ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

Establishing a Test Case Prioritization Technique Using Dependency Estimation of Functional Requirement

T.Dinesh Parthiban, Mr.R.Kamalraj, Dr.S.Karthik

PG Scholar, Department of CSE, SNS college of technology, Coimbatore, India

Associate Professor, Department of CSE, SNS college of technology, Coimbatore, India

Dean & Professor, Department of CSE, SNS college of technology, Coimbatore, India

Abstract: Test case prioritization is the process of ordering the execution of test cases to increase the rate of fault detection. Increasing the rate of error detection can be more feedback to system developers , improving debt establish activity and , ultimately , software delivery. Many existing test case prioritization techniques believe that tests can be performed in any order. Functional dependencies that may exist between a number of test cases , a test case should be performed before another , often not the case . A family of test case prioritization techniques is presented using the dependency information from a test suite to test suite that priority. The nature of the techniques preserves the ordering dependencies in the test . The hypothesis of this work is that dependencies between tests represent interactions in the system under test and perform less complex interactions would increase the error detection compared with random testing arrangements. Empirical evaluations built on six systems towards industry show that these techniques increase the speed of fault detection in comparison with the results of the untreated order rates , random assignments , and ordered test suites under existing purposes " - coarse grained " techniques based on function coverage.

Keywords: Software engineering, testing and debugging, test execution

1.INTRODUCTION

A software product development organization invests resources in product development and expects maximal added value from their investments. This means that providing value to different customer and end-user segments with products is a necessity for the business of product development companies. Providing value with the product requires, however, a successful selection of the requirements to be implemented in the products.Requirements prioritization is defined as an activity during which the most important requirements for the system (or release) should be discovered. In practice, only a limited set of requirements can be implemented in one release, but the product should, however, meet the needs of the customers and reach the markets in time. This means that trade-offs have to be made during the development work.

1.1 THE BACKGROUND OF THE RESEARCH

The ultimate sponsors of the project expect that the project's end result will be to add more value for them than they are paying the project team to create it. On a high level, this means that companies expect their product development organization to add more value to them than they invest in product development. Prioritizing requirements is recognized as an important activity to ensure value provision in product development. By definition, requirements prioritization is an activity during which the most important requirements for the system (or release) should be discovered. Origins for the importance of prioritization are in limited product development resources, since time and money are finite in practice. When customer expectations are high and timelines short, the product must deliver the most essential functionality as early as possible. However, the scope of each release must be limited. The challenge is therefore to select the 'right' requirements out of a given superset of candidate

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

requirements so that all the different key interests, technical constraints, and preferences of the critical stakeholders are fulfilled and the overall business value of the product is maximized. Requirements prioritization is, however, also recognized as a very challenging activity. It is widely 3 accepted that requirements prioritization involves complex decision-making. In order to prioritize requirements successfully, domain knowledge and estimation skills are required. . In addition, requirements depend on each other and priorities are always relative. An important requirement in one release or to a certain customer may not be as important in the next release or to another customer.

The aim is to investigate the current state of practice in the area of requirements prioritization in software companies operating in the product business and the relationship between industrial practice and requirements prioritization methods from the literature.

The focus is on how the prioritization and selection of requirements is organized in software product development organizations and what the practical challenges involved are. In addition, the suitability of prioritization methods for solving these challenges is investigated.

Requirements prioritization is an activity during which the most important requirements for the system (or release) should be discovered. This concept originates from the context of the development of customer-specific systems, where all the requirements are elicited, analyzed, documented, and validated within one project. Many of the findings concerning requirements prioritization in the literature, can, however, be generalized to concern the prioritization of features as well. To avoid unnecessary complexity, the term requirements prioritization is used as a general term for both feature and requirements prioritization. Additionally, the term prioritization practice is used as a general term for any activity performed to find the optimum implementation order of features or requirements.

2. RELATED WORKS

2.1 Test Case Prioritization

Prioritization is a process of scheduling test case to be executed in a particular order so that the test case with higher priority is executed first in the sequence. It is necessary to execute test suite in order of priority to utilize limited resource and time effectively. The main aim of is to increase the fault detection for a test suite. The priority is defined relative to some test criteria.

2.2 Open and closed dependency structure

A Open dependency structure is one in which a dependency between the two test cases t1 and t2 specifies that t1 must execute before t2 but not immediately.

A Closed dependency structure is not same as open dependency, dependency between the two test cases t1 and t2 specifies that t1 must execute immediately before t2. Some dependency structure contains both the open and closed dependency such structure is named as closed dependency structure. The closed dependency structure is regrouped into a single test, resulting in an open dependency structure.

2.3 Independent and dependent test case

Independent test case is a test case whose execution of one test case is not dependent on any other test cases.

Dependent test case is a test case whose execution of one test case is dependent on any other test cases.

There are several tests related to these two operations;however, we consider only five for illustration:

1. select a binary file,
2. select a record file (a nonbinary file),
3. attempt to read a binary file where the selected file is not a binary file,
4. attempt to update a binary file where the selected file is not a binary file, and
5. attempt to update a binary where the selected file is a binary file and is successfully read.

2.4 Prioritizing test cases based on dependency structure

The dependency structure between test cases is closely related to the interaction between the

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

parts of systems. They found that concatenating test together increases the fault detection rate of the tests due to the interaction occur between the tests.

Dependency structure prioritization is a technique that assign priority based on a graph coverage value. The graph coverage value of a test case is the measurement of the complexities of the dependents of the test case.

Two ways to measure the graph coverage value of a test case based on a dependency structure:

1. the total number of dependents of the test case, and
2. the longest path of direct and indirect dependents of the test case.

The prioritization for open dependency structure is based on the two measurements DSP Volume and DSP Height.

DSP Volume:

It is a measure which gives a higher weight to those test cases that have more dependents. To calculate the DSP volume of a test case, one need to calculate all direct and indirect dependents of that test case.

DSP Height:

The DSP height measure gives a higher weight to those test cases that have a higher dependents. To calculate the DSP Height of a test case, one needs to calculate the height of all paths form that test case, and take the length of the longest paths as a weight. This can be done using a straight forward depth-first search algorithm on the graph.The Prioritization for closed dependency structure is based on DSP Sum, DSP Ratio, DSP sum / ratio.

DSP Sum:

The DSP Sum coverage measure gives a higher weights to the paths that have more nonexecuted test cases. To calculate the DSP Sum of a path, one simply counts the number of nonexecuted test cases in that path.

DSP Ratio:

The DSP ratio coverage measure gives a higher weight to paths that have a higher ratio of nonexecuted tests to executed tests, while also giving weight to longer paths. To calculate the DSP ratio of a path, one first calculates the weighted sum of the path in which the weight of a test case is its index in the path if it has not been executed, or otherwise, and then divides this by the height of the path.

DSP Sum/Ratio

The DSP sum / ratio coverage is simply the number of nonexecuted test cases divided by the height of the path.

Ranking Algorithm:

The ranking approach to retrieval seems to be more oriented toward these end-users. This approach allows the user to input a simple query such as a sentence or a phrase (no Boolean connectors) and retrieve a list of documents ranked in order of likely relevance.

The main reason the natural language/ranking approach is more effective for end-users is that all the terms in the query are used for retrieval, with the results being ranked based on co-occurrence of query terms, as modified by statistical term-weighting (to be explained later in the chapter). This method eliminates the often-wrong Boolean syntax used by end-users, and provides some results even if a query term is incorrect, that is, it is not the term used in the data, it is misspelled, and so on.

The ranking methodology also works well for the complex queries that may be difficult for end-users to express in Boolean logic. For example, "human factors and/or system performance in medical databases" is difficult for end-users to express in Boolean logic because it contains many high- or medium-frequency words without any clear necessary Boolean syntax.

3. CONCLUSION

The techniques prioritize tests based on the dependency structure of the test suite itself. Six systems being developed toward use in industry are used to empirically assess the strength of these new techniques, measured by the average fault detection rate, in comparison to randomly generated test suites, greedily generated test suites, and the untreated test suite used by test engineers , where available. The results indicate that test suites prioritized of the techniques outperform the random and untreated test suites, but are not as efficient as the greedy test suites. In addition, for open dependency graphs, the techniques achieved better APFDs for most experiments than the state-of-the-art

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

coarse-grained function coverage techniques. For open dependency structures, this improvement was at a greatly lower execution cost. For closed dependency graphs, the techniques achieved better APFDs than total function coverage, and were comparable to additional function coverage. The results indicate that techniques offer a solution to the prioritization problem in the presence of test cases with dependencies. There are two significant strengths of this approach. First, information from previous test runs is not needed to calculate the priorities, so the techniques can be used on first versions of systems. Furthermore, it can be used even if previous test runs have not completed, which is useful in development processes containing short iterations. Second, maintaining fine-grained test suites and prioritizing these based on dependencies preserves the flexibility of fine-grained test suites, while also enabling larger test scenarios to be uncovered, thus increasing the probability of each test finding a fault. The dependency utilization rate towards the application functioning the ranking algorithm is used to measure the functional dependency and precision recall and F-measure is calculated. Functional estimation is based on the no. of accurate function in an application.

REFERENCES

[1] J. Bach, "Useful Features of a Test Automation System (Part iii)," Testing Techniques Newsletter, Oct. 1996.
[2] F. Basanieri, A. Bertolino, and E. Marchetti, "The Cow_Suite Approach to Planning and Deriving Test Suites in UML Projects," Proc. Fifth Int'l Conf. Unified Modeling Language, pp. 275-303, 2002.
[3] S. Elbaum, A. Malishevsky, and G. Rothermel, "Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization," Proc. 23rd Int'l Conf. Software Eng., pp. 329-338, 2001.
[4] S. Elbaum, A.G. Malishevsky, and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies," IEEE Trans. Software Eng., vol. 28, no. 2, pp. 159-182, Feb. 2002.
[5] R.W. Floyd, "Algorithm 97: Shortest Path," Comm. ACM, vol. 5, no. 6, p. 345, June 1962.
[6] D. Jeffrey and N. Gupta, "Experiments with Test Case Prioritization Using Relevant Slices," J. Systems and Software, vol. 81, no. 2, pp. 196-221, 2008.
[7] B. Jiang, Z. Zhang, W. Chan, and T. Tse, "Adaptive Random Test Case Prioritization," Proc. IEEE/ACM Int'l Conf. Automated Software Eng., pp. 233-244, 2009.
[8] J. Kim and D. Bae, "An Approach to Feature Based Modelling by Dependency Alignment for the Maintenance of the Trustworthy System," Proc. 28th Ann. Int'l Computer Software and Applications Conf., pp. 416-423, 2004.
[9] R. Krishnamoorthi and S.A. Sahaaya Arul Mary, "Factor Oriented Requirement Coverage Based System Test Case Prioritization of New and Regression Test Cases," Information and Software Technology, vol. 51, no. 4, pp. 799-808, 2009.
[10] D. Kundu, M. Sarma, D. Samanta, and R. Mall, "System Testing for Object-Oriented Systems with Test Case Prioritization," Software Testing, Verification, and Reliability, vol. 19, no. 4, pp. 97- 333, 2009.