J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

Supporting Systematic Software Test Process in R&D Project with Behavioral Models☆

Hyorin Choi[1] Jung-Won Lee[2] Byungjeong Lee[1*]

ABSTRACT

Various artifacts that are produced as software R&D project progresses contain research plan, research report, software requirements and design descriptions, etc. When conducting a software R&D project, it is necessary to confirm that the developed system has implemented its research requirements well. However, various research results make it difficult to design appropriate tests. So, there is a practical need for us to comprehensively handle the planning, execution, and reporting of software test for finding and verifying information related to the research. In this paper, we propose a useful method for software test process in R&D project which supports model based software testing. The proposed method supports automation of test design and generation of test data by explicitly separating each step of System Under Test (SUT). The method utilizes the various models representing the control flow of the function to extract the information necessary for testing the system. And it supports a systematic testing process based on TMMi and ISO 29119. Finally, we show the validity of the method by implementing a prototype with basic functionality to generate test data from software behavioral models.

☞ keyword : Software R&D Project, Test Support Tool, Model based Testing

1. Introduction

Various artifacts that are produced as software R&D project progresses contain research plan, research report, software requirements and design descriptions, etc. However, it is difficult to design appropriate software tests by analyzing a huge amount of R&D outcomes. It is necesaary to efficiently verify the requirements of software produced in R&D project. Thus, in this study we propose a method for supporting software test process including test planning, test execution, and test reporting, based on model based test[1] to overcome this diffculty in software R&D project. Also, we introduce a prototype that implements basic functionnality of the method to show the validity of the method. The proposed method supports automation of test design and generation of test data by explicity separating each step of System Under Test(SUT). The method utilizes the various models representing the control flow of the function to extract the information necessary for testing the system. And it supports a systematic testing process based on TMMi and ISO 29119. The contribution of this study is as follows.

- We propose a method to efficiently support software test process in R&D project.
- By automating test design and data generation, the cost of performing software test is reduced.

Section 2 describes related studies, and Section 3 outlines our method to support software test and describes the test artifacts produced in software R&D proejct. Section 4 introduces a prototype system to show our method. Finally we conclude in Section 5.

1 Department of Computer Science, University of Seoul(Seoul), 02504, South Korea
2 Department of Electrical and Computer Engineering, Ajou University(Suwon), 16499, South Korea
* Corresponding author: Byungjeong Lee
[Received 16 October 2017, Reviewed 6 November 2016(R2 4 January 2018), Accepted 5 February 2018]
☆ This research was supported by Next-Generation Information Computing Development Program (NRF-2014M3C4A7030504) and by Basic Science Research Program (NRF-2017R1A2 B4009937) through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning.
☆ A preliminary version of this paper was presented at APIC-IST 2017 and was selected as an outstanding paper.

2. Related Work

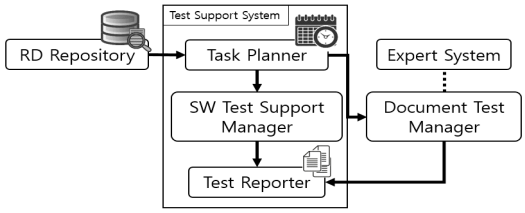
Many studies have been conducted on how to perform test process through tool support for systematic testing of the developed system. B. N. Nguyen et al.[2] proposed 'GUITAR'is a tool that supports test automation based on a model that expresses the structure of all graphical interfaces (GUIs) for users to operate the system. It is effective in reducing the test cost in that it automatically tests all GUIs that can be operated according to the given model. However, this method cannot be used on a system without a GUI, and there is a limitaion in that it is difficult to design a test for detailed functions of the system such as a unit level test[1]. The 'PLeTsPerf'tool, proposed by E. Rodrigues et al.[3], assists developers in their testing activities by automatically generating scenarios and data for the overall test execution of the developed system. It also designs the test based on model information, conducting pilot studies to define the necessary information and to show an example. This method is similar to our method in that it supports test design process based on model information. However, it designs test scenarios based on control activities of the researcher and does not support automated generation of the test data. On the other hand, there are related studies in terms of tools supporting software test standards and processes for satisfying TMMi maturity level [4-8]. K. Jin et al.[4] defined the criteria for writing test plans and reports according to the software development life-cycle. However, this method has limitations in that it is difficult to specify the development artifacts necessary for performing the test and does not deal with test data generation. In this study, we extended the method of [4][6] and defined it according to operating flow of our system. S. Back et al.[5] proposed a method for generating test data through a meta-heuristic algorithm based on UML behavioral model. Our prototype extended the method of [5,6]. The studies [7,8] presented a process to verify documents and program in software R & D project. However, they did not provide automation of test design and generation of test data.

3. Supporting Systematic Software Test

3.1 Overview

Software testing is a activity of ensuring that the functions implemented in software work as designed. The test is performed by executing all test cases on the actual system and checking whether the expected results are obtained. As the scale of the developed system increases, the cost of manually designing tests for all functionalities implemented by developers is very high[9]. In particular, systems developed in R&D project often require high reliability. However, domain experts have difficultis in obtaining adequate expertise for software testing[10]. Since this means a high increase in test cost, tools to support systematic test are required to reduce the cost.

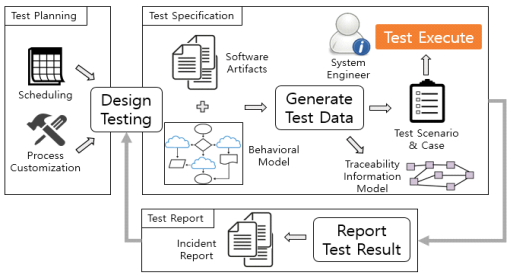
In this study, we propose a system that supports systematic software test process based on standards such as IEC 9126 and ISO 29119-2 in order to satisfy this requirement. The proposed system is a component system that supports software test in test framework of [11]. Fig. 1 illustrates relationships between other constituent system and our system. Other systems involved are: RD Repository that stores, resuses and manages all researchdocuments(RD) produced during R&D projects, an expert system which judges whether the research documents are wrritten well according to the quality management standard such as ISO 26262, and document test manager which uses the expert system to constantly verify the quality of research documents.



(Figure 1) System context

3.2 System Flow

The control flow of our system is as follows. It has three phases : Test Planning, Test Specification, and Test Report.



(Figure 2) System flow

Fig. 2 shows the technologies implemented in our system according to the flow.

The different operations are performed at each phase and the various artifacts are produced. This is described in the following.

Phase 1: Test Planning

First, in Test Planning stage, task setting, strategy formulation, and scheduling for software test are performed. A list of tasks to be performed is chosen according to the level of TMMi required for each project by applying a process customization technique[12] based on ISO 29119 standard that specifies the work of the test process. Each task of the list should be specified with the person responsible for the task and the duration. The system shows the tasks in the form of gantt chart. The outcomes produced through these operations are maintained in each entry of SLTP(Software Life-cycle Test Plan), which is described in Section 3.3.

Phase 2: Test Specification

When the previous test planning is completed, the test specification stage proceeds. In this stage, it collects all the information needed for the test design such as finding which functional items to test, and identifying related outcomes based on generated test plan. Collected information includes behavioral models expressing control flow of system. Our method automatically generates scenarios and cases that the researcher actually needs to test the system according to the test coverage. In addition, the relationships between these development products, behavioral models, and test data are maintained in a traceability information model. The researchers use the data to execute the test. The necessary information including relevant RD document, expected test function, test goal, and strategy is provided through the traceability information model. SLTS(Software Life-cycle Test Specification) written in Phase 2 is based on the generated test data, traceability information, and behavioral models.

Phase 3: Test Report

The researcher performs the test by using the SLTS and obtains the test results. SLTR(Software Life-cycle Test Report) is generated by using the results. SLTR helps project manager to make a decision by recording test data from SLTS, the person in charge from SLTP, the result data, etc. In particular, failed test cases will be written in Incident Report, which is used as a data to judge the reason of failure, type of defect, severity, etc. by utilizing the description of the researcher. Also, since failed test cases mean that there are defects in the system, they are considered again in the test specification phase to deal with the defects through feedback.

3.3 Test Artifacts

Table 1 summarizes test artifacts of each phase defined in our method. SLTP in Phase 1 is a test plan according to software development cycle. Target component in SLTP has the functionalities to be tested. Task List is a list of tasks obtained after customizing test process, and Test Strategy includes overall test conditions such as functional constraints to be satisfied, tools to be used, and work environment, etc. Test Goal expresses quantitatively how much to perform the test, such as the quality level to be satisfied.

(Table 1) Test artifacts

Phase	Artifacts		
1	Target Component	Task List	Test Strategy
	Test Goal	Member list	Schedule
2	Model Information	Established Strategy	Condition
	Test Data	Result	
3	Related Artifacts	Evaluation	Progress
	Last Updated	Manager Information	Comments

SLTS in Phase 2 is a specification written according to SLTP. It includes test data generated from the SUT's behavioral models and developmental artifacts, the information of the relevant model, and test strategy set in SLTP. Condition is a constraint to be observed. Result is a result of applying the test data. SLTR in Phase 3 is a report on test results. SLTP and SLTS in the previous phases are Related Artifacts. Total result represents success or failure, and unexecuted test cases, and Progress describes how further the tests should be performed within test scope. Each time the result is updated in SLTS, each attribute value of SLTR must be changed, so record the update date and its history. If tester creates a comment about tests they perform, it can be output to the Incident Report form. Our system uses the three mentioned artifacts to keep the information generated in each operating stage in a database. SLTP and SLTR are items that are generated one by one for each test cycle in the development project, and SLTS can be variously generated due to the function of the system under test, the target coverage, and so on.

3.4 Comparisons

This section provides comparisons of this study with other studies that support model-based testing. The comparisons are performed by analyzing the following 4 Research Questions(RQs).

RQ1. Does the system automatically generate test data?
RQ2. Is it independent of development language?
RQ3. Has this system built an integrated R&D testing process?
RQ4. Does this system perform systematic support to comply with test standards?

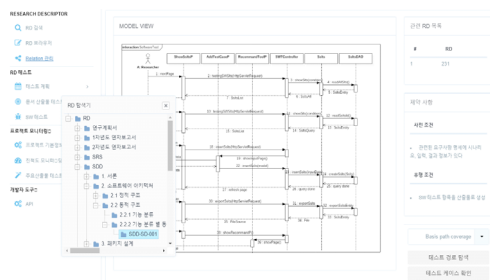
(Table 2) Comparisons

	GUITAR [2]	PLeTsPerf [3]	MOS [12]	TCG [13]	Our work
RQ1	P	Y	Y	Y	Y
RQ2	Y	Y	Y	Y	Y
RQ3	N	N	P	P	Y
RQ4	Y	N	N	N	Y

The studies for comparisons are 'GUITAR[2]' and 'PLeTsPerf[3]' introduced in Section 2. In addtion, 'MOS[13]' applying the search-based testing technique to the Model based test(MBT) and 'TCG[14]'applying the static analysis technique are used. The notation used by Shafique[15] was applied as the analysis result for each RQ. The results of the comparions are summarized in Table 2. Most studies have 'Y'(Support) in RQ1 and RQ2. This is because the purpose of applying model-based testing is test automation and they are independent of development language by using a freely expressible model for the system in the test design. However, GUITAR[2] has 'P'(Partially Support) in RQ1 because it performs test design based on GUI. It is impossible to generate test data of the system without GUI. RQ3 and RQ4 are the strengths of this system. Our method provides functions to comply with the integrated test process according to research and development procedures. And each task of the process is based on ISO 29119 and TMMi standard. GUITAR[2] and PLeTsPerf[3] have 'N'(Not Support) for RQ3, but MOS[13] and TCG[14] support some of the testing process by supporting test plans and automated design features. In case of RQ4, GUITAR[2] complies with the standard workflow of the GUI test.

4. Prototype

In this section, we introduce a prototype of our system. The prototype was developed using Java and run on the web by using Apache Tomcat, a web hosting service. Also, software requirement specification(SRS) and software design description(SDD) based on UML models are used to generate test data. Fig. 3 shows an example page of the prototype. Details can be found in our website URL : http://se.uos.ac.kr/s2/swtest. When developing the prototype, we used open source libraries such as Spring Framework.



(Figure 3) A prototype

5. Conclusion

In this paper, we proposed a method to support systematic software test process that maintains relationships between artifacts of software R&D project and automatically generates test data based on UML models. We implemented it as a prototype and showed the practical validity of the proposed method. In the future, we plan to expand test artifacts and domain standards to apply our method to projects in various domains. We will also continue to add functionalities to increase usability of our system.

References

[1] M. Utting, and B. Legeard, Practical Model -Based Testing: A Tools Approach, Morgan Kaufmann, 2010.

[2] B. N. Nguyen, B. Robbins, I. Banerjee, and A. Memon, "GUITAR: an innovative tool for automated testing of GUI-driven software," Automated Software Engineering, vol. 21, no.1, pp, 65-105, 2014.

[3] E. Rodrigues, M. Bernardino, L. Costa, A. Zorzo, and F. Oliveira, "PLeTsPerf - A Model-Based Performance Testing Tool," In Proc. Of IEEE International Conference on Software Testing, Verification and Validation, pp. 1-8, 2015. https://doi.org/10.1109/ICST.2015.7102628

[4] K. Jin, S. Song, J. Lee, B. Lee, "Test Planning and Reporting for Constant Monitoring of Software R&D Projects," In Proc. of Korea Computer Congress, pp. 597-599, 2015.

[5] S. Back, H. Choi, J. Lee, and B. Lee, "Evolutionary Test Case Generation from UML-Diagram with Concurrency," In Proc. of International Conference on Computer Science and its Applications, LNEE, vol. 421, pp. 674-679, 2016.

[6] H. Choi, J. Lee, and B. Lee, "Towards Supporting Model based Software Test for R&D Project," In Proc. of Asia Pacific Conference of Information Science and Technology, June 2017.

[7] A. Dashbalbar, S. Song, J. Lee, B. Lee, "Towards Enacting a SPEM-based Test Process with Maturity Levels," KSII Transactions on Internet and Information Systems, vol.11, no.2, pp. 1217-1233, Feb. 2017. http://www.dbpia.co.kr/Article/NODE07121666

[8] A. Dashbalbar, E. Lee, J. Lee B. Lee, "Describing Activities to Verify Artifacts(Documents and Program) in Software R&D," Journal of Internet Computing and Services, vo.17, no.2, pp.39-47, Apr. 2016. https://doi.org/10.7472/jksii.2016.17.2.39

[9] D. R. Kuhn, V. Hu, D. F. Ferraiolo, R. N. Kacker, and Y. Lei, "Pseudo-exhaustive testing of attribute based access control rules," In Proc. of Software Testing, Verification and Validation Workshops, pp. 51-58, 2016. https://doi.org/10.1109/ICSTW.2016.35

[10] H. J. Thamhain, "Assessing the effectiveness of quantitative and qualitative methods for R&D project proposal evaluations," Engineering Management Journal, vol. 26, no. 3, pp. 3-12, 2014. https://doi.org/10.1080/10429247.2014.11432015

[11] S. Song, A. Dashbalbar, J. Lee and B. Lee, "Test Framework Requirements to Verify Artifacts in Software R&D Project," International Journal of Software Engineering and Its Applications, vol. 10, no. 11, pp. 83-94, 2016. https://doi.org/10.14257/ijseia.2016.10.11.07

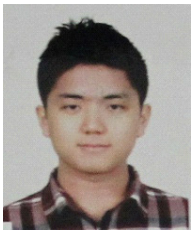
[12] D. Baek, B. Lee, and J. Lee, "Content-based Configuration Management System for Software Research and Development Document Artifacts," KSII Transactions on Internet & Information Systems, vol.10, no.3, pp. 1404-1415, 2016. http://www.dbpia.co.kr/Article/NODE06647459

[13] E. P. Enoiu, K. Doganay, M. Bohlin, D. Sundmark, and P. Pettersson, "MOS: an integrated model-based and search-based testing tool for function blockdiagrams," In Proc. of International Workshop on Combining Modelling and Search-Based Software Engineering, pp. 55-60, 2013. https://doi.org/10.1109/CMSBSE.2013.6605711

[14] L. L. Muniz, U. S. Netto, and P. H. M. Maia, "TCG: A Model-based Testing Tool for Functional and Statistical Testing," In Proc. of International Conference on Enterprise Information Systems, no. 2, pp. 404-411, 2015. https://doi.org/10.5220/0005398604040411

[15] M. Shafique, and L. Yvan, "A systematic review of model based testing tool support," Technical Report SCE-10-04, Carleton University, Canada, 2010.

◐ 저 자 소 개 ◑



Hyorin Choi

2015 B.S. from Korea National University of Transportation, Korea
2017 M.S. from Computer Science at University of Seoul, Korea
Research interests: software engineering, software evolution.
Email : yoinoichr2015@uos.ac.kr

Jung-Won Lee

1993 B.S. from Ewha Womans University, Korea
1995 M.S. in Computer Science from Ewha Womans University, Korea
2013 Ph.D. in Computer Science from Ewha Womans University, Korea
1995～1997 Researcher of LG Electronics, Korea
2006～Present Professor of the Department of Electrical and Computer Engineering at Ajou University, Korea.
Research interests: context-aware, embedded software, software engineering
Email: jungwony@ajou.ac.kr

Byungjeong Lee

1990 B.S. from Seoul National University, Korea
1998 M.S. in Computer Science from Seoul National University, Korea
2002 Ph.D. in Computer Science from Seoul National University, Korea
1990～1998 Researcher of Hyundai Electronics, Korea
2002～Present Professor of the Department of Computer Science and Engineering at the University of Seoul, Korea.
Research interests: software engineering, machine learning
Email: bjlee@uos.ac.kr