

REPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie
UNIVERSITE DE YAOUNDE 1
DEPARTEMENT
D'INFORMATIQUE
BP/P.O.Box 812
Yaounde-Cameroun



REPUBLIC OF CAMEROON
Peace-Work-Fatherland
UNIVERSITY OF YAOUNDE 1
COMPUTER SCIENCES
DEPARTMENT
BP/P.O.Box 812
Yaounde-Cameroun

EXPOSÉ INFO5029 (SCIENCES DES DONNÉES) :
**Scalable and Accurate Test Case Prioritization
in Continuous Integration Contexts**

Auteurs : Ahmadreza Saboor Yaraghi et al

Présenté par :

Prénom	Nom	Matricule	Option
CRESCENCE CATHERINE	AKAMBA MANI	18T2410	DS
MARTIAL	TEYOU GHOMFO	19M2364	GL
VICTOR NICO	DJIEMBOU TIENTCHEU	17T2051	DS

Superviseur : Pr. Norbert TSOPZE

Contents

1	Context	5
2	Limites des solutions existantes	5
3	Problème	5
4	Solution Proposé (Méthodologie)	5
5	Avantage de la solution proposé	5
6	Limites de cette proposition	5
7	Résultats expérimentaux (données, métriques d'évaluation, résultats)	5

Définition des mots clés

- **Le Machine Learning (apprentissage automatique)**[1] : est une branche de l'Intelligence Artificielle (IA) qui se concentre sur le développement de techniques permettant aux ordinateurs d'apprendre à partir de données et d'améliorer leurs performances sans être explicitement programmés. Le Machine Learning repose sur des algorithmes et des modèles statistiques qui permettent aux machines de reconnaître des schémas, de prendre des décisions et de faire des prédictions basées sur les données d'entrée. Il est utilisé dans de nombreux domaines, tels que la reconnaissance vocale, la vision par ordinateur, l'analyse prédictive et la recommandation personnalisée.
- **Le Software Testing (test logiciel)**[2] : est un processus d'évaluation systématique d'un logiciel afin de détecter les défauts, les erreurs et les incohérences qui pourraient affecter son bon fonctionnement. L'objectif du test logiciel est de s'assurer que le logiciel répond aux exigences spécifiées, de vérifier sa fonctionnalité, sa fiabilité, sa sécurité et sa performance. Le test logiciel peut inclure des activités telles que la création de cas de test, l'exécution de tests, la vérification des résultats et la documentation des problèmes identifiés.
- **La Test Case Prioritization (hiérarchisation des cas de test)**[3] : est une technique utilisée pour ordonner les cas de test en fonction de leur importance et de leur criticité. L'objectif est de déterminer l'ordre d'exécution optimal des cas de test afin de maximiser l'efficacité du processus de test. La hiérarchisation des cas de test peut se baser sur différents critères, tels que l'impact commercial, la criticité des fonctionnalités, la probabilité d'échec, les exigences spécifiques, etc.
- **La Test Case Selection (sélection des cas de test)**[4] : est le processus de choix des cas de test les plus pertinents et les plus appropriés pour être exécutés lors d'une campagne de test. La sélection des cas de test peut être basée sur différents critères, tels que la couverture des fonctionnalités, la complexité du code, les risques identifiés, les modifications récentes, etc. L'objectif est d'optimiser les ressources disponibles en se concentrant sur les cas de test qui sont les plus susceptibles de détecter des erreurs ou des défauts.
- **La Continuous Integration (intégration continue)**[5] : est une pratique de développement logiciel qui consiste à intégrer régulièrement les modifications de code réalisées par les membres d'une équipe de développement dans un dépôt centralisé. L'intégration continue est souvent associée à des processus d'automatisation tels que la compilation, les tests unitaires et l'analyse de code, qui sont déclenchés à chaque modification de code. L'objectif est de détecter rapidement les conflits, les erreurs et les incompatibilités, et de permettre une collaboration harmonieuse entre les développeurs.
- **Un Build** : fait référence à la construction automatique et à la compilation du code source d'une application logicielle, généralement dans le cadre d'un processus d'intégration continue (CI - Continuous Integration). Le build est une étape essentielle dans le cycle de vie du développement logiciel, car il permet de transformer le code source en un exécutable ou un artefact qui peut être déployé et testé.

1 Context

Le Genie logiciel pendant cette dernière décennie a connut des améliorations fulgurantes qui visent à faciliter la vie des développeurs et opérationnels. L'Intégration continue est l'une des clés de ces améliorations qui consiste à automatiser la chaîne de procédés de fusion des changements (modules ou patches) dans la chaîne logicielle tout en garantissant la non régression du logiciel en question. Ainsi, pour assurer cette non régression du logiciel plusieurs tests sont effectués de manière automatique pendant cette étape afin de vérifier la fiabilité des nouveaux changements. Ces tests de régression deviennent donc de plus en plus vaste et complexes ce qui entraîne notamment la mauvaise qualité des builds intégrés ainsi que le retard dans le déploiement et la livraison des produits logiciels d'où la problématique d'optimiser le temps et la qualité des tests pour pallier à ces problèmes.

En parallèle, avec l'évolution de la science des données notamment le Machine Learning (ML), d'autres optiques s'ouvrent pour pallier aux problèmes des tests de régression en Génie Logiciel.

2 Limites des solutions existantes

C'est ainsi que de nombreuses recherches ont fait usage des algorithmes de machine learning dans la priorisation des cas de test dans un processus d'intégration continue. Ses derniers avaient comme limites d'être extrêmement coûteux en temps et en ressources de calcul pour les meilleurs.

3 Problème

Comment optimiser la priorisation des cas de tests dans un processus d'intégration continue (IC) en utilisant des techniques basées sur l'apprentissage automatique (AA)?

4 Solution Proposé (Méthodologie)

5 Avantage de la solution proposé

6 Limites de cette proposition

7 Résultats expérimentaux (données, métriques d'évaluation, résultats)

References

- [1] Wikipedia, “Machine learning,” 2021, accessed: 2nd April 2024.
- [2] ——, “Software testing,” 2021, accessed: 2nd April 2024.
- [3] S. Yoo, M. Harman, G. Fraser, and H. Gross, “Test case prioritization techniques: a systematic literature review,” in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1105–1116.
- [4] S. Yoo, M. Harman, and G. Fraser, “Test case selection techniques: a systematic literature review,” *IEEE Transactions on Software Engineering*, vol. 42, no. 6, pp. 499–528, 2016.
- [5] Wikipedia, “Continuous integration,” 2021, accessed: 2nd April 2024.
- [6] A. S. Yaraghi, M. Bagherzadeh, N. Kahani, and L. C. Briand, “Scalable and accurate test case prioritization in continuous integration contexts,” *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1615–1639, 2022.