Mise en place d'une méthodologie de tests documentée, accessible et pilotable, au sein de la Direction des systèmes d'information des Hôpitaux universitaires de Genève

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Henry SENG

Conseiller au travail de Bachelor :

Marianne BAYAT-RICARD

Responsable de stage :

Evrard de BONDY, Responsable domaine DSI HUG

Genève, 22.06.2012

Haute École de Gestion de Genève (HEG-GE)

Filière IG

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre d'informaticien de gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 20.06.2012

Henry Seng

Remerciements

Je remercie les HUG et plus particulièrement Monsieur Evrard de Bondy et l'équipe ERP Logistique qui m'ont offert l'opportunité de réaliser ce travail grâce à un stage intéressant au sein de la Direction des systèmes d'information. Je remercie également Madame Marianne Bayat-Ricard pour son encadrement et ses conseils qui ont permis la finalisation de ce document.

Merci à Astrid, Maria et Sylvie pour leur précieuse aide.

Résumé

Ce travail de Bachelor a été réalisé dans le cadre d'un stage au sein de la DSI des Hôpitaux universitaires de Genève sous la responsabilité de M. de Bondy et s'inscrit dans un contexte d'harmonisation et de suivi des processus de test qui font encore défaut.

Y seront notamment traités le contexte, les moyens et outils à disposition de la DSI, ce que représente le test et, enfin le développement d'une méthodologie adaptée aux besoins des HUG.

Il s'avère que la compréhension de la nécessité des tests et de leur encadrement offre un atout de taille dans la production de logiciels, tant pour le client que les utilisateurs finaux. Cela représente aussi des gains financiers et administratifs non négligeables.

C'est donc naturellement que des stratégies de tests et des équipes qui y sont dédiées s'imposent comme étant la norme.

Cependant, même si dans la théorie il est facile de justifier une stratégie de tests, les choses sont différentes dans la pratique : résistance aux nouvelles pratiques, délais trop courts et tests qui passent à la trappe, etc.

Un des défis majeurs des années à venir sera donc de convaincre les directions et les responsables de projet des bénéfices apportés par l'organisation des projets de test.

Table des matières

Déclaration i
Remerciements ii
Résumé iii
Table des matières iv
Liste des Figures vi
Liste des Tableaux vi
Introduction 1
1. Le contexte 2
1.1 Les Hôpitaux universitaires de Genève 2
1.2 La Direction des systèmes d'information 2
1.3 Qualiac, l'ERP des HUG 2
1.4 La gestion actuelle des tests 3
2. Le test 3
2.1 Ses enjeux 3
2.2 Le besoin 4
Pourquoi avons-nous besoin de tester ? 4
2.3 Les limites et idées reçues 4
2.4 Les obstacles 5
2.5 Le métier de testeur 5
2.5.1 Les qualités requises 5
2.5.2 Rôles et responsabilités 6
2.5.3 Certifications 6
2.6 Les différents types de tests 7
2.6.1 Techniques générales 7
2.6.2 Les tests fonctionnels 8
2.6.3 Les tests non fonctionnels 8
2.6.4 Les tests concernant les modifications 8
Stratégie de tests de non-régression 9
2.6.5 Les tests de recette 9
2.7 La conception de tests 9
2.7.1 La traçabilité 10
2.7.2 La réutilisabilité 10
2.8 Un processus outillé 10
2.8.1 Les avantages 11
2.8.2 Gestionnaire du référentiel de tests 11
2.8.3 Générateur de tests 11
2.8.4 Automate de tests 12
2.8.4.1 Objectifs et avantages de l'automatisation 12
2.8.4.2 Les limites 12
2.8.4.3 ROI 12
2.8.5 Gestionnaire des anomalies 12

Le cycle de vie d'une anomalie 13
2.9 La gestion du planning 13
2.9.1 L'organisation du projet de test et de sa documentation en bref 14
2.9.2 Quand mettre fin aux phases de tests ? 15
3. La solution adoptée pour la DSI 16
3.1 La maintenance de Qualiac 16
L'apport d'un référentiel de tests 16
3.2 Les outils de gestion des tests du marché 16
3.2.1 Les critères d'adoption 17
3.2.2 Simplicité d'utilisation 17
3.2.3 Les fonctionnalités 17
3.2.4 TestRail 18
3.2.5 TestLink 19
3.2.6 R.T.M.R 20
3.2.7 Tableaux récapitulatifs 21
3.2.7.1 TestRail 21
3.2.7.2 TestLink 21
3.2.7.3 R.T.M.R 21
3.3 L'outil retenu : R.T.M.R 22
3.3.1 L'installation 22
3.3.2 Sa prise en mains 22
3.3.3 Qui utilise l'outil 26
3.4 Les livrables 26
3.4.1 Le plan de test 26
3.4.2 Les fiches de test 27
3.4.3 Le rapport de test 28
4. La suite 28
4.1 Promotion des bonnes pratiques de gestion de projet 28
4.2 Étendre la méthodologie dans les autres groupes des HUG 29
4.3 Utilisation de la méthodologie dans l'avenir 29
Conclusion 30
Bibliographie 31
Webographie 31
Annexe 1 Marche à suivre d'un projet de test 32
Annexe 2 Plan de test 33
Annexe 3 Modèle de fiche de test 36
Annexe 4 Modèle de rapport de test 37

Liste des Figures

Figure 1 Étapes du projet de test....14
Figure 2 TestRail....18
Figure 3 TestLink....19
Figure 4 R.T.M.R....20
Figure 5 Connexion à R.T.M.R....22
Figure 6 Nouveau projet....23
Figure 7 Actions....23
Figure 8 Nouveau cas de test....24
Figure 9 Nouvelle campagne....25
Figure 10 Progression d'une campagne....25

Liste des Tableaux

Tableau 1 Critères et pondération....17
Tableau 2 Score TestRail....21
Tableau 3 Score TestLink....21
Tableau 4 Score R.T.M.R....21

Introduction

La mise en place d'un processus de tests est un travail qui touche actuellement tous les acteurs du monde des logiciels. Ou en tout cas qui le devrait. En effet, c'est une phase nécessaire au développement d'applications de qualité, robustes et stables. Des certifications internationales, encadrées par des experts de tous horizons, existent dont notamment l'ISTQB (International Software Testing Qualifications Board) et son représentant suisse, le STB (Swiss Testing Board). Ces comités offrent un catalogue de formations et d'examens mettant en avant les compétences clés du test. Ces qualifications permettent aux testeurs d'étoffer leur expertise et aux recruteurs de mieux cibler leurs recherches de candidats.

Afin d'améliorer le pilotage des projets et des évolutions, ce travail a été mandaté par la Direction des systèmes d'information des HUG. Il devra aboutir à la mise en place d'une méthodologie de tests facile à prendre en main, mettant en lumière des outils de monitoring et un référentiel de tests reproductibles qui sera employé pour les applications du domaine de la logistique et plus particulièrement Qualiac, un ERP dont nous reparlerons plus en détail.

Nous commencerons par présenter les HUG, la DSI et sa gestion actuelle des tests, puis nous aborderons le test de manière globale et détaillée. Finalement, nous présenterons la méthodologie et les outils retenus et mis en place ainsi que la suite du projet.

1. Le contexte

1.1 Les Hôpitaux universitaires de Genève

Les HUG sont un établissement public médical avec trois missions principales : soigner, enseigner et rechercher. En 2010, ils étaient le premier hôpital universitaire de Suisse avec plus de 8'000 employés et environ 530'000m2 de surface exploitable. D'abord hôpital cantonal en 1856, ce n'est qu'en 1995 que les différents hôpitaux de la région sont regroupés pour former les Hôpitaux universitaires de Genève. A noter que les HUG ont reçu un prix décerné par le HIMSS (Healthcare Information and Management Systems Society) en 2010 pour le niveau de développement de son SI et font partie du top 3 européen.

1.2 La Direction des systèmes d'information

La DSI a été créée en 2010 et a pour mission de garantir la sécurité du SI et la continuité du service. En outre, elle est responsable, de la conception jusqu'au maintien, des logiciels permettant de gérer les HUG. Elle est divisée en trois services qui sont le service développement applicatif qui s'occupe des besoins en solutions informatiques, le service production et infrastructure qui s'occupe de l'optimisation des processus de production, de la mise à disposition de l'infrastructure et de son support, et enfin le service support utilisateurs qui distribue le matériel informatique et s'occupe de son support.

1.3 Qualiac, l'ERP des HUG

Les HUG utilisent les solutions ERP de la société française Qualiac depuis 1998, avec au lancement la mise en place de la gestion financière et celle des fournisseurs. Il faudra uniquement cinq mois pour que la solution soit opérationnelle. En très peu de temps, Qualiac deviendra un outil stratégique au sein des HUG dans la maîtrise des coûts et des processus, notamment grâce à son évolutivité, sa simplicité et à l'attention que porte l'éditeur à ses clients afin de leur offrir un produit de qualité.

1.4 La gestion actuelle des tests

Chaque application est suivie par un à trois responsables qui sont les correspondants principaux en cas de problèmes. Les applications développées en externe (ce qui se fait majoritairement au sein de la DSI) sont testées sur des serveurs de tests et de formation avant leur mise en production. Ce sont les responsables qui s'occupent de tester leur application, sans forme prédéfinie, et qui rapportent les résultats au responsable domaine, M. de Bondy.

Aucune méthodologie particulière ni outils spécifiques aux tests n'ont été mis en place. Le manque de visibilité et l'impossibilité de chiffrer les projets font parties des raisons pour lesquelles la décision de mettre en place une méthodologie a été prise.

Le nouveau processus, lorsqu'il sera utilisé, permettra d'accélérer les phases de tests tout en en améliorant leur qualité et couverture fonctionnelle.

2. Le test

> *« Le test est une activité destinée à déterminer si l'évaluation d'une caractéristique ou d'une aptitude d'un programme ou d'un système donne les résultats requis. »* *(John Watkins, 2001 :11)*

2.1 Ses enjeux

Le premier défi de taille est la qualité, ou en d'autres termes, la satisfaction des clients. Alors que dans les débuts de l'informatique, peu d'importance lui était consacrée, dû notamment à une relation client peu privilégiée, c'est aujourd'hui un élément clé de la réussite d'un projet. Face à un taux élevé de projets en échec et la non-satisfaction des clients, c'est une prise de conscience globale qui s'est produite et qui nous a amenés au point où nous en sommes aujourd'hui, où il est avéré que l'investissement placé dans la qualité permet de réduire significativement les pertes que la non-qualité apporte.

Tester, c'est aussi garantir que le logiciel respecte les spécifications émises en début de projet. Ce sont la confiance du client et son argent qui sont mis en jeu. Un non-respect de ses exigences peut avoir de lourdes conséquences tant pour le client que pour le prestataire.

Enfin, des projets de très grandes envergures n'auraient jamais pu voir le jour sans l'aide de tests. Prenons l'exemple de l'envoi d'une navette dans l'espace où un échec

représente des pertes immenses. L'implication des tests est ici énorme, afin d'assurer au mieux la faisabilité du projet et sa sécurité.

2.2 Le besoin

Pourquoi avons-nous besoin de tester ?

Les logiciels deviennent de plus en plus complexes, ce qui les rend difficiles à développer, débugger et maintenir. De plus, avec des délais et des pressions budgétaires subies sur un marché très compétitif et en constante croissance, les moindres défaillances découvertes par les clients engendrent des pertes d'image, de confiance, de contrats et finalement d'argent.

Les problèmes ne s'arrêtent pas là. Les défauts d'un logiciel peuvent en outre causer de réels dommages. Prenons l'exemple du tableau de bord d'un avion qui tombe en panne, ou des données critiques d'un patient d'hôpital qui sont perdues, et nous avons là des vies humaines mises en danger. Tout cela implique que les logiciels soient testés correctement afin d'éviter des pertes et des dommages regrettables.

Dans le cas de la maintenance, qui nous concerne plus précisément dans ce travail, l'objectif des tests est de valider les nouvelles fonctionnalités ou les corrections, ainsi que de contrôler la non-régression (l'absence d'effet de bord sur les autres fonctionnalités).

2.3 Les limites et idées reçues

Il est important de considérer quelques points inhérents au test afin de rectifier quelques croyances.

Tout d'abord, il n'est pas possible de tester une application à 100%. En effet, les logiciels aujourd'hui sont si complexes que le nombre de combinaisons à tester est bien trop élevé. La bonne stratégie est de trouver le bon ratio entre le nombre de tests et la couverture fonctionnelle.

De la même manière, il n'est pas possible de démontrer l'absence de défauts, qui reviendrait encore une fois à tout tester de façon exhaustive.

Se limiter à tester à la fin du développement se révèle être coûteux. En effet, plus les défauts sont découverts tard, plus leur correction est onéreuse. C'est pourquoi il est plus rentable de « tester au plus tôt » dans le cycle de développement. Cela évite aussi de devoir rogner sur la qualité en cas de délais trop court.

L'absence de défauts n'est pas un gage de réussite. Un logiciel qui ne présente aucune anomalie mais qui ne correspond pas aux besoins des utilisateurs sera considéré comme un échec.

2.4 Les obstacles

Bien souvent encore, tester est une tâche attribuée directement à des développeurs qui la considère rébarbative (avec raison). En effet, c'est une activité qui, bien que nécessaire à la qualité du produit final, va remettre en question leur travail, et leur en apporter davantage. C'est aussi, surtout lors des tests de non-régression, un travail très répétitif qui, heureusement, peut être confié dans certains cas à des automates de tests (dont nous reparlerons plus loin).

C'est aussi un travail peu reproductible où il faut recommencer toute une démarche d'une application à l'autre. De même que le fait d'avoir plusieurs testeurs sur le même logiciel ne produisant pas forcément les mêmes tests pourrait créer des écarts dans les résultats et introduire des erreurs d'analyse.

Il est encore difficile à concevoir que les tests ne ralentissent pas le développement mais qu'ils permettent au contraire d'apporter un produit fini de qualité plus rapidement. Malheureusement, cette incompréhension fait que très souvent les tests sont laissés pour la fin du développement, et que les délais étant très limités, ils se font souvent évincer du cycle de développement.

2.5 Le métier de testeur

2.5.1 Les qualités requises

Tester n'est pas aussi évident qu'il y paraît. Cela requiert un sens avancé des détails, de la curiosité et un sens de la communication développé. Le testeur doit avoir une connaissance du métier et dans certains cas pouvoir faire l'interface entre les utilisateurs et les développeurs.

Il doit en outre posséder de bonnes compétences relationnelles afin de pouvoir coopérer avec les équipes de développement. En effet, il peut être vu par ces dernières comme étant un élément cherchant à détruire le travail qu'ils ont accompli, à tort. Ce ne sont pas les testeurs qui créent les défauts, mais les développeurs. Les testeurs eux sont là pour les identifier, afin de limiter les risques, ce qui bénéficie à tous.

Cette vision qu'ont les développeurs des testeurs doit être prise sérieusement en considération car, premièrement, les développeurs recevant des feedback négatifs des tests de leur application se sentent faillibles et, deuxièmement, cela entraine une surcharge de travail dédiée à la correction des erreurs. Il est donc très important de savoir communiquer afin de ne heurter personne et de pouvoir continuer les projets positivement.

2.5.2 Rôles et responsabilités

Nous avons trois rôles principaux : Responsable des tests, analyste de test et testeur.

La tâche du responsable des tests est d'organiser les tests en les planifiant, en les contrôlant et en établissant des rapports de tests. Il doit en outre effectuer la coordination avec le responsable du développement.

L'analyste testeur s'occupe de la conception des tests et des données de tests. Il doit bien connaître l'application à tester. C'est lui qui alimente le référentiel de tests.

Le testeur quant à lui va s'occuper de l'exécution et du rapport des résultats. Il est important qu'il puisse contribuer au plan de test afin d'être en accord avec celui-ci. Enfin, il devra utiliser les outils de tests à sa disposition pour permettre au responsable de faire son travail de suivi des tests.

2.5.3 Certifications

Actuellement, il n'existe pas ou peu de formations de testeur, ce qui rend le travail des recruteurs très difficile. En effet, il n'existe que très peu de mots-clés pour repérer les testeurs qualifiés. Quelques certifications américaines ont vu le jour ces dernières années, mais elles ne présentaient pas l'indépendance nécessaire à une reconnaissance internationale.

Ce n'est qu'en 2002 que des experts de plusieurs pays se sont réunis pour former l'ISTQB (International Software Testing Qualifications Board), un comité proposant une certification internationale sur des critères unanimement reconnus. Les formations sont indépendantes des évaluations et sont basées sur un syllabus disponible sur leur site internet. Des centres de formations indépendants peuvent être accrédités par l'ISTQB ou un comité national tel que le STB (Swiss Testing Board) et dispenser ses propres formations.

La certification comprend trois niveaux :

1. **Fondation :** ce niveau couvre les fondamentaux du test, le cycle de vie du logiciel, les différentes techniques, la conception de tests, la gestion et les outils.

2. **Avancé :** ce niveau étend celui de Fondation, avec l'amélioration continue et l'impact des applications critiques notamment.

3. **Expert :** ce niveau étend celui d'Avancé, avec l'amélioration et la gestion du processus de test, l'automatisation et la sécurité.

2.6 Les différents types de tests

De nombreuses techniques de test existent, mais ce n'est pas le but de ce travail de toutes les détailler. Nous allons donc faire ici un survol de ce qui est important à retenir.

2.6.1 Techniques générales

Parmi les techniques générales, nous trouvons le test par affirmation ou négation, le premier servant à confirmer le respect des exigences et le second permettant d'observer le comportement de l'application hors du champ des spécifications. Le test par négation n'a virtuellement pas de limites (comprenez par là qu'il est possible de tester ce qui ne correspond pas à des spécifications d'une multitude de manières différentes) et qu'il faut donc utiliser cette technique avec parcimonie.

Nous avons également à notre disposition le test de boîte blanche ou boîte noire, l'un se faisant en connaissance des spécifications internes relative à l'application tester, avec des tests produits en conséquence, et l'autre ne s'appuyant sur aucune information concernant le mode de développement du logiciel et de son code. Le test de boîte blanche se fait généralement par le développeur très tôt car c'est lui qui possède la meilleure connaissance structurelle de l'application, alors que le test de boîte noire se fait plus tard, par les équipes de tests sans forcément avoir les mêmes connaissances internes du logiciel.

2.6.2 Les tests fonctionnels

Ces tests couvrent les fonctionnalités de l'application définies dans les exigences et spécifications, et se situent au cœur de la qualification logicielle. Leur objectif est de garantir l'absence d'anomalie dans les fonctions réalisées afin d'avoir une application opérationnelle en production.

2.6.3 Les tests non fonctionnels

Cette catégorie de tests comprend des questions de performance, de fiabilité, d'utilisabilité, de maintenabilité et de portabilité. Malgré l'importance de ces différents aspects, ces tests sont souvent effectués après les tests fonctionnels ou tout simplement négligés. Cela est aussi souvent dû à des lacunes au niveau des exigences, et ce n'est que tard dans le cycle de développement qu'on se rend compte de certains problèmes.

Ces négligences ne sont pas sans conséquences. En effet, un problème non fonctionnel découvert très tard peut remettre en question l'architecture même de l'application et pénaliser tout le projet.

2.6.4 Les tests concernant les modifications

Ces tests concernent les mises à jour et les corrections de défauts. Ils se composent de :

- **Tests de confirmation** qui vérifient que le défaut a bien été corrigé.
- **Tests de non-régression** qui assurent que le reste de l'application n'a pas subi d'impacts négatifs.

Alors que pour les tests de confirmation, il suffit de refaire les tests qui ont permis de découvrir les défaillances, les tests de non-régression consistent à réexécuter tous les tests pour s'assurer de l'absence de nouvelles anomalies.

Ces derniers tests ont donc un impact très élevé au niveau de la charge de travail des testeurs et représentent l'une des principales raisons pour lesquelles ce travail a été effectué. Ils sont en effet fortement liés à la maintenance des applications.

Stratégie de tests de non-régression

Il existe divers manières d'appréhender les tests de non-régression. Nous en abordons ici trois :

1. Réexécuter tous les tests fonctionnels (requiert beaucoup de temps, mais permet de couvrir toutes les fonctions et d'atténuer les risques le plus possible)
2. Exécuter un sous-ensemble des tests fonctionnels des cas les plus utilisés ou les plus risqués (plus rapide mais une plus faible couverture des fonctions, c'est un compromis entre charge de travail et exhaustivité)
3. Exécuter un sous-ensemble des tests des composants aillant eu le plus grand nombre de défauts (couvre les cas les plus critiques)

Il est évident qu'il faudra faire un choix entre coût modéré et couverture optimale des défauts. Ce choix devra être pondéré avec l'expérience des testeurs, des responsables de l'application, des développeurs et l'historique de l'application. Une application développée avec une architecture fiable et robuste, qui n'aura pas présenté de problème durant sa vie en production, ne sera pas testée de la même manière qu'une autre application avec encore en phase de maturité ou présentant des erreurs récurrentes.

2.6.5 Les tests de recette

Le test de recette permet de s'assurer que l'application ou la version livrée répond conformément aux exigences fonctionnelles et non fonctionnelles avant sa mise en production, et s'effectue avec les représentants des utilisateurs.

Encore une fois, ces tests sont particulièrement importants dans le cadre de ce projet. En effet, les nouvelles versions de Qualiac y sont systématiquement soumises.

2.7 La conception de tests

Afin de pouvoir concevoir des tests, il est important de bien connaître l'application à tester ainsi que le domaine traité. En effet, il faudra décrire des scénarios de tests avec la bonne couverture des fonctionnalités à traiter et les faire de manière à ce qu'ils puissent être exécutés par n'importe qui. Tout le processus doit être correctement documenté pour que les testeurs aient les informations nécessaires à la compréhension des tests à exécuter.

2.7.1 La traçabilité

Piloter les phases de tests requiert les tableaux de bord adéquats, créés à partir d'informations pertinentes. Ce sont ces indicateurs qui vont nous permettre de répondre à des questions telles que « Où en sommes-nous dans les tests ? » et « Quand allons-nous terminer ? ».

Une bonne pratique est de partir d'une liste des exigences auxquelles l'application doit répondre. Une fois cette liste établie, nous pouvons concevoir des cas de tests couvrant toutes ces exigences. Finalement, nous sommes en mesure de lancer les exécutions des tests (ou campagne de tests) conçus plus tôt et d'en tirer des résultats.

Si nous voulons pouvoir tracer et suivre la progression des tests, nous allons avoir besoin d'outils. Ceux-ci, en plus de nous permettre de piloter les tests, vont aussi nous aider à améliorer le processus de test, notamment grâce au référentiel de tests.

2.7.2 La réutilisabilité

Pouvoir réutiliser les tests déjà effectués est un élément important dans la réussite des tests de non-régression. Ce sont ces mêmes tests qui ont permis de valider l'application auparavant qui vont à présent nous aider à contrôler l'absence d'effets de bord lors des mises à jour et corrections diverses de l'application.

De plus, la réutilisabilité est aussi un gain de temps, généralement gaspillé à réécrire les mêmes tests.

Nous allons avoir besoin d'un outil, nommé gestionnaire de référentiel de tests qui va nous permettre de gérer :

- Les exigences et cas de test
- Les campagnes de test et les résultats

2.8 Un processus outillé

Nous allons commencer par une mise en garde : les outils eux-mêmes ne représentent pas une méthodologie mais c'est leur utilisation dans une procédure structurée qui l'est. De plus, leur utilisation va nécessiter l'apprentissage de compétences spécifiques, ce qui représente un coût non négligeable. Il est donc important de bien évaluer les besoins avant l'achat. Nous rappelons que ces outils doivent apporter une plus-value au processus de test, en augmentant son efficacité et en réduisant ses coûts.

Pour ce travail, l'outil qui nous intéresse particulièrement est le gestionnaire du référentiel de tests. Cependant, nous allons aussi nous pencher sur le générateur de tests, l'automate de tests et le gestionnaire des anomalies.

2.8.1 Les avantages

Utiliser des outils permet une meilleure reproductibilité des tests ainsi que de mieux en mesurer la qualité et le coût. En outre, ils offrent une structure sur laquelle nous pouvons nous appuyer afin de réaliser un travail efficace.

2.8.2 Gestionnaire du référentiel de tests

Le gestionnaire du référentiel de tests est un outil central qui va regrouper les éléments du projet de tests. Nous l'utilisons pour :

- la gestion des projets de tests
- la base de données de tests
- la gestion des campagnes de tests
- la gestion des résultats

Parmi ses nombreux avantages, nous pouvons noter la base de données de tests. En effet, elle va s'avérer très utile lors des phases de tests de non-régression, puisque nous pouvons réutiliser des tests qui ont déjà été définis et catégorisés auparavant. De plus, il serait possible, avec les résultats acquis lors de précédentes campagnes de tests, d'estimer plus précisément les charges d'une campagne supplémentaire.

Notons également le suivi des exécutions de tests, élément important pour le responsable de projet.

Utiliser un tel outil est une vraie démarche de progrès, en comparaison de l'utilisation de Word ou Excel pour les mêmes tâches. Ces derniers ont les gros défauts de ne pas être dynamiques et très peu collaboratifs. Cela rend tout le processus très difficile à maintenir, chronophage et difficilement pilotable.

2.8.3 Générateur de tests

Le rôle du générateur de tests est de créer des tests automatiquement sur la base d'un modèle représentant les fonctionnalités de l'application. Ses atouts principaux sont de créer rapidement des cas de tests couvrant les fonctionnalités voulues, tout en déchargeant l'analyste de test de ce travail, et la possibilité de les réadapter sans trop

d'efforts en cas de modification des exigences. En effet, si le modèle évolue, les tests en seront impactés et mis à jour.

2.8.4 Automate de tests

L'automate de tests peut apporter un grand bénéfice lorsqu'il est correctement utilisé. Il soulage le travail redondant des testeurs et leur permet de se concentrer sur des tâches plus importantes.

Les scripts de test peuvent être enregistrés (actions faite une première fois à la main) et recopiés avec différentes valeurs de test. Une autre solution est de séparer les données des actions, ce qui offre plus de souplesse dans la maintenance des scripts et des jeux de tests à réaliser.

2.8.4.1 Objectifs et avantages de l'automatisation

Elle offre la possibilité de réaliser des tests de non-régression ou de fonctionnalités, 24h sur 24 sans discontinuer, très rapidement et de manière répétée. C'est un avantage indéniable pour une application à tester mature et disposant d'un nombre de fonctionnalités important.

2.8.4.2 Les limites

Malheureusement, l'automatisation des tests est un travail difficile et n'est pas toujours adapté à la situation. En outre, elle requiert d'une application de la maturité et une interface utilisateur qui ne changera pas ou peu. En effet, chaque modification peut entraîner la réécriture des scripts de test.

2.8.4.3 ROI

Il est primordial de bien réfléchir avant d'automatiser ses tests afin de ne pas se retrouver dans une situation où ce travail apporte plus de charges que de bénéfices. Maîtriser et maintenir l'automatisation n'est pas évidente mais l'enjeu est de taille.

2.8.5 Gestionnaire des anomalies

Lors de leur exécution, rarement tous les tests vont passer. Au contraire, des défauts vont souvent être détectés. Il va alors nous falloir les rapporter dans un gestionnaire d'anomalies pour ensuite pouvoir les traiter. Nous avons donc trois étapes :

1. Détection des anomalies
2. Leur gestion

3. Leur traitement

Le cycle de vie d'une anomalie

Après avoir été détectée, une anomalie doit être détaillée et qualifiée. Nous devons savoir quelles fonctionnalités elle remet en cause et juger de sa criticité et de sa priorité.

Après avoir été rapportée, l'anomalie doit être traitée. Cela comprend son étude, à savoir si elle va être corrigée ou non, sa correction et de nouveaux tests. On peut la clore ou la rouvrir selon les résultats.

2.9 La gestion du planning

Planifier des tests est un travail très difficile car les charges de travail peuvent varier grandement et souvent. De plus, le fait de tester et de détecter des défaillances va encore augmenter ces charges de travail. C'est donc avec précaution qu'il va falloir répartir les tâches et c'est avec réactivité qu'il va falloir réadapter le planning quand cela sera nécessaire.

Plusieurs facteurs entrent en compte dans l'estimation du travail. Ce sont notamment des facteurs liés :

- **à la politique d'entreprise.** La place et l'importance accordées aux tests par l'organisation ont un impact significatif sur leurs résultats. Elle va permettre ou non d'impliquer les tests dès les débuts d'un projet de développement et ainsi d'avoir une meilleure évaluation de leur charge.
- **au matériel.** Un environnement de test représentatif de la production et les outils adéquats sont des facteurs permettant d'optimiser le processus de test.
- **aux capacités des testeurs.** Un testeur confirmé va tester les éléments à risque avec plus de pertinence et assurer ainsi une meilleure qualité qu'un testeur débutant.
- **au contexte.** En effet, l'application à tester peut être plus ou moins complexe et avoir beaucoup ou peu de défauts. De plus, avoir des équipes dispersées géographiquement ne facilite pas le processus et rend les échanges plus longs et leur compréhension plus difficile (il est plus facile d'expliquer et de reproduire un problème face à face qu'à distance).

- **aux techniques d'estimation.** Il y a deux grandes familles de méthodes d'estimation. Celles basées sur les métriques et celles basées sur l'expérience. Les premières prennent en considération le nombre de ligne de code ou la charge de développement. Les deuxièmes quant à elles sont basées sur le vécu des testeurs ou sur l'historique d'anciens projets similaires. Chacune des deux ont leurs avantages et leurs défauts dont il faut en tirer le maximum de profits en piochant dans le meilleur des deux.

2.9.1 L'organisation du projet de test et de sa documentation en bref

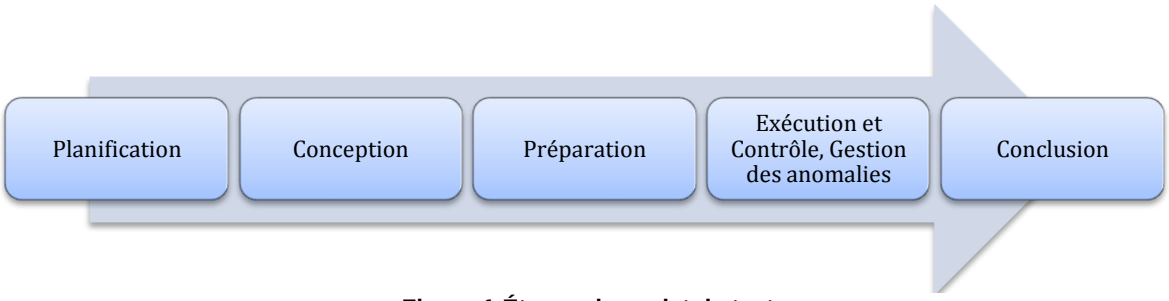

Figure 1 Étapes du projet de test

Sur la figure 1, nous pouvons observer les différentes étapes d'un projet de test. Chaque étape est accompagnée d'un certain nombre de documents et de tâches à réaliser :

- **Planification :** Estimer les charges de travail, les ressources nécessaires, préparer un plan de test.
- **Conception :** Identifier plus précisément les aspects à tester dans le plan de test. Lister également les tests à réaliser.
- **Préparation :** Écrire les cas de tests et les procédures de tests. Préparer l'environnement de test et les testeurs.
- **Exécution :** Exécuter les tests, faire remplir les fiches de test par les testeurs et faire insérer les résultats dans l'outil de gestion des tests par le responsable du projet.
- **Conclusion :** Rédiger un rapport des tests contenant les activités, les résultats et leur analyse.

2.9.2 Quand mettre fin aux phases de tests ?

Il nous faut choisir des critères qui vont nous permettre d'achever les phases de tests. Ceux-ci ne doivent ni être trop strictes afin de ne pas bloquer tout le projet, ni être trop faibles.

Un exemple de bon critère serait la couverture d'un pourcentage des fonctionnalités sans anomalies. Ce critère assure la qualité de l'application car il permet de valider ses spécifications.

Un mauvais critère serait la fin du temps imparti pour les activités de tests. Celui-ci n'assure pas la qualité de l'application car il peut être atteint sans qu'aucun test ne soit effectué.

Il très important, pour la suite du projet, d'achever les tests grâce à un bon choix des critères afin de ne pas se retrouver avec une application qui traîne ses défauts de phase en phase. En effet, cela va perturber l'efficacité globale du projet.

3. La solution adoptée pour la DSI

L'objectif est non seulement d'améliorer la qualité des applications mais aussi de rendre les projets de tests pilotables et chiffrables. Pour ce faire, la bonne utilisation des bons outils en combinaison avec la volonté d'améliorer le processus de tests de la part de la direction et des responsables de projets vont être essentielles à l'atteinte de ces objectifs.

3.1 La maintenance de Qualiac

Les tests vont concerner la maintenance et les évolutions de l'ERP Qualiac tout particulièrement. C'est un produit utilisé depuis déjà plus de 15 ans au sein des HUG et qui a subi de fortes évolutions tout au long de sa vie. La dernière en date est le passage en RIA (Rich Internet Application) qui offre une nouvelle interface améliorant le travail des utilisateurs et la possibilité d'utiliser l'ERP au travers d'un navigateur web.

Dans ce travail, ce sont surtout les tests fonctionnels et de non-régression qui nous intéressent.

L'apport d'un référentiel de tests

Jusqu'à à maintenant, des fiches de tests ont été préparées pour des campagnes de tests spécifiques et n'ont pas ou peu été réutilisées. Ce travail doit être refait à chaque mise à jour et requiert chaque fois de bonnes connaissances de l'application.

A présent, le but est de créer une base de données de tests une unique fois, qui soit détaillée et réutilisable sur plusieurs campagnes de tests sur plusieurs versions, par des testeurs n'ayant pas forcément des connaissances approfondies de l'application à tester.

3.2 Les outils de gestion des tests du marché

Il existe beaucoup de logiciels de gestion des tests, tous se ressemblant plus ou moins, avec des prix allant de 0.- à plusieurs milliers de francs. Pour ce travail, nous avons sélectionné et évalué trois outils pour finalement n'en retenir qu'un seul. Ce sont :

- **TestRail :** Solution web php, simple d'utilisation et payante.
- **TestLink :** Solution web php open source, relativement compliquée d'utilisation et gratuite.

- **R.T.M.R :** Solution client lourd avec base de données serveur, open source, relativement simple d'utilisation et gratuite.

3.2.1 Les critères d'adoption

Afin de sélectionner l'outil adéquat, nous avons préparé avec M. de Bondy une grille pondérée de critères.

Critères	Valeurs possibles	Pondération
Simplicité	**0 à 2 (0 = difficile, 2 = simple)**	**3**
Gestion des projets	**0,1**	**2**
Référentiel des cas de tests	**0,1**	**2**
Gestion des campagnes de tests	**0,1**	**2**
Prix	0 à 2 (0 = cher, 2 = gratuit)	1
Open source	0,1	1
Impression des fiches de tests	0,1	1
Gestion intégrée des anomalies	0,1	1

Tableau 1 Critères et pondération

L'outil doit posséder deux qualités essentielles :

1. **Être simple d'utilisation**
2. **Posséder les fonctionnalités dont nous avons besoin**

Une troisième qualité, moins importante mais dont nous tenons compte aussi, est le type de licence et son prix.

3.2.2 Simplicité d'utilisation

Nous entendons par là que le logiciel doit être facile à prendre en mains immédiatement et clair, c'est-à-dire que les éléments dans les différents écrans doivent être logiquement disposés et visibles.

3.2.3 Les fonctionnalités

L'outil devra au minimum posséder les fonctionnalités suivantes :

- **Gestion des projets de tests**
- **Référentiel des cas de tests**
- **Gestion des campagnes de tests**

3.2.4 TestRail

TestRail est une application web écrite en php et fonctionnant par défaut avec une base de données MySql. Elle offre une interface épurée, simple à utiliser et à prendre en mains, et elle possède les fonctionnalités dont nous avons besoin.

Cependant, elle n'est pas gratuite, son coût par utilisateur étant d'environ 200.- par année.

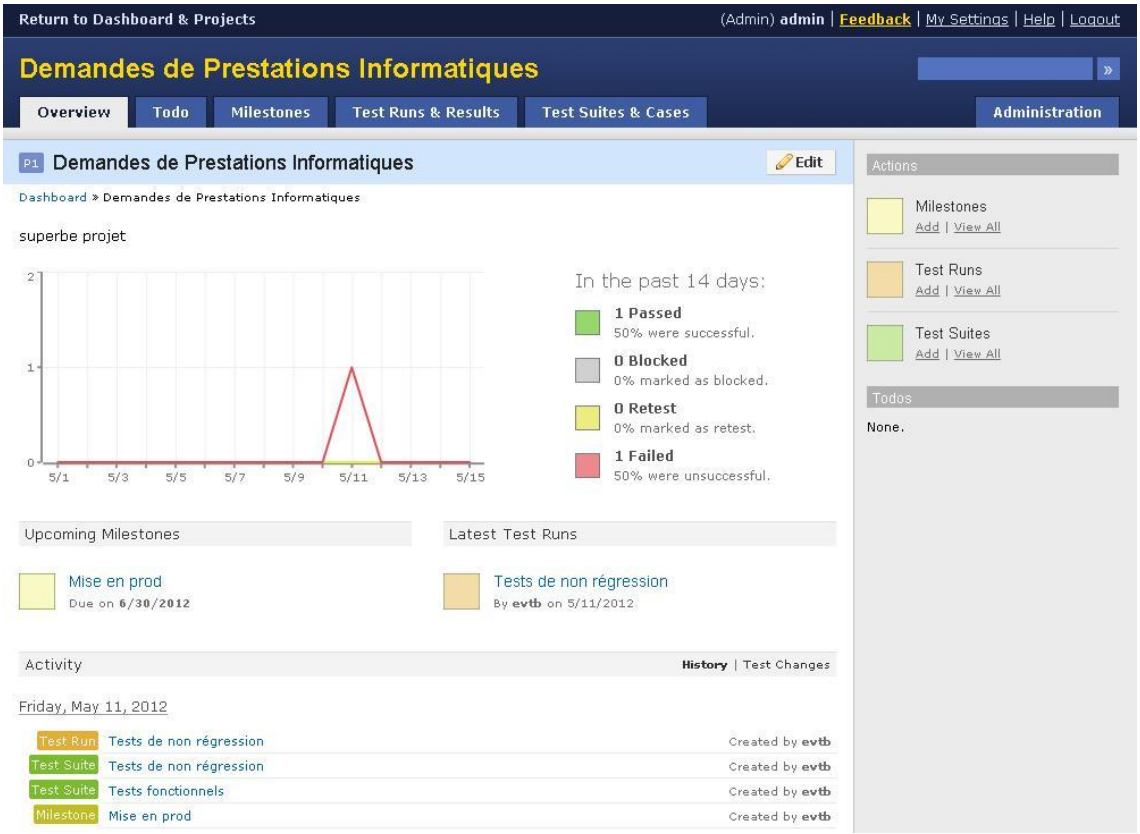

Figure 2 TestRail

3.2.5 TestLink

TestLink est aussi une application web écrite en php, mais n'est malheureusement pas aussi intuitive que TestRail. En effet, l'interface est austère et difficile d'accès. Lors de nos évaluations, nous n'avons pas réussi à lancer des campagnes de tests, bien que cela soit possible selon les spécifications de l'outil.

Contrairement à TestRail, elle offre la possibilité de spécifier des exigences qu'on peut lier à des cas de tests.

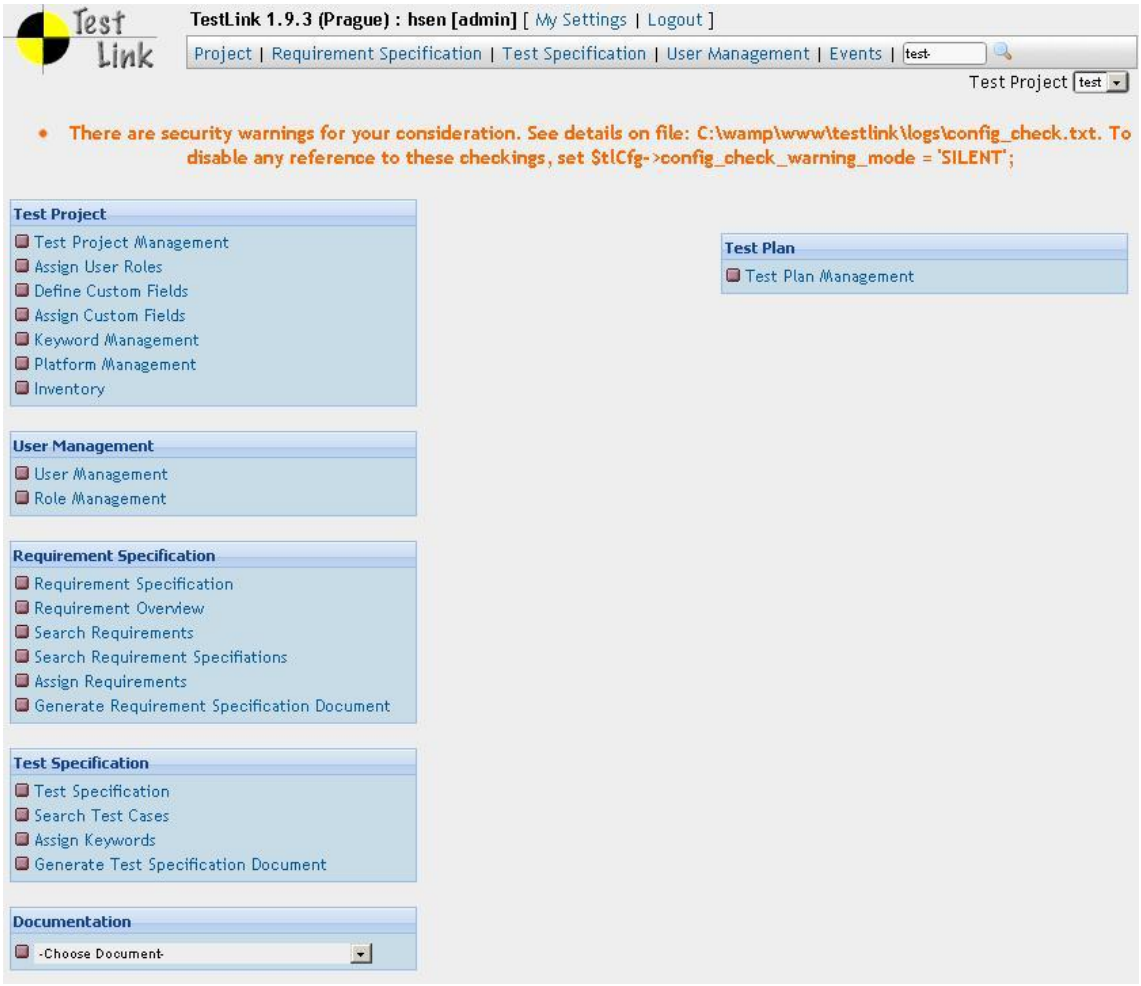

Figure 3 TestLink

3.2.6 R.T.M.R

R.T.M.R est une application client-serveur open source. À l'instar de TestRail, elle offre une interface simple à utiliser et, comme TestLink, permet de spécifier des exigences. De plus, elle possède un gestionnaire d'anomalies intégré.

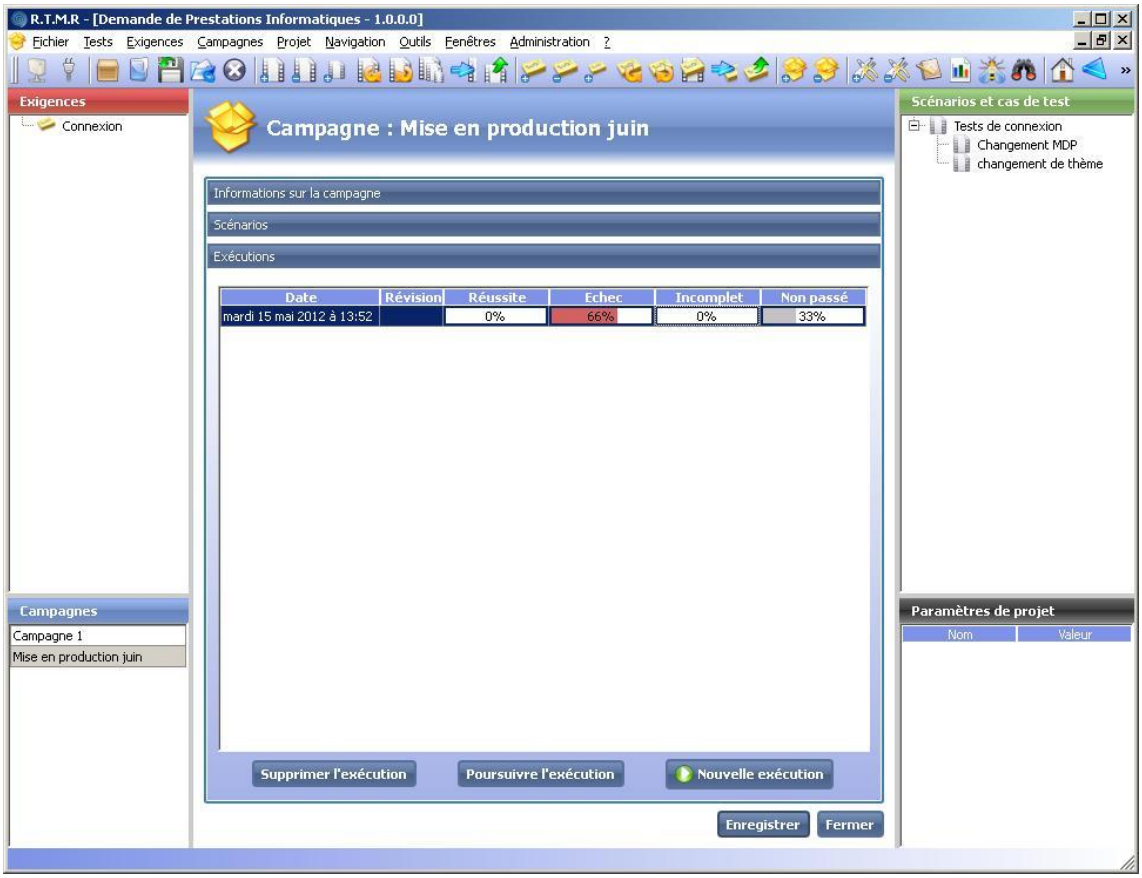

Figure 4 R.T.M.R

3.2.7 Tableaux récapitulatifs

3.2.7.1 TestRail

Critères	Score	Pondération	Total
Simplicité	**1**	**3**	**3**
Gestion des projets	**1**	**2**	**2**
Référentiel des cas de tests	**1**	**2**	**2**
Gestion des campagnes de tests	**1**	**2**	**2**
Prix	0	1	0
Open source	0	1	0
Impression des fiches de tests	1	1	1
Gestion intégrée des anomalies	0	1	0
Total			**10**

Tableau 2 Score TestRail

3.2.7.2 TestLink

Critères	Score	Pondération	Total
Simplicité	**0**	**3**	**0**
Gestion des projets	**1**	**2**	**2**
Référentiel des cas de tests	**1**	**2**	**2**
Gestion des campagnes de tests	**0**	**2**	**0**
Prix	2	1	2
Open source	1	1	1
Impression des fiches de tests	1	1	1
Gestion intégrée des anomalies	0	1	0
Total			**8**

Tableau 3 Score TestLink

3.2.7.3 R.T.M.R

Critères	Score	Pondération	Total
Simplicité	**2**	**3**	**6**
Gestion des projets	**1**	**2**	**2**
Référentiel des cas de tests	**1**	**2**	**2**
Gestion des campagnes de tests	**1**	**2**	**2**
Prix	2	1	2
Open source	1	1	1
Impression des fiches de tests	0	1	0
Gestion intégrée des anomalies	1	1	1
Total			**16**

Tableau 4 Score R.T.M.R

3.3 *L'outil retenu : R.T.M.R*

Nous avons décidé de retenir R.T.M.R car c'est l'outil qui correspond le plus à nos besoins et a l'avantage d'être gratuit.

Malgré ses bons atouts, TestRail n'a pas été retenu à cause d'un prix de licence trop onéreux. Quant à TestLink, c'est sa difficulté d'utilisation et l'impossibilité de gérer des campagnes qui l'ont écarté.

3.3.1 L'installation

R.T.M.R peut être installé sur un serveur et utilise le système de gestion de base de données PostgreSQL. L'accès peut ensuite facilement se faire depuis un client lourd sur des postes ayant accès au serveur.

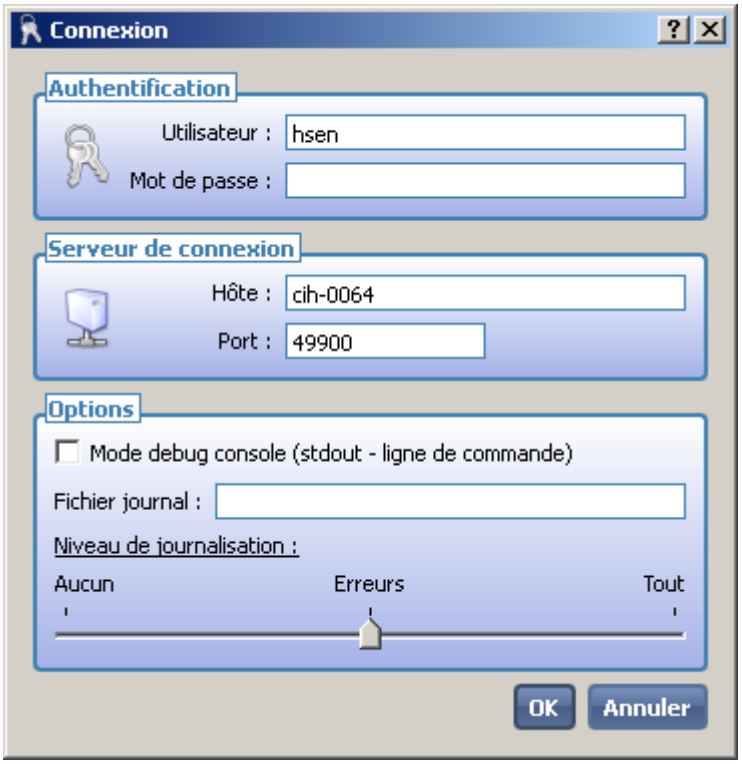

Figure 5 Connexion à R.T.M.R

3.3.2 Sa prise en mains

Une fois connecté, nous pouvons sélectionner un projet existant ou en créer un nouveau depuis le menu **Fichier → Nouveau projet…**

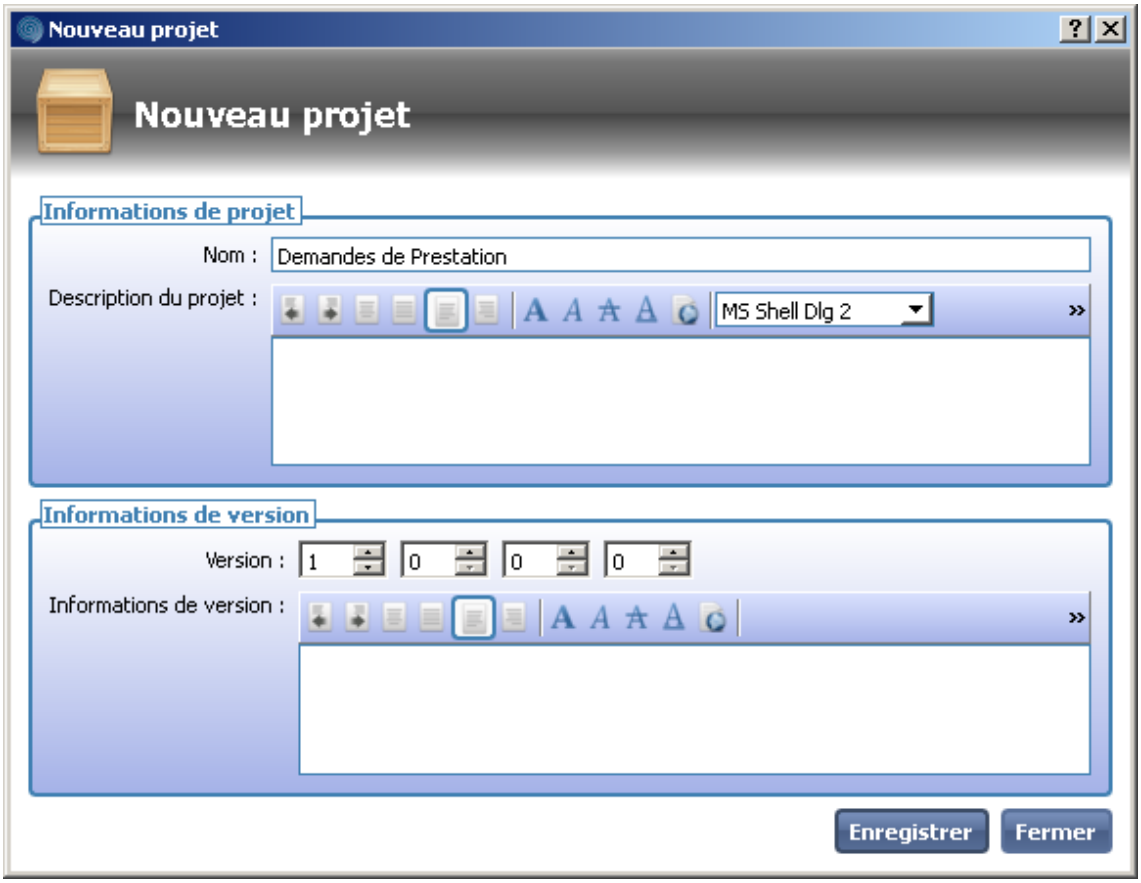

Figure 6 Nouveau projet

L'interface (voir la figure 4) se présente simplement, avec les exigences et campagnes de tests à gauche, et les scénarios et cas de tests à droite. L'espace du milieu permet de visualiser et de modifier les informations des différents éléments cités plus tôt.

Le menu **Tests → Nouveau test…** permet de créer un nouveau cas de test.

Un cas de test contient des actions à exécuter, qui peuvent être définies depuis l'onglet **Actions, exigences, pièces jointes et anomalies** du cas de test.

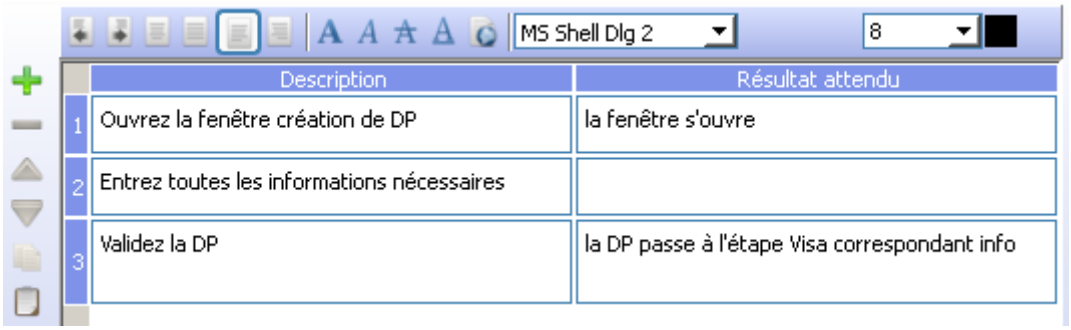

Figure 7 Actions

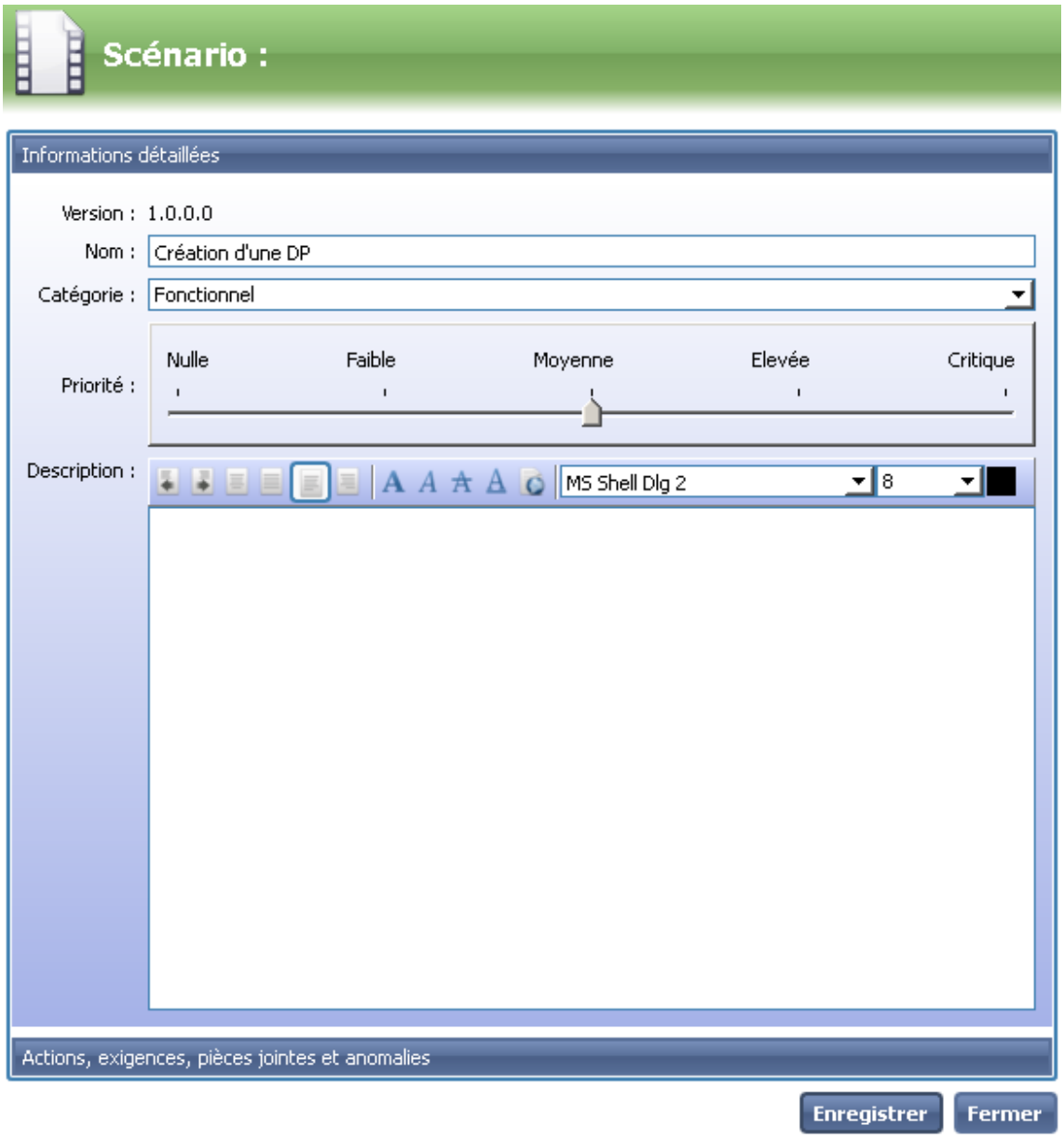

Figure 8 Nouveau cas de test

Après avoir défini les tests, nous pouvons créer une campagne de tests à partir du menu **Campagnes → Nouvelle campagne…**

La première fenêtre nous permet d'entrer un nom et une description, la seconde nous permet de sélectionner les tests à exécuter pour cette campagne.

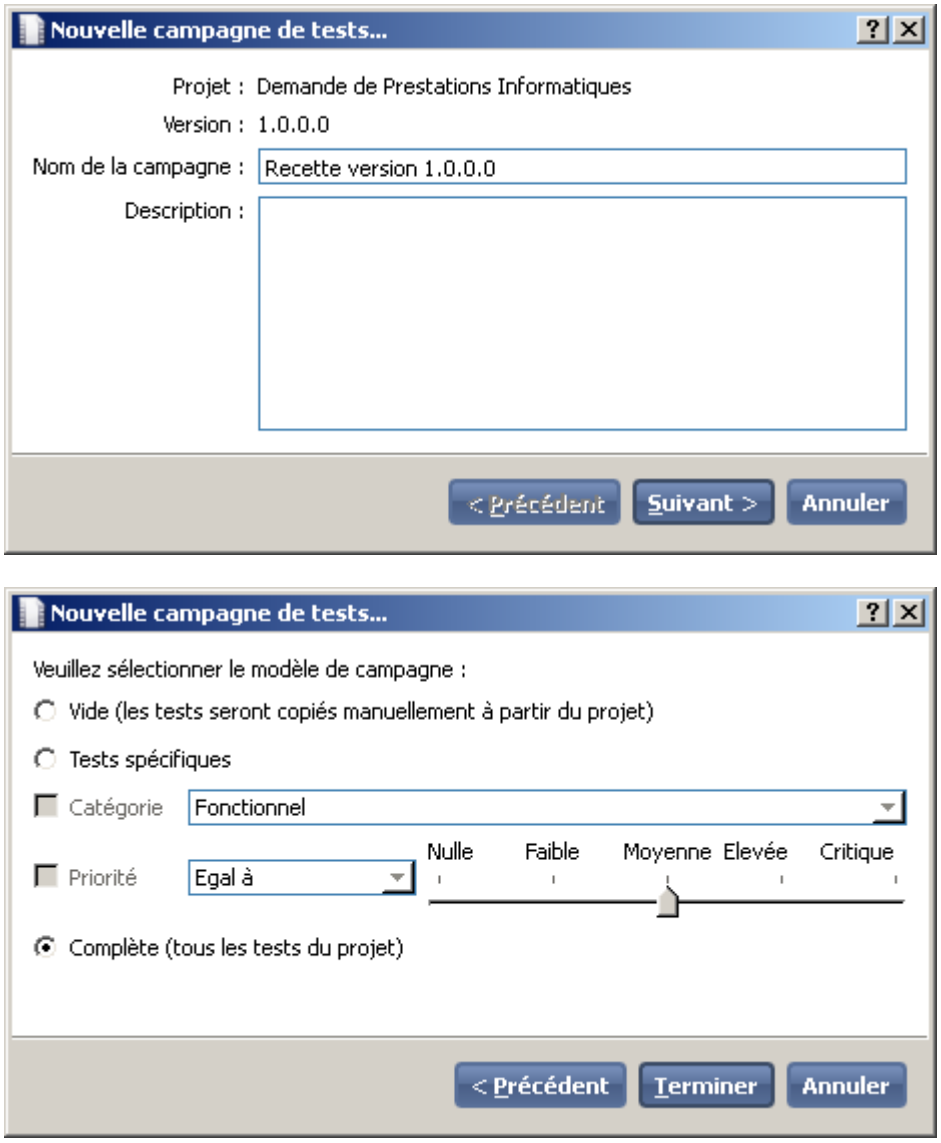

Figure 9 Nouvelle campagne

Nous avons enfin tous les éléments pour mener à bien une campagne de tests.

Lors de l'exécution d'une campagne, les tests peuvent être OK, KO ou remis à plus tard. Nous pouvons générer une anomalie pour un test qui échoue (KO) qui pourra être tracée sur les campagnes de tests des versions ultérieures.

Il est aussi possible de suivre l'avancement d'une campagne grâce à un tableau de bord.

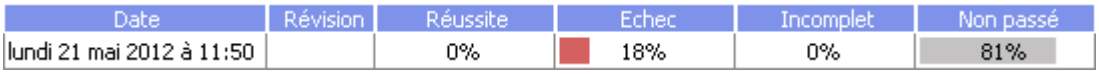

Figure 10 Progression d'une campagne

Enfin, des rapports des résultats de campagnes peuvent être générés au format pdf et html.

3.3.3 Qui utilise l'outil

Pour des raisons de praticité, seuls les responsables de projet utiliseront l'outil de gestion des tests. Les testeurs rempliront des fiches de tests imprimées au préalable qui seront remises aux responsables.

Dans la phase de recette, ce sont des supers utilisateurs qui apportent leur soutien en jouant le rôle de testeur, puis une fois que les plus gros défauts ont été corrigés, ce sont les correspondants informatiques qui testent à leur tour l'application.

3.4 Les livrables[1]

Nous avons un outil de tests, mais à lui tout seul il n'est pas suffisant pour mener à bien nos projets de tests. Nous devons également produire un certain nombre de documents.

Ce sont dans notre situation :

- **Le plan de test :** Ce document contient une présentation du contexte, de l'application à tester et des objectifs à atteindre. Nous y développons l'organisation des tests et les critères de fin des tests.
- **Des fiches de test avec résultat :** Ces fiches doivent être identifiées et datées. Elles contiennent une référence au projet de test, la version de l'application, la description du test, les étapes à réaliser et les résultats attendus et effectifs. Ces fiches sont distribuées aux testeurs qui les remplissent, et consignées sur notre outil de tests.
- **Le rapport de test :** C'est une synthèse des tests. Nous y inscrivons les résultats suivis d'une analyse, avec un rappel du contexte en introduction.

L'objectif est de rester simple, concis et efficace.

3.4.1 Le plan de test

Élément clé du projet de test, nous le décomposons de la manière suivante:

1. **Contexte :** Présentation du projet, de l'application à tester, de l'environnement de test et de l'outil de gestion des tests.

---

1 Vous trouverez les modèles du plan de test, des fiches de test et du rapport de test dans les annexes.

2. **Intervenants :** Présentation des participants et de leur rôle.
3. **Objectifs :** Présentation des objectifs du projet de tests, de la portée des tests et des cas à exclure.
4. **Risques :** Il s'agit d'identifier les risques durant la phase des tests et de trouver des mesures pour les éviter ou les réduire. Un exemple de risque serait que le responsable des tests tombe malade, ou que la matériel sur lequel les tests doivent être effectués ne soit pas livré à temps.
5. **Liste des tests :** Nous allons lister les tests à réaliser et leur couverture fonctionnelle.
6. **Organisation des tests :** Il va falloir planifier les tests et avoir des testeurs disponibles. En outre, il faudra aussi mettre en place l'environnement de test.
7. **Critères de fin des tests :** Ce dernier point traite des critères qui permettent de mettre fin aux tests.

Un modèle se trouve à l'annexe 2.

3.4.2 Les fiches de test

Ces fiches seront distribuées aux testeurs et contiendront les étapes à réaliser pour les différents tests. Elles sont décomposées de la manière suivante :

1. **Identifiant du projet, du test, sa date de réalisation, le testeur et la criticité, la base utilisée et la version de l'application :** Nécessaire pour s'y retrouver et insérer les résultats dans l'outil de test.
2. **Objectif :** L'objectif du test en quelques mots. Ce qui est testé.
3. **Pré-requis :** État nécessaire à la réalisation du test.
4. **Étapes à réaliser :** Il faut être clair et précis quant aux actions à effectuer, et ajouter des captures d'écran si nécessaire.
5. **Résultat escompté :** Si tout se passe bien, le résultat qui doit ressortir.
6. **Résultat obtenu :** Le résultat observé. Il faut préciser si le test a échoué ou non.

Un modèle se trouve à l'annexe 3.

3.4.3 Le rapport de test

Lorsque les tests sont achevés (ou interrompus), il faut rédiger un rapport contenant les informations suivantes :

1. **Identifiant du projet :** Nécessaire pour faire le lien avec le projet et son plan de test.
2. **Rappel des objectifs et résumé des activités :** Brève remise en contexte du projet et le déroulement des activités de test en quelques mots.
3. **Liste des tests effectivement réalisés et leur résultat (OK ou KO)**
4. **Analyse des résultats :** Il faut se concentrer sur les éléments relevés négatifs, ce qui ne va pas et pourquoi, et enfin quels en sont les impacts.
5. **Synthèse :** Contient les éléments à corriger, l'appréciation globale de l'application testée et celle du déroulement des tests.

Un modèle se trouve à l'annexe 4.

4. La suite

4.1 Promotion des bonnes pratiques de gestion de projet

Nous abordons ce sujet à cause d'éléments absents sur certains projets qui, lorsqu'ils sont présents, permettent de mettre en place des projets de tests cohérents.

Actuellement, c'est la méthode HERMES qui devrait être utilisée au sein des HUG, mais qui ne l'est pas toujours. Cette méthode, utilisée par la confédération suisse, découpe un projet en six phases qui sont initialisation, analyse préliminaire, conception, réalisation, introduction et finalisation. En outre, elle permet d'améliorer globalement le déroulement d'un projet informatique.

Malheureusement, dans la réalité, le manque de temps et les budgets restreints font que des projets sont lancés rapidement et des étapes sont brûlées. Cependant, ces projets aboutissent relativement souvent malgré tout et cela grâce à l'expertise des différents intervenants. Les problèmes surviennent lorsque des applications n'ont pas été explicitement et exhaustivement spécifiées, et que les phases de tests sont rallongées ou perturbées à cause de cela.

Les conséquences, humaines et financières, sont du temps perdu et du stress dû aux délais de mise en production.

4.2 Étendre la méthodologie dans les autres groupes des HUG

Après avoir mis en place l'outil et la méthodologie, l'objectif est d'en faire une évaluation dans un premier temps. Dans un second temps, et si les résultats sont satisfaisants, il faudra faire approuver et valider la méthodologie par la direction. Par la suite, ce sera le groupe intégration des HUG qui sera responsable de la mettre à la disposition de tous et de sensibiliser les collaborateurs concernés.

4.3 Utilisation de la méthodologie dans l'avenir

Malheureusement, actuellement rien ne peut assurer que la méthodologie sera réellement mise en pratique dans le futur. Beaucoup d'éléments font obstacle et il est toujours difficile de faire accepter de nouvelles pratiques par tous.

Heureusement, l'homogénéisation des procédures de tests est approuvée et fortement soutenue par le responsable domaine de la DSI. Cependant, cela sera-t-il suffisant ? Quel travail reste-t-il à faire pour que tout le monde l'adopte ?

Il reste encore bien d'autres questions telles que celles-ci qui doivent être posées, et des solutions qui doivent être recherchées, ceci dans un but qu'il ne faut pas perdre de vue et qui est d'offrir des logiciels de haute qualité, fiables et utilisables.

Conclusion

Mettre en place une méthodologie de test est un challenge difficile à réaliser. Cela requiert de l'ouverture d'esprit de la part des responsables de projet, qui doivent s'habituer à utiliser de nouveaux outils dont ils ne voient pas directement le bénéfice, et de la hiérarchie qui doit l'approuver aussi bien par la parole que par les actes, ne serait-ce qu'en débloquant les ressources et les délais nécessaires à la réalisation de tels projets. Cependant, lorsque la méthodologie est utilisée, et que le travail de test est bien fait, les bénéfices apportés deviennent indéniables.

Ce fût pour moi une expérience très enrichissante, tant au niveau humain que métier. Les contacts et dialogues, nécessaires à la réalisation de ce travail, furent nombreux et intéressants. Ainsi, j'ai eu la chance de pouvoir être confronté au terrain, à la réticence voire la résistance de certains, aux difficultés que cela amène et comment les outrepasser. Bien qu'une méthodologie ait été mise en place, le travail n'est pas encore fini et je suis impatient de pouvoir le poursuivre dans un processus d'amélioration continue afin de l'adapter au mieux aux besoins des HUG.

Ayant travaillé jusqu'à présent uniquement dans le développement, ce stage et ce travail de Bachelor auront été l'opportunité de voir le métier d'informaticien de gestion sous un tout autre angle. J'envisage maintenant bien différemment mon avenir dans ce domaine et je remercie les HUG et la HEG de m'avoir offert cette chance.

Bibliographie

B. Legeard, F. Bouquet, N. Pickeart**,** *Industrialiser le test fonctionnel pour maitriser les risques métier et accroitre l'efficacité du test 2e édition*, Paris, DUNOD, 2009, 2011.

J. Watkins, *Test logiciel en pratique*, Paris, Vuibert, 2002.

B. Homès, *Les tests logiciels fondamentaux*, Paris, Lavoisier, 2011.

Webographie

http://www.istqb.org/ consulté durant le mois de mai 2012

http://www.software-tester.ch/ consulté durant le mois de mai 2012

Annexe 1
Marche à suivre d’un projet de test

Les étapes à suivre d'un projet de test

Avant de débuter les tests

1. Etablir le plan de test
2. Préparer les tests dans R.T.M.R
3. Préparer les fiches de test
4. Prévenir les différents intervenants
5. Préparer l’environnement de test
6. Réserver les locaux pour les tests, si nécessaire

Pendant les tests

1. Réunir les testeurs
2. Exécuter les tests, si nécessaire sous l’encadrement du responsable des tests
3. Remplir les fiches de test

Après les tests

1. Reporter les résultats dans R.T.M.R
2. Rédiger le rapport de test
3. Transmettre le rapport pour la correction de l'application, si nécessaire

Annexe 2
Plan de test

Plan de Test		
HUG Hôpitaux Universitaires de Genève	**Projet :**	
	Date :	**Auteur :**

Contexte	
Application à tester :	**Version :**
Environnement de test :	
Date de début des tests :	**Date de fin des tests :**
Identifiant RTMR :	

Intervenants	Rôles

Objectifs, portée des tests, cas à exclure :

Risques :

Critères de fin des tests :

Liste des tests à réaliser	
Identification	**Description**

Organisation et planification des tests
Lieux des tests :
Ressources matérielles et humaines nécessaires :
Description de l'organisation des tests :

Annexe 3
Modèle de fiche de test

Fiche Test		
HUG Hôpitaux Universitaires de Genève	**Projet :**	
	Version :	**Base :**

X-XXX-01

Objectif du test :

Pré-requis :

Etapes	Résultats attendus	Résultats obtenus

Description des problèmes:

Test réalisé par	Statut du test		Test réalisé le	Criticité
	OK	KO		

Annexe 4
Modèle de rapport de test

Rapport de Test		
HUG Hôpitaux Universitaires de Genève	**Projet :**	
	Date:	**Auteur:**

Rappel des objectifs :
Résumé des activités de test :

Liste des tests réalisés		
Identification	**Description**	**Résultat**

Analyse des résultats :

Synthèse :