Amélioration des processus de développement logiciel

Ludovic Bernada

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

AIX EN PROVENCE

MEMOIRE

Présenté en vue d'obtenir

Le DIPLOME d'ingénieur CNAM

SPECIALITE : Informatique

OPTION : Ingénierie des systèmes d'information (ISI)

Par

Ludovic BERNADA

Amélioration des processus de développement logiciel

Soutenu le 11 juillet 2012

JURY

PRESIDENT : Yves Laloum

MEMBRES : Noël Quessada, Bastien Pesce, Axel Roetynck, Jean-Pascal Maire, Daniel Catoio

le cnam Provence-Alpes-Côte d'Azur

Région Provence-Alpes-Côte d'Azur

1 RÉSUMÉ

OBJET DU MEMOIRE

- Analyse des méthodes de travail du groupe SI
- Proposition de nouvelles solutions.

AXES PRINCIPAUX DE L'ETUDE

- Étude de deux projets : DS3 et ILSDB.
- Étude technique :
    - Améliorations proposées :
        - Assurance qualité du logiciel,
        - Fiabilité
        - Maintenabilité
        - Procédures de tests...
- Gestion du projet
    - Planification des développements
    - Amélioration des spécifications :
        - Fonctionnelles
        - Techniques

CONCLUSION

- La compréhension du besoin est le cœur de l'informatique de gestion.

Mots-clés : CMMI, Java, Python, spécifications, Redmine, Eclipse, SVN, analyse de besoin, amélioration des processus.

2 SUMMARY

PURPOSE OF THE DISSERTATION

- Analysis of working methods of the SI group
- Proposing new solutions.

PRINCIPAL AXES OF THE STUDY

- Study of two projects: DS3 and ILSDB.
- Technical study:
    - Proposed improvements:
        - Software quality assurance
        - Fiability
        - Maintainability
        - Testing procedures...
- Project management
    - Development planning
    - Improved specifications :
        - Functional
        - Techniques

CONCLUSION

- Understanding the need is the heart of business computing

Keywords : CMMI, Java, Python, specifications, Redmine, Eclipse, SVN, analysis of the need, process improvement.

3 SOMMAIRE

4 REMERCIEMENTS

CE MEMOIRE EST DEDICACE

A ma « belle grand-mère », Annie, qui a supporté avec patience et gentillesse mes innombrables fautes d' « ortaugrafe » et mes anglicismes douteux.

A mon père qui m'a toujours poussé (parfois à coup de pied…) à continuer mes études, à son soutien moral et à son porte-monnaie efficace.

Au « Grôle SI », entité aussi bizarroïde que sympathique dont je fais partie, et à son cri de guerre, que la morale m'interdit ici de reproduire (VMS...)

A ma famille et à mes amis pour leur soutien. Promis, j'arrête de parler de mémoire.

A « Raoudha les bon tuyaux » pour le sujet du mémoire, ainsi qu'à LGM pour les jours qui m'ont été accordés pour sa réalisation.

Et enfin au CNAM et à ses professeurs qui m'ont amené jusqu'ici et m'ont permis de reprendre mes études. Si on m'avait dit il y a dix ans que je finirais ingénieur …

CE MEMOIRE N'EST PAS DEDICACE

Aux « peut mieux faire », « manque de travail », « nul » et autres sympathiques remarques qui ont agrémenté ma scolarité et mes bulletins.

D'UNE FACON GENERALE, CE MEMOIRE EST DEDICACE AU CORPS ENSEIGNANT, AVEC QUI MES RELATIONS FURENT AUSSI DIVERSES QU'ENRICHISSANTES

5 INTRODUCTION

C'est dans le cadre de la formation "d'Ingénieur en Système d'Information" au sein du Conservatoire des Arts et métiers d'Aix en Provence que se présente ce Mémoire de fin d'étude intitulé :

« AMÉLIORATION DES PROCESSUS DE DEVELOPPEMENT LOGICIEL ».

Sur les conseils de mon professeur et de mon entourage professionnel, je vais m'appuyer sur une mission d'audit qui m'a été confiée en décembre 2011; La problématique étant la suivante :

« Quels processus devons-nous mettre en place pour accompagner le développement croissant de notre activité ».

Je travaille en effet au sein de la société LGM, créée en 1991, depuis juillet 2010. Elle a pour principal domaine d'activités le Management et l'Ingénierie de grands projets, et réalise des prestations de conseils et d'études. Rattaché au groupe Système d'Information, j'occupe un poste de consultant.

Ce groupe, crée en 2010 a pour objectif de répondre aux besoins de nos clients en termes de système d'information. Il connaît une forte croissance de son activité. Afin d'accompagner cette croissance, il est nécessaire de mettre en place certaines actions :

- Mise en place d'outils et de méthodes visant à améliorer la qualité de notre travail
- Amélioration de la gestion de projet et des processus de développement

C'est dans ce contexte que m'a été confiée la mission d'étude que je vais détailler dans ce document. L'objectif étant de décrire nos processus et de proposer des améliorations. Cette analyse prendra la forme d'un livre blanc qui sera remis à la direction des Systèmes d'Information, qui analysera et mettra en œuvre les propositions qu'elle jugera pertinentes.

J'ai en charge la description et l'analyse des processus pour la partie gestion de projet et la partie développement.

Ce mémoire s'articule autour de trois grands axes.

- Analyse de l'existant et recherche de solutions
- Gestion de projet et planification
- Prestation technique

Après la conclusion de l'étude, je terminerai en tentant de dresser un bilan personnel de ce travail.

6 PRÉSENTATION DU GROUPE LGM

6.1 LE GROUPE LGM

6.1.1 BREF HISTORIQUE DE LGM GROUPE

LGM se spécialise dans le conseil et l'expertise en management de grands projets. Cette entreprise a le statut de Société par Actions Simplifiées (SAS) contrôlée exclusivement par des consultants associés, assurant ainsi une indépendance vis-à-vis de groupes industriels. Ceci garantit une totale impartialité dans les missions effectuées et les recommandations faites.

Au cours de ses 20 années d'existence, LGM a étendu son savoir-faire à différents domaines tels que le Maintien en Condition Opérationnelle (MCO), le Soutien Logistique Intégré (SLI) ou encore la sûreté de fonctionnement (SdF). L'un des objectifs de LGM sur le long terme, est d'être reconnu comme référent dans le panel de métiers qu'il propose à ses clients.

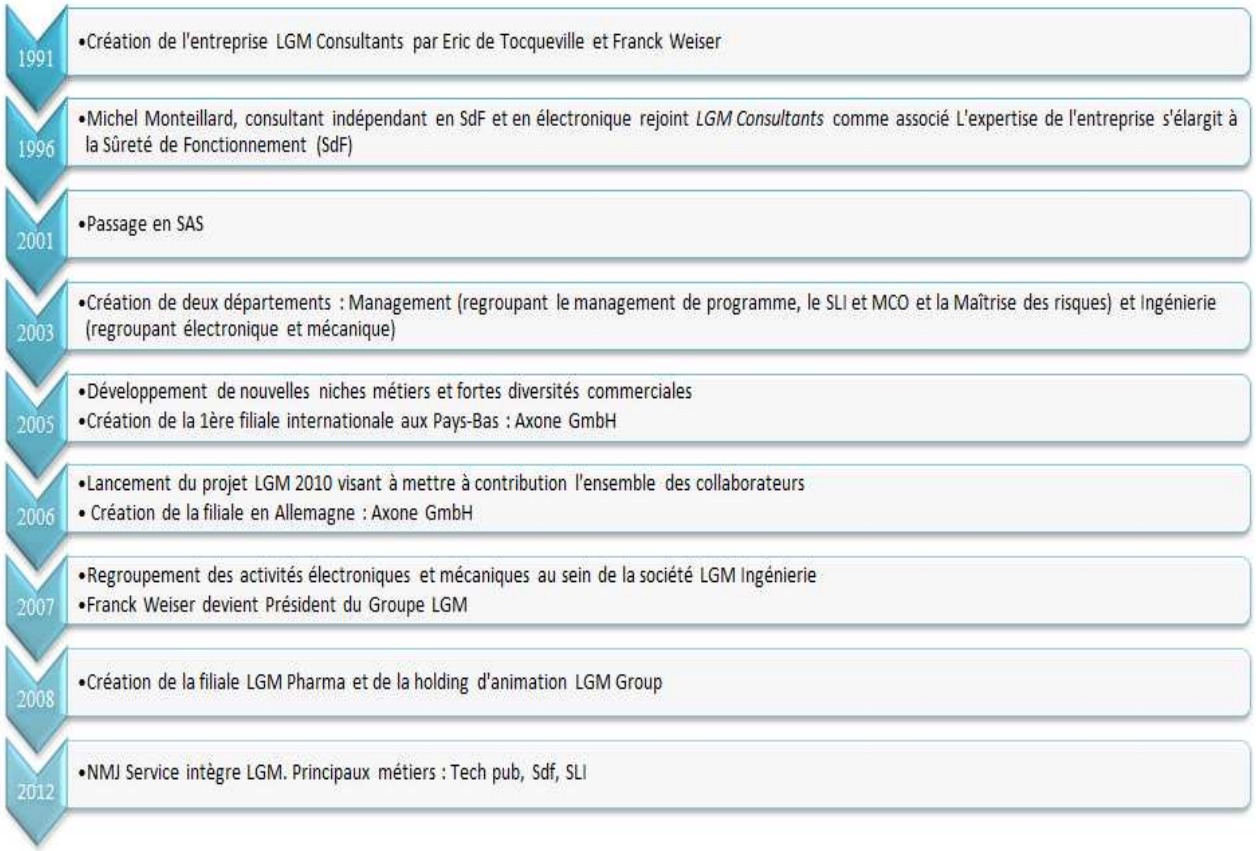


Figure 1 : Bref historique du groupe LGM

6.1.2 LGM EN QUELQUES CHIFFRES

LGM a réalisé un chiffre d'affaire de 46M€ en 2011. Le chiffre d'affaires de LGM est réparti de manière équilibrée, en termes de clientèle et de prestations. Ceci confère à l'entreprise une grande stabilité, et lui permet de maintenir la diversité et la richesse de ses prestations.

Figure 2 : Evolution du chiffre d'affaire LGM

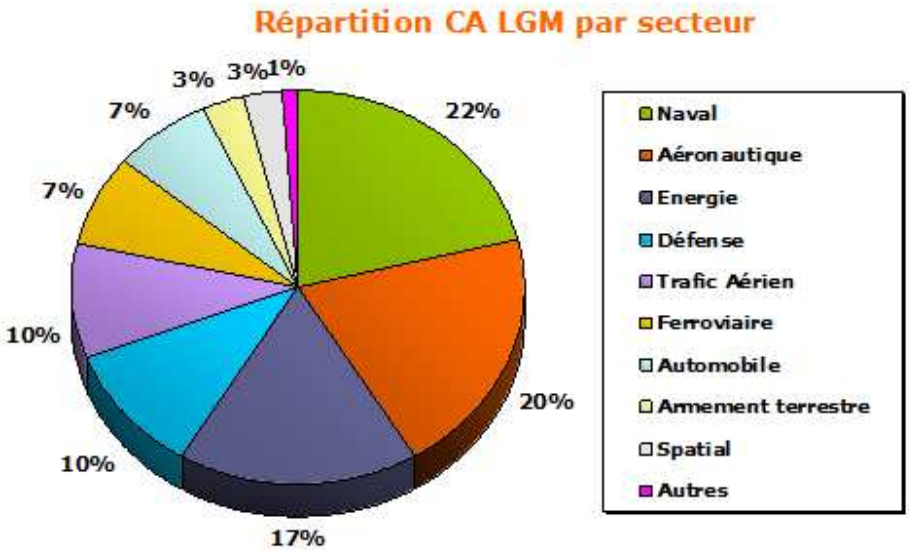


Figure 3 : Répartition du chiffre d'affaire par secteur

6.1.3 LES PÔLES DE COMPÉTENCES

LGM a orienté sa stratégie sur la maîtrise complète des grands projets industriels de hautes technologies, à la fois sur l'aspect ingénierie et sur l'aspect management de programme. Les pôles d'expertise de LGM sont les suivants :

Figure 4 : Les pôles de compétences LGM

6.1.4 PRINCIPAUX CLIENTS

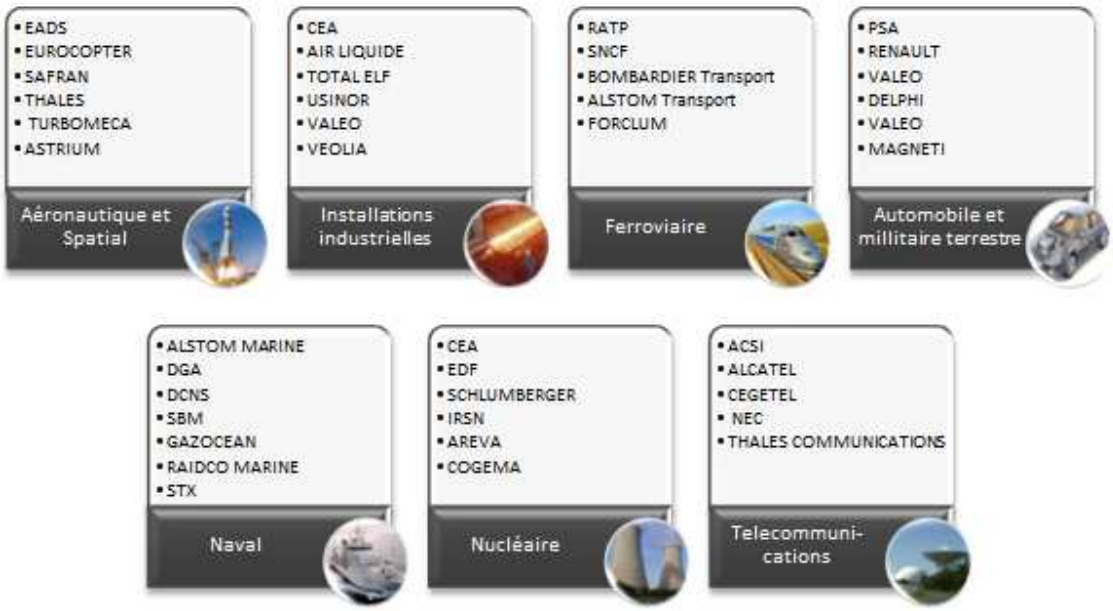


Figure 5 : Les principaux clients du groupe LGM

6.2 PRÉSENTATION LGM SUD EST

6.2.1 QUELQUES CHIFFRES

L'agence régionale Sud-Est basée à Aix-en-Provence a été créée en 2004 par Frédéric ARNAUD actuel responsable de l'agence qui emploie aujourd'hui 73 personnes : ingénieurs et techniciens centrés sur l'expertise. L'agence enregistre de bons résultats de développement depuis sa création (Figure 6).

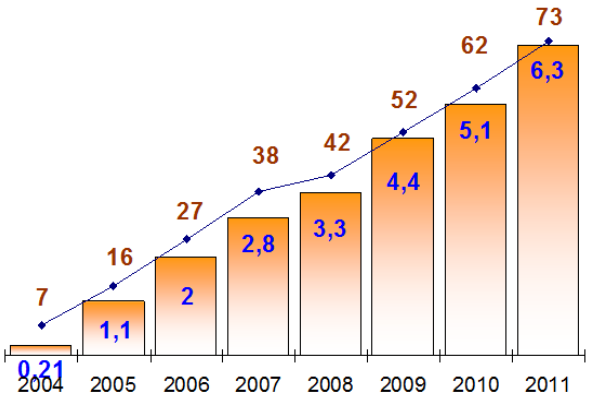


Figure 6 : Chiffre d'affaire LGM

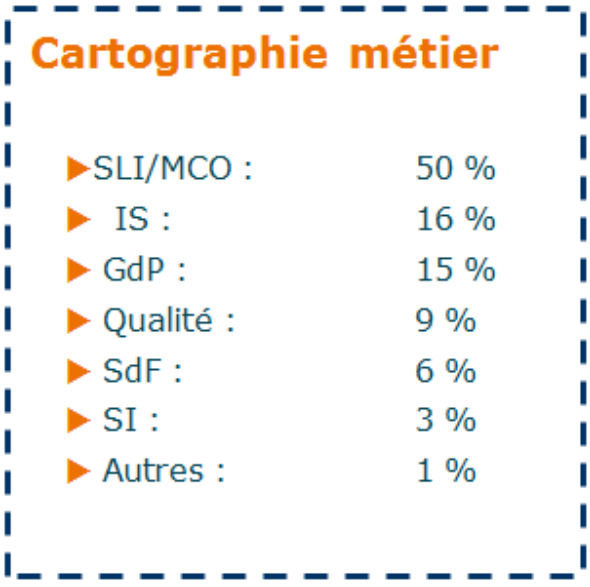


Figure 7 : Répartition du chiffre d'affaire par métier

6.2.2 LES PRINCIPAUX CLIENTS

LGM Sud-Est travaille essentiellement avec de grands groupes industriels notamment:

- EUROCOPTER & NHI
- DCNS (SIS, ING, ASM et LOG)
- CEA
- AREVA
- UDSI et ARMARIS
- Thales Under Water Systems
- Thales Alenia Space
- Single Boys Mooring (Offshore)

7 PRÉSENTATION DU GROUPE SI

7.1 BREF HISTORIQUE

Le groupe Système d'Information (groupe SI) a été créé en 2010. Il compte aujourd'hui quatre personnes, pour un chiffre d'affaire d'environ 130 000 euros. ***Ses principaux objectifs sont :***

- Proposer à nos clients, dans le cadre des métiers de LGM, des solutions outillées professionnelles et pérennes
- Ouvrir une nouvelle voie de développement des métiers LGM en abordant la problématique métier par le SI.

7.2 LES MÉTIERS DU GROUPE SI

Le groupe SI intervient sur trois types de mission :

- Déploiement et maintenance d'outils de GMAO
- Développement d'outils, en liaison avec les métiers de LGM
- Maintenance et évolutions d'outils existants

Ces activités sont réparties comme suit :

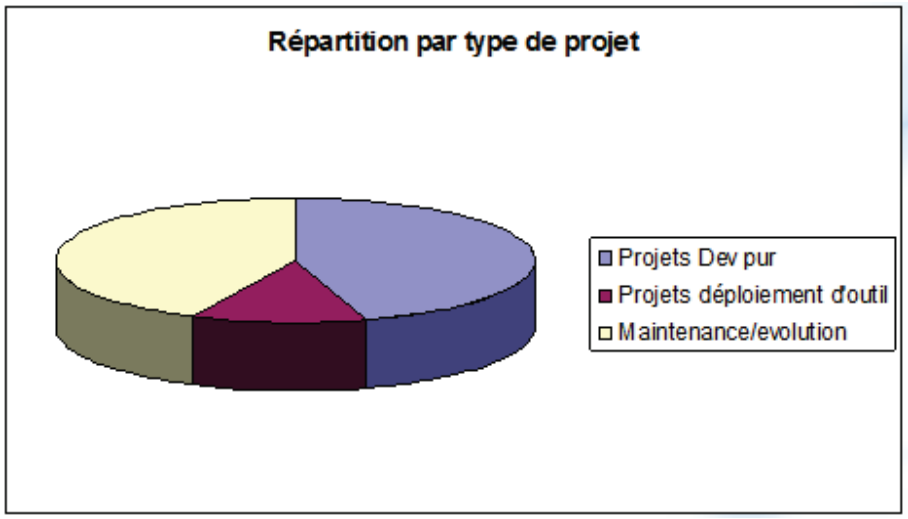


Figure 8 : Répartition par type de projet

7.3 QUELQUES PROJETS DU GROUPE SI

NH Industries

- Réalisation d'un outil de gestion des prix pour l'hélicoptère NH90.
- Technologies : SQL Server, Java EE, Struts2, Jquery. Serveur Tomcat

Eurocopter

- Réalisation d'une base LSA
- Technologies : SQL Server, Java EE, JSF2, Primeface
- Maintenance sur l'outil Mucoma (réalisé par le groupe SI). Mucoma est un outil de management du lien entre les métiers du bureau d'étude et ceux de la gestion de configuration

CEA

- Gestion de la documentation technique et du référentiel d'exigences associé
- Technologies : Sql Server, Java EE, Struts2, jquery. Serveur Tomcat

Gazocean

- Déploiement et maintenance de l'outil de GMAO AMOS

7.4 INTEGRATION DU GROUPE SI AU SEIN DU GROUPE LGM

Le groupe SI travaille en étroite collaboration avec les deux acteurs du système d'information LGM, la direction des systèmes d'information, et la direction technique de l'innovation

- La direction des systèmes d'information (DSI) définit la politique et l'architecture du système d'information du groupe LGM et dispose de sa propre cellule de développement, qui intervient sur la conception et la maintenance des outils internes (intranet, outils de gestion de carrières, GED…etc.).
- Le groupe SI est une entité indépendante de la DSI, intervenant principalement sur des projets clients.
- La direction technique de l'innovation (DTI) est une entité chargée de piloter l'innovation au sein du groupe LGM. Ses missions sont multiples
    - Veille technologique au travers d'un portail informatique dédié
    - Support technique aux projets innovants
    - Mise en place de politiques globales au niveau du groupe LGM.

La DTI est en train de mettre en place une « politique système d'information » pour tout le groupe LGM. Le Groupe SI Aix en Provence est partie intégrante de cette politique, dans la mesure où nous sommes aujourd'hui le premier groupe ayant une activité dédiée uniquement aux systèmes d'informations, dans le cadre des métiers LGM.

Nous travaillons avec la DSI du siège sur les axes suivants :

- Mise en commun de compétences métier :
- Mise en commun de ressources matérielles et logicielles

Ressources informatiques et liaisons entre le groupe SI et la direction des systèmes d'information.

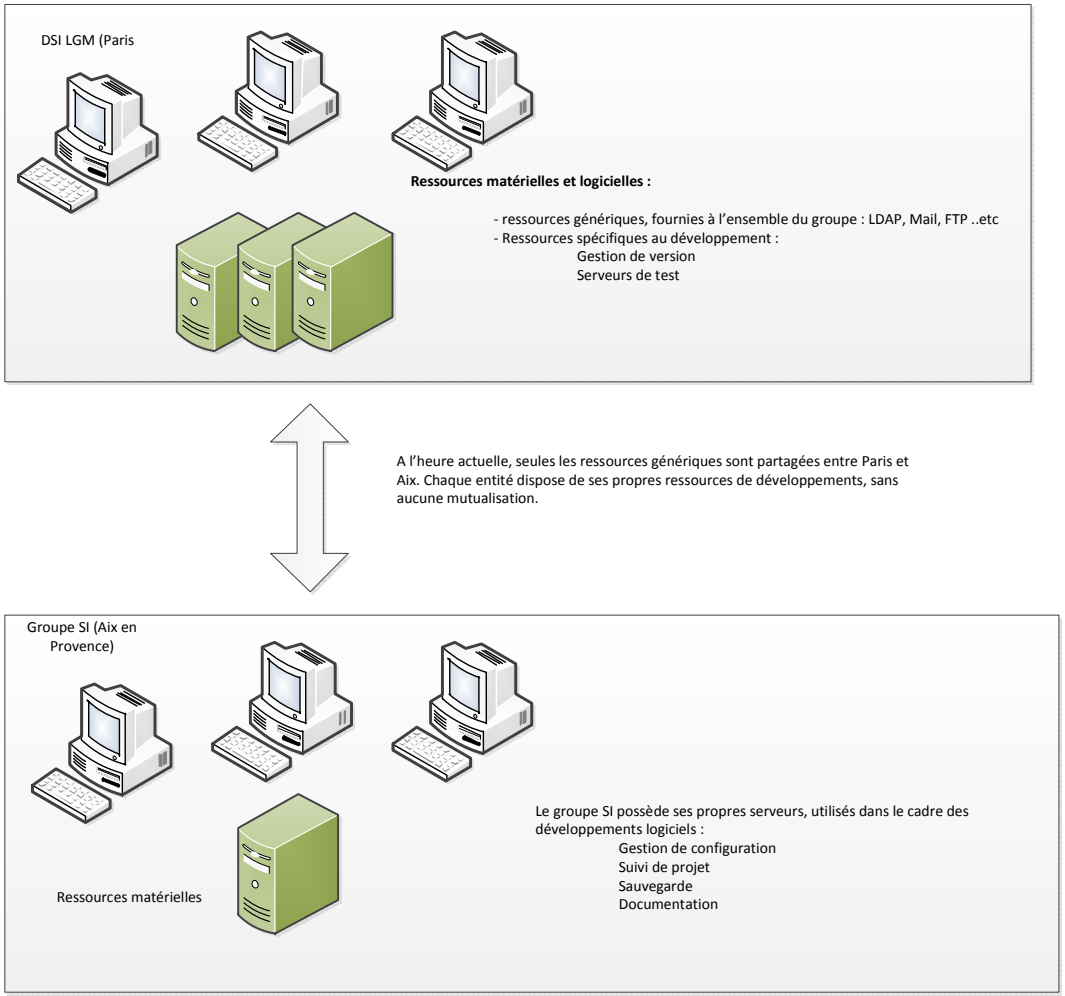


Figure 9 : Intégration du groupe SI au sein du groupe LGM

8 ANALYSE DE L'EXISTANT

L'objectif de cette étude est donc de proposer des solutions concrètes permettant à moyen terme d'améliorer la qualité de nos outils. Il convient de rendre notre activité plus performante, tout en limitant les coûts supplémentaires et la charge de travail.

Dans cette optique, j'ai choisi d'utiliser le niveau deux de la certification CMMI qui implique une gestion des exigences, une planification et un suivi de projet, une assurance qualité et une gestion de la configuration.

CMMI, sigle de Capability Maturity Model + Integration, est un modèle de référence, un ensemble structuré de bonnes pratiques, destiné à appréhender, évaluer et améliorer les activités des entreprises d'ingénierie.

CMMI a été développé par le Software Engineering Institute de l'université Carnegie Mellon, initialement pour appréhender et mesurer la qualité des services rendus par les fournisseurs de logiciels informatiques du Département de la Défense US (DoD). Il est maintenant largement employé par les entreprises d'ingénierie informatique, les Directeurs des systèmes informatiques et les industriels pour évaluer et améliorer leurs propres développements de produits *(wikipedia).*

Cette méthode largement éprouvée permet de définir la maturité d'un projet ou d'une organisation au travers de cinq niveaux :

- **Niveau 1** : Premier niveau. Lors de cette première étape, aucune organisation claire n'est définie. L'obtention des résultats repose quasi exclusivement sur la motivation et le savoir-faire de l'équipe de développement. La documentation est inexistante, et l'évaluation de l'efficacité des performances est impossible.

- **Niveau 2** : Ce second niveau met en place les premières briques d'une gestion des processus. Les principales caractéristiques applicables à notre groupe sont :
  - Gestion des exigences
  - Planification de projet
  - Suivi de projet
  - Gestion de la configuration
  - Assurance qualité
- **Niveau 3** : A ce niveau, l'organisation dispose de processus standards ajustés pour chaque projet. Les processus sont standardisés, et contrôlés par un processus externe. Ce niveau met également en place des boucles d'améliorations.

- **Niveau 4** : Ce niveau est la continuité du troisième niveau auquel il ajoute un contrôle statistique des processus en temps réel. Ces contrôles mettent en avant les résultats obtenus ainsi que les axes d'améliorations.

- **Niveau 5** : Dernier niveau CMMI. L'organisation est dans une boucle permanente d'optimisation. Deux domaines de processus caractérisent ce niveau.

    - Innovation organisationnelle
    - Analyse causale et solution des problèmes

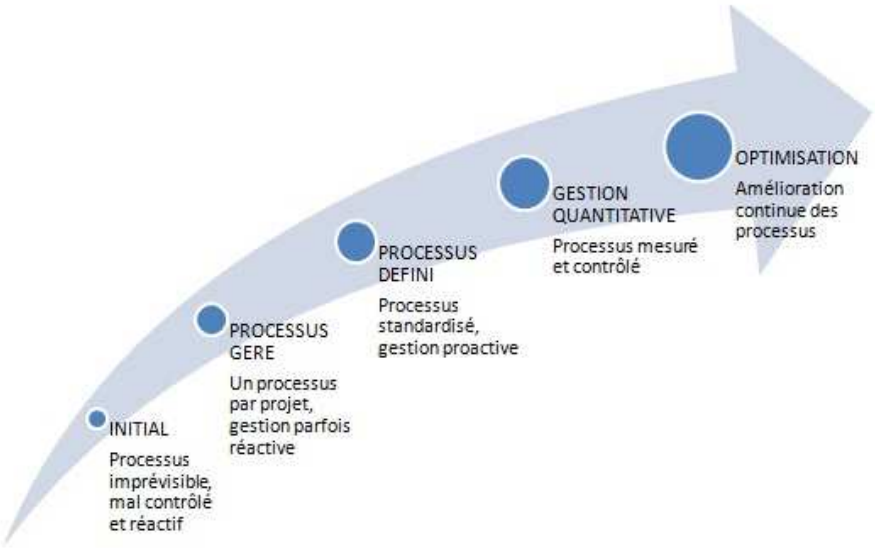


Figure 10 : Le modèle CMMI

CMMI sera utilisé ici non pas comme une méthode qu'il faudrait suivre à la lettre, mais comme un fil conducteur de mon analyse, et des propositions qui en découleront.

L'amélioration des processus est un vaste domaine. Il existe beaucoup de littérature sur ce sujet. Beaucoup de solutions théoriques ont été présentées. Nous allons essayer de mettre en pratique celles que nous jugeons les plus intéressantes. Il ne faut cependant pas perdre de vue que l'amélioration des méthodes est un travail de longue haleine, dans lequel l'expérience personnelle des différents acteurs joue un rôle prépondérant.

L'enjeu est donc de décliner ces méthodes d'améliorations de processus trop théoriques en solutions concrètes, adaptées à nos spécificités. Le but ici n'est pas de suivre à la lettre un ensemble de bonnes pratiques et de recommandations mais de s'appuyer sur ces dernières pour trouver notre propre dynamique d'optimisation, inspirée de méthodes existantes, et adaptée à nos spécificités. Améliorer et optimiser nos points faibles, tout en conservant ce qui fait notre force requiert une bonne connaissance du métier.

8.1 ÉTUDE DE L'EXISTANT AU SEIN DU GROUPE SI

Ce travail vise à mieux comprendre les processus au sein de notre groupe. Partant de cela, nous pourrons ensuite déterminer ceux que nous devons conserver, et ceux que nous devons améliorer. Cette analyse se fera notamment par l'étude de deux projets de développements que nous avons menés en 2010, 2011 et 2012. Le premier projet, DS3, est en phase de production depuis mars 2011, le second, ILSDB, est en phase de qualification depuis mars 2012.

Nous allons faire dans cette partie une analyse de l'existant. Cette analyse a pour but :

- De faire un bilan de nos activités de développement
- De mieux comprendre les attentes de nos clients
- D'évaluer la performance.

8.2 PRÉSENTATION DES PROJETS ÉTUDIÉS

8.2.1 LE PROJET DS3

NH Industries est une coentreprise européenne chargée de la coordination du Programme NH90. Elle a été établie en 1992 par Eurocopter (Allemagne et France), Agusta (Italie) et Stork Fokker Aerospace(Pays-Bas). NHIndustries a été spécifiquement établie pour coordonner le design, le développement, l'industrialisation, la production et la logistique pour la série d'hélicoptères NHI NH90 avec l'agence de l'OTAN chargée de ce programme, la NATO Helicopter Management Agency.

Le NH90 est un hélicoptère militaire biturbine de la classe des 10,6 tonnes, multifonctions, développé pour atteindre des fonctionnalités navales et tactiques pour le transport notamment et voulu initialement par l'Allemagne, la France, l'Italie et les Pays-Bas.

NHI en tant que gestionnaire des programmes internationaux NH90 assure la gestion des catalogues de pièces de rechanges et outillages pouvant être approvisionnés pour ses clients.

Les quatre compagnies partenaires ("Partners Companies" ou PC) Eurocopter France, Eurocopter Deutschland, Agusta/Westland, Fokker) engagées dans le programme NH90, ont entamé une démarche d'élaboration d'un catalogue de pièces incluant un tarif unique ("all consumers price list", ACPL), figé sur une durée d'un an. Le prix de vente pour un client donné ("customer price list", CPL) est alors calculé à partir du prix ACPL, pondéré d'un ratio spécifique harmonisé entre les partenaires. Ce calcul des prix CPL est effectué par NHI pour chaque contrat, et pour la liste de pièces dont il assure la vente. Ce calcul est basé sur divers indicateurs (Cours de la monnaie, durée d'approvisionnement, délai de paiement ...etc).

Les catalogues ACPL et CPL sont gérés sous forme de fichiers Excel par NHI. Les difficultés rencontrées par NHI dans la gestion des catalogues sont liées aux différentes configurations d'hélicoptères (plus de 20 variantes

différentes pour 560 appareils commandés par 14 clients), à l'évolution des pièces (nouvelles références et nouvelles caractéristiques) ou des prix, ainsi qu'à la complexité de l'organisation du programme NH90.

De plus, les catalogues de pièces devenant chaque jour plus volumineux (près de 10 000 références par catalogue client), leur gestion sous Excel présente de nombreux problèmes de performance et de fiabilité.

Une base de données référençant l'ensemble des prix ACPL et CPL avait été élaborée sous MS Access par la société LGM en 2009. NHI a décidé de lancer en 2010 un projet d'amélioration et de portage de l'application vers un système de gestion de base de données.

L'application résultante (DS3) doit faciliter la gestion des catalogues de pièces de rechanges et d'outillages, garantir une traçabilité des versions de catalogues par client et offrir des fonctionnalités avancées de mise en cohérence des catalogues CPL avec le catalogue ACPL courant et de calcul de prix CPL à partir des prix ACPL

Ce projet est le premier projet sur lequel j'ai travaillé lors de mon arrivée chez LGM. Initialement, ce projet devait se dérouler en deux phases :

- Analyse de l'application existante, correction des bugs, et ajouts de fonctionnalités supplémentaires.
- Création d'un modèle de données et portage de la base de données vers SQL Server, avec une interface web Java (Struts 2/Jquery)

Après une première analyse, il a été décidé de ne pas faire évoluer la première application, écrite en Visual basic (Ms Access), mais de passer directement à une application client-serveur. Nous avons néanmoins conservé, en la faisant évoluer, la spécification du besoin initial

Ce projet était un projet pilote, dans la mesure où il s'agissait du premier projet de développement entièrement réalisé par le groupe SI, de la phase de spécification à la phase de réalisation et de production.

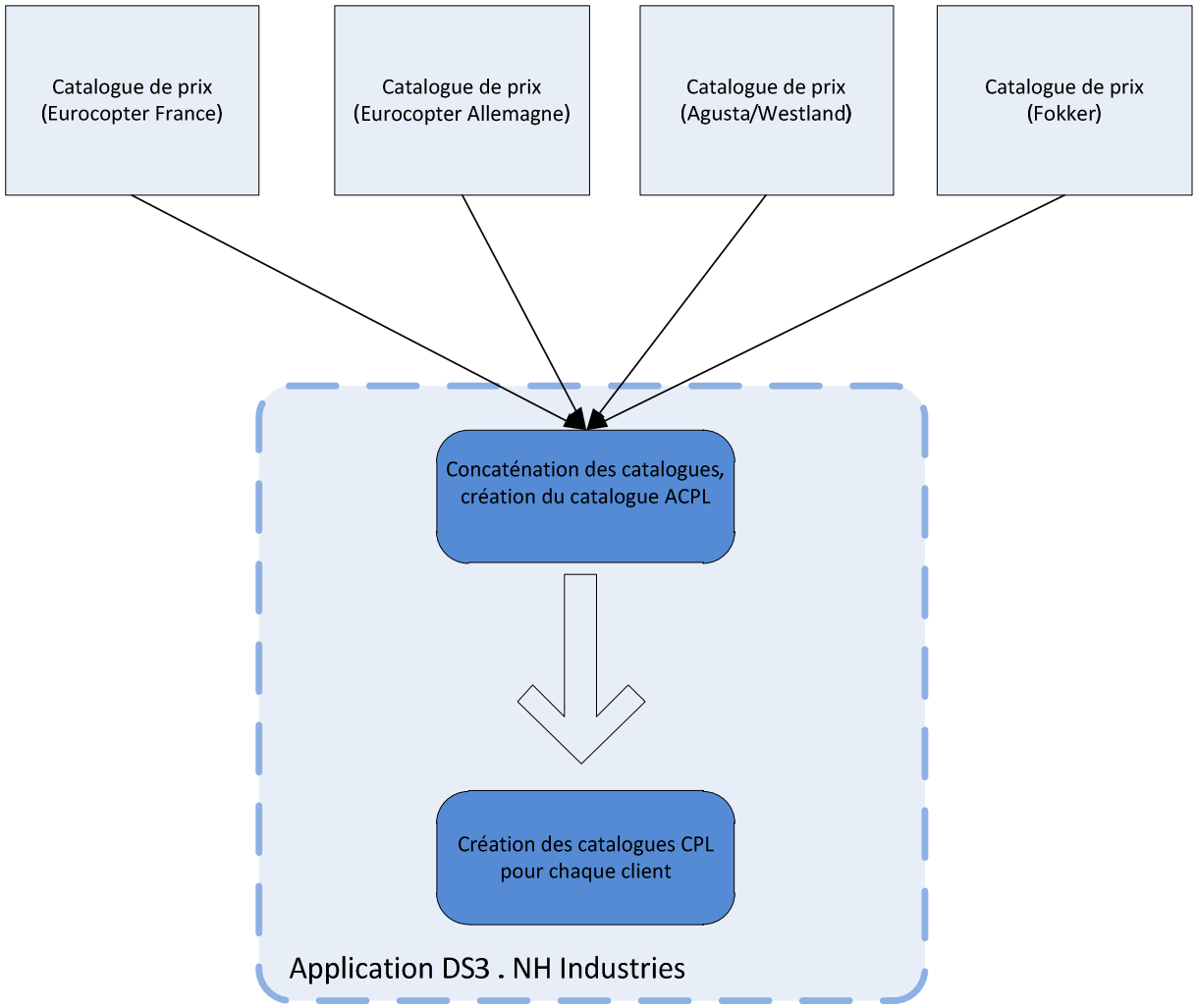


Figure 11 : Périmètre de l'application DS3

8.2.2 LE PROJET ILSDB

L'EC175 est un nouvel hélicoptère de transport civil de 7 tonnes développé conjointement par EUROCOPTER et Harbin Aviation Industry (Chine).

Dans le cadre de ce nouveau développement, EUROCOPTER souhaite mettre en place des outils garantissant la traçabilité et la cohérence des données de « configuration appareil » avec les différentes données « Support » (LSA, rechanges, documentation technique) et ceci dès la phase de développement de l'appareil.

EUROCOPTER souhaite réaliser un prototype opérationnel de cet outil informatique dédié au management des données Support afin :

- de valider les process, les méthodes de travail et les liens à créer entre les différentes bases de données.
- de produire des données Support cohérentes sur le programme EC175 en attendant la mise en place d'une solution industrielle EUROCOPTER.

Concrètement, les principales fonctionnalités de l'application sont les suivantes :

- Import des données provenant du LSA.
- Import des données de gestion de configuration.
- Création de lien entre les données (par exemple, création d'un lien entre une tâche de maintenance et la carte de travail associée).
- Création de requêtes de cohérence et export de données (par exemple, vérification des effectivités).

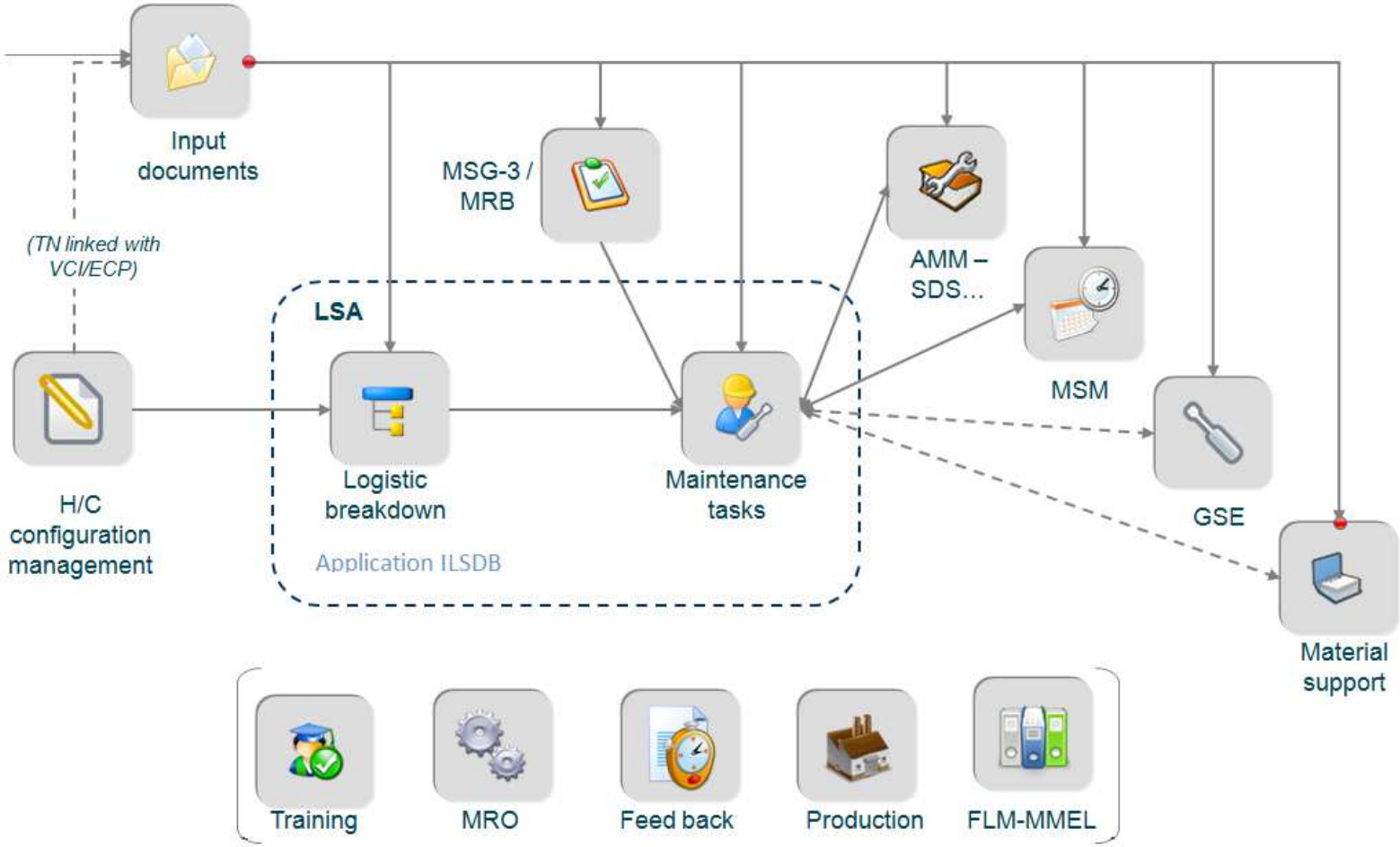


Figure 12 : Périmètre de l'application ILSDB

8.3 PORTÉE DE L'ANALYSE

Préalablement à toute analyse, il convient de définir ce que nous allons étudier. Dans notre cas, nous avons défini quatre axes de recherches :

- Processus de développements logiciels
- Interaction du groupe SI avec les Pôles métiers LGM
- Relation client / activités commerciales

8.4 ANALYSE DES PROCESSUS DE DÉVELOPPEMENT

Nous allons dans cette partie analyser les processus de développement mis en place lors de la réalisation des projets DS3 et ILSDB. Comme je l'ai expliqué en introduction, j'ai choisi d'utiliser la méthode CMMI.

Nous allons faire ici un parallèle entre les critères CMMI et les deux projets présentés ci-dessus. Nous conclurons ensuite en déterminant sur quel niveau CMMI se trouve le groupe SI, et le niveau optimum vers lequel nous voulons tendre.

Nous étudierons les points suivants :

- **Gestion des exigences :** La gestion des exigences consiste à gérer les différentes exigences d'un projet. Ces exigences peuvent être de plusieurs natures :

  - Exigences utilisateurs
  - Exigences métiers
  - Exigences Techniques

  Gérer les exigences d'un projet revient à trier, documenter, et répondre aux différentes demandes.

- **Planification du projet :** La planification consiste à définir les jalons qui rythmeront la vie du projet, de la phase de conception à la phase de production. Nous distinguons trois types de planification :

  - *Définition des jalons principaux :* Définition avec le client final des grandes phases du projet

  - *Gestion des tâches entre les différents acteurs du projet :* Cette étape consiste à coordonner les différents métiers LGM intervenant dans le projet. Par exemple, rédaction des spécifications par l'expert métier, rédaction des spécifications techniques par le chef de projet informatique…etc

  - *Organisation interne groupe SI :* Lors de cette dernière étape, le responsable de projet du groupe SI définit les différentes tâches du développement.

- **Suivi de projet :** Le suivi de projet est utilisé durant la phase de réalisation pour contrôler les actions en cours, et déterminer le cas échéant les écarts par rapport au planning initial.

- **Gestion de la configuration :** La gestion de configuration est un ensemble de pratiques visant à suivre au plus près l'évolution du code source de l'application et la documentation associée.

- **Assurance qualité :** dans notre cas l'assurance qualité consiste à mettre en place un ensemble de méthodes visant à s'assurer que le logiciel est conforme aux exigences et exempt de bugs.

8.4.1 ANALYSE DU PROJET DS3

Le projet DS3 a été pour nous riche d'enseignements. Il s'agit du premier projet de développement porté par le groupe SI. Après avoir connu quelques difficultés, cet outil est en phase de production depuis mai 2011. Suite à une période de prise en main le bilan de fin de projet a eu lieu en septembre 2011. Nous allons dans cette partie analyser les différents aspects de ce projet.

8.4.1.1 GESTION DES EXIGENCES

La gestion des exigences s'est faite au cours de réunions avec le client. Ces réunions ont conduit à la réalisation d'une spécification détaillée. Aucune méthode, ni aucun outil de gestion des exigences n'a été utilisé. Nous nous sommes basés sur la spécification de l'outil Access, les besoins n'ayant pas évolué entre ces deux projets.

8.4.1.2 PLANIFICATION DE PROJET ET PLAN DE DEVELOPPEMENT

Un planning général de déroulement de la prestation a été présenté au client lors de la phase de lancement du projet. Nous n'avons réalisé aucun planning interne entre le pôle Aéronautique, chargé de la partie métier, et le groupe SI, chargé de la réalisation. Du fait de la faible taille du projet, aucun plan de développement spécifique n'a été utilisé, ce qui a entrainé une mauvaise estimation du temps nécessaire aux activités de tests et d'intégration sur la plate-forme client.

8.4.1.3 SUIVI DE PROJET

Mis à part la réalisation de compte rendu d'activités mensuelles, nous n'avions pas mis en place d'outils de suivi des activités de développement. Nous avions notamment mal évalué l'avancement réel par rapport au planning initial.

L'applicatif était composé de trois modules distincts :

- import de catalogues ACPL (All cutomers price list)
- import de catalogue CPL (customers price list), catalogues spécifiques à un contrat
- calcul de prix pour les catalogues CPL.

8.4.1.4 GESTION DE LA CONFIGURATION

Durant les premiers mois de la prestation, il n'y avait qu'un seul développeur sur le projet. De ce fait, hormis des sauvegardes quotidiennes, aucune gestion de configuration n'a été mise en place. La gestion de configuration n'a été organisée que lors de l'arrivée d'un second développeur. Cette gestion a consisté en l'établissement d'un serveur SVN permettant le développement concurrent.

8.4.1.5 ASSURANCE QUALITÉ

Aucune politique particulière d'assurance qualité n'a été mise en place. Nous avons suivi un cycle itératif simple. Il y a eu très peu de documentation interne (Javadoc par exemple). Seules deux documentations ont été fournies au client final :

- Documentation administrateur : guide de déploiement du logiciel, destiné aux administrateurs réseaux.
- Document utilisateur : guide d'utilisation du logiciel

8.4.2 RETOUR D'EXPÉRIENCE DU PROJET DS3

Nous allons analyser ici les succès et les difficultés rencontrées lors du développement de cet outil.

8.4.2.1 POINTS NÉGATIFS :

- ***Gestion des exigences*** *:* La spécification était trop générale et ne détaillait pas certains points. Une méconnaissance des métiers par l'équipe de développement a amené des incompréhensions, et donc des retards dans la réalisation. Nous avons eu des difficultés à trouver un langage commun entre les utilisateurs finaux et l'équipe informatique. Ce manque de spécification a entraîné le développement de nouvelles fonctionnalités non prévues initialement.

- ***Planification des développements :*** Nous avons sous-évalué la difficulté technique du travail. Cela est dû d'une part au manque d'expérience de l'équipe, et d'autre part à une technologie trop récente, encore en phase de développement. Cette mauvaise planification a entrainé un retard conséquent sur la livraison, ainsi que des pics de charge très importants pour l'équipe. Nous nous sommes rendu compte beaucoup trop tard de ces dérives dues au manque de planification.

- ***Assurance qualité :*** Nos plans de tests n'ont pas été assez rigoureux. De trop nombreux bugs ont empêché la recette de l'application finale. Nous avons notamment sous-estimé la phase de traitement des données d'entrées. Nous n'avons pas évalué le fait que les fichiers d'entrées proviennent de quatre pays différents, utilisant chacun leurs propres formats.

8.4.2.2 POINTS POSITIFS :

- ***Satisfaction du client :*** Malgré un certain nombre de difficultés, le client est satisfait du résultat final. L'application correspond bien à ses attentes, tant du point de vue fonctionnel que technique.

- ***Adéquation avec le besoin :*** Après des difficultés initiales, nous avons pu, en liaison avec les référents métiers LGM définir clairement le besoin. Les difficultés initiales ont été effacées par une réactivité immédiate des équipes de développement.

- ***Facilité d'utilisation :*** Même si la technologie choisie a posé des problèmes, les performances et l'ergonomie sont sans commune mesure avec l'outil précédent (formulaires Access).

D'une manière plus générale, les reproches qui nous ont été faits portent non pas sur l'aspect technique et fonctionnel, mais sur l'aspect organisationnel.

Si l'on fait un parallèle entre ce projet et la méthode CMMI, nous pouvons dire que, au sortir de ce projet, nous sommes à l'étape un, communément appelée état initial ou chaotique. L'effort individuel prévaut sur l'effort collectif. La réussite n'est pas nécessairement reproductible, et il n'y a aucune évaluation de l'efficacité.

8.4.3 EVOLUTIONS ORGANISATIONNELLES ENTRE LES PROJETS DS3 ET ILSDB

Suite à ce projet, de nombreux changements, visant à améliorer nos développements ont été effectués. Les principaux travaux ont porté sur les axes suivants :

8.4.3.1 PLANIFICATION ET SUIVI DE PROJET :

L'accent sera mis sur la partie planification. Un planning précis aux normes LGM est fourni par le responsable métier. Ce planning est ensuite transposé en plan de développement. Ce plan de développement précis est désormais validé conjointement par le chef de projet métier et le chef de projet SI, avant d'être transposé en tâches bien définies (temps nécessaire, description…etc.). Ces tâches sont ensuite affectées à l'équipe de développement qui doit renseigner en retour le temps réel effectué et les difficultés rencontrées. Nous utilisons l'outil de suivi des tâches Redmine.

8.4.3.2 IMPLICATION DES RESPONSABLES MÉTIERS :

Les responsables métiers seront désormais impliqués non plus pendant les phases de tests, mais également durant les grands jalons du développement. Le but étant d'éviter l'effet « tunnel » : longue période sans échéance, avec risques de non adéquation au besoin.

8.4.3.3 GESTION DE LA CONFIGURATION :

Outre le code source du logiciel, tous les éléments constitutifs de l'application seront gérés via le serveur SVN : fichier de configuration, scripts de base de données, version de tests …etc.

8.4.4 ANALYSE DU PROJET ILSDB

8.4.4.1 GESTION DES EXIGENCES

La gestion des exigences a été menée conjointement par le responsable métier et le responsable client. Sur ce projet LGM intervient à la fois sur la partie analyse du besoin (pôle Aéronautique) et sur la partie réalisation de l'outil (Groupe SI).

Le travail de spécification a été beaucoup plus approfondi que sur le projet DS3. La base de données a été modélisée en utilisant des outils spécifiques. Le modèle de base a été présenté et validé durant la phase de spécification.

8.4.4.2 PLANIFICATION DE PROJET

Les principaux jalons du projet ont été définis conjointement par le client et le responsable métier. Ces jalons ont été déclinés en plan de développement, lui-même validé conjointement par le chef de projet métier, le chef de projet SI et le client final.

8.4.4.3 SUIVI DE PROJET

Le projet a été suivi à l'aide de l'outil Redmine (cf. schéma ci-dessous). Le plan de développement a été divisé en tâches unitaires, présentées sous la forme d'un diagramme de Gant. Le développeur visualise dans son environnement Eclipse les tâches qui lui sont affectées, et renseigne le temps qu'il a mis pour réaliser le travail. Le diagramme de Gant est mis à jour en direct.

8.4.4.4 GESTION DE LA CONFIGURATION

La gestion de la configuration a été effectuée via le serveur Subversion sur une branche unique. Tous les développeurs travaillent sur un répertoire centralisé et sauvegardent leur travail sur un serveur commun. Ce serveur est lui-même sauvegardé quotidiennement.

8.4.4.5 ASSURANCE QUALITÉ

Nous avons défini, en liaison avec les utilisateurs finaux des plans de tests destinés à qualifier l'application. Ces plans de tests ont été utilisés en phase de qualification.

8.4.5 RETOUR D'EXPÉRIENCE DU PROJET ILSDB

Ce projet, qui a débuté en septembre 2012 est toujours en phase active de développement. La première version est en phase de qualification depuis mars 2012. Les développements de la seconde version vont débuter prochainement. Ce projet est intéressant à plusieurs titres :

- ***Pluridisciplinarité :*** L'outil développé couvre plusieurs disciplines et métiers différents. Les difficultés sont d'autant plus grandes.

- ***Nouveauté :*** Ce projet a été l'occasion de monter en compétences sur des nouvelles technologies (JSF).

8.4.5.1 POINTS NÉGATIFS

- ***Plan de développement :*** Le temps nécessaire à la réalisation de l'outil a été sous évalué. Cette erreur est due, d'une part, à une mauvaise estimation des risques liés à la technologie, et d'autre part, à un manque de connaissance métier de l'équipe de développement.

- ***Gestion de projet*** **:** L'outil de gestion des tâches intégré à Redmine s'est révélé peu pratique à l'usage. Il n'existe pas de lien natif entre Microsoft Project et Redmine. Chaque modification sur le planning initial, implique un changement manuel du plan de développement. Cette synchronisation constante des tâches est complexe.

- ***Estimation de la charge :*** La charge de travail a été mal évaluée lors du chiffrage initial. A cette charge initiale se sont rajoutées de nombreuses difficultés techniques liées à la technologie utilisée.

- ***Sous-estimation de la phase de spécification :*** Le besoin étant très complexe, la phase d'analyse du besoin et de rédaction des spécifications a pris beaucoup plus de temps que prévu

8.4.5.2 POINTS POSITIFS

- ***Montée en compétence technique et fonctionnelle :*** Ce projet a permis à l'équipe de monter en compétence sur le plan fonctionnel (gestion de configuration, support) et sur le plan technique. ILSDB est le premier projet du groupe SI utilisant le Framework JSF et la bibliothèque de composants Primeface.

- ***Montée en compétence de notre organisation :*** Nous avons rencontré des problèmes de gestion, cependant la plate-forme de développement mise en place semble prometteuse.

Nous pouvons tirer plusieurs enseignements de ce projet :

- La principale difficulté rencontrée est la gestion de projet. Nous avons encore beaucoup de points à améliorer. Même si les choses ont sérieusement progressé depuis le projet DS3, les plannings ne sont pas tenus, et nous n'avons pas pu éviter les pics de charge. Ce point est l'axe majeur à travailler.

- La synergie entre les équipes métiers et informatique fonctionne bien. Nous avons réussi à mettre en commun nos savoir-faire pour apporter une réponse au besoin de notre client. Cette synergie est le principal atout de notre offre informatique.

Si nous mettons en parallèle ce projet avec le référentiel CMMI, nous pouvons conclure que nous avons franchi la première étape. Nous avons mis en place les fondements de l'étape 2 :

- ***Gestion des exigences*** **:** Réalisée lors de la phase de spécification.

- ***Planification et suivi de projet :*** Ces deux points mis en place lors de la phase de démarrage, sont difficiles à suivre au quotidien. Lors de la phase de démarrage du projet, le plan de développement a été défini par le chef de projet, et les tâches correspondantes ont été saisies dans Redmine. Cependant, nous nous sommes rapidement rendu compte que le plan de développement initial ne les couvrait pas toutes. Par exemple, la tâche de création de la connexion avec le serveur LDAP, qui nous a pris environ trois jours n'était pas définie. Nous avons donc dû modifier au fur et à mesure le travail affecté aux différentes personnes. Ce fonctionnement n'est pas valable. Dans un fonctionnement normal, c'est au développeur de s'adapter au plan de développement, et non l'inverse.

Une autre de nos erreurs a été de vouloir suivre au plus près le planning général de déroulement du projet, crée sous Microsoft Project. En théorie, Redmine peut importer des plannings « Project » et modifier dynamiquement les dates de début et de fin théoriques des tâches. Cette fonctionnalité est complexe à utiliser, et présente de nombreux bug. Le changement automatique des dates est par exemple très mal géré.

Ces disfonctionnements ont produit l'effet inverse de ce que nous attendions. L'outil de gestion des tâches a occasionné une surcharge de travail qui néanmoins, a été compensée par l'amélioration de la qualité du résultat, puisque tous les modules de notre outil sont à présent documentés.

Il faut donc recentrer l'utilisation que nous faisons de l'outil de suivi des activités. Deux solutions s'offrent à nous :

- ***Améliorer la liaison avec Microsoft Project*** : Améliorer la fonction d'import et de modification des tâches pour que notre outil soit une réplique du planning général.

- ***Abandonner la liaison avec Microsoft Project :*** Il s'agirait ici de recentrer l'outil Redmine sur la gestion des développements et de la documentation associée. Cet outil serait utilisé pour suivre uniquement les activités de développements, en complément du planning général. Par exemple, si le planning général prévoit un temps de développements de cent jours, Redmine sera utilisé uniquement pour détailler et suivre l'activité durant cette période ; le but étant de pouvoir consolider le planning général à l'aide de l'avancement réel constaté dans Redmine.

  Nous étudierons ces deux possibilités dans le chapitre neuf de ce document.

- ***Gestion de la configuration :*** Réalisée à l'aide de Redmine et SVN. Nous n'utilisons pas le concept de branches, et nous avons rencontré des problèmes lors de la fusion des sources. (merge)

- ***Assurance qualité :*** Documentation technique réalisée au format Javadoc. Plan de tests sommaires.

8.4.6 BILAN ET CONCLUSION

Nous allons dans cette dernière partie résumer nos points forts et nos points faibles.

Points forts

- Adéquation avec le besoin et satisfaction du client : Les clients sont satisfaits de nos outils, tant d'un point de vue technique (ergonomie et facilité d'utilisation) que d'un point de vue fonctionnel (réponse au besoin initial).
- Résilience : Le groupe Si a su appréhender avec succès les changements intervenus tout au long des projets : Nouvelles technologies, nouveaux métiers...etc.

Points faibles

- Spécifications : Les phases de spécifications ont été sous-estimées, ce qui a entrainé des erreurs de compréhension et des retards.
- Suivi des activités : Une mauvaise gestion de la charge de travail a provoqué des phases de suractivité qui ont nuit à la qualité du travail et à la vie de l'équipe.
- Gestion de la qualité : Les premières versions livrées comportaient un nombre trop important de bugs.

Cette première approche du CMMI a mis en relief nos difficultés et nos atouts. Nous envisageons à terme d'utiliser la méthode d'évaluation SCAMPI (Standard CMMI Appraisal Method for Process Improvement). Cette méthode est une technique d'audit dont l'objectif principal est d'évaluer les critères spécifiques en fonction d'un niveau donné de mise en pratique.

Cette analyse de l'existant nous a permis d'identifier de manière claire et concrète nos points forts et nos points faibles. Dans le cadre de ce mémoire de fin d'étude je vais tenter d'approfondir la réflexion afin d'envisager des améliorations fonctionnelles et techniques.

9 RECHERCHE DE NOUVELLES SOLUTIONS ET OBJECTIFS DE CETTE ETUDE

A partir de l'analyse de l'existant, nous avons pu déterminer plusieurs voies d'améliorations qui seront étudiées successivement :

- Amélioration du management et de la gestion de projet
- Améliorations techniques

Quant à l'amélioration de l'offre commerciale, elle sera approfondie par les responsables du groupe SI, selon vraisemblablement les pistes suivantes :

- établir un modèle d'offre standard et réutilisable
- Réaliser un outil interne prenant en compte les facteurs de risques liés aux difficultés fonctionnelles et techniques afin d'estimer correctement le coût de réalisation d'un projet et les risques qui en découlent. Une fois cet outil réalisé nous allons le tester en chiffrant à nouveau les projets sur lesquels nous nous sommes trompés, afin de mesurer l'écart éventuel entre notre réalisation et le but à atteindre.
- Mieux faire connaître l'action du groupe SI au sein du groupe LGM, notre principale source d'activité étant le soutien aux différents pôles d'activités dans le domaine des systèmes d'informations. Ainsi le travail que nous menons aujourd'hui avec la DTI vise à définir une offre commerciale liée à nos métiers.

Les principales actions menées dans ce sens sont les suivantes :

- Présentation du groupe SI par le biais de réunions méthodes. Ces réunions mensuelles sont organisées au niveau national et ont pour but de présenter une entité, un projet ou une nouvelle technologie.
- Présentation de notre travail par des documents mis à la disposition de tous sous forme de plaquettes présentant nos différents métiers.
- Présentation concrète de nos outils à l'aide de serveurs d'applications. L'un deux contenant des versions de tests de nos différentes applications. Les utilisateurs pourront ainsi concrètement voir un aperçu de nos réalisations.

L'analyse de l'existant n'est qu'un constat. Partant de là débute le véritable travail d'investigation. Il faut disséquer chaque problème, l'expliquer et trouver des solutions adéquates ; proposer des améliorations techniques concrètes, raisonnées ; Ne pas se borner à faire une liste d'améliorations, mais trouver un fil conducteur, sur lequel viendront s'appuyer les dites améliorations. J'étudierai donc de manière plus détaillée et approfondie les axes d'améliorations de la gestion de projet et les améliorations techniques.

9.1 AMELIORATION DU MANAGEMENT ET DE LA GESTION DE PROJET

Plusieurs axes d'améliorations ont été identifiés dans le domaine de la gestion de projet. Une SSII classique se borne à développer un produit, en se fondant sur une spécification. Le groupe SI, lui, a un fonctionnement différent. L'analyse du besoin est au cœur de ses préoccupations, et chaque spécification est supervisée par un chef de projet métier et un chef de projet système d'information.

Nos principaux axes de développement dans ce domaine portent sur:

- ***Spécifications :*** Nous cherchons à standardiser nos spécifications, notamment à l'aide d'outils comme UML. L'enjeu est de trouver un langage commun entre les utilisateurs finaux et l'équipe informatique.

- ***Cycle de développement :*** Nous utilisons à l'heure actuelle un cycle itératif simple. Cette méthode de travail a montré ses limites lors du projet DS3. Nous explorons d'autres voies, par exemple, les méthodes agiles.

- ***Suivi de projets :*** La problématique du suivi de projet est la suivante : Comment faire le lien entre les jalons d'un projet et le plan de développement. Nous avons vu que la liaison entre notre outil de suivi de projet (Redmine) et l'outil utilisé par les responsables métiers (MS Project) ne donnait pas entière satisfaction. Nous réfléchissons à une solution alternative.

9.2 AMÉLIORATIONS TECHNIQUES

Nos principaux axes de développement dans ce domaine portent sur:

Assurance qualité du logiciel :

L'assurance qualité du logiciel est au cœur de nos préoccupations. La problématique est de mieux analyser les sources potentielles d'erreurs et de les corriger au fur et à mesure des travaux.

Plusieurs pistes sont envisageables :

- Analyse statique et dynamique de code source
- Méthodologie de test

Plate-forme de développement :

Le terme plate-forme de développement regroupe l'ensemble des outils utilisés lors du développement d'un logiciel. Nous étudierons le fonctionnement de la plate-forme actuelle et les modifications éventuelles à apporter.

Normes de codage :

Nous souhaitons mettre en place des normes de codage. Ces normes établiront la façon dont nos logiciels devront être réalisés.

Normes de conception :

Ces normes détermineront l'architecture standard de nos applications en fonction de la technologie utilisée.

Gestion de la configuration :

Le terme gestion de la configuration regroupe l'ensemble des processus et outils mis en place pour gérer et conserver les modifications apportées à un logiciel.

10 AMÉLIORATIONS TECHNIQUES

Nous allons traiter dans ce chapitre de toutes les améliorations techniques que nous souhaitons mettre en place lors de nos futurs développements. Nous étudierons notamment les points suivants :

- Assurance qualité du logiciel
- Plate-forme de développement
- Normes de codage
- Normes de conception
- Gestion de la configuration

10.1 ASSURANCE QUALITÉ DU LOGICIEL

10.1.1 QU'EST-CE QUE LA QUALITE DU LOGICIEL ?

Nous allons dans cette partie aborder la question de la qualité logicielle. Dans le domaine du génie logiciel la qualité logicielle est une appréciation globale d'un logiciel, basée sur de nombreux indicateurs. Les principaux facteurs de qualité sont la fiabilité, la tolérance des pannes, la simplicité et l'intégrité des informations. Nous détaillerons chacun de ces facteurs dans les chapitres suivants.

Un logiciel est un produit qui ne se détériore pas, et qui évolue constamment. Sa qualité dépend donc principalement de la manière dont il a été conçu. Cette question de qualité est donc au cœur du processus de fabrication, et fait l'objet de nombreuses études. L'étude des processus de développement a débuté avec les problèmes de qualité des logiciels, connus depuis les années 60.

Nous allons aborder dans cette partie la question de la qualité logicielle au sein de notre groupe. Nous allons analyser les problèmes rencontrés lors de nos développements logiciels et proposer des axes d'améliorations.

Ces analyses porteront sur les projets menés par le groupe SI depuis 2010 en tant que maîtrise d'œuvre. Nous n'aborderons pas les projets sur lesquels le groupe n'est intervenu qu'en tant que ressource de développement (projet IETM, langage Python).

Derrière cette question de la qualité logicielle se cache un intérêt économique. Pourquoi améliorer la qualité de nos produits ? On pourrait arguer que nos logiciels fonctionnent bien, et que, par là même il n'est pas nécessaire de dépenser du temps, de l'argent, et des ressources pour faire mieux. Je pense au contraire que derrière la gestion de la qualité se trouve le vrai savoir-faire d'une entreprise. Toutes les sociétés peuvent produire du code, mais le véritable enjeu est de produire un logiciel de meilleure qualité, et si possible plus rapidement que leurs concurrents.

10.1.2 CRITÈRES DE QUALITÉ

Nous allons dans cette partie définir des caractéristiques que nous pourrons ensuite mesurer. Il existe plusieurs normes ISO permettant d'évaluer la qualité d'un produit (ISO 9126), ou le cycle de vie d'un logiciel (ISO 12207). Selon la norme ISO 9126, les principaux critères de qualité d'un logiciel sont les suivants :

- **Capacité fonctionnelle** : la capacité fonctionnelle d'un logiciel est sa capacité à répondre au besoin pour lequel il a été conçu.
- **Fiabilité :** La fiabilité d'un logiciel est sa capacité à répondre au besoin, dans des conditions précises et durant une période donnée. Par exemple, un critère de fiabilité pourrait être le temps de réponse du logiciel, compte tenu d'une forte montée en charge.
- **Facilité d'utilisation :** La fiabilité d'un logiciel regroupe tous les critères d'ergonomie, de facilité d'utilisation et de simplicité de prise en main. Ce denier critère est très important si le produit final est destiné à des utilisateurs non informaticiens.
- **Rendement / Efficacité** : Le rendement et l'efficacité regroupent les notions de coût d'exploitation et de rendement. Par exemple un critère d'efficacité pourrait être le coût annuel d'exploitation en fonction du nombre d'utilisateurs.
- **Maintenabilité** : La maintenabilité d'un logiciel est sa capacité à évoluer en même temps que le besoin.
- **Portabilité :** La portabilité d'un logiciel est sa capacité à pouvoir être déployé sur des environnements différents.

10.1.3 CRITERES DE QUALITE APPLICABLES AU GROUPE SI

Le groupe SI développe et déploie des solutions logicielles dans le domaine des systèmes d'informations. Le management de ces systèmes est une discipline de l'informatique bien particulière qui regroupe l'ensemble des connaissances et des techniques assurant la gestion des données.

Nous allons dans cette partie décliner les critères de qualité de la norme 9126 applicables à cette discipline.

- **Capacité fonctionnelle** : En informatique de gestion la capacité d'un logiciel à répondre à un besoin est définie principalement par le travail préalable d'analyse et de rédaction des spécifications. Ce travail de spécification est développé dans la première partie de ce mémoire.

- **Fiabilité :** Tous nos outils, mêmes si les technologies sont différentes, utilisent une base de données, sur laquelle vient se connecter une interface, qui peut être une application web, un client lourd, une application mobile, etc. La fiabilité se décline donc selon deux axes :

    - *Fiabilité de la base de données :* Elle est le cœur de l'application, c'est elle qui contient toute les données. Cette base est la clé de voûte du système d'information. Sa fiabilité est liée à sa conception, au Système de gestion de données choisi et aux capacités techniques du serveur hébergeant la base

    - *Fiabilité de l'applicatif:* La fiabilité de l'interface d'accès aux données est primordiale. Cette interface permet aux utilisateurs d'accéder aux données de la base et de les modifier.

- **Facilité d'utilisation :** La facilité d'utilisation est un critère primordial en informatique de gestion. Les logiciels que nous concevons vont être utilisés, pour leur majorité, par des personnes dont l'informatique n'est pas le métier. Il faut donc mettre l'accent sur l'ergonomie et la documentation de l'utilisateur ; domaine qui rejoint celui du graphisme. Nous avons choisi d'utiliser des librairies graphiques pour gérer ces aspects.

- **Rendement et efficacité :** En informatique de gestion, le rendement pourrait se définir par le coût de la plate-forme en fonction du nombre d'utilisateurs pendant une période donnée. Ce calcul comprend le coût de la maintenance corrective, le coût de maintenance technique (réseau, serveurs).

- **Maintenabilité :** Ce critère est également très important dans le domaine des systèmes d'informations. Le besoin évolue rapidement, le logiciel doit être facilement adaptable aux demandes des utilisateurs.

- **Portabilité** : La portabilité n'est pas un critère déterminant dans notre cas. Même si les technologies utilisées pourraient être déployées dans d'autres environnements, les solutions logicielles sont conçues pour s'adapter au système d'information de l'entreprise dont l'évolution est relativement lente.

J'ai choisi dans cette analyse de la qualité de n'aborder que les problématiques de fiabilité, de maintenabilité et d'efficacité. Les problèmes de capacité fonctionnelle ont été abordés dans la première partie de ce document.

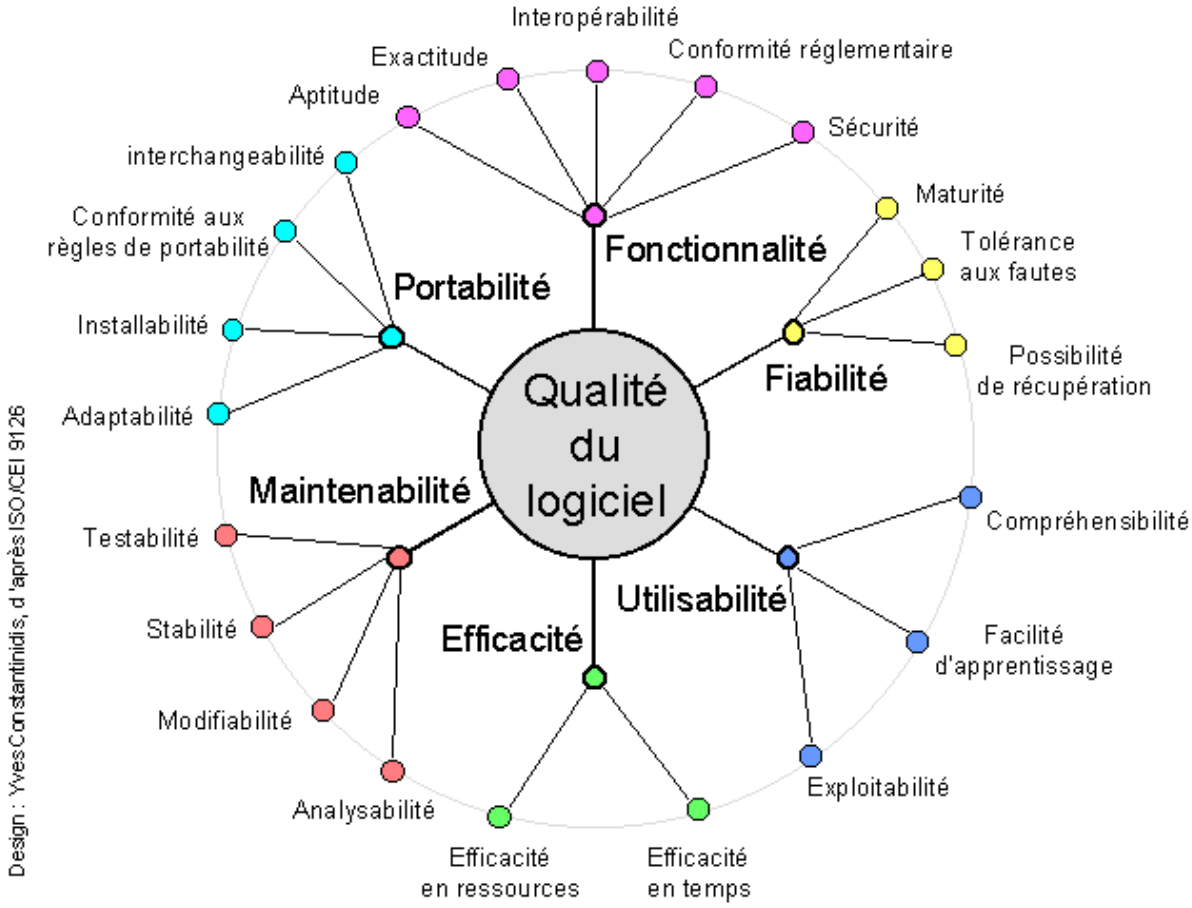


Figure 13 : La norme ISO 9126

10.1.4 FIABILITÉ

10.1.4.1 INTRODUCTION

Nous allons dans cette partie aborder les questions de fiabilité du logiciel. Nous nous sommes rendu compte que nos produits, lors de leurs déploiements initiaux, comportaient un trop grand nombre de bugs. Cette mauvaise qualité initiale entraîne des temps de corrections importants, donc du retard, et une insatisfaction du client qui a parfois l'impression d'utiliser un produit inachevé. Ces retards signifient également des coûts supplémentaires importants, donc une forte baisse de la rentabilité.

Partant de ce constat d'échec, nous avons voulu analyser précisément les problèmes. Cette analyse nous a conduits à établir une classification des bugs suivant leur type et leur importance.

Il n'existe pas de méthode formelle permettant de détecter et corriger les bugs, ces derniers étant par définition aléatoires. Il existe cependant un certain nombre de techniques permettant de détecter les failles potentielles d'un logiciel.

Lors de la réalisation d'un logiciel, les erreurs peuvent être détectées pendant la phase de réalisation ou pendant la phase de tests. Une détection pendant la phase de tests implique une correction du code, puis de nouveaux tests. On voit donc que l'enjeu est de détecter le plus de dysfonctionnements possibles durant la phase de création de l'application.

Nous distinguons deux types de bugs :

- **Bugs fonctionnels :** Ce ne sont pas à proprement parler des erreurs. L'application fonctionne correctement, mais ne correspond pas aux besoins du client ou à la spécification initiale. Plusieurs causes peuvent être identifiées :
    - Mauvaise compréhension du besoin : La demande du client n'a pas été comprise par l'équipe chargée de l'analyse initiale.
    - Mauvaise expression du besoin : Ce besoin a été compris, mais mal exprimé dans la spécification.

  Ces bugs peuvent être contournés par une amélioration de la phase de spécification et d'analyse du besoin.

- **Bugs techniques** : Ils regroupent l'ensemble des erreurs purement techniques. Ces bugs ont des origines diverses : erreur du développeur, bug d'une librairie… etc

Nous n'analyserons dans ce chapitre que les bugs techniques. Les bugs fonctionnels seront étudiés dans le chapitre neuf de ce document.

10.1.4.2 CLASSIFICATION ET IMPORTANCE DES BUGS TECHNIQUES

Nous avons regroupé les problèmes techniques en deux familles :

- **Bugs liés à l'application :** Cette famille regroupe toutes les erreurs de conception du logiciel. Ces erreurs entraînent un dysfonctionnement de l'application, qui est classifié en fonction de sa gravité. Par exemple, une fonction qui renvoie une erreur.

- **Bugs liés à la base de données :** Cette famille regroupe toutes les erreurs liées à la base de données. Par exemple, une procédure stockée qui ne fonctionne pas.

Parmi ces familles de bugs, nous différencions trois niveaux de gravité :

- **Type 1** : Bug majeur empêchant la mise en place d'une fonctionnalité vitale de l'application qui ne peut pas être livrée en l'état : le bug doit donc être corrigé en priorité. Si celui-ci n'a pas été détecté pendant la phase de test alors que l'application est en production, son utilisation est immédiatement suspendue. Par exemple dans le cas du projet DS3, un calcul de prix erroné.

- **Type 2 :** Bug non bloquant, pour lequel une solution de contournement temporaire existe. Par exemple dans le cas du projet ILSDB, un écran de recherche ne fonctionnant pas. Ce bug doit être corrigé rapidement, mais n'empêche pas le bon fonctionnement de l'application.

- **Type 3** : Bug mineur. Cette famille regroupe les incidents mineurs, par exemple une erreur d'orthographe sur un libellé.

L'importance du bug est déterminée par le maître d'ouvrage, et peut entraîner des pénalités financières.

10.1.4.3 DETECTION ET CORRECTION DES BUGS APPLICATIFS

Pour ce faire, l'industrie du développement utilise depuis de nombreuses années des outils d'analyse statique de code source.

L'analyse statique du code source consiste en une série de méthodes utilisées pour analyser le comportement d'un logiciel sans réellement l'exécuter. Ces outils sont utilisés pour mettre en lumière des erreurs de conceptions pouvant entrainer des erreurs d'exécution.

Les principaux bugs rencontrés sont :

- **Erreur de segmentation :** Le logiciel essaie d'écrire dans un emplacement mémoire qui n'existe pas ou qui est utilisé par un autre programme.
- **Dépassement de mémoire :** Un logiciel tente de stocker des informations ou le résultat d'un calcul dans un espace mémoire insuffisant.
- **Une fuite de mémoire :** Un programme fonctionne en utilisant la mémoire vive de l'ordinateur. On parle de fuite de mémoire quand la consommation de mémoire d'un logiciel augmente continuellement. Si la totalité de la mémoire de la machine est utilisée, le système d'exploitation ne fonctionnera plus.

Nous n'avons jamais utilisé d'outil d'analyse statique lors de nos développements. Après étude des logiciels existants, nous avons retenu deux solutions d'analyses de code qui couvrent chacune un domaine différent :

Checkstyle **:** Checkstyle est un outil de contrôle du code source Java. Il s'attache uniquement à vérifier l'écriture et la syntaxe du code. Cet outil permet notamment de vérifier entre autres :

- La présence de commentaires au format "Javadoc" pour les classes, les méthodes et les attributs.
- Les conventions de « nommages » (cf. chapitre "normes de codage").
- Les bonnes pratiques d'écriture.

Cet outil de contrôle peut s'utiliser dans le cadre de la mise en place de normes de codage

FindBugs **:** Findbug est une solution d'analyse qui vérifie la qualité du code en cherchant des motifs d'erreurs connus. Il ne vérifie pas la syntaxe du code. L'outil analyse le code source et émet un avertissement dès qu'il rencontre un bug potentiel.

Les deux solutions sont intégrées à notre plate-forme de développement, Eclipse, via des plugins qui seront testés en profondeur lors de la prochaine phase de développement du projet ILSDB, en juin 2012.

Après une première phase de tests et d'intégration en mars et avril 2012, nous pouvons d'ores et déjà tirer les conclusions suivantes :

- L'utilisation de tels outils entraîne une charge de travail supplémentaire puisque il faut analyser les rapports d'erreurs qu'ils produisent. Il faudra donc évaluer si la charge de travail induite est inférieure au temps de corrections des bugs supposés. Cette charge de travail supplémentaire est à mettre en rapport avec l'augmentation de la satisfaction du client.

- Nous manquons encore d'expérience dans le déploiement de ces outils. La configuration reste complexe, et, dans l'état actuel, ces rapports d'erreurs contiennent un grand nombre de "faux positifs", c'est à dire de bugs qui n'en sont pas. Ces « faux-positifs » sont dus à des règles très strictes dans l'organisation du code.

- Les tests effectués jusqu'à maintenant ne permettent pas encore de statuer sur la réelle utilité de ces outils.

10.1.4.4 DETECTION DES ERREURS LIEES A LA BASE DE DONNEES

En informatique de gestion, l'intégrité de l'information, stockée dans la base de données est au cœur de toutes les préoccupations. Par exemple dans le cas NH Industrie, présenté dans les chapitres précédents, l'outil DS3 est utilisé comme référence de prix pour l'ensemble des compagnies partenaires. A ce titre un écart, ne serait-ce que d'un centime sur un prix, n'est pas envisageable.

Nous allons donc dans cette partie aborder les bugs liés à la base de données. Ces bugs ont plusieurs causes, par exemple

- **Procédure stockée qui ne fonctionne pas**

- **Erreur d'insertion de données**

- **Problème de types de données**

Ces erreurs sont bien plus difficiles à détecter que les bugs applicatifs, et ne sont parfois pas visibles immédiatement. Il n'existe pas de solutions de tests unitaires et automatisables de base de données compatible avec tous les systèmes de gestion de base données. Ces tests sont très spécifiques à une base, une architecture et un métier.

La meilleure façon de tester une procédure stockée ou une requête est d'écrire manuellement un script qui appelle cette procédure et vérifie son fonctionnement. Cette solution est très efficace mais rajoute une charge de travail non négligeable, puisqu'il faut écrire manuellement tous les tests. Par exemple, dans le cas du projet DS3, un des tests a consisté à calculer manuellement une série de prix, et à comparer ces résultats manuels avec les résultats calculés par l'outil.

10.1.5 MAINTENABILITÉ

On entend par « maintenabilité » du code sa capacité à être corrigé et modifié tout au long de sa vie. Ce paramètre est particulièrement important dans le domaine des systèmes d'informations car les logiciels et les bases de données sont conçus et utilisés de nombreuses années. Cette capacité à être maintenu est essentielle. Quatre-vingt pour cent du temps de développement d'un logiciel est consacré à sa maintenance, et bien souvent, la maintenance est effectuée par des personnes n'ayant pas participé au développement initial.

Il convient de distinguer deux types de maintenance :

- **La maintenance corrective :** Cette pratique consiste à corriger des bugs dans le logiciel.
- **La maintenance évolutive** : Cette pratique consiste à ajouter de nouvelles fonctionnalités au logiciel.

Tout comme pour l'analyse des bugs, il n'existe pas de méthode formelle pour s'assurer qu'un code source pourra être maintenu. Il existe cependant un certain nombre de pratiques que nous essayons de mettre en place

- **Utiliser des composants connus et maintenus :** Un logiciel est composé d'un ensemble de "briques" ou composants. Ces composants évoluent au fil du temps. Il faut s'assurer qu'ils soint maintenus par leurs éditeurs respectifs. Par exemple, le projet ILSDB utilise la bibliothèque "Apache POI" pour créer des fichiers Excel. Cette dernière est supportée par la communauté Apache, une des plus importantes communautés open-source. Il y donc très peu de risques qu'elle soit abandonnée. A l'inverse, utiliser une bibliothèque peu connue entraîne des risques qu'elle ne soit abandonnée par ses créateurs et que les bugs qu'elle comporte ne soit plus corrigés. Pour le projet DS3, nous avons utilisé la bibliothèque JExcel pour gérer les fichiers excel. Cette bibliothèque a été abandonnée, et ne supporte pas nativement les nouveaux formats de fichiers Excel. Nous devons faire migrer toutes les fonctions utilisant ces librairies vers Apache POI, ce qui entraîne un coût de développement non négligeable.

- **Utiliser une architecture connue :** L'architecture d'un logiciel est la façon dont sont organisés ses différents composants. Utiliser une architecture standard et reconnue permet à d'autres développeurs de comprendre et modifier le code. Nous utilisons pour nos projets une architecture MVC, appuyée sur des Framework Java tels que JSF 2 ou Struts 2. Ces Framework sont des standards de l'industrie et sont maîtrisés par un grand nombre de développeurs.

- **Documentation du code :** Une bonne documentation est la clé de l'évolution d'un logiciel. Cette documentation est formalisée en utilisant des outils tels que la Javadoc. A l'heure actuelle, l'utilisation de la Javadoc n'est pas systématique cependant nous avons décidé pour nos prochains développements de rendre son utilisation obligatoire. La présence de cette documentation peut être vérifiée à l'aide d'outils comme Checkstyle.

10.1.6 PROCÉDURES DE TEST

Les tests logiciels ont pour but de vérifier que le logiciel est conforme aux attentes et aux spécifications. Un test est composé d'un ensemble de cas d'utilisation. La quantité des tests est proportionnelle à la taille du logiciel. On en distingue trois types :

- ***Les tests unitaires*** : Sont des tests réalisés par le développeur après chaque activité de développement. Ils visent uniquement à vérifier que la fonction développée fonctionne

- ***Les tests d'intégration :*** Sont des tests réalisés lorsque l'ensemble des développeurs participant au projet ont mis en commun leurs travaux. Durant cette phase, on vérifie que les modifications apportées par une personne n'ont pas d'incidences sur l'ensemble du programme

- ***Les tests de non-régression :*** consistent, à la suite d'une évolution du produit, à tester des anciens modules de l'application afin de s'assurer que les nouvelles modifications n'ont pas altéré leur bon fonctionnement

Généralement, l'ensemble des cas d'utilisation sont regroupés sous la forme d'un document appelé plan de test. Ce document décrit l'ordre et la manière de tester le logiciel. Dans les projets les plus importants, les tests sont menés par une équipe dédiée. Dans notre cas, nous les réalisons nous-mêmes.

Le manque de test sur le projet DS3 a été une des principales sources de problèmes, particulièrement sur la partie import des données. Sans plan de test, nous n'avions pas envisagé la totalité des cas possibles. Concrètement, nous avons utilisé, durant la phase de validation, des fichiers d'exemple ne reflétant pas suffisamment la réalité. Nous nous sommes rendu compte de l'erreur lors du déploiement de l'application en phase de qualification, ce qui a entrainé un retard de livraison important.

A la suite de ces problèmes, nous avons mis en place de nouvelles règles et une nouvelle procédure de tests pour le projet ILSDB.

10.1.6.1 TRAÇABILITÉ DES TESTS UNITAIRES

Sur le projet DS3, les tests unitaires ont été réalisés par la personne ayant en charge la fonction ou le module, aucun document n'a été rédigé. Il n'y a donc aucun moyen de voir si la couverture du code est totale (c'est-à-dire si l'ensemble du code a été testé). Nous avons mis en place, pour le projet ILSDB un modèle de fiche de test unitaire. Désormais, chaque tâche Redmine affectée à un développeur devra être accompagnée de sa fiche de test, qui nous permettra d'évaluer les réalisations et de réagir plus rapidement en cas de problème.

Fiche de test unitaire

Nom du projet	test	Date	01/05/2012
Nom du developpeur	BERNADA	Nb ticket Redmine	356
Module	Interface d'administration des utilisateurs		

DESCRIPTION DE LA FONCTION

Création d'un lien sur le menu principal
Création d'un tableau d'affichage des utilisateurs (jqgrid)
Création des procédures stockées
Crétion des fonctions d'ajout/suppression/modification des utilisateurs

DESCRIPTION DES TESTS EFFECTUES

Affichage des utilisateurs
tri des utilisateurs dans le tableau (jqgrid)
ajout d'un uilisateur
suppression d'un utilisateur
modification d'un utilisateur

Figure 14: Exemple de fiche de test unitaire attachée à une tâche Redmine

10.1.6.2 TESTS D'INTÉGRATION

Nous avons mené les tests d'intégration en utilisant une méthode que j'avais eu l'occasion de mettre en pratique dans un précédent projet. Cette méthode présente l'avantage d'être très simple à mettre en place et peu couteuse.

Le principe de cette méthode est d'écrire tous les cas unitaires de l'application. Ces cas sont ensuite regroupés sous forme de scénarios, et croisés dans une matrice, permettant de voir si la couverture est totale. Nous identifions trois niveaux de bugs : majeur, bloquant et mineur.

Chaque scénario est ensuite testé, et les résultats sont croisés afin d'obtenir le taux de réussite d'un scénario. Ces procédures semblent fonctionner correctement. Même si le projet ILSDB n'est qu'en phase de qualification, le nombre de bugs bloquants remontés est inférieur à celui du projet DS3, pourtant de taille inférieure. Nous allons travailler dans nos prochains projets à l'automatisation des tests unitaires.

Num cas unitaire	SC01 : Import de données	SC02 : Création des liens Lsa /	SC03 : Création des taches de	SC04 : création des sous-tach	SC05 : Modification d'un LCN	SC06 : Création d'un utilisate	Couv
LOG01	x	x	x	x	x	x	
APP01	x	x					
APP02	x			x			
APP03		x					
APP04		x		x	x		
FI01				x		x	
FI02	x						
FI03		x		x			
FI04		x					
FI05		x					
FI06				x			
FI07		x				x	
FI08							
FI09			x				
FI10	x				x		
DI01			x			x	
DI02			x		x	x	
DI03	x					x	
DI04			x				
DI05	x						
DI06			x			x	
DI07							
DI08						x	
DI09						x	
DI10			x	x		x	
DI11						x	
DI12						x	
DI13						x	

Figure 15 : Matrice de cas unitaires

CAS D'UTILISATION

RUN	1	NUMERO	LOG 01	DATE	01/04/2012
		TESTEUR	BERNADA		

DEROULEMENT DES TESTS

Le but de ce test est de vérifier que la connexion entre le serveur tomcat et le serveur LDAP est fonctionelle. Nous utiliserons pour les tests le serveur LDAP LGM. Le but est donc de se connecter à l'application en utilisant son compte LGM.

1 : ouvrir un navigateur web à l'adresse : http://groupesi:8080/ilsdb/ la fenêtre suivante s'affiche :

2 : Le test est valide si la connexion avec le login/mdp LGM fonctionne. La page d'accueil doit s'afficher :

Figure 16 : Exemple de cas d'utilisation

SC01	RUN 1	RUN 2	RUN 3	RUN 4
LOG01	OK	OK	OK	OK
APP01	KO	ko	OK	OK
APP02	OK	OK	OK	OK
APP03	KO	KO	OK	OK
APP04	ko	ko	OK	OK
FI01	OK	OK	OK	OK
FI02	OK	OK	OK	OK
FI03	KO	ko	ko	ko
FI04	ko	OK	OK	OK
FI05	OK	OK	OK	OK
FI06	OK	OK	OK	OK
FI07	OK	OK	OK	OK
FI08	OK	OK	OK	OK
FI09	OK	OK	OK	OK
FI10	OK	OK	OK	OK
DI01	OK	OK	OK	OK
DI02	KO	KO	KO	OK
DI03	KO	KO	KO	ko
DI04	ko	OK	OK	OK
DI05	OK	OK	OK	OK
DI06	OK	OK	OK	OK
DI07	KO	OK	OK	OK
DI08	OK	OK	OK	OK
DI09	ko	OK	OK	OK
DI10	ko	ko	ko	OK
DI11	OK	OK	OK	OK
DI12	OK	OK	OK	OK
DI13	KO	OK	OK	OK

Figure 17 : Bilan des phases de tests pour le scénario 1

Bilan des tests				
% OK	RUN 1	RUN 2	RUN 3	RUN 4
SC01	57	75	85	93
SC02	60	85	90	95
SC03	62	70	90	100
SC04				
SC05				
SC06				
% ko	RUN 1	RUN 2	RUN 3	RUN 4
SC01	18	14	7	7
SC02	20	10	5	95
SC03	18	15	8	100
SC04				
SC05				
SC06				
% KO	RUN 1	RUN 2	RUN 3	RUN 4
SC01	25	10	7	0
SC02	20	5	5	0
SC03	20	15	2	0
SC04				
SC05				
SC06				

Figure 18 : Bilan global de la phase de tests

10.1.6.3 AUTOMATISATION DES TESTS JAVA

Nous cherchons une solution afin d'automatiser les tests unitaires. Automatiser des tests signifie que ces derniers seront réalisés par un programme informatique, et non plus par le développeur. Cette méthode permet de gagner un temps non négligeable. La méthode de test la plus connue et la plus utilisée dans le monde Java est JUNIT.

JUNIT est une bibliothèque de tests unitaires pour le langage de programmation Java crée par Kent Beck et Erich Gamme. Junit définit deux types de fichiers tests. Les « TestCase » sont des classes Java contenant un certain nombre de tests, et les « testSuite » permettent d'exécuter un certain nombre de TestCase préalablement définis (Wikipedia).

Nous n'avons jamais eu l'occasion de tester ces librairies. Nous allons les mettre en place dans le cadre d'un projet « pilote » interne, qui devrait commencer en juillet 2012. Ce projet nous permettra de prendre en main cette librairie, de voir les principaux avantages et inconvénients, avant une intégration éventuelle à plus grande échelle, lors de la seconde partie des développements du projet ILSDB, qui doit commencer en septembre 2012. Nous utiliserons JUNIT, dans un premier temps, pour tester les composants communs à tous les projets.

Les bénéfices apportés par la librairie JUNIT semblent prometteurs. Nous cherchons à évaluer et dimensionner la surcharge de travail induite par cette dernière.

10.1.7 AMÉLIORATION DE LA QUALITÉ

L'amélioration de la qualité est un processus incrémental. Plusieurs axes sont envisagés, parmi lesquels, outre les améliorations techniques, nous envisageons :

- Un partage des connaissances
- Une mutualisation des compétences
- Un effort de formation

10.1.7.1 PARTAGE DES CONNAISSANCES

Les difficultés techniques ou fonctionnelles rencontrées sur un projet doivent être documentées, afin de ne pas les reproduire. Nous avons donc décidé de mettre en place un wiki, commun à tous nos projets.

Ce wiki a pour but de mettre à disposition des différents acteurs une plate-forme d'échange de savoirs. Il sera intégré à notre plate-forme de suivi des tâches. Le premier wiki a été mis en place pour le projet ILSDB. Il comporte deux sections principales :

- Processus métiers : Cette section regroupe toutes les informations relatives au processus métiers de l'application. Par exemple, l'article « gestion des applicabilités particulières » traite de l'applicabilité de certaines tâches pour certaines variantes d'un appareil.
- Technique : Cette section regroupe tous les articles techniques liés à l'application. Par exemple l'article « coloriser des lignes dans une datable » explique en détail comment colorier certaines lignes sur un composant « Datatable ».

D'une façon générale, tous les points ayant posé problème seront documentés dans le wiki.

10.1.7.2 MUTUALISATION DES COMPÉTENCES

LGM favorise le partage des connaissances au travers de « réunions méthodes » durant lesquelles un consultant présente un client, un projet, ou une méthode. Jusqu'à présent le groupe SI n'avait jamais organisé ce type de réunions : Un de nos objectifs pour l'année 2012 est d'en organiser aux moins deux.

Nous discutons également avec la Direction des systèmes d'information, la direction technique de l'innovation, et l'équipe de développement parisienne de la mutualisation de nos compétences. Nous souhaitons mettre en commun certaines ressources. Par exemple, nous mettons en place, dans le cadre d'un projet interne, un serveur d'application Tomcat qui sera hébergé à Paris, accessible à nos clients depuis Internet, et administré par l'équipe d'Aix.

10.1.7.3 FORMATION

L'amélioration de la qualité nécessite un effort de formation, et donc d'investissement important. Plusieurs actions ont été menées en 2011 et 2012 :

- **Formation interne aux nouveaux Framework Java :** Suite au projet ILSDB, un travail de formation aux nouveaux Framework Java a été mené. J'ai ainsi pu bénéficier d'une formation au Framework JSF.
- **Formation aux métiers de l'aéronautique (soutien logistique) :** Depuis le projet ILSDB, chaque développeur travaillant sur un nouveau domaine de compétence métier (Sûreté de fonctionnement, soutien logistique intégré, etc…) est formé par un chef de projet métier.
- **Montée en compétence des membres de l'équipe :** Cette montée en compétence se traduit, dans mon cas, par l'obtention de deux jours par mois pour préparer mon mémoire CNAM.

Cette volonté de montée en compétence de l'équipe s'est également concrétisée par l'arrivée d'un nouveau chef d'équipe, spécialisé dans le management et la gestion de projet.

10.1.7.4 CONCLUSION

L'amélioration de la qualité est certainement l'activité la plus délicate. Les efforts à fournir sont continus, et les résultats ne seront visibles qu'à long terme. Il n'existe pas de recette toute faite. Seules l'expérimentation et la recherche permettent de valider ou d'infirmer la justesse d'une solution.

Cette amélioration nécessite également une forte motivation de l'équipe de développement. Ce n'est que très impliquée et motivée qu'elle pourra produire un travail de qualité.

10.2 PLATE-FORME DE DÉVELOPPEMENT

10.2.1 INTRODUCTION

Nous allons dans cette partie aborder le choix de la plate-forme de développement (EDI).

Un environnement de développement intégré (IDE) est un programme regroupant un ensemble d'outils pour le développement des logiciels. En règle générale un EDI regroupe un éditeur de texte, un compilateur, et un débogueur.

Beaucoup d'anciens langages n'ont pas eu d'EDI associé, car les développements étaient faits via des organigrammes, des formulaires de codification et des cartes perforées soumises à l'ordinateur. Les EDI sont donc apparus avec les développements « sur console ».

Le premier langage créé — et livré — avec un EDI fut ainsi le Dartmouth BASIC en 1964, premier langage conçu pour être utilisé avec un terminal d'ordinateur. Son EDI était basé sur des commandes saisies « en ligne » : Pour de tels langages, les EDI consistaient en une interface graphique sommaire, assortie à un système de construction (compilation et édition de liens) de programme « par makefile » : Ils permettent de décrire les options de compilation et édition voulues pour la construction du programme. La syntaxe de plus en plus évoluée de ces fichiers a abouti à des pseudo-langages de plus en plus complexes et rébarbatifs tant leurs possibilités se sont étendues (ils permettent par exemple d'ajouter des structures de contrôles aux instructions de compilations ou d'édition de liens), conduisant à la mise en place de fichiers de configuration pour décrire et personnaliser ces options et automatiser leurs enchaînements.

Les EDI récents sont conçus pour des interfaces plus évoluées : menus, boutons, utilisation combinée clavier/souris, etc. Ils masquent et contrôlent les commandes techniques sous-jacentes depuis des IHM graphiques, évitant ainsi aux programmeurs débutants d'être confrontés à la rugueuse et exigeante syntaxe du makefile.(*Source wikipedia*)

Il existe deux grandes familles d'environnement de développement : Les EDI généralistes et les EDI spécialisés. Un EDI généraliste peut être utilisé pour concevoir des logiciels utilisant des technologies différentes. Un EDI spécialisé est conçu pour n'utiliser qu'un seul langage de programmation.

Nous allons dans cette première partie :

- Décrire les différentes plates-formes de développement utilisées au sein du groupe SI.
- Analyser le besoin d'évolution
- Proposer une solution

Nous utilisons aujourd'hui un certains nombres d'outils, liés à une technologie, ou aux préférences du développeur. Cette multiplication des plates-formes de développement pose un certain nombre de problèmes :

- **Difficultés de maintenance :** La multiplication des outils entraîne une maintenance plus importante. Il faut, par exemple, vérifier les mises à jour de chaque outil indépendamment, et connaître les spécificités particulières de chaque outil pour l'utiliser au mieux.

- **Interaction avec les outils LGM :** Tous les outils utilisés aujourd'hui ne sont pas intégrés aux outils LGM. Ainsi les éditeurs de texte "simple" ne peuvent pas, par exemple, utiliser le gestionnaire de version SVN. Il faut leur adjoindre des outils tiers. Cela ajoute de la complexité et crée un risque d'erreur.

Suite aux problèmes rencontrés ci-dessus, nous avons décidé de faire migrer nos projets vers une plate-forme de développement commune à l'ensemble. Cette migration se fait en plusieurs étapes :

- Cartographie des différents outils utilisés par le groupe SI
- Analyse du besoin
- Analyse des outils permettant de répondre à ce besoin
- Proposition de solution, et mise en place de cette dernière si elle est acceptée.

10.2.2 DESCRIPTION DES DIFFERENTS OUTILS DE DEVELOPPEMENT UTILISES AU SEIN DU GROUPE SI

Au fur et à mesure de ses missions, le groupe SI a été amené à travailler sur différentes technologies: PHP, JAVA, Python ...etc. Pour chacune de ces technologies, nous avons utilisé une plate-forme de développement différente. Nous utilisons aujourd'hui trois éditeurs différents, plus ou moins évolués. Nous excluons de cette analyse les technologies Windows, qui nécessitent pour la plupart leur propre environnement de développement : Access, C#, etc.

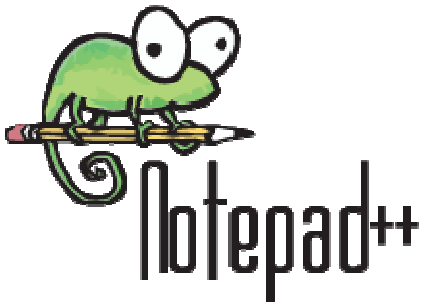
Nous utilisons les outils suivants :

- **Eclipse** : Environnement de développement intégré utilisé pour tous nos développements Java. Ce logiciel présente l'avantage d'être connecté via des plugins aux différents composants de notre environnement de développement (SVN, Redmine)

Eclipse est un EDI généraliste. Initialement utilisé pour le développement Java, il est aujourd'hui adaptable à tous les langages existants.

- **Scintilla** : Scintilla est un éditeur de texte uniquement open-source. Historiquement il est utilisé pour tous nos développements Python. Scintilla n'est pas connecté à SVN. Il faut donc lui adjoindre un client SVN externe, Tortoise SVN par exemple.

- **Notepad++** : Notepad++ est un éditeur de texte rapide et léger, basé sur Scintilla. Nous l'utilisons principalement pour l'édition et la création de fichiers de configurations.



- **Autres éditeurs** : Nous utilisons parfois certains éditeurs de texte, liés à une plate-forme particulière (par exemple, "vi" sous les systèmes Solaris et Linux).

- **Tortoise SVN :** Est un client SVN. C'est un logiciel qui permet de synchroniser les fichiers entre le serveur SVN et le poste client. Nous l'utilisons avec les éditeurs ne disposant pas de connexions avec notre serveur SVN (Scintilla et Notepad ++).

10.2.3 ANALYSE DU BESOIN

Après avoir décrit les outils utilisés et les problèmes rencontrés, je vais maintenant étudier nos besoins. Nous souhaitons mettre en place un écosystème de développement, incluant, outre un éditeur de code, une solution de gestion de configuration (SVN) et une solution de suivi de projets(Redmine). Nous avons donc besoin d'un éditeur capable de s'intégrer à cet environnement.

Quels sont nos besoins ?

- Plate-forme de développement multi-langages : Java / PHP / Python
- Interaction avec les différents outils groupe SI : Outil de gestion des tâches, et outil de gestion de la configuration.
- Evolutivité : La solution retenue devra être évolutive dans le temps et les fonctionnalités.
- L'informatique évoluant rapidement, nous serons probablement amenés à utiliser de nouveaux langages et de nouveaux outils dans les années à venir.

Pourquoi ce choix d'évolution ?

- Uniformisation des outils : La standardisation de nos process passe par une uniformisation de nos outils. Ainsi un développeur changeant de projet régulièrement utilisera toujours les mêmes outils. Le temps de prise en main et d'apprentissage est donc fortement réduit. Cette uniformisation sera accompagnée de normes de codages.
- Uniformisation des processus : Quel que soit le projet, les processus de développement sont les mêmes. Les outils de gestion configuration sont également les mêmes.
- Facilité de maintenance : Il est plus facile de maintenir une plate-forme entière qu'une multitude d'outils disparates.

Partant de ce constat, Il apparaît donc que les éditeurs de texte "simple" de type Scintilla ou Notepad ++ ne sont plus adaptés à nos besoins. Ces logiciels nécessitent l'ajout de plugins externes pour interagir avec les outils utilisés (ex Redmine et SVN). Après réflexion, nous avons donc décidé de faire migrer la majorité de nos projets de développement vers une plate-forme commune.

Nous avons donc besoin d'un outil global et avons choisi de nous tourner vers un EDI généraliste. Ce type d'outil apporte les fonctionnalités suivantes :

- Multi-langages : hormis les langages proposés par Microsoft (type C#, Visual basic) que nous n'utilisons pas, un EDI supporte la majorité des langages. Un développeur pourra travailler sur des projets différents en conservant le même environnement de travail. La phase de prise en main d'un nouvel outil sera donc plus rapide.
- Connectivité avec l'outil de gestion de configuration : L'outil peut se connecter à notre serveur SVN ainsi le développeur peut enregistrer ces modifications directement depuis l'EDI.
- Connectivité : L'EDI se connecte directement à l'outil de gestion des tâches

Plusieurs environnements de développements sont disponibles sur le marché. Nous privilégierons, pour des raisons de coûts une solution open-source, quand cela sera possible.

10.2.4 CHOIX DE L'ENVIRONNEMENT DE DEVELOPPEMENT :

10.2.4.1 ANALYSE DES SOLUTIONS EXISTANTES :

Il existe sur le marché plusieurs EDI qui ont chacun leurs avantages et leurs inconvénients. Nous avons évalué les EDI suivants :

- **Eclipse :** Environnement très complet, historiquement lié au langage Java, peut être utilisé pour tous les autres langages via des plugins. (cf. présentation ci-dessus)
- **Kdevelop** : Environnement de développement lié au système Linux, et au gestionnaire de fenêtre KDE.
- **NetBeans** : Développé par Sun Microsystems. Lié au langage Java. Peut être utilisé pour d'autres langages.
- **Microsoft Visual Studio** : Environnement de développement lié aux langages Microsoft.
- **Php Designer** : EDI multiplateforme lié au développement web.

10.2.4.2 CONCLUSIONS

Microsoft Visual Studio : L'environnement Visual Studio ne peut pas être utilisé, dans la mesure où il ne supporte que les langages de programmation Microsoft. Nous ne l'utiliserons que dans le cas de développements spécifiques

Kdevelop : Kdevelop est lié à l'environnement Linux et nous ne pouvons pas l'utiliser (nous travaillons quasiment exclusivement dans un environnement Windows).

Php Designer : Php Designer est orienté développement web. Il ne permet pas le développement d'outils bureautiques de type client lourd.

Le choix se fera donc entre les environnements NetBeans et Eclipse. Nous utilisons actuellement Eclipse pour nos développements Java. La solution NetBeans a été également utilisée, dans le but de la comparer à Eclipse.

Quels sont les avantages et les inconvénients de ces deux plates-formes ?

Eclipse

Avantages

- Déjà utilisé au sein du groupe SI. L'outil est bien maitrisé par les développeurs LGM.
- Support de la communauté : Eclipse offre l'avantage d'être soutenu et développé par un grand nombre d'utilisateurs. De nouvelles versions et un grand nombre de plugins améliorent de façon continue son fonctionnement.

Inconvénients

- Ressources : Eclipse requiert une machine puissante pour pouvoir être correctement utilisé
- Complexité : Eclipse est compliqué à prendre en main à cause de son grand nombre de fonctionnalités

NetBeans

Avantages

- Simplicité d'utilisation : NetBeans est beaucoup plus simple à prendre en main qu'Eclipse

Inconvénients

- Support du Python : Le support du langage Python n'est pas aussi performant que celui d'Eclipse

10.2.4.3 PROPOSITION DE SOLUTION

Après avoir analysé les problèmes d'outils rencontrés lors d'un développement logiciel, et après avoir analysé notre besoin, nous avons décidé de faire migrer nos projets vers une plate-forme de développement commune, à l'aide d'un EDI.

Deux EDI répondent correctement à nos besoins : Eclipse et NetBeans. Ces deux EDI, présentent globalement les mêmes fonctionnalités. Une seule différence de taille les sépare : Eclipse est déjà utilisé sur une partie de nos projets. Dans le cas d'une migration vers Eclipse, l'opération portera uniquement sur les projets Pythons. Dans le cas d'une migration vers NetBeans, il faudra déplacer l'ensemble de nos projets.

Nous avons donc choisi la plate-forme Eclipse. La migration portera uniquement sur les projets Python, les projets Java utilisant déjà Eclipse. Elle se fera au fur et à mesure des nouveaux développements. Nous ne déplacerons pas les anciens projets, qui ne sont plus en phase de développement actif.

10.3 NORMES DE CODAGE

10.3.1 INTRODUCTION

Nous allons dans cette partie aborder la question des normes de codage. D'une manière générale, en informatique, on estime que la phase de maintenance représente la majeure partie du cycle de vie d'un logiciel. Pour faciliter cette maintenance, Il est donc primordial que le logiciel soit construit selon des normes précises et respecte un certain nombre de règles définies lors de sa création.

Ces règles constituent un ensemble de bonnes pratiques, dont le but est :

- Améliorer la lisibilité et la compréhension du code par des personnes n'ayant pas participé au développement initial.
- Faciliter la maintenance et l'évolution du produit

L'analyse des outils précédemment développés nous a amené aux conclusions suivantes :

- **Nous ne disposons d'aucune norme de codage** : Nos outils sont développés en suivant un certain nombre de règles, propres à chaque développeur. Ces règles sont issues de l'expérience de chacun, et n'ont jamais été formalisées.
- **La complexité grandissante de nos outils entraîne un besoin de normalisation** : Cette normalisation fait partie intégrante de l'assurance qualité du logiciel, et s'inscrit dans le cadre de la professionnalisation de nos développements.

Nous fondant sur cette analyse, nous avons décidé de mettre en place une norme de codage pour nos applications. Le but de ce chapitre est donc de décrire cette règle qui devra être appliquée lors de nos prochains développements et lors de la maintenance d'anciens outils.

Ces règles sont élaborées à partir des règles communément admises dans le développement informatique, et à partir de l'analyse de règles de codage existantes fournies notamment par les éditeurs des langages de programmation.

Ce ne sont pas des règles absolues qui doivent être appliquées coûte que coûte mais des recommandations. Nous avons décidé dans un premier temps de les formaliser pour les langages Python et Java, que nous utilisons en ce moment. D'autres préconisations seront proposées si de nouveaux langages sont utilisés.

10.3.2 RÈGLES GÉNÉRALES

10.3.2.1 TAILLES DES FICHIERS

Généralement en informatique, la taille maximale d'un fichier est d'environ 1000 lignes au maximum. Cette limite est théorique et peut dans certains cas être dépassée. Néanmoins des fichiers de grandes tailles sont difficiles à lire et traduisent généralement une erreur de conception.

10.3.2.2 TAILLES DES MÉTHODES

De même que pour la taille des fichiers, il est généralement déconseillé d'écrire des méthodes de plus de 100 lignes. De trop longues méthodes traduisent souvent un mauvais découpage des fonctions. La maintenance de telles méthodes sera plus compliquée.

10.3.2.3 NOM ET DÉCLARATION DES VARIABLES

- Première lettre en minuscule
- nom de type "CamelCase" (Mélange de minuscule, majuscule avec la première lettre de chaque mot en majuscule)
- Donner des noms simples, décrivant précisément ce que fait la classe.
- Seuls les caractères alphabétiques, les chiffres et les points sont autorisés.

10.3.2.4 NOM ET DÉCLARATION DES CLASSES

- Première lettre en majuscule (dans le cas d'un développement Java)
- Mêmes recommandations que pour les variables.

10.3.2.5 NOM ET DÉCLARATION DES FONCTIONS

- Première lettre en minuscule.
- caractère de séparation "_" entre les mots. Par exemple : « ma_super_fonction() ».

10.3.2.6 TAILLE DES LIGNES

Toute l'équipe travaille sur des écrans 22 pouces. Nous avons décidé de limiter la taille maximum d'une ligne à 180 caractères, ce qui correspond à la largeur d'un écran d'édition Eclipse.

10.3.2.7 ENCODAGE DES FICHIERS

Tous les fichiers sources utilisent le jeu de caractères UTF-8. Ce jeu de caractères international est compatible avec toutes les langues.

10.3.2.8 GESTION DES EXCEPTIONS

Que ce soit en Java ou en Python, toutes les fonctions devront gérer les exceptions.

Par exemple, la fonction suivante n'est pas valide :

```
public void maFonction(String maVariable){
    int a = 0;
    int b = 1;
    int c = a + b;
}
```

La fonction doit gérer les erreurs :

```
public void maFontion(String maVariable) {

    try {
        int a = 0;
        int b = 1;
        int c = a + b;

    }catch (Exception e){
        e.printStackTrace();
    }


}
```

10.3.3 NORME D'AFFICHAGE ET CONFIGURATION DE LA PLATE-FORME DE DEVELOPPEMENT

Dans le cadre de la standardisation de nos développements, nous avons décidé de faire migrer nos projets vers la plate-forme Eclipse .Ce chapitre décrit la configuration d'Eclipse pour les postes de développement.

10.3.3.1 INDENTATION DU CODE

L'indentation du code est un élément important de la configuration de notre plate-forme. En Python, elle est le seul élément permettant de différencier le début et la fin d'un bloc de code. Les blocs sont indentés en utilisant des tabulations, comme ci-dessous :

```
package com;

public class Test {

    private  String maVariable;

    public String getMaVariable() {
        return maVariable;
    }

    public void setMaVariable(String maVariable) {
        this.maVariable = maVariable;
    }

}
```

Il existe deux manières de faire des tabulations : soit en utilisant le caractère "tab" du clavier, soit en utilisant un certain nombre de caractères "espace". L'environnement de développement doit être configuré pour utiliser une méthode ou l'autre. Nous avons choisi de faire les tabulations en utilisant quatre espaces. Cette méthode présente l'avantage d'être compatible avec la majorité des éditeurs de codes existants. Ce paramétrage est très important, le langage Python n'autorisant pas le mélange de tabulations et d'espaces.

10.3.4 REGLES SPECIFIQUES AU LANGAGE PYTHON

Le langage Python reprend en grande partie les règles énoncées dans les paragraphes précédents, avec cependant quelques particularités, listées ci-dessous.

10.3.4.1 INDENTATION DES BLOCS DE CODE

Cette indentation est primordiale en Python. Ce langage n'utilise pas, à l'instar de Java d'accolades pour séparer les blocs de code qui sont différenciés par les indentations. Ces dernières doivent toutes être du même type, Python n'autorisant pas, dans le même document, un mélange de tabulations et d'espaces (cf. paragraphe précédent "indentation du code").

10.3.4.2 RÈGLES DE NOMMAGE

- Les règles de « nommage » des classes et variables sont les mêmes qu'en Java (cf. ci-dessus)
- Les méthodes privées sont précédées de deux caractères "__"

10.3.4.3 ÉCRITURE DU CODE

- Les méthodes sont séparées entre elles par une ligne vide.
- Les classes sont séparées entre elles par deux lignes vides

10.3.5 RÈGLES SPÉCIFIQUES AU LANGAGE JAVA

10.3.5.1 VISIBILITE DES CLASSES ET DES VARIANTES

D'une manière générale en Java, toutes les variables utilisées dans une classe ne doivent pas être accessibles à l'extérieur de la classe. On accède aux variables via des accesseurs. Néanmoins les deux syntaxes sont correctes du point de vue de la programmation

Déclaration incorrecte :

```
package com;

public class Test {

    public String maVariable;

}
```

Déclaration correcte :

La variable n'est pas accessible à l'extérieur de la classe. Il faut utiliser ses accesseurs.

```
package com;

public class Test {

    private  String maVariable;

    public String getMaVariable() {
        return maVariable;
    }

    public void setMaVariable(String maVariable) {
        this.maVariable = maVariable;
    }

}
```

10.3.5.2 LISIBILITÉ DU CODE

D'une manière générale, il faut favoriser la clarté et la lisibilité du code. Un code maintenable et évolutif est un code facilement compréhensible. Par exemple, l'expression ci-dessous, bien que correcte syntaxiquement n'est pas facile à lire :

```
d = (a = b + c) + r;
```

Il faut la remplacer par l'expression ci-dessous :

```
a = b + c;
d = a + r;
```

10.3.5.3 REGLE DE NOMMAGE POUR LES PACKAGES

- Le nom du package doit systématiquement être en minuscule
- Seuls les caractères alphabétiques, les chiffres et les points sont autorisés
- Le package doit être nommé com.lgm. <**CLIENT**>.<**PROJET**>. Ex : com.lgm.eurocopter.ilsdb

10.3.5.4 CONCLUSION

Nous avons défini dans ce chapitre un ensemble de règles de développement qui seront complétées au fur et à mesure de l'avancée des travaux. Les projets les plus anciens seront, eux, mis à jour au fur et à mesure des nouveaux développements.

10.4 NORMES DE CONCEPTION

Les normes de conception regroupent l'ensemble des règles définissant l'architecture de nos applications. Nous ne traiterons dans cette partie que les normes de conception logicielles liées à l'environnement Java, qui est le principal langage que nous utilisons.

La principale norme de conception que nous avons appliquée depuis le début de nos activités de développement est le « design pattern » « Modèle Vue Contrôleur (MVC) ». Un « design pattern » ou encore « patron d'architecture » est un modèle général de conception d'application.

Le « design pattern » MVC sépare l'application en trois couches distinctes :

- ***Le contrôleur*** : est chargé de l'interaction entre le modèle et les vues. Il ne modifie aucune donnée. Il intercepte par exemple les interactions de l'utilisateur, et appelle les fonctions correspondantes au modèle.

- ***Le modèle :*** Représente la couche métier de l'application. C'est donc la partie la plus importante. Le modèle contient la base de données et toutes les fonctions permettant de la manipuler.
-
  - ***Les vues*** : Les vues représentent l'interface homme-machine (IHM).

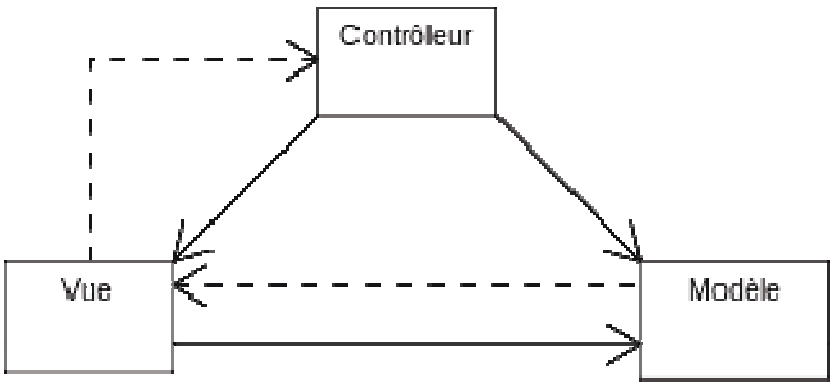


Figure 19 : le modèle MVC

Cette conception en couche permet une séparation des traitements, et garantit une réutilisation de la couche métier. Par exemple, dans le cas du projet DS3, si notre client ne souhaite plus utiliser une application web mais un « client lourd », il suffit de redévelopper les couches Contrôleur et Vue, en conservant notre modèle. A l'inverse, nous avons développé une application pour un de nos clients en utilisant la librairie «Swing » (client lourd). Cette application évoluera dans un second temps vers une application web. Il suffira pour cela de développer de nouvelles vues.

10.5 GESTION DE CONFIGURATION LOGICIELLE

Une gestion de la configuration efficace est une des clés du système qualité. La gestion de la configuration logicielle regroupe plusieurs disciplines du génie logiciel. Dans notre cas la gestion de la configuration est le processus qui consiste à historiser toutes les modifications qui sont apportées à un logiciel au cours de sa vie.

Lorsqu'on parle de gestion de configuration, deux modèles s'opposent.

- ***Le modèle verrouiller-modifier-libérer*** **:** Dans ce modèle, quand un développeur veut utiliser un fichier, il le verrouille, et lui seul peut y accéder. Historiquement ce système est le premier à avoir été développé. Il se révèle assez peu pratique à l'usage quand de nombreux développeurs ont besoin de la même ressource. Pour pallier ces restrictions, est apparu le second modèle.

- ***Le modèle copier-modifier-fusionner*** **:** Dans ce modèle, tout le monde peut accéder simultanément à la même ressource, et le logiciel se charge de fusionner les modifications, en proposant le cas échéant de valider ou de supprimer des modifications. Ce modèle, beaucoup plus évolué, est le plus répandu. Il présente cependant quelques lacunes. Le système de fusion des fichiers montre ses limites dans le cas de trop nombreuses modifications, et son utilisation est parfois délicate.

Nous avons mis en place en 2011 une gestion de configuration logicielle, sous la forme d'un logiciel de gestion de configuration logicielle, SVN. L'agence de développement parisienne a mis en place sa propre gestion de la configuration, utilisant un logiciel concurrent, Bazaar. Les deux logiciels utilisent le modèle copier-modifier-fusionner. Leur architecture est cependant complètement différente. Bazaar utilise un système décentralisé, c'est-à-dire que chaque développeur possède sur sa machine une copie du dépôt sur laquelle il peut travailler librement. Svn est un système centralisé, dans lequel le développeur ne possède que la copie locale de la partie du logiciel sur laquelle il travaille.

A l'heure de la standardisation des process de développement, la question de faire migrer tous nos outils vers une plate-forme commune s'est posée. Nous avons donc mené une réflexion sur ce sujet. Quels sont les avantages respectifs de chaque logiciel ? Quels intérêts avons-nous à mutualiser notre plate-forme ? Quels sont les avantages et les inconvénients de notre gestion actuelle.

10.5.1 PRESENTATION ET COMPARAISON DE BAZAAR ET SUBVERSION

Subversion

Subversion (Svn) est un logiciel de gestion de versions, distribué sous licence Apache et BSD. Il est fondé sur le modèle du copier/modifier/fusionner. Ce projet a été lancé en février 2000. Ce logiciel est fiable et très répandu. Sa prise en main peut s'avérer délicate pour un néophyte.

Avantages

- Très bien documenté
- fiable

Inconvénients

- Parfois difficile à prendre en main
- Peut occasionner des erreurs, notamment lors des fusions de fichiers.

Bazaar

Bazaar est un logiciel de gestion de version supporté par la société Canonical, à l'origine du système d'exploitation Ubuntu. Comme cela a été expliqué ci-dessus, Bazaar utilise normalement une architecture décentralisée, mais peut aussi être configuré pour utiliser une architecture centralisée. Bazaar peut s'interfacer avec un serveur SVN existant.

Avantages

- Meilleure gestion des conflits lorsque plusieurs personnes utilisent la même ressource

Inconvénients

- Configuration complexe
- Peu de retours d'expériences

10.5.2 INCONVENIENTS DE NOTRE PLATE-FORME ACTUELLE ?

A partir de notre travail d'audit, s'est posée la question de mutualiser la plate-forme de gestion du code source. A première vue il peut sembler anormal d'utiliser deux systèmes distincts, répondant au même besoin, et pouvant fonctionner de la même manière.

L'équipe de développement parisienne utilise un serveur Bazaar, qui héberge actuellement un projet sensible, sur lequel aucune migration ne sera possible dans l'immédiat. De plus la politique de la direction des systèmes d'information est d'héberger tous les serveurs critiques sur le siège Parisien, qui dispose des ressources nécessaires (humaines et matérielles) et suffisantes pour gérer de telles machines. La migration ne sera donc possible que de SVN vers Bazaar.

Le serveur SVN de l'agence d'Aix est géré par l'équipe de développement du groupe SI. Ce fonctionnement, même si il apporte une grande souplesse d'utilisation, présente néanmoins quelques inconvénients :

- L'agence d'Aix ne dispose pas d'une infrastructure réseau lui permettant d'héberger notre serveur dans de bonnes conditions. C'est-à-dire que le serveur fonctionne très bien sur un réseau local, mais la bande passante ne permet pas d'ouvrir l'accès à ce serveur à toutes les agences du groupe LGM.

- Le serveur utilisé actuellement, dont la puissance est suffisante à l'heure actuelle, ne suffirait pas si un grand nombre d'utilisateurs devait l'utiliser quotidiennement. Son remplacement s'inscrit dans le cadre de l'amélioration des processus. Doit-on investir dans une nouvelle machine, ou au contraire utiliser les ressources mises à disposition par la DSI ?

- La maintenance du serveur est assurée par le groupe SI. Cette maintenance, même si elle n'est pas quotidienne entraîne à la longue des coûts de structure plus importants pour notre groupe.

- Le contenu du serveur est sauvegardé quotidiennement. Cependant, le serveur lui-même n'est pas redondé. Si le serveur tombait en panne physique (composant matériel défectueux) ou en panne système (Système d'exploitation corrompu par un virus par exemple), nous ne disposons pas de solutions de remplacement immédiat. Selon la loi de Murphy, la probabilité que la machine tombe en panne un jour de livraison est importante...

Au vu de ces inconvénients, l'hébergement de la machine SVN sur le site de Paris semble être la solution. Nous étudions également la piste de l'externalisation de notre serveur SVN via un prestataire extérieur. Cette solution présente l'avantage d'être fiable et économique. Son principal inconvénient se situe au niveau de la confidentialité des données.

10.5.3 AVANTAGES DE NOTRE PLATE-FORME ACTUELLE ?

La solution actuelle, bien qu'imparfaite présente néanmoins l'avantage d'être très souple et très réactive. Nous pouvons créer les projets et configurer le serveur immédiatement, sans devoir en faire la demande à l'équipe parisienne. Les délais s'en trouvent donc réduits.

10.5.4 DEVONS-NOUS MIGRER VERS BAZAAR ?

Nous avons décidé dans un premier temps d'héberger notre serveur SVN actuel au sein de l'agence parisienne. La migration des projets actuels vers Bazaar semble difficile pour les raisons suivantes :

- Risque de perte d'information (historisation) durant la migration
- Certains projets sont critiques, on ne peut pas prendre le risque de les pénaliser
- Coûts importants

Même si les projets actuels pourront difficilement être orientés vers SVN pour le moment, cela ne pose pas de problèmes pour les nouveaux projets.

10.5.5 CONCLUSION

Dans le cadre de la standardisation de nos process, nous avons décidé de mettre en place une solution de gestion de configuration globale au sein du groupe LGM. Comme nous l'avons vu, cette migration est difficile, voire impossible pour les projets actuels. Les risques sont trop importants et nous ne voulons pas pénaliser des projets critiques.

Nous envisageons de déplacer notre serveur SVN vers l'agence parisienne, qui dispose des ressources nécessaires pour en assurer la gestion.

Nos prochains projets de développement utiliseront la plate-forme Bazaar mise à disposition par la DTI. Nous avons ainsi privilégié une migration progressive de notre gestion de configuration.

11 DEFINITION DU BESOIN ET GESTION DE PROJET

11.1 DÉFINITION D'UN PROJET

Selon le « PROJECT MANAGEMENT INSTITUT », un PROJET est une entreprise temporaire visant à créer un produit ou un service. On distingue les projets-ouvrages, dont le but est la production unique d'un produit (un logiciel par exemple) et les projets-produits, dont la finalité est la production d'un bien ou d'un service (la production d'une voiture par exemple).

Un projet comporte cinq aspects majeurs :

- Fonctionnel : réponse à un besoin
- Technique : respect des spécifications et des contraintes de mise en œuvre
- Organisationnel : respect d'un mode de fonctionnement (rôles, fonctions, culture, résistance au changement) de la structure cible
- Délais : respect des échéances
- Coûts : respect du budget

Les deux facteurs de réussite principaux d'un projet sont le management et la solution technique choisie. Ces deux facteurs sont indissociables. Si la solution technique retenue est la bonne alors que le management ne permet pas de planifier correctement les actions à mener, le projet ne réussira pas, il n'aboutira pas davantage si la gestion de projet est correcte et la solution technique inadaptée aux attentes du client.

Nous allons donc dans cette partie analyser la démarche visant à organiser de bout en bout le déroulement d'un projet. La gestion de projet s'est précisée au début du 20e siècle avec l'apparition des tâches répétitives, et s'est ensuite développée durant la seconde guerre mondiale avec la nécessité de coordonner des tâches complexes. Dès la fin de la guerre, ce système a été utilisé dans les milieux spatiaux, avant de s'étendre à tous les domaines d'activité.

Nous avons vu dans la partie « analyse de l'existant » que les principaux points négatifs des projets ILSDB et DS3 portent sur

- La définition du besoin : La communication et la compréhension entre les experts métiers, le groupe SI et le client, lors de la phase de rédaction des spécifications, doivent être améliorées. La spécification est la clé de voûte de l'outil final.
- La planification des développements : doit être très précise et très réactive afin d'anticiper les pics de charge.

Je m'attacherai donc dans cette partie à décrire ces deux processus et à proposer des solutions d'améliorations.

Le dessin humoristique suivant résume les principaux problèmes rencontrés lors du déroulement d'un projet

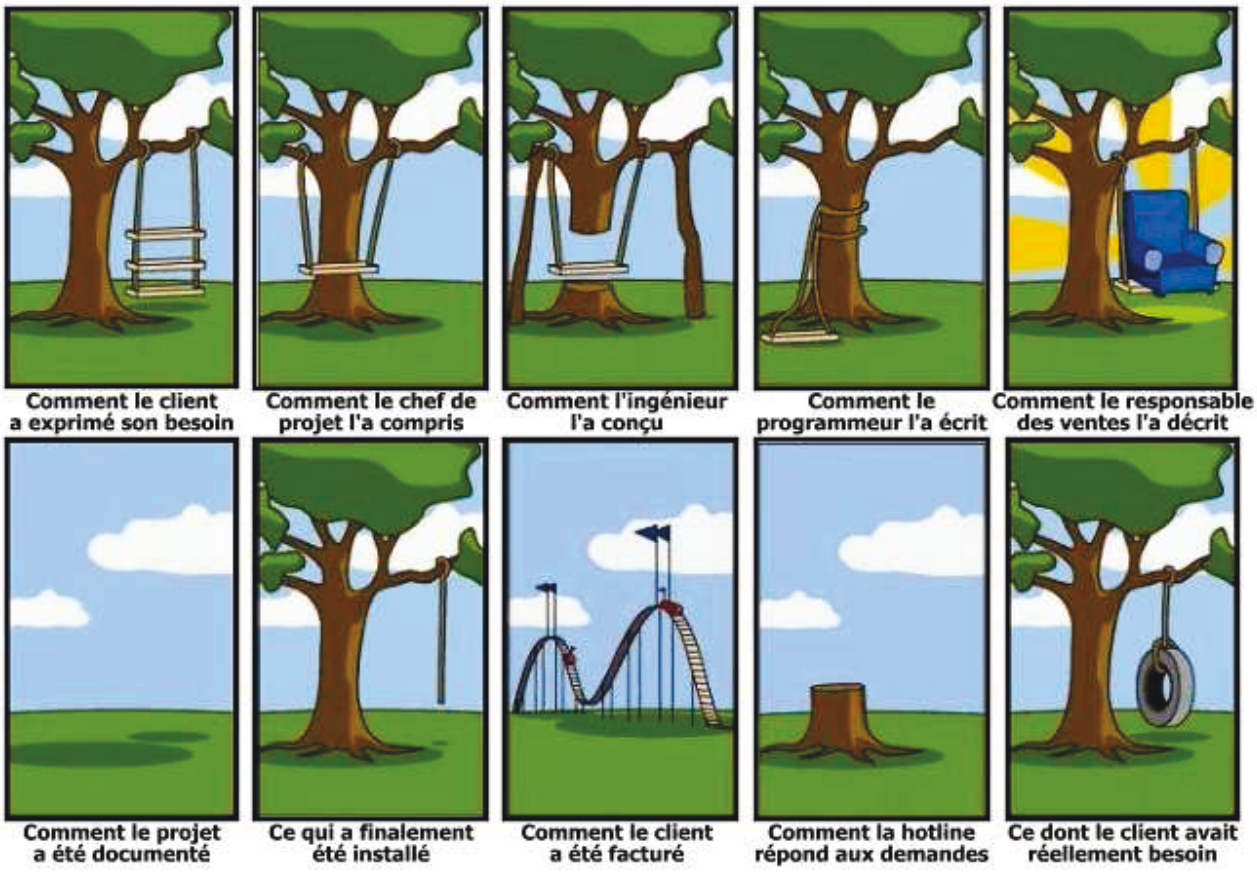


Figure 20 : Les problèmes de la gestion de projet

- Le client exprime mal son besoin
- Le produit final ne correspond ni aux besoins, ni à ce qui a été conçu initialement
- La documentation est souvent inexistante

-

11.2 DÉFINITION DU BESOIN

La définition du besoin est la première étape de la réalisation du projet. C'est le moment où le client exprime concrètement ses attentes, au moyen d'une spécification fonctionnelle. Plusieurs problèmes se posent :

- Le premier est la difficulté qu'a le client à exprimer précisément ce qu'il souhaite.
- Le second est la capacité du chef de projet informatique à comprendre la problématique de son client.

C'est avant tout un problème de communication entre deux mondes différents. Le client a une très bonne connaissance de son métier et de ce qu'il souhaite, mais a du mal à l'exprimer sous la forme d'un outil informatique. C'est dans ce contexte qu'intervient généralement un expert métier qui vient assister ces différents acteurs dans la rédaction de la spécification.

La difficulté est de déterminer un langage commun entre les différents métiers. Il s'agit de trouver le meilleur moyen de traduire un besoin fonctionnel en outil informatique. De nombreuses solutions de modélisation ont été proposées au fil du temps, la plus connue étant certainement UML.

La spécification est le document de référence sur lequel va s'appuyer l'outil final ; ce document, mal interprété, peut être une source d'incompréhension et de conflits entre le client et le chef de projet.

Nous allons dans cette partie étudier les différentes phases d'analyse du besoin menées par le groupe SI afin de déterminer nos points faibles et de proposer des axes d'améliorations à cet égard.

11.3 ANALYSE FONCTIONNELLE ET REDACTION DES SPECIFICATIONS FONCTIONNELLES ET TECHNIQUES.

11.3.1 QU'EST-CE QUE L'ANALYSE FONCTIONNELLE ?

En informatique, l'analyse fonctionnelle consiste à étudier et lister l'ensemble des fonctions d'un futur outil en vue de répondre aux demandes du client. Ce travail initial d'analyse est primordial. De sa qualité dépendra en grande partie l'adéquation de l'outil final avec le besoin.

11.3.2 QU'EST-CE QUE LA SPECIFICATION FONCTIONNELLE ?

La spécification fonctionnelle est la formalisation du travail d'analyse sous la forme de documents, schémas, etc. Elle est totalement indépendante du produit final et de la technologie utilisée. Il n'est pas rare que, dans les projets les plus importants, le travail de spécification et le travail de réalisation soient effectués par deux équipes différentes. Même si, a priori, ce travail ne requiert pas de compétences purement techniques, il est important que le chef de projet informatique puisse intervenir afin de déterminer ce qu'il est possible de faire, techniquement parlant. Par exemple, il arrive que certaines demandes ne soient pas réalisables immédiatement, à cause de restrictions techniques. Dans le cas du projet DS3, le client souhaitait afficher sur le même écran la totalité d'un catalogue de pièces (100 000 lignes) ; cette demande, non réalisable compte tenu de l'architecture technique, a été remplacée par un affichage paginé, et un écran de recherche.

11.3.3 QU'EST-CE QUE LA SPECIFICATION TECHNIQUE

Cette spécification décrit les aspects techniques du futur produit. Ce document présente notamment, la technologie utilisée et l'architecture du futur logiciel. La spécification technique est rédigée par le groupe SI, en s'appuyant sur la spécification fonctionnelle. La technologie est choisie en fonction des contraintes imposées par le client

11.3.4 ANALYSE FONCTIONNELLE ET SPECIFICATIONS (PROJET DS3)

Le projet DS3 a pour but de remplacer, et d'améliorer les fonctions d'un outil Access, pour lequel il existait déjà une spécification fonctionnelle. Il a donc été décidé de se fonder sur ce document en corrigeant et ajoutant les points manquants. Cette étape a duré environ deux mois. Au terme de cette phase la spécification a été relue et modifiée avec le client. Nous avons choisi d'écrire une spécification fonctionnelle et technique, décrivant autant les process métiers que les solutions techniques choisies par le chef de projet SI.

11.3.5 ANALYSE FONCTIONNELLE ET SPECIFICATIONS (PROJET ILSDB)

Ce travail de spécification a été beaucoup plus approfondi que pour le projet DS3, du fait du périmètre fonctionnel nettement plus important et de la complexité du projet. La spécification a été rédigée par le groupe SI, assisté d'un chef de projet métier. Nous avons connu certaines difficultés lors de cette phase initiale. La principale a été le manque de connaissances par l'équipe de développement des processus métiers de notre client.

Le temps d'analyse et de rédaction a été très mal estimé : Il était prévu de passer cinq jours sur ce travail, et au final nous avons travaillé une vingtaine de jours. Ce retard important, dès la phase de spécification, associé à un environnement technique complexe a fait que la première version de l'outil a été livrée bien après la date prévue initialement.

11.4 AXES D'AMELIORATIONS POUR LA REDACTION DES SPECIFICATIONS TECHNIQUES ET FONCTIONNELLES

Nous avons retenu plusieurs axes d'améliorations pour l'analyse du besoin et la rédaction des spécifications fonctionnelles :

11.4.1 STANDARDISATION DU LANGAGE DE MODÉLISATION

Comme cela a été expliqué dans l'introduction, toute la difficulté du travail d'analyse est de trouver un langage commun entre les utilisateurs de la future application et les équipes chargées de sa réalisation. Les schémas de processus des projets ILSDB et DS3 ont été réalisés sans utiliser de méthodes particulières. Nous utilisons en revanche sur tous nos projets la méthode MERISE pour modéliser les bases de données.

La spécification du projet DS3 est issue de la spécification d'un outil plus ancien, utilisant la plate-forme Microsoft Access. Nous avons choisi, eu égard à la taille peu importante du projet, de ne pas reécrire entièrement la spécification mais d'amender le document existant : ce choix est une erreur. La spécification finale, trop imprécise, a entraîné de nombreuses erreurs et donc des retards, une insatisfaction du client et une perte financière non négligeable.

Utiliser un langage de modélisation nous permet de définir clairement le besoin avec les utilisateurs. C'est en outre un gage de la qualité de notre travail. En utilisant un langage standard, le travail pourra être réutilisé par n'importe quelle équipe maîtrisant son utilisation.

La décision d'utiliser un tel langage ayant été prise, il restait à choisir lequel : ce fut UML, standard de l'industrie, modèle très largement utilisé, ce qui facilite sa compréhension par un maximum de personnes. Ce langage de modélisation fondé sur des pictogrammes, est apparu dans le monde du génie logiciel en même temps que la "conception orientée objet (POO)". UML est la fusion de plusieurs langages de modélisation. C'est à présent un standard défini par l'Open Management Group (OMG).

UML est utilisé pour modéliser une application informatique. Cette modélisation se fait sous la forme de diagrammes. Les différents éléments représentables sont :

- Activité d'un objet/logiciel
- Acteurs
- Processus
- Schéma de base de données
- Composants logiciels
- Réutilisation de composants

UML n'est pas une méthode mais un ensemble d'outils que l'on choisit, pour modéliser tout ou partie d'un logiciel ; treize diagrammes sont disponibles, répartis en plusieurs sous-ensembles : vues, diagrammes, modèles d'éléments.

Tous les projets postérieurs aux projets DS3 et ILSDB ont utilisé ce modèle. L'amélioration significative de nos spécifications, associée à une meilleure maîtrise de la technologie a entraîné une meilleure performance de notre travail. Cette montée en compétence a nécessité un investissement important en termes de formation des équipes. Je n'avais pour ma part utilisé le langage UML que durant ma formation et n'avais jamais eu l'occasion de le mettre en pratique. D'autres formations UML sont prévues dans les mois à venir.

11.4.2 FORMATION AUX MÉTIERS CLIENTS

Nous nous sommes aperçus que certains membres de l'équipe, dont moi, ne connaissaient pas ou mal les métiers de leur client. Cette mauvaise connaissance du métier est préjudiciable, notamment lors de la phase d'analyse du besoin et de rédaction des spécifications.

11.5 PLANIFICATION DES DÉVELOPPEMENTS

11.5.1 PLANIFICATION ET SUIVI DES DEVELOPPEMENTS

Nous allons dans cette partie aborder la question de la planification des développements et voir dans ce chapitre comment est organisée cet activité, à partir de la spécification.

Qu'est-ce qu'un plan de développement ?

La planification des développements est l'activité qui consiste - à partir d'une spécification technique - à organiser concrètement le travail. C'est donc une partie purement technique et opérationnelle. Le chef de projet détermine les fonctions à développer et leur affecte des ressources.

La réalisation du plan de développement est une activité complexe. Il s'agit de répartir le travail, en évitant les pics ou les creux de charge, et en respectant les délais impartis. Ce plan est complémentaire du planning général du projet.

Qu'est-ce que le suivi des développements ?

Le suivi des développements consiste à contrôler au jour le jour le bon déroulement du plan de développement, et à anticiper- les retards ou les avances - afin de piloter au mieux l'activité. La difficulté est de prévoir suffisamment à l'avance les modifications pour prendre les décisions adéquates.

Nous étudierons en détail dans un premier temps comment ont été menés les développements sur les projets ILSDB et DS3, avant de proposer des axes d'améliorations.

11.5.2 OUTIL DE PLANIFICATION DES PROJETS UTILISE PAR LE GROUPE SI

Comme nous l'avons mentionné dans la partie analyse de l'existant, nous avons souhaité, à la suite des dérives constatées lors du projet DS3, mettre en place une solution de gestion de projet. Nous avons donc étudié les différentes solutions existantes.

11.5.3 CRITÈRES DE SÉLECTION

La solution retenue devra satisfaire aux exigences suivantes :

- Compatibilité avec notre plate-forme de développement : la solution choisie devra être intégrée à l'environnement de développement Eclipse. Les demandes d'évolutions devront être visibles et modifiables depuis ce logiciel.
- Gestion de plusieurs projets : Il faut dans ce cas que l'outil de planification gère simultanément et distinctement les différents projets, avec les droits d'accès correspondants. Un utilisateur peut être par exemple administrateur d'un projet, avec les droits afférents, et développeur d'un autre projet.
- Gestion des développements et des bugs : L'outil doit traiter deux types de demandes : celles de développement, émises par le chef de projet, suivant le plan défini, et les demandes de corrections suite à un bug d'une fonctionnalité existante.
- Solution de partage d'informations : Dans le cadre de l'amélioration de nos processus, il est nécessaire de posséder un outil de partage d'informations techniques. Cette solution peut prendre par exemple la forme d'un wiki.
- Coûts d'achat, d'installation et de maintenance raisonnables : La taille des projets de développements sur lesquels nous intervenons étant modeste, nous privilégions aujourd'hui une solution open-source, quand elle existe.

11.5.4 OUTILS EXISTANTS SUR LE MARCHÉ

Nous avons évalué les outils suivants :

Trac

Trac est une application web open source de gestion de projet écrite en Python. Elle propose les fonctionnalités suivantes :

- Wiki
- Historique,
- Rapport de bugs
- Explorateur subversion

J'ai eu l'occasion d'utiliser Trac sur un projet Python. L'utilisation au quotidien est simple, cependant la configuration initiale est relativement complexe, à cause du grand nombre de fonctions et de plugins disponibles. Trac est supporté par une large communauté d'utilisateurs.

Redmine

Tout comme Trac, Redmine est une application web open source de gestion de projet écrite en Python qui propose les fonctionnalités suivantes

- Wiki
- Historique,
- Rapport de bugs
- Explorateur subversion, CVS, Bazaar
- Identification LDAP
- Support de plusieurs bases de données
- Notification par courrier
- Multilingue
- Forums multi-projet

Nous utilisons Redmine à des fins de tests depuis le début du projet ILSDB. L'interface est plus moderne que Trac, et il n'y a aucune configuration supplémentaire à effectuer.

Redmine a cependant un défaut important : La connexion avec Microsoft Project ne fonctionne pas correctement.

Jira

Jira est un système de suivi de bug développé par la société Atlassian Software System. L'application est gratuite pour les projets open-source. Nous l'utilisons actuellement dans le cadre d'un projet interne, en vue de remplacer éventuellement Redmine. Jira présente l'avantage de pouvoir être hébergé par la société Atlassian pour un coût modique.

Nous avons donc décidé de conserver pour le moment notre plate-forme Redmine, tout en continuant de tester l'application Jira.

12 CONCLUSION

L'amélioration des processus repose donc sur trois axes majeurs :

- **Mieux comprendre le besoin** : Utiliser un langage de modélisation et former les équipes de développements aux métiers de leurs clients.
- **Améliorer la gestion de projet et le suivi des activités** : Mettre en place un outil de suivi et de planification des tâches.
- **Mettre l'accent sur les améliorations techniques** : Normes de conception, analyse statique de code, procédures de tests, plate-forme de développements, etc…

Ces trois piliers sont absolument indissociables. Un logiciel, ou un système d'information, n'est pas simplement composé de codes. C'est un ensemble de contraintes, d'idées, de personnes qu'il faut coordonner pour répondre à un besoin :

LA COMPREHENSION DES ATTENTES EST LE CŒUR DE L'INFORMATIQUE DE GESTION

L'amélioration des processus de développement requiert en premier lieu une amélioration de la compréhension des processus métiers. C'est un travail infiniment plus complexe que le développement logiciel qui doit s'inscrire dans un cercle vertueux : un meilleur produit ou service entraine une reconnaissance, et donc potentiellement une augmentation du chiffre d'affaire. L'équation du succès, simple à formuler, est plus compliquée à résoudre : satisfaire du mieux possible les clients, tout en conservant une rentabilité et une croissance suffisante pour poursuivre le développement.

Nous n'avons pas abordé dans ce travail l'aspect managérial et relationnel. L'amélioration ne peut en effet se faire sans une forte motivation des équipes. Tous les efforts sont vains si ces dernières ne se sentent pas sérieusement impliquées.

Sur un plan personnel, ce travail a été très enrichissant. J'ai eu l'occasion d'expérimenter des idées, d'explorer des pistes nouvelles. Si la gestion de projet est un sujet ancien, les idées que je présente ne sont pas totalement novatrices. La difficulté plus immédiate se trouve dans la recherche de solutions adaptées à notre environnement et à nos clients. Il ne s'agit en aucun cas d'appliquer des « recettes » toutes faites. Il faut apprendre à travailler à partir d'une « feuille blanche », réussir à sortir du cadre très strict du développement pour tenter d'offrir des propositions innovantes.

Il convient de noter que la qualité et la réussite de nos derniers projets est sans commune mesure avec ce que nous faisions il y a deux ans. Il reste cependant beaucoup à faire, et les propositions présentées dans ce mémoire ne sont que les prémices du travail qui nous attend dans les années à venir : Utilisation de méthodes agiles, architecture orientée services …

13 GLOSSAIRE

Access : logiciel de gestion de base de donnée commercialisé par Microsoft

ACPL : All Customer price list

AMOA : Assistance à la maîtrise d'ouvrage

AMOE : Assistance à la maîtrise d'œuvre

C# : Langage de programmation commercialisé par Microsoft

CA : Chiffre d'affaire

CEA : Commissariat à l'énergie atomique

CMMI: Capability Maturity Model Integration

CPL: Customer price list

DoD : Department of defense (Armée des États-Unis)

DS3 : Outil de gestion développé par LGM pour la société NH Industries

DSI : Direction des systèmes d'information

DTI : Direction technique de l'innovation

EDI : Environnement de développement intégré

Framework : Bibliothèque de composants logiciels

GED : Gestion électronique de données

Groupe SI : Groupe système d'information

IHM : Interface homme machine

ILSDB : base de données LSA développé par LGM pour Eurocopter

IRFM : Institut de recherche pour la fusion magnétique

JUNIT : Bibliothèque de tests unitaires pour Java

Gdp : Gestion de projet

Java : langage de programmation

Java EE : Java Enterprise Edition

Javadoc : Documentation associée au langage Java

Jquery : Framework de développement JavaScript

Jsf : Java Server Faces. Framework de développement Java

KDE : K Desktop environnement. Gestionnaire de bureau utilisé sous Linux

LCC : Life cycle cost

Linux : Système d'exploitation libre.

LSA : Line support activity

Merge : Action de fusionner deux fichiers textes.

MCO : Maintien en condition opérationnelle

MVC : Modèle vue contrôleur

NATO/OTAN : Organisation du traité de l'atlantique nord

NH90 : Hélicoptère militaire de l'OTAN

NHI : NH Industries

O.S: Operating system

OMG: Open Management group

PC: Partner Company

PHP : Langage de programmation web libre

PMD : Framework d'analyse de code source Java

PMI : Project Management Institut

POO : Programmation orientée objet

Primeface : Framework de développement Java

Python : Langage de programmation

RAD/DRA : Rapid application development, ou DRA en français.

Redmine : Logiciel de suivi des tâches

Scampi: Standard CMMI Appraisal Method for Process Improvement

Sdf : Sûreté de fonctionnement

SI : Système d'information

Sql Server : Système de gestion de base de données relationnelle

SLI : Soutien logistique intégré

SI : Système d'information

SSII : Société de service en informatique

Struts : Framework de développement Java

SVN : Subversion. Logiciel de gestion de version utilisé dans le développement informatique

Tech pub: Technical publication

UML: Unified Modeling language

14 BIBLIOGRAPHIE

14.1 SITES INTERNET

Qualité du logiciel

- Qualité du logiciel, Yves Constantinidis, 03 décembre 2011
http://www.yves-constantinidis.com/doc/yves-758.htm

Suivi d'activité

- Découverte de Redmine, blog Epita, Pierre-Lou Dominjon, 25 Février 2011
http://www.mti.epita.fr/blogs/2011/02/25/decouverte-de-redmine-un-outil-simple-et-puissant-de-gestion-de-projet/

Environnement de développement

- Les meilleurs RAD et EDI Java, developpez.com, 05 Avril 2011
http://Java.developpez.com/outils/edi/?page=advanced

CMMI

- Capacity maturity model integration, Wikipedia, janvier 2012
http://fr.wikipedia.org/wiki/Capability_Maturity_Model_Integration
- CMMI Frequently asked question, cmmifaq.info, 9 octobre 2011

  http://www.cmmifaq.info/

Développement

- Utilité des normes de codage, developpez.com, 27 mars 2009
http://hugo.developpez.com/tutoriels/genie-logiciel/utilite-normes-codage/
- Style guide for Python code, Python.org 24/04/12

  http://www.Python.org/dev/peps/pep-0008/
- Code Conventions for the Java TM Programming Langage, Oracle, mai 1999

  http://www.oracle.com/technetwork/Java/codeconvtoc-136057.html
- Java, convention de nommage, Loribel.com

http://www.loribel.com/Java/normes/nommage.html

- Vérifier votre code Java avec PMD, Unixgarden, romain Pelisse, 17 mars 2009

http://www.unixgarden.com/index.php/gnu-linux-magazine/verifier-votre-code-Java-avec-pmd

- Java theory and practice: I have to document THAT, Brian Goetz, 1 About 2002

Http: //www.IBM.com/developerworks/Java/library/j-jtp0821/index.html

- Conventions de syntaxe en Python, all4dev, 2004

http://all4dev.libre-entreprise.org/index.php/Conventions_de_syntaxe_en_Python

- Qualité du logiciel, Alain Tisserand, École des mines de Nancy

http://www.mines.inpl-nancy.fr/~tisseran/cours/qualite-logiciel/index.html

- Qualité logiciel, Wikipedia, 2012

http://fr.wikipedia.org/wiki/Qualit%C3%A9_logicielle

- Analyse statique de programme, Wikipedia, 2012

http://fr.wikipedia.org/wiki/Analyse_statique_de_programmes

- ISO 9126, Wikipedia, 2012

http://fr.wikipedia.org/wiki/ISO_9126

- Management du système d'information, Wikipedia, 2012

http://fr.wikipedia.org/wiki/Informatique_de_gestion

- Java logging Framework, Wikipedia, 2012

http://en.wikipedia.org/wiki/Java_Logging_Frameworks

- Gestion de la configuration logicielle, Wikipedia, 2012

http://fr.wikipedia.org/wiki/Gestion_de_configuration_logicielle

14.2 LIVRES ET DOCUMENTS

Rédaction du mémoire

- Guide de rédaction et de présentation des thèses. - Ed. Ministère de l'Education Nationale, 2007

Développement

- Guide Francophone des conventions de nommage en langage Java. Hugo Etievant, 2004
- Amélioration de la qualité logicielle par l'analyse statique de code Java. Jeremie Guidoux, 03 septembre 2009

Qualité du logiciel

- Software quality requirement and evaluation. PSM technical working group, 2004
- Measuring software product quality : ISO 25000 Series and CMMI. Dave Zubrow, 2004
- Guide pour l'ingénierie des exigences avec UML et d'autres techniques, centre de recherche public Henri Tudor (Luxembourg), 2004

Méthodologie et qualité

- Togaf Version 9, Andrew Josey. - the open group, 2009
- CMMI pour le développement logiciel, version 1.2. - l'équipe produit CMMI, 2006
- Agilité et CMMI-DEV, Complémentarités utiles pour une organisation, Avril 2011
- Version control with Subversion , Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato, 2008