A Set of Measurements to Improve Software Testing Process

Taratip Suwannasart[1]and Prapass Srichaivattana[2]
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Bangkok 10330, Thailand
E-mail: staratip@chula.ac.th[1], prapass20@hotmailcom[2]

Abstract: A set of measurements related to test-related data is developed to be included in the Testing Maturity Model (TMM). The objective of measurements is to track testing practices status in each level of the TMM with the exception of level one. We use the Goal/Question/Metric paradigm to derive the set of measurements. The set of measurements will help software development organizations improve their testing processes. Future research will be focused on developing a tool to collect the measurements and evaluating the usefulness of the measurements.
Key Words: Software testing, TMM, Measurement, Metrics

1. Introduction

Developing quality software is a goal of every software development organization. To achieve the goal, an organization must have a mature testing process. Since testing is key activities related to software quality. A Testing Maturity Model (TMM) has been developed to help software development organizations evaluate, and improve their testing processes. The TMM guides the organizations improve their testing processes by recommending a set of testing practices at each level of maturity. The details of TMM development can be found in [1].

In this paper we propose a set of recommended measurements to be included in the TMM. The proposed set of measurements assures that the organizations perform the set of recommended practices. Moreover, the measurements will show the status of the testing practices performed. A major goal of the measurements is to guide which test-related data to be collected at each level of the TMM (except level one). These quantitative data will be serve as a foundation for managing and tracking testing process, and a baseline for collecting organization's historical test-related data.

This paper we discuss an overview of the TMM. How to define the set of recommended metrics is then described. The set of recommended measurements of each maturity level is discussed. Summary and future research are then outlined.

2. An Overview of The TMM

The TMM development was guided by four sources: the Capability Maturity Model (CMM) version 1.1 [2], Gelperin and Hetzel's evolutionary testing model [3], industry testing practices [4], and Beizer's progression phases of a tester's mental model [5]. The CMM is a comprehensive process improvement model developed by the Software Engineering Institute (SEI) that has been accepted by the software industry. The Gelperin and Hetzel's testing model serves as the foundation for level differentiation in the TMM. The industry testing practices illustrates the testing practices and environments in the software industry. The Beizer's model gives the individual tester's thinking process.

The TMM version 1.0 has 2 major components[6,7]: a set of levels and an assessment model. The details of each component are described as follows:

2.1 A Set of Levels

The TMM is characterized as five maturity levels. The TMM levels define a position in a testing maturity hierarchy. Characteristics of each level are described in terms of testing capabilities and organization goals. With the exception of level one, each level has the framework of maturity goals, maturity subgoals, activities, tasks, and responsibilities. The structure of the TMM is shown in figure 1.

The set of maturity goals identifies key process areas that must be addressed to improve testing capability. The maturity goals of each level (except level one) are illustrated in figure 2. The maturity subgoals support one or more maturity goals and define the scope, boundaries, and needed accomplishments for a particular level. Activities, tasks, and responsibilities are organized by three critical views that represent each of the key participants involved in the testing process: managers, developers and testers, as well as users and clients.

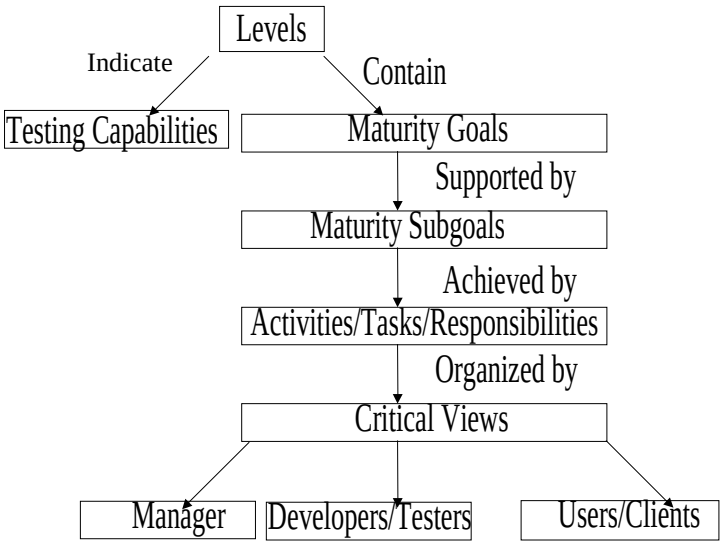

Figure 1: The Structure of the TMM

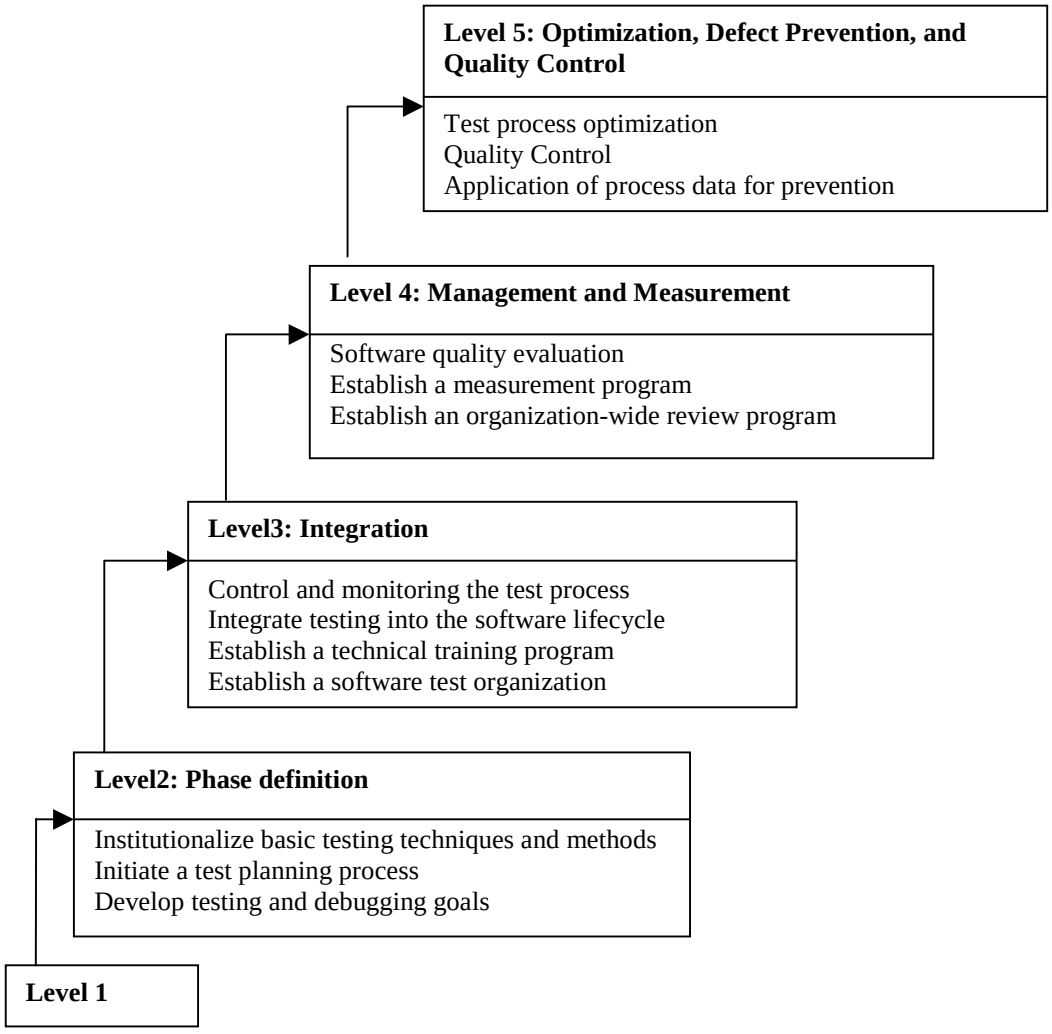

Figure 2: The Maturity Goals by Level

2.2 The Assessment Model

The TMM Assessment Model (TMM-AM) [8] can be used by software organizations to assess and improve their testing process capabilities. The TMM-AM consists of three components:

- The assessment procedure: provides a series of steps to be followed by an organization that needs to carry out a testing process self assessment.
- A questionnaire: is the assessment instrument that helps collect and record assessment information.
- Assessment training and team selection criteria: The assessment training provide a preparation of the assessment team to conduct the self-assessment. The assessment team members are selected based on their understanding of the assessment goals, knowledge, experiences, skills, and commitments to test process improvement.

Following the TMM assessment procedure, an organization obtains the outputs of the assessment as follows:

- TMM level (on a scale from 1 to 5)
- The organization testing process strengths and weaknesses
- The Test process profile (a summary of the organization's testing process)
- Action plans for testing process improvement
- Assessment record

3. How do we derive A Set of Measurements?

We use Goal/Question/Metric (GQM) paradigm [9] to derive a set of measurements for each level of the TMM. The GQM is developed to counter the problem that many software metric programs have failed because they have poorly defined, or even non-existent objectives. Originally, it was used for evaluating defects for a set of projects in the NASA/GSFC environment. Because of its intuitive nature, the approach has gained widespread appeal. According to the GQM, there are three stages as follows:

1. Set goals specific to needs. Goals may be defined for any object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment.
2. Refine the goals into quantifiable questions that are tractable.
3. Deduce the metrics and data to be collected to answer the questions.

In this research, goals will be the maturity goals of each level. Questions will be formulated to achieve the goals. Then we derive the set of measurements to answer those questions. The details of each question can be found in [10]. In this paper, we present only the maturity goals and what to be measured in order to achieve the goals.

4. A Set of Measurements

As we discussed in the section 3 that we use the GQM paradigm to derive a set of measurements for the TMM. This section we describe brief characteristics of each level of the TMM with the exception of level 1, maturity goals along with their brief practices, and a set of recommended measurements.

4.1 Level 2: Phase Definition

The testing goal at this level is to demonstrate that the software product satisfies its specifications and requirements. Testing and debugging concepts are still not clearly defined. Execution is considered as a primary testing activity. There is some test planning, but it is done after the code is developed. This level consists of 3 maturity goals, which are described below. A set of measurements of this level is shown in Table 1.

Goal 2.1: Develop testing and debugging goal

In order to distinguish the differences between testing and debugging process, an organization should separate their cost.

Goal 2.2: Initiate a test planning process

The project manager should track status of developing the test plan and define testing activities, i.e. cost and schedule.

Goal 2.3: Institutionalize Basic Testing Techniques and Methods

In order to know the status of using basic testing techniques and methods, the organization may track outputs from their uses, i.e. test cases.

Goals	A Set of Measurements
2.1	Testing cost Debugging cost
2.2	Cost in developing test plans Test schedule
2.3	Numbers of test cases derived from test techniques: black-box, white-box

Table 1: A Set of Measurement at level 2

4.2 Level 3: Integration

An organization at this level has as its primary testing goal: the ability to detect faults in order to demonstrate that the program is error-free and satisfies its requirements. The concept of testing and debugging are clearly defined. Tests are designed early to force the organization to be concerned with testing at the earlier phases of the software development process. The maturity goals of this level are described below and a set of measurements is shown in Table 2.

Goal 3.1: Establishing the Software Test Organization

In order to know the status of the test organization, its manpower and cost should be tracked.

Goal 3.2: Establish a Technical Training Program

The organization should provide a suitable quantity of test training to testers and SQA groups, i.e. the number of test-related courses per year.

Goal 3.3: Integration of Testing into the Software Life Cycle

In order to track the results of integration of testing into the software development life cycle, the organization may use defects removed (found), defects injected (origin) and defects escaped during testing in software development life cyclein order to derive the effectiveness of integration.

Goal 3.4: Controlling and Monitoring the Testing Process

For activities defined in test plans, the organization should control and monitor to see if they are performed as planned. Test cases are very important in the testing process, so the organization should monitor their running status such as the number of test cases passed and failed. The organization should monitor the quality of unit testing by measuring the test coverage: statement, branch and basis path coverage. Moreover, the status of integration testing should be tracked

Goals	A Set of Measurements
3.1	Numbers of independent testers Cost of test organization
3.2	Numbers of test-related courses offered per year
3.3	Numbers of defects found by phase
3.4	Schedule slippage Numbers of test cases passed Numbers of test cases failed Numbers of tested modules % statement coverage % branch coverage % basis path coverage Numbers of integrated and tested modules

Table 2: A Set of Measurements at Level 3

4.3 Level 4: Management and Measurement

The primary goal of testing is to detect requirements, design, and implementation faults. The testing activities start at the earliest phases of software development life cycle. Testing is parallel, and not sequential to software implementation. The testing life cycle is fully coordinated with the software development life cycle and the review process. The maturity goals are described below and a proposed set of measurements at this level is shown in Table 3.

Goal 4.1: Establish an Organization-wide Review Program

The organization should control the effectiveness of the review process. The number of defects removed by reviews is measured to indicate the effectiveness.

Goal 4.2: Establish a Test Measurement Program

The TMM mentions that the organization should collect details about defects, such as their severity and type. Furthermore, the productivity of testers should be measured.

Goal 4.3: Software Quality Evaluation

The organization should define quality attributes of a software product and measure these attributes.

Goals	A Set of Measurements
4.1	Numbers of defects removed by reviews
4.2	Numbers of defects founded in each severity Numbers of defects found in each defect type Testers' productivity
4.3	Correctness Efficiency Flexibility Interoperability Maintainability Portability Reliability Reusability Testability Usability

Table 3: A Set of Measurements at Level 4

4.4 Level 5: Optimization/Defect Prevention and Quality Control

The primary goal of an organization's testing activities of this level is defect prevention. Prevention applies to requirements, design and implementation faults. At this level, we test to demonstrate that the software satisfies its specification, to detect faults, and to prevent faults. This level consists of 3 maturity goals, which are described below. The Table 4 shows a set of measurements at this level.

Goal 5.1: Application of Process Data for Defect Prevention

The organization should track activities of the defect prevention, i.e. cost of finding root causes of defects.

Goal 5.2: Quality Control

The organization should define the lower control limit (LCL) and the upper control limit (UCL) of metrics in the control chart to achieve statistical process control. Good indicators of testing process, that should be controlled, are as follows defect density, review effort rate, test effort rate, % statement coverage, % branch coverage, and % basis path coverage.

Goal 5.3: Test Process Optimization

The organization should track the status in evaluating new testing tools and technologies. The TMM mentions that the organization should determine when to stop testing. One possible way is to compare the goal and actual failure rate. If the differences are acceptable, the software is ready to be released.

Goals	A Set of Measurements
5.1	Total effort in finding root causes of defects
5.2	LCL and UCL for defect density, review effort rate, test effort rate, % statement coverage, % branch coverage, and % basis path coverage
5.3	Total effort in evaluating new testing tools and technologies Total effort in defining when to stop testing Failure rate goal Actual failure rate

Table 4: A Set of Measurements at Level 5

5 Summary

This paper presents a set of recommended measurements for each maturity goal of the TMM version 1.0. The set of measurements is derived by the GQM approach. It reflects an organization's testing process and product so that the organization can use the collected data to improve the testing process effectively. What we propose here is only a minimum set of measurements which we believe that they are enough to reflect an organization's testing process. We define only what to be collected, not how to collect. The characteristics of each measure and a full set of measurements can be found in [10].

For future research, we propose the following:

1. Characteristics of software quality metrics in the level 4 are not clearly defined. A Framework to measure them needs to be developed.
2. A tool for gathering data sources to calculate each measure will be developed. The tool will help an organization to analyze and report test-related data. It also should be developed to be able to interface with other testing tools such as test coverage tool, test case data generator tool, and project management tool.
3. The proposed set of measurements should be used and evaluated by software development organizations. Feedback from them will help us to improve the TMM.

6 References

[1] Burnstein, I., Suwannasart, T., Carlson, C.R., "Developing a Testing Maturity Model for Software Test Process Evaluation and Improvement," IEEE International Test Conference, Washington D.C., October 20-25, 1996, pp. 581-589.

[2] Paulk, M., et al. " Key Practices of the Capability Maturity Model, Version 1.0," Technical Report CMU/SEI-93-TR-25, Software Engineering Institute, Pittsburgh, Pa., 1993.

[3] Gelperin, D., and Hetzel, B., "The Growth of Software Testing," Communication of the ACM, Vol. 31, No. 6, 1988, pp. 687-695.

[4] Durant, J., Software Testing Practices Survey Report, TR5-93, May 1993.

[5] Beizer, B., Software System Techniques, 2nd ed., Van Nostrand Reinhold, New York, 1990.

[6] Burnstein, I., Suwannasart, T., Carlson, C.R., "Developing a Testing Maturity Model: Part I," Crosstalk, The Journal of Defense Software Engineering, August 1996, pp. 21-24.

[7] Burnstein, I., Suwannasart, T., Carlson, C.R., "Developing a Testing Maturity Model: Part II," Crosstalk, The Journal of Defense Software Engineering, September 1996, pp. 19-26.

[8] Burnstein, I., Homyen, A., Grom, R., Carlson, C.R., A Model to Assess Testing Process Maturity, Crosstalk, The Journal of Defense Software Engineering, November 1998, pp. 26-30.

[9] Basili, V.R., and Weiss,, D.M., A Methodology for Collecting Valid Software Engineering Data, IEEE Transactions on Software Engineering, Vol.SE-10, 1984, pp.728-738.

[10] Srichaivattana, P., "The Set of TMM Metrics," Technical Report, Software Engineering Lab, Department of Computer Engineering, Chulalongkorn University, 1999.