A Minor Project Synopsis

on

Bug and Fault Detection using ML

Submitted to Manipal University Jaipur

towards the partial fulfillment for the award of the degree of

Bachelor of Technology

In Information Technology

By

Saubhagya Chopra

209302107

IT-6F

Arihant Sogani

209302345

IT-6F

Under the Guidance of

Dr Sudhir Sharma

Department of Information Technology

School of Information Technology

Manipal University Jaipur

2022-2023

Synopsis

Introduction

In spite of diligent planning, documentation, and proper process adherence in software development, occurrences of defects are inevitable. In today's cutting-edge competition, it is important to make conscious efforts to control and minimize these defects by using techniques to allow in-process quality monitoring and control. Predicting the total number of defects before testing begins improves the quality of the product being delivered and helps in planning and decision making for future project releases.

With growing demand and technology, the software industry is rapidly evolving. Since humans do most of the software development, defects will inevitably occur. In general, defects can be defined as undesired or unacceptable deviations in software documents, programs, and data. Defects may exist in requirements analysis because the product manager misinterprets the customer's needs, and as a result, this defect will also carry on to the system design phase. Defects may also occur in the code due to inexperienced coders. Defects significantly impact software quality, such as increased software maintenance costs, especially in healthcare, and aerospace software defects can have serious consequences. If the fault is detected after deployment, it causes an overhead on the development team as they need to re-design some software modules, which increases the development costs. Defects are nightmares for reputed organizations. Their reputation is affected due to customer dissatisfaction and hence reduces its market share.

Therefore, software testing has become one of the main focuses of industrial research. With the rise in software development and software complexity, the number of defects has increased to the extent that traditional manual methods consume much time and become inefficient. The rise of machine learning has made automatic classification of defects a research hotspot. In this paper, we initially discuss software defects in detail and their different categories available in the literature and then discuss the manual classification methods proposed by various researchers. Finally, we present the analysis of the state of the art machine learning algorithms for automatic software detection.

Motivation

This area of study is important because of the cost saving potential it poses to Ericsson. If fault prone files are detected and testing is focused on the set of fault prone files, the testers can save time by not focusing the testing effort on files that are unlikely to contain faults and instead use that time to thoroughly test the fault prone files. By 1 Chapter 1. Introduction 2 identifying files likely to fail and focusing unit testing improvements to these areas, the testing process can be made more effective. The external quality of the product is also raised when defects are found and corrected before the product is shipped. An implementation of a tool that provides testers with information on which files that are more likely to fail can lead to some negative effects as well. Developers and testers may rely too much on the tool and actual faulty files that are not caught by the tool are not considered. This false sense of security can lead to faults slipping to later stages of the development and, as described before, introduce additional cost when correcting these faults. Several studies that investigate defect prone modules in the source code have been conducted, even within the domain of telecommunication. This master thesis differs in its approach, compared to previous studies. The novel approach we propose is to use metrics from test(s), that are testing the source code file, in a combined metric set. To the authors' knowledge, this approach have not been previously studied.

Project Objectives

i) It helps predict faults early in the development phase and is helpful in improving the software quality of the final product in a fast and cost effective manner.

ii) This also cuts down the project maintenance and support costs.

iii) It enables the software to significantly predict and recover software faults so that the software runs more effectively with less cost and time.

iv)To reduce software defects and improve software quality by identifying the potential faults before they occur.

v)To minimize the cost of fixing faults by detecting them early in the software development cycle.

vi)To prioritize software testing and maintenance efforts by focusing on the most critical parts of the software.

vii)To improve software reliability and performance by reducing the number of faults in the code.

Methodology

Data Collection: Software development data, including source code, bug reports, and development history, will be collected and pre-processed for analysis.

Feature Extraction: Relevant features will be selected from the pre-processed data for use in training the models.

Model Training: Machine learning/deep learning algorithms, such as decision tree, KNN,SVM will be used to train the models on the selected features.

Model Evaluation: The performance of the trained models will be evaluated using metrics such as accuracy, precision, and recall.

Fine-Tuning: Based on the results of the model evaluation, the models will be fine-tuned for improved accuracy.

GANTT CHART:

Task Name	Start Date	End Date	Duration (days)
Data Collection	1/2/2023	15/2/2023	15
Feature Extraction	16/2/2023	28/2/2023	15
Model Training	1/3/2023	20/3/2023	20
Model Evaluation	21/3/2023	10/4/2023	20

Facilities required for proposed work

OS: Windows/Mac

Web browser: Chrome,safari etc

RAM: 4 GB

Programming Languages: Python 3.8.8

Application Tool: Google Colab

References

1. The Impact of Software Fault Prediction in Real-World Application: An Automated Approach for Software Engineering (International Conference on Computing and Data Engineering (ICCDE2020) Sanya, China)

2. H. B. Bolat, G. T. Temur, and IGI Global, Agile approaches for successfully managing and executing projects in the fourth industrial revolution.
3. G. P. Bhandari and R. Gupta, “Machine learning based software fault prediction utilizing source code metrics,” in 250 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), 2018, pp. 40–45.
4. R. Malhotra, “A systematic review of machine learning techniques for software fault prediction,” Appl. Soft Comput., vol. 27, pp. 504–518, Feb. 2015.
5. L. Son et al., “Empirical Study of Software Defect Prediction: A Systematic Mapping,” Symmetry (Basel)., vol. 11, no. 2, p. 212, Feb. 2019.