Rapport des améliorations du GTDClientKrom

Caillaud Anthony, Fortun Manoël, Garnier Alexandre

9 avril 2010

Table des matières

1 Introduction

Dans le cadre du module de vérification et tests, nous avions dans ce projet pour but de tester et améliorer une application - GTDClientKrom - codée par des étudiants en master 2 au premier semestre.

Pour ce faire, nous devions suivre un cahier des charges mis en place par nos collègues : Julien Durillon, Peter Moueza et Coralie Poisson. Afin d'améliorer la qualité du projet, nous avons repris le rapport fourni afin de connaître les points à améliorer et son utilisation.

2 L'amélioration du GTDClientKrom

Les différents points cités dans le rapport d'évaluation ont été repris. Les améliorations effectuées sont présentées ci-dessous.

2.0.1 Documentation

L'utilisateur peut trouver la documentation expliquant la méthode GTD à l'adresse suivante : http://fr.wikipedia.org/wiki/Getting_Things_Done

2.0.2 Implémentation

Pour tester le degré d'abstraction du package fr.alma.gtd.util, nous avons utilisé JDepend.

Concernant le degré d'abstraction de ce package, nous l'avons séparé en deux packages afin d'avoir un package contenant l'interface et l'autre contenant son implémentation. Ceci nous a permis d'obtenir un degré d'abstraction de chaque package respectivement de 1 et de 0.

De plus, toutes les erreurs générées par PMD ont été corrigées.

2.0.3 Tests

Les couvertures de tests avec Cobertura n'ont pas pu être effectuées car celui-ci s'utilise uniquement avec Ant. Le projet étant entièrement développé avec Maven, et par manque de temps, nous n'avons pas pu utiliser Cobertura.

Les classes que nous avons testées sont ChoseAFaireDao.java, ContactDao.java, ContexteDao.java et ProjetDao.java même si cela n'a pas beaucoup d'intéret car ces classes sont en lien avec hibernate.

En ce qui concerne les tests de mutation, avec Jumble ou autre, nous n'avons pu les mener à bien, principalement en raison du fait que la plupart des classes à tester sont des Javabean ou des classes d'IHM, ou encore des classes liées à la gestion de la persistence d'objets dans une base de données (Hibernate).

2.0.4 Efficacité

Même en modifiant le fichier Log4J, cela n'a pas abouti. De fait nous ne pouvons réellement proposer une amélioration de l'efficacité de l'application.

2.0.5 Maniabilité

Le point à améliorer demandant d'enlever de l'interface les opérations non implémentées ne nous semble pas adapté car il va à l'encontre de l'extensibilité du projet.

Nous n'avons pas eu le temps de régler les autres demandes concernant l'interface.

2.0.6 Sécurité

La longueur des mots de passe doit maintenant être au minimum de six caractères. La façon dont l'application gère cette nouvelle fonctionnalité vient s'ajouter à la confirmation du mot de passe.

2.0.7 Confort

La vérification de l'automatisation du bouton d'actualisation n'a pas été faite, par manque de temps.

3 Conclusion

Au travers du déploiement de tests et de l'amélioration du code, nous avons eu à faire des choix ; parfois contraints (l'abandon de Cobertura par exemple), parfois parce qu'ils nous apparaissaient les plus justes. Faire des tests sur plusieurs classes du projet serait incohérent car beaucoup d'entre-elles sont soit des JavaBean soit des classes liées à hibernate.

En guise de conclusion, nous pensons être parvenus à proposer une batterie de tests ainsi qu'un code plus clair, toujours dans l'optique de respecter les objectifs vu au cours de ce module.