Université De Montpellier
Faculté Des Sciences

Niveau : Master 2

Module : IA pour le Génie Logiciel

HAI916I

Résumer d'un article scientifique traitant de l'IA pour le Génie Logiciel

---

Supervisé par :
M. Marianne Huchard

Réalisé par :
Kaci Ahmed
Yanis Allouch

2021/2022

Table des matières

Dans ce rapport, nous traitons l'article scientifique intitulé "***Empirical Review of Java Program Repair Tools***" [2] [1]

Introduction

La résolution manuelle des bugs logiciels est une des tâches les plus difficiles qui consomme énormément de temps. Pour faire face à cette difficulté une nouvelle branche de recherche intitulée « **automatic program repair** » a émergé, introduisant des approches de correction automatique de bugs.

Afin d'élaborer ce résumé, nous nous sommes basées sur trois questions proposées en TD qui nous ont permis de le structurer avec les trois sections suivantes :

- section 1 : Qu'ont-ils cherché à faire ?
- section 2 : Qu'ont-ils utilisé ?
- section 3 : Qu'ont-ils produit comme résultat ?

1. PDF Moodle n°**1905.11973.pdf**

1 Qu'ont-ils cherché à faire ?

Actuellement, les outils de ce domaine de recherche (automatic program repair) se basent sur des évaluations empiriques ne détectant pas tous les types de bugs. Ainsi, les objectifs de cet article sont d'une part, de vérifier expérimentalement si les outils de réparation de bugs existants ont le même comportement lorsqu'on les expérimente sur différents benchmarks de tests. D'autre part, la compréhension de la cause de la non-détection et réparations de bugs non connus par la communauté de réparation.

2 Qu'ont-ils utilisé ?

Dans cet article, l'auteur présente un examen de la littérature sur les évaluations existante des outils de réparation de code qui consistent, d'une part, en le rassemblement de ces outils. D'autre part, en la collecte des résultats obtenus en se basant sur le benchmark utilisé et le nombre de bugs fournit en entrée.

La conception de cette étude se base sur la vérification des performances des outils de réparation de bugs lors de leur utilisation sur des benchmarks différents de Defects4J [2]. Et ce, en détaillant les questions de recherche sur lesquelles est fondé cette étude, la sélection systématique des outils de réparation et des benchmarks de bugs, ainsi que la collecte et l'analyse des données.

Par ailleurs, ils ont développé un framework nommé ***RepairThemAll***[3], permettant d'abstraire l'exécution et l'analyse des outils et benchmarks et en facilitant l'extension et l'adaptabilité d'outils et de benchmarks.
Le framework RepairThemAll est composé de trois principaux composants :

1. Plug-in d'outil de réparation, où se trouve l'abstraction autour des outils de réparation, permettant l'ajout et le retrait d'outils,

2. Plug-in de benchmark, où il y a l'abstraction autour des benchmarks, permettant également l'ajout et la suppression de benchmark,

3. Runner de réparation, qui fonctionne comme une façade pour l'exécution de l'outil de réparation sur des bugs spécifiques.

2. René Just, Darioush Jalali, and Michael D. Ernst. 2014. Defects4J : A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs. In Proceedings of the 23rd International Symposium on Software Testing and Analysis (ISSTA '14). ACM, New York, NY, USA, 437–440.

3. Thomas Durieux, Fernanda Madeiral, Matias Martinez, and Rui Abreu. 2019. The RepairThemAll framework repository. https://github.com/program-repair/RepairThemAll

3 Qu'ont-ils produit comme résultat ?

Pour répondre aux questions de recherches, le framework RepairThemAll a permis l'obtention d'un ensemble de patchs utilisés pour l'analyse. Et cela grâce au lancement des outils de réparation pendant des durées limitées et sur des graines prédéfinies aléatoirement du fait qu'une grande puissance de calcul est requise, dû à la grande échelle d'exécution de ces outils et à la recherche des causes de la non-génération des patchs.

Les résultats statistiques obtenus concernant la réparabilité des outils de réparations de bugs sélectionnés ont permis la déduction des réponses aux questions de recherche. Ainsi, en ce qui concerne la génération de patchs, pour les bugs émanant de divers benchmarks, par les outils de réparation basés sur des suites de tests, ces derniers sont complémentaires les uns avec les autres. En effet, chaque outil détecte un ensemble de bugs unique et la réparabilité des outils dépend de leurs approches respectives ainsi que du framework de génération de patchs où ils sont mis en œuvre.

Concernant, la similarité de la réparabilité sur les différents benchmarks, il a été conclu que la réparabilité des outils diffèrent lorsqu'elle est appliquée des benchmarks distincts.

Enfin, les causes de la non-génération des patchs peuvent se résumer au fait que l'outil de réparation ne peut pas résoudre certains bugs, la localisation incorrecte des bugs, la répartition des bugs sur plusieurs emplacements, l'insuffisance du temps des tests, la configuration incorrecte, etc.

Références bibliographiques

[1] Valentin Amrhein, Sander Greenland et Blake McShane. *Scientists rise up against statistical significance*. 2019.

[2] Thomas Durieux et al. « Empirical review of Java program repair tools : A large-scale experiment on 2,141 bugs and 23,551 repair attempts ». In : *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2019, p. 302-313.

[3] Ronald L Wasserstein, Allen L Schirm et Nicole A Lazar. *Moving to a world beyond "p< 0.05"*. 2019.