RESEARCH METHODOLOGY

ROLL NO: 24K-8203
MUHAMMAD MOIZ KHAN

APPLIED SOFTWARE TESTING REVOLUTION

ABSTRACT

Covering automated testing, manual testing, and more difficult approaches like stress testing, regression testing, and mutation testing. In addition to discussing the development of software testing tools and highlighting the shift from manual to automated testing, the study evaluates the advantages and disadvantages of each technique. In order to help software developers and testers select the best testing strategy for their projects, the difficulties that come with each methodology are also covered, Because automated software testing can improve testing efficiency and decrease human error, it has become increasingly common. The main difficulties in automated testing, including test coverage problems, tool compatibility, and test script maintenance, are covered in this study. In order to overcome these obstacles, it investigates a number of strategies, such as model based testing and self healing test automation. Additionally, the study shows how new developments in the industry, such the incorporation of AI and machine learning into testing frameworks, could influence automated software testing in the future. The optimization methods known as genetic algorithms (GAs), which draw inspiration from natural selection processes, have been used in software testing and other software engineering domains. The application of evolutionary algorithms to software testing to optimize test case generation is examined in this research. The goal of the combinatorial testing method known as pair wise testing is to create test cases that encompass every possible combination of input values. By using it on various software systems and contrasting its results with more conventional testing techniques, this study examines the efficacy of paired testing in software development. The findings demonstrate that paired testing can decrease testing time and expenses by achieving higher fault detection rates with fewer test cases. The study also talks about how paired testing might help find flaws that conventional testing techniques might miss. Software testing frequently uses code coverage as a statistic to evaluate how comprehensive testing efforts are. This study examines how code coverage metrics and software testing quality are related, examining how well the various forms of coverage statement, branch, and path detect flaws. The paper uses empirical data to show that although there is a general correlation between better code coverage and defect detection, discovery of all potential mistakes is not guaranteed. Best practices for using coverage metrics are recommended in the article, which also highlights the significance of striking a balance between coverage and other testing considerations including test case efficacy and execution time.