A Strategic Test Process Improvement Approach Using an Ontological Description for MND-TMM

Hoyeon Ryu[1], Dong-Kuk Ryu[2], Jongmoon Baik[1]
[1]*School of Engineering, ICU*
119 Munjiro, Yuseong-Gu, Daejeon, 305-714, Korea
{hoyeon, jbaik}@icu.ac.kr
[2]*Agency for Defense Development*
Mt. 25, Geoyeo-dong, Songpa, Seoul 138-600, Korea
dkryu@add.re.kr

Abstract

It is well known that most of mission critical systems like defense systems demand higher quality than general software systems. In order to assure the high quality of the systems, those software systems must be thoroughly tested with well defined testing processes. As software systems evolve, software testing should also be improved with continual-base. Even if there are some previous researches on test process model, those models have some limitations and problems to be applied for software testing process improvement in a specific domain like military weapon systems. In this paper, we propose a new testing maturity model, called MND-TMM(Ministry of National Defense-Testing Maturity Model), which can be adopted effectively for the weapon software system development. We also present OWL ontology for the MND-TMM, called MTO(MND-TMM Ontology) to help software organizations set up their effective testing strategies and improve their testing processes, which can lead to high quality product developments.

1. Introduction

Most of defense systems require higher quality than other general software systems do because of their mission criticalities, and the possibility that failures in any of these systems could potentially cause disasters such as the loss of human lives, the loss of strategically important positions against enemies, or the loss of billions of dollars. In the military domain, Reliability, Security, and Interoperability are mainly emphasized quality attributes. As the size of systems becomes larger and more complex, much intensive and effective testing techniques must be performed. much more mature test processes need to be applied to operate those systems in their specific operational environments, with consideration of different quality attributes, such as reliability, security and interoperability, etc. as well. In order to assure the qualities of those systems, defense systems should be tested according to well-defined and mature test processes. In addition to it, those processes must be improved continuously for the system maintenance and evolution.

High quality software systems can be produced by improving both the process quality and product quality. The software process models, such as CMM[1], CMMI[2], SPICE[3] provide guidelines to improve the software development processes. The quality assurance activities, such as formal reviews, inspections and software testing, can help to guarantee the software product quality, such as reliability, availability, and maintainability, etc.

In particular, software testing is regarded as one of the critical tasks in the software development process to check the conformities to specifications and customers' requirements. Without effective testing, software organizations cannot be sure that they will deliver the required quality products to their customers. To guarantee high quality software products, they must be tested according to well-defined test processes, and those processes must be improved continuously.

Although process models have been developed to improve the software development process including the test process, they do not deal sufficiently well with issues in all test process areas. In order to solve this problem, many test process models, such as TMM[4], TIM[5], and TPI[6] models, have been developed. However, they have several limitations which make their applications to the test process improvements in a specific domain like a defense domain restrictive. In other words, these test process models have difficulties and complex structures for organizations to adopt. They also have too many process areas and practices on which software organizations must focus on for their process improvements. To analyze the gaps in

978-0-7695-3131-1/08 $25.00 © 2008 IEEE
DOI 10.1109/ICIS.2008.78

focused test process areas, those models require the experts' process assessment. They are also difficult to be tailored to a specific testing environment.

In this paper, we propose a new testing maturity model and its ontological description, called MND-TMM and MTO. The MND-TMM (Ministry of National Defense-Testing Maturity Model) was developed for testing process improvements in the defense domain, and help software organizations produce more reliable weapon systems. The MTO can help software organizations to set up an effective testing strategy, and improve their test processes to produce high quality products via self-assessment. It also supports test process tailoring via sharing the knowledge of the experts and other process models.

The rest of the paper is organized as the following. Section 2 describes why we suggest the MND-TMM and why use ontology for the model. In Section 3, we introduce a new testing maturity model, MND-TMM, for testing process improvements for weapon software system developments in the defense domain. The MTO is explained in Section 4. Finally, we describe the conclusion and future works.

2. Problem and Motivation

Previous process maturity models have some limitations when applied to weapon systems. In this section, we describe the analysis results based on the limitations of previous process maturity models. We also depict why both MND-TMM and its ontology were developed.

2.1. Limitations of Process Maturity Models

2.1.1. CMM and CMMI. CMMs(Capability Maturity Models) are widely used improvement approaches. They provide software organizations with guidance on how to gain control of their software system development processes. The CMM(Capability Maturity Model)[1] was developed to guide organizations on how to select an improvement strategy and how to identify critical elements regarding software quality and process improvements. It can also be used to assess an organization's process capability against a scale of five process maturity levels. CMM has 5 maturity levels, which contain 18 key process areas(KPAs). Except for Level 1, each maturity level has several key process areas on which an organization should focus to improve its process capability.

After the CMM was introduced, many similar maturity models have been developed. There is a lack of coverage for the whole life cycle in the CMM. Even if there are some overlaps among those models, software organizations have made lots of effort to implement those models within their organizations for process improvements. The CMMI (Capability Maturity Model Integration)[2] was developed to integrate process improvement models of systems engineering (SE), software engineering (SW), integrated product and process development (IPPD), and supplier sourcing (SS). There are two kinds of representation models in CMMI: Staged Representation and Continuous Representation. In Staged Representation, there are 5 maturity levels from the lowest level 1 to the highest level 5, as in the CMM. In Continuous Representation, there are 6 capability levels from the lowest level 0 to the highest level 5. There are 25 process areas in CMMI, including Verification, Validation, Product and Process Quality Assurance, and so on.

Although the CMM and CMMI have been developed to improve the entire software process including the test process, they are not enough to support the improvements of test process areas sufficiently, since they have some limitations when dealing with detailed test process areas and practices. For example, CMMI have the PA like V&V to deal with test process improvement. However, The PA does not deal with the testing process areas in more detail such as testing environment, testing technique training, and operation of testing organizations. And organizations have difficulties to understand them because of their complex structures.

2.1.2. TMM and TPI. The TMM (Testing Maturity Model)[3] was developed by the Illinois Institute of Technology(IIT) in 1996. One of the main reasons why the TMM was developed was that existing maturity models like the CMM did not adequately address testing related issues. In addition, the nature of a mature testing process was not well defined. The main purpose of the TMM is to support the test process assessment and improvement within an organization. Like the CMM, the TMM uses the idea of maturity levels for process evaluation and improvement. It provides 5 stages of maturity level, which is similar to the CMM and CMMI. It also provides an evolutionary path, and each maturity level consists of several key areas which have to be satisfied for archiving that level, and each consists of several goals.

There are several limitations of the TMM. First, because it was developed as a supplementary model of the CMM, it will be very efficient when it is used alongside the CMM. However, it is difficult to use only the TMM. It is also difficult to integrate its use with other models. Second, key areas are very broad and not easy to achieve. Third, the TMM does not have important key areas such as Test Environment, Office

Environment, Reporting, Defect Management, and Testware Management, which are very important and addressed in the TPI Model.

The TPI(Test Process Improvement)[5] was developed by Martin Poll and Tim Kooman in 1997. It is a maturity model that supports test process improvements. TPI consists of four main components, Key Areas, Levels, Checkpoints, and Improvement Suggestions. Key areas and Levels can be represented in the form of a Test Maturity Matrix. The TPI has 20 specific and detailed Key Areas covering the entire process associated with testing activities.

The TPI model provides practical guidelines to assess the testing maturity level for an organization. It helps a software organization improve their processes incrementally. The TPI was built on the practical knowledge and experiences of the test process developments in real practice. It supports several key process areas that are not addressed in the TMM, such as Test Environment, Office Environment, Reporting, Defects Management, and Testware Management.

Although the TPI helps to develop an Improvement Strategy for testing processes through the entire software development life-cycle, there are some weaknesses in the TPI. Those would be briefly explained as follows; First, the TPI is not compatible with the CMM/CMMI and TMM because it does not have the same structure of the five levels as the others do. Second, there are no checking items in the Matrix to evaluate several state-of-the-art testing techniques after they were introduced.

2.2. Ontology and Its Usage

An ontology is an explicit specification of concepts in a domain, which has been widely used in the AI community[7]. An ontology consists of three primitives, (concepts, relations, and axioms), in order to define concepts and their relations. It became a very useful and prevailing representation for the following reasons [8]:

- It makes it possible to share a common understanding of the structure of information among people or software agents.
- It encourages the reuse of domain knowledge.
- It enables domain assumptions to be made explicit.
- It separates domain knowledge from operational knowledge.
- It makes it easy to analyze domain knowledge.

Many ontology libraries in a specific domain or a generic domain have been applied to various application areas, such as the semantic web, intelligent information integration, information retrieval, or knowledge-based systems, etc. Ontology should be helpful and intelligent both to computers and humans. In the software engineering field, we can obtain the reasonable benefits of ontology. Ontologies in software engineering introduced in [10] have two parts; first, the ontologies are used in representing the software engineering processes, the others are ontologies in the different software engineering domains. Ontologies in software engineering processes provide the core knowledge of software engineering about the activities and artifacts for software engineers. The primary goal of ontologies is to share the knowledge about processes and experiences from previous software development experiences. They represent several variants of ontologies, such as the SPO(Software Process Ontology)[12], and GPO(General Process Ontology)[11]. There is much research about ontology based on process modeling[13][16], and software environments[14][18].

3. MND-TMM: New Testing Maturity Model for Weapon System Development

A testing maturity model can serve as a guide to organizations for test process improvement and assessment. We developed a testing maturity model, called the MND -TMM, to improve test processes and assure the quality of weapon systems. The MND-TMM can be used to:

- Baseline the current testing process capability level of maturity.
- Identify the testing process areas that can be improved.
- Provide the best practices for test process improvements.
- Measure the effects of testing improvement efforts.

3.1. Framework of MND-TMM

The MND-TMM has the five levels like TMM and four categories - Military, Process, Infrastructure, and Techniques. Each category consists of several Test Process Areas (TPAs). The MND-TMM is similar to a continuous representation of the CMMI in that it is organized into four categories and select levels for improvements based on the corresponding TPAs. All TPAs are not spread throughout the 5 levels because some TPAs are not applicable to some entire levels.
Figure 1 illustrates the five maturity levels of the MND-TMM. Each maturity level in the MND-TMM is described below:

- Level 1: An organization recognizes the necessity of testing, but the organization's testing process is

not established. Some testing activities, like simple debugging, are performed unsystematically. This level does not have any KPAs in the category of Process.

- Level 2: An organization understands the test process and necessary test activities. It has a basic and immature test process. Also, the test process is repeatable for projects, and a failure cost concept is applied to the testing process.
- Level 3: The test process is defined as a standard for an organization and it is used across all projects after being tailored to reflect various and different characteristics of the project.
- Level 4: The test process is measured and managed quantitatively. The performance of the testing process is analyzed statistically using the collected metrics on the testing processes. According to the results of the analyses, some corrective actions are taken.
- Level 5: The test process is focused on defect prevention and continuous process improvements. The test process is optimized to the organization and the product, and process quality is under control.

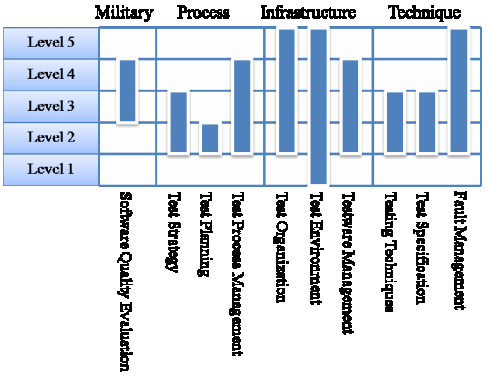

Figure 1. Maturity Levels of the MND-TMM

The MND-TMM is comprised of total 10 TPAs regarded as important process areas which must be emphasized for test process improvements. All TPAs are broken into four categories: Military(M), Process(P), Infrastructure(I), and Techniques(T). The Military category contains TPAs related to characteristics (Systems of Systems, Mission Critical Systems, Embedded Systems, Software Intensive Systems, and Interoperable Systems) in the military domain, and quality requirements for weapon systems-software quality evaluation. The Process category includes TPAs related to the test process and its artifacts, management-test strategy, test planning, and test process management. The Infrastructure category has TPAs associated with the test organization and the testing supporting environment-test organization, test environment, testware management. The Techniques category includes TPAs related to testing specification, design, and testing techniques, test specification, fault management. Each TPA can be applied independently to its own specific level, regardless of other TPAs which are chosen by the organization. Each TPA consists of Specific Goals which must be satisfied for the TPA. The MND-TMM has 32 specific goals and 96 specific practices. In this paper, we do not describe the details on those specific goals and practices since it is beyond the scope of the paper.

3.2. Structure of MND-TMM

A TPA is comprised of activities to achieve a specific maturity level. Each TPA of the MND-TMM has Specific Goal(SG)s and Generic Goal(GG)s. An SG describes the unique characteristics that must be presented to satisfy the TPA and is used in assessment to help determine whether a TPA is satisfied or not. An SG has some Specific Practices(SP). Each SP contains several Sub Practices. SPs describe activities to achieve the SGs of a TPA. A Sub practice is a detailed description that provides guidance for interpreting and implementing a SP. A GG describes the characteristics that must be applied to all TPAs and is used in assessment to determine whether a TPA is satisfied or not. A GG has Common Features which stand for maturity characteristics of each level. Figure 2 shows the structure of MND-TMM.

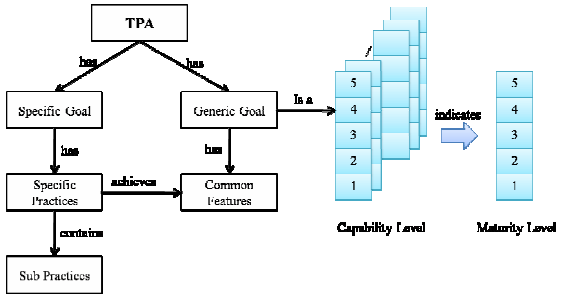

Figure 2. A structure of the MND-TMM

For instance, a Test Planning TPA in Process category helps the organization document the test plan in software development life cycle. It has 2 SGs and 8 SPs. This Test Planning TPA enable an organization establish the test plan and test strategy with the defense standard such as Defense Component-based development Methodology MND Software Development Process, etc. Figure 3 shows the summary of Test Planning TPA.

3.2.2 TPA P.2 – Test Planning

Purpose

The primary goal of this TPA is to set a test plan which is included in the overall test plan and software development life cycle and must be performed at each level.

Description

Test plan is the plan which describes the activities performed at a specific level. An organization must document the test plan detailed with a test strategy according to the software process life cycle.

Related TPAs

Test Strategy

References

Defense Component-based development Methodology MND Software Development Process

IEEE Std 829 Software Test Documentation

Specific Goals

SG1 Identify Software Process Life Cycle

SG2 *Document a overall test plan*

Figure 3. The Structure of Test Planning TPA

4. Ontological Description of the MND-TMM

4.1 MTO and Its Description in OWL

An ontology consists of concepts and relations among the concepts, such as inheritance relations and semantic relations. It is formalized in OWL (Web Ontology Languages)[17] standardized by the W3C (WWW Consortium). A class can be defined by extracting concepts from related terms among classes. There are three approaches such as top-down, bottom up, and hybrid approaches to define class hierarchies. A top-down method is used for the class hierarchy development in this work. Namely, a super-class is defined first by extracting the general concepts from related terms, and the following subclasses are defined by extracting the specific concepts from related terms. The internal structure of concepts is described by defining properties of a class (classes). All subclasses inherit the properties from super classes in the hierarchical structure. Therefore, most general properties should be located in the top class. The properties of classes are data types for general data, and object properties for the relations to other classes.

A class hierarchy of the MND-TMM and a part of its OWL description using Protégé ontology editor[18] is shown in Figure 4.

Each class and relations extracted from the MND-TMM and OWL expressions for logical relations are added. There are five top-level classes such as Maturity Level, Category, Goal, Practice and Process Area. Relations among them represent semantics. These top-level classes have their subclasses. In general, super classes and sub classes are expressed as the relation of is-a or kind-of. There are some relations represented by object properties such as; has_Infrastructure_Goal, has_Techniques_Goal, has_Process_Goal, has_Military_Goal, has_Generic_Goal, etc. The OWL description in Figure 4 shows a part of the MTO. There are classes (Category, Generic_Goal, etc.), object properties (has_Generic_Goal, has_Infrastructure_Goal, etc.), and OWL logic expressions (InverseFunctionalProperty).

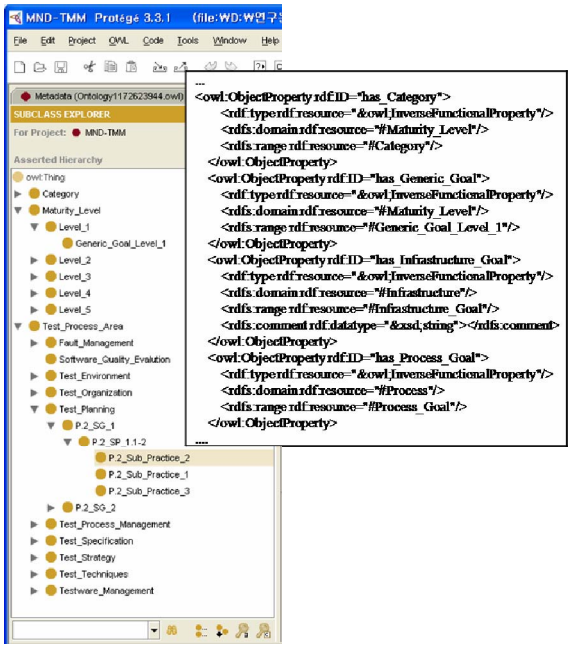

Figure 4. MTO and its OWL description in Protégé

4.2 A Contribution of the MTO

The main contributions of the presented ontological description of the MND-TMM, the MTO can be summarized as follows and the MTO can use the test process improvement strategically:

- The MTO provides a better understanding and sharing of core knowledge about MND-TMM for software development organizations.
- The MTO supports test process improvements and test process tailoring via sharing the knowledge of experts and other process models.
- The MTO enables an organization to use it with other ontologies, such as the CMMI-SW ontology[9], Software Process Ontology(SPO), Software Measurement Ontology, and Domain Specific Ontology, etc.
- The self-assessment based on MTO using OWL expression and its inference enable organizations find out problems in their test processes, and provide them with guidelines of test process improvements.

- The MTO enables organizations to use the test maturity model effectively and easily for test process evaluation, without professional assessors, since it contains knowledge of the domain and expertise about the test maturity model.

5. Conclusion and Future Work

In this paper, we introduced the MND-TMM and suggested its ontology for effective use and execution of the test maturity model. The MND-TMM is the test process maturity model that can help software organizations to implement effective testing, and improve their testing processes for high quality products. The MND-TMM has the five maturity levels and four categories which have the corresponding TPAs.

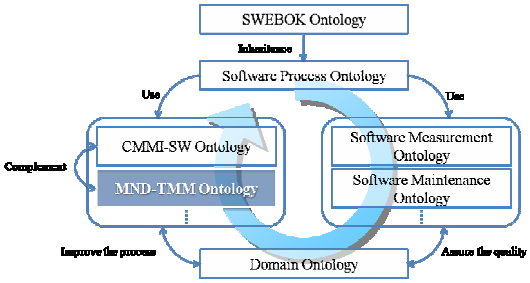

Figure 5. A framework for reliable, high quality software systems

The ontology consists of concepts and relations formalized in OWL. The top level concepts of ontology for the MND-TMM are five; Maturity Level, Category, Goal, Practice and Process Area. And relations among them represent semantics.

The detailed structure of the MND-TMM and its detailed ontologies are under development. The MTO should be pursued to the point where it links up with cooperation with other ontologies, and usage of a web based assessment tool for software organizations. Figure 5 shows a framework of ontologies for reliable and high quality software system developments. The initial MND-TMM can be tailored by sharing knowledge from the ontologies for other domains in software engineering. Therefore, we will work on the integration of the initial ontology with other ontologies to provide more effective and efficient test process improvements.

6. Acknowledgement

This work was partially supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract(2008-SW-11-IM-04).

7. References

[1] CMU/SEI, Capability Maturity Model for Software, Version 1.1, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, 1993.

[2] CMU/SEI, Capability Maturity Model Integration (CMMI) Version 1.1 - Staged Representation, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, 2002.

[3] Alex Dorling, "SPICE: Software Process Improvement and Capability dEtermination", *Software Quality Journal*, 1993, pp.209-224.

[4] I. Burnstein, A. Homyen, A. Suwanassart, G. Saxena, R. Grom: "A Testing Maturity Model for Software Test Process Assessment and Improvement", *Software Quality Professional*, September 1999, pp. 8- 21.

[5] Ericson, T., Subotic, A., Ursing, S., "TIM-A Test Improvement Model", *Software Testing, Verification and Reliability,* 7(4), 1997, pp. 229-246.

[6] Koomen, T., Pol, M., *Test Process Improvement: A practical step-by-step guide to structured testing*, Addison-Wesley, 1999.

[7] Gruber, T. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Technical Report KSL-93-04. Knowledge Systems Laboratory. Stanford University, CA. 1993.

[8] Natalya Fridman Noy, Deborah L. McGuinness. "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05, March 2001.

[9] Kokar Soydan, "An OWL Ontology for Representing the CMMI-SW Model", SWESE2006, 2006.

[10] Hans-Jörg Happel and Stefan Seedorf, "Applications of Ontologies in Software Engineering", SWESE 2006, 2006.

[11] Lin, Y., Strasunskas, D., "Ontology-based Semantic Annotation of Process Templates for Reuse", EMMSAD'05, June 2005.

[12] Li Liao, Yuzhong Qu and Hareton K. N. Leung, "A Software Process Ontology and its Application", ISWC 2005, November 2005.

[13] Ceravolo P., Damiani E., Marchesi M., Pinna S., Zavatarelli F., "A ontology-based process modeling for XP", APSEC2003, December 2003, pp. 236-242.

[14] M. A. Musen, "Domain Ontologies in Software Engineering: Use of protégé with the EON Architecture", *Methods of Information in Medicine*, 37, 1998, pp. 540-550.

[15] Fabiano Borges Ruy, Gleidson Bertollo, and Ricardo de Almeida Falbo, "Knowledge-based Support to Process Integration in ODE", *CLEI Electron. J*, 7(1), 2004.

[16] Jari Palomäki, Harri Keto, "A Process-Ontological Model for Software Engineering", PHISE'06, 2006.

[17] W3C. "OWL Overview," http://www.w3.org/TR/2003/CR-owl-features-20030818/.

[18] The Protégé Ontology Editor and Knowledge Acquisition Systems, http://protege.stanford.edu/.