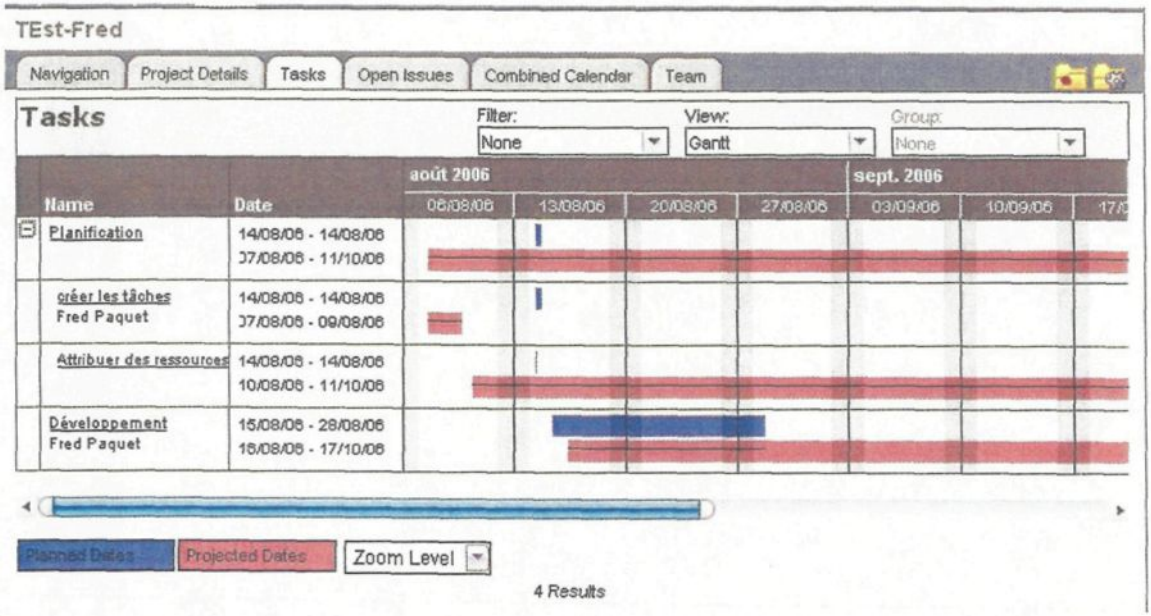
5.1.2 – Création d'un projet

La création d'un nouveau projet dans *@task* est plutôt simple et peut se faire par l'importation d'un fichier *MS-Project*; il est préférable d'avoir la version anglaise. Une fois les propriétés du nouveau projet complétées, entre autres la description, la date de départ, le budget estimé, les tâches peuvent être ajoutées.

La figure A-21 présente l'interface Web du diagramme de Gantt dans *@task*. Puisque le diagramme de Gantt est probablement l'un des outils les plus utilisés et les mieux compris en planification de projet par les membres des différents secteurs, nous conseillons de marquer cette option comme page par défaut de l'application. Pour ce faire, il suffit de cliquer sur le premier icône du coin supérieur droit de l'écran. Dans le diagramme de Gantt, la barre du haut, de couleur bleu, représente la durée estimée lors de la planification, la création de la tâche, et la barre du bas, de couleur rouge indique l'échéance possible ou réelle selon l'avancement du projet.

L'avancement du projet peut se faire de façon manuelle ou *automatique*, soit en fonction de l'avancement de chacune des tâches. Puisque nous sommes en faveur de l'implication des membres de l'équipe dans l'estimation de l'avancement, il peut être intéressant d'utiliser le mode automatique pour ainsi donner au gestionnaire de projet une bonne idée de l'avancement global du projet, même s'il ne peut se servir uniquement de cet indicateur pour évaluer la progression du projet.

Figure A-21 : Membres assignés et rôles nécessaires à un projet dans *@task*



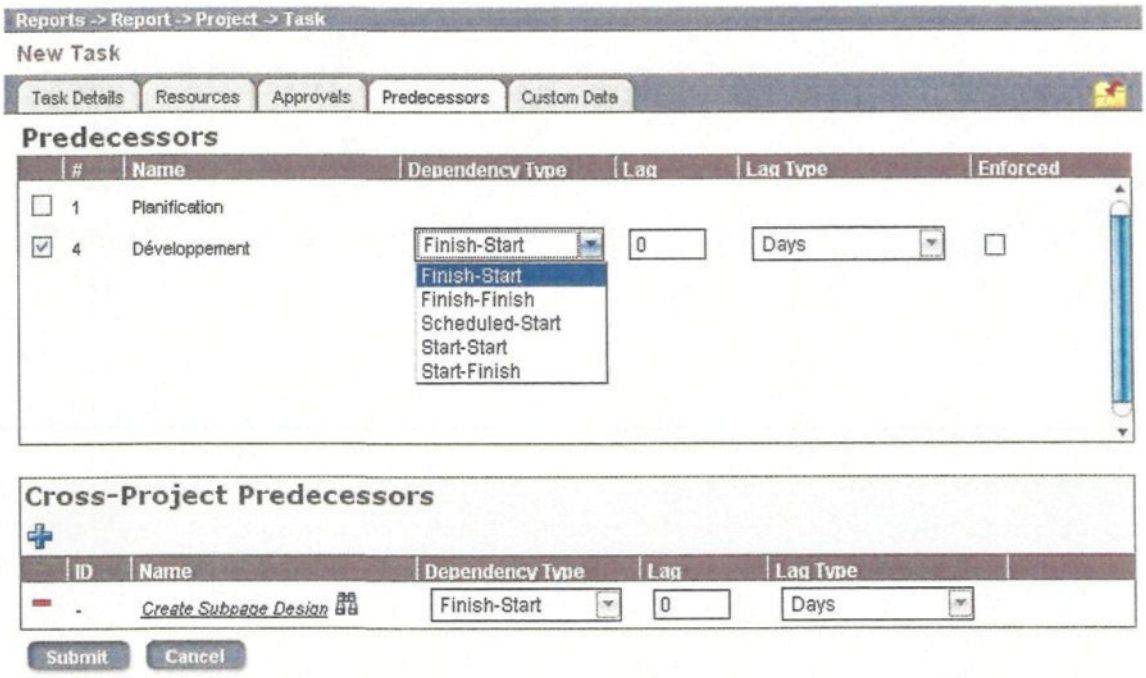
5.1.3 – Structure de découpage des tâches

Puisque *@task* est fait pour supporter plusieurs projets, l'une des fonctionnalités intéressantes de ce logiciel est la possibilité d'introduire des dépendances entre les tâches (prédécesseurs) provenant d'autres projets (dépendances externes).

La figure A-22 illustre l'interface pour l'insertion des prédécesseurs d'une tâche et l'ajout d'une dépendance externe. Dans cet exemple, la nouvelle tâche devra attendre la fin de la tâche « *Create Subpage Design* » d'un autre projet; il est possible d'en voir la source

en cliquant sur l'icône des jumelles, avant de pouvoir commencer. Les différentes dépendances possibles dans *@task* sont celles identifiées dans le PMBOK, soit les liaisons « fin-début », « fin-fin », « début-début » et début-fin ».

Figure A-22 : Gestion des dépendances entre les tâches dans *@task*

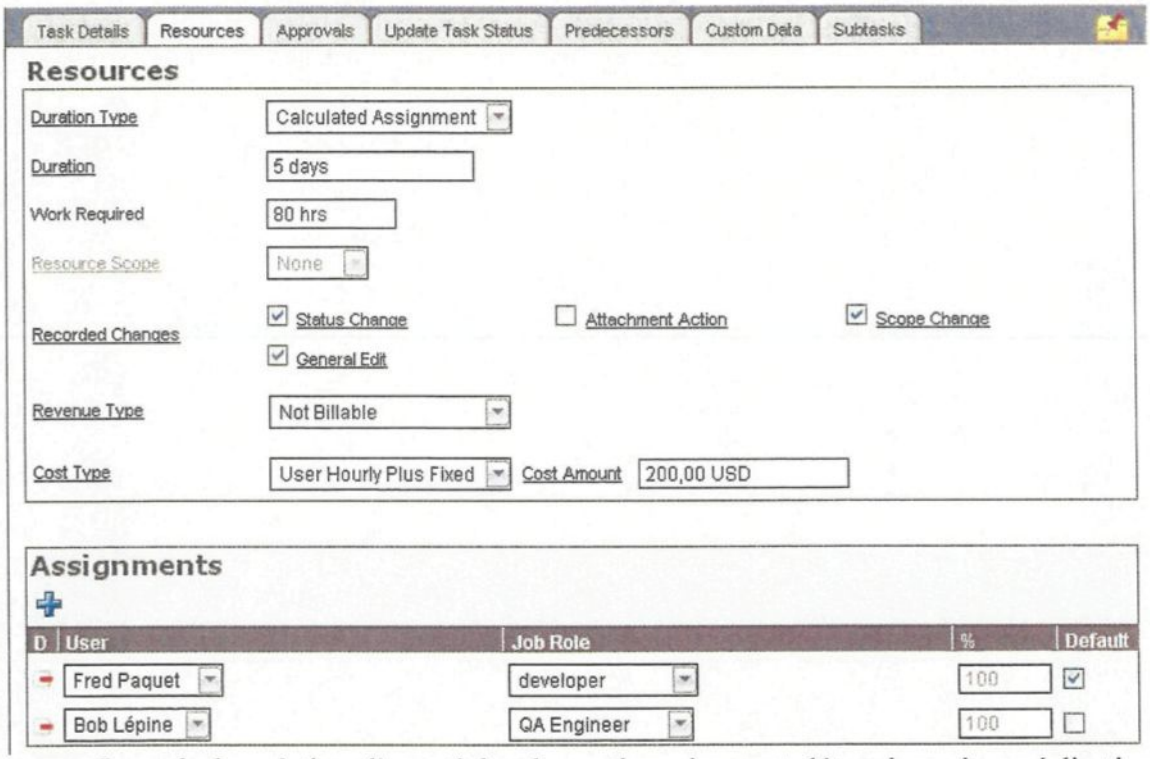


L'estimation des échéances et de l'effort nécessaire pour la réalisation des tâches fonctionne sensiblement de la même façon que dans *MS-Project*. La durée d'une tâche peut être pilotée par l'effort (*effort driven*), ce qui indique que lorsque l'on ajoute des ressources, le temps devrait diminuer, ou pilotés par jalon, c'est-à-dire le nombre de jours ou d'heures de travail, qui implique que l'échéance ne change pas mais que les coûts augmentent.

La figure A-23 représente le tableau des ressources essentielles à l'achèvement d'une tâche. Dans l'optique du partage des ressources dans un mode de consultation, il est aussi possible d'inclure les revenus que nous procure une personne versus le salaire qu'elle nous coûte. De plus, lorsqu'une tâche nous coûte un montant fixe, par exemple la location d'équipement ou l'achat de matériel, un endroit spécifique est prévu dans cette interface

pour permettre de mieux évaluer le prix de chacune des tâches et donc, du projet. L'assignation des ressources, ainsi que l'attribution d'un pourcentage d'implication dans la tâche se font dans le bas de l'écran. Le montant versé à une ressource dépend de son salaire, ou si aucun salaire n'est associé, le salaire par défaut du rôle, par exemple un développeur AI, sera utilisé.

Figure A-22 : Assignation des tâches et calcul du budget dans *@task*



Lors de la création d'une tâche, le gestionnaire peut déterminer si sa réalisation engendre un livrable et si elle doit être validée par quelqu'un. Dans ces deux cas précis, la tâche ne pourra être terminée que lorsque le livrable sera complété, c'est-à-dire l'envoi confirmé et la commande livrée, reçue, ou terminée, ou lorsque la personne en question aura validé l'accomplissement de la tâche.

5.1.4 – Surveillance du projet

Comme la plupart des outils de planification de projet, *@task* possède un bon nombre de rapports différents ou de vues permettant de suivre l'évolution d'un projet ou d'une tâche selon plusieurs facteurs. Le tableau A-2 présente l'ensemble des rapports possible dans *@task* pour suivre un projet.

Tableau A-2 : Rapport disponible dans *@task*

Project Reports

View: Reports List | Group: UI View Type

Name	Desc	Scope	Type
Name			
Calendar			
Combined Calendar	A calendar of tasks and issues for the project	Global	Task
Issue Calendar	A calendar of issues for the project	Global	Task
Task Calendar	A calendar of tasks for the project	Global	Task
Name			
Chart			
Completed Issues By Week	Completed issues grouped by week of completion displayed in a bar chart	Global	Chart
Completed Issues By Week By User	Completed issues grouped by week of completion and by user displayed in a bar chart	Global	Chart
Hours By User Chart	Hours grouped by the user who submitted them displayed in a bar chart	Global	Chart
Issues By Status Chart	Issues grouped by status displayed in a pie chart	Global	Chart
Tasks By Condition Chart	Current tasks grouped by Progress Status	Global	Chart
Tasks By Status Chart	Tasks grouped by status displayed in a pie chart	Global	Chart
Name			
Custom Data			
Custom Data	Custom data defined for the current object	Global	Global
Name			
History			
Notes	A list of notes attached to the current object	Global	Note
Name			
List			
All Documents	A list of all documents, including documents attached to child items, attached to the current object	Global	Document
Documents	A list of documents attached to the current object	Global	Document
Hours	A list of hours added to this	Global	Hour
Open Issues	Open issues belonging to the project	Global	Issue
Project Reports	A list of reports and charts available to the Project object	Global	Portal Section
Tasks	Tasks belonging to the project	Global	Task
Team Job Roles	Job roles on the team for this project	Global	Job Role
Team Members	Users on the team for this project	Global	Project User
Name			
Three-Column Layout			
Project Details	Detailed information about the project	Global	Project
Name			
Two-Column Layout			
Project Summary	Summary information about the project	Global	Project

Le gestionnaire peut décider de recevoir un courriel d'alerte à chacune des modifications apportées sur le projet ou lors de quelques autres événements déclencheurs. De plus, il peut décider d'imposer son approbation pour valider l'achèvement de chacune des tâches, tel que mentionné dans le point précédent. Cependant, cette mesure peut venir à fausser certaines données, puisque la tâche sera considérée comme « en cours », ce qui signifie que le coût d'une tâche peut continuer à monter lorsque la tâche est pilotée par jalon, même si elle est en vérité terminée.

5.2 – Membres de l'équipe

Bien qu'on ne demande pas aux membres des différentes équipes de devenir des professionnels de la gestion de projet, il est cependant utile de les former aux bases de l'outil qui sera employé, pour que ceux-ci puissent en tirer avantage. Il leur est essentiel de connaître les tâches auxquelles ils sont associés et les échéances, et de pouvoir inscrire le temps qu'ils y mettent ainsi que les causes des variations dans les temps planifiés.

Tout comme les gestionnaires, il peut être intéressant pour les membres des différentes équipes de configurer le diagramme de Gantt comme interface par défaut d'un projet, mais surtout l'ensemble des tâches de tous les projets avec lesquelles ils sont associés. Cet écran, présenté dans la figure A-23, leur permet de connaître le travail qu'ils doivent accomplir (*workload*) et le statut des tâches à réaliser.

Figure A-23 : Travail à effectuer et statut des tâches dans *@task*

My Tasks

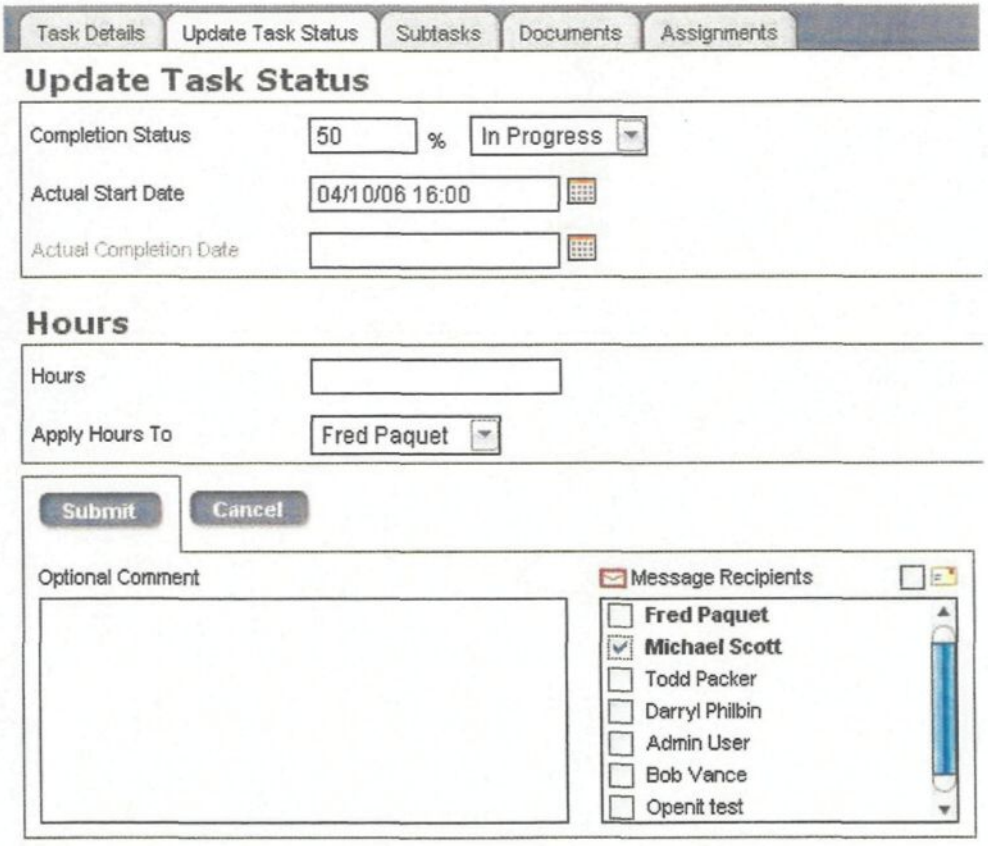
Filter: None | View: Standard | Group: None

ID	Task	Dur	Assigned	Project	Pln Start	Pln Comp	%	!	Flags
18204	Determine Transactions	5 Days	Michael Scott	UT Test Project	27/07/06	03/08/06	50	2	
34567	List Entities	15 Days	Michael Scott	Another	23/08/06	12/09/06	0	2	
45319	BAU / Support placeholder	365 Days	Ode Nederlof	Ongoing support / BAU	12/09/06	04/02/08	0	1	
34568	Determine Transactions	5 Days	Michael Scott	Another	13/09/06	19/09/06	0	2	
45314	Define	2 Days	Ode Nederlof	Storage project	18/09/06	19/09/06	0	2	
45316	Analyse	3,5 Days	Meredith Palmer	Storage project	22/09/06	27/09/06	0	2	

5.2.1 – Avancement des tâches

Puisqu'il est possible de compléter l'avancement d'un projet et la mise à jour du budget automatiquement via la progression de chacune des tâches qui le compose, il est nécessaire pour arriver à ce résultat, que chacun inscrive le temps qu'il a passé sur la tâche et le niveau de progression de celle-ci. La figure A-24 est un exemple de l'interface qui permet d'inscrire le temps et l'avancement, mais aussi la possibilité d'informer les personnes impliquées et le gestionnaire de projet par un message à l'interne de l'outil, ou via un courriel. Cette dernière fonctionnalité peut être fort intéressante lors d'une mise à jour importante, ou d'un pépin qui survient. À partir de cet endroit (onglets adjacents), le membre peut aussi insérer un document, une copie d'écran par exemple, ou un commentaire sur le temps qu'il vient de passer à travailler sur une tâche.

Figure A-24 : Mise à jour de l'avancement et des heures travaillées dans *@task*



5 – Points à développer

L'un des aspects les plus irritants de cette infrastructure est la gestion des utilisateurs. Malgré le fait que la plupart des outils présentés dans ce travail sont accessibles via un navigateur et peuvent conserver les paramètres d'authentification (nom d'utilisateur et mot de passe), il n'en demeure pas moins que cela fait plusieurs comptes à créer et surtout à gérer. L'idéal dans ce cas-ci serait de centraliser la gestion des noms d'utilisateurs et des mots de passes sur une même page. Donc à partir d'une simple page Web, la personne pourrait changer son mot de passe et cela se répercuterait sur l'ensemble des outils de l'infrastructure. De plus, il pourrait être intéressant de développer un outil pour le gestionnaire du réseau qui lui offrirait la capacité de créer rapidement des comptes et des droits d'utilisation sur tous les logiciels et plus spécialement sur Subversion, puisque cela est plus long qu'avec les autres.

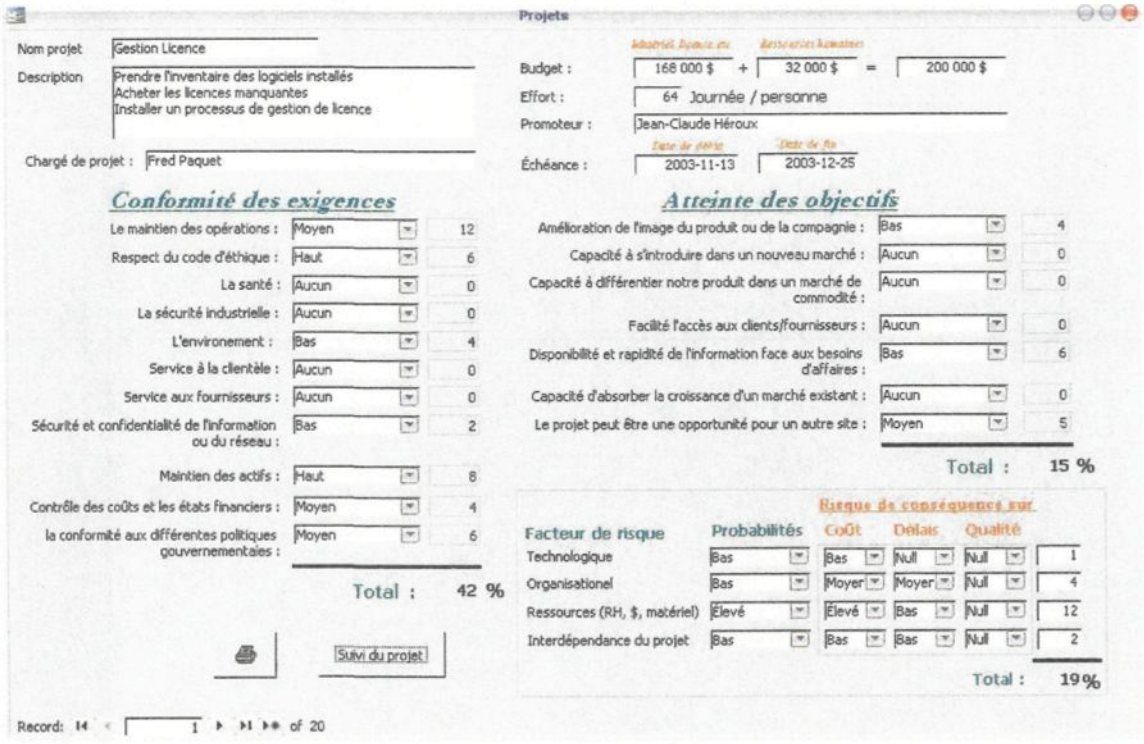
Dans le même ordre d'idées, les différents outils pourraient aussi être légèrement transformés afin d'utiliser pour chacun la même session d'authentification. Ceci impliquerait donc de n'avoir qu'à se connecter une seule fois sur l'infrastructure et de pouvoir naviguer entre les outils de façon plus transparente. Cette amélioration serait utile afin d'augmenter encore davantage l'immersion des systèmes et réduire au maximum les irritants. Cela permettrait une diminution des risques de rejet et de rébellion contre les changements au sein des processus de l'organisation.

ANNEXE 2 : GESTION DES RISQUES

Les trois prochaines figures (A-25, A-26 et A-27) démontrent un exemple d'application maison pouvant être développée pour gérer les risques au sein des portefeuilles de projets. Plusieurs outils de gestion de projet offrent ce genre de fonctionnalités, mais il est facile et rapide de développer un logiciel maison capable de répondre aux critères parfois précis des hauts gestionnaires.

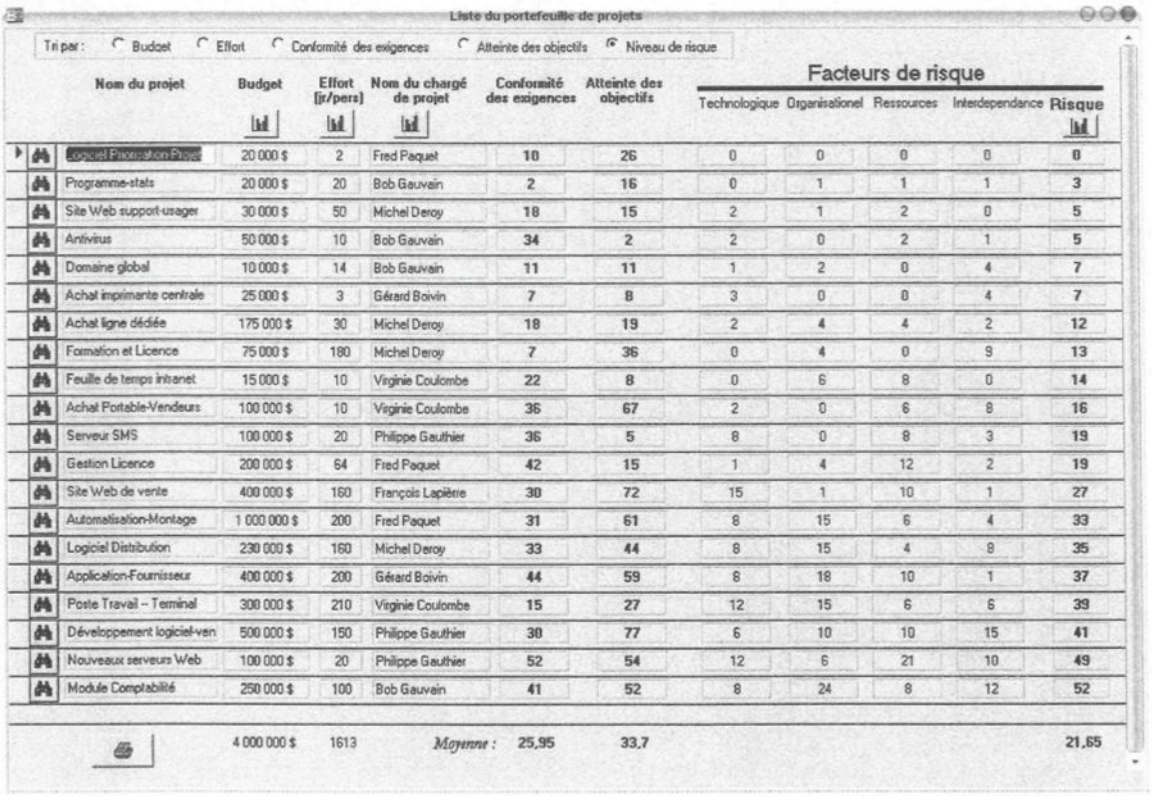
La figure A-25 démontre l'interface pour saisir l'ensemble des projets que l'organisation désire mettre en place et le type de risque pris en considération dans ce cas-ci, avec une pondération pour chacun d'eux.

Figure A-25 : Portefeuille de projet et gestion des risques



Lorsque tous les projets sont entrés dans l'application, il est ensuite facile de générer une liste de tous les projets, triée en fonction des risques de chacun. La figure A-26 est un exemple de ce genre de liste. Cet outil permet aux hauts gestionnaires de prendre des décisions à savoir quels seront les projets qui seront mis de l'avant et lesquels seront mis en attente, grâce à plusieurs critères dont le coût du projet, l'effort nécessaire, les conformités des exigences, l'atteinte des objectifs et les niveaux de risques.

Figure A-26 : Portefeuille et projet trié selon plusieurs critères

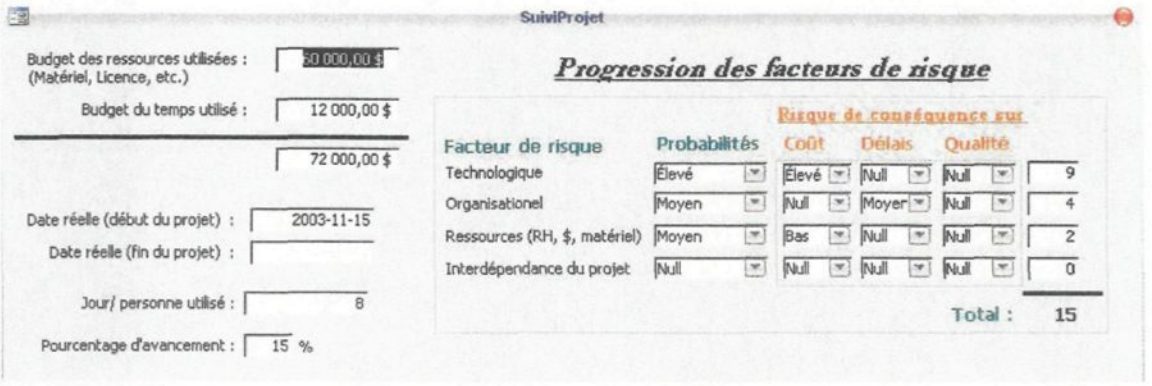


Nom du projet	Budget	Effort (jr/pers)	Nom du chargé de projet	Conformité des exigences	Atteinte des objectifs	Technologique	Organisationel	Ressources	Interdependance	Risque
Logiciel Formation Projet	20 000 $	2	Fred Paquet	10	26	0	0	0	0	0
Programme-stats	20 000 $	20	Bob Gauvain	2	16	0	1	1	1	3
Site Web support-usager	30 000 $	50	Michel Deroy	18	15	2	1	2	0	5
Antivirus	50 000 $	10	Bob Gauvain	34	2	2	0	2	1	5
Domaine global	10 000 $	14	Bob Gauvain	11	11	1	2	0	4	7
Achat imprimante centrale	25 000 $	3	Gérard Boivin	7	8	3	0	0	4	7
Achat ligne dédiée	175 000 $	30	Michel Deroy	18	19	2	4	4	2	12
Formation et Licence	75 000 $	180	Michel Deroy	7	36	0	4	0	9	13
Feuille de temps intranet	15 000 $	10	Virginie Coulombe	22	8	0	6	8	0	14
Achat Portable-Vendeurs	100 000 $	10	Virginie Coulombe	36	67	2	0	6	8	16
Serveur SMS	100 000 $	20	Philippe Gauthier	36	5	8	0	8	3	19
Gestion Licence	200 000 $	64	Fred Paquet	42	15	1	4	12	2	19
Site Web de vente	400 000 $	160	François Lapierre	30	72	15	1	10	1	27
Automatisation-Montage	1 000 000 $	200	Fred Paquet	31	61	8	15	6	4	33
Logiciel Distribution	230 000 $	160	Michel Deroy	33	44	8	15	4	8	35
Application-Fournisseur	400 000 $	200	Gérard Boivin	44	59	8	18	10	1	37
Poste Travail – Terminal	300 000 $	210	Virginie Coulombe	15	27	12	15	6	6	39
Développement logiciel-ven	500 000 $	150	Philippe Gauthier	30	77	6	10	10	15	41
Nouveaux serveurs Web	100 000 $	20	Philippe Gauthier	52	54	12	6	21	10	49
Module Comptabilité	250 000 $	100	Bob Gauvain	41	52	8	24	8	12	52
	4 000 000 $	1613	*Moyenne :*	25,95	33,7					21,65

Puisque l'on affirme dans la majorité des méthodologies qu'une gestion des risques sans suivi est inutile, la figure A-27 permet d'entrer la mise à jour du projet afin que les dirigeants puissent suivre son évolution et être au courant des difficultés ou des facilités pouvant faire évoluer les risques positivement ou négativement. Par la suite, un tableau de bord peut être élaboré pour déterminer des indicateurs précis comme le niveau de risque, le

budget, le temps, et bien d'autres, qui permettront aux gestionnaires de prendre des décisions sur l'avenir de chacun des projets.

Figure A-27 : Suivi des risques



Bien qu'il soit préférable d'utiliser les fonctionnalités offertes à cet effet par les différents outils de planification de projet, ce qui empêche la redondance, cet exemple permet de démontrer qu'un logiciel peut être développé très rapidement, pas plus de quelques heures avec un outil comme *MS-Access*, puis répondre à plusieurs besoins plus spécifiques en matière de gestion des risques.