Information and Software Technology 69 (2016) 71–83

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Requirements based test prioritization using risk factors: An industrial study

Hema Srikanth[a], Charitha Hettiarachchi[b], Hyunsook Do[c,*]

[a] IBM Product Strategy, Waltham, USA
[b] Computer Science, North Dakota State University, Fargo, ND, USA
[c] Computer Science and Engineering, University of North Texas, Denton, TX, USA

ARTICLE INFO

Article history:
Received 25 August 2014
Revised 17 September 2015
Accepted 19 September 2015
Available online 26 September 2015

Keywords:
Software testing
System testing
Test prioritization
Cloud application
SaaS

ABSTRACT

Context: Software testing is an expensive and time-consuming process. Software engineering teams are often forced to terminate their testing efforts due to budgetary and time constraints, which inevitably lead to long term issues with quality and customer satisfaction. Test case prioritization (TCP) has shown to improve test effectiveness.
Objective: The results of our prior work on requirements-based test prioritization showed improved rate of fault detection on industrial projects; the customer priority (CP) and the fault proneness (FP) were the biggest contributing factors to test effectiveness. The objective of this paper is to further investigate these two factors and apply prioritization based on these factors in a different domain: an enterprise level cloud application. We aim to provide an effective prioritization scheme that practitioners can implement with minimum effort. The other objective is to compare the results and the benefits of these two factors with two risk-based prioritization approaches that extract risks from the system requirements categories.
Method: Our approach involved analyzing and assigning values to each requirement based on two important factors, CP and FP, so that the test cases for high-value requirements are prioritized earlier for execution. We also proposed two requirements-based TCP approaches that use risk information of the system.
Results: Our results indicate that the use of CP and FP can improve the effectiveness of TCP. The results also show that the risk-based prioritization can be effective in improving the TCP.
Conclusion: We performed an experiment on an enterprise cloud application to measure the fault detection rate of different test suites that are prioritized based on CP, FP, and risks. The results depict that all approaches outperform the random prioritization approach, which is prevalent in the industry. Furthermore, the proposed approaches can easily be used in the industry to address the schedule and budget constraints at the testing phase.

1. Introduction

Software testing can often be a tedious and expensive process [1,2], and can often add up to 50% of the total software cost [3,4]. With limited time and resources, companies are often unable to complete their testing efforts, resulting in software that might not meet customer needs. In order to facilitate effective testing, the concept of test case prioritization (TCP) is often applied to the suite of test cases such that the test cases are run in an order that improves the rate of fault detection [5–7]. By improving the rate of fault detection, the testing teams can mitigate many of the testing issues by reducing the time and cost associated with testing.

To date, the research conducted in TCP has primarily focused on improving regression testing efforts using white box, code level and coverage-based approaches [5,6,8]. Regression testing allows software engineers to test changes made to the system to ensure the code changes made do not introduce new faults in the software system [9]. Regression testing is a necessary and important maintenance activity for improving software quality, but it can require a substantial amount of time and effort as software systems and numbers of test cases grow. While TCP techniques help address this problem by identifying important test cases to run earlier, the majority of them use code coverage information [10], which can be expensive for practitioners to apply [11]. Further, a software system is built upon its requirements, so utilizing requirements information could potentially

* Corresponding author. Tel.: +1 9405654864.
E-mail addresses: srikanth_h@us.ibm.com (H. Srikanth), charitha.hettiarachc@ndsu.edu (C. Hettiarachchi), hyunsook.do@unt.edu, dohyunsook@yahoo.com (H. Do).

http://dx.doi.org/10.1016/j.infsof.2015.09.002
0950-5849/© 2015 Elsevier B.V. All rights reserved.

help identify more important or more error-prone test cases than just using source code information.

To address this problem, in our prior work, we introduced a prioritization of requirements for test (PORT) 1.0 scheme where we showed the efficacy of TCP at the system level by considering four factors for each requirement [13]. Test cases were prioritized based on the priority of the requirement that was derived by assessing the four factors, customer priority (CP), implementation complexity (IC), fault proneness (FP), and requirements volatility (RV) [13] for each requirement. Test cases that map to requirements with higher priority were ordered earlier for execution. We demonstrated the efficacy of PORT 1.0 technique on four large industrial projects to show the improved rate of fault detection, and thus test effectiveness [12,14]. From our prior study, with extensive sensitivity analysis, we learned that CP was the biggest contributor to improving the rate of fault detection [14]. These findings can be useful for the practitioners when they have limited time and resources to execute the entire tests during regression testing, but these studies were applied to the projects under the same application domain, so whether these results generalize to other application domains is an open question. Further, PORT 1.0 was applied mostly to software applications supporting hardware that usually have a longer release cycle (products having yearly releases). These software applications usually follow software process similar to waterfall model and tend to have a longer release cycle. While PORT 1.0 was validated in industrial projects on a hardware-centric domain, PORT 2.0 is validated on an enterprise cloud application for analytics that has customers around the globe for several years.

Our goal in this paper is to present PORT 2.0 where we apply only a set of factors toward prioritization. In our prior study, we found CP and FP as most significant factors; thus we share the results of PORT 2.0, which uses only these two factors for prioritization of tests. Additionally we validate the approach on software as a service application that follows a very iterative software process where release cycles are as frequent as monthly. In addition to utilizing these two factors, we also investigate whether the use of risk information extracted from the system can improve the effectiveness of test case prioritization. In this paper, we show the application of these prioritization techniques on an enterprise-level software system as a service (SaaS) application. The software application, which has several million lines of code, is an enterprise level marketing analysis system that has thousands of customers around the globe. The product team for this application is spread across five geographical locations with thousands of use cases being used by customers every day.

The contributions of this paper include development and validation of two requirements-based prioritization approaches and the validation on an enterprise-level cloud application. Our results indicate that the use of CP and FP can improve the effectiveness of test case prioritization. The results also show that the risk-based prioritization can be effective in improving the test case prioritization. Further, we found that there are some cost-benefit tradeoffs among these approaches, thus we believe that the findings from this study can help practitioners to select an appropriate technique under their specific testing environments and circumstances.

The remainder of the paper is structured as follows. Section 2 describes the research motivation and *two* different requirements-based test prioritization approaches. Sections 3 and 4 present our experiment including research questions, experiment setup, and analysis. Section 5 discusses the results of our study, and Section 6 describes background information and the related work. Finally, Section 7 presents conclusions and future work.

2. Research motivation and approach

In this section, we discuss our research motivation and then describe two proposed requirements-based approaches: PORT 2.0 that applies two factors CP and FP, toward prioritizing test cases, and a risk-based prioritization technique that prioritizes based on risks associated with requirements categories. To facilitate these approaches, we had developed a tool called requirements based testing (ReBaTe) and we provide an overview of ReBaTe tool architecture in Section 2.3.2.

2.1. Research motivation

Although SaaS has gained acceptance in the past years, it presents the community with a new set of software engineering challenges. Because SaaS is delivered to the customer on cloud, there is one version hosted online for all customers. Thus, to remain competitive, the vendors have to make frequent software updates as often as every two weeks, and maintain high software quality and reliability standards. While SaaS has benefits for customers, quality management has been a major challenge for providers. The system has to be available and functional 24 hours seven days a week. Because all customers are using one version of software, the impact of faults in the system is amplified and there is a need to minimize escape of faults to the field. Finally, frequent updates and software releases provide very little time to testing teams to effectively complete testing efforts, and so there is a need to have effective regression testing techniques. The test case prioritization techniques add much more value to the applications on cloud because of the timeliness involved in releasing updates and the need to ensure quality because of the broader impact of a single failure to the entire customer base. These issues make the regression testing efforts very difficult and there is a dire need for test case prioritization techniques.

In our prior validation of PORT 1.0 on industrial applications [12], we observed that CP was the highest contributing factor toward improved test effectiveness. Based on our prior research we found that these factors contributed toward quality and therefore were selected in PORT 1.0 [13,14]. Our motivation toward selecting these requirements engineering factors is discussed in detail in [13,14]. In this paper, we extend our prior study to PORT 2.0 that applies prioritization only on two factors – CP and FP, both are critical in understanding customer usage of the applications. It is imperative to present an approach that practitioners can find easy to use and apply with minimal effort, and also achieve testing effectiveness. In addition to these two factors, risk information associated with requirements could provide a means to identify important test cases that can reveal highly risky defects. By identifying and fixing such defects earlier, we can further speed up the regression testing process and release more reliable products. With these motivations, in this work, we investigate whether these factors can indeed improve the effectiveness of test case prioritization. We believe that our research outcomes can help practitioners apply this technique without requiring advanced mathematics or statistics, and with minimal effort of collecting and analyzing these factors in the test planning phase.

2.2. Prioritization strategies

In this work, we consider three PORT-based prioritization approaches and two risk-based approaches. The following subsections describe them in detail.

2.2.1. PORT-based prioritization

Our prior contribution to the test case prioritization problem was the development of PORT 1.0 [12,13,14]. Evaluation of PORT 1.0 enabled the software engineering team to assign values to four critical factors: customer priority (CP), fault proneness (FP), requirements volatility (RV), and implementation complexity (IC). The selection of the PORT factors was based on prior research and a more thorough discussion of the factors and their justifications can be found in our previous work [12,14]. In our prior work, we validated PORT 1.0 on