Towards Enacting a SPEM-based Test Process with Maturity Levels

Amarmend Dashbalbar[1], Sang-Min Song[1], Jung-Won Lee[2] and Byungjeong Lee[1*]
[1]Department of Computer Science, University of Seoul
Seoul, 02504 - South Korea
[e-mail: {iicegrad201439, maro0419, bjlee}@uos.ac.kr]
[2] Department of Electrical and Computer Engineering, Ajou University
Suwon, 16499 - South Korea
[e-mail: jungwony@ajou.ac.kr]
*Corresponding author: Byungjeong Lee

Received September 20, 2016; revised February 5, 2016; accepted February 27, 2017; published February 28, 2017

Abstract

Effective monitoring and testing during each step are essential for document verification in research and development (R&D) projects. In software development, proper testing is required to verify it carefully and constantly because of the invisibility features of software. However, not enough studies on test processes for R&D projects have been done. Thus, in this paper, we introduce a Test Maturity Model integration (TMMi)-based software field R&D test process that offers five integrity levels and makes the process compatible for different types of projects. The Software & Systems Process Engineering Metamodel (SPEM) is used widely in the software process–modeling context, but it lacks built-in enactment capabilities, so there is no tool or process engine that enables one to execute the process models described in SPEM. Business Process Model and Notation (BPMN)-based workflow engines can be a solution for process execution, but process models described in SPEM need to be converted to BPMN models. Thus, we propose an approach to support enactment of SPEM-based process models by converting them into business processes. We show the effectiveness of our approach through converting software R&D test processes specified in SPEM in a case study.

Keywords: Software Test Process, R&D Test Process, Maturity Levels, Enactment, SPEM

A preliminary version of this paper was presented at APIC-IST 2016, and was selected as an outstanding paper. 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).

https://doi.org/10.3837/tiis.2017.02.034 **ISSN : 1976-7277**

1. Introduction

Software systems are getting bigger and more complex as they support various operating systems, languages, and platforms. Testing such software systems is becoming even more difficult to complete [1]. If a systematic test process is defined and executed, software testing itself can be reliable. Also, if there is a test process that includes systematic maturity levels that can be selected to suit the characteristics of the project in various domains for not only general software research and development (R&D) projects, but also medical device software, automobile software, etc., it will be very helpful for effective testing. And if verification of document artifacts [2-3] generated from a research step is tested before the software development process [4], it is possible to accurately verify the activity step by step. Therefore, high reliability is provided not only for software but for the whole project through a framework including a software test and a document artifact test [5]. This paper is the definitive version of a conference paper that defined correspondence items of the Software & Systems Process Engineering Metamodel (SPEM) [6] and Business Process Model and Notation (BPMN) [7] so that processes can be executed through a Business Process Execution Language (BPEL)-based workflow engine [8]. The major contributions from this paper are summarized as follows.

- In this paper, activity, task, and outcome in the test process, including maturity level, are defined based on ISO 29119-2 and Test Maturity Model integration (TMMi).
- The test process defined in this paper is divided into a SPEM model and a BPMN model, and we then prove its effectiveness through related research and quantitative comparison.
- This study demonstrates the possibilities of our approach by converting a SPEM model to a BPMN model in a case study.

The remainder of the paper is structured as follows. Section 2 provides background to this paper, and Section 3 summarizes related studies. Section 4 introduces an integrity level–based test process, and Section 5 describes the mapping between the two standards. An example of this paper's contribution is illustrated in Section 6, and Section 7 expands on it with a discussion. Section 8 concludes the paper, giving a direction for future research.

2. Background

2.1 SPEM

SPEM is a software process modeling standard and language that was released by the Object Management Group (OMG) in 2008. SPEM is widely used among process engineers in the software process field. It has high expressiveness in modeling software processes as it provides a wide variety of elements, such as Activity, Task, Role, Work Product, etc. [9].

Fig. 1 shows a SPEM model diagram, which presents the basic elements of the SPEM metamodel. In **Fig. 2**, there is a task named Use Case Analysis, and it performs two roles (Designer Analyst and System Analyst) as well as three work products; two of them are input artifacts of the task, and one is an output artifact. Activity represents part of the work to be done in the software development cycle, and it can be divided into one or more tasks. Task is also a part of the work to do, but is smaller than Activity. Role presents the person who

performs a Task, and a Work Product includes the artifacts that are used or generated by the Role when the Task is performed.

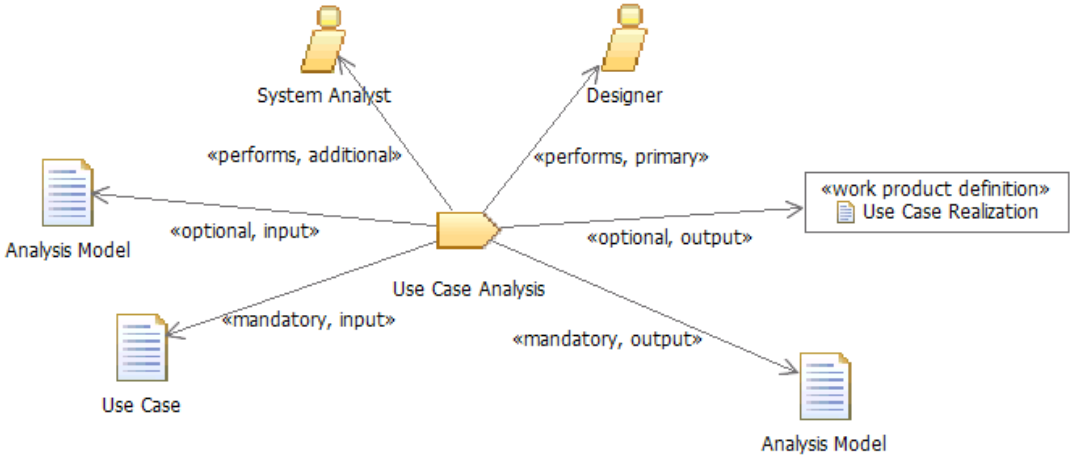


Fig. 1. SPEM Process Model Example [6]

2.2 BPMN

Business Process Model and Notation is notation language for modeling business processes in a variety of fields. It provides over 100 notation elements to represent business process models. There are elements (like a sub-process and a task) to represent a unit of work that needs to be done in order to produce work products. Work products are presented in BPMN by data objects.

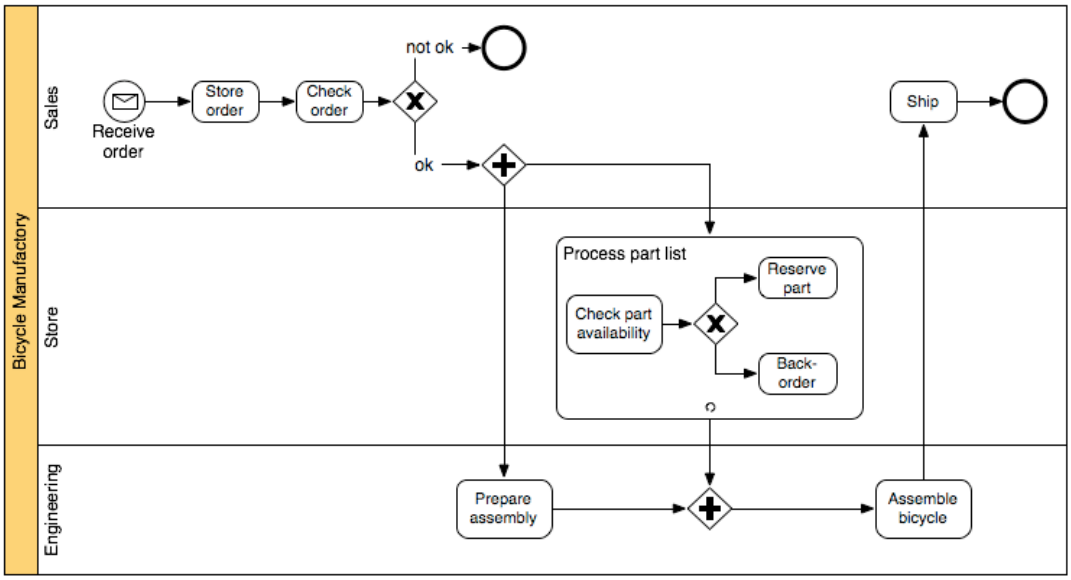


Fig. 2. BPMN Process Model Example [7]

Fig. 2 shows a business process modeled in BPMN, which gives us an opportunity to get familiar with the basic BPMN elements. There is a business process named Bicycle Manufacturer, which has three lanes where each presents roles. The element contained within the bounds of a lane means it is executed by a worker corresponding to the lane.

2.3 TMMi

Test Maturity Model integration is a test-maturity reference model developed by the TMMi Foundation. Its structure is the same as the Capability Maturity Model (CMM), and the concept was first introduced in 1996. TMMi was made to improve testing effectiveness, and made it possible for organizations to determine the fulfillment and effectiveness of their testing. A quality assurance framework is included in the TMMi model and is used for a connection that provides information on concepts and ideas between workers in a large organization.

Many studies have been published on improving the software testing process and evaluating the maturity level of a test process [10-14], and most of those studies are based on TMMi. As shown in **Fig. 3**, TMMi is a five-level hierarchically structured reference model for a software test process, where process managers can follow the levels in order to improve a test process. TMMi has five maturity levels: Initial, Managed, Defined, Measured, and Optimization. Each level includes process areas that need to be done in order to advance to the next level. Process engineers can get an evaluation of the test process from TMMi-licensed organizations.

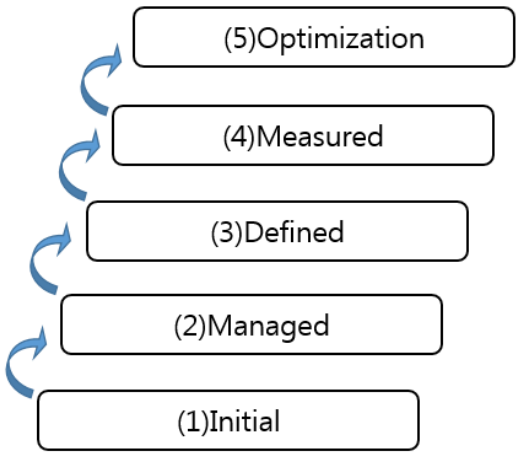


Fig. 3. TMMi levels for process maturity

TMMi process areas contain two kinds of practice (specific and generic), and both are the lowest units of the TMMi model. A specific practice is included only in one particular process area. A generic process is connected to two or more process areas, so fulfillment for several process areas is dependent on one generic practice. There are also specific and generic goals, which indicate the purpose of specific and generic practices and that need to be satisfied by those practices when they are done.

3. Related Work

Debnath et al. [15] offered a mapping approach between SPEM and BPMN using Query/View/Transformation (QVT), the model transformation language. However, the mapping is not detailed enough, and the result of the transformation was not shown in the study.

Portela et al. [16] introduced a comparative analysis between SPEM and BPMN in order to demonstrate the expressiveness of the two standards in a software process–modeling context. Their study compared the two standards' elements, which are contained in the standard process model derived from the Ontology-based software Development Environment (ODE) project. Only a few elements were included in the comparison, and it is not comprehensive enough for process conversion.

Elvesæter [9] performed a comparison of Essence 1.0 and SPEM 2.0 specifications, both OMG standards in software engineering, and Scrum is modeled on two standards for the purposes of a case study. Both standards have a lot in common, but there are some differences between them. The primary one is a process execution problem—SPEM lacks built-in execution capabilities, but Essence has better possibilities on enactment.

4. Process Customization by Maturity Levels

A test process can perform more efficient testing when it is suitable for the attributes and situations of the system being tested [17-19]. Test processes can vary due to factors such as product size, available human resources, time remaining until product release, etc. Therefore, we introduce five integration levels of the proposed test process that satisfy all TMMi maturity levels. In TMMi Level 1, the test process is undefined, and an organization may not be able to provide a stable environment for the test.

At this process level, software products may be released without being tested for quality and risks. However, according to STA Consulting Engineers, test processes for most of the small/medium-sized companies in South Korea are at Level 1. Therefore, we defined a Level 1 version of the Software R&D Test Process, because (due to lack of resources, time, etc.) it is difficult to avoid using a Level 1 process in small and medium-sized enterprises. TMMi describes Level 2 test processes as required in order to document a test policy and strategy that describes content like possible risks that can occur while testing, and the solutions for them.

Also, a test plan that describes test execution as well as test design methods, and which generates a test case, is an essential part of the process for satisfying the TMMi Level 2 conditions. According to TMMi, a Level 3 process is a defined process, and testing is not an activity that starts after coding but is fully integrated into the development life cycle. All the requirements of Level 2 are supposed to be included in Level 3 and further improved. A Level 4 process must have self-assessment practices, and tests the quality of the software product. Level 5 has practices that analyze common reasons for fault occurrences, and finds methods to prevent them. Moreover, a process at this level focuses on improvement of the process itself.

The test process defined in this paper has the structure shown in **Fig. 4**. The defined test process corresponds to the process item at the top. And below, there are the 'Planning' and 'Testing' Phases. Therefore, the Level, Activity, and Task that comprise the 'Planning' Phase are shown in **Table 1**, and the 'Testing' Phase is shown in **Table 2**.

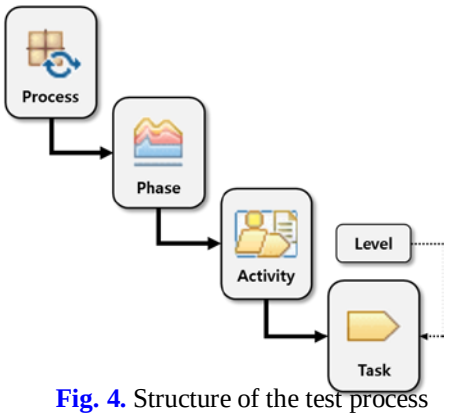


Fig. 4. Structure of the test process

As shown in **Table 1** and **Table 2**, the proposed test process consists of the smallest Task Set in Level 1. Because it is the simplest process, it requires less time and effort than higher levels. However, as the maturity level of the process increases, the number of tasks involved increases. So, higher levels require more time and effort, but provide specific and meticulous testing. Thus, a tester who selects a level suitable to the project characteristics can perform efficient testing.

Table 1. List of planning phases by maturity level

Activity	Tasks	1	2	3	4	5
Non-functional Test Planning	Perform a Non-functional Product Risk Assessment			o	o	o
	Establish a Non-functional Test Approach			o	o	o
Test Organization	Establish a Test Organization			o	o	o
	Establish Test Career Paths			o	o	o
	Establish an Organizational Test Training Capability			o	o	o
	Provide Test Training			o	o	o
	Determine, Plan and Implement Test Process Improvements			o	o	o
Test Management Planning	Project Goals for Product Quality and their Priorities are Established				o	o
	Determine Common Causes of Defects					o
	Prioritize and Define Actions to Systematically Eliminate Root Causes of Defects					o
	Establish a Statistically Controlled Test Process					o
Test Planning	Understand Context		o	o	o	o
	Identify & Estimate Risk		o	o	o	o
	Identify Risk Treatment approaches		o	o	o	o
	Design Test Strategy	o	o	o	o	o
	Determine Staffing and Scheduling	o	o	o	o	o

In **Table 1**, activities include the 'Non-functional Test Planning' Activity, which plans non-functional testing; the 'Test Organization' Activity, which plans and performs tasks in the Test Organization; the 'Test Management Planning' Activity, which manages the entire test project; and the 'Test Planning' Activity, which makes a specific plan about testing.

Table 2. List of testing phases by maturity level

Activity	Tasks	1	2	3	4	5
Test Environment Set-Up & Maintenance	Establish Test Environment	o	o	o	o	o
	Maintain Test Environment	o	o	o	o	o
Test Design & Implementation	Identify Feature Set	o	o	o	o	o
	Derive Test Conditions	o	o	o	o	o
	Derive Test Coverage	o	o	o	o	o
	Derive Test Cases	o	o	o	o	o
	Derive Test Procedures	o	o	o	o	o
Non-functional Testing	Perform Non-functional Test Analysis and Design			o	o	o
	Perform Non-functional Test Implementation			o	o	o
	Perform Non-functional Test Execution			o	o	o
Test Management	Testing is Performed using Statistical Methods					o
	Actual Progress toward Achieving the Project's Product Quality Goals is Quantified and Managed				o	o
Test Execution	Execute Test Procedures	o	o	o	o	o
	Compare Test Results	o	o	o	o	o
Test Incident Reporting	Analyze Test Result			o	o	o
	Create Incident Report			o	o	o

Table 2 shows the 'Test Environment Set-Up & Maintenance' Activity, which sets up and maintains an environment for performing testing tasks, and the 'Test Design & Implementation' Activity, which designs and implements testing for making a test case. The 'Non-functional Testing' Activity performs testing according to the plan created in the 'Non-functional Test Planning' Activity, and the 'Test Management' Activity manages the overall test (for quality). The 'Test Execution' Activity tests designed and implemented test cases, and the 'Test Incident Reporting' Activity is an incident report on the results after testing.

Table 1 and **Table 2** specify tasks belonging to the Activity and the maturity levels of the tasks in levels 1 to 5. Of the five levels, Level 1 and Level 2 have different tasks for document artifact tests and software testing. The target of the software test is the model or the source code, and the target of the document artifact test is the artifact, such as the Software Requirement Specification (SRS), because it focuses on how to perform a successful and reliable test according to each given object. However, at Level 3 and above, we focus on improving the Test Process and the organization for performing the tests, as well as the ability

of members of the organization, and we aim to improve the process without distinguishing between software tests and document artifact tests. That is because testing based on the artifact is not the purpose.

Table 3 shows the outcomes of the software test and the document artifact test for Level 1 and Level 2 tasks. 'Scope', 'Analyzed Risks', 'Risk Treatment Approaches', 'Test Strategy', plus 'Schedule and Staffing Profile' are outcomes for each task from the 'Understand Context' Task up to the 'Determine Staffing and Scheduling' Task, which correspond to the 'Planning' Phase. However, the output of the task corresponding to the 'Testing' Phase is somewhat different.

Table 3. The outcomes of the test processes at Level 1 and Level 2

<table>
<tr><th>Tasks</th><th>Software Test Outcome</th><th>Document Artifact Test Outcome</th></tr>
<tr><td>Understand Context</td><td colspan="2">Scope</td></tr>
<tr><td>Identify & Estimate Risk</td><td colspan="2">Analyzed Risks</td></tr>
<tr><td>Identify Risk Treatment approaches</td><td colspan="2">Risk Treatment Approaches</td></tr>
<tr><td>Design Test Strategy</td><td colspan="2">Test Strategy</td></tr>
<tr><td>Determine Staffing and Scheduling</td><td colspan="2">Schedule and Staffing Profile</td></tr>
<tr><td>Establish Test Environment</td><td colspan="2">Test Environment Readiness Report</td></tr>
<tr><td>Maintain Test Environment</td><td colspan="2">Test Environment Report</td></tr>
<tr><td>Identify Feature Set</td><td rowspan="2">Test Item, Test Environment Requirement</td><td rowspan="2">Configuration Item & Test Condition By Configuration Item</td></tr>
<tr><td>Derive Test Conditions</td></tr>
<tr><td>Derive Test Coverage</td><td>Test Design Specification</td><td>Corresponding Item</td></tr>
<tr><td>Derive Test Cases</td><td>Test Case Specification</td><td>RLIM(Relevance Link Information Model)</td></tr>
<tr><td>Derive Test Procedures</td><td colspan="2">Test Procedures Specification</td></tr>
<tr><td>Execute Test Procedures</td><td colspan="2" rowspan="2">Test Report</td></tr>
<tr><td>Compare Test Results</td></tr>
</table>

The software test identifies the targets of the items to be tested in the 'Identify Feature Set' Task, the 'Derive Test Conditions' Task, and the 'Derive Test Coverage' Task. The test conditions and the test coverage are determined, and then 'Test Items' and 'Test Design Specification' are generated. 'Test Environment Requirements' are created as outcomes. However, document artifact test generates the 'configuration item' of the document and the 'Test Condition By Configuration Item' to be tested, plus a 'corresponding item' to compare and test the documents. In the 'Derive Test Cases' Task, 'Test Case Specification', which corresponds to outcome of the software test, is generated. And 'Relevance Link Information Model (RLIM)' [20] is an outcome in the document artifact test.

Table 4 shows the outcomes for tasks from Level 3 to Level 5. Perform a 'Non-functional Product Risk Assessment' Task and 'Establish a Statistically Controlled Test Process' Task belong to the 'Planning' Phase, and most outcomes of the software test and the document artifact test are the same. 'Project Goals for Product Quality and their Priorities are Established' Task are the final output of the software, which generate Software Product and 'Product Quality Goals', which are goals of Product Quality. However, in the document artifact test, the 'Project Quality Goals (using Artifact Quality)' appear, since the output product is not Software Product.

Table 4. The outcomes of the test process at levels 3, 4 and 5

Tasks	Software Test Outcome	Document Artifact Test Outcome
Perform a Non-functional Product Risk Assessment	Non-functional Product Risks	
Establish a Non-functional Test Approach	Non-functional Test Approaches	
Establish a Test Organization	Test Organization Description	
Establish Test Career Paths	Test Career Path Plan	
Establish an Organizational Test Training Capability	Test Training Plan	
Provide Test Training	Test Training Report	
Determine, Plan and Implement Test Process Improvements	Test Process Improvement Plan	
Project Goals for Product Quality and their Priorities are Established	Product Quality Goals	Project Quality Goals (Using Artifact Quality)
Determine Common Causes of Defects	Analyzed Defects	
Prioritize and Define Actions to Systematically Eliminate Root Causes of Defects	Defect Treatment Approaches	Defect Treatment Approaches & Proposals
Establish a Statistically Controlled Test Process	Statistically Controlled Test Process Description	Statistical Indices of Results of Content-based Document Artifact Test
Perform Non-functional Test Analysis and Design	Non-functional Test Specification	
Perform Non-functional Test Implementation		
Perform Non-functional Test Execution	Non-functional Test Report	

Tasks	Software Test Outcome	Document Artifact Test Outcome
Testing is Performed using Statistical Methods	Statistical Test Report	Establish/Systematize Content-based Document Artifact Test
Actual Progress toward Achieving the Project's Product Quality Goals is Quantified and Managed	Test Management Report	
Analyze Test Result	Incident Report	
Create Incident Report		

The 'Establish a Statistically Controlled Test Process' Task in **Table 4** generates the 'Statistically Controlled Test Process Description' as an outcome of the software test, and the 'Statistical Indices of Results of the content-based document artifact test' is an outcome of the document artifact test. In addition, the content from the 'Perform Non-functional Test Analysis and Design' Task to the 'Create Incident Report' Task, correspond to the 'Testing' Phase. Outcomes are different in the 'Statistical Test Report', and the 'Establish/Systematize Content-based document artifact test' in 'Testing is Performed using the Statistical Methods' Task.

5. Mapping from SPEM to BPMN

SPEM 2.0 was introduced in 2008 by the Object Management Group as the successor to SPEM 1.1, and it became a new software process modeling standard. SPEM 2.0 has a wide range of components representing software-related terms and concepts. In the chapter titled Enacting SPEM 2.0 Processes in the official specifications [6] introduces the two most common ways to enact the SPEM process. The first is a mapping process into Project Plans, and a translated process can be enacted by project planning systems, such as IBM Rational Portfolio Manager or Microsoft Project. The second is a mapping process into a business flow or into process execution languages. And then, a translated process can be run using workflow engines, such as BPEL-based workflow engines.

We chose the second way, because most BPEL-based workflow engines are open source. Our target process metamodel BPMN is a general business process–modeling standard; thus, it has no element for expressing terms or concepts of a specific field, such as software. Therefore, it is used in a variety of fields, and that offers a wider choice of enactment engines and better portability of process models.

As mentioned earlier, SPEM is a software process–modeling standard, and it has many variations of process elements to express detailed components of the software development life cycle. Therefore, our approach is to help process engineers to execute their SPEM-based process by providing comprehensive process mapping between the two standards. However, BPMN is not a software process–modeling language, so it only includes general business process elements. The mapping table between SPEM and BPMN [8] is available at http://selab.uos.ac.kr/APIC_IST_16/table.pdf.

The process in SPEM is a business process, or simply a BPMN file under the BPMN standard. The process pattern in SPEM represents a set of elements prepared for process reuse.

Thus, it can be mapped to a reusable sub-process of BPMN. Phase and Activity in SPEM are mapped to sub-processes in BPMN. According to the SPEM standard, the task is an atomic work–breakdown structure element, which is contained within the Activity task. In BPMN, a task is also the smallest unit of work in the breakdown structure, and it represents a number of variations in tasks, such as User Task, Send Task, Receive Task, Manual Task, Service Task, Business Rule Task, and Script Task. Iteration in SPEM is a set of work breakdown elements in a loop, and can be represented by a sub-process with the Standard Loop attribute. Outcome in SPEM is a one-of-a-kind Work Product class, and it represents non-tangible work products that are usually the output of a single task. Thus, in BPMN, the element named Message represents the input and output data of tasks. Other Work Product types in SPEM are Artifact, Deliverable, WorkProductDefinition, and WorkProductUse, and they can be represented by Data Object elements in BPMN. The SPEM specifications describe Domain as a group of related Work Products, and the similar element in BPMN is Data Object Collection, which also represents a collection of work products.

A worker or a group performing a specific task is termed RoleUse, CompositeRole, or TeamProfile, etc. However, there are not enough elements to express the variety of elements in SPEM; only Pool and Lane are similar to them. Lane represents a Role, and Pool represents a group of Roles. The element named Milestone expresses a significant event in the software development life cycle. Thus, it can be mapped to Event elements in BPMN.

Category in SPEM is a group of related elements in cross types, and Discipline represents a group of related tasks. Thus, both can be expressed by Group in BPMN. Step in SPEM is a list of steps for performing a single task, which provides information on what to do in order to complete the task; but BPMN does not provide any elements for information about a Task, except for Text Annotation. Thus, SPEM's step-like information can be described by text, and it can be modeled by the Text Annotation element in the process model. Another SPEM element that can be represented by BPMN's Text Annotation is ToolDefinition, which represents a specific tool or automation unit used by a Role to perform a task, and provides information about it. Thus, this element can also be represented by Text Annotation. The last element that can be expressed by Text Annotation is Guidance. Actually, Guidance has a number of forms, such as Checklist, Concept, Example, Guideline, Practice, Report, Reusable Asset, Roadmap, SupportingMaterial, Template, TermDefinition, ToolMentor, and WhitePaper, but there are no specific elements that have meanings similar to them. Thus, only Text Annotation is the most compatible element in BPMN.

6. Case Study

In this section, we show a model for the entire Level 2, which is one of the test processes defined in Section 4 for case research. Model Level 2 in SPEM using Eclipse Process Framework (EPF) [21], and model it as BPMN along the lines of the mapping table defined in Section 5.

Fig. 5 shows that Test Process represented by Delivery Process in the top-level structure consists of the 'Planning' Phase and the 'Testing' Phase, and a phase consists of the activities. Activity is composed of Task, Role, and Work Product, and shows that the 'Test Execution Tool' is used when performing testing work in the 'Execute Test Procedures' Task and the 'Compare Test Results' Task. Activity, Task, and Work Product (Outcome) were each modeled according to the items defined in **Table 1**, **Table 2**, **Table 3**, and **Table 4**.

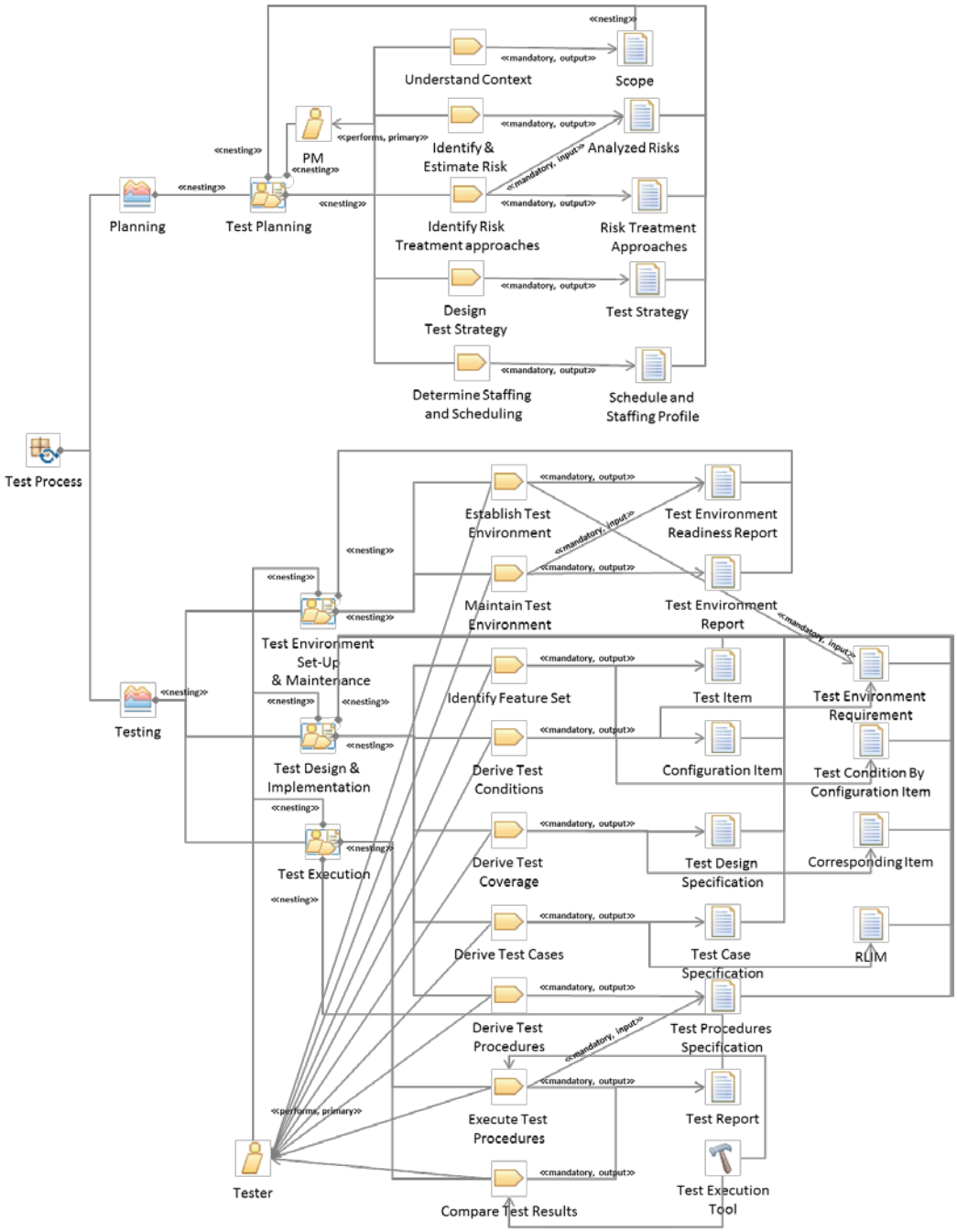


Fig. 5. Test process Level 2 model modeled in SPEM

A model is shown in **Fig. 6** and **Fig. 7** in which the test process represented by the SPEM model in **Fig. 5** is converted to BPMN. In SPEM, items represented as Delivery Process and Phase are represented by business processes. The ‘Planning’ Phase is shown in **Fig. 6** and the ‘Testing’ Phase is represented in **Fig. 7**.

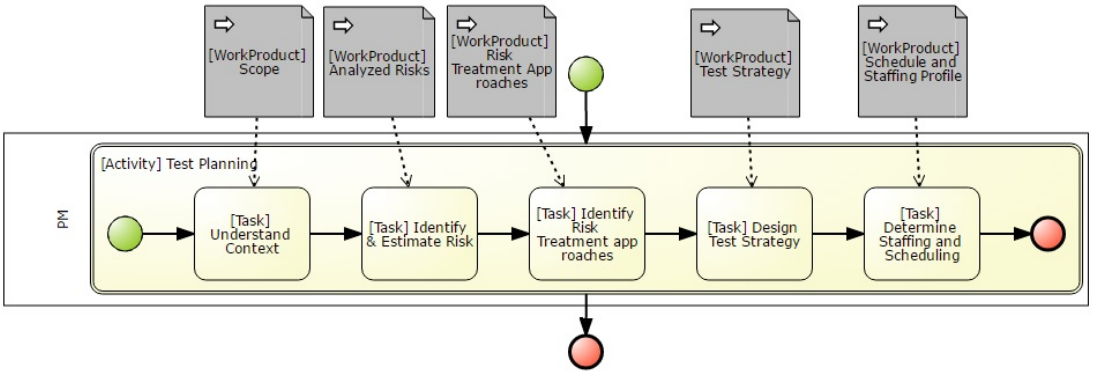


Fig. 6. Planning phase for a test process Level 2 modeled as BPMN

First of all, 'Test Planning' Activity in SPEM is represented as embedded sub-processes: Role as Lane, and Work Product (Outcome) as Data Object. Tasks such as the 'Understand Context' Task are also expressed as tasks in BPMN.

In **Fig. 7**, Activity, Role, Work Product, and Task are converted into embedded sub-processes Lane, Data Object, and Task, respectively, in BPMN. Various activities exist and are expressed in each embedded sub-process, and the several outcomes from the 'Identify Feature Set' Task or the 'Derive Test Conditions' Task are each transformed into a Data Object.

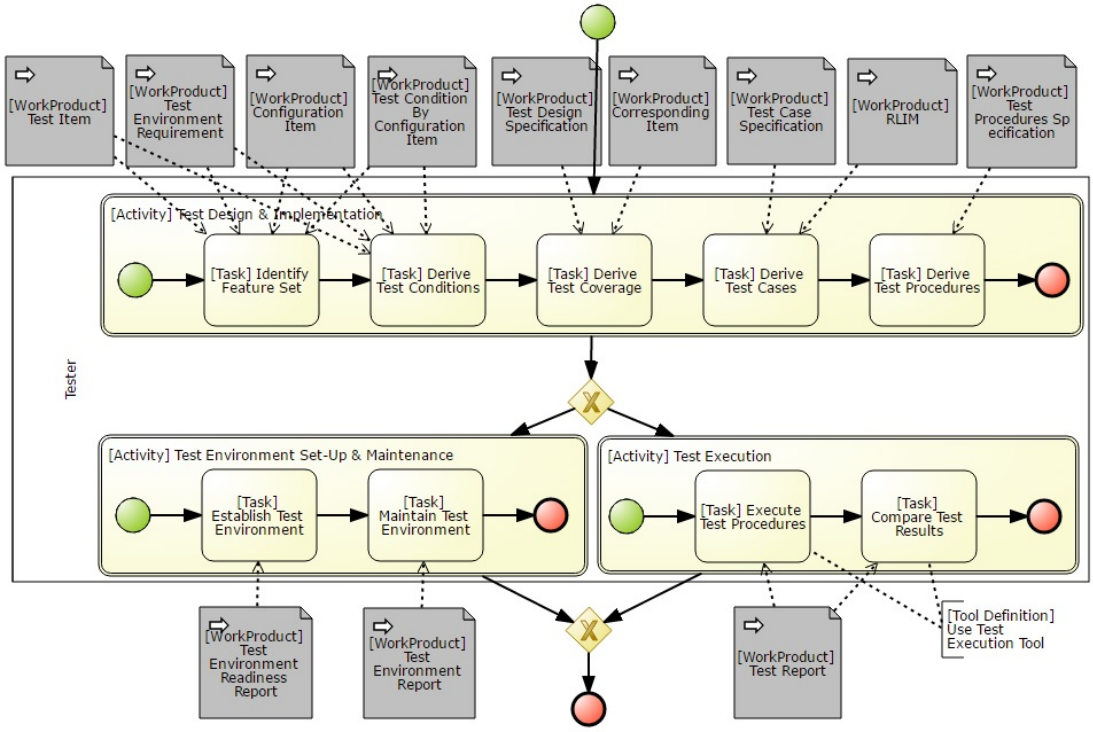


Fig. 7. Testing phase for a test process Level 2 modeled as BPMN

In addition, 'Test Execution Tool' used in the 'Execute Test Procedures' Task and the 'Compare Test Results' Task was converted into a Text Annotation and modeled under

BPMN. The test process converted from SPEM to BPMN can be executed through the BPEL-based workflow engine.

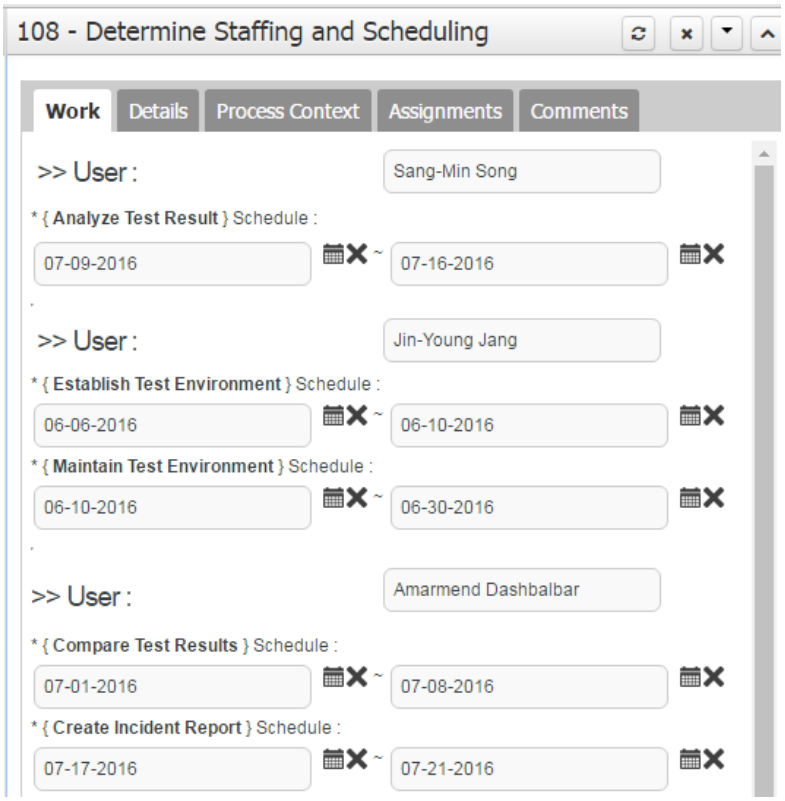


Fig. 8. Translated process in jBPM

After the process is translated to BPMN, it can be executed in any BPEL-based engine. In this paper, we used the Java Business Process Model (jBPM) engine [22], developed by jBOSS, for both modeling and process execution. **Fig. 8** shows the user interface of a task form in jBPM when the process is executed. Users can input results or the output of tasks through the form shown in the figure. Every task in a BPMN process has its own user interface form, and it can be customized.

7. Discussion

We defined a test process for verifying R&D projects in the software field and described it in Section 3. Therefore, the process needs to be executed in order to be used by process participants. However, as mentioned before, the SPEM software process–modeling standard lacks built-in enactment functionalities, and thus, we propose an approach that supports process translation by mapping SPEM to BPMN.

We analyzed both standards and the semantics of every element. However, not every element has one corresponding on the other side, and there are some SPEM elements that cannot be mapped to BPMN, because specific software-related elements cannot be found in the business process. SPEM elements such as MethodConfiguration, PackageSelection, BaseConfiguration, MethodLibrary, MethodPlugin, MethodContentPackage, and ProcessPackage cannot be mapped to BPMN because of the specific content in SPEM.

The test process reflecting maturity level proposed in this paper was compared with Veenendaal's research [23] for quantitative analysis with existing research. Both our paper and Veenendaal's research [23] were compared with ISO 29119-2 and TMMi, and **Table 5** compares a number of items in each standard.

Table 5. Applicable comparison of two standards

	ISO 29119-2		TMMi	
	Process	Activity	PA	SG
Veenendaal [23]	8 (100%)	12 (36%)	6 (38%)	8 (16%)
This Paper	5 (63%)	16 (48%)	6 (38%)	16 (32%)

In Veenendaal's research [23], the number of items in ISO 29119-2 are consist with eight under Process and with 12 under Activity. And the number of items in TMMi consists of six under Process Area (PA) and eight under Specific Goal (SG). The number of items in ISO 29119-2 consists of five under Process and 16 under Activity, and six for PA and 16 for SG in TMMi. And the number of test processes proposed in this paper is less than the number of ISO 29119-2 processes by Veenendaal [23]. Although it shows corresponding items between Process and PA, only two items are included in Process and PA in the details showing corresponding items between Activity and SG. Because Veenendaal's research [23] includes only two items in Process and PA, the ratio including two standards in this paper is higher than for Veenendaal [23].

8. Conclusions

This study introduced a customized software test process and an approach to executing the process by introducing a mapping approach that makes it possible to enact the SPEM process model in BPEL-based workflow engines by translating it to BPMN. We used a software R&D test process in a case study to show our approach's efficiency. The case study shows that SPEM-based software processes can be translated to BPMN and enacted in supported engines.

Currently, process models in SPEM are translated to BPMN manually, and improvement in automating the translation is required. Also, some elements in the two notations cannot be mapped because of conceptual differences between the two standards, where SPEM is a software process–modeling language, and BPMN is for simple business process notation.

In the future, we plan to study automating the transformation process based on mapping by using another OMG standard: Query/View/Transformation. We will also improve the maturity of our testing capability to verify software R&D projects.

References

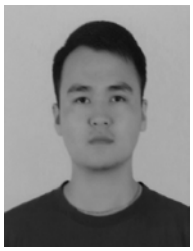
[1] G. J. Myers, T. Badgett and C. Sandler, "The Art of Software Testing, Third Edition," *John Wiley & Sons Inc*, 2015. Article (CrossRef Link).

[2] D. S. Baek, B. J. Lee and J. W. 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. Article (CrossRef Link).

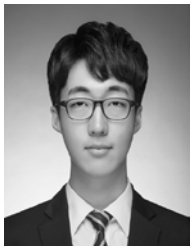
[3] D. S. Baek, J. H. Shin, B. J. Lee and J. W. Lee, "Towards Development of a Traceability Model Measuring Compliance with Guidelines," in *Proc. of the 11th Asia Pacific International Conference on Information Science and Technology*, pp. 37-38, 2016. Article (CrossRef Link).

[4] A. Dashbalbar, E. C. Lee, J. W. Lee and B. J. Lee, "Describing Activities to Verify Artifacts

(Documents and Program) in Software R&D," *Journal of Internet Computing and Services*, vol. 17, no. 2, pp. 39-47, 2016. Article (CrossRef Link).
[5] S. M. Song, A. Dashbalbar, J. W. Lee and B. J. 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. Article (CrossRef Link).
[6] OMG, "Software & Systems Process Engineering Metamodel Specification, Version 2.0," *Object Management Group(OMG)*, 2008. Article (CrossRef Link).
[7] OMG, "Business Process Model and Notation (BPMN)," *Object Management Group(OMG)*, 2011. Article (CrossRef Link).
[8] A. Dashbalbar, S. M. Song, J. W. Lee and B. J. Lee, "Enacting Test Process by mapping from SPEM to BPMN," in *Proc. of the 11th Asia Pacific International Conference on Information Science and Technology*, pp. 223-225, 2016. Article (CrossRef Link).
[9] 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. Article (CrossRef Link).
[10] E. Veenendaal, "Test maturity model integration (TMMi)," *TMMi Foundation*, 2008. Article (CrossRef Link).
[11] E. Veenendaal, R. Grooff and R. Hendriks, "Test Process Improvement using TMM(i)," *Testing Experience: The Magazine for Professional Testers*, vol. 3, no. 19, pp. 21-25, 2008. Article (CrossRef Link).
[12] I. Burnstein, A. Homyen, R. Grom, and C.R. Carlson, "A model to assess testing process maturity," *Crosstalk The Journal of Defense Software Engineering*, vol. 11, no. 11, pp. 26-30, 1998. Article (CrossRef Link).
[13] I. Burnstein, S. Taratip, and C. Robert, "Developing a testing maturity model for software test process evaluation and improvement," in *Proc. of Test Conference. International*, pp. 581–589, 1996. Article (CrossRef Link).
[14] 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. Article (CrossRef Link).
[15] N. Debnath, F. A. Zorzan, G. Montejano and D. Riesco, "Transformation of BPMN subprocesses based in SPEM using QVT," in *Proc. of 2007 IEEE International Conference on Electro/Information Technology*, 2007. Article (CrossRef Link).
[16] C. Portela, A. Vasconcelos, A. Silva, A. Sinimbú, E. Silva, M. Ronny, W. Lira and S. Oliveira, "A Comparative Analysis between BPMN and SPEM Modeling Standards in the Software Processes Context," *Journal of Software Engineering and Applications*, vol. 5, no. 5, pp. 330-339, 2012. Article (CrossRef Link).
[17] "ISO/IEC/IEEE 29119 Software Testing Standard," International Organization for Standardization, 2013. Article (CrossRef Link).
[18] R. Rakitin, "Software verification and validation for practitioners and managers," Artech House Inc., 2001. Article (CrossRef Link).
[19] K. Georg, and M. Kuhrmann, "Criteria for software process tailoring: a systematic review," in *Proc. of the 2013 International Conference on Software and System Process. ACM*, 2013. Article (CrossRef Link).
[20] D. S. Beak, B. J. Lee and J. W. Lee, "Relevance Analysis System Design based on Content of Software Research and Development Document Artifacts," In *Proc. of the 10th Asia Pacific International Conference on Information Science and Technology*, pp. 125-126, 2015. Article (CrossRef Link).
[21] P, Haumer, "Eclipse process framework composer," Eclipse Foundation, 2007. Article (CrossRef Link).
[22] "jBPM 6.3 Documentation," JBoss, 2015. Article (CrossRef Link).
[23] E. Veenendaal, "TMMi and ISO/IEC 29119: Friends or Foes?," TMMi Foundation, 2016. Article (CrossRef Link).



Amarmend Dashbalbar, majored in Computer Science at Mongolian University of Science and Technology and received B.S in 2013. He is currently master's student at Computer Science department of University of Seoul. His research areas of interest include software engineering and software testing.



Sang-Min Song, received the B.S. degree in Game Programming at Academic Credit Bank System, Korea in 2015. He is currently studying toward the M.S. degree in Computer Science at University of Seoul, Korea. His research areas of interest include software engineering, test process and software testing.

Jung-Won Lee, is an associate professor of the Department of Electrical and Computer Engineering at Ajou University, Korea. She received her PhD. Degree in Computer Science and Engineering from Ewha Womans University, Korea, in 2003. She was a researcher of LG Electronics and did an internship in the IBM Almaden Research Center, USA. Her areas of research include context-aware, embedded software and software engineering.

Byungjeong Lee, He received the B.S., M.S., and Ph.D. degrees in Computer Science from Seoul National University in 1990, 1998, and 2002, respectively. He was a researcher of Hyundai Electronics, Corp. from 1990 to 1998. Currently, he is a professor of the Department of Computer Science and Engineering at the University of Seoul, Korea. His research areas include software engineering and web science.