ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

소프트웨어 R&D에서 산출물(문서와 프로그램) 검증을 위한 활동☆

Describing Activities to Verify Artifacts(Documents and Program) in Software R&D

아마르멘드[1] 이 은 철[1] 이 정 원[2] 이 병 정[1*]
Amarmend Eun-Chul Lee Jung-Won Lee Byeongjeong Lee

요 약

일반적으로 소프트웨어 R&D 프로젝트에서는 프로그램 코드와 문서 산출물이 생성된다. 이러한 소프트웨어 R&D 산출물들은 두 가지로 분류할 수 있다. 첫 번째 분류는 연차 실적 계획서, 연구개발과제계획서, 연구성과보고서, 연구 노트와 같은 소프트웨어 연구 산출물들이 포함된다. 그리고 다른 분류는 소프트웨어 요구사항 명세서, 소프트웨어 설계 명세서, 소프트웨어 테스트 계획서, 프로그램 코드와 같은 소프트웨어 개발 산출물들이다. 프로젝트의 진행 방향을 확인할 때 프로그램 코드를 테스트하고 문서 산출물을 검증하는 것이 중요하다. 또한 연구 문서와 개발 산출물 사이에 완전성, 일관성 등의 관계를 확인해야 한다. 그러한 검증과 테스트는 프로젝트 관리자와 연구자들이 프로젝트를 진행하는 동안 올바르게 진행하고 있다는 확신을 준다. 그러므로 본 연구에서는 소프트웨어 R&D에서 생성되는 문서와 프로그램을 검증하는 프로세스를 제안한다. 본 프로세스는 문서 산출물을 검토하고 프로그램 코드를 테스트하는 활동으로 구성되어 있으며, Essence를 사용하여 정의된다. 그리고 본 연구에서 제안하는 프로세스의 효율성을 사례 연구를 통해 보인다.

☞ 주제어 : 소프트웨어 연구개발, 검증 활동, Essence, 산출물 테스팅

ABSTRACT

In software R&D artifacts including documents and program code are produced. There are two kinds of software R&D artifacts: Research artifacts and development artifacts. Research artifacts include software R&D planning document, annual report, final report, research note and so on. Software development artifacts include software requirements, software design description, testing plan, test report, and program code. It is important to verify the documents and to test code to check the direction of the R&D. Moreover, we should check relationships as such completeness and consistency between research and development artifacts. The verification and testing help project manager and researchers understand what they do during software projects. Therefore, in this study, we present a process to verify documents and program in software R & D. In the process we check documents produced in software R&D and test program code. We describe the process by using Essence elements including alpha, activity, and competency. We present a case study to show the effectiveness of the process.

☞ keyword : Software R&D, Validation Activities, Essence, Artifact Testing

1. INTRODUCTION

Researchers and developers produce program code and documents in software R&D. They examine documents and code to monitor project progress and quality during software project where they verify both the code and documents. The purpose of this verification is to make contribution to software testing activity. Artifacts of software R&D project are divided into two groups. One is software research documents including R&D planning document, annual report, final report, research note and so on. The other includes software development artifacts including requirement specification, design description, test plan, test report, program code and so on.

Researchers and developers should check the documents and test program code to confirm the direction of the R&D.

[1] Dept. of Computer Science, University of Seoul, Seoul, 20504, Korea.
[2] Dept. of Department of Electrical and Computer Engineering, Ajou University, Suwon, 16499, Korea.
* Corresponding author (bjlee@uos.ac.kr)
[Received 30 November 2015, Reviewed 21 January 2016, Accepted 24 March 2016]
☆ This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014M3C4A7030504), and by Seoul Creative Human Development Program funded by Seoul Metropolitan Government (No.CAC15106).
☆ A preliminary version of this paper was presented at APIC-IST 2015 and was selected as an outstanding paper.

Moreover, they should investigate relationships as such completeness and consistency between research and development artifacts. Thus, they have to be aware of how the verification goes. In this paper, we describe a test process with activities to verify artifacts in software R&D project.

We use Essence framework by Object Management Group for our process. Essence kernel has 6 alphas and every alpha changes its status in case of activities of the alpha had fulfilled. We use high-level test planning, detailed-level test planning, unit-level test planning, unit testing, detailed-level testing and high-level testing as alphas in our process.

The contribution of this paper is that we introduce a test process that conforms to software R&D projects and both software product and documents are able to be tested within the process. Specifications are essential for successful software testing, but other similar studies do not consider correctness of the related documents and do not include any activities on verifying specifications.

Remainder of the paper is structured as follows: Section 2 provides some basic information on relative technologies, Section 3 contains a brief content of related works, Section 4 presents a test process we are introducing, Section 5 describes the association of our process and Essence model and Section 6 illustrates conclusion of this paper and how our study will continue in future.

2. RELATED WORK

2.1 ESSENCE

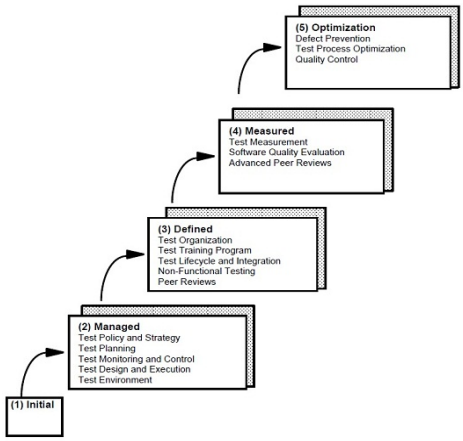
Essence [1-3] is a framework that presents the state of software development progress and provides the common ground for those diverse software engineering methods and theories. Essence mainly consists of four parts such as Methods, Practices, The Kernel and The Language. Method is a set of Practices and it's not just a description of what is expected to be done, but a description of what is actually done. A practice is a repeatable approach to doing something with a specific objective in mind. The Essence Language is the domain-specific language to define methods, practices and kernels. Kernel contains Alphas, Activity Space and Competencies. There are seven alphas in Essence Kernel that are Stakeholders, Opportunity, Requirements, Software System, Team, Way of Working and Work and each alpha changes its state during the progress.

Seven alphas of essence kernel are grouped into three areas of concern. Firstly, Opportunity and Stakeholder alphas are included in Customer area. Solution area contains alphas that Requirement and Software System. Lastly, Work, Team and Way of Working alphas are included in Endeavour area. Every alpha has its own states and those states shows the progress of fulfillment of project development. Alphas contain the things to do or Activity Spaces which need to be done in order to change to next state of alpha. We used essence kernel for test process model in our study and we created whole other new alphas which are essential for our model. There is also "Competencies" or "Abilities Needed" in the kernel, it determines the competencies in each area, for example, Analysis, Development and Testing are the needed abilities of Solution area. Those competencies have five levels that 1-Assists, 2-Applies, 3- Masters, 4-Adapts and 5-Innovates.

2.2 TMM(i)

TMMi is test maturity reference model [4-6] which is developed by TMMi foundation. Its structure is same as Capability Maturity Model (CMM) and concept was firstly introduced in 1996. TMMi was made to improve the testing effectiveness and made it possible for organizations that they determine their fulfillment and effectiveness of their testing. A quality assurance framework is included in TMMi model and it is used for connection that provides information on concept and ideas between workers in large organization.

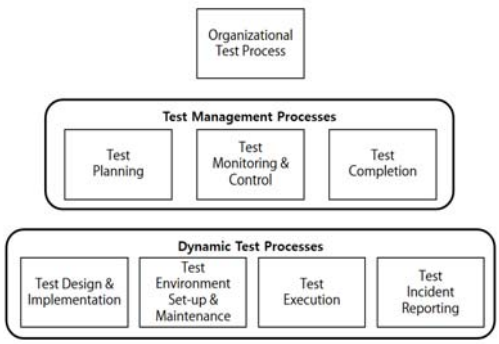
As shown in Figure 1, TMMi has five maturity levels which are Initial, Managed Defined, Measured and Optimization. Each level includes process areas which need to be done in order to advance next level. Process areas contain two kinds of practices which are Specific and Generic practices and both the lowest unit of the TMMi model. Specific practice is included only in one particular process area. Generic process is a process which is connected to two or more process areas so it means several process areas' fulfillment is dependent to one Generic Practice. There are also Specific and Generic goals which indicate the purpose of specific and generic practices and need to be satisfied by those practices when they are done.



(Figure 1) TMMi LEVELS. [7]

2.3 ISO 29119

ISO 29119 is a standard for software testing standardized by International Organization for Standardization in 2013. This standard is consists of 5 parts such as Concepts and Definitions, Test Processes, Test Documentation, Test Techniques and Keyword Driven Testing. This standard was implemented in order to draw the baseline for the testing discipline and settle the conflict in current definition and processes. In our study we use part 2 of the standard, test process which uses risk-based approach. Firstly, the part 2 of ISO 29119 also includes Static Test Process as fourth part. However, in recent update Static Test Process part was removed and the part 2 has become three layered model. The layers with their activities are shown in Figure 2.



(Figure 2) LAYERS AND ACTIVITIES OF ISO 29119-2

The Organizational Test Process layer is divided into two parts: Test policy and Test Strategy parts of Software Testing. Test Management Process layer consists of three activities which are Test Planning, Test Monitoring & Control and Test Completion. In Test Planning activity, the test planning document is generated and sent to Dynamic Test Process layer. While Test Monitoring & Control activity checks the progress of test process by test measures that sent from Dynamic Test Process.

From Organizational Test Process layer Organizational Test Strategy is passed to Test Management Process layer and Test Management Process layer feedback to the Organizational Test Process layer. Thus, Test Management Process layer produces Level Test Plans based on Test Strategy and passes it to the Dynamic Test Process layer with Control Directives. Lastly, Dynamic Test Process layer passes the test measures to Test Management Process layer after testing activities are finished on all levels.

We utilize a study [8] about Test Maturity Models while we study software testing field. TMMi is a test maturity model developed by TMMi foundation for the purpose of determining testing quality by organization itself. We considered TMMi has some similar prospects and so we researched related papers to TMMi. Since 2007, Erik van Veenendaal published number of papers on test process improvement and introduced TMMi as the reference model.

In 2013 Pan-Wei Hg et al. [9] raised a problem by his paper about how the student's knowledge that learned in college differs from employee's requirement. He mentioned in his paper that the lack of framework makes it hard to understand and compare the college education system and industry needs. Pan-Wei Hg suggested Essence can be bridge among the gap between industry, research and education.

B. Elvesæter et al. [10] illustrates Essence Kernel and Language and how they we used in the study. This study shows key language concept difference between SPEM and Essence - which both OMG projects. The comparison is based in REMICS project and result showed in spite of both developed by same author they have some key differences in method. From this paper we have obtained some useful knowledge about Essence 1.0 and SPEM 2.0 and gap between them.

D. J. Han et al. [11] is a study which describes guidelines

for implementing Capability Mature Model Integration (CMMI) based configuration management in Extreme Programming (XP). Therefore, this paper provides basic knowledge on CMMI and Extreme Programming and configuration management (CM) practices of CMMI were redefined in order customize them for XP. This study was useful for our research because we are developing a TMMi-based software test process where TMMi and CMMI are conceptually similar.

K. S. Lee et al. [12] proposed a software development process that based on Rational Unified Process (RUP). The proposed process is a tailored version of RUP for Korean Core Instrumentation System. They evaluated the result by with typical waterfall lifecycle model and RUP.

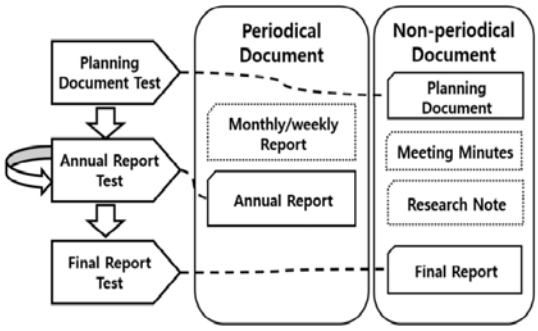
In S. W. Shin et al. [13], a number of improvement models were used such as SPICE(Software Process Improvement and Capability dEtermination) and CMMI which is can be used for improving quality of mobile embedded software. In this paper, XP is also used because it has the iteration development feature. Thus, authors proposed a XP-based software process improvement framework that can achieve CMMI level 2 or 3.

J. A. Kim [17] proposed a quality assessment framework for evaluating medical software R&D Project. Authors identified the critical features of medical device software such as safety, standardization and continuing change. Those features are used for evaluation of medical software R&D project. However, the study limited to focus on only the medical field of R&D.

3. DEFINING TEST PROCESS IN SOFTWARE R&D PROJECT

3.1 TEST PROCESS FOR SOFTWARE RESEARCH DOCUMENTS

Test Process for R&D Project tests document artifacts such as planning document, annual report and final report which are sequentially made from testing activities. At planning level, researcher creates planning document which includes content like overall plan, final research goal and annual research goal. Thus, it is verified by tester and result will be informed to researcher.



(Figure 3) TEST PROCESS FOR R&D PROJECT

According to contents in Planning document submitted, research progress and next year plan are verified by researcher submits annual report every year. Finally, the end of research, research result is verified by contents of final report.

R&D Test Process documents are classified into periodical and non-periodical documents as showed in Figure 3.

In Figure 3, the documents which are used for software testing are in bold frame. Planning document and Final report are non-periodical documents that both are tested once in project lifetime. Annual Report is tested every year because it is documented every year needed to be tested periodically.

As shown in Figure 3, the periodical documents like monthly/weekly report and also non-periodical documents such as meeting minutes, research note are generated through R&D process but not used for software testing. The documents that are not used in software testing like Monthly/weekly report, meeting minute and research note are written in free form and simple example is shown in Figure 4.

Title : Design Test Framework ~ 2018-06-27
Execution Date : 2015-02 for R&D Project
Plan Date : 2015-02-02 ~ 2015-02-10
Progress : 15%

Manager	Date	Title	Content	Time
Amarnend Dashbalbar	2015-05-11	Designing a Test Framework	I/O Definition for Content-base test module	7
Eun Chul Lee	2015-06-02	Designing a Test Framework	Design of the framework architecture using defined level and module	8

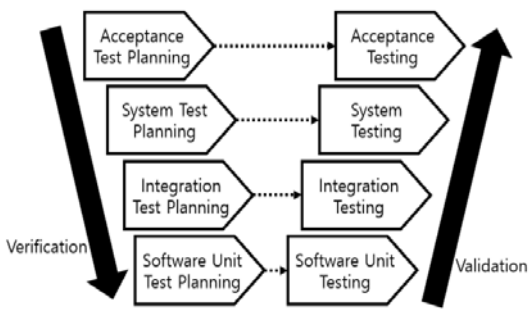
(Figure 4) EXAMPLE OF RESEARCH NOTE

Research documents not used for testing, shown in Figure 4, are helpful for monitoring project progress because they are produced constantly during the progression of project unlike non-periodical documents such as planning document, annual report and final report.

Therefore, in this paper, the test process for software R&D includes testing activities which are based on those non-periodical documents such as meeting minutes, research note and weekly report submitted constantly. Usage of periodical documents in testing makes it possible to monitor project progress effectively. Therefore, they are used as additional information for accurate testing in each level of test process.

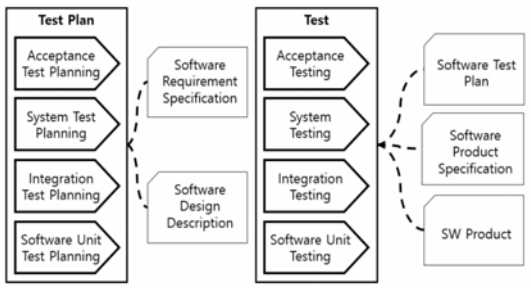
3.2 TEST PROCESS FOR SOFTWARE DEVELOPMENT

Software development is required to be tested often for the reason that invisibility feature of software [14]. Therefore, software development test process is based on common model V&V (Verification and Validation). The verification level of V&V test process consists of acceptance test planning, system test planning, integration test planning and software unit test planning. Thus, validation level consists of acceptance testing, system testing, integration testing and software unit testing.



(Figure 5) SOFTWARE V&V PROCESS

As pointed by arrow in Figure 5, Acceptance test plan is associated with acceptance testing activity and system test plan is associated with system testing activity respectively and so on. If test plan and test activities are structured like this pattern in the process we introduced, complex software artifacts can be tested more precisely.



(Figure 6) ARTIFACTS IN SOFTWARE V&V PROCESS

As shown in Figure 6, only source code items are not enough for software development testing also various document artifacts are required. There are development documents like Software Design Description (SDD) and Software Requirement Specification (SRS) which associated to both test planning and test activities. There are also SW Product that provides source code which is considered as software related artifact and document artifacts such as Software Test Plan (STP) and Software Product Specification (SPS).

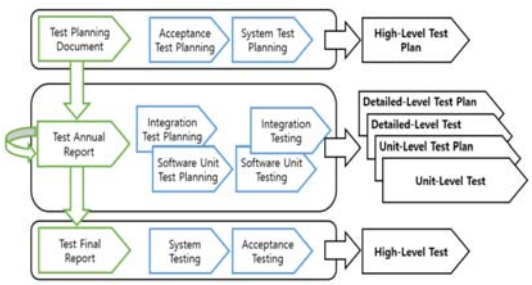
Figure 6 shows dependencies between software activities and software artifacts. SRS contains user requirements analyzed by researcher and it can be used in system testing and acceptance testing. SDD includes items like module design and software structure. Thus, SDD is used in integration test plan level and also used in unit testing and integration testing levels. The artifacts that SPS, STP and SW product are used in all validation activities.

We are introducing a test process that contains both V&V planning and testing activity because we set software R&D as our target. This test process utilizes software development documents such as SRS, SDD and SPS.

3.3 TEST PROCESS FOR SOFTWARE R&D PROJECT

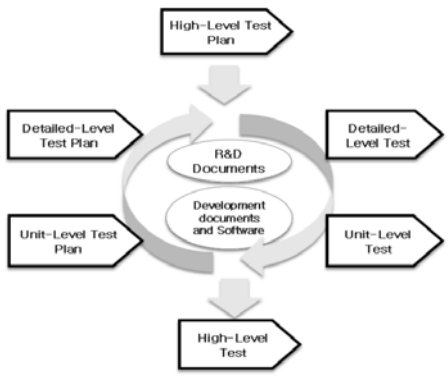
We define the process that tests not only software research artifacts but also software development artifacts by analyzing both software R&D test process and software test process. The software R&D test process is based on TMMi model and ISO 29119-2 standard.

The activities of software R&D test process levels are defined by activities of R&D process and software test process. Figure 7 shows activities of high level test plan, detailed level test plan and unit level test plan as test planning activities. Testing activities such as high level testing, detailed level testing and unit level testing are also shown.



(Figure 7) DEFINITION OF TEST PROCESS ACTIVITIES

System test plan and acceptance test plan for testing R&D planning document are defined as High level test plan. The annual report test plan and integration test plan are defined as detailed-level test plan. The research note test plan and unit test plan defined as unit-level test plan. High-level testing examines the final report and software system. Detailed-level testing examines the annual report and software integration. And unit-level testing confirms research note and software module.



(Figure 8) SOFTWARE R&D TEST PROCESS

Structure and cycle of the proposed process is shown in Figure 8. The test process needs to be performed the high-level testing activity with project management related documents and software related documents. After high-level testing activity is finished, the detailed level test plan, unit level test plan, detailed level testing and unit level testing activities will be performed repeatedly. In last stage of process, comprehensive testing will be performed in High-level testing activity.

(Table 1) ACTIVITIES AND ARTIFACTS

Software R&D Test Activity	**Artifacts(input → output)**
High-Level Test Planning	R&D Project Plan with Template, Software Requirement Specification(SRS) → Software Lifecycle Test Plan (for Project Plan and System Testing)
Detailed-Level Test Planning	Project Annual Report Template, Software Design Description(SDD) → Software Lifecycle Test Plan (for Annual Report Items and Integration Testing)
Unit-Level Test Planning	Project Research Note Template, SDD → Software Lifecycle Test Plan (for Research Note and Unit Testing)
Unit-Level Test	Software Lifecycle Test Plan, Research Note, Program Code → Software Lifecycle Test Report (for Research Note and Unit Testing)
Detailed-Level Test	Software Lifecycle Test Plan, Project Annual Report, Program Code → Software Lifecycle Test Report (for Annual Report and Integration Testing)
High-Level Test	Software Lifecycle Test Plan, Project Final Report, Program Code → Software Lifecycle Test Report (for Final Report and System Testing)

Input documents are defined for each by activities of software R&D test process. During the planning activities of the process is being performed, the Software Life-Cycle Test Plan(SLTP) is generated and used in the next activities such as high-level test, detailed-level test and unit-level test.

After test activities performed and concluded as instructed in SLTP, the test result is shown through Software Life-Cycle Test Report (SLTR). Table 1 shows all input and output documents of process activities.

3.4 DOCUMENTATION OF TEST PROCESS

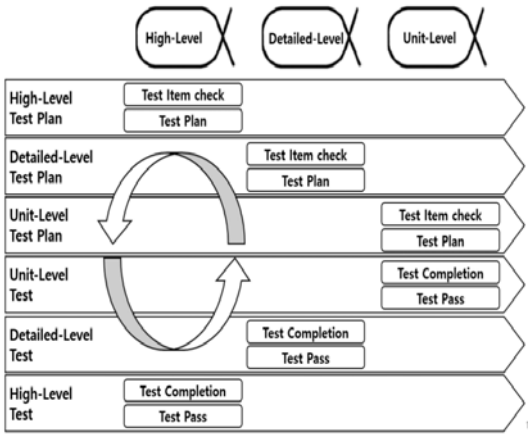
The proposed test process has test planning document and test result document, SLTP and SLTR [15]. SLTP includes

contents that related to planning of software testing such as test strategy, risk approach, test environment requirement, test case and scenario etc.

SLTR contains test result of both software product and document artifacts. The test results of three levels are shown separately. For example, software test result of High-level testing is shows by 6 metrics that defined in Software Quality standard ISO9126. The quality software product is evaluated by its functionality, reliability, usability, efficiency, maintainability and portability. On the other hand, document artifacts are evaluated by three metrics, Traceability, Completeness and Consistency.

4. CASE STUDY: MODELING TEST PROCESS USING ESSENCE

In software process modeling field, there are two well-known software process modeling languages, SPEM and Essence, both created by Object Management Group. We used a comparative study [10] in order to make a choice for which one best suits our process. In the comparative study, SPEM language architecture has some shortage in process enactment. On the other hand, Essence contains some better concept and structure for supporting enactment.



(Figure 9) TEST PROCESS MODEL IN ESSENCE

Therefore, we choose Essence, where its detailed information is in Section 2, to model our process and process can be structured by essential object (alpha), element activity (activity) and element role (Competency). We present a model in Figure 9 and define high-level, detailed-level and unit-levels as alphas of our model. Thus, activities of planning and testing are defined as activities of the model.

We compared our paper with Imoto[16] in order to show our process model's benefit and advantages. Artifacts used in project evaluation model can show how properly the projects evaluated by specific model. Table 2 shows R&D related artifacts used in our proposed approach and Imoto's[16].

(Table 2) ACTIVITIES AND ARTIFACTS

Approach	Artifacts used
Our approach	Planning Document, Annual Report, Research Note, Meeting minute
Imoto[16]	Planning Document

Imoto's study used seven evaluation indices and they all can be found in R&D project planning document. On the other hand, our approach used other periodical and non-periodical documents such as annual report, research note, meeting minute. Those artifacts are useful for more proper evaluation of R&D project and makes project real monitoring even possible.

5. CONCLUSION

Software R&D colleagues should examine research documents such as project planning document, annual report and final report. And they should check development documents like requirements specification, design description and test program code. Thus, in this paper, we described a test process with activities to validate artifacts produced in software R&D. The process enables them to monitor the project progress more effectively because the process utilizes constantly generated artifacts.

However, we have to define more detailed activities to apply our process to software R&D project. We also need a tool to support our process. Therefore, we plan to define process activities in detail. Finally, we will implement a tool to enact a detailed test process which is modeled by Essence.

References

[1] I. Jacobson, S. Huangb, M. Kajko-Mattssonc, P. McMahond, and E. Seymoure, "Semat—three year vision," Programming and computer software, vol. 38, no. 1, pp.1-12, 2012. http://dx.doi.org/10.15514/syrcose-2011-5-inv

[2] I Jacobson, P. Ng, P. McMahon, I. Spence and S Lidman, "The essence of software engineering: the SEMAT kernel," Queue - Networks, vol. 10, no. 10, 2012. http://dl.acm.org/citation.cfm?id=2389616

[3] T. Sedano, and P. Cécile, "State-based Monitoring and Goal-driven Project Steering: Field Study of the SEMAT Essence Framework," In Proc. of the 36th International Conference on Software Engineering, pp.325-334, 2014. http://dx.doi.org/10.1145/2591062.2591155

[4] I. Burnstein, A. Homyen, R. Grom, and CR Carlson, "A model to assess testing process maturity," Crosstalk the Journal of Defense Software Engineering, vol. 11, no. 11, pp.6-30, 1998. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.434.1067&rep=rep1&type=pdf

[5] T. Ericson, A. Subotic, and S. Ursing. "TIM - A Test Improvement Model," Software Testing Verification and Reliability, vol. 7, no. 4, pp. 229-246, 1997. http://dx.doi.org/10.1002/(sici)1099-1689(199712)7:4<229::aid-stvr149>3.3.co;2-d

[6] I. Burnstein, S. Taratip, and C. Robert, "Developing a testing maturity model for software test process evaluation and improvement," In Proc. of International Test Conference, pp. 581–589, 1996. http://dx.doi.org/10.1109/test.1996.557106

[7] E. van Veenendaal, J. Jaap Cannegieter, "Test Maturity Model Integration (TMMi) Results of the first TMMi benchmark - where are we today?", pp.3, Euro Star Software Testing Community, 2013.

[8] E. van Veenendaal, R. Grooff and R. Hendriks, "Test Process Improvement using TMMi," Testing Experience: The Magazine for Professional Testers, vol. 3, no. 19, pp.21-25, 2008. http://www.erikvanveenendaal.nl/NL/files/Test%20Process%20Improvement%20using%20TMM(i).pdf

[9] P. Ng, and S. Huang, "Essence: A framework to help bridge the gap between software engineering education and industry needs," In Proc. of IEEE 26th Conference on Software Engineering Education and Training (CSEE&T), pp-304-308, 2013. http://dx.doi.org/10.1109/cseet.2013.6595266

[10] B. Elvesæter, G. Benguria and S. Ilieva, "A comparison of the Essence 1.0 and SPEM 2.0 specifications for software engineering methods," In Proc. of the Third Workshop on Process-Based Approaches for Model-Driven Engineering, no. 2, p. 2, 2013. http://dx.doi.org/10.1145/2489833.2489835

[11] D. J. Han and H. S. Han, "Guidelines for Implementing Configuration Management in Extreme Programming based on CMMI," Journal of Internet Computing and Services, vol. 9, no. 2, pp. 107-118, 2008.

[12] K. S. Lee and T. G. Lee, "A Software Development Process of Core Instrumentation System Based on the Rational Unified Process," Journal of Internet Computing and Services, vol. 5, no. 4, pp. 95-113, 2004.

[13] S. W. Shin, H. K. Kim and S. W. Kim, "Framework for Improving Mobile Embedded Software Process," Journal of Internet Computing and Services, vol. 10, no. 5, pp. 195-209, 2009.

[14] J. Cangussu, R. DeCarlo, A. MATHUR, "Using sensitivity analysis to validate a state variable model of the software test process," IEEE Transactions on Software Engineering, vol. 29, no. 5, pp.430-443, 2003. http://dx.doi.org/10.1109/tse.2003.1199072

[15] K. H. Jin, S. M. Song, J. W. Lee and B. J. Lee, "Test Planning and Reporting for Constant Monitoring of Software R&D Projects," Korea Computer Congress, Vol. 42, No. 1, pp.597-599, 2015.

[16] S. Imoto, Y. Yoshiyuki and W. Junzo. "Fuzzy regression model of R&D project evaluation." Applied Soft Computing, vol. 8, no. 3, pp.1266-1273, 2008. http://dx.doi.org/10.1016/j.asoc.2007.02.024

[17] J. A. Kim, J. H. Kim, "Quality Assessment Framework for Medical Device specific SW R&D Project." International Journal of Software Engineering and Its Applications, vol. 8, no. 1, pp.371-376, 2014. http://dx.doi.org/10.14257/ijseia.2014.8.1.32

◐ 저 자 소 개 ◑

아마르멘드 (Amarmend Dashbalbar)
2013년 Mongolian University of Science and Technology 졸업 (학사)
2014년~ 현재 서울시립대학교 컴퓨터과학과 석사 재학 중
관심분야: 소프트웨어 테스팅, 결함관리 시스템
E-mail: amaraa2848@gmail.com

이 은 철 (Eun-Chul Lee)
2007년 안양대학교 컴퓨터공학과 졸업 (학사)
2014년~ 서울시립대 컴퓨터과학과 졸업 (석사)
관심분야: 테스트 프로세스, 테스트 프레임워크
E-mail: herolec@naver.com

이 정 원 (Jung-Won Lee)
1993년 이화여자대학교 전자계산학과 졸업 (학사)
1995년 이화여자대학교 전자계산학과 졸업 (석사)
2003년 이화여자대학교 컴퓨터공학 졸업 (박사)
2012년 - 현재 아주대학교 전자공학과 부교수
관심분야: SOA, Ubiquitous Computing, Embedded Software and Software engineering
E-mail: jungwony@ajou.ac.kr

이 병 정 (Byunjeong Lee)
1990년 서울대학교 계산통계학과 졸업(학사)
1998년 서울대학교 대학원 전산과학과 졸업(석사)
2002년 서울대학교 대학원 전기 컴퓨터공학부 졸업(박사)
1902～현재 서울시립대학 컴퓨터과학부 교수
관심분야: 소프트웨어테스트, 소프트웨어 진화, 소프트웨어공학
E-mail: bjlee@uos.ac.kr