INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume3, Issue3)
Available online at www.ijariit.com

Review on Structural Software Testing Coverage Approaches

Monika Thakur
Himachal Pradesh Technical University, Hamirpur
mtmonu22@gmail.com

Sanjay
Himachal Pradesh Technical University, Hamirpur
samjess.jess@gmail.com

Abstract: ***With the aim of discovering software bugs (blunders or different deformities) the test procedures incorporate the way toward executing a program or application and confirming that the software product is fit for utilize. To assess at least one properties of intrigue the software testing includes the execution of a software part or framework segment. In this paper, we review the different software testing and its methods.***

Keywords: Software testing, Bug, White box testing, Black box testing, Dynamic Testing.

I. INTRODUCTION

With information about the quality of the product or service under test to provide stakeholders the software testing is investigation conducted. To allow the business to appreciate and understand the risks of software implementation the software testing can also provide an objective, independent view of the software. With the intent of finding software bugs (errors or other defects), the test techniques include the process of executing a program or application and verifying that the software product is fit for use. To evaluate one or more properties of interest the software testing involves the execution of a software component or system component.

In general, these properties indicate the extent to which the component or system under test:

- Meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs,
- Performs its functions within an acceptable time,
- Is sufficiently usable,
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholder's desire.

For even simple software components is practically infinite as the number of possible tests, for the available time and resources all the software testing uses some strategy to select tests that are feasible. As a result, with the intent of finding software bugs (errors or other defects) the software testing typically (but not exclusively) attempts to execute a program or application. When one bug is fixed the job of testing is an iterative process, deeper bugs can even create new ones, or it can illuminate other. About the quality of software and risk of its failure to users or sponsors, the software testing can provide objective, independent information. As soon as executable software (even if partially complete) exist the software testing can be conducted. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

II. SOFTWARE TESTING METHOD

Static vs. dynamic testing

In software testing, there are many approaches available. Reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing.

Static testing is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program

itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for this are either using stubs/drivers or execution from a debugger environment.

Static testing involves verification, whereas dynamic testing involves validation. Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test cases will detect errors which are introduced by mutating the source code.

The box approach
Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

White-box testing
White-box testing (also known as clear box testing, glass box testing, and transparent box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

Black-box testing
Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

III. LITERATURE REVIEW

Chengying Mao et.al. [2] In this paper, the basic ACO algorithm is reformed into discrete version so as to generate test data for structural testing. First, the technical road map of combining the adapted ACO algorithm and test process together is introduced. In order to improve algorithm's searching ability and generate more diverse test inputs, some strategies such as local transfer, global transfer, and pheromone update are defined and applied.

Phil McMinn et. al. [3] Search-Based Software Testing is the use of a meta-heuristic optimizing search technique, such as a Genetic Algorithm, to automate or partially automate a testing task; for example the automatic generation of test data. The key to the optimization process is a problem-specific fitness function. The role of the fitness function is to guide the search to good solutions from a potentially infinite search space, within a practical time limit. Work on Search-Based Software testing dates back to 1976, with interest in the area beginning to gather pace in the 1990s. More recently there has been an explosion of the amount of work.

Dudekula Mohammad Raf et. al. [4] this paper tries to close the gap by investigating both views regarding the benefits and limits of test automation. The academic views are studied with a systematic literature review while the practitioner's views are assessed with a survey, where it received responses from 115 software professionals. The results of the systematic literature review show that the source of evidence regarding benefits and limitations is quite shallow as only 25 papers provide the evidence. Furthermore, it was found that benefits often originated from stronger sources of evidence (experiments and case studies), while limitations often originated from experience reports.

Shkodran Zogaj et.al. [5] Crowdsourcing intermediaries play a key role in crowdsourcing initiatives as they assure the connection between the crowdsourcing companies and the crowd. However, the issue of how crowdsourcing intermediaries manage crowdsourcing initiatives and the associated challenges has not been addressed by research yet. It addresses these issues by conducting a case study with a German start-up crowdsourcing intermediary called test Cloud that offers software testing services for companies intending to partly or fully outsource their testing activities to a certain crowd. The case study shows that test Cloud faces three main challenges, these are: managing the process, managing the crowd and managing the technology.

Alessandro Orso et. al. [6] goal of this paper was to present and discuss (some of) the most successful software testing research performed since the year 2000. It is undoubtedly difficult, if not impossible, to summarize in any complete way almost 15 years of research and cite all relevant papers in a relatively short report such as this one. We, therefore, did not attempt to cover all relevant themes and efforts, but rather focused on those that our colleagues and we thought were particularly relevant, had already had a considerable impact, or seemed likely to have an impact in the (near) future.

Mark Harman et. al. [7] Testing involves examining the behaviour of a system in order to discover potential faults. Determining the desired correct behaviour for a given input is called the "oracle problem". Oracle automation is important to remove a current bottleneck which inhibits greater overall test automation; without oracle automation, the human has to determine whether observed behaviour is correct. The literature on oracles has introduced techniques for oracle automation, including modeling, specifications, contract-driven development, and metamorphic testing. When none of these is completely adequate, the final source of oracle information remains the human, who may be aware of informal specifications, expectations, norms and domain specific information that provides informal oracle guidance. All forms of the oracle, even the humble human, involve challenges of reducing cost and increasing benefit. This paper provides a comprehensive survey of current approaches to the oracle problem and an analysis of trends in this important area of software testing research and practice.

Vahid Garousi et. al. [8] it report the survey design, execution and results in this article. The survey results reveal important and interesting findings of software testing practices in Canada. Whenever possible, it also compares the results of this survey to other similar studies, such as the ones conducted in the US, Sweden, and Australia, and also two previous Alberta-wide surveys, including our 2009 survey. The results of the survey will be of interest to testing professionals both in Canada and worldwide. It will also benefit researchers in observing the latest trends in software testing industry identifying the areas of strength and weakness, which would then hopefully encourage further industry-academia collaborations in this area.

Wang Jun et. al. [9] Cloud testing is the method of software testing based on cloud computing technology. In this paper, the definition of cloud testing was derived from the concept of cloud computing. It analyzed the questions of which software testing projects can do the cloud testing, why do clouds testing, how to do cloud testing. This paper was a research for the future software testing methods.

Selvam R et. al. [10] Mobile devices are poised to challenge PCs as the application platform of choice, with 500 million mobile internet devices expected to ship in 2012 compared to 150 million PCs. The convergence of all digital devices into mobile platform model augments the software companies, software developer, and venture capitalist firms to turn their focus into mobile application platform (for example mobile social networking application like Facebook and mobile VOIP like Skype) a futuristic platform for increased revenue, new challenges and growth potential. But the commercial success of these applications depends on their working smoothly and securely on a wide variety of handheld devices and wireless networks. More and more virtual mobile application stores are built on the web. The web itself is in the transforming form to adapt to the mobile devices to thrive on. The sudden growth in the mobile application and the complexity in the divergence of the devices that uses these applications present increased challenges and opportunities for the software testing companies and software testers to conquer this small device. Performing such testing quickly and cost-effectively greatly expands the market for such applications. This paper deals the nuances of Automated Test Case Design Strategies for Mobile Software Testing.

Eliane Collins et. al. [11] Testing Automation has been growing in software engineering. Many organizations are investing in automated testing in order to prevent defects and increase testing effectiveness during software development. In agile methodologies, this task is considered an important activity, considered the key to the agile testing. This paper presents three testing automation strategies applied to three different software projects adopting Scrum agile methodology. The results indicated positive agile practices to be considered when adopting testing automation strategy, such as team collaboration, task distribution, testing tools, knowledge managements. The challenges, results, and lessons learned from this experience are also discussed.

Author Name	Year	Technology Used	Description
Chengying Mao et.al.	2015	Adapting ant colony optimization	In this paper, the basic ACO algorithm is reformed into discrete version so as to generate test data for structural testing. First, the technical road map of combining the adapted ACO algorithm and test process together is introduced. In order to improve algorithm's searching ability and generate more diverse test inputs, some strategies such as local transfer, global transfer, and pheromone update are defined and applied.
Phil McMinn et. al.	2011	Search-based software testing	Search-Based Software Testing is the use of a meta-heuristic optimizing search technique, such as a Genetic Algorithm, to automate or partially automate a testing task; for example the automatic generation of test data. The key to the optimization process is a problem-specific fitness function. The role of the fitness function is to guide the search to good solutions from a potentially infinite search space, within a practical time limit. Work on Search-Based Software Testing dates back to 1976, with interest in the area beginning to gather pace in the 1990s. More recently there has been an explosion of the amount of work.
Dudekula Mohammad Raf et. al.	2012	automated software testing	This paper tries to close the gap by investigating both views regarding the benefits and limits of test automation. The academic views are studied with a systematic literature review while the practitioner's views are assessed with a survey, where it received responses from 115 software professionals. The results of the systematic literature review show that the source of evidence regarding benefits and limitations is quite shallow as only 25 papers provide the evidence. Furthermore, it was found that benefits often originated from stronger sources of

			evidence (experiments and case studies), while limitations often originated from experience reports.
Shkodran Zogaj et.al.	2014	crowdsourced software testing	Crowdsourcing intermediaries play a key role in crowdsourcing initiatives as they assure the connection between the crowdsourcing companies and the crowd. However, the issue of how crowdsourcing intermediaries manage crowdsourcing initiatives and the associated challenges has not been addressed by research yet. It addresses these issues by conducting a case study with a German start-up crowdsourcing intermediary called testCloud that offers software testing services for companies intending to partly or fully outsource their testing activities to a certain crowd. The case study shows that testCloud faces three main challenges, these are: managing the process, managing the crowd and managing the technology.
Alessandro Orso et. al.	2014	Software testing	the goal of this paper was to present and discuss (some of) the most successful software testing research performed since the year 2000. It is undoubtedly difficult, if not impossible, to summarize in any complete way almost 15 years of research and cite all relevant papers in a relatively short report such as this one. Therefore did not attempt to cover all relevant themes and efforts, but rather focused on those that our colleagues and we thought were particularly relevant, had already had a considerable impact, or seemed likely to have an impact in the (near) future.
Mark Harman et. al.	2013	A comprehensive survey	Testing involves examining the behaviour of a system in order to discover potential faults. Determining the desired correct behaviour for a given input is called the "oracle problem". Oracle automation is important to remove a current bottleneck which inhibits greater overall test automation; without oracle automation, the human has to determine whether observed behaviour is correct. The literature on oracles has introduced techniques for oracle automation, including modeling, specifications, contract-driven development, and metamorphic testing. When none of these is completely adequate, the final source of oracle information remains the human, who may be aware of informal specifications, expectations, norms and domain specific information that provides informal oracle guidance. All forms of the oracle, even the humble human, involve challenges of reducing cost and increasing benefit. This paper provides a comprehensive survey of current approaches to the oracle problem and an analysis of trends in this important area of software testing research and practice.
Vahid Garousi et. al.	2013	software testing	it reports the survey design, execution, and results in this article. The survey results reveal important and interesting findings of software testing practices in Canada. Whenever possible, it also compares the results of this survey to other similar studies, such as the ones conducted in the US, Sweden, and Australia, and also two previous Alberta-wide surveys, including our 2009 survey. The results of the survey will be of interest to testing professionals both in Canada and worldwide. It will also benefit researchers in observing the latest trends in software testing industry identifying the areas of strength and weakness, which would then hopefully encourage further industry-academia collaborations in this area.

Wang Jun et. al.	2011	Software testing	Cloud testing is the method of software testing based on cloud computing technology. In this paper, the definition of cloud testing was derived from the concept of cloud computing. It analyzed the questions of which software testing projects can do the cloud testing, why do clouds testing, how to do cloud testing. This paper was a research for the future software testing methods.
Selvam R et. al.	2011	Mobile Software Testing- Automated Test Case Design Strategies	Mobile devices are poised to challenge PCs as the application platform of choice, with 500 million mobile internet devices expected to ship in 2012 compared to 150 million PCs. The convergence of all digital devices into mobile platform model augments the software companies, software developer, and venture capitalist firms to turn their focus into mobile application platform (for example mobile social networking application like Facebook and mobile VOIP like Skype) a futuristic platform for increased revenue, new challenges and growth potential. But the commercial success of these applications depends on their working smoothly and securely on a wide variety of handheld devices and wireless networks. More and more virtual mobile application stores are built on the web. The web itself is in the transforming form to adapt to the mobile devices to thrive on.
Eliane Collins et. al.	2012	agile software testing automation	Testing Automation has been growing in software engineering. Many organizations are investing in automated testing in order to prevent defects and increase testing effectiveness during software development. In agile methodologies, this task is considered an important activity, considered the key to the agile testing. This paper presents three testing automation strategies applied to three different software projects adopting Scrum agile methodology. The results indicated positive agile practices to be considered when adopting testing automation strategy, such as team collaboration, task distribution, testing tools, knowledge managements. The challenges, results and lessons learned from this experience are also discussed.

CONCLUSION

With data about the nature of the item or administration under test to give partners the software testing is examination directed. To enable the business to acknowledge and comprehend the dangers of software usage the software testing can likewise give a target, the free perspective of the software. With the goal of discovering software bugs (blunders or different imperfections), the test methods incorporate the way toward executing a program or application and confirming that the software item is fit for utilize. To assess at least one properties of intrigue the software testing includes the execution of a software component or framework component. In this paper software testing and its method is reviewed.

REFERENCES

[1] https://en.wikipedia.org/wiki/Software_testing.

[2] Mao, Chengying, et al. "Adapting ant colony optimization to generate test data for software structural testing." *Swarm and Evolutionary Computation* 20 (2015): 23-36.

[3] McMinn, Phil. "Search-based software testing: The Past, present, and future." *Software testing, verification and validation workshops (icstw), 2011 ieee fourth international conference on*. IEEE, 2011.

[4] Rafi, Dudekula Mohammad, et al. "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey." *Proceedings of the 7th International Workshop on Automation of Software Test*. IEEE Press, 2012.

[5] Zogaj, Shkodran, Ulrich Bretschneider, and Jan Marco Leimeister. "Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary." *Journal of Business Economics* 84.3 (2014): 375-405.

[6] Orso, Alessandro, and Gregg Rothermel. "Software testing: a research travelogue (2000–2014)." *Proceedings of the on Future of Software Engineering*. ACM, 2014.

[7] Harman, Mark, et al. "A comprehensive survey of trends in oracles for software testing." *The university of Sheffield, Department of Computer Science, Tech. Rep. CS-13-01* (2013).

[8] Garousi, Vahid, and Junji Zhi. "A survey of software testing practices in Canada." *Journal of Systems and Software* 86.5 (2013): 1354-1376.

[9] Jun, Wang, and Fanpeng Meng. "Software testing based on cloud computing." *Internet Computing & Information Services (ICICIS), 2011 International Conference on*. IEEE, 2011.

[10] Selvam, R., and V. yKarthikeyan. "Mobile Software Testing- Automated Test Case Design Strategies." *International Journal of Computer Science and Engineering* 3.4 (2011): 1450-1461.

[11] Collins, Eliane, April Dias-Neto, and Vicente F. de Lucena Jr. "Strategies for agile software testing automation: An industrial experience." *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th*.

