HAI913I
Evolution et restructuration des logiciels

TP1 - Exercice 2

Auteur :

Canta Thomas (21607288)

Master 2 - Génie Logiciel
Faculté des sciences de Montpellier
Année universitaire 2021/2022

Exercice 2 - Quelques facettes de la maintenance/évolution logicielle

Reverse Engeenering

En utilisant des outils d'analyses tels qu'un désassembleur ou un décompileur nous pouvons retrouver les patrons de conceptions d'un programme et ainsi reformer le diagramme UML associé à notre application. Cette méthode ne peut pas être totalement automatisée car le code assembleur créé est souvent imparfait et nécessite l'intervention d'un développeur pour améliorer ou corriger le code. Côté base de données, il est possible de reconstituer des modèles de données en utilisant des structures physiques comme des fichiers ou tables.

Réingénierie

Il faut décortiquer notre application actuelle pour en comprendre son fonctionnement (soit par la documentation ou encore par rétro-ingénierie), par la suite on peut établir un plan d'action, notamment en déterminant les potentielles évolutions (mises à niveau, nouvelles fonctionnalités...) ou même la conception de l'application au format mobile (UML, méthode et logiciel de développement, adaptation à la nouvelle plateforme...). Une fois les coûts et la méthodologie obtenus on peut commencer à migrer notre application.

Qualité et refactoring

Pour évaluer la qualité de la précédente application on peut effectuer un benchmark avec notre application actuelle. Par exemple on peut utiliser des méthodes automatisées comme PerfDiff [1] ou encore Google Lighthouse [2] pour les sites webs. Autrement, il est possible de comparer point par point chaque élément cité ci-dessous et juger les points qui ont évolué ou au contraire régressé.

Pour pouvoir estimer la qualité d'une application on peut utiliser les métriques suivantes :

- Performance
- Sécurité
- Fiabilité
- Documentation
- Maintenabilité
- Taille

On peut améliorer notre application en corrigeant les problèmes des points cités précédemment, mais pour ce faire notre application devras parfois se refaire une beauté et migrer vers une nouvelle technologie, souvent plus moderne et plus efficace.

Compréhension

Pour une meilleur compréhension du programme il faut se renseigner auprès de la documentation si disponible. Sinon une analyse dynamique du programme s'avérer être nécessaire, accompagné d'une analyse statique et d'un "code review".

1. Source : http://web.cs.ucla.edu/ tianyi.zhang/perfdiff.pdf
2. Source : https://developers.google.com/

Localisation des features et traçabilité

Il est possible de localiser le code associé à une feature à l'aide des "pull request" présente sur git, si celui-ci est bien formé, sinon il faut opter pour de la compréhension de logiciel.