Université De Montpellier
Faculté Des Sciences

Niveau : Master 2

Module : Évolution et restructuration des logiciels

HAI913I

Rapport 2 du TD N°1 : Généralités

Supervisé par :
M. Abdelhak-Djamel Seriai
M. Marianne Huchard

Réalisé par :
Yanis Allouch

2021/2022

Table des matières

1 Quelques facettes de la maintenance/évolution logicielle

1.1 Reverse engineering

Pour pouvoir construire un modèle structurel (ici un diagramme de classes) d'une application en reverse engineering, il est possible d'effectuer une analyse statique ou bien dynamique. L'analyse statique peut se décomposer en une analyse syntaxique ou sémantique.

Dans le cas où nous souhaiterions construire un modèle dynamique (ici un graphe d'appels) de l'application. Il est nécessaire d'effectuer une analyse dynamique qui peut se dérouler (en fonction de l'objectif de l'analyse) en boite noire ou bien en boite blanche[1].

Enfin, dans les deux cas, nous pouvons observer les données générées et véhiculées par l'application pour complémenter nos analyses.

1.2 Réingénierie

Pour migrer une application ou partie d'une application d'une plateforme à une autre, j'imagine que j'établirai un modèle A (source) de l'application et un modèle B (cible) et d'utiliser les techniques vues et développées en ingénierie des modèles permettant de faire des transformations de modèles.

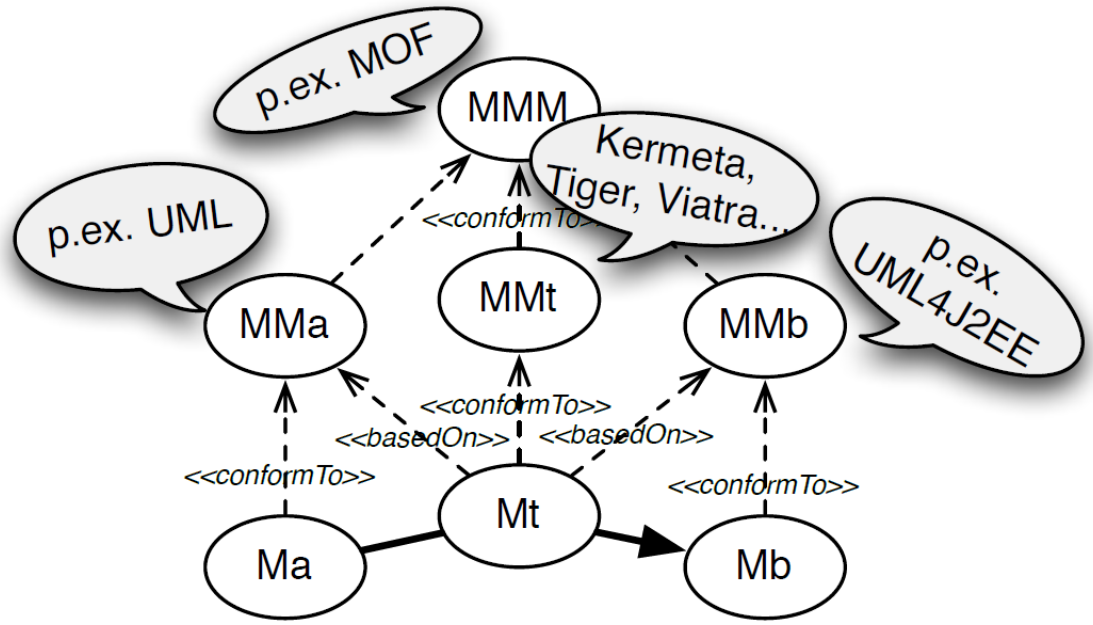

FIGURE 1 – Principes de la transformation de modèle [2]

1.3 Qualité et refactoring

1. Comment faites-vous pour évaluer la qualité de l'application précédente ?

   Dans un premier temps, je prends connaissance du «README.md».
   Puis un parcours du dépôt superficiel permet de se familiariser avec la structure apparente dans l'arborescence (au sens des fichiers) et des branches.
   Je vérifie les statistiques à ma disposition, comme le nombre de contributeurs (157) et le nombre d'étoiles (1.4K) du projet sur GitHub. Le nombre de fork du projet (778) est aussi un bon indicateur.
   Enfin, je vérifie si la structure syntaxique du versionning est standard (bonne pratique) et le nombre de release (282).

   — Présence de script facilitant le déploiement, l'exécution et la maintenance.

2. Donnez quelques métriques pour pouvoir estimer la qualité de cette application ?
   — Utilisation de logiciel d'évaluation du code (SonarCube, etc.) qui vérifie des métriques reconnues (bad-smells, tests, etc.),
   — Bugs, Securité, Documentation coverage,
   — Le code m'est compréhensible,
   — Communauté open-source active,
   — Dette technique de l'application mesurée par des outils spécialisés,

3. Que faut-il faire pour améliorer la qualité de cette application si elle n'est pas satisfaisante ? Donnez quelques exemples.
   — Un bon début serait de documenter l'API et de réduire le coût de déploiement, d'utilisation et de maintenance de l'application. Cela permet d'engager plus aisément les bons développeurs. Il est préférable d'engager la discussion/critique autour de code-review et des MR.

1.4 Compréhension

Nous pouvons identifier toutes les fonctionnalités offertes par cette application au travers du «README.md» ainsi que de la documentation technique offrant des exemples de codes avec des inputs et des outputs.

1.5 Localisation des features et traçabilité

J'utiliserai la liste des branches crée par feature pour directement isoler les parties du code introduit dans l'application en lien avec la feature recherché.

Références bibliographiques

[1] URL : https://web.archive.org/web/20210917112122/https://quentin-dupont.com/wp-content/uploads/2018/03/Outils-et-techniques-utilis%C3%A9s-dans-le-Reverse-Engineering.pdf.

[2] Jean-Marc JÉZÉQUEL, Benoit COMBEMALE et Didier VOJTISEK. *Ingénierie Dirigée par les Modèles : des concepts à la pratique...* Ellipses, 2012.