N° d'ordre NNT : 2022LYSE2035

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de

L'UNIVERSITÉ LUMIÈRE LYON 2

École Doctorale : ED 512 Informatique et Mathématiques

Discipline : Informatique

Soutenue publiquement le 6 juillet 2022, par :

Abir M'BAYA

A graph based end to end defect prediction framework.

Devant le jury composé de :

Hervé PANETTO, Professeur des universités, Université de Lorraine, Président

Teresa GONÇALVES, Professeure des universités, Université d'Evora, Rapporteure

Virginie GOEPP, Maîtresse de conférences HDR, INSA Strasbourg, Rapporteure

Sebti FOUFOU, Professeur des universités, Université de Bourgogne, Examinateur

Nejib MOALLA, Professeur des universités, Université Lumière Lyon 2, Directeur de thèse

Contrat de diffusion

Résumé

Avec une taille et une complexité sans cesse croissante, les logiciels modernes sont de plus en plus sujets aux défaillances lors des différentes montées de version, avec en moyenne 15 à 50 bugs qui sont comptabilisés toutes les 1000 lignes de code. C'est pourquoi, chaque année, les entreprises dépensent au total 312 milliards de dollars pour déboguer leurs logiciels, dont 40 à 50% d'entre eux sont tout de même mis en production avec des défauts. Avec des délais de développement relativement cours, les développeurs aussi bien que les testeurs n'ont tout simplement pas suffisamment de temps pour effectuer des tests complets et efficaces, ce qui se traduit par une détection des anomalies dans une phase tardive du cycle d'ingénierie logicielle. Or justement, une détection plus en amont dans le cycle en V permettrait de diminuer considérablement le nombre de bugs et maximiser par conséquent la qualité du logiciel ainsi que la satisfaction client. En effet, une détection rapide des bugs permettrait de réduire le cout de 200%, tandis qu'une détection dès la phase de conception coûte 100 fois moins cher qu'une détection après la livraison du produit.

C'est pourquoi, pour minimiser les coûts et maximiser la qualité, les chercheurs ont proposés de nombreuses techniques de prédiction des défauts logiciels permettant d'analyser le code source de la prochaine version à livrer. Ces modèles de prédiction sont typiquement construits à partir des données d'historique du logiciel et leur maitrise permet d'offrir une prédiction efficace des défauts logiciels en identifiant les éléments « à risque », pouvant être potentiellement bogués. Une importante étude intitulée Perceptions, Expectations, and Challenges in Defect Prediction, réalisée en 2018 auprès de 395 développeurs, testeurs et chefs de projets de 33 nationalités différentes, a d'ailleurs permis de montrer que 92% des professionnels questionnés envisagent d'utiliser ce genre de techniques, s'ils en avaient la possibilité, ce qui prouve l'énorme demande autour de ce besoin. Mais avant d'espérer un déploiement global et généralisé, il faudrait parvenir à maitriser un haut niveau de précision dans la prédiction et une meilleure prise en compte des typologies de bugs possibles.

Depuis des décennies que le sujet de prédiction des défauts est d'actualité, de nombreux chercheurs ont d'abord proposés de sélectionner manuellement des métriques (McCabe, CK etc.) pour caractériser et évaluer les zones de code source à prédire comme étant sujettes à des défaillances. Il en existe d'ailleurs aujourd'hui une vingtaine qui sont régulièrement utilisées,

que la communauté scientifique qualifie de « traditionnelles ». Puis avec l'arrivée du Machine Learning et du Deep Learning dans le domaine de la prédiction des défauts, les valeurs d'entrées fournies au classifieur ne concernent plus quelques métriques calculées mais représentent l'ensemble des éléments structurels des classes du code source qui sont automatiquement générées. En proposant une représentation de la classe sous forme d'arbre syntaxique abstrait (AST) puis sous forme de graphe avec le Control Flow Graph (CFG), les chercheurs ont permis de prendre en compte les informations structurelles et sémantiques ainsi que les processus d'exécution des programmes et donc de couvrir plus de typologies de bugs. Il en a découlé des améliorations successives de la précision de la prédiction.

Les modèles existants n'offrent pas encore une performance suffisante et une couverture de tous les types de bugs permettant de répondre aux évolutions constantes des programmes logiciels en termes de complexité. Malgré que le CFG ait donné de meilleurs résultats par rapport aux techniques basées sur l'AST et les métriques, il ne permet que de capturer le processus d'exécution au sein de la classe et non pas les informations et les relations entre les classes, autrement dit il ne permet pas de capturer le comportement du programme. Or de nombreux bugs sont directement liés aux relations entre les classes, les méthodes et les dépendances des composants logiciels entre eux en règle générale. Les problèmes de dépendances circulaires apparaissent par exemple au niveau des dépendances entre classes ou entres méthodes. De nombreux chercheurs ont déduit de leurs travaux qu'il y a une plus grande probabilité d'apparition de bugs avec une complexité du logiciel croissante, elle-même liée aux dépendances de ses composants.

Partant de ce constat, nous proposons une nouvelle technique de prédiction des défauts logiciels basée sur le Deep Learning, qui améliore les approches existantes, imprécises puisqu'elles ne permettent pas d'extraire toutes les propriétés sémantiques des programmes. Notre technique de prédiction plus complète, se basant sur Code Property Graph (CPG) permet de détecter plus de typologies de bugs et en particulier celles associées aux dépendances. Cela a permis d'atténuer la fragilité des méthodes d'apprentissage actuelles et d'améliorer la précision des techniques de prédiction grâce à la prise en compte des liens de dépendance entre les composants logiciels.

Pour construire un modèle de prédiction efficace, il est nécessaire d'alimenter le processus d'apprentissage avec suffisamment de données historiques relatives au projet. Par conséquent,

pour les projets récents ne disposant pas de données suffisantes, le modèle de prédiction doit se construire en se basant sur les données historiques d'autres projets similaires. Afin de sélectionner ces projets, les approches traditionnelles se basent sur des comparaisons statistiques. Cependant, celles-ci ne suffisent pas à capturer les différences importantes entre les projets sur plusieurs aspects tels que l'architecture, l'expérience du développeur, le style de codage, la sémantique, etc., et rendent la tâche de sélection plus complexe.

Ainsi, dans ce travail de thèse, le focus a été mis sur deux objectifs principaux : Premièrement, afin de combler l'écart entre les dépendances des programmes et les caractéristiques du code de prédiction des défauts, nous proposons un algorithme d'apprentissage en profondeur de bout en bout pour apprendre de manière plus complète la représentation du code source incluant différents niveaux d'abstractions tels que la syntaxe, la sémantique, les dépendances et ainsi construire un classifieur de prédiction de défauts plus performant, qui prend en compte toutes ces caractéristiques complexes. Les résultats expérimentaux montrent que notre approche améliore considérablement les approches existantes de prédiction des défauts. Deuxièmement, nous proposons une nouvelle méthode pour mieux sélectionner les projets similaires qui seront utilisés pour leurs données historiques, en se basant sur un apprentissage en profondeur des caractéristiques des projets. En évaluant notre méthode sur des projets open source, les résultats montrent qu'une sélection rigoureuse des projets améliore sensiblement les performances des techniques existantes et même notre approche de prédiction proposée, lorsqu'elle n'inclut pas de stratégie de sélection des projets externes d'apprentissage.