Volume-4, Issue-5, May-2015 • ISSN No 2277 - 8160

Research Paper Technology

Prioritization of Test Cases in a Scenario Using Mutation and Crossover

TEJAL SHARMA CSE , Mtech

Mahesh Singh Asst. Professor CSE Department, Advanced Institute of Technology & Management

ABSTRACT

Software testing provides a means to reduce errors, cut maintenance and overall software costs. Numerous software development and testing methodologies, tools, and techniques have emerged over the last few decades promising to enhance software quality. While it can be argued that there has been some improvement it is apparent that many of the techniques and tools are isolated to a specific lifecycle phase or functional area. One of the major problems within software testing area is how to get a suitable set of cases to test a software system. This paper presents a paper for testing in our software project. In this we will make some test cases and apply different types of testing to it. We will also make use of genetic algorithm i.e. mutation and crossover in it.

KEYWORDS : Software Testing, Test cases ,types of testing.

Introduction

In the testing of this project, we created different test cases in order to perform module-wise testing. The test cases were formulated keeping the overall objectives of the applications into consideration. In other words, we tested a certain module to ensure that it should perform its own function in addition to some other related functionality with other modules, if necessary. We recorded our testing results by giving different inputs to the modules and observing the actual result as against the expected one. Wherever, the test result failed, we incorporated the essential modifications to correct it.

Background work

Software has been tested as early as software has been written. The concept of testing itself evolved with time. The evolution of definition and targets of software testing has directed the research on testing techniques In the significant book Software Testing Techniques [2], which contains the most complete catalog of testing techniques, Beizer stated that "the act of designing tests is one of the most effective bug preventers known," which extended the definition of testing to error prevention as well as error detection activities. This led to a classic insight into the power of early testing. In 1991, Hetzel gave the definition that "Testing is planning, designing, building, maintaining and executing tests and test environments." A year before this, Beizer gave four stages of thinking about testing: 1. to make software work, 2, to break the software, 3, to reduce risk, and 4, a state of mind, i.e. a total life-cycle concern with testability. These ideas led software testing to view emphasize the importance of early test design throughout the software life cycle. The prevention-oriented period is distinguished from the evaluation-oriented by the mechanism, although both focus on software requirements and design in order to avoid implementation errors. The prevention model sees test planning, test analysis, and test design activities playing a major role, while the evaluation model mainly relies on analysis and reviewing techniques other than testing

Test cases

S.NO.	MODULE	START DATE	END DATE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	TEST RESULT
1)	Project Title (Home Screen)			Blank	The user should be prompted through an error message	The error message is appearing	Ok
2)	Story based requirements input			Blank	The user should be prompted through an error message	The error message is appearing	ok
3)	Story based requirements input			Proper requirements with commas and full stops	The strings tokenization or parsing should happen at commas & full-stops	The requirements are parsed and extracted on commas & full-stops with pairing	Ok
4)	Requirements Extraction & Validity			User's selection of valid requirements through a check box.	The checked requirements should be correctly stored in the database	The checked requirements are correctly getting stored in the database	Ok
5)	Application of quality attributes and sub-attributes to the extracted requirements			At least one attribute is not applied to set of extracted requirements	The user should be alerted and prompted about the same.	The alert message is generated in the form of a dialog box.	Ok
6)	Application of quality attributes and sub-attributes to the extracted requirements			At least one attribute is applied to set of extracted requirements	The overall desirability factor of the individual requirements should be successfully calculated	Overall desirability factor is successfully getting calculated	Ok
7)	Obtaining a set of 5 requirements prioritization from different users			The 5 requirements set are not correctly inputted in a proper, formatted manner	The user should be alerted about same	Prompt message in the form of dialog box is appearing	ok

8)	Obtaining a set of 5 requirements prioritization from different users			1 user inputs multiple sets of prioritized requirements set	An alert message should be generated	The alert message is getting generated that 1 user can furnish only 1 set of requirements prioritization	Ok
9)	Obtaining a set of 5 requirements prioritization from different users			If all the 5 requirements prioritization sets are inputted correctly in a formatted manner and one per user	Disagreement count factor should be correctly generated for each requirement set after its comparison with the main set of developer's perspective of requirements prioritization	Disagreement count factor is correctly getting calculated and displayed in the form of a grid for each of the 5 inputted requirements prioritization sets	Ok
10)	Once the disagreement count factors are generated, they should be minimized using the Mutation & Crossover operations			After the counts are generated, they are correctly un-equalized and then applied the mutation and crossover functions for further minimization of the disagreement count	The best set of prioritized requirements set should be generated.	The best set is getting generated and outputted.	Ok

Types of testing involved

Type of Test	Will Test be Performed	Comments	Software Component
Requirements Testing	Yes	In this testing, we test the feasibility of the project requirements in terms of complexity, development environment and time	In this phase, we will test all the software modules that are developed in context to the individual project requirements. These include Furnishing of requirements, Parsing, Disagreement factor calculation & Mutation & Crossover.
Unit Testing	Yes	In this testing, we will perform the individual modules of the project	Home Page, Requirements Collection (Text & Graphics based), Extraction, Disagreement & Mutation & Crossover
Integration	Yes	In this phase, we have tested whether several modules are working in cohesion	This testing included testing of Home module with Requirements Furnishing module. Requirements Furnishing module with extraction module and extraction module with desirability factor and desirability factor module with disagreement calculation and it with mutation & crossover.
Stress / Load	No		
Performance	Yes	In this testing we evaluated the performance of every module whether it is working as per specified expectations or not	All modules.
Security	No		

Conclusion

Testing can show the presence of faults in a system; it cannot prove there are no remaining faults. Component developers are responsile for component testing; system testing is the responsibility of a separate team.Integration testing is testing increments of the system; release testing involves testing a system to be released to a customer. Use experience and guidelines to design test cases in defect testing. Interface testing is designed to discover defects in the interfaces of composite components. Equivalence partitioning is a way of discovering test cases - all cases in a partition should behave in the same way.♣ Structural analysis relies on analysing a program and deriving tests from this analysis. Test automation reduces testing costs by supporting the test process with a range of software tools.

REFERENCES

1. Application of Genetic Algorithm in Software Testing Praveen Ranjan Srivastava1 and Tai-hoon Kim2 1, Computer Science & Information System Group, BITS PILANI – 333031 (INDIA) praveenrsrivastava@gmail.com 2 Dept. of Multimedia Engineering, Hannam University, Korea taihoonn@hnu.kr | 2. The Automatic Generation of Software Test Data Using Genetic Algorithms by Harmen - Hinrich Sthamer University of Glamorgan / Prifvsgol Morgannwg for the degree of a Doctor of Philosophy. November 1995 University of Glamorgan | 3. Software Testing Techniques Technology Maturation and Research Strategies Lu Luo School of Computer Science Carnegie Mellon University | 4. International Journal of Advanced Research in Computer Science and Software Engineering Research Paper Available online at: www.ijarcsse.com Software Testing Techniques Shivkumar Hasmukhrai Trivedi [B.com, M.Sc I.T (Information Technology), P.G.D.B.M –Pursuing] Senior System Administrator, S.E.C.C [Socio Economic and Cast Census] Central Govt. Project – Bhavnagar [Gujarat – India], | 5. Software Testing: A Research Travelogue (2000–2014) Alessandro Orso College of Computing School of Computer Science Georgia Institute of Technology Atlanta, GA, USA orso@cc.gatech.edu Gregg Rothermel Department of Computer Science and Engineering University of Nebraska - Lincoln Lincoln, NE, USA grother@cse.unl.edu |