HAL
open science

Étude des solutions d'amélioration de la Maintenance Applicative par l'intégration de logiciels open source en entreprise

Stéphane Glattleider

To cite this version:

Stéphane Glattleider. Étude des solutions d'amélioration de la Maintenance Applicative par l'intégration de logiciels open source en entreprise. Génie logiciel [cs.SE]. 2011. dumas-00574247

HAL Id: dumas-00574247

https://dumas.ccsd.cnrs.fr/dumas-00574247v1

Submitted on 7 Mar 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Conservatoire National des Arts et Métiers
Paris

MEMOIRE

Présenté en vue d'obtenir le

DIPLOME D'INGENIEUR DU CNAM

en

INFORMATIQUE OPTION DES

SYSTEMES D'INFORMATION

Par

Stéphane GLATTLEIDER

Etude des solutions d'amélioration de la Maintenance Applicative par l'intégration de logiciels open source en entreprise

Soutenu le 21/01/2011

Jury:

PRESIDENT : I. WATTIAU (CNAM)

MEMBRES : N. LAMMARI (CNAM)

J. AKOKA (CNAM)

L. MOUKOUKENOFF (Paritel)

P. SUAEZ (Paritel)

REMERCIEMENTS

Mes premiers remerciements s'adressent à Monsieur Ludovic MOUKENNOF directeur des systèmes d'information du groupe Global Concept qui m'a accordé sa confiance pour la réalisation de cette étude durant ma mission de prestataire au sein de son entreprise.

Ensuite, je tiens à remercier les personnes qui m'ont conseillé et soutenu durant l'écriture de cette étude de cas, en particulier

- Frédéric DELAUNAY, Jean Damien MOTTOT et Anthony MARECHAL pour leur patience et leurs judicieux conseils techniques.
- Michèle COLLET GLATTLEIDER et Liliane MESSIKA pour leur rôle ingrat et difficile de premières lectrices.

Enfin, un dernier remerciement au « Conservatoire National des Arts et Métiers » pour l'enseignement suivi et plus particulièrement à Monsieur Jacky AKOKA, pour m'avoir accompagné avec bienveillance durant toute la durée de ce mémoire.

RESUME

La société Viatelease du groupe Global Concept recherche des solutions fiables et économiques pour améliorer la qualité de ses processus de maintenance applicative. Dans ce cadre, la direction informatique a souhaité mettre en œuvre les logiciels open source Mantis de suivi d'anomalies et MediaWiki de gestion des connaissances. Ce mémoire se présente comme un cas d'étude dans lequel nous allons mesurer la capacité de ces logiciels open source à s'adapter et à s'intégrer dans un système d'information d'entreprise. Après avoir défini les processus de maintenance cibles, nous présenterons leur installation, l'amélioration de leurs fonctionnalités et leur interconnexion avec d'autres applications. Enfin, nous confronterons les aspects techniques de ce projet aux résultats d'une enquête de retour d'expérience.

Mots clés : Mantis, MediaWiki, Open Source, Maintenance applicative, Cas d'étude, Retour d'expérience

SUMMARY

Viatelase, a company belonging to Global Concept group, researches reliable and economical solutions to improve the quality of its application maintenance process. In this framework, the IT department decided to implement open source software: Mantis for bug tracking, and MediaWiki for knowledge management. This thesis appears as a case study in which we will measure the ability of these open source software to be adapted and integrated into a corporate information system. After defining the target maintenance processes, we will explain how they were installed, how their functionalities were improved and how they were interconnected with other applications. Eventually, we will compare the technical aspects of this project with the results of a feedback survey.

Key words: Mantis, MediaWiki, Open Source, Application Maintenance, Case Study, Feedback Survey.

TABLE DES MATIERES

INTRODUCTION **6**

CHAPITRE 1 - PRESENTATION DE L'EXISTANT **9**

1.1 PRESENTATION DE VIATELEASE 9

1.1.1. Présentation des activités de Viatelease *9*

1.1.2. Présentation du Système d'Information de Viatelease *10*

1.1.3. Présentation de la stratégie de Viatelease *11*

1.2 PRESENTATION DE LA DSI 13

1.2.1. Présentation de l'organisation de la DSI *13*

1.2.2. Projets de la DSI *14*

1.2.3. Les besoins *15*

1.3 LES SOLUTIONS CIBLES 17

1.3.1. Processus de maintenance existant *17*

1.3.2. Cartographie applicative existante *18*

1.3.2.1 Altiris 19

1.3.2.2 Dotproject 19

1.3.2. Processus de maintenance applicative cible *20*

1.3.2.1 Processus de gestion des demandes de résolution 20

1.3.2.1 Processus de gestion des projets de maintenance applicative 23

1.3.3. Cartographie applicative cible *25*

1.3.3.1 Mantis 25

1.3.3.2 Altiris - Mantis 26

1.3.3.3 Mantis - MediaWiki 28

CHAPITRE 2 - ETUDE SUR L'AJOUT DE BRIQUES APPLICATIVES OPEN SOURCE30

2.1. METHODE ET OUTILS D'INTEGRATION DES APPLICATIONS OPEN SOURCE 30

2.1.1. Gestion des versions : Subversion *30*

2.1.2. Modification des fichiers de code sources : Eclipse *31*

2.1.3. Manipulation des données : PhpMyadmin *32*

2.2. ÉTUDE SUR L'AJOUT D'UNE SOLUTION D'AMELIORATION DU SUIVI DES ANOMALIES : MANTIS BUG TRACKER 34

2.2.1. Installation de Mantis *34*

2.2.1.1. Architecture technique 34

2.2.1.2. Description des flux sur le serveur SVN 35

2.2.2. Paramétrage de Mantis *36*

2.2.2.1. Personnalisation des demandes 36

2.2.2.2. Personnalisation de la gestion des utilisateurs/acteurs 41

2.2.2.3. Gestion des notifications 43

2.2.3. Modification du code source de Mantis *45*

2.2.1.1. Présentation du code source de Mantis 45
2.2.1.1. Ajout d'un calendrier de saisie de date 46
2.2.1.2. Modification du workflow de statuts 49
2.3. ÉTUDE SUR L'AJOUT D'UNE SOLUTION DE GESTION DES CONNAISSANCES : MEDIAWIKI 53
2.3.1. Installation de MediaWiki 53
2.3.1.1 Architecture technique 53
2.3.2. Paramétrage de MediaWiki 54
2.3.2.1 Principes d'organisation des connaissances 54
2.3.2.2 Adaptation de l'interface 55
2.3.2.3 Ajout d'un éditeur convivial 58
2.3.3. Modification du code source de MediaWiki 60
2.3.3.1. Modification du code de localsetting.php : sécurisation des données de production 60
2.3.3.2. Modification linker.php : amélioration de la consultation des images 62
CHAPITRE 3 - ETUDE DE L'INTERCONNEXION DES APPLICATIONS OPEN SOURCE 65
3.1. INTEGRATION AVEC L'ANNUAIRE DE L'ENTREPRISE : ACTIVE DIRECTORY 65
3.1.1. Objectif de l'intégration avec l'annuaire Active Directory 65
3.1.2. Architecture Active Directory - Mantis - MediaWiki 66
3.1.2.1. Architecture Active Directory existante 66
3.1.2.2. Architecture Active Directory cible 67
3.1.2.3. Intégration de Mantis 67
3.1.2.3. Intégration de MediaWiki 71
3.2. INTERCONNEXION DE MEDIAWIKI ET MANTIS 72
3.2.1. Valeur ajoutée de la solution 72
3.2.2. Réalisation de la solution 74
3.2.2.1 Analyse de l'extension MantisIntegration.php 74
3.2.2.2 Modification du code de l'extension 77
3.2.3. Limites de la solution 79
3.3. RETOUR SUR EXPERIENCE 80
3.3.1. Méthode de retour d'expérience 80
3.3.2. Exploitation des résultats de retour d'expérience de Mantis 83
3.3.3. Exploitation des résultats de retour d'expérience de MediaWiki 84
CONCLUSION 86
INDEX DES FIGURES 89
INDEX DES EXTRAITS DE CODE 90
BIBLIOGRAPHIE 92

Introduction

Traditionnellement les services informatiques connaissent deux types d'activités : d'un côté, le développement et l'intégration de nouvelles applications dans l'entreprise, de l'autre, la maintenance des applications et des infrastructures des systèmes d'information. La distinction entre ces deux grandes activités repose essentiellement sur la notion de gestion de projet. En effet, si toutes les activités de l'entreprise peuvent être décrites par un ou plusieurs processus, un projet est caractérisé par un début et une fin. Dans le modèle conceptuel de gestion de projet en «V», modèle standard pour l'industrie logicielle des années 1980, le projet débute par une analyse des besoins et de la faisabilité et s'achève par la vérification d'aptitude au bon fonctionnement (recette). Chaque projet a sa propre temporalité. A l'opposé, la temporalité des activités de maintenance se confond avec la temporalité de l'entreprise, à condition, bien sûr, que cette dernière utilise des logiciels informatiques. Du point de vue interne à l'entreprise, les activités de maintenance n'ont pas de début ni de fin. Elles sont continues.

Les activités de maintenance peuvent être découpées en plusieurs sous-catégories. La maintenance corrective assure la correction des défauts de fonctionnement ou des non conformités des applications. La maintenance adaptative consiste à adapter les applications en cas de changement d'environnement technique ou de système ou en cas de survenue de nouveaux besoins fonctionnels. On la désigne également sous le terme de « maintenance évolutive ». La maintenance préventive, elle, est destinée à réduire la probabilité de défaillance ou la dégradation du fonctionnement. Enfin, le maintien des connaissances nourrit chacune de ces sous-catégories et assure une amélioration continue des processus qui les sous-tendent. A l'instar de la gestion de projet, chacune de ces activités de maintenance peut être décrite par un processus, autrement dit par une succession de traitements structurés et ordonnés, ayant chacun un objectif et des résultats définis à l'avance. Par exemple, la chaîne de correction d'une application peut être décrite ainsi : d'abord la réception et la qualification de l'anomalie. Une analyse est réalisée jusqu'à l'identification d'une solution corrective. Puis, cette solution est mise en œuvre. Enfin, la correction est diffusée dans l'ensemble des systèmes d'information de l'entreprise. Si les activités de maintenance sont continues, elles sont également structurées en processus plus ou moins compliqués.

Une étude récente du SYNTEC informatique rapportait une augmentation régulière, entre 7 et 10%, des demandes d'infogérance de maintenance applicative depuis 2006. Cette augmentation

s'explique en partie par le souci constant des Directeurs de Système d'information de réduire leurs coûts, notamment en sous-traitant une partie de leurs activités à des sociétés de services informatiques aux tarifs attractifs, ainsi que par la difficulté croissante, pour les ressources internes, de réaliser elles-mêmes les activités de maintenance. Complexité des technologies hétérogènes et parfois vieillissantes, complexité des systèmes d'informations souvent organisés sur des principes historiques plutôt que logiques, enfin complexité des processus de maintenance qui s'organisent de plus en plus en projets ; la maintenance applicative devient un mélange de genres dont la mise en œuvre nécessite une expertise.

Ma mission pour la société Viatelease consiste à fournir cette expertise et à atteindre les objectifs suivants : définir une organisation fonctionnelle, décrire les processus adaptés et mettre en œuvre les solutions applicatives nécessaires pour assurer l'efficience des activités de maintenance applicative. L'efficience étant définie comme le rapport coût/efficacité lié à la réussite d'une activité, la recherche de logiciels open source gratuits a été privilégiée, à condition qu'ils répondent aux objectifs définis.

L'appellation « open source » s'applique aux logiciels dont la licence respecte des critères établis par l'Open Source Initiative, notamment la possibilité de libre redistribution et d'accès au code source. La modification du code source permet à la fois d'enrichir les fonctionnalités et de les adapter du logiciel aux besoins des utilisateurs. Ce choix s'inscrit bien dans une démarche d'ingénierie logicielle où les besoins des utilisateurs sont définis indépendamment des solutions déjà existantes. Il présente l'avantage de fournir un socle et de réduire l'effort de réalisation. Toutefois, si les fonctionnalités attendues venaient à trop s'écarter des fonctionnalités prévues d'origine, les bénéfices de la réutilisation du code source disparaitraient quand bien même le logiciel serait gratuit. La direction serait dans l'obligation de revenir à l'arbitrage habituel entre le développement d'une solution spécifique et l'achat d'une solution payante chez un éditeur spécialisé.

L'enjeu des travaux que j'ai réalisés pour Viatelease est de vérifier que le choix de logiciels open source est pertinent pour améliorer la maintenance applicative. Il s'agit d'étudier l'effort de modification du code source pour enrichir les fonctionnalités de base et interagir avec les autres applications, open source ou non, du système d'information. Si les conclusions de cette étude s'avèrent positives, la démarche et les outils seront étendus à l'ensemble du groupe Global Concept dont Viatelease est une des 7 filiales.

Le chapitre 1 introduira Viatelease et sa direction informatique. Il s'agit de présenter l'existant et d'identifier les besoins auxquels devra répondre la nouvelle organisation ainsi que les processus cibles. Cette solution sera mise en perspective par rapport à la stratégie à moyen terme de

Viatelease et à la cartographie applicative de son Système d'Information. Enfin, l'organisation et les processus de maintenance cibles seront décrits afin de fournir un cadre référentiel à la suite de l'étude.

Le chapitre 2 sera consacré à l'examen de l'ajout des deux solutions open source : Mantis et MediaWiki. Après une description de la méthodologie et des outils de développement du code natif de ces applications, nous présenterons la réalisation de deux fonctionnalités majeures dans le process cible : un workflow adapté au traitement des projets de maintenance et l'organisation des informations d'une base de connaissance avec MediaWiki.

Le chapitre 3 s'intéressera à la capacité d'intégration de ces applications. Dans un premier temps, nous étudierons comment elles peuvent s'interfacer avec l'annuaire de l'entreprise. Dans un deuxième temps, nous examinerons dans quelle mesure Mantis et MediaWiki peuvent s'interconnecter entre elles afin de fournir une suite logicielle de maintenance. Enfin, nous établirons un retour d'expérience sur la valeur ajoutée de ces applications après leur mise en œuvre dans l'entreprise de notre cas d'étude.

Chapitre 1 - Présentation de l'existant

1.1 PRESENTATION DE VIATELEASE

1.1.1. Présentation des activités de Viatelease

Viatelease est une des filiales du groupe Global Concept, lui-même le premier intégrateur de solutions Télécom en France pour le Petites et Moyennes Entreprise. C'est une Broker/Leaser de contrats de crédit (location financière, location avec option d'achat, crédit bail) pour les services et matériels de télécommunication. Autrement dit, il achète des contrats de crédit à des agences commerciales internes (Paritel) et externes au groupe pour les revendre à des organismes bancaires (Broker) ou pour les financer lui-même (Leaser).

La chaine d'activité de Viatelease est la suivante :

- Une agence commerciale propose de vendre des produits de télécommunication, service et matériel, à une PME ou à une TPE. Il peut s'agir d'une affaire nouvelle dans le cas d'un nouveau client, d'une mutation si la vente implique le rachat d'un contrat, ou encore d'une adjonction si la vente porte sur l'ajout de service ou de matériel par rapport au contrat actuel.

- Une fois le type de contrat, le montant et les conditions financières définis entre l'agence et le client final, l'agence va demander un financement à Viatelease.

- Viatelease reçoit une demande de financement qu'il va soit réaliser lui-même afin de maximiser sa marge commerciale, soit négocier auprès d'organismes bancaires (leaser) afin de minimiser le risque tout en conservant une marge importante.

Chaque leaser partenaire de Viatelease reçoit des demandes de financement auxquelles il peut répondre par un accord, un refus ou une demande d'informations complémentaires.

Après l'accord trouvé entre Viatelease et un leaser, Viatelease communique à l'agence son accord. Celle-ci informe le client final qui signe le contrat de crédit.

Le schéma ci-dessus illustre les échanges des demandes entre les différents acteurs.

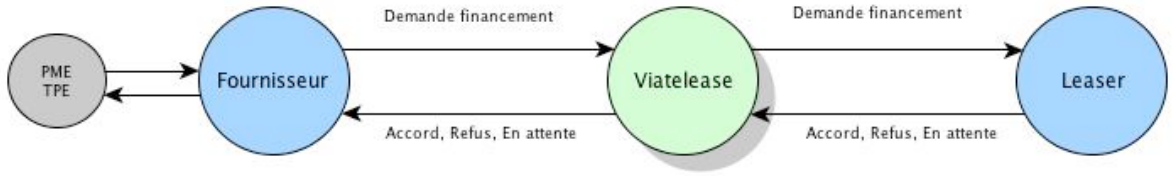


Figure 1 : Processus existant de gestion des incidents

Actuellement, Viatelease dégage 6,7 millions d'euros de bénéfice annuel avec la gestion de plus de 5000 contrats confiés seulement à 6 personnes :

- Un directeur général.
- 4 opérateurs de gestion des demandes.
- Une personne récemment engagée pour développer les relations avec les fournisseurs hors groupe, ce qui ne représente aujourd'hui que 5% des demandes de financements reçues.

Le directeur général de Viatelease relève directement du Président Directeur Général du groupe Global Concept.

1.1.2. Présentation du Système d'Information de Viatelease

Viatelease utilise essentiellement 2 applications métiers pour réaliser ses activités : SIV et Extranet. Ces deux applications ont été développées en interne par la direction informatique du groupe.

Le schéma ci-dessous présente l'architecture applicative de la société Viatelease.

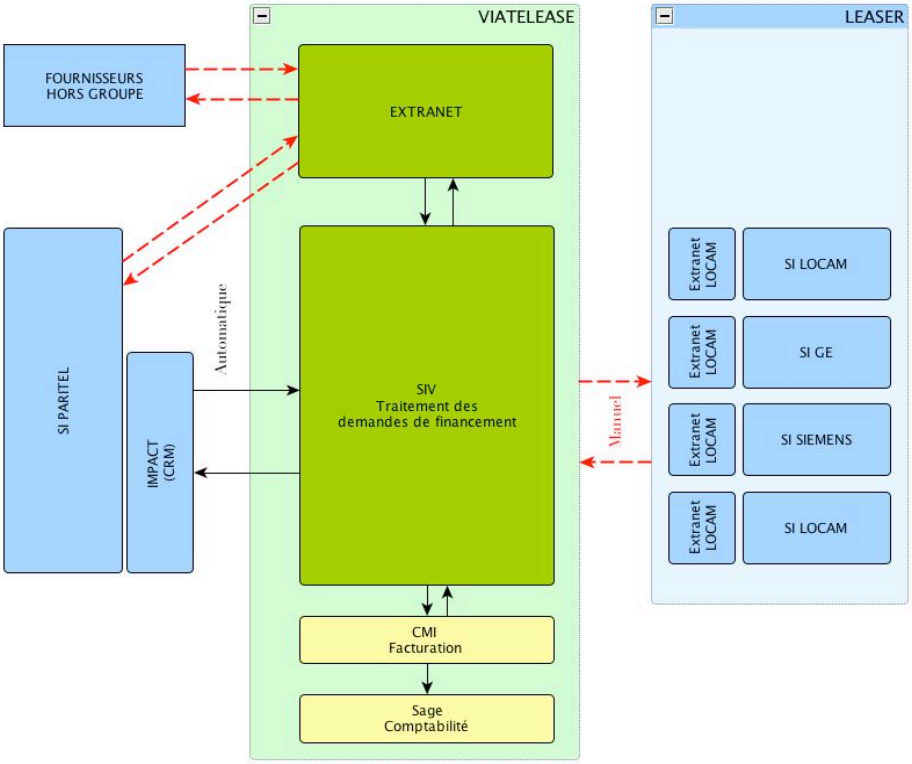


Figure 2 : Cartographie des applications de la société Ipnotic Optimitel

Extranet est une application développée pour faciliter les demandes de financement des fournisseurs hors groupe. Elle est également accessible pour les fournisseurs du groupe. Elle nécessite la saisie de la demande sur un site internet. Elle permet le suivi de l'évolution de la demande ainsi que le résultat de la demande : accord ou refus.

SIV est une application développée pour faciliter le suivi et le traitement des demandes de financement. Elle comporte des outils de filtre et de recherche ainsi qu'un historique des demandes refinanceurs effectuées pour chaque demande fournisseur, et les réponses obtenues. La transmission des demandes aux refinanceurs est manuelle. Les opérateurs doivent saisir les demandes sur les portails extranets des différents leasers.

Les applications CMI et Sage sont utilisées pour les activités de facturation et de comptabilité. Elles ne portent pas d'enjeux stratégiques métiers pour Viatelease.

Extranet et SIV ont été réalisés il y a trois ans pour remplacer le logiciel historique de gestion d'événements, Ubiqus, qui ne correspondait plus au métier de Broker/Leaser de Viatelease.

SIV et Extranet ont été développés en PHP en 6 mois par une équipe de trois développeurs internes sans cahier des charges fonctionnel ou technique. Il n'existe pas non plus de cahier de recette. Du reste, la mise en production de SIV a nécessité pour l'équipe informatique une reprise manuelle de l'existant de 18h à 5h du matin le jour de la mise en production. Cette reprise fut imparfaite et l'on trouve encore à ce jour des anomalies qui datent de plus de deux ans et qui sont à l'origine de nombreux incidents. Enfin, le code est peu commenté et il n'existe pas de description détaillée du schéma de base de données de l'application, ce qui achève de rendre la maintenance de ces applications difficile et les risques de régression importants

1.1.3. Présentation de la stratégie de Viatelease

Viatelease a défini 3 objectifs majeurs pour l'année 2010.

- Développer le Business to Business (B2B) avec les fournisseurs hors groupe.
- Augmenter la capacité de traitement du nombre de demandes fournisseurs par Viatelease.
- Industrialiser les échanges avec les refinanceurs partenaires actuels et futurs.

Ces trois objectifs se déclinent en deux projets informatiques qui sont : premièrement l'utilisation des services du site société.com pour compléter automatiquement les informations clientes nécessaires à la création d'une demande de financement à Viatelease ; deuxièmement, la création de services universels de demandes de financement aux leasers qui respectent les règles métiers actuelles de Viatelease (condition d'envoi, de réexamen, d'accord ou de refus).

Pour réaliser ces objectifs, les équipes informatiques ont été renforcées par un chef de projet, un développeur senior et un développeur junior. On notera que des projets similaires sont menés pour toutes les autres filiales de Global Concept, ce qui a pour conséquence de doubler l'effectif de la direction informatique et d'augmenter les risques de mise en production qui sont toujours groupées.

Pour les raisons qui ont été évoquées à la fin du paragraphe 1.1.2, des opérations de maintenance correctives et évolutives sont simultanément réalisées. Les corrections, comme les évolutions, nécessitent souvent la modification du code.

La maintenance des applications de SIV et Extranet est complexe car elle implique à la fois un processus de gestion de projet difficile de par le manque de documentation fonctionnelle et technique et un processus de correction d'anomalies continuel qui conditionne la production et tend à rendre instable le code source.

1.2 PRESENTATION DE LA DSI

1.2.1. Présentation de l'organisation de la DSI

La direction informatique est constituée des équipes de la société Optimitel. Néanmoins, le DSI est placé sous l'autorité du Directeur adjoint d'une autre filiale du groupe : Paritel.

Elle comporte 3 départements

- Le département d'infrastructure qui assure, avec 5 emplois temps plein, l'administration des systèmes et réseaux du groupe, la gestion du parc informatique et le support niveau 1, autrement dit la prise d'appel pour les utilisateurs du groupe.
- Le département de gestion de projets qui est chargé, avec 17 emplois temps, de la réalisation des projets logiciels de la plupart des applications métiers du groupe, ainsi que de leur support.
- Le département technique dont les 9 membres assurent le support de niveau 2 et 3 des équipements de télécommunication chez les clients finaux ainsi que le test des nouveaux modèles du catalogue produit vendu au client.

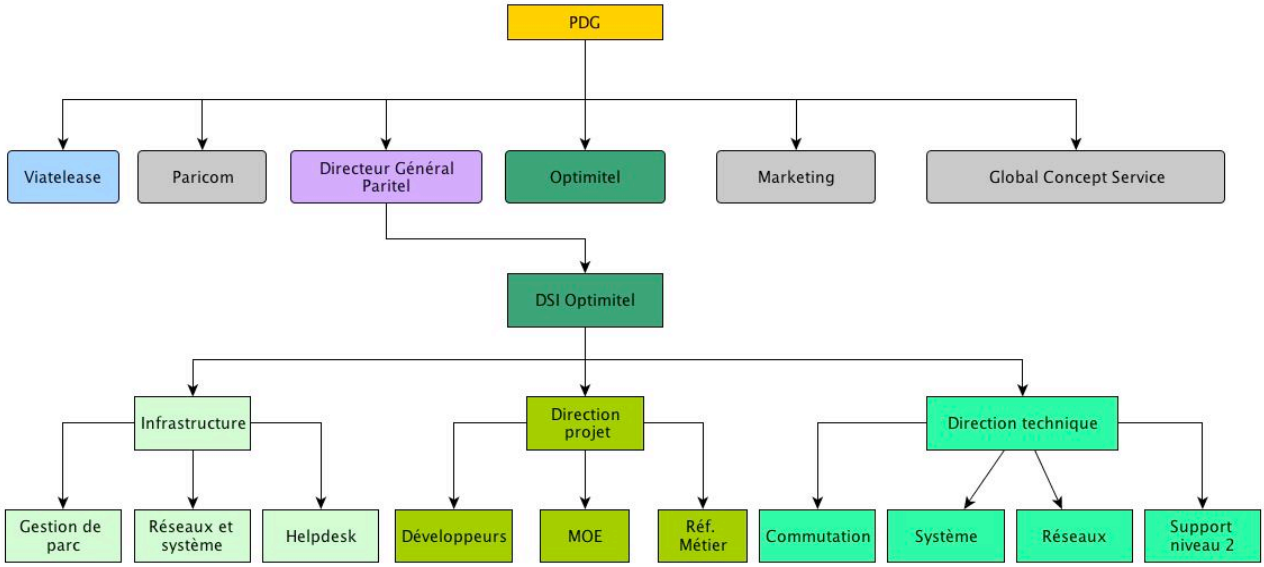


Figure 3 : Organisation de la DSI chargé de la maintenance de Viatelease

La gestion de projets est le département le plus volumineux de la DSI. On peut noter qu'il comprend pratiquement autant de développeurs (Dev : 10) que de maitrise d'ouvrage (MOA : 6) et qu'il n'existe qu'une personne pour assurer les fonctions de maîtrise d'œuvre (MOE : 1).

Cette répartition est inhabituelle et entraine à la fois un déficit de documentation technique (le MOE étant dépassé par la charge de suivi) et un rapprochement entre la MOA et les développeurs. Cette proximité amène souvent les MOA à faire réaliser directement par les développeurs les fonctionnalités demandées par les utilisateurs, faisant du même coup l'économie d'un cahier des charges.

Une autre conséquence immédiate de la faiblesse de la MOE est l'absence de procédure normalisée de mise en production des projets. On constate fréquemment l'apparition de régression à la suite d'une mise en production.

Enfin, on notera que l'équipe de direction projet est assez récente puisque la majorité des développeurs (8/10) et un tiers des MOA (2 seniors) sont des prestataires embauchés depuis moins d'un an.

1.2.2. Projets de la DSI

A l'origine, Global Concept est un opérateur Télécom switchless, c'est à dire qu'il ne possède ni réseaux physiques ni commutateurs. Il est dépendant des opérateurs télécom dont il loue les réseaux qu'il revend sous forme de forfaits à des clients finaux.

En 2009, Global Concept a racheté l'entreprise Ipnotic Optimitel qui est propriétaire de son infrastructure. Cette fusion permet à Global Concept de gagner son indépendance vis-à-vis des opérateurs et d'allier son savoir-faire commercial avec le savoir-faire technique de cette nouvelle filiale.

Cette stratégie a un impact direct sur la DSI, qui doit absorber l'ensemble des systèmes d'information de Ipnotic Optimitel.

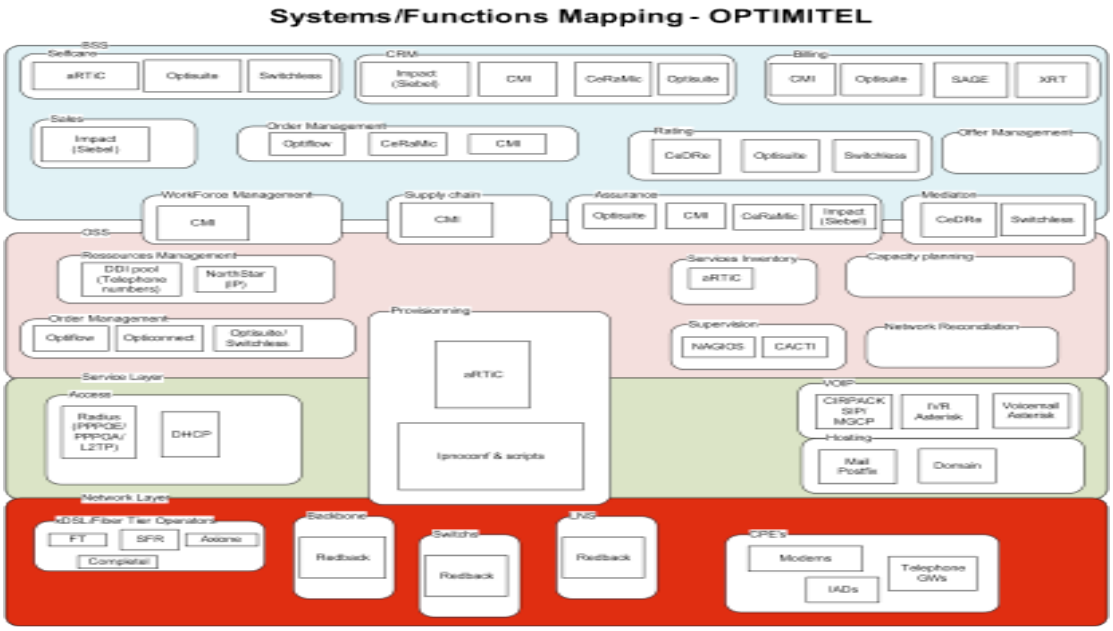


Figure 4 : Cartographie des applications de la société Ipnotic Optimitel

Le schéma ci-dessus illustre le nombre d'applications existantes dans la société Optimitel, que la DSI doit prendre en compte dans le cadre de cette fusion.

Dans le monde des télécom, on distingue deux types de solutions. Les solutions Operating Support System (OSS) pour dialoguer avec les autres opérateurs de réseaux et maintenir le réseau lui-même d'une part et d'autre part les solutions Business Support System (BSS) nécessaires pour

gérer le service rendu aux clients (commande, facturation, etc.). Optimitel étant une acquisition fondée sur le modèle stratégique d'intégration verticale[1], Global Concept, qui ne possède pas de solutions OSS propres aux opérateurs de réseau, ne peut absorber le système d'information de la société Ipnotic sans enrichir l'existant. .

Un audit récent a identifié plus d'une trentaine de projets pour réaliser la fusion du SI d'Optimitel. Ce chiffre doit être mis en rapport avec la nécessité absolue, dans le marché des télécom, de sortir rapidement des produits novateurs. Dans un entretien pour le *Journal du Net*, Georges Penalver, directeur exécutif en charge du marketing stratégique d'Orange France Télécom, rappelait qu'en 2007, Orange était sur « un rythme d'un lancement majeur par semaine et par pays ».

Pour Global Concept, l'acquisition d'Optimitel est un enjeu stratégique pour s'imposer sur le marché des opérateurs. Elle implique que la DSI conduise de nombreux projets informatiques d'intégration ou d'adaptation dans un délai le plus court possible.

1.2.3. Les besoins

La première nécessité pour Global Concept est de s'assurer de la qualité de l'analyse des besoins et des spécifications des projets à réaliser. Parce que cette phase amont d'expression des besoins conditionne le succès du projet[2] et que le coût d'une erreur est 5 à 10 fois plus important que durant la phase de réalisation, il est important d'augmenter le nombre de MOE chargés de la définition des spécifications techniques. L'organisation actuelle (cf. 1.2.1. Présentation de l'organisation de la DSI) dénote la volonté de la DSI de renforcer ses équipes en prévision de l'effort à fournir pour les prochains projets. Néanmoins, les personnes qui sont engagées sont pour la plupart des développeurs et un MOA. Le MOE reste unique et ne peut pas produire rapidement des spécifications techniques de qualité pour tous les projets.

La direction informatique doit structurer la gestion de projet et le processus de mise en production. La gestion de projet informatique implique la mise en production, c'est à dire la mise à disposition des utilisateurs du logiciel développé et testé, conformément aux spécifications fonctionnelles et techniques établies. Elle n'est pas la seule activité dans ce cas. Bien que plus courte, la maintenance nécessite également la gestion de projet et la mise en production de corrections ou d'évolutions. Dans la mesure où cette étape présente des risques pour les activités de l'entreprise, les bonnes pratiques suggèrent de gérer les mises en production comme un processus à part entière. On rappelle qu'un processus implique un responsable ainsi que des

1 Les sociétés intégrées verticalement sont unies par l'intermédiaire d'une hiérarchie et ont un propriétaire commun. En général, chaque membre de cette hiérarchie élabore un produit ou un service différent, ces produits et services se combinant pour satisfaire un besoin final commun.

2 13,1% des projets échouent car les exigences sont incomplètes. http://www.standishgroup.com

acteurs pour exécuter les différentes tâches ou actions. Actuellement, aucun responsable ou acteur pour le processus de mise en production n'est identifié dans l'organisation de la DSI. Le risque d'apparition d'incidents consécutifs à une mise en production mal maîtrisée est donc plus important.

Le processus de maintenance applicative doit être organisé et renforcé. Actuellement, il n'existe pas de responsable de la maintenance qui contrôle et coordonne l'ensemble des niveaux. Le référentiel de bonnes pratiques ITIL (Information Technology Infrastructure Library) recommande de hiérarchiser l'organisation de maintenance en niveaux.

- **Le niveau 1** est chargé de la prise d'appel, de sa qualification et de sa résolution si ce dernier est connu et nécessite peu de compétence technique.
- **Le niveau 2** est chargé de la résolution des incidents que le niveau 1 n'arrive pas à résoudre. On parle d'escalade pour désigner l'assignation d'un incident du niveau n au niveau n+1.
- **Le niveau 3** est composé d'experts qui interviennent soit pour résoudre des incidents techniques ardus, soit pour mettre en œuvre la cause réelle d'un ensemble d'incidents ayant une même cause. On parle alors de problème.
- **Le niveau 4**, s'il existe, désigne les prestataires extérieurs à l'organisation dont l'intervention est encadrée par un engagement contractuel.

L'organisation actuelle de la maintenance respecte le principe de niveaux. Le niveau 1 est constitué par deux personnes de l'infrastructure. Les niveaux 2 et 3 sont indistinctement constitués d'un chef de projet et de développeur(s) de l'équipe de direction projet. Ce manque de clarté dans les niveaux autres que 1 doit être questionné. Le principe d'escalade n'est pas observé, puisque les utilisateurs peuvent directement s'adresser aux personnes du niveau 1 puis, selon la réponse, à celles des niveaux 2 et 3.

Enfin, l'organisation de l'activité de maintenance est contrainte par la stratégie de la direction informatique qui souhaite :

– Privilégier le développement d'applications métiers spécifiques plutôt que des solutions d'éditeurs. Par conséquent, les personnes engagées doivent être formées sur chacun des logiciels avant de pouvoir être productives.

– Faire monter en compétence les ressources internes, au contact de prestataires seniors par exemple, plutôt que d'engager des experts techniques. Ce transfert de compétences doit être maîtrisé notamment en privilégiant la formation d'une équipe mixte prestataire senior - interne junior.

– Maintenir les personnes en charge des niveaux 2 et 3 au contact des projets de développement afin d'entretenir leur connaissance des futurs logiciels à maintenir et de faire remonter au chef de projet les demandes et les tendances des utilisateurs.

1.3 LES SOLUTIONS CIBLES

1.3.1. Processus de maintenance existant

Le schéma ci-dessous décrit le processus de maintenance existant. Il ne s'agit pas d'un processus officiel, mais de la modélisation résultant d'entretiens avec chacun, dans le département de la direction technique, de la direction des projets et de l'Infrastructure. Les niveaux de maintenance sont indicatifs.

Dans le processus actuel, l'utilisateur de Viatelease rencontrant une anomalie avec une application interne peut s'adresser par mail ou par téléphone soit au centre d'appel ou support de niveau 1, soit directement aux experts techniques du support niveau 3. Il n'existe pas de support distinct de niveau 2.

Quand le support de niveau 3 a résolu l'incident, il peut directement le clôturer en informant (par mail) ou non le demandeur. Si ce dernier ne le rappelle pas, cela confirme que l'incident à été résolu définitivement.

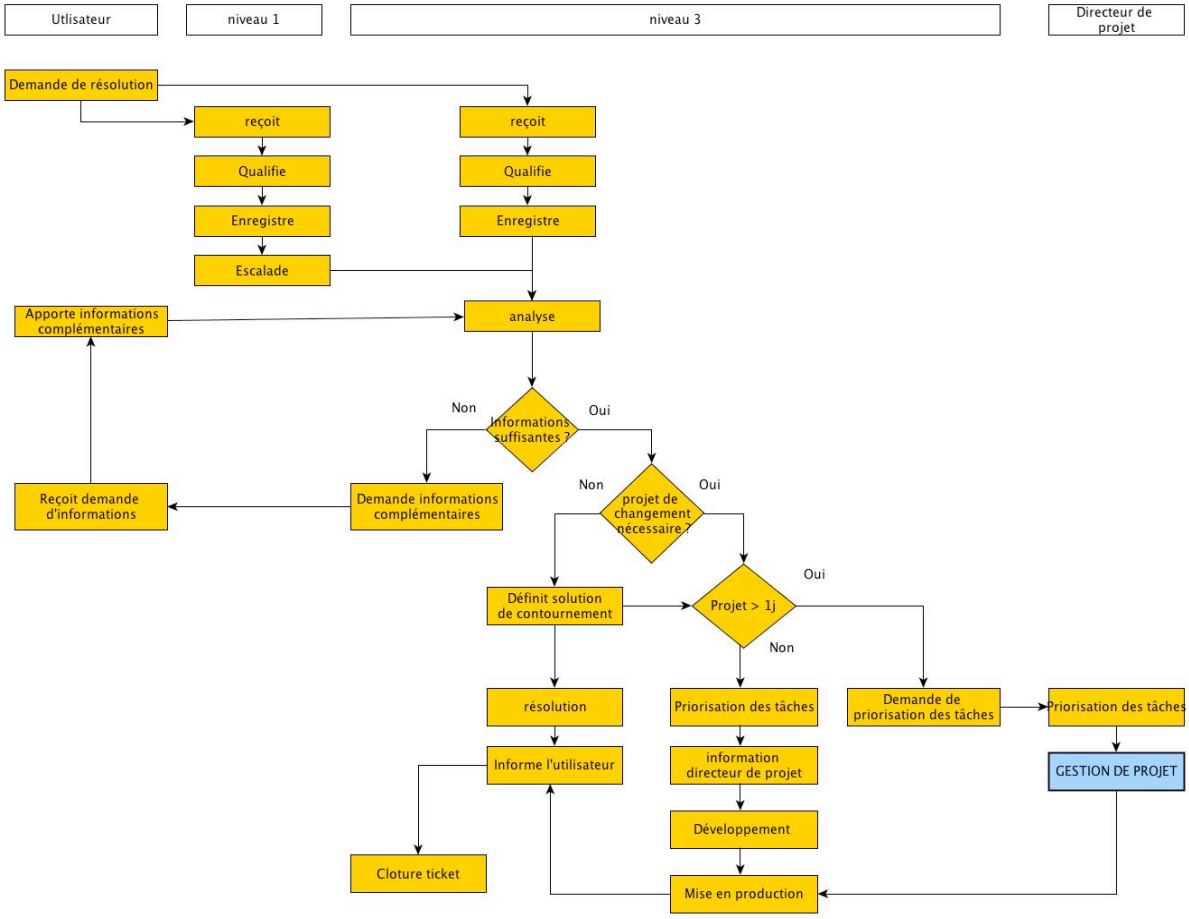


Figure 5 : Processus existant de gestion des incidents

Cette méthodologie a de nombreux inconvénients :

- Surcoût du support, qui peut recevoir plusieurs fois la même demande.
- Pas de trace de l'historique d'une demande.
- Pas de détection des incidents récurrents.
- Opacité du processus de résolution pour les utilisateurs.
- Non capitalisation des connaissances du support dans une base de connaissances.

L'impact business n'est pas un critère de priorisation des tâches de résolution des incidents du support de niveau 3. Le directeur de projet, qui a une connaissance des priorités des métiers, intervient pour ajouter et prioriser de nouveaux projets pour la résolution d'incidents complexes.

La mise en production n'est pas un processus, mais une simple tâche. Le plus souvent, elle est réalisée par la même personne qui a effectué les développements. De fait, il n'y a pas de mise en production groupée mais une multiplication de mises en production unitaires. Ceci a pour conséquence de diminuer la stabilité de l'environnement de production et d'induire des incidents ou des régressions dans l'existant.

1.3.2. Cartographie applicative existante

Le schéma ci-dessous représente la cartographie applicative actuelle pour la maintenance applicative de Viatelease.

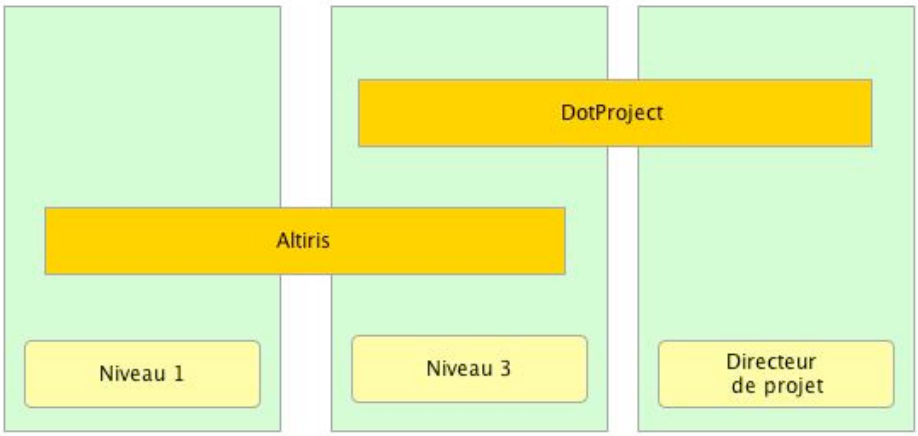


Figure 6 : Cartographie applicative existante

Elle comporte 2 applications : Altiris Helpdesk 6 et Dotproject. La participation de chacune de ces applications au process de maintenance existant (cf. 1.2.1) est figurée par leur représentation

dans des carrés symbolisant les niveaux de support. Elles ne s'interfacent aucunement entre elles. Nous allons à présent décrire leur utilité pour les différents niveaux de support.

1.3.2.1 Altiris

Altiris est un logiciel d'éditeur payant de gestion de tickets.

Le support de niveau 1 l'utilise pour enregistrer les demandes de résolution d'incidents et de services des utilisateurs, y compris celles concernant les applications internes de Viatelease : SIV et l'Extranet.

Lors de l'enregistrement, le niveau 1 qualifie les éléments suivants :

- La catégorie d'incident : exemple Application interne, SIV
- Le type de demande : résolution d'incident, demande de service, etc.
- L'impact de l'incident : élevé, moyen, faible
- L'état de l'incident : en attente, planifiée, en traitement, etc.
- Le titre de l'incident
- Les commentaires de l'utilisateur
- La date de création.

La demande est ensuite transmise au niveau 3, c'est-à-dire aux référents métiers du Département informatique. L'escalade peut être adressée à l'ensemble des personnes du niveau 2 ou à un référent métier particulier, suivant la connaissance de l'organisation de la personne du support niveau 1.

Les personnes du support niveau 3 utilisent Altiris pour communiquer avec le demandeur ou/et le bénéficiaire de la demande afin de lui demander des informations complémentaires ou de l'accompagner dans la résolution de l'incident.

1.3.2.2 Dotproject

Dotproject est une application open source gratuite de gestion de projets.

Un projet correspond à une série de tâches, effectuées par une ou plusieurs personnes et possédant une date de début et une date de fin. Le statut d'une tâche correspond à un pourcentage d'avancement. DotProject répond aux besoins de surveillance du planning et des coûts de projet importants (effort supérieur à 10 jours homme).

Il est moins adapté au suivi de gestion de projets de maintenance dans lesquels la durée des tâches est définie de manière dynamique et nécessite une grande réactivité de la part des différents acteurs du processus.

La structure de projet dans DotProject est statique. Les utilisateurs indiquent le niveau de réalisation des tâches qui leur incombent.

Le directeur de projet utilise DotProject pour réaliser les tâches suivantes :

- Enregistrement du planning prévisionnel de projet, de maintenance évolutive ou corrective, impliquant au moins un développeur
- Assignation des tâches d'un projet à un ou plusieurs développeurs
- Consultation des comptes rendus de tâche des développeurs (durée réelle, résumé des actions réalisées)
- Enregistrement du temps consacré à l'analyse fonctionnelle ou à la recette de projet de développement.

Les projets de maintenance évolutive et corrective saisis par le support niveau 3 dans DotProject et les demandes de résolution d'incident ou de services renseignés par le même support de niveau 2 ne sont pas liés, dans Altiris, par un identifiant commun.

Le support de niveau 3 utilise DotProject pour consulter les tâches qui lui sont assignées et tracer les actions qui sont réalisées ainsi que leur durée. Ces tâches peuvent être relatives à :

- Un projet de nouvelle application
- Un projet de maintenance évolutive
- Un projet de maintenance corrective.

Les informations saisies dans DotProject ne sont pas capitalisables comme base de connaissances. En effet, la visibilité des informations saisies est limitée aux personnes participant au projet. DotProject ne propose pas de moteur de recherche

1.3.2. Processus de maintenance applicative cible

1.3.2.1 Processus de gestion des demandes de résolution

Le diagramme ci-dessous illustre le processus cible de gestion des demandes de résolution. Avant d'exposer les bénéfices attendus de ce nouveau processus, il est important de rappeler qu'il ne répond pas à des objectifs de conformité à la norme ISO 20 000 ou à un référentiel de bonnes pratiques comme ITIL. Si ces outils permettent d'organiser une activité de maintenance, l'attente

de la direction informatique était de mettre rapidement en œuvre des solutions pragmatiques en limitant l'impact du changement sur les équipes impliquées.

Dans le référentiel ITIL, les processus de gestion d'incident, de problème, de changement des configurations, de gestion de connaissances et de mise en production sont distincts. Un responsable et des indicateurs de performance sont attribués à chacun d'entre eux. Un tel découpage serait trop lourd dans l'entreprise de notre étude de cas et nécessiterait un changement contraire aux contraintes énoncées en début de projet.

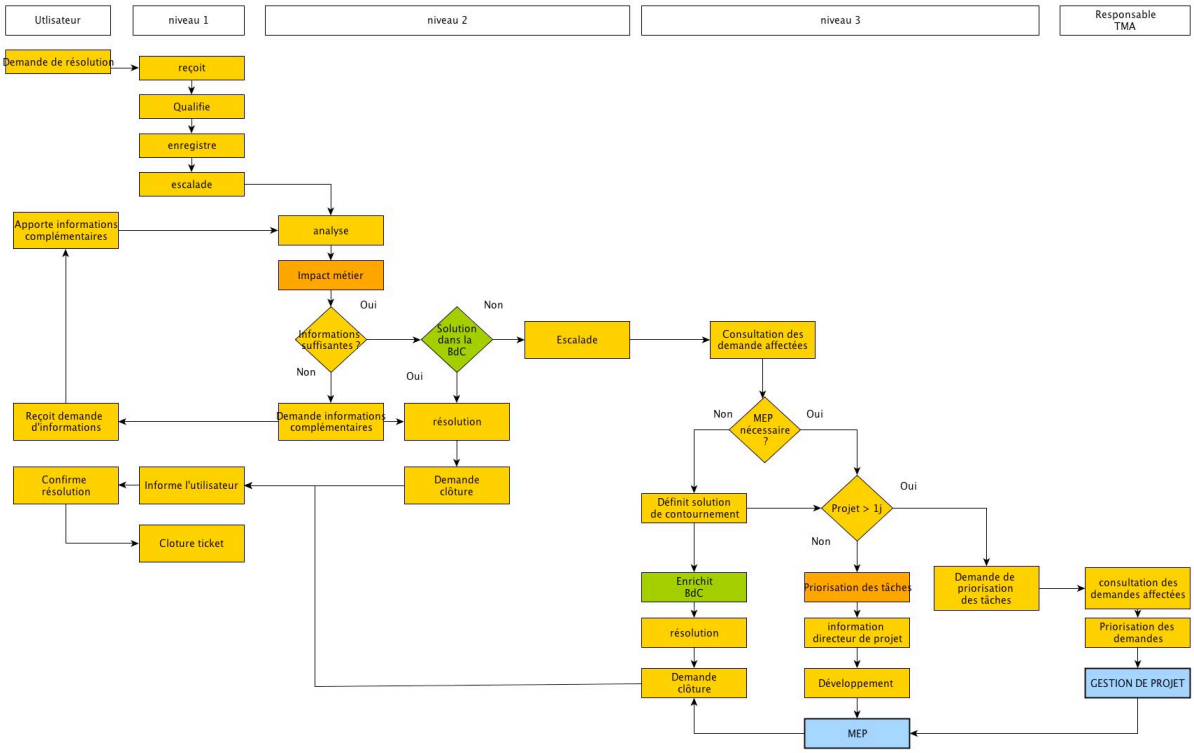


Figure 7 : Processus cible de gestion des incidents

Les utilisateurs ont un seul point de contact[3], le support de niveau 1, pour demander la résolution d'incident. Cette unique point permet de :

- Simplifier et d'harmoniser la transmission des demandes
- Surveiller les délais de résolution de l'ensemble des incidents
- Identifier les problèmes et les incidents récurrents
- Mesurer l'évolution des performances du processus.

3 Le Point de contact unique (Single Point Of Contact en anglais) est une des bonnes pratiques du référentiel unique pour la gestion des demandes

Seul, le support de niveau 1 est autorisé à clôturer une demande après information/consultation des utilisateurs. L'utilisateur est replacé au centre du processus puisqu'il a le pouvoir de créer et de clore les tickets.

Un support de niveau 2 distinct du niveau 1 et 3 est défini. Il a pour fonction de résoudre les incidents dont la solution définitive ou de contournement est déjà connue et documentée dans la base de connaissances.

On désignera un responsable de la maintenance applicative qui répondra de l'ensemble du processus. En raison de la nouveauté du processus et notamment de la mise en œuvre de nouvelles applications, la nomination d'une personne distincte du directeur de projet est préconisée.

On désignera également un responsable des Mises En Production (MEP) pour gérer la planification, la préparation, le regroupement et la mise en production des développements réalisés dans le cadre de la gestion de projet et de la gestion des incidents. Il collaborera étroitement avec les chefs de projets et les développeurs (MOE) qui lui fourniront les procédures de mise en production.

On notera le recours à une base de connaissances (BdC). Cette dernière est enrichie par le niveau 3 (tâche en vert) après la définition d'une solution de contournement. Elle est consultée par le niveau 2 (tâche en vert) qui, le cas échéant, escaladera le ticket au niveau supérieur. Enfin elle est également enrichie par le chef de projet (MOA) et le responsable technique (MOE) durant le processus de projet.

Le traitement des demandes prend en compte l'impact métier (rectangles en rouge) à tous les niveaux du support. La définition de l'impact métier est possible grâce à la base de connaissances qui définira pour chaque application la criticité de l'indisponibilité.

On note l'existence d'un cycle de tâches du support de niveau 2 pour l'analyse des demandes. Cette analyse doit permettre de qualifier au mieux les éléments suivants de la demande :

- Description des impacts techniques de l'incident (applications et matériel)
- Description de l'impact métier de l'incident.

Le support de niveau 2 s'adresse directement à l'utilisateur et consulte la base de connaissance avant de choisir d'escalader la demande au support niveau 3.

1.3.2.1 Processus de gestion des projets de maintenance applicative

Le diagramme ci-dessous illustre le processus de gestion de projet de maintenance. Il est mis en œuvre pour des projets entre 2 et 10 jours hommes. Au-delà de cette durée, le projet est considéré comme un projet « classique » et doit suivre un processus qui ne fait pas partie du sujet de cette étude.

On rappelle que les demandes de maintenance peuvent être de deux types :

- **Correctif** : le projet a pour objectif de corriger une anomalie sur une fonction décrite dans les spécifications fonctionnelles.
- **Evolutif** : il s'agit d'ajouter une fonctionnalité absente ou de modifier une fonctionnalité existante dans les spécifications fonctionnelles.

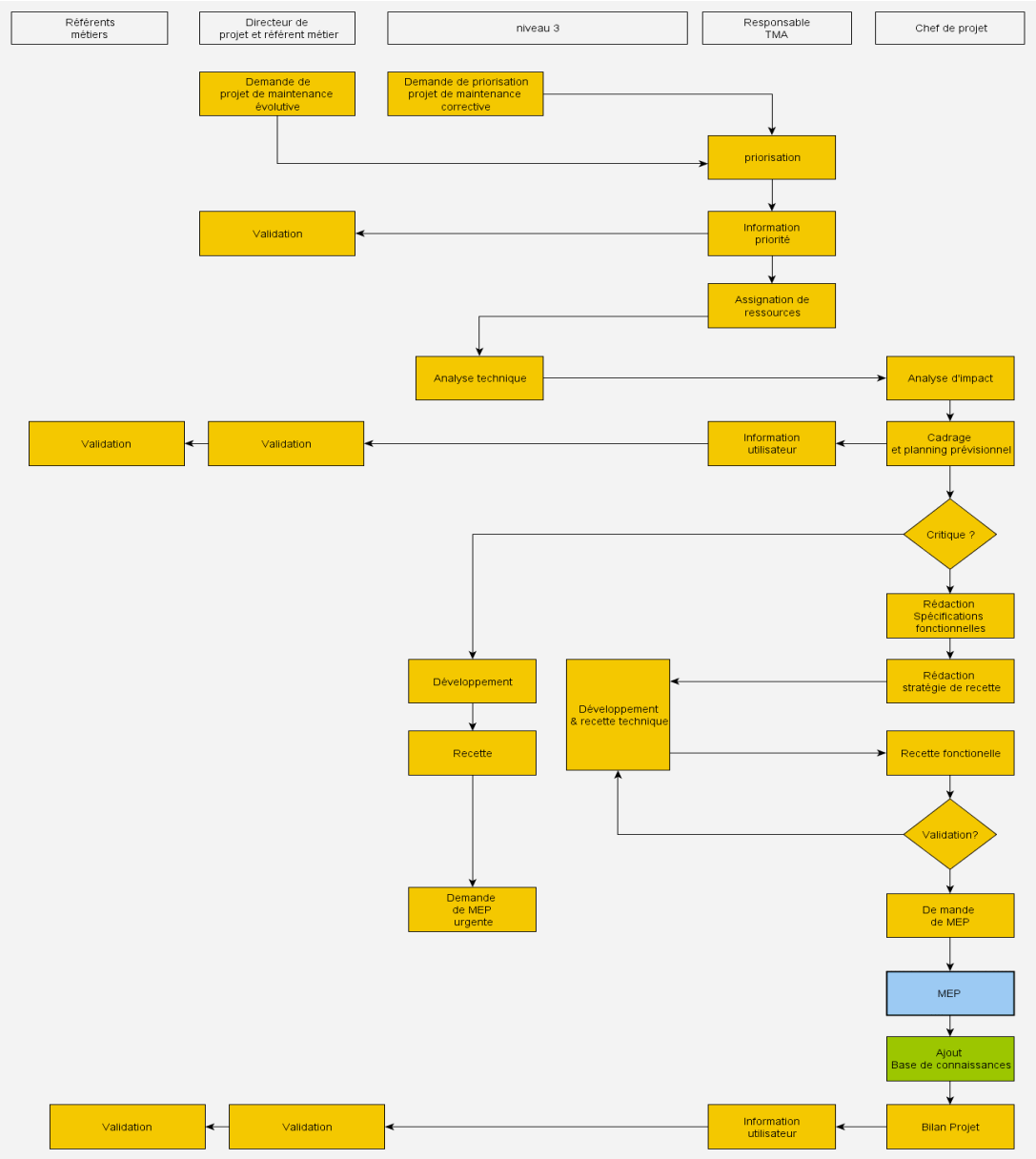


Figure 8 : Processus cible de gestion des projets de maintenance applicative

Ce processus est initié soit par le processus de gestion des demandes de résolution vu précédemment, soit par un référent métier. Dans les deux cas, le responsable de TMA a pour tâche de (re)prioriser l'ensemble des projets en cours et de (ré)assigner des tâches au chef de projet et aux développeurs de l'équipe de gestion de projet.

La priorisation des projets dépend de la qualité de l'expression des besoins ou de la qualification de la demande de résolution. Dans ce deuxième cas, la définition de l'impact métier, par le support de niveau 2 à partir de la base de connaissance, est un atout significatif. Le responsable TMA est capable de l'évaluer, mais la base de connaissance rend légitime cette évaluation en se fondant sur des accords validés. La description de la criticité d'une application dans la base de connaissances facilite l'ordonnancement des projets de maintenance.

A l'instar du processus parent de gestion des demandes de résolution, on note la présence d'un cycle de tâches durant la réalisation de la solution entre le chef de projet chargé des spécifications fonctionnelles et les développeurs du support niveau 3.

Ce cycle traduit la grande proximité et la grande réactivité entre le chef de projet et le développeur en charge de la réalisation de la solution de maintenance. En effet, si ce processus s'inspire des méthodes de développement par lot utilisées (pour la gestion de projets complexes, il est bien moins contraignant). Etant donné la culture de rapidité de l'entreprise et la durée moyenne des projets de 3 jours/homme, certaines phases de projets ont été simplifiées dans ce processus. La phase de conception ne fera pas l'objet d'un cycle de validation impliquant sponsor, référent métier et directeur informatique. De plus, il n'est préconisé qu'une seule recette fonctionnelle avant la demande de mise en production au lieu d'une recette par itération du cycle de développement.

Autre point commun avec le processus de gestion des demandes de résolution, il existe une tâche d'enrichissement de la base de connaissances. Elle vise à pallier l'une des principales difficultés pour toute direction des systèmes d'information : le maintien à jour de la description des applications en production pour les équipes en charge de l'évolution ou de la correction du système d'information.

Enfin, on notera que le processus s'achève par la validation du bilan du projet par les référents métiers et le directeur informatique. Ce bilan comprend les spécifications fonctionnelles et le P.V. de recette. On applique également ici le principe de transparence évoqué précédemment dans la gestion des demandes.

1.3.3. Cartographie applicative cible

Le schéma ci-dessous représente la cartographie applicative cible pour la maintenance applicative de Viatelease.

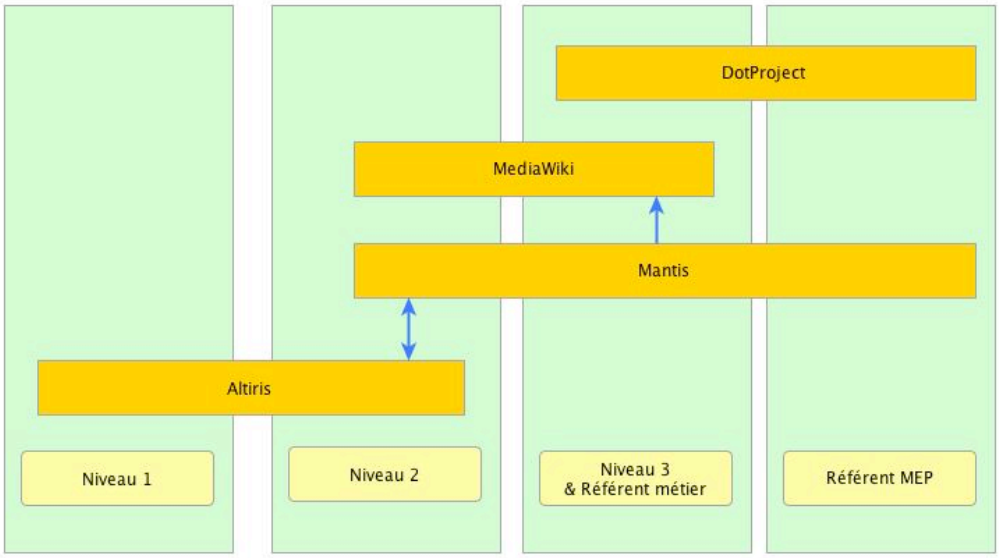


Figure 9 : Cartographie applicative cible

Elle comporte 4 applications : Altiris Helpdesk, Mantis, MédiaWiki et DotProject. La participation de chacune de ces applications au process de maintenance existant (cf. 1.2.1), est figurée par leur représentation dans des carrés symbolisant les niveaux de support. Certaines de ces applications communiquent entre elles deux par deux.

- Altiris Helpdesk et Mantis
- Mantis et MediaWiki.

Après une présentation de Mantis dans la mise en œuvre du processus cible de maintenance applicative, nous décrirons les nouvelles interfaces qui seront créées au cours de notre étude de cas.

1.3.3.1 Mantis

Mantis permet de mettre en œuvre des workflows pour le traitement de demandes désignées sous le nom de bogue dans l'application.

Dans les processus cibles de gestion des demandes de résolution et de gestion de projet de maintenance, nous avons pu noter l'existence de cycles de tâches. Ces cycles induisent la mise en œuvre de gestion de flux ou workflow[4].

Le diagramme UML [5] d'état transition ci-dessous représente les différents statuts d'une demande

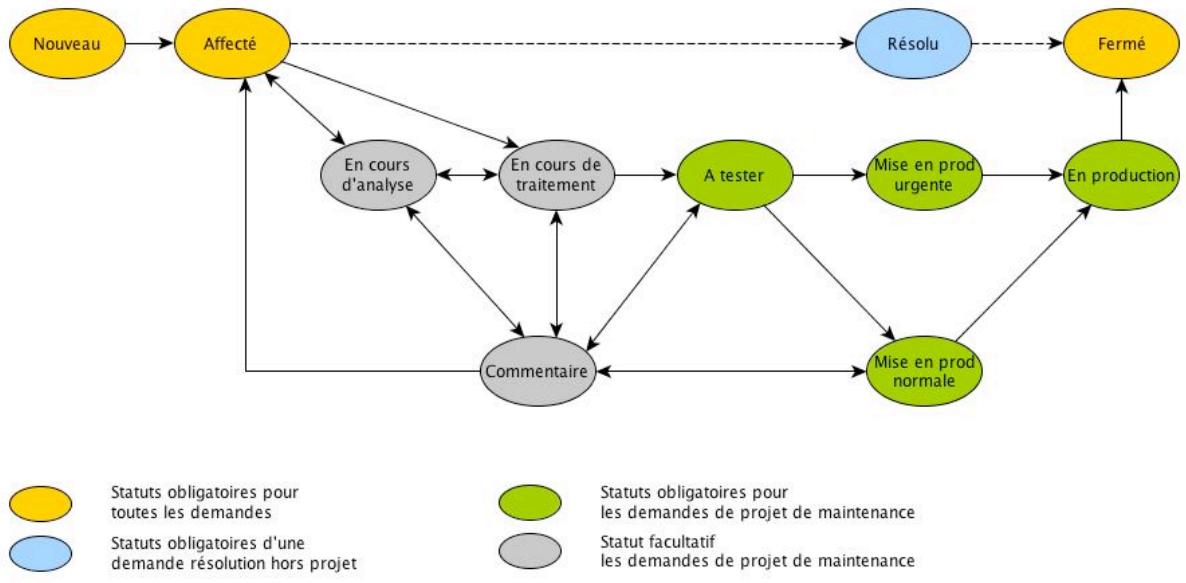


Figure 10 : Graphe cible des statuts d'une demande de maintenance applicative

On remarque dans ce diagramme la présence de deux chaînes, correspondant aux deux types de workflow possibles pour une demande de maintenance :

Chaine haute en pointillés : workflow de traitement d'une demande hors projet car sa durée est inférieure à 1 jour par homme. Elle est critique.

Chaine basse en trait plein : workflow de traitement d'une demande dont la durée se situe entre 2 et 10 jours homme dans le cadre d'un projet de maintenance.

1.3.3.2 Altiris - Mantis

Il existe une relation entre les demandes de résolution d'incident pour les applications internes enregistrées dans Altiris et les demandes traitées par le niveau 3 et par les référents métiers.

---

4 Nous retiendrons l'anglicisme de Worklow car il est à la fois représentatif de la culture d'entreprise du département informatique de Global Concept et de la littérature sur la gestion de projet.

5 Unified Modeling Langage est un langage de modélisation couramment utilisé dans les projets logiciels (cf. http://fr.wikipedia.org/wiki/Unified_Modeling_Language)

Bien que le workflow de traitement des tickets Altiris et celui des demandes Mantis soient différents, il existe une relation basée sur les statuts de début et de fin de gestion de demande ou bogue, comme l'illustre la figure ci-dessous.

L'application Mantis vient compléter Altiris en prenant en charge la gestion de projet de maintenance dont seuls les statuts terminaux (En production ou Résolu) intéressent le support niveau 1.

Les personnes du support niveau 3, le référent métier et le responsable de MEP (Mise En Production) utiliseront uniquement Mantis pour consulter et traiter les demandes.

Le diagramme ci-dessous illustre ces relations.

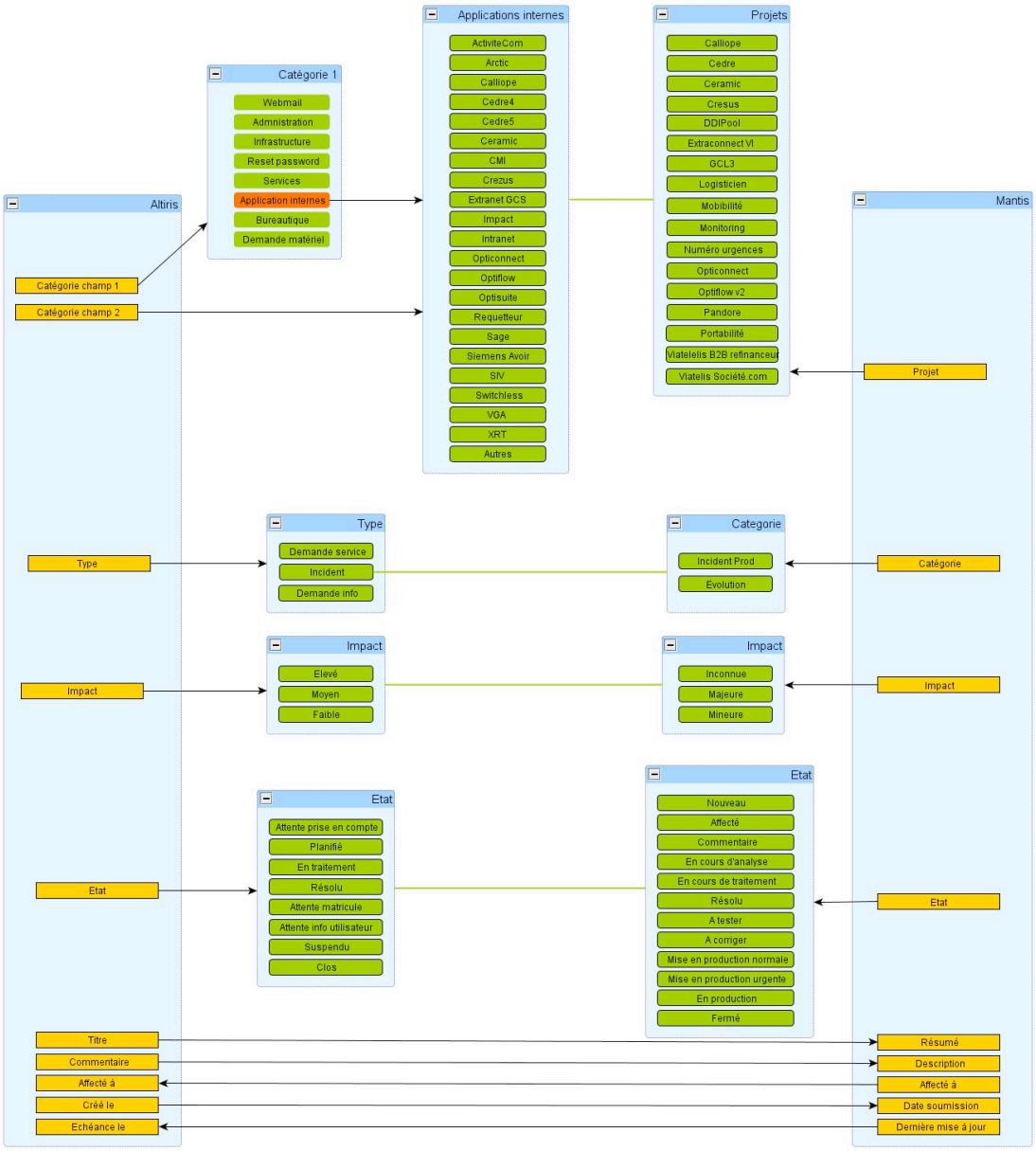


Figure 11 : Diagramme de conversion des données entre Altiris et Mantis

1.3.3.3 Mantis - MediaWiki

MediaWiki est une solution open source gratuite de gestion de connaissances de type « Wiki », à l'instar du célèbre Wikipedia

Les objectifs de la base de connaissances sont les suivants :

- Faciliter la qualification (impact, priorité) des demandes par la documentation fonctionnelle et technique des applications maintenues.
- Rechercher et consulter les solutions de contournement ou de résolution connues par application.
- Maintenir à jour la description fonctionnelle et technique des applications.
- Enrichir la base des solutions de contournement et de résolution.
- Informer les personnes du département informatique de l'actualité des opérations de maintenance.

MediaWiki facilite la recherche d'information grâce à

- Un puissant moteur de recherche par mot clés et par catégorie dans les pages.
- Une structuration des informations (cf. figure ci-dessous).

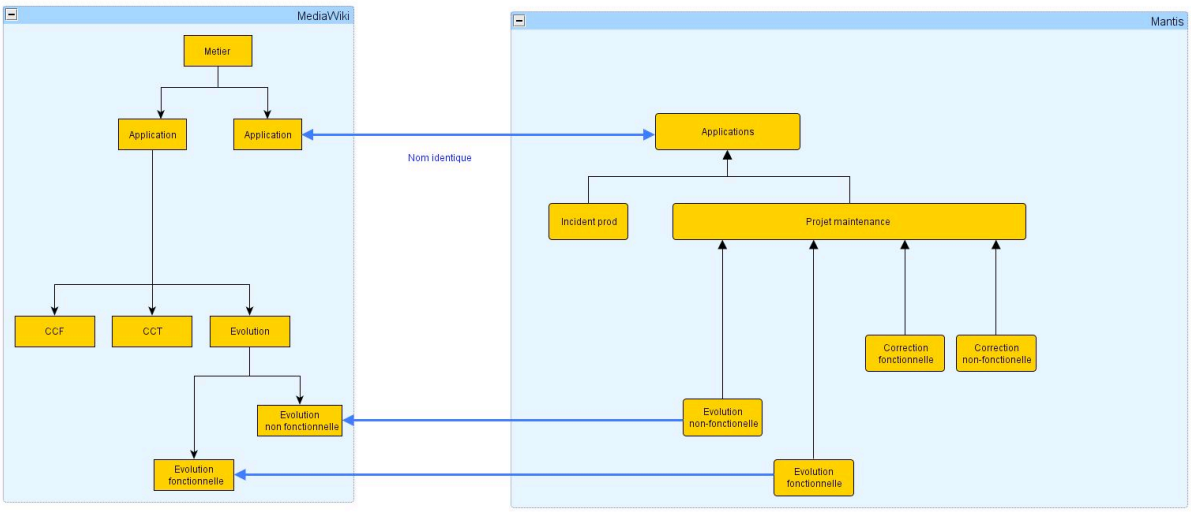


Figure 12 : Diagramme de conversion des données entre MediaWiki et Mantis

L'enrichissement de la base de connaissance est facilité par les mécanismes automatiques d'importation de contenu depuis Mantis dans MediaWiki. Nous aborderons ces mécanismes lors des chapitres suivants.

MediaWiki est une solution web. Une page de news peut être tenue facilement à jour par le responsable de la maintenance (mise en production d'une application, ordre du jour et compte

rendu de réunion, etc.) et permettre aux personnes du département informatique de s'informer rapidement des dernières actualités de la maintenance.

Viatelease souhaite augmenter le nombre de contrats vendus à des refinanceurs en développant son offre de demande de financement auprès d'apporteurs d'affaires hors groupe. Cette stratégie implique à la fois une consolidation et une évolution des fonctionnalités des applications de son système d'information.

Le processus de maintenance cible présenté dans ce premier chapitre répond à cette double contrainte de maintenance en mode continu et en mode projet. Elle nécessite, d'une part une nouvelle organisation du département informatique avec la création de deux nouveaux postes : un référent applicatif et un responsable de mise en production, et d'autre part, la mise en œuvre de nouvelles applications open source : Mantis et MediaWiki. Nous allons à présent étudier en quoi l'accès au code de ces deux applications est un atout pour répondre aux besoins de structuration des tâches (Mantis) et d'organisation des connaissances (MediaWiki) des projets de maintenance applicative.

Chapitre 2 - étude sur l'ajout de briques applicatives open source

2.1. METHODE ET OUTILS D'INTEGRATION DES APPLICATIONS OPEN SOURCE

2.1.1. Gestion des versions : Subversion

Avant d'étudier comment modifier le code source des applications Mantis et Wiki afin de les adapter aux processus définis précédemment, nous allons rapidement présenter les outils logiciels qui nous ont permis de réaliser le suivi des versions du code source, sa modification dans les fichiers ainsi que la modification des bases de données associées aux traitements. Cela est d'autant plus pertinent que chacun de ces logiciels, à savoir Subversion, Eclipse et Phpmyadmin, est lui-même open source et gratuit. De cette façon, nous préservons la cohérence de notre démarche de qualité et d'économie. En effet, l'achat de solutions d'éditeur pour la modification d'applications open source aurait semblé paradoxal.

Subversion est un logiciel open source de gestion de version de code source développé par la fondation Apache, une organisation à but non lucratif américaine existant depuis 1999. Il vise à faciliter le travail de développement en équipe, notamment en permettant à l'équipe de développement de :

- Centraliser le code source de plusieurs applications sur un serveur unique. On parle de dépôt (repository en anglais).
- Copier la version la plus récente du code depuis un serveur centralisé vers leur environnement de développement local
- Mettre à jour depuis leur poste le code archivé sur le serveur central.
- Garder un historique des modifications du code source avec leur nature, leur date et leur auteur.
- Revenir à quelque version antérieure que ce soit.

La gestion des versions dans Subversion se base sur la gestion du code de l'application en tronc, branche et release.

Le tronc représente la version la plus stable des fichiers de code applicatif. En cours de développement, pour répondre aux besoins d'évolution ou de corrections, les fichiers de code sont amenés à être modifiés. Plutôt que de les modifier directement dans le tronc et compromettre la stabilité de l'application, le développeur va créer, dans SVN, une branche qui reprendra une copie du tronc au moment de la création. Il disposera ainsi d'un code similaire pour pouvoir réaliser et tester les développements nécessaires. A terme, la branche sera fusionnée avec le tronc ou abandonnée par les développeurs.

Une Release correspond à l'enregistrement du code du tronc ou d'une branche à un moment donné dans le temps. On parle également de marquage (tag en anglais) du code. Lors de la mise en production, une release dite candidate est copiée sur le serveur. En cas de dysfonctionnement, un retour arrière à la release précédente est facilité. On notera que chaque release a un nom unique comportant la date du jour et si cela est nécessaire le numéro de version. Cette fonction est automatique avec SVN.

2.1.2. Modification des fichiers de code sources : Eclipse

L'adaptation des applications Mantis et MediaWiki nécessite la consultation et l'édition du code source en PHP (cf. les paragraphes ....). Eclipse est un environnement de développement intégré open source écrit en Java par la fondation Eclipse créée en 2001. Son comité de direction est composé de représentants de grandes sociétés comme HP, IBM, Intel ou SAP.

La particularité d'Eclipse est sa modularité. Elle accepte de nombreux modules (plugin), dont le PHP Development Tools Framework (PDT) destiné au développement d'applications web basées sur le langage PHP côté serveur. Les principaux avantages de ce framework sont :

- L'édition en onglets des fichiers PHP, qui facilite le travail sur de nombreux fichiers source
- La coloration syntaxique du PHP, qui améliore la lisibilité du code source
- La création automatique de la documentation des classes, méthodes, fonctions et variables de PHP écrites par les développeurs de l'application (phpdoc)
- L'auto complétion des noms de classes, méthodes, fonctions et variables de la phpdoc, qui réduit les fautes de syntaxe
- L'affichage des paramètres et valeurs de retour des méthodes, qui permet à une équipe de développeurs de pouvoir rapidement utiliser des méthodes développées par d'autres
- La navigation entre les fichiers de code source par méthodes, fonctions et variables, qui permet, lors de l'analyse d'une anomalie, de trouver rapidement la partie du code source à l'origine de l'erreur.

La recherche par expression à l'intérieur des fichiers du code. Le résultat de cette recherche se représente sous forme d'arborescences de lignes de fichiers et de fichiers de branches éditables dans un nouvel onglet. La partie de code correspondant à l'expression est surlignée pour une lecture rapide.

Autre particularité d'Eclipse, il existe un module d'intégration avec Subversion. Depuis le menu contextuel de chaque fichier ou répertoire de l'arborescence d'une application, comme l'illustre la figure ci-dessous, il est possible de réaliser rapidement les principales actions suivantes :

Check out, c'est-à-dire la copie en local, pour la première fois, des fichiers existants sur un dépôt

Update, c'est-à-dire la mise à jour de la copie de travail locale des fichiers avec la dernière version des fichiers du dépôt

Commit, c'est-à-dire la mise à jour du dépôt depuis la copie de travail des fichiers en local. Un texte court contenant une description des modifications effectuées est associé à cette mise à jour d'un ou plusieurs fichiers. En cas de conflit, un message d'erreur apparaîtra lors du commit.

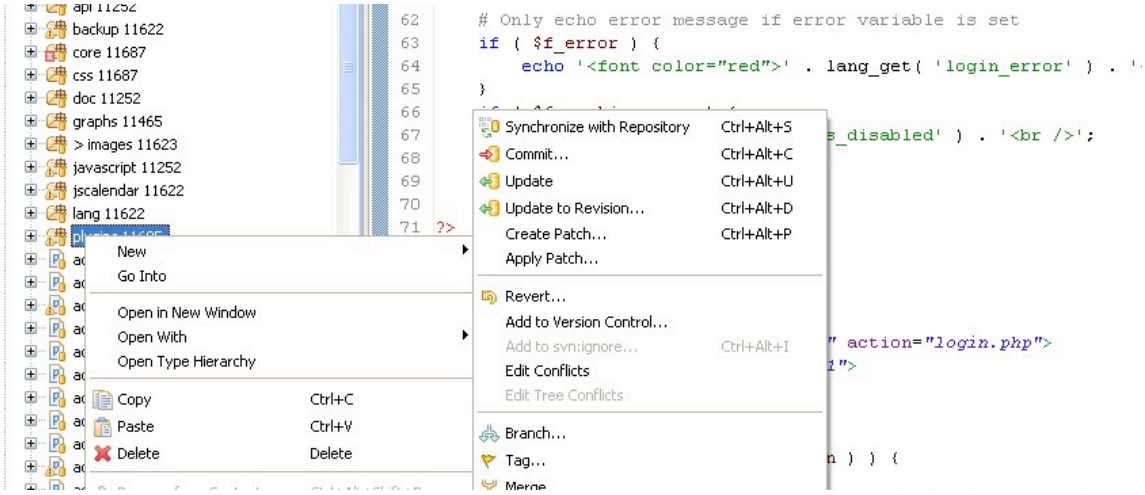


Figure 13 : Fonctionnalité du menu contextuel dans Eclipse

2.1.3. Manipulation des données : PhpMyadmin

Eclipse et SVN facilitent la manipulation du code pour l'affichage et le traitement des données. PhpMyadmin vient compléter ces deux applications en fournissant une interface graphique pour la définition et la manipulation des données du serveur de base de données relationnelles open source gratuit Mysql.

PhpMyadmin est une application web open source gratuite créée en 1998 et écrite en PHP. Elle fournit une interface graphique web, donc multiplateforme, à partir de client léger pour l'administration des serveurs Mysql et la création de requête de traitement des données. Il est toujours maintenu par l'ensemble de la communauté de la plateforme d'hébergement et de distribution de logiciel open source Sourceforge.

L'image ci-dessous illustre l'interface de gestion d'une table. Le menu de gauche liste l'ensemble des tables d'une base de données, pendant que la partie centrale fournit des onglets de gestion d'une table, dans cet exemple, la table archive. On notera que les développeurs ont le choix entre les outils graphiques de définition, de traitement des données et la création de requête en langage SQL.

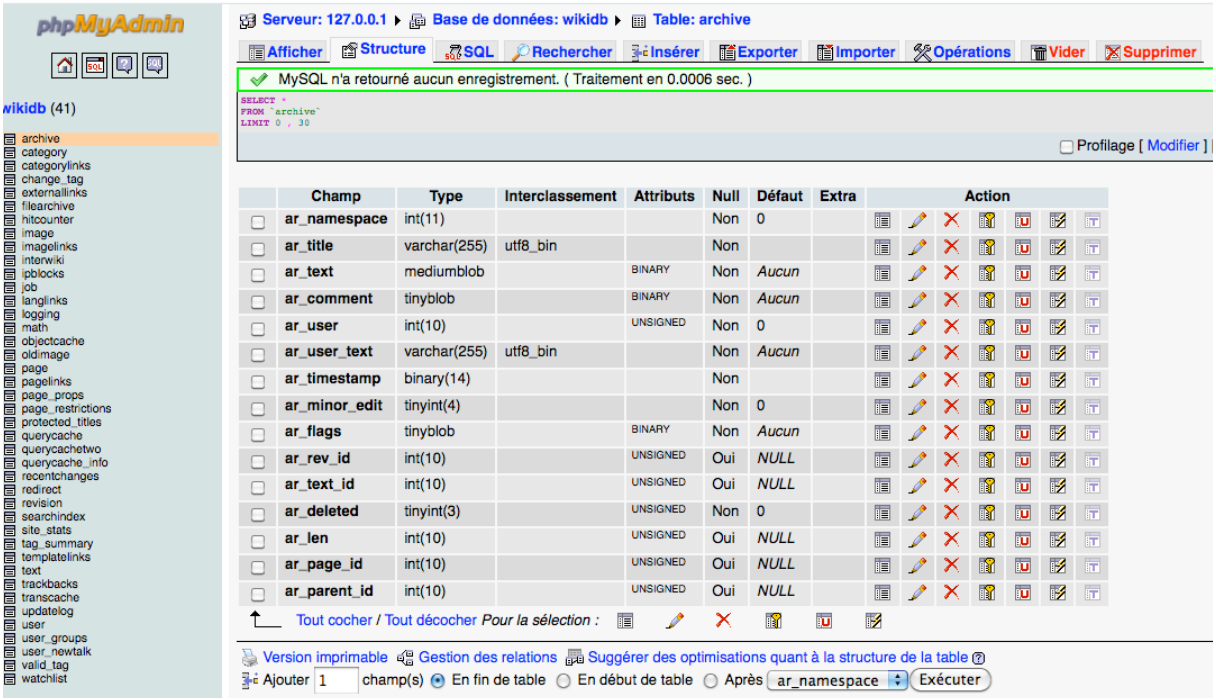


Figure 14 : Fonctionnalité de gestion des données avec phpMyadmin

SVN, Eclipse et PhpMyadmin, sont trois solutions « open source » gratuites offrant une trousse à outils complète pour le développement rapide et professionnel d'applications web en PHP.

Nous allons à présent étudier leur mise en œuvre pour l'ajout de solution d'amélioration du suivi des anomalies et de gestion des connaissances.

2.2. ETUDE SUR L'AJOUT D'UNE SOLUTION D'AMELIORATION DU SUIVI DES ANOMALIES : MANTIS BUG TRACKER

2.2.1. Installation de Mantis

2.2.1.1. Architecture technique

Le schéma ci-dessous décrit l'architecture de Mantis.

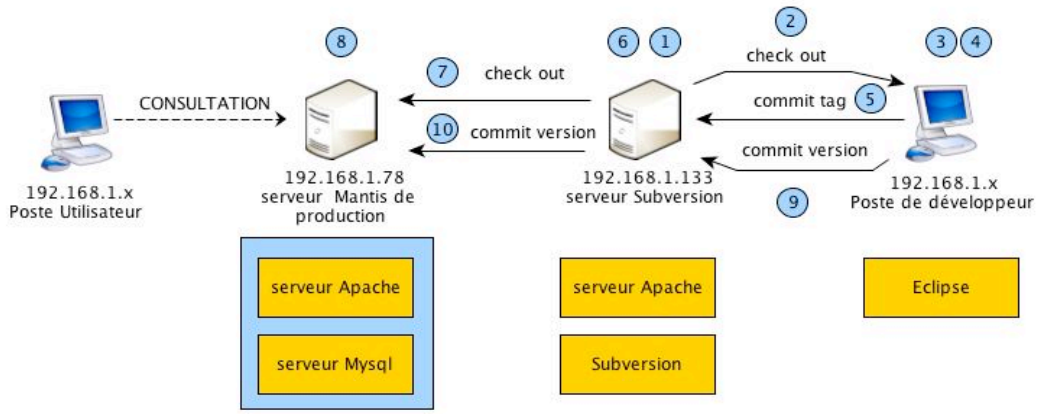


Figure 15 : Architecture technique de Mantis

On notera qu'il n'existe qu'un seul environnement de production pour le développement de l'application. En effet, cette dernière n'est destinée qu'à l'équipe informatique et n'est développée que par une personne.

Les coûts d'un environnement de développement ou/et de test sur les serveurs de Paritel ne sont pas justifiés. Le poste de travail local tient lieu d'environnement de développement et de test avant de valider une révision sur le serveur SVN et de mettre à jour le code source du serveur de production.

Le poste de travail local est doté d'un environnement complet de développement et de test composé des éléments suivants : Un serveur web Apache, un serveur applicatif PHP, un serveur de gestion de base de données Mysql pour l'exécution de l'application côté serveur, Eclipse et ses extensions PHP et SVN pour le développement de l'application proprement dite.

Tous les serveurs sont dotés d'un serveur Apache. Il est utilisé pour

- Présenter le résultat de l'exécution du code PHP dans le serveur de production.
- Manipuler les données avec PhpMyadmin dans le serveur de bases de données.
- Faciliter la gestion des droits d'accès et le transfert de fichiers d'un client à un dépôt et vice et versa dans le serveur Subversion.

Tous les échanges sont réalisés sur le protocole http. Dans cette cartographie, tous les ordinateurs sont sur la même classe d'adresse. Néanmoins, les flux pourraient être routés entre réseaux de classes d'adresse différentes.

2.2.1.2. Description des flux sur le serveur SVN

Les numéros indiqués sur la figure sont ici détaillés.

1. Depuis le client Eclipse, création d'un nouveau dossier et définition de ce dossier comme un nouveau projet PHP. SVN va créer un tronc, des branches et des releases.

2. Copie du tronc de ce nouveau projet en local sur le poste de développement : checkout. Création d'un espace de travail (workspace) sur l'environnement local.

3. Ajout des fichiers de Mantis téléchargés sur l'espace de travail.

4. Installation de Mantis en local (avec création d'une base de données sur le serveur Mysql de production).

5. Mise à jour du dépôt central avec le code du tronc en local par la commande commit.

6. Création d'une release à partir du tronc du dépôt.

7. Création d'un répertoire Mantis dans le dossier projet du serveur de production. On notera que les droits sont modifiés pour que le serveur Apache puisse exécuter les fichiers dans ce répertoire. La commande ci-dessous illustre la mise à jour des droits.

```
00 22 * * * svn up /var/www/projects
01 00 * * * chmod -R ug+rw /var/www/projects && chown -R www-data:www-data /var/www/projects
info-dev-01:~#
```

Extrait de code 1 : mise à jour des droits pour le checkout

8. Copie de cette release sur le serveur de production et test de la release depuis le navigateur du poste de développement

Par la suite, les flux 9 et 10 seront uniquement réitérés

Les évolutions seront réalisées et testées sur le poste de développement.

9. Mise à jour du dépôt central avec le code du tronc en local par la commande SVN update.

10. Mise à jour du serveur Mantis avec le code du tronc sur le serveur SVN par la commande check out.

Comme il a été dit précédemment, vu le nombre réduit et limité d'impact limité de l'application sur les utilisateurs finaux, il ne sera mis en œuvre ni branche de développement, ni serveurs de test.

Les mises à jour ne feront pas l'objet de release, mais seront directement copiées sur le serveur de production.

2.2.2. Paramétrage de Mantis

Nous avons défini, dans le paragraphe 1.3.2., un processus cible mettant en jeu des acteurs chargés d'exécuter des tâches pour gérer les demandes de maintenance applicative. Avant d'étudier comment le code peut être modifié pour rendre Mantis le plus conforme possible à ce processus cible, nous allons expliquer comment les acteurs et les anomalies peuvent déjà être personnalisés sans modification de code, uniquement par paramétrage contredisant l'a-priori d'expertise en programmation que peut susciter le monde de l'open source et de l'usage de ses applications

2.2.2.1. Personnalisation des demandes

La maintenance applicative repose sur la notion centrale de demande ou bogue. Nous allons à présent voir les paramétrages que Mantis offre pour la description des anomalies (bogues) conformes au processus cible.

Ajout de champs personnalisés

Par défaut, une demande, ou bogue dans le vocabulaire de Mantis, est décrite par les éléments suivants :

- Numéro unique de la demande.
- Catégorie de la demande.
- Impact de la demande.
- Reproductibilité.
- Date de soumission.
- Date de la dernière mise à jour.
- Nom du rapporteur, c'est-à-dire la personne ayant créé la demande dans Mantis.
- Statut de la demande.
- Résumé ou titre.
- Description.
- Balise ou mots clés.

Bien que cette liste d'attributs soit déjà conséquente, de nouveaux attributs sont nécessaires :

ID Altiris : Dans le cadre du processus cible, certaines demandes sont créées dans Mantis à la suite d'une escalade du niveau 2 vers le niveau 3. Une demande créée dans Mantis est liée à au moins une demande dans Altiris. C'est le numéro unique de cette demande qui sera saisi dans ce champ.

La date de création de la demande : Dans le cas d'une escalade, elle est antérieure à la date de création de la demande dans Mantis. Cette information est intéressante en particulier en termes de rapport de performance sur la durée totale de traitement d'une demande. Pour distinguer les deux dates de création dans Altiris et dans Mantis, nous désignerons la première comme le champ « Demandé » et la seconde comme « Créé ».

Un numéro d'ordonnancement : Plusieurs demandes peuvent avoir la même priorité. Dans la plupart des cas, à priorité égale, la méthode de traitement est FIFO. Néanmoins, dans certains cas, l'impact métier permet de définir un ordre de traitement des demandes de même priorité. On utilisera ce champ à cet effet.

Mantis propose un formulaire pour créer rapidement ces champs personnalisés et les attribuer à un ou plusieurs projets. Les utilisateurs de Mantis, autrement dit les personnes du support niveau 3, le responsable de la TMA et le directeur informatique, seront amenés à saisir et/ou consulter ces nouveaux champs.

Comme l'indique la capture d'écran ci-dessous, le type du champ «Demandé» est date. Lors de la création d'un nouveau bogue, l'utilisateur pourra saisir la date à laquelle a été créée la demande en remplissant successivement un champ pour le jour, le mois et l'année.

Figure 16 : paramétrage d'un champ personnalisé dans Mantis

Nous verrons dans la partie modification du code source de Mantis comment alléger cette saisie en proposant un calendrier à l'utilisateur qui remplira alors tous les champs en un seul clic comme le représente la figure ci-dessous.

Figure 17 : champ date cible dans Mantis

Paramétrage des projets et sous-projets

Dans l'application Mantis, un bogue est rattaché à un projet et à un seul. Chaque projet est unique par son nom, mais peut être lui-même rattaché à un autre projet formant ainsi une arborescence.

Il est possible de paramétrer cette arborescence de fichiers en allant dans le menu administration et en s'authentifiant en tant qu'administrateur de l'application. Il faut alors saisir, comme l'illustre la figure ci-dessous, les attributs du projet :

- Le nom du projet
- L'état du projet : les valeurs possibles sont « développement », « livré », « stable » ou « obsolète ».
- L'affichage de l'état : les valeurs possibles sont « public » ou « privé ».
- L'héritage des catégories globales : « oui » ou « non ».
- Le chemin pour le dépôt de fichier: exemple pour le projet Viatelease dans un système de fichier linux du serveur de production : /var/www/mantis/Viatelease/
- La description

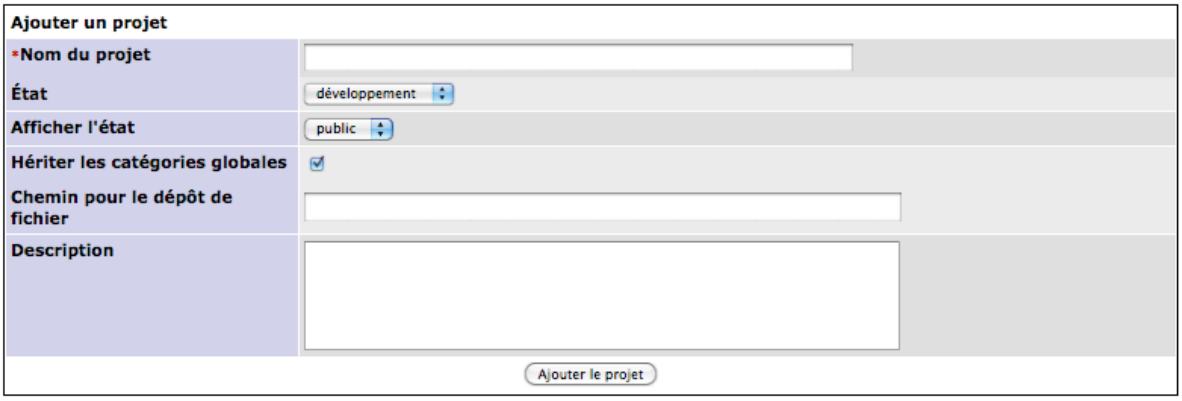


Figure 18 : paramétrage d'une arborescence de projet dans Mantis

Dans notre cas d'étude, chaque application est considérée comme un projet racine. Chaque application ou projet peut avoir autant de sous-projets que nécessaire. Chaque sous-projet correspond à un regroupement de bogues traités ensemble dans le cadre d'une opération de maintenance importante.

Mantis fournit les outils pour créer facilement des sous-projets. Pour créer un sous-projet, il faut que le projet père existe. A partir du menu de gestion des projets existants du module administration de Mantis, on saisit les informations nécessaires dans le bloc « sous-projets » représenté dans la capture d'écran ci-dessous.

Figure 19 : Paramétrage des sous projets dans Mantis

Pour toutes les applications, un sous-projet «Incidents de production» sera créé afin de faciliter l'importation de bogue depuis Altiris. En effet, la notion de sous-projet étant absente dans Altiris, il est nécessaire de créer un sous-projet invariable par application pour simplifier l'interface entre les deux applications. Tous les tickets d'incidents applicatifs enregistrés dans Altiris et escaladés au niveau 2 seront également enregistrés dans le sous-projet «Incidents de production» de l'application correspondante.

Paramétrage des catégories

Nous avons identifié les incidents de production comme une première catégorie de bogue. Les autres bogues sont :

- Evolution fonctionnelle.
- Evolution non fonctionnelle.
- Correction fonctionnelle.
- Correction non fonctionnelle.

Une évolution définit l'ajout de fonctionnalités non prévues dans le cahier des charges de l'application.

Une correction définit la modification de fonctions existantes mais dont l'exécution est source d'anomalies et dysfonctionnements.

Un bogue est qualifié de **fonctionnel** s'il se rapporte directement aux tâches métiers.

Un bogue est qualifié de **non fonctionnel** s'il se rapporte au contexte technique, légal, etc. du métier.

La capture d'écran ci-dessous illustre le bloc de création de catégorie dans le menu «gestion des projets» du module administration.

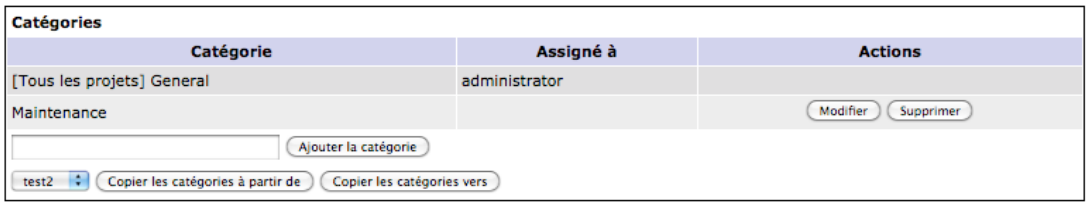


Figure 20 : Paramétrage des catégories dans Mantis

Le résumé, la description et la date de dernière mise à jour ne nécessitent pas de paramétrage particulier. Le résumé et la description sont des champs de saisie libre.

La date de dernière mise à jour est directement gérée par l'application.

Le bloc Sévérité fournit par défaut les valeurs suivantes : « fonctionnalité », « simple », « texte », « cosmétique », « mineur », « majeur », « critique » et « bloquant. »

Ces valeurs ne dépendent ni du même champ (cosmétique et bloquant) ni du même paradigme sémantique (texte, critique). Il n'est pas possible de modifier ces valeurs depuis le module d'administration. Nous étudierons la modification de cet attribut dans la partie « Modification du code source ». Il en va de même pour le champ cible « Impact » qui n'est pas proposé comme attribut par défaut par l'application Mantis.

Le champ « Affecté à » relève du paramétrage des utilisateurs. Nous allons l'aborder dans le paragraphe suivant en traitant la personnalisation de la gestion des utilisateurs.

Lors de l'ajout d'un nouveau bogue, dans le formulaire de qualification du bogue, le champ de « Reproductibilité » apparaît. A l'instar du champ « Sévérité », ce champ n'est pas paramétrable dans le module administration. Dans un souci d'économie de champs, afin d'alléger la saisie des bogues et de faciliter le changement auprès des utilisateurs, nous verrons dans la partie « Modification du code source de Mantis » comment supprimer ce champ du formulaire d'ajout de nouveau bogue.

2.2.2.2. Personnalisation de la gestion des utilisateurs/acteurs

Un processus est mis en œuvre par des acteurs. Chaque acteur doit réaliser des tâches à un moment donné. Mantis met à la disposition de l'administrateur des outils de paramétrage des utilisateurs qui seront acteurs du processus de maintenance applicative.

Mantis propose les 6 types d'utilisateurs suivants :

- Invité
- Rapporteur
- Testeur
- Développeur
- Gestionnaire
- Administrateur

Chaque profil se distingue par des droits d'accès différents. Les sous menus « Seuil de cheminement de travail » et « Rapport de permission » du menu « Configuration » du module Administration permettent de personnaliser les droits de chacun de ces profils. Le tableau ci-dessous représente les droits par défaut de chacun des utilisateurs.

Tableau 1 : Les droits des utilisateurs dans Mantis

Droits	Inv	Rap	Test	Dév	Gest	Adm
Rapporter un bogue		x	x	x	x	x
Mettre à jour l'information d'un bogue			x	x	x	x
Permettre aux demandeurs de fermer un bogue						x
Démarrer la surveillance d'un bogue		x	x	x	x	x
Gérer un bogue				x	x	x
Assigner un bogue				x	x	x
Déplacer un bogue				x	x	x
Supprimer un bogue				x	x	x
Mettre à jour un bogue qui est en lecture seule				x	x	x
Mettre à jour le statut d'un bogue				x	x	x
Voir les bogues privés				x		x
Définir l'état d'affichage lors du rapport d'un nouveau bogue ou note		x	x	x	x	x
Mettre à jour l'état d'affichage des bogues ou notes existantes			x	x	x	x
Afficher la liste des utilisateurs surveillant un bogue				x	x	x
Changer l'état de l'assignement				x	x	x
Ajouter des notes		x	x	x	x	x
Mettre à jours les notes				x	x	x
Permettre aux utilisateurs de modifier leurs propres notes de bogue				x	x	x
Effacer une note				x	x	x
Voir les notes privées des autres utilisateurs				x	x	x
Voir Changements	x	x	x	x	x	x
Voir Assigné à	x	x	x	x	x	x

Droits	Inv	Rap	Test	Dév	Gest	Adm
Voir Historique du bogue	x	x	x	x	x	x
Envoyer des rappels				x	x	x
Afficher la liste des pièces jointes	x	x	x	x	x	x
Télécharger les pièces jointes	x	x	x	x	x	x
Supprimer les pièces jointes					x	x
Mettre à jour les pièces jointes		x	x	x	x	x
Enregistrer les filtres				x	x	x
Enregistrer et partager les filtres					x	x
Utiliser les filtres enregistrés		x	x	x	x	x
Créer un projet						x
Supprimer un projet						x
Gérer les projets					x	x
Gérer les droits d'accès aux utilisateurs					x	x
Automatiquement inclus dans les projets prévus						x
Afficher les documents du projet	x	x	x	x	x	x
Envoyer les documents du projet					x	x
Gérer les champs personnalisés						x
Liés des champs personnalisés aux projets					x	x
Voir la synthèse					x	x
Voir les adresses de courriel des autres utilisateurs						
Envoyer des rappels				x	x	x
Ajouter des profils		x	x	x	x	x
Gérer les utilisateurs						x
Signaler la création d'une nouvelle utilisation						x

On note que les profils Développeurs ont légèrement moins de droit que les administrateurs et notamment :

- Ils n'ont pas le droit de fermer un bogue.
- Ils n'ont pas de droit sur la gestion d'un projet.
- Ils n'ont pas de vision de synthèse.

L'application Mantis est utilisée en interne par la DSI de Paritel. Si on se réfère au processus d'application cible (cf. 1.1.2.), les acteurs suivants interviennent dans l'application Mantis :

- Support de niveau 3
- Référent métier
- Référent Mise en production

A la différence du support et du référent de mise en production, le rôle du référent métier le conduit à créer, supprimer, modifier des projets. Il aura un profil de gestionnaire.

Les deux autres acteurs sont amenés essentiellement à modifier des bogues en ajoutant du contenu (résumé de résolution, demande d'informations supplémentaires) et en changeant le statut. Ils ont un profil équivalent au développeur.

On constate donc que le paramétrage par défaut est suffisant pour l'utilisation interne à la DSI de Mantis comme solution de maintenance applicative.

2.2.2.3. Gestion des notifications

Mantis va faciliter la mise en œuvre de Workflow réactif grâce à son mécanisme de notification.

Lors de l'étude de la cartographie cible, nous avons indiqué que Mantis permettait de mettre en œuvre une gestion de statuts des demandes de résolution ou de projet de maintenance. Certains changements de statut conditionnent le début d'une tâche d'un acteur.

La liste des statuts est la suivante :

Résolu indique que le support niveau 1 doit clôturer la demande dans Altiris et le chef de projet dans Mantis.

Commentaire indique que le chef de projet doit fournir des informations supplémentaires au support niveau 3.

Affecté indique au support niveau 3 qu'il doit commencer / poursuivre l'analyse et le traitement d'une demande.

Testé indique au chef de projet le début de la recette.

Mise en production normale indique au responsable des MEP la prise en compte de ce développement pour la prochaine MEP.

Mise en production urgente indique au responsable des MEP la nécessité de mettre en production le correctif le soir même.

En production indique au chef de projet qu'il doit rédiger un bilan de projet et informer le directeur informatique du nouveau statut.

Mantis intègre un mécanisme de gestion de notification afin d'informer chaque utilisateur des changements de statuts qui le concerne. La figure ci-dessous illustre l'écran de gestion des notifications par utilisateur

Tableau 2 : gestion des notifications dans Mantis

NOTIFICATION PAR COURRIEL										
Message	**Utilisateur ayant rapporté le bogue**	**Utilisateur prenant en charge le bogue**	**Utilisateurs surveillant ce bogue**	**Notes ajoutées par les utilisateurs.**	**Niveaux d'accès**					
					invité	**rapporteur**	**testeur**	**développeur**	**gestionnaire**	**administrateur**
Courriel en cas de nouvelle assignation	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Courriel en cas de réouverture	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Courriel à la suppression	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Courriel en cas de nouvelle note	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Courriel au changement d'une relation	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'nouveau'	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐
Changer l'état à 'affecté'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'commentaire'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'en cours d'analyse'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'en cours de traitement'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'résolu'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'à tester'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'mise en prod normale'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'mise en prod urgente'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'en production'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐
Changer l'état à 'fermé'	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐

2.2.3. Modification du code source de Mantis

Il est possible de modifier le code source de l'application Mantis pour personnaliser les demandes. Cette modification vise à faciliter la saisie et la consultation des demandes conformément aux workflow décrit dans le processus cible.

2.2.1.1. Présentation du code source de Mantis

Mantis étant un logiciel open source, nous pouvons facilement inspecter son code source afin de mieux comprendre son architecture

La figure ci-dessous illustre l'arborescence du code source de Mantis.

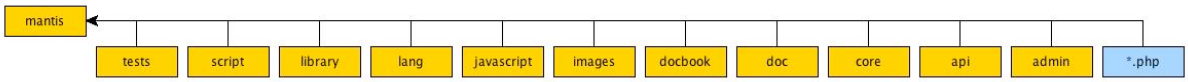


Figure 21 : Arborescence du code source de Mantis

L'application est principalement composée de fichiers PHP, c'est à dire de fichiers écrits en langage de script pour produire des pages web via un serveur de pages web (cf. 2.2.1.1 Architecture technique). PHP est un langage interprété, il est donc possible de consulter, éditer et modifier le code source de l'application Mantis sans avoir à le recompiler.

On notera, au premier niveau de l'arborescence, des répertoires dont les noms dénotent des fonctions architecturales.

- Core (cœur en anglais) dénote une fonction centrale et critique de l'application.
- Api, fait écho à l'abréviation Application Programming Interface fréquemment utilisée en ingénierie. Elle dénote des fonctions mises à disposition par les développeurs pour le reste de l'application.
- Lang, dénote des fonctions linguistiques, certainement pour proposer des interfaces dans différents langages.

Nous allons à présent détailler cette architecture, en abordant notamment les relations de dépendance existantes entre les fichiers pour l'amélioration de la saisie de la date et la personnalisation du workflow des demandes.

2.2.1.1. Ajout d'un calendrier de saisie de date

Nous avons établi la nécessité d'ajouter un champ personnalisé « Demandé » dans les attributs des projets de Mantis. Lors de la création d'une nouvelle demande dans Mantis, l'utilisateur doit saisir le jour, le mois et l'année pour renseigner ce champ. Il est possible d'en améliorer l'ergonomie en proposant un calendrier pour définir simultanément la valeur de ses 3 champs.

La page affichée lors de la saisie du champ « Demandé » est «bug_report_page.php». La figure ci-dessous illustre les pages associées à cette page par le serveur lors de son interprétation. En effet, PHP permet la réutilisation de code. Les fonctions «require_once» et «include» rendent possible le chargement de fichiers dans un script PHP et ainsi la centralisation des fonctions récurrentes.

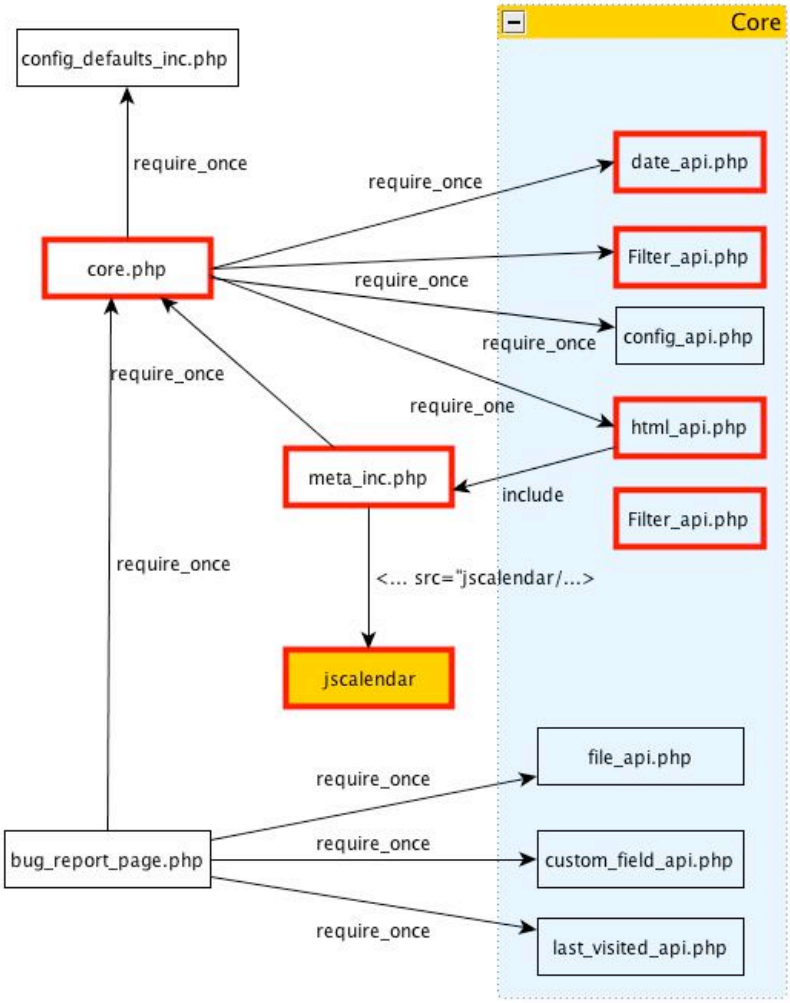


Figure 22 : Relation de dépendances pour l'ajout d'un calendrier

Le schéma ci-dessus représente les relations entre les pages de code de Mantis pour l'affichage d'un calendrier de saisie du champ «Demandé». Chaque rectangle blanc représente un fichier PHP. Les rectangles jaunes représentent des répertoires. Enfin, les rectangles dont la bordure est colorée en rouge sont les fichiers et répertoires modifiés pour la mise en œuvre d'un calendrier pour la saisie des dates.

La fonction «require_once» qui est utilisée au début du fichier «bug_report_page.php» inclut dynamiquement le fichier «core.php» lors de l'exécution du code par le serveur Apache. A la différence de l'instruction «include» utilisée dans le fichier «meta_inc.php» pour incorporer le fichier «html_api.php», cette instruction s'assure que le fichier n'a pas déjà été appelé précédemment avant de l'inclure et ne réévalue qu'une fois le code appelé. Require_once permet d'éviter les erreurs de redéclaration.

Le fichier «core.php» et les fichiers api contiennent des fonctions qui visent à être fréquemment réutilisées par le reste des fichiers de l'application, c'est pourquoi elles sont appelées par l'instruction «require_once».

L'affichage d'un calendrier depuis la page «bug_report_page.php» repose sur l'appel de script javascript, lors de la saisie d'une date dans le champ «Demandé». Ces scripts sont contenus dans le répertoire jscalendar. Le fichier «meta_inc.php» définit le contenu des métadonnées des pages html générées par le serveur. Le code ci-dessous est un extrait du fichier meta_inc.php. Il illustre l'intégration des fichiers javascript depuis le nouveau répertoire jscalendar grâce aux balises caractéristiques <script>

```
<?php

  # Include jscalendar if configured
  if ( config_get('date_use_calendar') == ON) {

    ?>
    <style type="text/css">@import url(jscalendar/calendar-win2k-1.css);</style>
    <script type="text/javascript" src="jscalendar/calendar.js"></script>
    <script type="text/javascript" src="jscalendar/lang/calendar-en.js"></script>
    <script type="text/javascript">
```

Extrait de code 2 : Méta_inc.php, ajout d'un calendrier javascript

Le fichier «meta_inc.php» est inclus dans le fichier «html_api.php» par la fonction «config_get()», définie dans le fichier «config_api.php» et appelée dans la fonction «html_page_top1()».

Comme l'indique le commentaire en gris, la fonction «html_page_top1() génère les métadonnées dans la page html qui sera lue par le navigateur du client.

```
# --------------------
# Print the part of the page that comes before meta redirect tags should
#  be inserted
function html_page_top1( $p_page_title = null ) {
    html_begin();
    html_head_begin();
    html_css();
    html_content_type();
    include( config_get( 'meta_include_file' ) );
    html_rss_link();
    echo '<link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon" />';
    html_title( $p_page_title );
    html_head_javascript();
}
```

Extrait de code 3 : html_api.php, fonction d'affichage de page mantis

La fonction «config_get()» (cf. extrait de code ci-dessus) a pour argument la chaine «meta_include_file» qui renvoie à la variable globale $g_meta_include_file définie dans le fichier de configuration par défaut «config_defaults_inc.php»

La logique de l'architecture est respectée. L'affichage d'un calendrier dans la page de saisie des demandes, comme l'illustre la capture d'écran ci-dessous, se fonde sur le fichier «core.php» (cœur) qui contient l'ensemble des modifications, à savoir :

- la modification des métadonnées définie dans le fichier «meta_inc.php», ce dernier faisant dynamiquement référence aux scripts contenus dans le répertoire jscalendar.
- la modification du comportement des champs date dont l'affichage est défini dans le fichier «date_api.php».

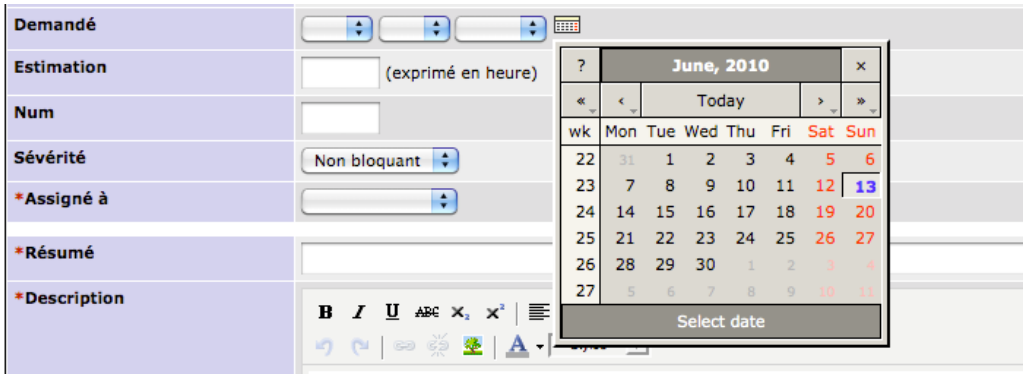


Figure 23 : Un calendrier javascript dans une fenêtre pop up de Mantis

Le fichier «filter_api.php» permet l'affichage de champs de recherche par date dans la page «Afficher les bogues». Son code, notamment la fonction print_filter_do_filter_by_date(), sera modifié de manière équivalente à «bug_report_page.php»

2.2.1.2. Modification du workflow de statuts

Dans le chapitre 1.3.3.1, nous avons défini les états nécessaires pour la gestion des demandes conformément au processus cible. Mantis ne fournit pas dans son menu d'administration, de fonction de modification du worklow des statuts par défaut.

Tableau 3 : matrice des statuts d'une demande dans Mantis

	nouveau	reconnu	confirmé	affecté	commentaire	en cours d'analyse	en cours de traitement	a tester	a corriger	mise en production normale	mise en production urgente	en production	résolu	fermé
nouveau		x	x	x	x	x	x	x		x	x		x	
reconnu			x	x	x								x	
confirmé				x	x								x	
affecté					x	x	x	x		x	x		x	x
commentaire		x	x	x		x	x	x		x	x			
en cours d'analyse				x	x		x	x		x	x			
en cours de traitement				x	x	x		x		x	x			
a tester				x	x	x	x		x	x	x			
a corriger				x		x	x	x		x	x			
mise en production normale				x		x	x		x		x	x		
mise en production urgente				x			x		x	x		x		
en production														x
résolu				x										x
fermé				x										

La matrice ci-dessus représente les transitions possibles d'un état à un autre dans le workflow par défaut et dans le workflow cible. Les cellules en vert correspondent aux statuts cible tandis que celles en violet correspondent aux statuts anciens qui n'ont pas été conservés dans la cible. Nous allons à présent étudier comment créer et supprimer les statuts des demandes, puis comment modifier le workflow par défaut.

La déclaration de ces nouveaux états de transition, autrement dit les statuts de la demande, est faite dans le fichier «custom_constant_inc.php» comme l'illustre l'extrait de code ci-dessous.

```
<?php
        define ( 'ANALYSED', 52 );
        define ( 'RUNNING', 54 );
        define ( 'TOTEST', 82 );
        define ( 'TOPUBLISHNORMAL', 84 );
        define ( 'TOPUBLISHNOW', 85 );
        define ( 'INPRODUCTION', 86 );
?>
```

Le fichier «custom_strings_inc.php» permet d'instancier la variable $s_status_enum_string par l'ensemble des statuts, en français et en anglais. On écrit le code ci-après :

```
if ( lang_get_current() == 'french' ) {
  $s_status_enum_string =
'10:nouveau,20:affecté,30:commentaire,52:en cours d\'analyse,54:en cours de
traitement,80:résolu,82:√É¬† tester,84:mise en prod normale,85:mise en prod urgente,86:en
production,90:fermé';
```

Extrait de code 5 : custom_strings_inc.php, instanciation des valeurs de statut de demande

Au cours du processus de résolution ou de mise en production d'une demande, Mantis affiche des informations sur les statuts présents et futurs dans la page d'édition de la demande. Par défaut, il n'existe pas de labels correspondants aux nouveaux statuts que nous avons créés, nous allons donc également ajouter et instancier les variables nécessaires :

- label du bouton dans la page d'édition de la demande
- titre de la page lors de l'édition de la demande
- titre du mail de notification lors de la validation du changement de statut

L'extrait de code ci-dessous illustre comment ces variables sont créées pour le statut « Mise en production normale ». Enfin, il reste à créer les chemins du graphe représenté dans le graphe d'état transition du processus cible.

```
$s_topublishnormal_bug_button = "Mise en production normale";
$s_topublishnormal_bug_title = "Mise en production normale";
$s_email_notification_title_for_status_topublishnormal ="Mise en production normale.";
```

Extrait de code 6 : Custom_strings_inc.php, instanciation des variables de présentation:

Nous allons pour cela modifier le fichier «config_inc.php» en indiquant pour chaque statut les chemins possibles, comme l'illustre l'extrait de code ci-dessous pour les premiers statuts :

```
$g_status_enum_workflow[NEW_]=
'20:assigned,30:feedback,52:analysed,54:running,80:resolved,82:totoest,84:topublishnormal,85:t
opublishnow,86:inproduction';
$g_status_enum_workflow[ASSIGNED] =
'30:feedback,52:analysed,54:running,80:resolved,82:totoest,84:topublishnormal,85:topublishnow,
86:inproduction';
$g_status_enum_workflow[FEEDBACK] =
'20:assigned,52:analysed,54:running,80:resolved,82:totoest,84:topublishnormal,85:topublishnow,
86:inproduction';
$g_status_enum_workflow[ANALYSED] =
'20:assigned,30:feedback,54:running,80:resolved,82:totoest,84:topublishnormal,85:topublishnow,
86:inproduction';
$g_status_enum_workflow[RUNNING] =
'52:analysed,80:resolved,82:totoest,84:topublishnormal,85:topublishnow,86:inproduction';
$g_status_enum_workflow[RESOLVED] =
'30:feedback,84:topublishnormal,85:topublishnow,86:inproduction,90:closed';
```

Extrait de code 7 : config_inc.php, configuration du workflow

De même que pour la création du calendrier, nous pouvons synthétiser les relations de réutilisation et d'appel des fichiers par le diagramme ci-dessous.

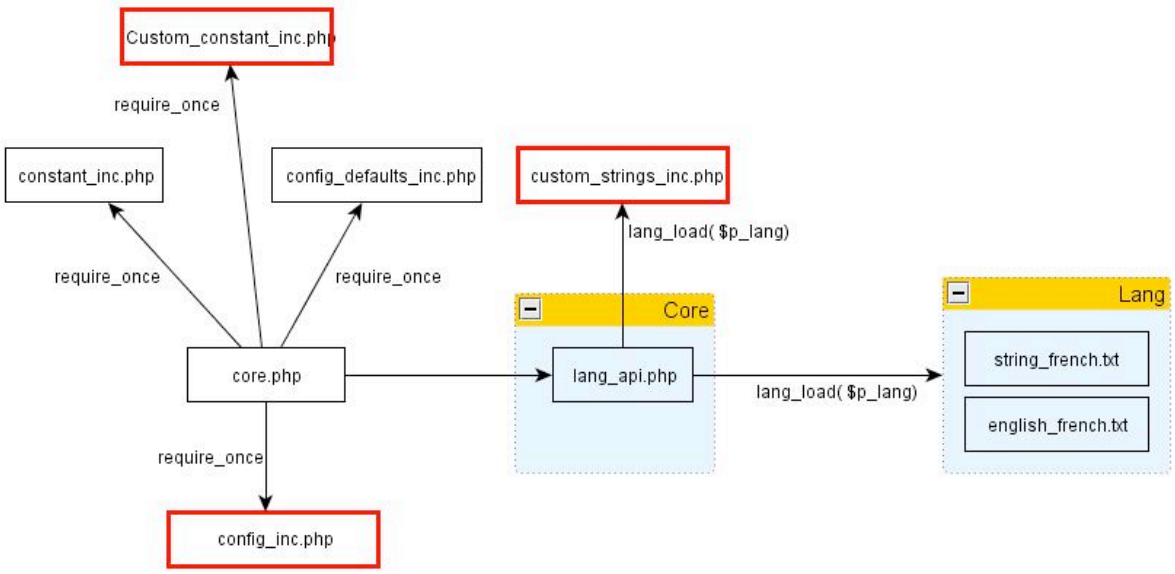


Figure 24 : Relations de dépendances pour la gestion d'un workflow

On notera la souplesse du mécanisme d'allocation des variables de Mantis. Le fichier Core.php, qui est chargé en premier, va parcourir en priorité les fichiers dans lesquels le développeur est susceptible de déclarer des variables de personnalisation « Custom_constant__inc.php et Custom_settings_inc.php. Dans le cas contraire, il va parcourir les fichiers de configuration par défaut.

Le code ci-dessous est un extrait du fichier Core.php et illustre l'appel et le parcours des fichiers « personnalisés » créés par le développeur.

```
# Load constants and configuration files
require_once(dirname(__FILE__
).DIRECTORY_SEPARATOR.'core'.DIRECTORY_SEPARATOR.'constant_inc.php' );
if
( file_exists( dirname( __FILE__ ).DIRECTORY_SEPARATOR.'custom_constant_inc.php' ) )
{require_once( dirname( __FILE__ ).DIRECTORY_SEPARATOR.'custom_constant_inc.php' );}
$t_config_inc_found = false;

require_once( dirname( __FILE__ ).DIRECTORY_SEPARATOR.'config_defaults_inc.php' );

# config_inc may not be present if this is a new install
if ( file_exists( dirname( __FILE__ ).DIRECTORY_SEPARATOR.'config_inc.php' ) )
{require_once( dirname( __FILE__ ).DIRECTORY_SEPARATOR.'config_inc.php' );
$t_config_inc_found = true;}
```

Extrait de code 8 : core.php, chargement des fichiers personnalisés

Dans ce chapitre, nous avons pu étudier comment Mantis pouvait être rapidement et simplement mis en œuvre dans un environnement de production d'un service informatique. Cette simplicité découle des nombreuses interfaces de paramétrage fournies par l'application aux utilisateurs avancés de type administrateur, mais également de l'architecture du code qui, par sa modularité, permet l'ajout d'extensions ou la déclaration de nouvelles variables (au sens objet) avec peu de modifications.

2.3. ETUDE SUR L'AJOUT D'UNE SOLUTION DE GESTION DES CONNAISSANCES : MEDIAWIKI

2.3.1. Installation de MediaWiki

2.3.1.1 Architecture technique

Comme l'illustre la figure ci-dessous, l'architecture technique de MediaWiki est similaire à celle de Mantis.

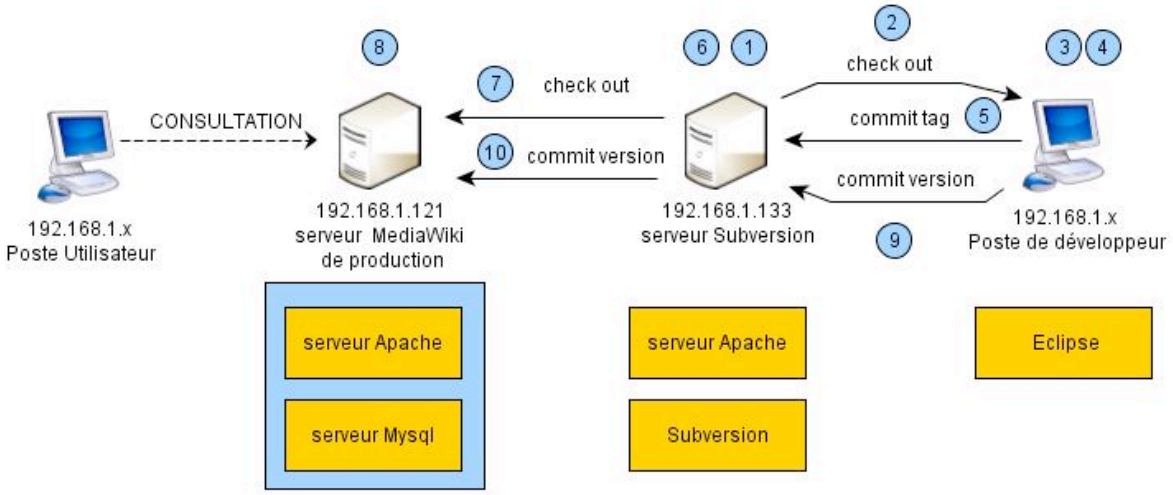


Figure 25 : Architecture technique de MediaWiki

La base de données Mysql et le serveur PHP - Apache sont installés sur la même machine. En effet, ces deux applications sont destinées à des activités de support peu critiques. En l'absence de Datacenter ou de politique de virtualisation, c'est l'hébergement mutualisé qui est le plus adapté en termes de coûts.

Les fichiers sources de MediaWiki sont téléchargeables depuis le site français de l'application[6]. On notera qu'il existe également une procédure de mise à jour de l'application. Elle se base sur :

- Un script de base de mise à jour des bases de données qui va inspecter le schéma et le mettre à jour en ajoutant des tables et des colonnes si nécessaire.

6 http://www.mediawiki.org/wiki/MediaWiki/fr

- La copie des nouveaux fichiers, dans le respect de l'arborescence des répertoires de l'application.

L'utilisation du mécanisme de release de Subversion rend l'opération sûre et rapide.

2.3.2. Paramétrage de MediaWiki

2.3.2.1 Principes d'organisation des connaissances

MediaWiki est un site web permettant d'organiser des volumes importants et riches (multimédia) de connaissances. Il permet d'organiser les connaissances à la fois par arborescence, à la manière d'un fond bibliothécaire, et par association de pages ou de termes, à la manière d'un thesaurus.

Les utilisateurs disposent d'une syntaxe spécialisée pour structurer le contenu qu'ils produisent. Les principales notions de cette syntaxe sont les suivantes :

Les espaces de nom : Ils permettent un découpage, un partitionnement des pages. Ils sont remarquables dans l'url d'une page du site. Les espaces de nom ont été créés à l'origine pour séparer les connaissances des discussions à leur propos. Leur utilisation peut être étendue à d'autres fonctions telles que le regroupement de pages par droit d'accès, par mise en forme, par type de contenu, etc.

Par défaut, MediaWiki fournit les 17 espaces de noms suivants :

Tableau 4 : liste des espaces de nom dans MediaWiki

Name	Sujet
Main	Connaissances, ensemble d'articles
Talk	Discussions sur les articles
User	Pages de comptes utilisateurs
User Talk	Discussions Pages de comptes utilisateurs
Project	Information sur le site
Project Talk	Discussions Information sur le site
File	Description du contenu multimédia
File Talk	Discussions Description du contenu multimédia
MediaWiki	Paramétrage de l'interface du site
MediaWiki Talk	Discussions Paramétrage de l'interface du site
Template	Modèle de page
Template Talk	Discussions Modèle de page
Help	Aide
Help Talk	Discussions Aide
Category	Description des catégories
Category Talk	Discussions Description des catégories
Special	Pas d'administration du site
Media	Alias des liens vers les fichiers multimédia

Il est possible d'ajouter d'autres espaces de nom mais cela requiert une légère modification du code source de MediaWiki.

Les catégories : Elles permettent une organisation par association entre les pages d'un même espace de nom. Une page peut appartenir à plusieurs catégories à différence d'une arborescence classique. En effet, en théorie des graphes, une arborescence est un arbre comportant un sommet racine à partir duquel il existe un chemin unique vers tous les autres sommets. Dans cette organisation, ce chemin n'est pas unique puisqu'un sommet « enfant » peut avoir plusieurs sommets « parents ». Elles sont remarquables dans l'url d'une page du site.

Pour créer une nouvelle catégorie, il faut publier une page vierge dont l'url se termine par le tag souhaité Par exemple pour créer la catégorie CNAM, on crée une page dont l'url sera :

```
http://{urldusite}/catégorie:CNAM
```

Pour ajouter une nouvelle page, ou un nouvel article à une catégorie, le contenu de la page doit contenir le code wiki suivant [[Catégorie:{nom}]] où nom est l'appellation de la catégorie à laquelle on souhaite l'ajouter.

Par exemple, pour ajouter la rubrique « mon mémoire » à la rubrique CNAM, on crée une page « mon mémoire » à la fin de laquelle on écrit :

```
[[Catégorie:CNAM]]
```

La page mon mémoire sera automatiquement indexée dans la page de la catégorie CNAM

Les liens : Ils sont de différents types (internes, externes, inter wikis, inter langues) et permettent la navigation vers un article ou une page regroupant les catégories.

En plus de cette organisation des connaissances, MediaWiki permet de gérer des modèles de présentation (templates en anglais) auxquels une feuille de style peut être associée.

Dans notre étude de cas, l'ensemble des connaissances doit être accessible aux personnes du support et aux développeurs pour faciliter la recherche ou/et la production et/ou la mise à jour d'informations sur les applications en production. De même, il n'est pas nécessaire d'adapter la présentation en fonction des types d'utilisateurs. Par conséquent, la création d'espace de noms supplémentaires a été écartée.

On a privilégie l'organisation des articles par catégorie. Les catégories retenues sont conformes à l'organisation des connaissances présentée au paragraphe 1.3.3.3 Mantis – MediaWiki

2.3.2.2 Adaptation de l'interface

La puissance de MediaWiki réside dans l'utilisation conjointe d'une syntaxe structurée pour l'organisation des connaissances dans un site et de la syntaxe HTML qui structure la présentation de ces connaissances dans une page.

Lors de l'étude de l'existant réalisée lors du chapitre 1, nous avions mis en avant le besoin de capitaliser les connaissances du support dans une base de connaissances qui permettrait la consultation rapide des solutions de contournement par le support niveau 2, avant l'escalade au niveau supérieur.

Le premier élément d'organisation des connaissances mis à la disposition d'un utilisateur de l'application MediaWiki est le menu de gauche.

- Il est toujours présent à l'écran : pendant la recherche ou la consultation d'informations.
- Il propose un premier niveau d'organisation des connaissances.

La capture d'écran ci-dessous représente l'écran d'accueil par défaut de l'application MediaWiki.

Figure 26 : écran d'accueil par défaut de MediaWiki

Le menu de gauche n'est pas conforme à l'organisation définie dans le processus cible (chapitre 1, §3.3.3 Mantis - MediaWiki) puisqu'il n'existe pas d'entrée pour chaque métier ou filiale de Global Concept.

La gestion des catégories, telle que vue au paragraphe précédent, est une entrée supplémentaire. Elle est compatible au principe d'organisation en arborescence qu'elle enrichit en fournissant une vision transverse de l'ensemble des applications du SI. A titre d'exemple, le menu cible contiendra les entrées suivantes :

- Accueil
- Filiales : redirige vers une page qui liste l'ensemble des filiales pour lesquelles il existe des articles (Viatelease, Paritel, Paricom, etc.).
- Applications : redirige vers une page qui liste l'ensemble des applications pour lesquelles il existe des articles.
- Modifications récentes : redirige vers une page qui liste l'ensemble des articles récemment modifiés.
- Aide : redirige vers une page qui liste des articles d'aide.

Pour modifier ce menu gauche, il suffit à un utilisateur possédant les droits d'administration de modifier la page «menu gauche» dans le nom de domaine réservé MediaWiki[7] : `http://{urldusite}/MediaWiki:Sidebar`. Cette page contient la description de l'arborescence du menu de gauche avec le titre affiché et la page de redirection correspondante comme l'illustre l'extrait suivant :

```
* navigation
** mainpage|mainpage-description
** portal-url|portal
```

Cette page fait partie des «pages spéciales». Elle est chargée par la fonction «initializeSpecialCases()» qui est une méthode de la classe MediaWiki contenue dans la page «Wiki.php» appelée lors du chargement de la page «Index.php», qui est le principal point d'accès au Wiki.

Elle est également appelée par la page «Autoloader.php», incluse dans la page «Webstart.php», elle même incluse dans la page «Index.php».

Le schéma ci-dessous illustre l'ensemble des dépendances entre les fichiers PHP pour mettre en œuvre la personnalisation du menu gauche dans MediaWiki.

7 Cf. la description des espaces de nom

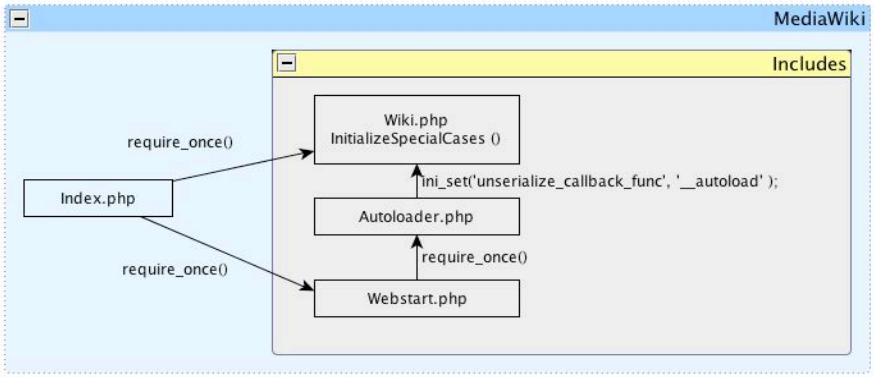


Figure 27 : Relation de dépendance pour la personnalisation du menu gauche MediaWiki

2.3.2.3 Ajout d'un éditeur convivial

Après avoir vu le paramétrage de l'interface de consultation, nous allons à présent voir comment améliorer l'interface de saisie des articles contenant les connaissances.

Les personnes du support niveau 3 et les responsables fonctionnels sont amenés à ajouter du contenu dans l'application.

Le formatage du texte dans MediaWiki se base sur l'utilisation d'une syntaxe normalisée appelée syntaxe «wiki». Elle utilise des caractères normaux comme des astérisques et des apostrophes ou accolades qui ont une fonction spéciale sur le wiki. Par exemple pour formater le terme mémoire en gras traduit par '''mémoire''' en syntaxe wiki alors que ''mémoire (espace) » le présentera en italique.

La figure ci-dessous représente les outils d'édition de texte par défaut :

Cet éditeur dispense l'utilisateur de connaitre par cœur la syntaxe wiki. Néanmoins, les

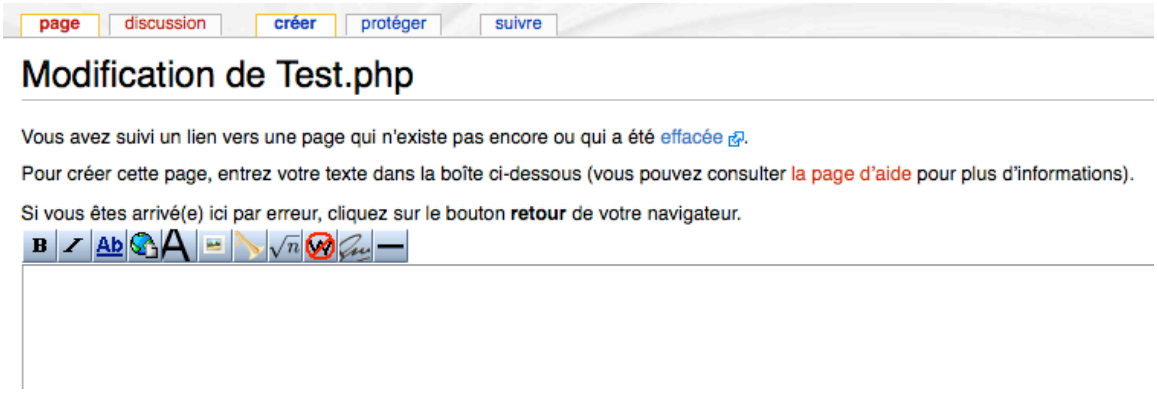


Figure 28 : éditeur de texte par défaut de MediaWiki

possibilités de formatage sont pauvres comparées à un éditeur de texte comme Microsoft Word ou Writer d'Open Office.

L'ergonomie est un des facteurs clés du succès pour un projet d'intégration d'un nouveau logiciel. Il est nécessaire de fournir aux utilisateurs de MediaWiki, habitués à des éditeurs de texte « riches », un éditeur de qualité équivalente.

Nous allons installer FCKeditor, un éditeur intuitif de type WYSWYG[8] qui fournit aussi à l'utilisateur des fonctions de mise avancées, comme l'illustre le diagramme suivant [9] :

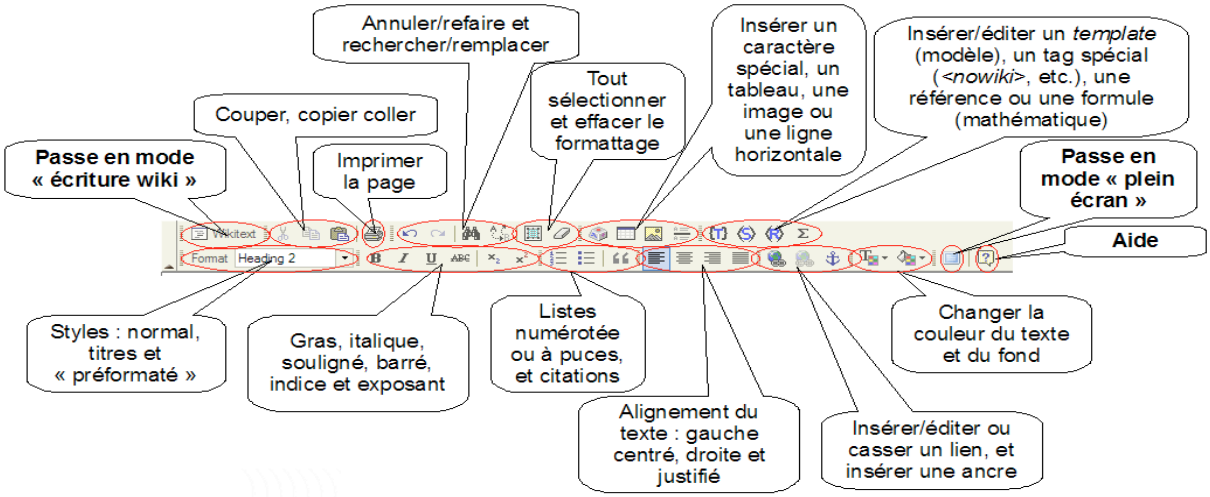


Figure 29 : Editeur de texte cible de MediaWiki

Pour installer FCKeditor, il faut récupérer les fichiers soit sur le serveur SVN à partir du client installé dans Eclipse et utilisé pour gérer les versions (cf. 2.3.1.1 Architecture technique), soit à partir du site communautaire. Après avoir importé ces fichiers dans le répertoire «Extension» de l'application, on obtient l'arborescence suivante :

```
[MEDIAWIKI ROOT]
-- AdminSettings.php
-- LocalSettings.php
-- extensions
        -- FCKeditor
                -- FCKeditor.php
                -- fckeditor
                        -- editor
```

8 What You See is What You Get : vous avez ce que vous voyez. Autrement dit, ce qui est affiché à l'écran est identique à ce qui sera consulté

9 http://rdac.zoomacom.org/wiki/

```
-- _source
...
```

Il suffit de modifier le fichier de configuration en ajoutant la ligne suivante :

```
require_once("$192.168.0.10/extensions/fckeditor/fckeditor.php");
```

Extrait de code 9 : appel d'une extension éditeur de texte riche dans MediaWiki

On considérera que l'ajout de cette ligne et l'import de fichiers dans le répertoire de Mediawiki ne sont pas équivalents à une modification du code source par un développeur. En effet, ces actions s'insèrent dans une stratégie modulaire prévue dans MediaWiki.

On retrouve le principe déjà vu lors de notre étude de l'ajout d'un calendrier dans Mantis (cf. 2.2.1.1) de cœur applicatif et d'extension. Dans le cas de Mantis, ce cœur était le répertoire Core. Dans le cas de MediaWiki, ce cœur applicatif est le répertoire «includes» qui contient l'ensemble des classes fondamentales.

Le répertoire «extension» est prévu pour contenir des groupes de fichiers PHP qui ajouteront de nouvelles fonctionnalités, telles qu'un éditeur riche, adapté aux besoins des personnes du support niveau 2 et 3.

2.3.3. Modification du code source de MediaWiki

L'ajout d'un éditeur riche a permis de mettre en évidence l'usage d'extension pour ajouter des nouvelles fonctionnalités de MediaWiki. Nous allons à présent voir comment améliorer les fonctionnalités existantes sans importer de fichiers d'extension. Il s'agit :

- D'améliorer la sécurité des données de l'environnement de production
- D'améliorer la consultation des articles avec des images dans la base de connaissances.

2.3.3.1. Modification du code de localsetting.php : sécurisation des données de production

L'architecture décrite lors de l'installation de MediaWiki se base sur l'utilisation d'un serveur Subversion pour la gestion des versions. Grace à lui, chaque développeur peut facilement développer et tester en local des extensions ou des évolutions du code de MediaWiki sans risque pour l'environnement de production mis à disposition des utilisateurs. Le serveur Subversion

garantit la sécurité du code source. Par contre, il ne garantit pas la sécurité des données. En effet, c'est la même base de données qui est utilisée dans l'environnement de développement et de production.

Le schéma ci-dessus représente l'évolution de l'architecture existante pour améliorer la sécurité des données.

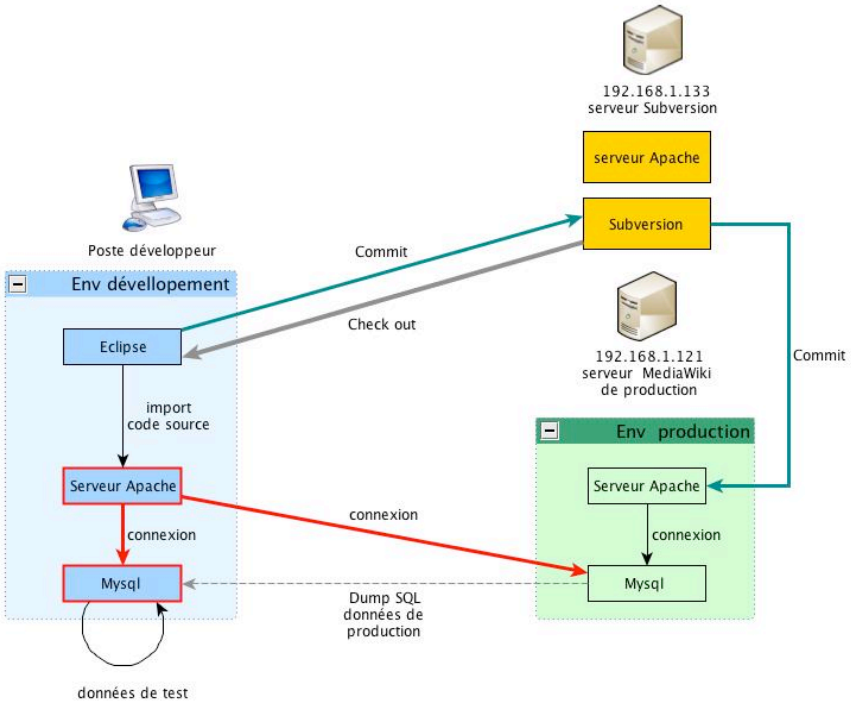


Figure 30 : Architecture cible des environnements MediaWiki

Un serveur Mysql de test est ajouté dans l'environnement de développement local. Les développeurs peuvent ainsi ajouter, modifier, supprimer des connaissances (catégories, articles, images) sans risque pour l'environnement de production. On note que cette base de données de test peut, à tout moment, être mise à jour pour être équivalente à celle en production.

Le serveur Apache de développement doit être capable de se connecter aussi bien à la base de données mysql de développement que de production. Ceci n'est pas possible sans modification du code pour deux raisons :

- Par défaut, la connexion à la base de données dans l'environnement local de développement est la même que dans l'environnement de production. En effet, cette connexion est paramétrée à l'installation et est importée lors du «check out» du code depuis le serveur subversion.
- Il n'existe pas d'Interface Homme Machine (IHM) pour le re-paramétrage de cette connexion.

Il est nécessaire de modifier le code source du cœur de l'application MediaWiki pour modifier la connexion par défaut. Le fichier localsetting.php contient les variables de connexion. Il est nécessaire de modifier les variables suivantes :

- $wgDBserver : adresse logique ou adresse ip du serveur de gestion de base de données
- $wgDBpassword : mot de passe, en clair, de connexion au serveur de gestion de base de données.

Les extraits de code ci-dessous illustrent la modification à effectuer dans le fichier localsetting.php après le premier check out. Par la suite, si le fichier Localsetting.php n'est pas modifié par le développeur, il sera exclu des fichiers à prendre en compte lors du commit et du check out sur le client et serveur SVN.

```
## Localsetting.php de PRODUCTION
## Database settings
$wgDBtype           = "mysql";
$wgDBserver         = "192.168.1.133";
$wgDBname           = "wikidb";
$wgDBuser           = "root";
$wgDBpassword       = "passwordréseau";
```

```
## Localsetting.php de DEVELOPPEMENT
## Database settings
$wgDBtype           = "mysql";
$wgDBserver         = "Localhost";
$wgDBname           = "wikidb";
$wgDBuser           = "root";
$wgDBpassword       = "passwordlocal";
```

Extrait de code 10 : localsetting.php, configuration des accès aux SGBD

2.3.3.2. Modification linker.php : amélioration de la consultation des images

Nous allons à présent améliorer la gestion de la consultation des articles et en particulier celle des images contenues dans les articles. Nous avons déjà mis en évidence l'importance de l'ergonomie de consultation pour l'adoption de MediaWiki comme base de connaissance pour les personnes du support de niveau 2 et 3 lors de la modification du menu gauche.

Nous ne traiterons pas des fonctions de téléchargement qui s'appuient largement sur l'utilisation d'extensions à l'instar de l'ajout d'un éditeur riche.

La consultation d'images est un élément important de l'ergonomie. Nous allons voir comment améliorer la consultation des articles contenant des images. En effet, lors de sa consultation, un utilisateur peut cliquer volontairement ou non sur une image. Par défaut, toutes les images sont

également des hyperliens[10]. Il est alors redirigé vers une page qui décrit les propriétés de cette image à savoir : sa date et heure d'ajout, ses dimensions et l'identité de l'utilisateur qui l'a ajoutée.

Hormis le fait que cette redirection peut venir perturber la lecture des articles, il n'est pas pertinent de fournir ces informations à tous les utilisateurs. C'est pourquoi, nous allons désactiver la fonction d'hyperlien associée automatiquement aux images dans les articles. Les personnes du support de niveau 3 et l'administrateur du site pourront les éditer à partir des pages spéciales à leur disposition depuis le menu gauche.

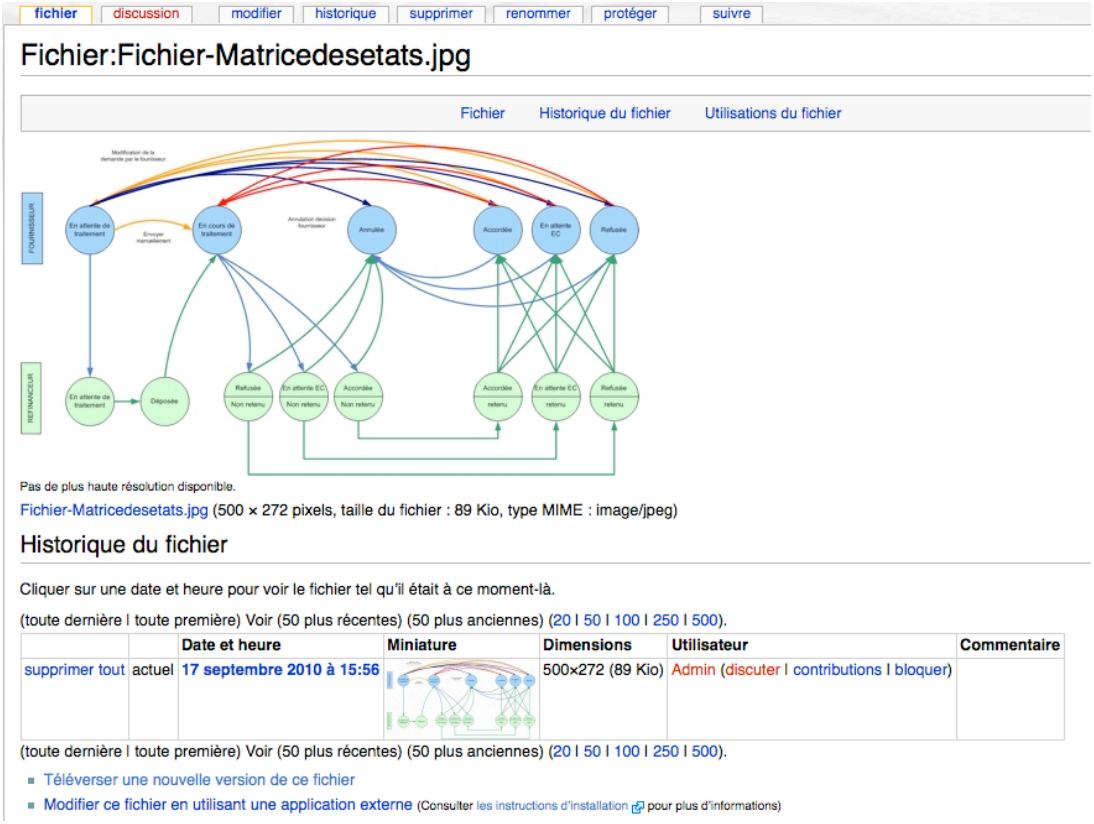


Figure 31 : Edition d'une image MediaWiki depuis la consultation d'une page

La capture d'écran ci-dessus illustre une page de redirection obtenue à partir d'un clic sur l'image Matrice des états contenu dans l'article sur les spécifications fonctionnelles de SIV.

Pour désactiver la propriété d'hyperlien, nous allons modifier le code source de la méthode makeImageLink2 du fichier Linker.php dans le répertoire Includes, c'est à dire dans le cœur applicatif de MediaWiki.

10 Un hyperlien ou lien hypertexte ou simplement lien, est une référence dans un système hypertexte permettant de passer automatiquement d'un document consulté à un document lié. Les hyperliens sont notamment utilisés dans le World Wide Web pour permettre le passage d'une page Web à une autre d'un simple clic. http://fr.wikipedia.org/wiki/Hyperlien

```
if ( !$thumb )
{ $s = $this->makeBrokenImageLinkObj( $title );
}
else
{ $s = $thumb->toHtml( array( 'desc-link'  =>  False,'alt'  =>  $fp['alt'],'valign'  =>  isset( $fp['valign'] ) ? $fp['valign'] : false ,'img-class' => isset( $fp['border'] ) ? 'thumbborder' : false ) );
 }
```

Extrait de code 11 : linker.php, modifiation de la fonction d'hyperlien d'affichage des images

Dans le code ci-dessus, on change en faux la valeur par défaut du paramètre 'desc-link' de la méthode toHtml appliquée à l'objet $thumb. Autrement dit, la présentation de l'image dans l'article (qui peut être différente de l'image elle-même d'où le terme de thumbnail qui signifie vignette en anglais) ne cumule pas, dans le code HTML, de tag d'hyperlien vers une page de propriété.

On notera que dans les versions les plus récentes de MediaWiki, cette amélioration peut être obtenue sans modifier le fichier linker.php du cœur applicatif. Cela confirme deux points :

- La gestion de la consultation des images est suffisamment importante pour être traitée sans modification d'un fichier du cœur applicatif
- La personnalisation de l'application doit se faire indépendamment de la modification des fichiers du cœur applicatif afin de préserver la stabilité de l'application et réduire les régressions fonctionnelles en cas de mise à jour de l'application.

Dans ce chapitre, nous avons étudié des cas concrets et représentatifs d'adaptation de Mantis pour la gestion des anomalies et de MediaWiki pour la gestion des connaissances. Pour chacun d'eux, nous avons pu constater à la fois la richesse des outils de paramétrage intégrés dans les IHM et la possibilité d'ajouter des nouvelles fonctionnalités en utilisant des extensions développées indépendamment des fichiers de base ou du cœur applicatif. Lorsqu'aucune extension ne répondait aux besoins de l'entreprise, nous avons vu qu'il était également possible pour un développeur de modifier directement les fonctions du cœur sans avoir à développer une extension. Cette solution, qui est la plus rapide, implique des risques, proportionnels à l'importance des modifications apportées. Les évolutions et les nombreuses extensions proposées régulièrement par la communauté open source risquent de ne plus être compatibles avec le code source modifié spécifiquement pour l'entreprise. Il s'agit d'une limite importante qui impose une expertise du développeur qui doit comprendre et respecter l'architecture logicielle dans laquelle s'inscriront ses développements. On notera que Subversion fournit dans tous les cas des outils de journalisation des modifications apportées au code sources quelle que soit la logique retenue.

Chapitre 3 - étude de l'interconnexion des applications open source

L'installation et le paramétrage de Mantis et MediaWiki permettent d'approcher le processus cible de support que nous avons exposé dans le premier chapitre. Après avoir constaté, dans le chapitre deux, les facilités et les limites de la mise en œuvre de chacun de ces applications, nous allons à présent étudier leur capacité à pouvoir s'intégrer dans le système d'information de Global Concept.

Un système d'information désigne l'ensemble des moyens humains, financiers, organisationnels et technologiques qui concourent à la réalisation d'activités (opérationnelles, tactiques, stratégiques) et de fonctions (vente, marketing, etc.) de l'entreprise[11]. MediaWiki et Mantis sont deux applications du Système d'information de support informatique de Global Concept. L'étude de leur capacité d'intégration permet de mesurer comment elles partagent des données et des règles de gestion avec les autres applications et également entre elles puisqu'elles font partie du même système d'information.

Nous allons, dans un premier temps, étudier l'intégration de Mantis et MediaWiki avec l'application d'annuaire de Global Concept puis leur interconnexion entre elles. Enfin, nous terminerons ce dernier chapitre par un retour d'expérience sur l'utilisation de ces deux applications par les personnes du service informatique chargées du développement et de la maintenance applicative.

3.1. INTEGRATION AVEC L'ANNUAIRE DE L'ENTREPRISE : ACTIVE DIRECTORY

3.1.1. Objectif de l'intégration avec l'annuaire Active Directory

Un annuaire est un recueil de données dont le but est de pouvoir retrouver facilement les personnes et/ou les organisations à l'aide d'un nombre limité de critères. Active Directory est la mise en œuvre par Microsoft des services d'annuaire Lightweight Directory Access Protocol. LDAP est, à l'origine, un protocole permettant l'interrogation des services d'annuaire. Ce protocole a évolué pour représenter une norme pour les systèmes d'annuaires, incluant un modèle de données, un modèle de nomination, un modèle fonctionnel basé sur le protocole Active Directory, un modèle de sécurité et un modèle de réplication.

L'annuaire Active Directory de Global Concept centralise l'ensemble des informations sur les utilisateurs de l'entreprise. Il est sollicité en particulier pour leur authentification et la gestion de

11 Définition issue du cours «Système d'information Web» de Jacky Akoka

leurs droits d'accès aux différentes applications et sources de données (serveurs de fichiers, Système de gestion de base de données) de l'entreprise

Mantis et MediaWiki, conformément aux processus cibles, nécessitent une gestion des utilisateurs qui peuvent avoir des profils de référents métiers, directeur de projet, support niveau 2 et 3, responsable TMA, chef de projet. L'intégration avec l'annuaire Active Directory évite aux utilisateurs la multiplication des mots de passe à mémoriser et simplifie la gestion de ceux-ci pour les administrateurs informatiques : le mot de passe sera commun aux différentes applications.

3.1.2. Architecture Active Directory - Mantis - MediaWiki

3.1.2.1. Architecture Active Directory existante

Le schéma ci-dessous représente les différentes connections nécessitant des mots de passe. Il n'existe aucun mécanisme de réplication par défaut. Chaque couple (nom d'utilisateur, mot de passe)[12] est représenté par une couleur différente.

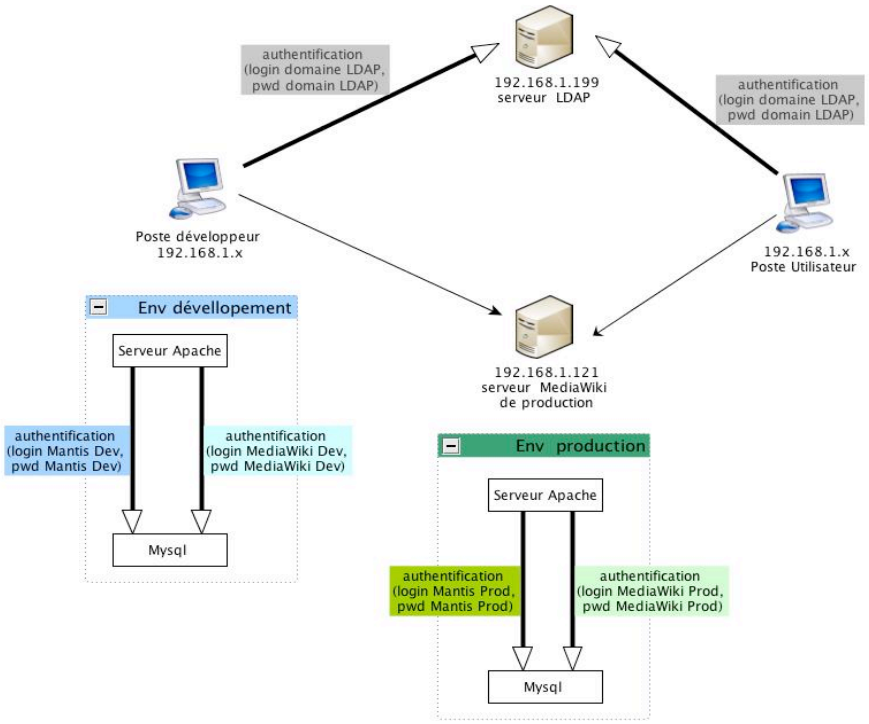


Figure 32 : Couples d'authentification des utilisateurs avant intégration

12 Traduit dans le schéma par le couple (login, password). Ici, on suppose que le login est le même dans toutes les connexions.

3.1.2.2. Architecture Active Directory cible

Dans cette architecture cible, le serveur Active Directory est le référentiel maître pour toutes les authentifications, que ce soit pour l'environnement de développement ou de production. Par commodité graphique, les mécanismes de réplication au sein des bases Mysql des applications Mantis et MediaWiki ne sont pas représentés. Ils seront détaillés dans la description de l'intégration.

On constate dans cette architecture, illustrée ci-dessous, qu'il n'existe plus qu'un seul couple pour l'ensemble des applications.

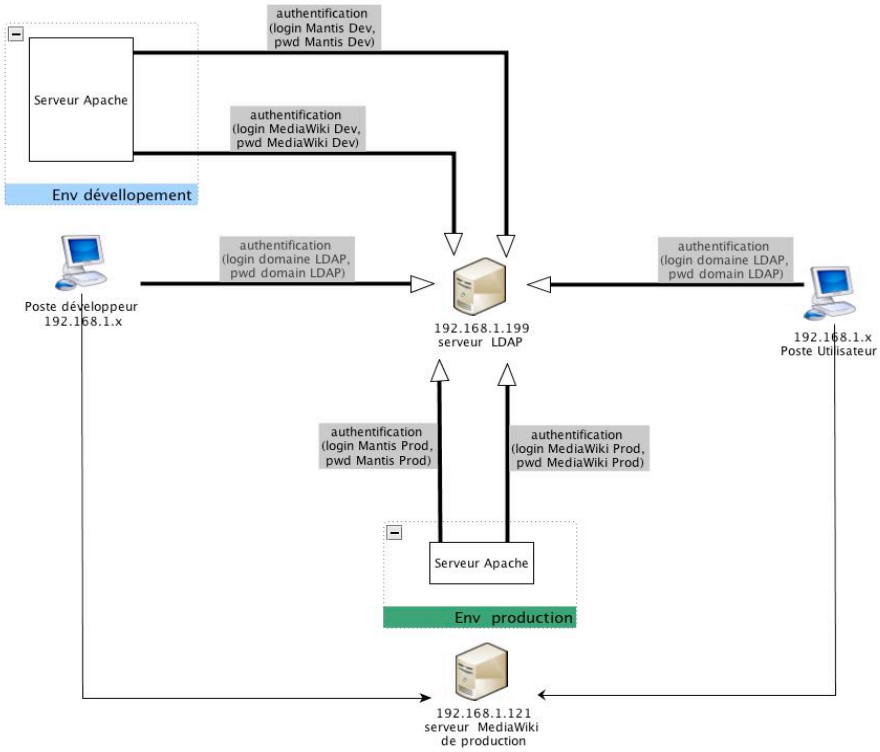


Figure 33 : Couples d'authentification des utilisateurs après intégration

Cette architecture technique cible réduit le nombre de mots de passe à retenir pour chaque utilisateur et facilite la gestion du changement, autrement dit la sécurité du système d'information.

3.1.2.3. Intégration de Mantis

Il n'existe pas de plugin ou d'extension pour interconnecter Mantis et l'Active Directory. Cependant, il existe dans le core, un fichier d'API qui contient les fonctions de gestion d'authentification des utilisateurs : authentification_api.php ainsi qu'un fichier pour la gestion

des échanges conforme au protocole LDAP : ldap_api.php. Enfin, il existe dans le fichier de paramétrage général par défaut : config_default_inc.php, une variable globale dont la valeur définit le type d'authentification

```
/**
* login method
* CRYPT or PLAIN or MD5 or LDAP or BASIC_AUTH
* You can simply change this at will. MantisBT will try to figure out how the passwords were encrypted.
* @global int $g_login_method
*/
        $g_login_method = MD5;
```

Extrait de code 12 : config_default_inc.php, déclaration de la méthode d'authentification

Le code ci-dessus est extrait du fichier config_default_inc.php. On y voit décrites les différentes valeurs possibles pour la méthode d'authentification : LDAP, MD5. Par défaut, la méthode est MD5. A la connexion, l'utilisateur va entrer dans la mire d'authentification son nom d'utilisateur et son mot de passe. Mantis va alors créer à partir d'eux une chaine chiffrée unique en utilisant la fonction de hachage MD5[13]. Cette chaine unique est comparée à celle stockée en base (Mysql) pour autoriser ou non la connexion de l'utilisateur. Si la comparaison est positive, alors la chaine issue du hachage est stockée dans un cookie[14] sur le poste de l'utilisateur afin d'éviter à Mantis de la recalculer au cours de la navigation lorsque le système vérifie l'identité de l'utilisateur. Le schéma ci-dessous illustre cette séquence d'authentification.

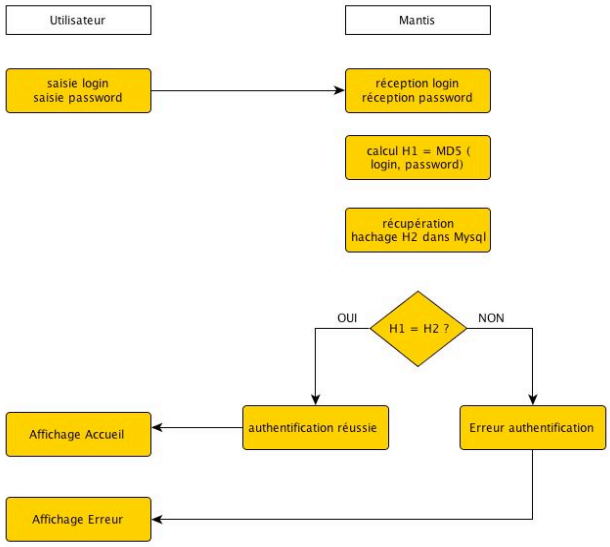


Figure 34 : Séquence technique d'authentification après intégration

13 MD5 (Message Digest 5) est une fonction de hachage cryptographique qui calcule, à partir d'un fichier numérique, son empreinte numérique (en l'occurrence une séquence de 128 bits ou 32 caractères en notation hexadécimale) avec une probabilité très forte que deux fichiers différents donnent deux empreintes différentes. Cf. http://fr.wikipedia.org/wiki/Md5

14 En informatique, un cookie (aussi appelé témoin) est défini par le protocole de communication HTTP comme étant une suite d'informations envoyée par un serveur HTTP à un client HTTP, que ce dernier retourne lors de chaque interrogation du même serveur Cf. HTTP. http://fr.wikipedia.org/wiki/Cookie

Pour permettre aux utilisateurs de s'authentifier en utilisant leur nom d'utilisateur et le mot de passe enregistrés dans l'Active Directory, nous allons modifier cette méthode d'authentification. En effet, le mot de passe de l'Active Directory n'est plus stocké en base de données mais dans l'Active Directory. La fonction comparaison sera dès lors toujours négative.

Dans notre cas d'étude, l'application Mantis était déjà utilisée par de nombreux utilisateurs. Or les noms d'utilisateur employés pour s'authentifier sur Mantis ne sont pas les mêmes que dans Active Directory.

Dans AD, le nom d'utilisateur est formé par la première lettre du prénom suivie d'un point suivi du nom de la personne. Ex : s.glattleider pour Stéphane Glattleider.

Dans Mantis, le nom d'utilisateur est formé par la première lettre du prénom directement suivie du nom. Ex : sglattleider pour Stéphane Glattleider.

Nous devrons nous assurer que la compatibilité descendante est préservée, le temps d'informer tous les utilisateurs. Autrement dit, les utilisateurs devront toujours s'authentifier avec leur ancien nom d'utilisateur et mot de passe.

Nous allons d'abord modifier la fonction d'authentification dans le fichier authentification_api.php dans le core.

```
#   true is returned.  If $p_perm_login is true, the long-term
 #   cookie is created.
       function auth_attempt_login( $p_username, $p_password, $p_perm_login=false ) {

       $t_user_id = user_get_id_by_name( $p_username );
       $t_login_method = config_get( 'login_method' );

       if ( false === $t_user_id ) {
       if ( BASIC_AUTH == $t_login_method || LDAP == $t_login_method ) {
               if ( BASIC_AUTH == $t_login_method) {
               $t_cookie_string = user_create( $p_username, $p_password );
                }
               elseif ( LDAP == $t_login_method ) {
                $t_cookie_string = user_create( $p_username, '');
                }

               if ( false === $t_cookie_string ) {
               # it didn't work
                     return false;
                }

               if ( BASIC_AUTH == $t_login_method ) {
               # attempt to create the user if using BASIC_AUTH
               $t_cookie_string = user_create( $p_username, $p_password );

               if ( false === $t_cookie_string ) {
               # it didn't work
               return false;
```

```
}
```

Extrait de code 13 : authentification_api.php, modification de la fonction d'authentification

Dans l'extrait de code ci-dessus, le code en bleu a été ajouté à la partir du code en noir et vert qui est celui d'origine. Si le mode d'authentification LDAP est retenu, la chaine unique est créée à partir du nom d'utilisateur ($p_username) et non plus à partir du nom d'utilisateur et du mot de passe comme dans le mode d'authentification «basique», en vert. Le principe de création du cookie est conservé.

Par défaut, Mantis n'autorise pas les noms d'utilisateur avec un point. Cette contrainte peut être contournée en modifiant les caractères acceptés dans le fichier de configuration général par défaut config_default_inc.php

```
$g_user_login_valid_regex = '/^[\w \.\-]+$/';
```

La variable $g_user_login_valid_regex est utilisée par la fonction native php preg_match dans la fonction user_is_name_valid, ci-dessous, qui est elle-même appelée par la fonction user_ensure_name_valid visant à vérifier la validité du nom d'utilisateur saisi dans la mire d'authentification dans le fichier api, user_api.php du core

```
# Check if the username is a valid username (does not account for uniqueness)
#  realname can match
# Return true if it is, false otherwise
function user_is_name_valid( $p_username ) {

        # The DB field is hard-coded. USERLEN should not be modified.
        if( utf8_strlen( $p_username ) > USERLEN ) {
                return false;
        }

        # username must consist of at least one character
        if( is_blank( $p_username ) ) {
                return false;
        }

        # Only allow a basic set of characters
        if( 0 == preg_match( config_get( 'user_login_valid_regex' ), $p_username ) ) {
                return false;
        }

        # We have a valid username
        return true;
}
```

Extrait de code 14 : user_api.php, validation de la forme du login

Cette partie de code permet à Mantis de vérifier l'existence de l'utilisateur et le mode d'authentification à mettre en œuvre. Nous devons nous questionner à présent sur la séquence d'authentification en cas de mode Active Directory.

Pour pouvoir interroger l'Active Directory conformément au protocole LDAP, il faut fournir à Mantis :

- Le chemin de l'arborescence à interroger.
- Le compte et le mot de passe système autorisé à interroger l'Active Directory.
- La version du protocole LDAP à utiliser.

```
# look in README.LDAP for details

# --- using openldap -------------
$g_ldap_server = 'ldap://192.168.1.99/';
$g_ldap_port   = '389';
$g_ldap_root_dn = 'CN=Users,DC=globalconcept,DC=fr';
$g_ldap_organization  = '';
$g_ldap_uid_field = 'sAMAccountName'; # Use 'sAMAccountName' for Active Directory
$g_ldap_bind_dn = 'CN=ldap,CN=Users,DC=globalconcept,DC=fr';
$g_ldap_bind_passwd = 'Password';
$g_use_ldap_email = OFF;
$g_ldap_protocol_version = 3;
```

Extrait de code 15 : config_defaults_inc.php, configuration de l'accès LDAP pour Mantis

Les lignes de code ci-dessus sont ajoutées au fichier de configuration par défaut config_defaults_inc.php. Elles fournissent toutes les variables nécessaires Mantis :

- L'adresse du serveur LDAP
- Le port de communication
- La description de la branche contenant les utilisateurs dans l'annuaire
- Le nom et le mot de passe du compte système pour interroger Active Directory
- Le type de protocole LDAP à utiliser avec un serveur Active Directory Microsoft 2000.

Enfin, on notera que dans ce mode d'authentification, l'unicité des noms d'utilisateur (seul élément présent en base) qui est garantie par Active Directory.

3.1.2.3. Intégration de MediaWiki

Dans le cas de MediaWiki, L'intégration avec le serveur Active Directory nécessite l'installation d'une extension disponible depuis le site officiel de l'application15. A l'instar de l'extension FCKeditor, on importe les fichiers dans le répertoire extension.

Le fichier localsetting.php qui est chargé lors de l'accès à l'application doit être modifié pour prendre en compte l'extension Active Directory.

15 http://www.mediawiki.org/wiki/Extension:Active Directory_Authentication#Installation

```
#DNs in $wgActive DirectoryRequiredGroups must be lowercase, as search result attribute values
are...
$wgActive DirectoryDomainNames = array( "Active Directory");
$wgActive DirectoryServerNames = array("Active Directory"=>"192.168.1.99");
$wgActive DirectoryEncryptionType = array("Active Directory"=>"clear");
$wgActive DirectorySearchStrings = array("Active Directory"=>"CN=USER-
NAME,CN=Users,DC=globalconcept,DC=fr"
);
require_once( "$IP/extensions/Active DirectoryAuthentication.php" );
$wgAuth = new Active DirectoryAuthenticationPlugin();
```

Extrait de code 16 : localsetting.php, configuration de l'accès à LDAP pour MediaWiki

L'extrait de code illustre la définition des variables propres à Global Concept pour la connexion au Active Directory, à savoir :

- Le nom du domaine Active Directory
- Le nom du serveur Active Directory qui est ici désigné par son adresse IP plutôt que par son nom logique DNS
- Le type de chiffrement utilisé pour la communication. Dans notre cas, la communication n'utilise ni le protocole Secure Socket Layer (SSL), ni le protocole Transport Layer Security (TLS).
- Les éléments, common name (cn) et domain component (dn) recherchés pour l'authentification. Dans notre cas, la connexion au serveur Active Directory est directe et ne passe pas par un serveur proxy.

Après avoir instancié ces variables, on inclut par la fonction require_once, le code source du fichier Active Directory «authentification.php» qui constitue notre extension. On peut alors instancier une nouvelle classe de «Active DirectoryAuthentificationPlugin» afin de fournir à Mantis les méthodes nécessaires pour assurer son intégration avec le serveur Active Directory.

3.2. INTERCONNEXION DE MEDIAWIKI ET MANTIS

3.2.1. Valeur ajoutée de la solution

L'interconnexion de Mantis et MediaWiki avec l'Active Directory simplifie l'authentification des utilisateurs qui n'ont plus qu'un seul nom d'utilisateur et un seul mot de passe à retenir pour utiliser toutes ces applications. De plus, elle améliore le support de ces applications puisque le support n'a pas assuré la synchronisation de plusieurs référentiels d'authentification.

L'interconnexion entre Mantis et MediaWiki présente d'autres avantages pour la maintenance des applications. En liant un ticket Mantis à un article de la base de connaissance dans MediaWiki, on améliore les points suivants :

- Ajout simple et rapide de contenus à la base de connaissances
- Mise à jour rapide des articles de la base de connaissances par rapport à l'existant en production et à son évolution.
- Amélioration de la recherche de solutions de contournement ou de solutions de résolution par les supports niveau 2 et 3 grâce à une documentation complète et à jour des applications à maintenir.

Il est donc souhaitable de mettre en place la cartographie applicative cible décrite au paragraphe 1.3.3.3 dont le schéma ci-dessous rappelle les principes.

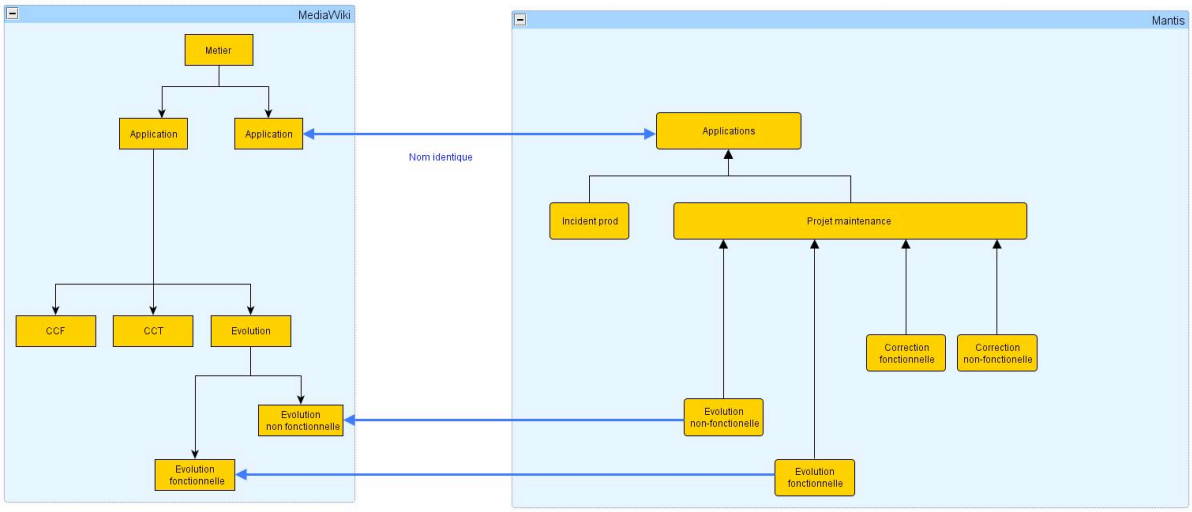


Figure 35 : architecture applicative cible

Les incidents et problèmes gérés dans Mantis sont une partie des connaissances abordées dans MediaWiki. Tous les tickets dans Mantis n'ont pas vocation à être référencés dans la base de connaissances. Le schéma ci-dessus décrit l'organisation des connaissances et des tickets ainsi que les équivalences de notions qui garantissent la pertinence de l'interconnexion. La notion d'application est transverse. Les tickets portant sur les évolutions fonctionnelles sont équivalents aux articles portant sur les évolutions des applications dans la base de connaissances. Idem pour les évolutions non fonctionnelles.

On notera que dans l'organisation cible de MediaWiki, nous ne faisons pas apparaître d'équivalence entre les incidents correctifs et les articles. Il s'agit d'un choix fonctionnel et non technique. La base de connaissances a pour objectif de décrire l'existant. Pour une meilleure

adoption des outils, nous souhaitons éviter une confusion entre les outils de suivi des demandes (Mantis, Altiris) et l'outil de gestion des connaissances (MediaWiki) pouvant servir à l'analyse et au diagnostic de ces demandes.

3.2.2. Réalisation de la solution

Il s'agit de permettre à MediaWiki d'afficher le contenu d'un ticket Mantis ou d'une partie d'un ticket Mantis. Pour ce faire, il faut fournir les fonctionnalités suivantes :

- Permettre à MediaWiki de consulter la base de données de Mantis.
- Sélectionner dans la base de données de Mantis un ticket et son contenu ou une partie de son contenu (Titre, Résumé, Date de mise à jour, date de création, personne affectée et description)
- Définir une syntaxe spécifique dans MediaWiki pour spécifier le ticket et le contenu de ce ticket à afficher.

En nous basant sur le principe d'ajout d'extension, nous allons réutiliser une extension existante que nous allons améliorer pour mieux répondre à nos besoins.

3.2.2.1 Analyse de l'extension MantisIntegration.php

Il existe sur le site communautaire de MediaWiki[16], une extension écrite par Alexander Botero qui permet l'interconnexion de Mantis avec MediaWiki. Elle se base sur 2 fonctions : wfMantisExtension et showMantis .

wfMantisExtension ajoute à la classe de l'analyseur syntaxique (Parser) les mots clés «mantis».

Le code source est reproduit ci-dessous

```
function wfMantisExtension() {
	global $wgParser;
	$wgParser->setHook( "mantis", "showMantis" );
```

Extrait de code 17 : mantisIntegration.php, fonction wfMantisExtension

ShowMantis contient les paramètres pour se connecter sur le serveur Mantis, sélectionner un ticket et retourner le contenu de ce ticket dans MediaWiki. Il est important de noter que la

16 http://www.mediawiki.org/wiki/Extension:MantisIntegration

fonction ne permet pas de sélectionner une partie du ticket tel que défini dans les besoins. Nous allons nous baser sur l'extrait de code ci-dessous pour étudier le mécanisme de cette fonction.

```
function showMantis( $input, $argv, &$parser) {

  $mantisDBSERVER="";
  $mantisDBUSER="";
  $mantisDBUSERPW="";
  $mantisDBNAME="";
  $mantisDBPrefix="mantis_";
  $mantis_home = "http://mantis.xxxxxx.de/mantis/view.php?id=";
```

Extrait de code 18: mantisIntegration.php, fonction showMantis

On définit dans la fonction les variables pour se connecter au serveur Mysql de Mantis.

$input est le premier argument de la fonction, il représente le numéro du ticket dont le contenu est recherché depuis MediaWiki. Il est appelé lors de la requête principale de la fonction.

```
$sql = sprintf("SELECT id, convert(convert(summary using utf8), binary), status FROM ".$mantisDBPrefix."bug_table where id=%d", mysql_real_escape_string($input));
$qryres = mysql_query($sql);
```

Extrait de code 19 : mantisIntegration.php, sélection du contenu d'une demande

La requête correspond à la sélection d'un identifiant, du résumé (summary) et du statut d'un enregistrement de la table mantis_bug_table dont l'id est le numéro correspondant à la variable $input.

La variable $sql correspond à la requête sous forme de chaîne formatée. Elle est ensuite exécutée et son résultat est stocké, sous forme de tableau, dans la variable $qryres (abréviation de query result, résultat de requête en anglais).

L'image ci-dessous illustre la structure de la table mantis_bug_table et des champs qui la constituent.

Figure 34 : Structure de la table de gestion des demandes de Mantis

local ▸ mantis ▸ mantis_bug_table

Afficher | Structure | SQL | Rechercher | Insérer | Exporter | Importer | Opérations | Vider | Supprimer

Champ	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
id	int(10)		UNSIGNED	Non	*Aucun*	AUTO_INCREMENT	
project_id	int(10)		UNSIGNED	Non	0		
reporter_id	int(10)		UNSIGNED	Non	0		
handler_id	int(10)		UNSIGNED	Non	0		
duplicate_id	int(10)		UNSIGNED	Non	0		
priority	smallint(6)			Non	10		
severity	smallint(6)			Non	0		
reproducibility	smallint(6)			Non	0		
status	smallint(6)			Non	10		
resolution	smallint(6)			Non	10		
projection	smallint(6)			Non	10		
category	varchar(64)	latin1_swedish_ci		Non			
date_submitted	datetime			Non	1970-01-01 00:00:01		
last_updated	datetime			Non	1970-01-01 00:00:01		
eta	smallint(6)			Non	10		
bug_text_id	int(10)		UNSIGNED	Non	0		
os	varchar(32)	latin1_swedish_ci		Non			
os_build	varchar(32)	latin1_swedish_ci		Non			
platform	varchar(32)	latin1_swedish_ci		Non			
version	varchar(64)	latin1_swedish_ci		Non			
fixed_in_version	varchar(64)	latin1_swedish_ci		Non			
build	varchar(32)	latin1_swedish_ci		Non			
profile_id	int(10)		UNSIGNED	Non	0		
view_state	smallint(6)			Non	10		
summary	varchar(128)	latin1_swedish_ci		Non			
sponsorship_total	int(11)			Non	0		
sticky	tinyint(4)			Non	0		
target_version	varchar(64)	latin1_swedish_ci		Non			

Le résultat affiché est la chaine constituée du numéro identifiant du ticket, son statut et son résumé. Le libellé du statut n'étant pas stocké dans la table (qui ne contient que sa valeur numérique), la fonction, dont l'extrait de code est affiché ci-dessous, contient une structure conditionnelle qui permet de déduire et d'afficher le libellé correspondant.

```
if ($row = mysql_fetch_row($qryres)) {
      switch ($row[2]) {
          case 10:
               $state_msg="new";
               break;
          case 20:
               $state_msg="feedback";
               break;
          case 30:
               $state_msg="acknowledged";
               break;
          case 40:
               $state_msg="confirmed";
               break;
          case 50:
```

```
            $state_msg="assigned";
            break;
        case 60:
            $state_msg="working on";
            break;
        case 80:
            $state_msg="resolved";
            break;
        case 90:
            $state_msg="closed";
            break;
        default:
            $state_msg="unkown";
    }
    $res = "Bug ID:".$row[0]." (".$state_msg.") : ".$row[1];
```

Extrait de code 20 : mantisIntegration.php, sélection simple de contenu de ticket

Ce résultat sera soumis à une méthode de la classe $Parser et instancié comme variable $output par l'illustration du code ci-dessous.

```
$output = $parser->parse("[$bug_page $res]", $parser->mTitle, $parser->mOptions, false, false);
return $output->getText();
```

3.2.2.2 Modification du code de l'extension

Nous allons à présent modifier le code de cette extension pour permettre une sélection plus fine et un affichage plus précis des attributs d'un ticket. Il existe plusieurs façons d'envisager la mise en œuvre de cette modification. Nous allons privilégier celle qui a l'impact le plus faible sur le code existant.

Nous nous baserons sur la possibilité de passer un tableau d'argument $argv à la fonction en plus du numéro de ticket représenté par $input. Les arguments correspondent aux éléments du ticket que nous souhaitons spécifiquement afficher :

- Le titre
- le résumé
- La dernière date de mise à jour
- La date de création
- Le nom de la personne qui a résolu le ticket ou qui est en charge de le résoudre
- La description.

Puisque nous avons adopté la stratégie la moins impactante sur le code, la requête initiale n'est pas suffisante pour remonter la valeur de tous les éléments du ticket. Plutôt que de modifier cette requête initiale, nous allons créer une nouvelle structure conditionnelle basée sur la valeur de la

variable $argv en fonction de l'élément spécifié à afficher. Une nouvelle requête sera exécutée et son contenu sera retourné pour être affiché.

```
if(!empty($argv))
{
$fields = array_values($argv);
$field = $fields[0];
$res = '';
switch($field)
 {
 case 'summary':
 case 'resume':
 $sql = "SELECT b.id, b.summary FROM ".$mantisDBPrefix."bug_table b where
 b.id=".mysql_real_escape_string($input);
 $qryres = mysql_query($sql);
 if (!$qryres) { return "<hr><b>Erreur pendant la requÃªte :". mysql_error() ;}
 if($row = mysql_fetch_row($qryres)) {
 $res = $row[1];
 }
 else{
 $res = "Aucun bug mantis Ã  ce numÃ©ro.";
 }
 break;
 case 'lastupdate':
 case 'lastup':
 case 'maj':
 case 'dateup':
 case 'dateupdated':
 $sql = "SELECT b.id, b.last_updated FROM ".$mantisDBPrefix."bug_table b where
b.id=".mysql_real_escape_string($input);
 $qryres = mysql_query($sql);
 if (!$qryres) { return "<hr><b>Erreur pendant la requÃªte :". mysql_error() ;}
 if($row = mysql_fetch_row($qryres)) {
 $res = $row[1];
 }
 else{
 $res = "Aucun bug mantis Ã  ce numÃ©ro.";
 }
 break;
```

Extrait de code 21 : mantisIntegration.php, sélection avancée de contenu dans les demandes

L'extrait de code ci-dessus illustre une partie de la nouvelle structure conditionnelle. La variable $field, extraite du tableau d'argument $argv passé par l'utilisateur, est testée. En fonction de sa valeur, une nouvelle requête, $sql dont la condition id est la même que la requête initiale, est

créée puis exécutée. Le résultat est stocké dans un variable de type tableau $res[17]. Si aucun argument n'est passé, le code initial de la fonction est exécuté.

On notera dans la structure conditionnelle et la possibilité d'utiliser plusieurs mots clé pour l'affichage de la même valeur. Cela permet à MediaWiki de s'adapter aux habitudes de saisie des différents utilisateurs. Par exemple les mots clés «resume» et «summary» retournent tous les deux le titre du ticket.

3.2.3. Limites de la solution

La mise en œuvre de l'interconnexion entre Mantis est MediaWiki est conforme aux spécifications fonctionnelles que nous avions décrites. Elle permet depuis MediaWiki un affichage d'une grande finesse des éléments pertinents qui facilitent, pour le support niveau 2 et 3, la recherche, le diagnostic et la résolution d'une demande de maintenance. Néanmoins, malgré cette richesse, deux limites importantes sont à souligner :

L'enrichissement de la base de connaissance n'est pas automatique. Les utilisateurs doivent créer les articles et/ou les liens vers le contenu de Mantis qu'ils jugent significatif. Dans le cas contraire, la base de connaissances restera faible et sa valeur ajoutée peu importante.

Les sources d'enrichissement de la base de connaissances sont limitées. Le schéma de la base de données de l'application de gestion des anomalies doit être accessible et interrogeable depuis MediaWiki. Dans notre cas d'étude, les deux applications étant open source, il est aisé d'étudier le schéma de la base de données de Mantis et de définir la requête de sélection. Mais, dans le cas d'une application d'éditeur plus fermée, cette tâche peut être plus difficile, voire impossible.

17 Il s'agit d'une chaîne de caractères

3.3. RETOUR SUR EXPERIENCE

Avant de tirer une conclusion de l'ensemble de cette étude, nous allons dresser un bilan sur l'utilisation des applications Mantis et MediaWiki. L'intégration de ces deux logiciels à été réalisée dans l'entreprise Global Concept durant les mois de février et mars 2010. Cela fait donc plus de 6 mois que l'ensemble des personnes du service de direction projet a été sensibilisé et formé à leur utilisation.

Il est à noter que si les applications décrites dans cette étude ont été mises en œuvre, les processus et l'organisation cibles décrits dans la première partie, impliquant l'embauche de nouvelles personnes sur de nouveaux postes, n'ont pas été suivis par la direction.

Notre étude de retour d'expérience vise à mesurer auprès des personnes de la direction de projet, pour l'activité de maintenance applicative, la valeur ajoutée de la mise en œuvre technique des applications open sources Mantis et MediaWiki pour l'entreprise.

3.3.1. Méthode de retour d'expérience

La méthodologie retenue se base sur les 6 points suivants :

1. Définition des objectifs

Notre retour d'expérience vise les trois objectifs suivants :

- Mesurer la valeur ajoutée quantitativement et qualitativement, de la mise en œuvre des applications Mantis et MediaWiki sur le processus de maintenance applicative de niveaux 2 et 3.
- Identifier quelles sont les limites techniques et organisationnelles actuelles
- Déterminer les actions à poursuivre avec la direction au vu de ces résultats.

2. Définition du périmètre

Le pilote de l'étude est l'auteur de la présente. Les personnes interrogées sont celles utilisant directement au moins l'une des applications au sein de la direction projet, directeur compris. Les utilisateurs métiers internes à l'origine des demandes de résolution d'incident sont exclus. En effet, c'est le logiciel Altiris qui gère les tickets d'incident. Nous n'avons ni paramétré ni modifié ce logiciel d'éditeur.

3. Définir les sources de collecte

Nous avons identifié 3 sources d'information pour cette étude :

- L'application Mantis
- L'application Altiris
- Les questionnaires semi directifs distribués aux 9 personnes de notre périmètre.

4. Recueillir les expériences

Le recueil se base à la fois sur l'interrogation des bases de données des applications Mantis et MediaWiki et sur l'analyse des réponses au questionnaire

a) Requêtes SQL[18] :

REQ 1 : évolution du nombre de tickets créés par mois

```
SELECT count( id ) , year( date_submitted ) , month( date_submitted )
FROM `mantis_bug_table`
GROUP BY year( date_submitted ) , Month( date_submitted )
```

REQ 1 : évolution de la durée moyenne de résolution des tickets clos

```
SELECT year( date_submitted) , month( date_submitted),
sec_to_time(avg(Timestampdiff(Second,`date_submitted`, `last_updated`)))
from mantis_bug_table where
status = 90 OR status = 86 OR status = 80
group by year( date_submitted) , month( date_submitted)
```

REQ 2 : évolution du nombre d'articles créés par mois

```
Select  year(ar_timestamp), month(ar_timestamp), count(ar_title) from wiki_archive
group by year(ar_timestamp), month(ar_timestamp)
```

REQ 2 : évolution du nombre d'articles créés à partir de référence à Mantis, par mois

```
Select  year(ar_timestamp), month(ar_timestamp), count(ar_title) from wiki_archive
where ar_comment like "%mantis%"
group by year(ar_timestamp), month(ar_timestamp)
```

b) Questionnaire

18 SQL (sigle de Structured Query Language) est un langage informatique normalisé qui sert à demander des opérations sur des bases de données.
http://fr.wikipedia.org/wiki/Structured_Query_Language

Le questionnaire sera composé de deux parties. La première mesure la satisfaction des utilisateurs. Pour chacune des questions suivantes, les utilisateurs pourront choisir de noter de 1 à 4 leur niveau de satisfaction. Plus le chiffre est élevé, plus le niveau de satisfaction est élevé. L'utilisateur est satisfait à partir de 2.

- Comment jugez-vous l'ergonomie de l'application ?
- Êtes-vous satisfait des fonctions de recherche et de consultation ?
- Êtes-vous satisfait des fonctions de saisie et d'ajout ?
- Comment jugez-vous l'application dans son ensemble ?

Les questions portent sur les applications modifiées conformément aux chapitres précédents de cette étude.

La deuxième partie du questionnaire permet aux utilisateurs de saisir librement du texte en répondant au deux questions suivantes :

- Quelles sont les fonctionnalités que vous souhaiteriez voir développées sur l'application Mantis (citez en 5 au maximum)
- Quelles sont les fonctionnalités que vous souhaiteriez voir développées sur l'application MediaWiki (citez en 5 au maximum)

On exploitera ces résultats en créant des nuages de mots-clés. Cette technique issue des sites internet permet de représenter visuellement les termes saisis, dans les recherches ou dans les contenus du site. Les mots s'affichent dans des polices de caractères d'autant plus grandes qu'ils apparaissent souvent.

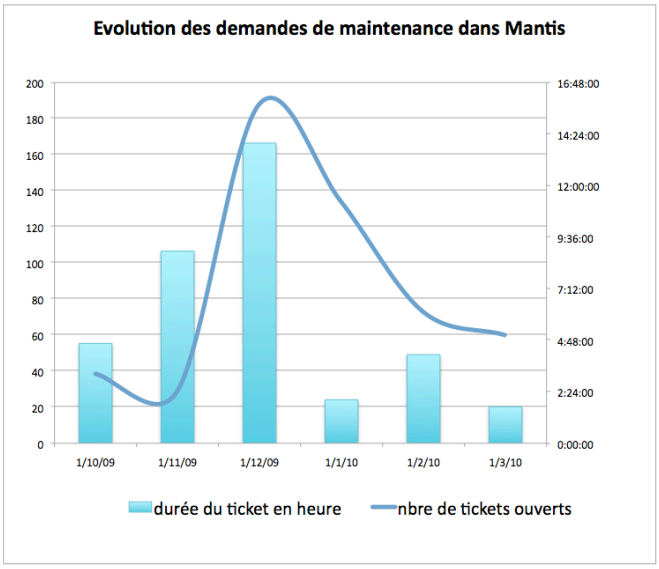
Le nuage de mots clés est généré par le site http://www.wordle.net/ créé par Jonathan Feinberg, chercheur chez IBM.

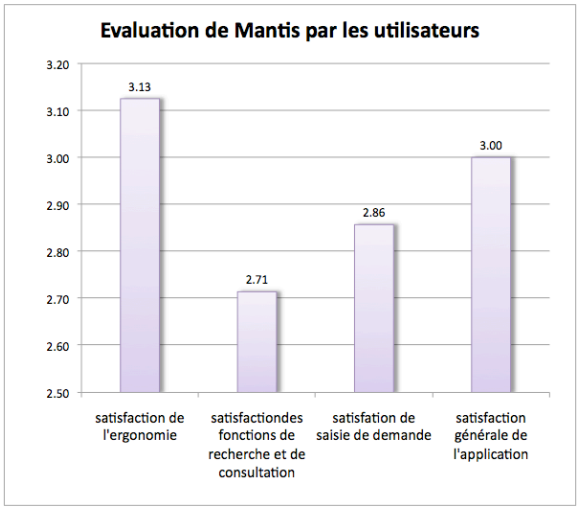
5. Analyser et synthétiser les données

6. Partager et discuter sur les résultats obtenus

3.3.2. Exploitation des résultats de retour d'expérience de Mantis

Les graphiques ci-dessous illustrent les résultats obtenus à partir des requêtes SQL (Mysql) et du questionnaire (première partie du questionnaire).





Les mesures démarrent avec la mise en service de Mantis, en octobre 2009.

Le nombre de ticket connaît une acmé à la fin de l'année. Cette évolution s'explique à la fois par la mise en production d'une nouvelle application et par le contrôle, par le directeur de projet, de la bonne utilisation de Mantis auprès des responsables de la maintenance. Par la suite, la courbe décroît et se stabilise autour de 60 tickets par mois. Encore une fois, l'absence de nouvelle mise en production en début d'année est la cause principale de cette baisse.

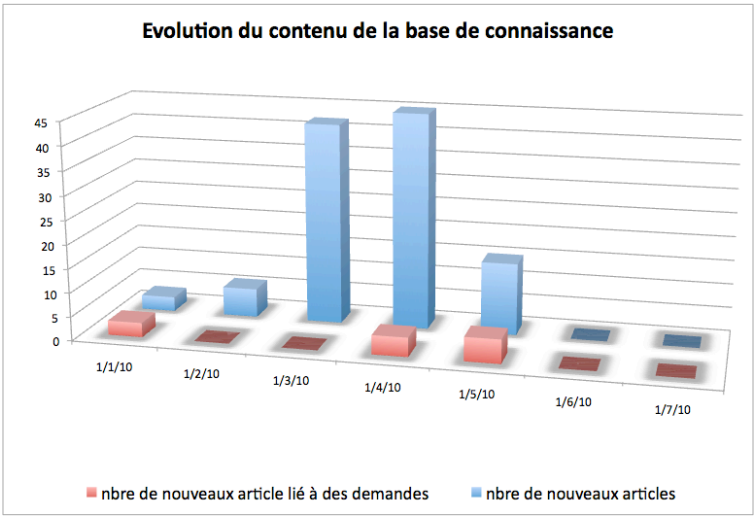
On note une évolution proportionnelle entre le nombre de tickets ouverts et leur durée de traitement, à l'exception du mois de février 2010 où la durée de traitement des tickets augmente légèrement. Cette singularité peut s'expliquer à la fois par le transfert du suivi des indicateurs

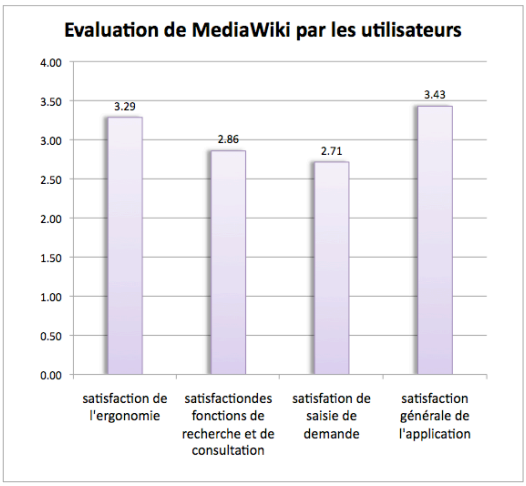
Mantis aux personnes du niveau 2 et par mon affectation sur d'autres projets **tels** que la mise en œuvre d'une base de connaissances MediaWiki

L'application est jugée satisfaisante de façon générale par les membres de la direction projet. Mais cette appréciation est à nuancer par la proximité entre les personnes interrogées et l'enquêteur. On soulignera que la recherche dans Mantis est le point le moins satisfaisant et que les termes de «reporting» et «causes» sont les éléments qui reviennent le plus souvent dans les demandes d'évolutions comme l'illustre le nuage de mots ci-contre.

3.3.3. Exploitation des résultats de retour d'expérience de MediaWiki

Les graphiques ci-dessous illustrent les résultats obtenus à partir des requêtes SQL (Mysql) et du questionnaire (première partie du questionnaire).





Les mesures démarrent avec la mise en service de MediaWiki, en janvier 2010.

On remarque une progression importante de nouveaux articles depuis le mois de janvier, date de la mise en production de l'application, jusqu'au mois d'avril. Cette augmentation est due à la formation des personnes du niveau 2 et 3 et au soutien de la direction dans la mise en œuvre de la base de connaissances. Un comité de pilotage bihebdomadaire a été organisé en présence du directeur des systèmes d'information et des personnes chargées de la maintenance durant le mois d'avril et de mai. Ceci explique le nombre très important de nouveaux articles à cette période.

A partir du mois d'avril, on constate d'abord une chute importante du nombre de tickets, puis leur absence. Une fois encore, cette évolution peut s'expliquer par le manque d'encadrement et de contrôle du processus de gestion des connaissances. En effet, forte du succès des deux premiers mois, la direction a souhaité m'affecter à un nouveau projet applicatif, ne désignant aucun autre responsable du suivi de l'enrichissement de l'application.

Paradoxalement, le niveau de satisfaction des utilisateurs est encore plus élevé que pour Mantis. Cette satisfaction traduit sans doute l'absence de critiques pour cause de «non utilisation». Ce paradoxe se traduit à travers les termes clés du nuage de mots ci-contre : «article» ou «MediaWiki» renvoie au fondement même de l'application.

Enfin on notera les mots «code» et intégration» qui laissent penser que le développement d'interconnexion avec l'outil de développement Eclipse, un outil open source, pourrait motiver les utilisateurs, à savoir les développeurs, à réutiliser la base de connaissances.

Conclusion

Au cours du premier chapitre, nous avons décrit la cible technique, organisationnelle et fonctionnelle de la maintenance applicative. Il a été exposé notamment les conditions, difficiles, dans lesquelles la société Viatelease a acquis ses applications métiers et les besoins en termes de gestion des corrections et d'évolutions qui en découlent. Il s'agit d'une forme de rançon du succès pour le groupe Global Concept dont la DSI doit accompagner la forte croissance de l'activité. Le besoin d'une maintenance applicative performante trouve son origine dans la multiplication rapide des projets qui doivent être mis en production coûte que coûte. La mise en œuvre des solutions gratuites open source MediaWiki et Mantis répond aux contraintes organisationnelles et financières de la DSI pour la mise en œuvre rapide d'une maintenance applicatives. Si notre étude de cas s'attache par la suite aux aspects techniques de cette mise en œuvre, nous avons souhaité dans ce premier chapitre décrire les processus cibles qui ont guidé le choix, la modification et l'intégration de Mantis et MediaWiki.

Dans le deuxième chapitre, nous avons mesuré la facilité avec laquelle il est possible d'apporter des améliorations fonctionnelles, conformément aux processus de maintenance cibles. En premier lieu, nous avons constaté la facilité d'obtention de ces applications et des outils logiciels pour les adapter. En effet, que ce soit la gestion des versions avec Subversion, l'édition du code source avec Eclipse ou la manipulation des données avec PhpMyadmin, tous ces outils sont gratuits et reconnus pour leurs qualités professionnelles (fonctionnalité, fiabilité, sécurité). Il s'agit ensuite des nombreuses possibilités de paramétrage dans les menus d'administration de Mantis et de MediaWiki pour répondre à des demandes fréquentes mais néanmoins importantes pour leur utilisation en entreprise. Enfin, il s'agit de souplesse fonctionnelle procurée par l'architecture logicielle. En effet, l'édition du code source de chacune de ces applications nous a permis d'identifier, dans les deux cas, la présence d'un cœur applicatif chargé de fournir les fonctionnalités de bases. Ces fonctions peuvent être améliorées ou enrichies par l'ajout ou la modification de code source dans des fichiers supplémentaires. L'architecture applicative est faite de sorte que ce nouveau code source est automatiquement pris en compte et que ces nouvelles fonctionnalités sont immédiatement disponibles sans risque de régression pour celles de base. Cette architecture incite les utilisateurs à échanger et partager leurs connaissances et leurs développements, comme le prouve la richesse des extensions disponibles sur les sites de la communauté open source. Néanmoins, cette architecture a également quelques limites. Elle implique, comme c'est le cas dans notre cas d'étude, la présence de personnes suffisamment compétentes techniquement pour participer ou tout du moins s'approprier le contenu mis à sa

disposition par la communauté. Enfin, elle implique un choix de méthode de développement des nouvelles fonctionnalités. Dans le cas ou il n'existe aucune réponse aux besoins de l'entreprise à travers les extensions ou les forums de l'application, le développeur est contraint d'analyser les fonctions d'interfaces fournies par le cœur applicatif pour développer une extension conforme à l'architecture logicielle. Il peut s'agir d'un effort non négligeable, mais qui conditionne la compatibilité de l'application avec les futures versions que la communauté proposera.

Dans le dernier chapitre, nous avons apprécié la capacité de Mantis et de MediaWiki à s'interconnecter à d'autres applications du système d'information, y compris entre elles. Autrement dit, nous avons apprécié leur capacité d'intégration. S'agissant du premier point, nous avons pu constater que Mantis et MediaWiki sont facilement capables de déléguer à l'annuaire d'entreprise, Active Directory, la tâche d'authentification des utilisateurs. Ceci permet de simplifier la gestion des mots de passe et d'éviter un accroissement des demandes de services au support chargé de leur administration, un comble pour des applications visant à améliorer le processus de maintenance. S'agissant du deuxième point, nous avons présenté comment aisément enrichir la base de connaissances à partir du contenu des demandes traitées dans Mantis. Cette fonctionnalité permet de maintenir à jour la liste des erreurs connues avec un effort de saisie moindre pour les personnes du support. L'intégration de Mantis et MediaWiki a ici une valeur plus fondamentale, puisqu'elle permet de simplifier le processus de maintenance et de convaincre les utilisateurs de la valeur ajoutée de leur utilisation dans leurs pratiques. Les résultats médiocres du retour d'expérience exposés à la fin de ce chapitre pourraient mettre en évidence qu'une solution technique performante sera peu utilisée et aura un impact faible sur les performances de maintenance si elle n'est pas accompagnée de modifications organisationnelles et d'une implication durable de la direction. Cette conclusion pourra être nuancée par des éléments récents. En effet, depuis novembre 2010, cinq mois après la fin de l'enquête de retour d'expérience, un service de maintenance applicative a été créé ainsi qu'une direction de l'innovation chargée, entre autre, de la gestion des connaissances dans l'entreprise. S'il n'existe pas de relation directe de cause à effet entre la mise en œuvre de Mantis et ces changements organisationnels, on peut y voir une volonté forte d'améliorer la maintenance applicative dans laquelle les briques open sources servent de fer de lance.

En conclusion, il apparaît que le choix technique de solutions applicatives open source est pertinent pour l'amélioration de la maintenance applicative de Viatelease. Il s'agit d'applications fiables, maintenables et rapides à mettre en œuvre. Paradoxalement, ce sont ces atouts qui représentent le plus grand risque pour le succès du projet. L'un des principaux facteurs de succès d'un projet est l'implication de la direction générale. Cette implication est souvent

proportionnelle à l'investissement financier que représente ce projet. Dans ces conditions, le faible coût d'acquisition des briques open sources peu représenter un véritable handicap. La réussite du projet repose alors sur la capacité de la DSI à conduire le changement et à convaincre les métiers de s'y impliquer.

INDEX DES FIGURES

Figure 1 : Processus existant de gestion des incidents 9
Figure 2 : Cartographie des applications de la société Ipnotic Optimitel 10
Figure 3 : Organisation de la DSI chargé de la maintenance de Viatelease 13
Figure 4 : Cartographie des applications de la société Ipnotic Optimitel 14
Figure 5 : Processus existant de gestion des incidents 17
Figure 6 : Cartographie applicative existante 18
Figure 7 : Processus cible de gestion des incidents 21
Figure 8 : Processus cible de gestion des projets de maintenance applicative.....23
Figure 9 : Cartographie applicative cible 25
Figure 10 : Graphe cible des statuts d'une demande de maintenance applicative 26
Figure 11 : Diagramme de conversion des données entre Altiris et Mantis 27
Figure 12 : Diagramme de conversion des données entre MediaWiki et Mantis 28
Figure 13 : Fonctionnalité du menu contextuel dans Eclipse.....32
Figure 14 : Fonctionnalité de gestion des données avec phpMyadmin.....33
Figure 15 : Architecture technique de Mantis 34
Figure 16 : paramétrage d'un champ personnalisé dans Mantis.....37
Figure 17 : champ date cible dans Mantis 38
Figure 18 : paramétrage d'une arborescence de projet dans Mantis 38
Figure 19 : Paramétrage des sous projets dans Mantis 39
Figure 20 : Paramétrage des catégories dans Mantis 40
Figure 21 : Arborescence du code source de Mantis 45
Figure 22 : Relation de dépendances pour l'ajout d'un calendrier.....46
Figure 23 : Un calendrier javascript dans une fenêtre pop up de Mantis 48
Figure 24 : Relations de dépendances pour la gestion d'un workflow 51
Figure 25 : Architecture technique de MediaWiki 53
Figure 26 : écran d'accueil par défaut de MediaWiki 56
Figure 27 : Relation de dépendance pour la personnalisation du menu gauche MediaWiki.....58
Figure 28 : éditeur de texte par défaut de MediaWiki 58
Figure 29 : Editeur de texte cible de MediaWiki.....59
Figure 30 : Architecture cible des environnements MediaWiki 61
Figure 31 : architecture cible de sécurisation des données 62
Figure 32 : Couples d'authentification des utilisateurs avant intégration.....66
Figure 33 : Couples d'authentification des utilisateurs après intégration 67
Figure 34 : Structure de la table de gestion des demandes de Mantis 76

INDEX DES EXTRAITS DE CODE

Extrait de code 1 : mise à jour des droits pour le checkout 35
Extrait de code 2 : Méta_inc.php, ajout d'un calendrier javascript.....47
Extrait de code 3 : html_api.php, fonction d'affichage de page mantis.....48
Extrait de code 4 : Custom_constant_inc.php, définition des nouveaux statuts de demande:.....49
Extrait de code 5 : custom_strings_inc.php, instanciation des valeurs de statut de demande.....50
Extrait de code 6 : Custom_strings_inc.php, instanciation des variables de présentation: 50
Extrait de code 7 : config_inc.php, configuration du workflow 50
Extrait de code 8 : core.php, chargement des fichiers personnalisés.....51
Extrait de code 9 : appel d'une extension éditeur de texte riche dans MediaWiki.....60
Extrait de code 10 : localsetting.php, configuration des accès aux SGBD 62
Extrait de code 11 : linker.php, modifiation de la fonction d'hyperlien d'affichage des images ...64
Extrait de code 12 : config_default_inc.php, déclaration de la méthode d'authentification 68
Extrait de code 13 : authentification_api.php, modification de la fonction d'authentification.....70
Extrait de code 14 : user_api.php, validation de la forme du login.....70
Extrait de code 15 : config_defaults_inc.php, configuration de l'accès LDAP pour Mantis 71
Extrait de code 16 : localsetting.php, configuration de l'accès à LDAP pour MediaWiki 72
Extrait de code 17 : mantisIntegration.php, fonction wfMantisExtension.....74
Extrait de code 18: mantisIntegration.php, fonction showMantis.....75
Extrait de code 19 : mantisIntegration.php, sélection du contenu d'une demande.....75
Extrait de code 20 : mantisIntegration.php, sélection simple de contenu de ticket.....77
Extrait de code 21 : mantisIntegration.php, sélection avancée de contenu dans les demandes 78

Index des tableaux

Tableau 1 : Les droits des utilisateurs dans Mantis 41
Tableau 2 : gestion des notifications dans Mantis 43
Tableau 3 : matrice des statuts d'une demande dans Mantis 48
Tableau 4 : liste des espaces de nom dans MediaWiki 54
Tableau 5 : structure de la table mantis_bug_table de Mantis 75

BIBLIOGRAPHIE

Ouvrages

Benoît C., Logiciels libres, 2005. Open source: qu'est-ce que c'est ?
Editions H&K.

Elie F., 2009. Économie du logiciel libre.
Éditions Eyrolles, Collection : Accès libre.

Yvon R., 2008. Le logiciel libre dans les PME.
Hermès Science Publications, Collection : Etudes logiciels informatiques

Lerdorf R., Tatroe K., MacIntyre P., 2006. *Programming PHP, Second Edition*
O'Reilly Media.

Barrett D. J., 2008. *MediaWiki, Wikipedia and Beyond*
O'Reilly Media.

Nawrocki C., 2007. Gestion des Incidents du Si - *Cadrage et Mise en œuvre Base Sur Itil V2/V3*
Éditions Eyrolles.

Sites internet

Marlalapocket, 2009, Personnalisation de Mantis BT.
http://www.commentcamarche.net/faq/16960-personnalisation-de-mantis-bt

Rus, 2005, Add support for using jscalendar.
http://www.mantisbt.org/bugs/view.php?id=5774

2010, MediaWiki pour débutants
http://fr.wikibooks.org/wiki/MediaWiki_pour_d%C3%A9butants/Configuration_de_votre_wiki

2010, FCKeditor integration guide
http://mediawiki.fckeditor.net/index.php/FCKeditor_integration_guide