Chapter 20

Use of Software Metrics to Improve the Quality of Software Projects Using Regression Testing

Arshpreet Kaur Sidhu
Chandigarh University, India

Sumeet Kaur Sehra
GNDEC, India

ABSTRACT

Testing of software is broadly divided into three types i.e., code based, model based and specification based. To find faults at early stage, model based testing can be used in which testing can be started from design phase. Furthermore, in this chapter, to generate new test cases and to ensure the quality of changed software, regression testing is used. Early detection of faults will not only reduce the cost, time and effort of developers but also will help finding risks. We are using structural metrics to check the effect of changes made to software. Finally, the authors suggest identifying metrics and analyze the results using NDepend simulator. If results show deviation from standards then again perform regression testing to improve the quality of software.

INTRODUCTION

Testing of software is fundamental and central part of software development process. UML is the most prevailing standard language used in modeling test cases (A. K. Jena et al, 2014). Therefore, if it is pleasingly exploited it will reduce the cost and effort of testing as well as modification in code. Activity diagrams can improve the quality of the generated test cases as well as use these test cases for regression testing because it shows the overall flow of control between activities and object using activity-based relationships (Ye et al., 2011).

Testing of Software

Before moving to the details of work done in this chapter, we are focusing on testing first. Testing of software in a simple language is a method to find out errors and missed requirements as specified by the user. Testing can be performed by developers as unit testing and can be performed by software testing professionals to get good quality product.

Testing can be started from the early phase of software i.e. requirement phase and can be done up till deployment. Early start of testing will help in reducing the cost and time of developers because the errors and gaps can be removed as earliest possible. Testing is one process which is not confined at testing phase only. At requirement phase testing is performed in the form of analyzing and verifying the requirements. At design phase reviewing the design documents with gathered requirements is also a part of testing. At coding phase, the part of unit testing performed by developer at completion of code is also a part of testing.

Testing and Quality Assurance

Testing and Quality Assurance are interrelated terms. Testing insures the identification of errors and bugs in software whereas quality assurance focuses on accomplishment of processes and standards in the direction of verification of software. Testing is error and bug identifying technique and do not fixes bugs. To fix bugs at coding level developers can use debugging which is a part of white box and black box testing.

Types of Testing

Testing is categorized as manual and automated. In manual testing software is tested manually and end users take the role of testers to test software for bugs and unforeseen behavior. Stages at which manual testing is done are unit, integration, system and user acceptance testing. In automated testing also known as test automation the tester writes script and uses different software to test the functionality of given software. In this testing test scenarios are re-run repeatedly which were performed manually for time saving. It is assumed that automated testing improves test coverage and accuracy. GUI items, connections can be effectively tested with the help of test automation (Rathi et al., 2015; Bhullar et al., 2017).

Methods of Testing

Black Box Testing

It is a technique of testing software without knowing the internal details and working of software. Generally, in black box testing testers interact with the user interface of system by giving inputs to the system and analyzing the outputs. They hardly have details about where the inputs are working in code. This method is preferred for large segment of code but has limited coverage of test cases.

Related Content

A Novel Spatial Data Pipeline for Streaming Smart City Data
Chase D. Carthen, Araam Zaremehrjardi, Vinh Dac Le, Carlos G. Cardillo, Scotty Strachan, Alireza Tavakkoli, Frederick C. Harris, Jr.and Sergiu M. Dascalu (2024). *International Journal of Software Innovation (pp. 1-15).*
www.irma-international.org/article/a-novel-spatial-data-pipeline-for-streaming-smart-city-data/359180

Industrial Applications of Emulation Techniques for the Early Evaluation of Secure Low-Power Embedded Systems
Norbert Druml, Manuel Menghin, Christian Steger, Armin Krieg, Andreas Genser, Josef Haid, Holger Bockand Johannes Grinschgl (2014). *Handbook of Research on Embedded Systems Design (pp. 328-346).*
www.irma-international.org/chapter/industrial-applications-of-emulation-techniques-for-the-early-evaluation-of-secure-low-power-embedded-systems/116116

Fault-Tolerant Protocols Using Single- and Multiple-Version Software Fault-Tolerance
Vincenzo De Florio (2009). *Application-Layer Fault-Tolerance Protocols (pp. 53-132).*
www.irma-international.org/chapter/fault-tolerant-protocols-using-single/5123

Disciplined Teams vs. Agile Teams: Differences and Similarities in Software Development
Antonio Alexandre Moura Costa, Felipe Barbosa Araújo Ramos, Dalton Cézane Gomes Valadares, Danyllo Wagner Albuquerque, Emanuel Dantas Filho, Alexandre Braga Gomes, Mirko Barbosa Perkusichand Hyggo Oliveira de Almeida (2022). *Research Anthology on Agile Software, Software Development, and Testing (pp. 40-55).*
www.irma-international.org/chapter/disciplined-teams-vs-agile-teams/294457

Fostering Analysis from Industrial Embedded Systems Modeling
Michel Bourdellès, Shuai Li, Imran Quadri, Etienne Brosse, Andrey Sadovykh, Emmanuel Gaudin, Frédéric Mallet, Arda Goknil, David Georgeand Jari Kreku (2014). *Handbook of Research on Embedded Systems Design (pp. 283-300).*
www.irma-international.org/chapter/fostering-analysis-from-industrial-embedded-systems-modeling/116113