An Online Decision Support Framework for Integration Test Selection and Prioritization (Doctoral Symposium)

Sahar Tahvili
SICS Swedish ICT & Mälardalen University
sahart@sics.se

Under supervision of Markus Bohlin, Daniel Sundmark, Wasif Afzal and Stig Larsson

ABSTRACT

Test case prioritization and selection techniques can lead to early detection of faults and can also enable more efficient usage of testing resources. The available methods of test case selection and prioritization suffer from one or several weaknesses. For example, most of them are only applicable at unit level and do not consider the increasing complexity when subsystems get integrated, especially in the context of embedded system development. Furthermore, the existing methods do not take into account results of current test execution to identify and optimize order for rest of the current execution (i.e., they are not online). In this paper, we propose a tool-supported framework, as an online decision support system (DSF), for prioritizing and selecting integration test cases for embedded system development. DSF provides a complete loop for selecting the best candidate test case for execution based on a finite set of criteria. The results of multiple case studies, done on a train control management subsystem from Bombardier Transportation (BT) in Sweden, demonstrate how our approach helps in a systematic way to select test cases such that it can lead to early detection of faults while respecting various criteria. We are also working towards proposing a customized return on investment (ROI) metric to quantify the economic benefits in optimizing system integration testing using our proposed DSF.

Keywords

Software testing, Integration testing, ROI, Multi-criteria decision making, Fuzzy Logic

1. INTRODUCTION AND MOTIVATION

Software testing processes generally suffer from time and budget limitations. An additional level of complexity is added when testing for integration of subsystems due to inherent functional dependencies and various interactions between integrating subsystems. Therefore improving the testing process, especially at integration level, is beneficial from both product quality and economic perspectives. Towards this goal, application of more efficient testing techniques and tools as well as automating different steps of the testing process (e.g., test case generation, test execution, report generation and analysis, root cause analysis, etc.) can be considered. With respect to test execution, the decision of what test cases to select for execution and the order in which they are executed can also play an important role in efficient use of allocated testing resources.

Numerous techniques for test case selection and prioritization have been proposed in the last decade [1], [2], [3]. Most of the proposed techniques for ordering test cases are offline, meaning that the order is decided before execution while the current execution results do not play a part in prioritizing or selecting test cases to execute. Furthermore, only few of these techniques are multi-objective whereby a reasonable trade-off is reached among multiple, potentially competing, criteria. The number of test cases that are required for testing a system depends on several factors, including the size of the system under test and its complexity. Executing a large number of test cases can be expensive in terms of effort and wall-clock time. Moreover, selecting too few test cases for execution might leave a large number of faults undiscovered. The mentioned limiting factors (allocated budget and time constraints) emphasize the importance of test case prioritization in order to identify test cases that enable earlier detection of faults while respecting such constraints. While this has been the target of test selection and prioritization research for a long time, it is surprising how only few approaches actually take into account the specifics of integration testing, such as dependency information between test cases. Exploiting dependencies in test cases have recently received much attention (See e.g., [4, 5]) but not for test cases written in natural language, which is the only available format of test cases in our context. Furthermore, little research has been done in the context of embedded system development in real, industrial context, where integration of subsystems is one of the most difficult and fault-prone task. Lastly, managing the complexity of integration testing requires *online* decision support for test professionals as well as trading between multiple criteria; incorporating such aspects in a tool or a framework is lacking in current research. In this paper, we propose an *online* multi-criteria decision support framework (DSF) for prioritizing and selecting integration test cases in the context of embedded system development. We discuss the challenges that exist in developing such a framework and the work done so far in this direction.

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

2. PROBLEM FORMULATION & CONTRIBUTIONS

The lack of a systematic approach to decision making might lead to a non-optimal usage of resources. Nowadays, the real-world decision making problems are multiple criteria, complex, large scale and generally consist of uncertainty and vagueness. Such properties are true for software testing in general and integration testing in particular. The current line of research is the application of multiple-criteria decision making (MCDM) to improve integration testing of embedded systems. The goal of this research is to help testers, dynamically prioritize and select test cases for execution at integration testing. This will be achieved by a systematic, multi-criteria design making approach for integration testing that will be empirically validated by industrial case studies at BT. We are looking to address the following research questions in this research:

RQ1: How can the dependency information between test cases be used, together with multiple other criteria, to reach an *online* prioritization and selection of test cases at integration testing of embedded systems?

RQ2: What is the effectiveness (both in terms of cost and quality) of using multi-criteria decision making techniques for prioritizing and selecting test cases for integration testing of embedded systems?

The main contributions of the research direction, both those that have been achieved so far and those that are expected, are the following:

- We provide a novel, multi-criteria test case prioritization method for integration testing [6]. Given the uncertainty and vagueness of the problem domain, two MCDM methods (Fuzzy Analytic Hierarchy Process (FAHP)) and The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)) are applied to relax the need for having precise values for different criteria [7, 6, 8]. Using the concept of fault failure rate [9] as an indicator, it is also shown that the testers are able to detect the hidden faults earlier [8].

- We expect to provide a semi-automated, tool-supported framework as an *online* decision support system (DSF), which is based on MCDM techniques, for selection and prioritization of the best candidates (test cases) for execution. The architecture of the proposed DSF is initially given in [6] and consist of offline and *online* phases. The offline phase assigns a dependency degree to each test case and prioritizes the test cases with same dependency degree using the MCDM technique. The *online* phase dynamically selects test cases to execute based on the execution results of test cases [6]. A slightly modified and more user-friendly architecture of the proposed DSF is shown later in this paper as Figure 1 and discussed in Section 4.

- We are also working towards proposing and validating a parametric cost estimation model for the DSF. We plan to analyze cost factors implied in the integration testing process. This is expected to provide a measure of the value of Return on Investment (ROI) in using the DSF that will further be empirically validated through industrial case studies at BT [10].

3. RESEARCH METHODOLOGY

Until now, we have applied a case study approach [11] as the research method of choice. We intend to continue with conducting further industrial case studies at BT. Once the proposed DSF gets implemented, controlled experimentation [12] can also be used as an alternative research method to more rigorously compare the gains in efficiency and effectiveness with e.g., the current execution order of test cases at BT. We started our research with pre-studying and also trying to identify real optimization problems at the level of integration testing at BT. During the pre-studies, the lack of a dynamic testing schedule, was identified as one of the key opportunities to improve the integration testing process. We then abstracted the problem as a multi criteria, multi-objective optimization problem. During the course of our research, several data collection methods have been used. These consists of interviews, questionnaires and analysis of test archival data in the local database system at BT. The analysis of the test archival data was required to build our understanding of the current state of practice of integration testing at BT. At other times, interviews were required from several test experts to clarify misunderstandings and to narrow down the scope of research. Furthermore, since integration testing spans across departmental boundaries at BT, interviews were used also as a method to identify critical and right resources to contact for further information. Also since we wish to quantify the cost benefits in introducing the DSF, such information can only be asked from senior test managers through interviews. Questionnaires were used to get rating of different criteria from a test expert at BT. A set of linguistic variables (such as low, medium and high) were defined which were used by the expert to rank test cases for each criterion.

4. APPROACH AND CHALLENGES

The research conducted so far is expected to result in an *online* decision support framework for integration test selection. We use the term 'framework' to emphasize that it can be instantiated in contexts similar to ours, i.e., test cases written in natural language, meant for testing of integration of subsystems in embedded system development. Here we describe the several steps (also shown in Figure 1) that we believe would eventually result in a semi-automated decision support framework for optimizing integration test selection. We also highlight important research challenges in implementing these steps.

- **Step 1:** This step consists of newly developed test cases that become part of the *test pool*. The test pool receives a continuous inflow of test cases, as they get developed. It is assumed that these test cases are not ordered and are thus candidates for prioritization.

- **Step 2:** One of the challenges in our current line of research is to devise methods for automatic measurement of different criteria (such as execution time, dependency and requirements coverage) that have an impact on execution order of test cases. *Criteria determination* in Figure 1 consists of various algorithms for measuring such criteria. For example, a counting algorithm counts the number of steps in a test case that can act as a surrogate measure for the test case execution time. The requirements coverage information

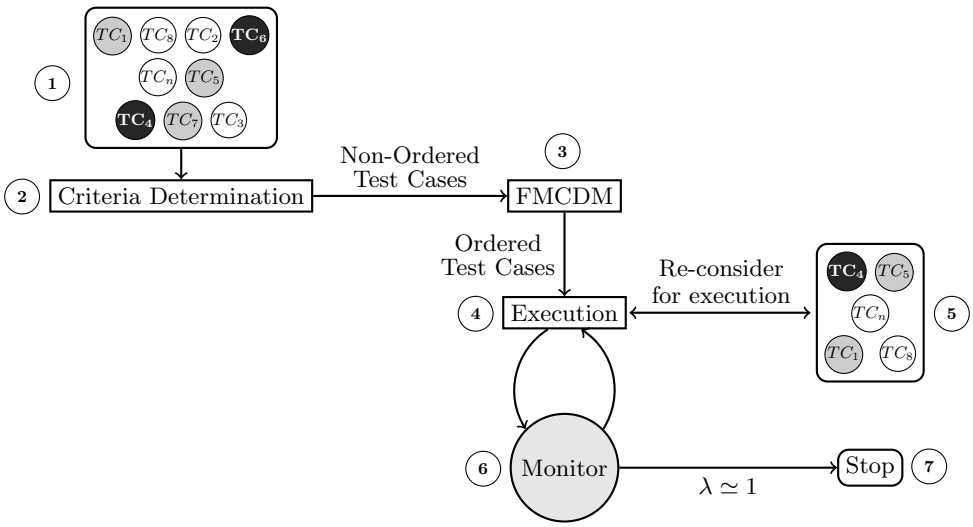

Figure 1: The steps of the proposed online DSF

per test case at BT is available in DOORS[1] that can be automatically extracted. As described in Section 1, the dependencies between test cases is the most critical criterion at integration testing. Previously in [6], we used questionnaires whereby dependencies were manually identified by a test expert at BT. We also defined the concept of *dependency degree* in [6], where test cases with the minimum degree of dependency should be executed earlier. The different colors for test cases in Figure 1 represent such classification of test cases based on dependency. To reduce the manual effort in identification of dependencies, we plan to investigate dependency analysis using an English Natural Language Processing parser (such as in [13]).

- **Step 3:** FMCDM stands for fuzzy multi-criteria decision making technique such as FAHP and TOPSIS. In this step, we prioritize test cases based on the selection criteria. In some cases, we can have just one criterion as the most critical one or we can have same values for other criteria. FMCDM techniques are able to solve both situations. In this step, the test cases having the same dependency degree are prioritized based on other criteria and sent for execution in step 4.

- **Step 4:** Test cases are executed in this step according to the recommended order. The result of executing each test case could either be *Pass* or *Fail*. We have previously in [6] shown that by detecting the dependency between test cases, we are able to avoid the redundant executions. When a test case fails during the execution, all its dependent test cases will be disabled for execution.

- **Step 5:** The failed test cases from the previous step enters a queue for troubleshooting.The failed test cases will be reconsidered for execution once the reason of their failure is resolved. There could be multiple reasons for a test case failure such as identification of a fault, changes in requirements or implementation, change in test case specification, etc.

- **Step 6:** In this step, after the execution of each test case and based on its result, the *executability condition* (see [6]) of the test cases that are dependent on it are (re-) evaluated. In selecting test cases from the ordered set of test cases at each *dependency degree*, if the *executability condition* of a test case is false, it will be skipped and not selected for execution. Furthermore, the completeness of the testing process will be monitored through the metric *fault failure rate* (see [14]), denoted by λ in Figure 1. This metric is simply the proportion of the failed test cases to the total number of executed test cases. The goal is to reach successful execution of maximum test cases to be able to finish the current test execution cycle.

- **Step 7:** The current test execution cycle will stop once the fault failure rate approaches close to 1.

Another research challenge is to motivate the cost effectiveness of introducing the proposed DSF. Implementing and executing the proposed DSF will consume resources. Also there will be situations where it will not be cost-effective to run this framework, e.g., when the number of test cases to execute are small. We are in the process of modeling various cost elements to come up with a customized ROI metric. The metric is supposed to argue in favor or not of instantiating this framework for a particular situation. One can also think of integrating such ROI calculation as an external component to the existing framework to further assist decision making for test professionals.

5. RELATED WORK

Previous researchers have used dependency information to prioritize, select and minimize test suites while others have used fuzzy computation approaches for multi-faceted test case fitness evaluation, prioritization and selection. However, combining these two aspects, in an unified framework has not been addressed before. While a thorough related

[1]Dynamic Object Oriented Requirements System

work is presented in our previous work [6], we mention some representative papers below. Bates and Horwitz [15] use program dependence graphs and slicing with test adequacy data to identify components of the modified program that can be tested using files from the old test suite and to avoid unproductive retesting of unaffected components. Rothermel and Harrold [16] also used slicing with program dependence graph to identify changed def.-use pairs for regression testing. Our approach uses a black-box approach in the sense that it is independent of the source-code modifications. We do not have access to implementation details of functions which is realistic for testing at higher levels. Also we do not address regression testing in particular. Arlt et al. [17] use logical dependencies between requirements written in a structured format to automatically detect a redundant test case. Their approach is essentially a test suite reduction technique based on the current status of successful tests and failed tests. While being similar in purpose, our proposed framework is able to recommend priority even after the identification of dependencies. Malz et al. [18] combine software agents and fuzzy logic for automated prioritization of system test cases. Xu et al. [19] use a fuzzy expert system to prioritize test cases based on knowledge represented by customer profile, analysis of past test case results, system failure rate, and change in architecture. A similar approach is used by Hettiarachchi et al. [20] where requirements risk indicators such as requirements modification status, complexity, security, and size of the software requirements are used in a fuzzy expert system to prioritize test cases. Not many studies quantify the cost-effectiveness of introducing a new test technique or a process improvement approach. Nikolik [21] proposes a set of economic metrics such as test case cost and return on testing investment. A test cost model to compare regression test strategies is also presented by Leung and White [22]. Although these studies are useful, we need to develop a customized ROI metric that takes into account specific costs related to our proposed framework.

6. CONCLUDING REMARKS

This paper presented the concept of a tool-supported framework, as an *online* decision support system (DSF), for prioritizing and selecting integration test cases for embedded system development. The results of the BT case studies show that the concept of multi-criteria test case selection and prioritization is applicable and can reduce test execution time. When the testers did not follow and consider test case dependency relations, some test cases were selected which failed due to the failure of test cases they were depending on. On the other hand, e.g., a test case with higher probability of fault detection, should be executed earlier. Consequently, using the proposed DSF will enable higher test execution efficiency, earlier fault detection and also achieving a positive value for ROI.

7. REFERENCES

[1] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*, 2012.

[2] C. Catal and D. Mishra. Test case prioritization: A systematic mapping study. *Soft. Qual. Journal*, 2013.

[3] E. Engström, P. Runeson, and M. Skoglund. A systematic review on regression test selection techniques. *Info and Software Technology*, 2010.

[4] J. Bell. Detecting, isolating, and enforcing dependencies among and within test cases. In *22nd International Symp. on Foundations of Software Engineering*, USA, 2014.

[5] S. Zhang, D. Jalali, J. Wuttke, K. Mucslu, W. Lam, M. Ernst, and D. Notkin. Empirically revisiting the test independence assumption. In *International Symposium on Software Testing and Analysis*, 2014.

[6] S. Tahvili, M. Saadatmand, S. Larsson, W. Afzal, M. Bohlin, and D. Sundmark. Dynamic integration test selection based on test case dependencies. In *11th Workshop on Testing: Academia-Industry Collaboration, Practice and Research Techniques*, 2016.

[7] S. Tahvili, M. Saadatmand, and M. Bohlin. Multi-criteria test case prioritization using fuzzy analytic hierarchy process. In *10th International Conf. on Software Engineering Advances*, 2015.

[8] S. Tahvili, W. Afzal, M. Saadatmand, M. Bohlin, D. Sundmark, and S. Larsson. *Information Technology: New Generations: 13th International Conf. on Information Technology*, chapter Towards Earlier Fault Detection by Value-Driven Prioritization of Test Cases Using Fuzzy TOPSIS. 2016.

[9] V. Debroy and W. Wong. On the estimation of adequate test set size using fault failure rates. *Journal of Systems and Software*, 2011.

[10] S. Tahvili, M. Bohlin, M. Saadatmand, S. Larsson, W. Afzal, and D. Sundmark. Cost-benefit analysis of using dependency knowledge at integration testing. In *PROFES 2016, under submission.*

[11] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 2009.

[12] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: An introduction.* 2000.

[13] M. Marneffe and C. Manning. *Stanford typed dependencies manual.* The Stanford NLP Group.

[14] V. Debroy and W. Wong. On the estimation of adequate test set size using fault failure rates. *Journal of System and Software*, 2011.

[15] S. Bates and S. Horwitz. Incremental program testing using program dependence graphs. In *Symposium on Principles of Programming Languages.* ACM, 1993.

[16] G. Rothermel and M. Harrold. Selecting tests and identifying test coverage requirements for modified software. In *International Symposium on Software Testing and Analysis*, 1994.

[17] S. Arlt, T. Morciniec, A. Podelski, and S. Wagner. If A fails, can B still succeed? Inferring dependencies between test results in automotive system testing. In *8th IEEE International Conf. on Software Testing, Verification and Validation*, 2015.

[18] C. Malz, N. Jazdi, and P. Gohner. Prioritization of test cases using software agents and fuzzy logic. In *5th International Conf. on Software Testing, Verification and Validation*, 2012.

[19] X. Zhiwei, G. Kehan, T. Khoshgoftaar, and N. Seliya. System regression test planning with a fuzzy expert system. *Inf. Sciences*, 259, 2014.

[20] C. Hettiarachchi, H. Do, and B. Choi. Risk-based test case prioritization using a fuzzy expert system. *Information and Software Technology*, 2016.

[21] B Nikolik. Software quality assurance economics. *Information and Software Technology*, 2012.

[22] H. Leung and L. White. A cost model to compare regression test strategies. In *Conf. on Software Maintenance*, 1991.