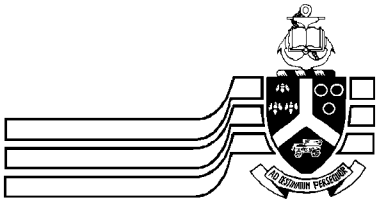
SOFTWARE TESTING in a small company: a case study

TECHNICAL REPORT

Johan van Zyl[1]

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA: **March 2010**



[1] This Technical Report is a slightly shortened version of Johan van Zyl's M.IT.-dissertation, Department of Computer Science, University of Pretoria, Republic of South Africa, 2nd Semester of 2009.

Table of Contents

Table of Figures

Table of Tables

Abstract

Small software companies comprise the majority of the software industry worldwide. In order for these small software companies to survive in a highly competitive marketplace, they must produce high quality software that will ensure a sustainable business model. However, this is not the case for the company being investigated in this research. In this report we critically analyse the software test process of a small software company in South Africa. This company is plagued by software that is fraught with defects, often causing customers to abandon the software completely after being purchased and severing its business ties with this company. We investigate the state of software testing in industry in general and attempt to provide company X with a practical and basic test process that will enable the creation of higher quality software. The concrete problems with the test process in company X are determined with the use of a questionnaire and are also confirmed by the software engineering literature as common software testing problems. We investigate two prevalent test process improvement models and use them as the basis for our proposed test process. The improved test process presents a structured, clearly defined process that can be implemented in company X over a two year period.

1 Introduction

Software has become so intrinsic in the fabric of modern society that its value cannot be ignored. We interact with software, mostly unknown to us, in almost all aspects of our daily lives. This has opened a window for many businesses wanting a slice of the software pie. This phenomenon has given birth to thousands of small software companies (SSC) across the world. Small software companies, with between one and fifty employees, comprise the majority of the software industry as opposed to their larger counterparts [4-11].

1.1 Problems facing small software companies

SSCs face many challenges in their pursuit to create quality software and survive in the market place.

- The first challenge is that SSCs do not believe that the same processes and methods used by large software companies are applicable to them. The reasons for not adopting these methods are cost, lack of resources, time, and complexity [9, 12].
- The second challenge facing SSCs is that their processes and organisational structures are informal and lead to a chaotic environment [5, 9-10, 13]. A possible reason for this is that SSCs have to focus on time-to-market to survive and often neglect the more resource intensive formal processes.
- The third challenge is a lack of resources, skills and experience [12, 14]. SSCs can often not afford to employ experienced software developers.
- The fourth challenge is that despite the many Software Process Improvement (SPI) programs such as CMM, CMMI and ISO/IEC15504 out there, SSCs are either not aware of them or the software engineering practices employed by SPIs are not focused on the needs and problems facing small companies [12, 15]. A study conducted by Broy and Rombach [16] on the German software industry indicated that only 5% of all German software houses have a CMM level of 2 or higher. The other 95% have a CMM level 1 maturity.

1.2 Problems in the company of this case study

The problems facing small software companies mentioned above are similar to the problems faced by company X, a small South African software company. The software process is defined, but not executed accordingly in each project. The reason for not adhering to the defined process during projects can be attributed to a lack of historical project data which leads to unachievable and unreasonable deadlines in current projects. The project becomes rushed as it falls more and more behind schedule, which in turn leads to shortcuts being taken in the process. The result of this rushed and ad-hoc process is software products that are not properly tested and more often than not, lack quality. The other major problem facing the company is the lack of resources in terms of skills and experience. Due to a lack of financial resources, all the software developers currently employed are university graduates with no industry experience. This leads to actual customer facing projects being a training ground for software developers that spend most of their time doing research on how to implement the required functionality of the product. The software process used in this company is the Waterfall model [17], where the software testing phase follows at the end of the process. Due to the rushed and chaotic nature of all software projects at

company X, this phase is mostly ignored and used for implementation of outstanding functionality. This leads to late, untested software products fraught with defects. This is not a sustainable model as frustrated and unsatisfied customers will not return for future products.

1.3 Description of Company X

Company X is based in South Africa, and was established in 2005. It initially started with one employee and has steadily grown to 43 employees in 2009. The company is made up of three divisions: consulting, support, and software development. The software development division employs eight software developers, all with less than three years industry experience. The type of software products developed by company X covers a wide spectrum that ranges from small desktop applications to large content management web applications, running on the Microsoft Windows and Linux platforms. The company focuses on current customer needs and will attempt any type of development project required by a customer, even if there are no skills in that particular domain. The reason for this wide array of products is to ensure a continuous flow of income.

The success or failure of these small software companies depend on the quality of their software products. Software testing forms part of quality assurance (QA), a process aimed at building quality into a product (during development) to ensure that the required level of quality is achieved. These two processes are integral parts of the software development process, but are often left until last due to a lack of resources and time-to-market pressures [18]. Without proper software testing, high quality software products cannot be created and the future of many small software companies lies in the balance. The cost implications for releasing software fraught with defects are immense, not to mention the costs incurred when fixing defects in the latter stages of a project [19].

1.4 Problem Statement

Based on the challenges facing small companies such as company X, the following problem statement can be derived as a list of questions that this research will need to answer. The main question to be answered by this empirical study is:

- How can we improve the current software testing process for a small company such as Company X?

In order to answer the main research question above, the research must answer the following sub-questions:

- What is the current software testing process of company X?
- What are the strengths and weaknesses of the current testing process?
- How can Company X build on the strengths and eliminate the weaknesses so as to improve their current software testing process?
- What does the software engineering literature recommend in order to improve the software testing process?

1.5 Goal of the project

The aim of this report is to critically assess the software testing process implemented at company X and propose an improved process, based on the software engineering literature, and insight

gained from experience, i.e. conversations with industry test practitioners.

1.6 Structure of the remainder of report

This report is structured as follows: Firstly the problem analysis is presented in chapter 2 that depicts the current software testing process at company X. This will be followed by chapter 3, which will present two test assessment and improvement models. These two models will be described in detail and a comparison between these models will be given. The current testing process at company X will be critically assessed with these models to determine its current maturity. Chapter 4 will present the findings from the software engineering literature to the problems identified in chapter 2. This will be followed in chapter 5 by the presentation of an improved work-flow process for testing at company X. Chapter 6 will critically assess the newly proposed testing process according to the test process maturity and improvement models presented in chapter 3. Chapter 7 will discuss the old process in conjunction with the proposed process and the steps involved in employing the new process. The report will conclude with chapter 8 on future work.

2 Problem Analysis

Before we are able suggest improvements to the current test process of company X, we need to analyse the current process. With suggestions from [20-21] regarding the research methods in information technology, we determined that a questionnaire and interviews would be most effective for achieving the research outcomes. We discuss the structure of the questionnaire, followed by the analysis of the results. The last section in this chapter presents the concrete problems that were identified in the current test process of company X.

2.1 The Questionnaire

In order to study the current testing practices of company X more closely, interviews were conducted with all the developers. The questionnaire used in the interviews contained fifteen open-ended and two close-ended questions. Each question was clearly explained to the developer to ensure that he understood what was asked of him. The interview lasted about 30 minutes. The last question asked the developer to model the testing process as he currently performs it. They were given one day to complete this question. The main objective of the interview was to determine the current state of testing practices and identify the concrete problems within the current testing process. More specifically, the questionnaire focused on eliciting information from the following three categories:

i. The level of software testing skills and knowledge of the developers.

ii. The company's commitment to software testing as perceived by the developers.

iii. The current software testing process used in the company.

2.2 Research Population

The population of this research included the eight developers currently employed by company X. The role of the author is also important as he is currently the software development manager. Five out of the eight developers are university graduates with qualifications ranging from three year diplomas to four year bachelor degrees in Information Technology and Computer Science related fields. Two developers are still in the process of completing their studies, while one developer never completed his tertiary education and has no further plans to do so. Six of the eight developers attended a Technical College while only two developers received their education at a University.

Company X was the first employer for all eight developers, including the author. These eight developers, including the author, began their working careers with an industry based learning programme. This programme provides students with industry experience while in the final semester of their tertiary studies. The benefit to an employer is bright young minds at very low salaries, while the advantage to the student is a possible permanent position after successful completion of the programme. The author completed his industry based learning at company X in December of 2005 and was permanently employed by the company. Since then he has worked as a software developer until October 2008 when he was appointed as software development manager.

2.3 Interview Results

The three categories of the questionnaire are discussed in this section.

2.3.1 The Level of Software Testing Skills of Developers

This section of the questionnaire aimed to determine the current skill level and knowledge of the developers regarding software testing. Only 50% of the developers indicated that they received formal training on testing at tertiary level. On further investigation into the extent of tertiary training received, this 50% indicated that it was introductory testing theory that was part of a Software Analysis and Design course. The topic of software testing constituted one chapter from their textbook.

Since completing of their tertiary education, all developers indicated that they have not attended any training courses or seminars on software testing. To further examine their testing knowledge, the author enquired whether they knew how to design and write unit tests. Not surprisingly, 75% indicated that they possessed no knowledge on this topic. The remaining 25% acquired this knowledge through self study.

On the question relating to their knowledge on the use of test tools to assist in the testing process, 100% suggested that they had no knowledge of any test tools. Two developers initially suggested that they used SoapUI to test their web services. Upon further investigation of this tool, it was found that one of its features provides automatic creation of test cases from web service methods. Developers did not use this functionality, but merely applied it as a debugging tool to view the http messages exchanged between the web service server and client.

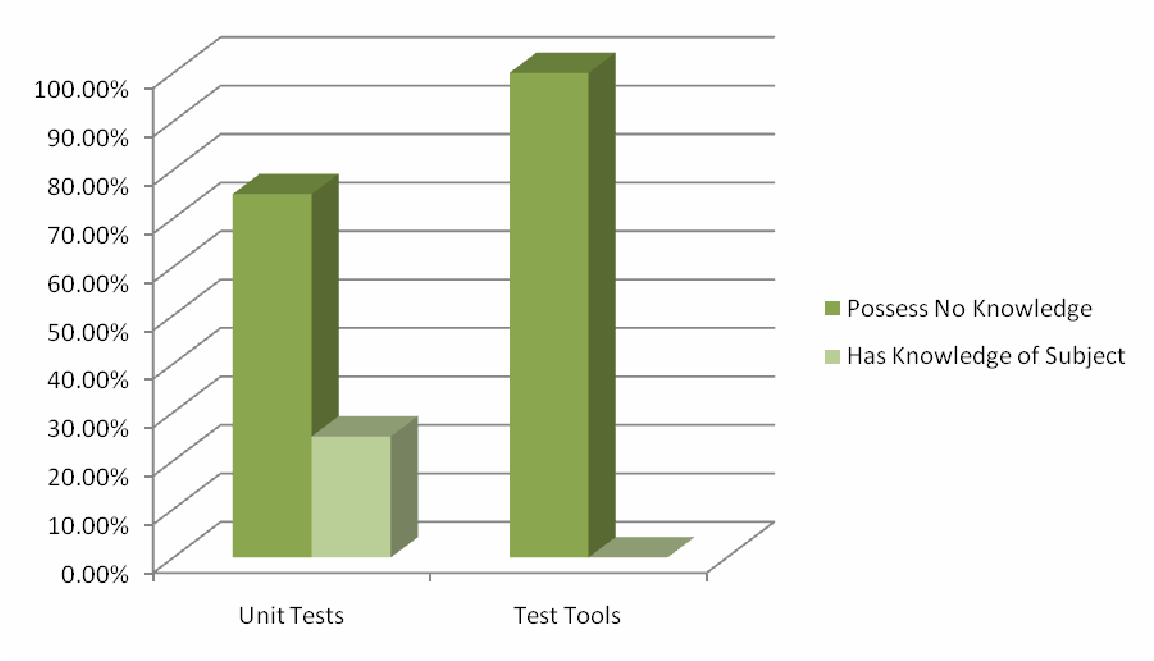


Figure 1: Developer Knowledge of Unit tests and Test tools

The current software process used in company X is based on the Waterfall [17] model that approximately comprises the following phases: requirements analysis, architecture, design, implementation, testing, release, and maintenance. The participants were asked to indicate where in this software process they performed testing activities. 87.5 % argued that they only do testing upon completion of a class function or unit of their code. The size of a unit represented a single method, or interdependent methods that represented one of the project's requirements. One participant indicated that he tested his code as soon as he encountered a defect.
To gain further insight into the participants' understanding of software testing as a complete process, the author asked whether software testing 'only entailed the execution of code'. All but two participants agreed that this was indeed the case. Only two participants argued that the software design could also be tested. One of these two participants also suggested that test cases could be written before coding.

2.3.2 Company X's Commitment to Testing as Perceived by Developers

The participants were asked four questions to gain a better understanding of how they perceived the investment and commitment of the company towards testing. The author chose not to get the views of the directors of the company so as to avoid possible biased answers. 87.5 % indicated that there were no clearly defined principles and objectives on software testing provided by the company. One participant argued that he has once heard a senior manager comment on the testing that must be performed on a specific project. The senior manager advised that a test environment be created where integration and system testing could be performed before moving the software to the production environment.
Participants were asked whether a test environment that closely represented the production environment was readily available on projects. 25% indicated that this has never happened. 62.5% suggested that there is rarely a test environment available. One participant argued that he mostly has access to a test environment. It was found that he has only been working on one project since joining the company eight months ago. Upon further investigation into the available test environments, participants commented that they make use of virtual machines that run on their notebooks. These virtual machines were configured by the developers themselves and are swapped around among them, depending on the operating system and installed software required.
On the availability of test documentation templates for use on projects, 75% indicated that there was no documentation available. 25% commented that a test plan template was provided on one project. This template was in fact created by the author after he started his research on software testing.
The author subsequently explained to the respondents about the different levels of testing that included unit testing, integration testing, system testing and user-acceptance testing. He elaborated on the tasks and objectives that form part of each level. Upon enquiring whether any of these testing levels were clearly defined on each project, 75% indicated that they have never heard of testing levels. 25% argued that integration and system testing was explained on one project they were involved with. Coincidentally, this was also done by the author as a result of this research.

2.3.3 The Current Testing Process

The final section of the questionnaire was aimed at discovering the testing process that is currently being followed by developers. The time typically spent on testing activities by 87.5% of developers constituted less than 10% of the total time of project. One developer indicated that he spent between 10% and 30% of his time on testing.

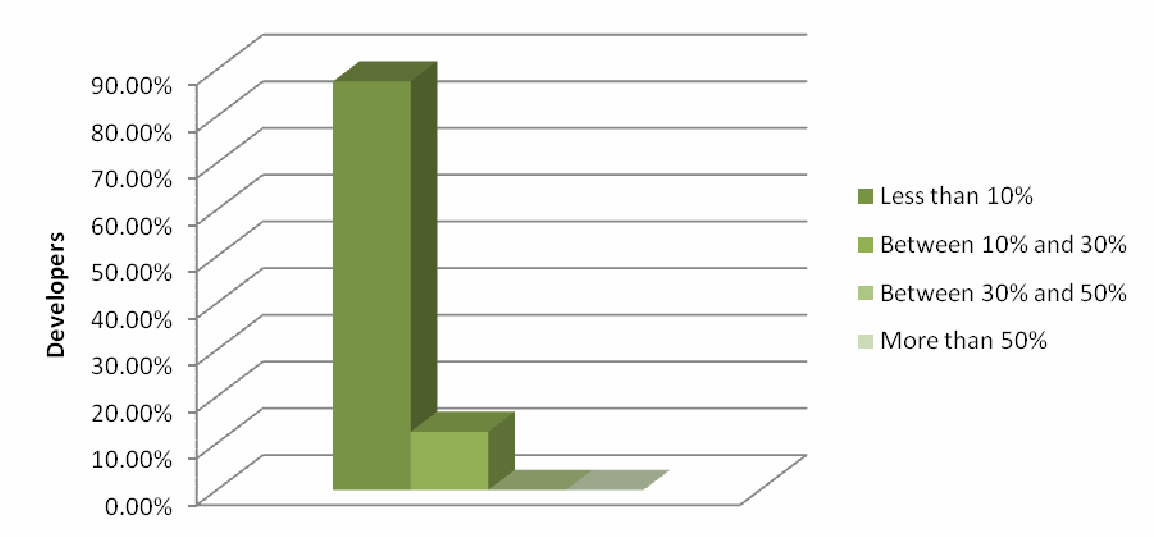


Figure 2: Percentage of Time Spent on Testing per Project

The respondents were asked whether they performed any test planning before they started testing. 87.5% commented that no form of planning was performed. 12.5% indicated that some degree of planning was performed. This included the creation of check-lists that contained the functional requirements of the product.

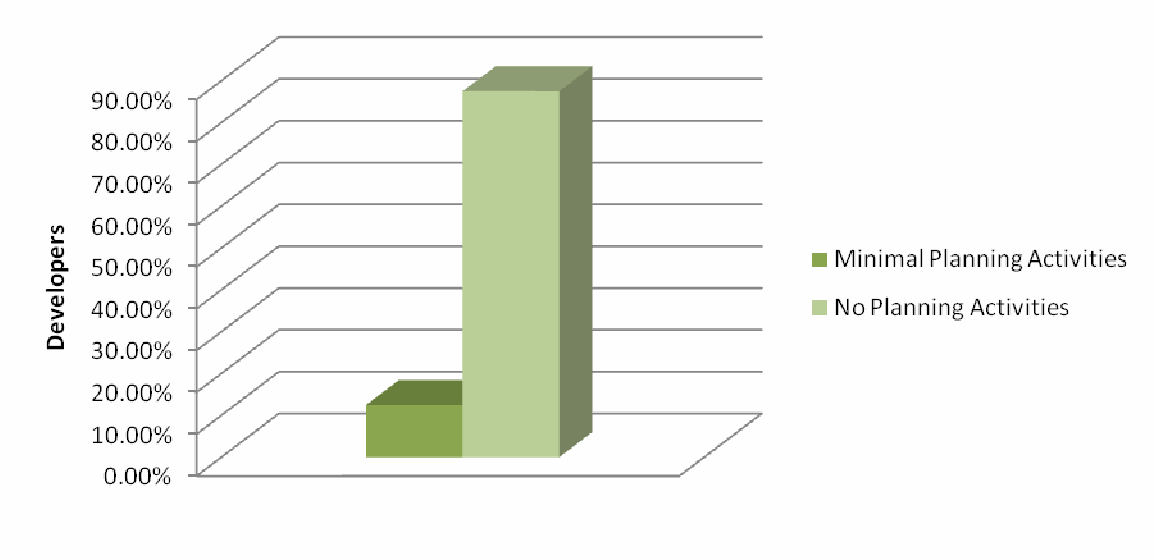


Figure 3: Test Planning Activities

On the question regarding the creation of documentation to help track progress and report test results, 62.5% indicated that they 'sometimes' created documentation. One participant argued that he always creates documentation, but that it is only informal documentation on a piece of paper for his own use. 25% indicated that they never create any form of documentation.
Developers were asked to give an indication on the number of test cases that they typically write on a project. 75% do not write any test cases. 25% indicated that they write between one and ten test cases per project. These two participants were also the only two that knew how to write unit tests as indicated in section above.

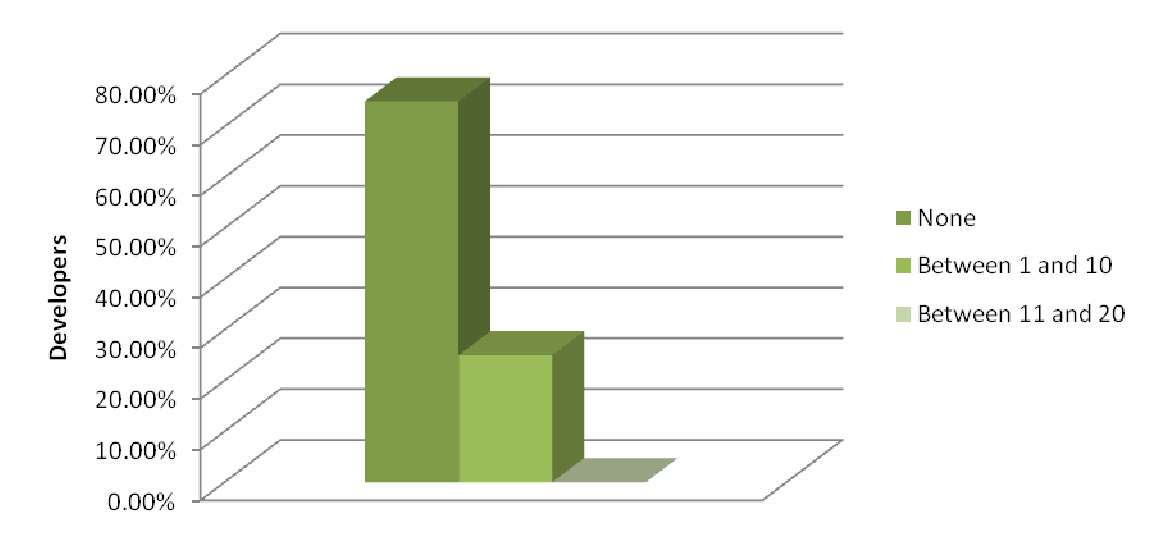


Figure 4: Number of Test Cases Typically Developed per Project

Participants were asked how they keep track of defects in their code. 75% indicated that they scribbled it on a piece of paper. One participant tries to remember all the defects and never writes anything down. Another participant fixed defects as soon as they were detected and did not see the need to keep track of them.

The participants were given an open-ended question where they could make suggestions on how they would improve the current testing process. This question also provided valuable insight into the current problems of the testing process. Two participants indicated that a test strategy was needed to provide guidance on projects. Five participants argued that dedicated testers must be employed by the company. Three participants indicated that more emphasis must be placed on the testing phase in the software process. One participant mentioned more active user involvement. 37.5% suggested the use of buddy testing, where a developer other than the one who wrote the code, performs testing on the code. 75% of the participants suggested that an internal training program on software testing was needed. Two participants mentioned that proper test documentation templates be provided on projects. Only one participant suggested the use of test tools training. It could be argued that the other participants wanted this included as part of the internal training program. One participant suggested the use of a defect tracking system. One participant suggested that better requirements analysis should be performed. This was an interesting answer as the participant made the connection between testing and the quality of requirements. The final open-ended question asked the participants to model the testing process as they currently performed it. All the test process models created by the developers were

generalized into a single process. The process is modelled using the Software Process Engineering Metamodel [22] notation and can be seen in Figure 5 below. The process starts off with the requirements specification document work product. This work product serves as input to the implementation of features. Once a feature has been implemented by a developer, the developer would compile (deploy, in the case of interpreted programming languages) and run the software to verify that it fulfils the requirement. Should a defect occur at this stage, the developer will immediately attempt to fix the defect? After fixing the defect, the developer will once again compile and run the code. If the code performs the expected functionality, the developer will implement the next feature. This process will iterate until all the features are completed. Upon completion of all the features, the software will be released to the customer.

2.4 Analysis of Results

After analysing the results of the questionnaire, the following concrete problems and shortcomings in the current testing process can be identified. These problems will be discussed in the same context as they were presented above, by categorizing them into three sections. Problems could also be identified, based on improvements suggested by the developers.

2.4.1 Testing Skills and Knowledge of Developers

1. The developers did not receive rigorous training at tertiary level. The little training that was received by some of the developers was only a theoretical overview of testing and was not explored in depth or practically implemented.
2. No further training courses or seminars have been attended after tertiary education was completed. This meant that none of the developers were properly trained on software testing.
3. No internal training programs are conducted by company X to further educate developers.
4. Unit tests that are usually performed by developers [23] in the testing process were also not performed by 75% of the developers. The reason for this is a lack of knowledge which can mainly be attributed to the first three problems of not receiving any training on the subject.
5. No test tools are used to assist in the testing process. This also implies that no test automation is performed.
6. Software testing as it is currently implemented is not integrated into the software process. 87.5% of developers only perform testing after they have implemented their code.
7. Developers do not know that testing entails more than just the execution of code. According to 75% of the developers, the only way of testing is to have runnable code available.

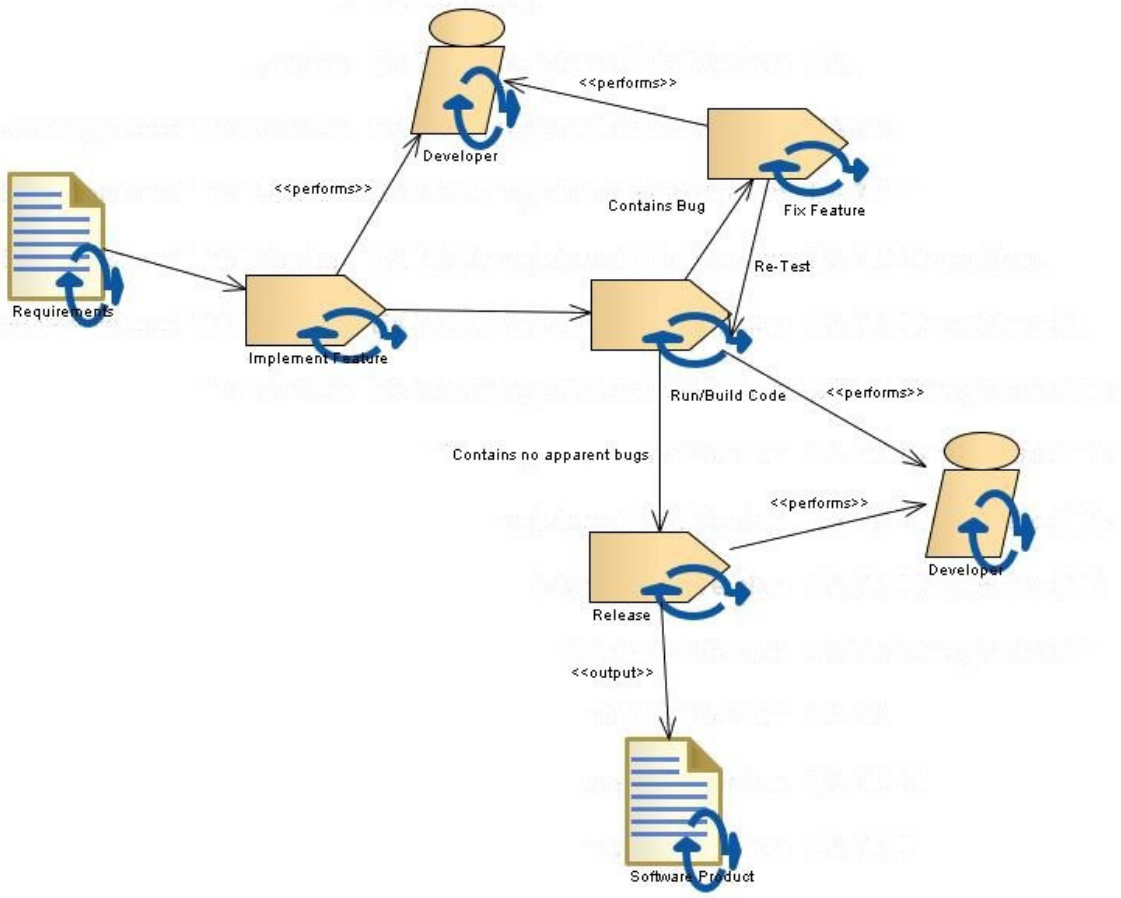


Figure 5: Current Test Process in Company X

2.4.2 Company's Commitment to Testing

8. The company does not provide developers with a company-wide testing policy. There are no testing objectives or principles that indicate the view or objectives of the company regarding testing.
9. No proper testing environments are provided for testing. Developers use virtual machines running on their notebooks or desktop computers to perform their testing. These virtual machines are not properly managed. They are mostly reused among developers and do not present a pristine testing environment.
10. No test templates or documentation from previous software projects are provided. Developers sometimes create their own test documentation that does not adhere to any standards.
11. None of the test levels that comprise the testing process are clearly defined. The tasks that should be performed within each level are also not defined by the company.

2.4.3 Current Testing Process

12. Less than 10% of total project time is spent on testing activities.

13. The testing process cannot be monitored as most developers keep informal documentation on their testing progress. No test results are available as these documents are not part of the deliverables of the project. There is thus no measurement of the current testing process.
14. There is a lack of planning on how the testing on a project will be performed. The testing is mostly ad-hoc and the decision on what needs to be tested lies solely with the developer.
15. Due to the lack of testing skills and knowledge, little or no test cases are written. This implies that most code written for testing purposes forms part of the production code. There is thus no separation between production code and test code.
16. Defects are not captured in a defect tracking system where they can be properly monitored, managed, and prioritized. Developers mostly write the defects they encounter on pieces of paper.

2.4.4 Problems Identified Based on Improvements Suggested by Developers

17. There is no dedicated test team or testers available to only perform testing. All testing activities are performed by the developers.
18. What to test is currently determined by the requirements specification document. The quality of testing can thus be directly attributed to the quality of the requirements.

In this chapter we analysed the results from the questionnaire and identified eighteen concrete problems in the current test process of company X. We also presented a model of the current test process in SPEM notation. The following chapter will discuss two prevalent test improvement models that can possibly be used to improve the test process in company X.

3 TMMi and CTP

The Test Maturity Model Integration (TMMi) and Critical Testing Processes (CTP) are process

improvement models that focus on software testing. In this chapter the structure and activities of these models will be discussed. This chapter will also compare the CTP to the TMMi and map the various processes of the CTP to the process areas of the TMMi. This mapping will demonstrate the TMMi maturity level that can be attained by implementing the processes of the CTP.

The ISTQB [24] defines testing as "the process, consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation, and evaluation of software products and related work products to determine that they satisfy specific requirements, to demonstrate that they are fit for purpose and to detect defects." Based on this definition, it is clear that the test process is not just a phase that follows coding, but is made up of specific core activities that form part of a structured approach. This view of testing has led to the development of test process improvement models that provide organisations with step-by-step guidelines and best practices of core activities that should form part of a test process. Various test process improvement models have been developed to assist companies in establishing a mature test process. These include the TMM [25], TMap [26], TPI [27], and TIM [28].

We have selected the TMMi by the TMMi Foundation [2] and the CTP by Black [1] as the two models to assess and improve the current test process in company X. The reason for selecting the TMMi model is that it is fast gaining recognition as an industry standard as was set out by its creators. It is relatively new and is also derived from the TMM and CMMi. The CTP follows a more agile approach to testing and was borne out of twenty years' experience in the software testing industry.

In order to understand how these test process models have evolved, a discussion of the history and evolution of testing will follow. This history and evolution of testing by Gelperin and Hetzel [3] served as the input to both the TMM and the TMMi.

3.1 Evolution of Testing

According to [3] the process of software testing has evolved over five phases in the period from 1950 until 1990. Each phase can be viewed as having a certain orientation and goal of testing in that specific time period. During each period, as the perspective of testing changed, a new test process model was born. The models include debugging, demonstration, destruction, evaluation, and prevention, as further described in the following:

- The debugging-oriented period was more focused on testing hardware than software. The reason for this according to [3] was that this period was the dawn of the digital computer and it was fraught with hardware reliability problems. Publications in this period on digital computer testing revolved around hardware components. There were software errors present, but it was more important to get the hardware working. The constructs of debugging and testing were not clearly differentiated as some viewed testing as part of debugging while others used the terms interchangeably. [3]
- The demonstration-oriented period was marked by a distinction in the meaning of debugging and testing. The aim of debugging was to ensure that "the program runs" while the aim of testing was to ensure that the "program solves the problem" says [3]. The hardware and software became more complicated, but it was also more widely used. This in turn led to more financial risks as it was costly to correct and maintain software. Hardware and software producing organisations knew that they had to produce reliable

software and they became aware of the importance of testing. The definitions of debugging and testing took on new meaning, but it still aimed to prove that the software ran and solved the problem says [3].

- The definition of testing as given by Myers [29] in 1979 distinctly characterized the destruction-oriented period. According to [3] emphasis was placed on finding faults. The reasoning behind this was that if a program was tested with the aim of finding faults, then the tests would have a higher probability of finding the faults. This in turn would lead to improved testing. [3]
- In 1983 a guideline on the verification, validation and testing was published by the Institute for Computer Sciences and Technology of the National Bureau of Standards. According to [3] this document earmarked the evaluation-oriented period by placing an emphasis on activities to be performed during the software life cycle. These activities included integrated review, analysis and test of the software product at each phase of the software life cycle. [3]
- The prevention-oriented period emphasizes the importance of early test planning and design. This requires the testing process to start alongside the software development process and not as a phase that follows at the end of the process. Testing is viewed as a form of risk management. It aims to reduce project risk and reduce the cost of software failure. According to [25] the prevention model is represented at level 5 of both the CMM and TMM.

Figure 6: The Evolution of Testing, taken from [3]

The following section discusses the Critical Testing Processes (CTP) developed by Black [1]. The approach to testing followed by the CTP is similar to that found in the evaluation and prevention phases. The CTP encourages the adoption of reviews during each phase of the software life cycle as stated in the evaluation phase. The CTP also places a strong emphasis on the management of risks during the testing process. This correlates to the prevention-oriented period of testing where testing was viewed as a form of risk management. Another factor that places the CTP alongside the prevention-oriented period is early integration. This allows the test

process to start when the software development life cycle starts and not only when it reaches completion.

3.2 CTP

The CTP was developed by [1] as a test process improvement model. The model contains 12 critical processes that can be applied in any software organisation. Black [1] argues that his processes "aren't pie-in-the-sky theory", but has grown out of his experiences in software testing over the last twenty years. Black [1] presents the processes as "lightweight check-lists" and not "bureaucratic regulations". He goes on to define the criteria that apply when a process becomes critical:

- The process is repeated frequently.
- The process has the ability to affect the cooperation of a team.
- Peers and management are involved in the process. This makes the process highly visible to others. For example, testers must effectively communicate their test results and findings to other testers, developers and management.
- The failure of the process could lead to severe and lasting negative outcomes.

The CTP is made up of four parts: plan, prepare, perform and perfect. When used in its entirety, the CTP provides a full test life cycle. The structure of the CTP is depicted in Figure 5.

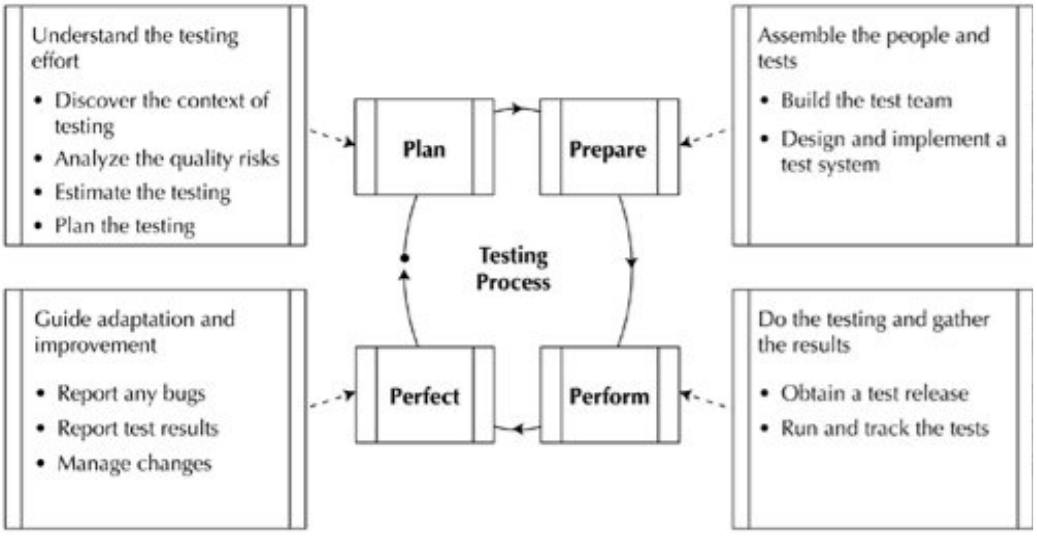


Figure 7: The Critical Testing Processes, taken from [1]

I. Plan

1) Part one is concerned with planning of testing effort. Process one of the CTP pertains to understanding the testing effort. This process presents a high-level overview of the complete testing process, comprising the subsequent eleven processes. Any of the twelve processes can be implemented, depending on the needs of the organisation.

2) The second process is concerned with understanding the operational and organisational context of testing. This discovery phase should attempt to determine why testing is performed as it currently is and how it has evolved to reach this state. The steps included in this phase are:

   - Understanding the current software development life cycle implemented in the organisation. Black [1] recommends integrating the testing process into the software development life cycle using the V-Model [18].
   - Studying all test-related and quality-related documentation to understand the function they served. This includes bug reports, test plans, and project documentation.
   - Have discussions with employees involved in the testing process, ranging from testers, to test managers. These discussions should revolve around the expectations that they have regarding testing.

3) Process three is concerned with the definition and prioritization of risks. Black [1] emphasizes the importance of a quality risk analysis to determine the tests that are most critical to system quality.

4) Process four relates to estimation of the test effort and obtaining support from management. Estimation involves the alignment of the testing effort with that of the software development project's schedule and budget. The schedule and budget must be developed in close cooperation with the test team and approved by management.

5) Process 5 is aimed at generating a plan for the test effort and once again obtaining management's support. The deliverable for this process is a test plan that communicates the testing strategy on the project. All stakeholders must be informed of their role in the test process and how the test process will be executed.

II. Prepare

The preparation phase commences upon completion of the planning phase. This phase includes the assembling of a test team and construction of a test system.

6) Process 6 entails the building of a test team. This can include the hiring of testers or training existing staff. Black [1] argues that “only a team composed of qualified test professionals can succeed at the critical testing processes”.

7) Process seven is concerned with the building of a test system that includes: creation of test suites to cover the risks identified, selection of the appropriate test techniques based on the test conditions, and creating and configuration of the test environment. Once the test team and test system is in place, phase 3 of the CTP starts.

III. Perform

The purpose of this phase of the CTP is to execute the test cases developed and to record the test results.

8) Process eight is concerned with the test release. This includes the selection of bug fixes, new features, and documentation for a particular test release. The source code must be prepared for

testing and the build must be assigned a version number. Once everything is in place, an intake test can be performed to ensure that the system is ready to be tested. The ISTQB [24] defines an intake test as "a test to decide if the component or system is ready for detailed and further testing."

9) Process nine is concerned with the management of test cases. This process includes the selection of relevant test suites, based on the risks identified. The test cases in each suite are executed and results are recorded in test summary worksheets. Defects must be reported in a defect tracking system. The final phase of the CTP is concerned with the perfection of the testing process.

IV. Perfect

The purpose of this phase is to adapt and improve the testing process. The foundation of this phase includes bug reporting and results reporting.

10) Process ten is the bug reporting process where the various aspects surrounding a bug must be clearly defined in a bug report. The completed bug report is then entered into the defect tracking system.

11) Process eleven communicates the test results to all relevant stakeholders.

12) Process twelve is concerned with the adjustment to changes in the project and refining the testing process. The actual purpose of this process is change management to ensure that all features and defects are detected.

3.3 TMMi

According to the TMMi Foundation [2] the Test Maturity Model Integration (TMMi) has been developed to serve as a guideline and point of reference for test process improvement. The sources that served as input to the development of the TMMi was the CMMI, TMM, and Gelperin and Hetzel's [3] growth of software testing discussed above. According to [2] the TMMi can be used as a complementary model to the CMMI.

The TMMi uses a staged approach for process improvement and consists of five maturity levels. An organisation must work through all the lower levels of maturity before it can attempt to reach the higher levels. Each maturity level consists of key process areas containing specific and generic goals. The specific and generic goals in each process area must be present to satisfy that process area. Each specific and generic goal is made up of specific and generic practices respectively. All the specific and generic goals of each process area must be reached before the maturity at that level can be attained. This case study will mainly focus on TMMi level 2 and level 3. The first reason for this is that a test process improvement process that aims to reach TMMi level 2 can take up to two years and is thus not a small endeavour [30]. The second reason is that at the time of writing, only TMMi levels 2 and 3 were defined by the TMMi Foundation. The definition of TMMi levels 4 and 5 are planned for release in late 2009 or early 2010. Therefore, this research will not focus on a higher level of maturity than TMMi level 3.

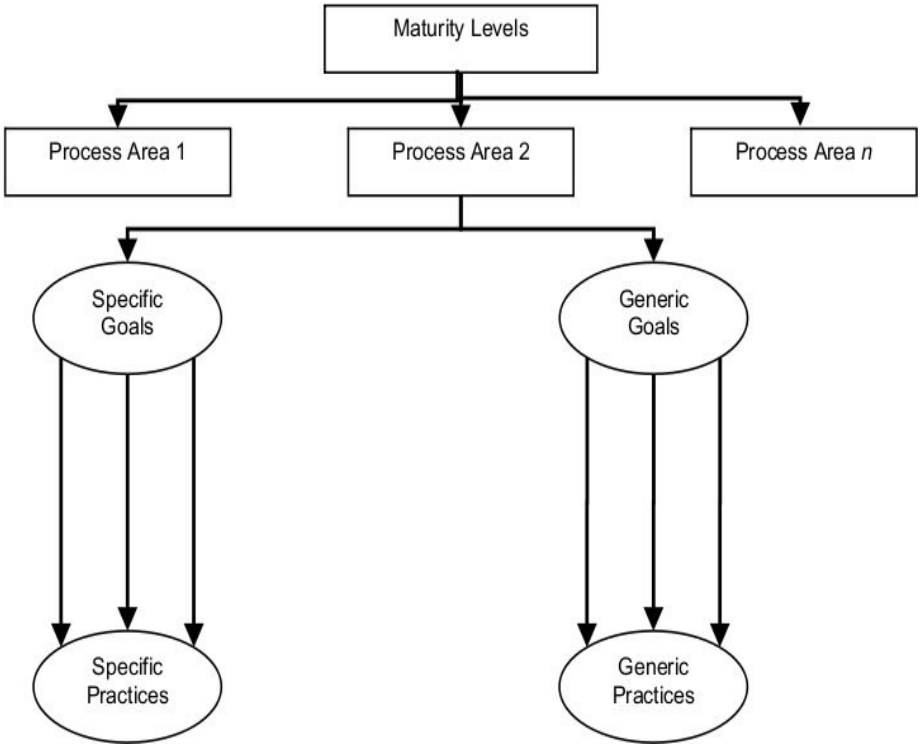


Figure 8: TMMi Structure and components, taken from[2].

Figure 8 depicts the structure of the TMMi. A maturity level indicates a company's current testing capability in specific process areas. Each maturity level is made up of several process areas that must be satisfied before that maturity level can be reached. Each process area is made up of specific and generic goals. The specific goals are unique characteristics that must be present in each processes area. Within the specific goals are specific practices. These practices describe activities that must be implemented to achieve the specific goal. The generic goals are found in more than one process area. A generic goal describes characteristics that must be present to implement the processes in a specific process area. The generic practices also appear in multiple process areas and describe the activities that are important in achieving a generic goal.

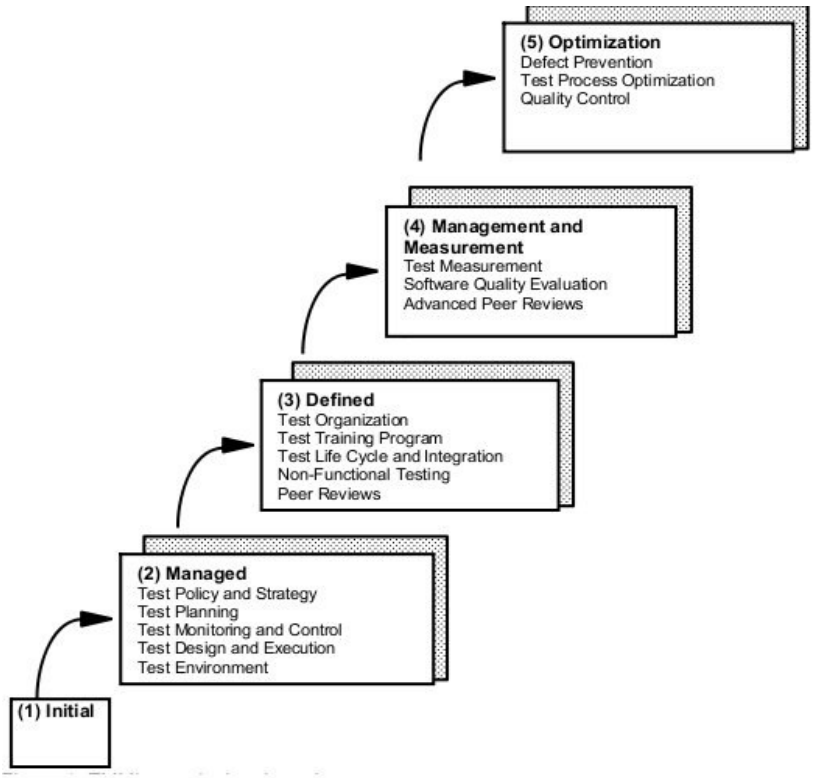


Figure 9: The TMMi Maturity Levels, taken from[2]

3.3.1 Level 1: Initial

According to the TMMi [2] the test process of an organisation at this level has the following characteristics:

- The test process is chaotic and undefined.
- There is no difference between testing and debugging.
- There is not a stable organisational environment that supports the test process.
- Most testing is performed in an ad-hoc manner after coding has been completed.
- The goal of testing at level 1 is to demonstrate that the software runs without any critical failures.
- The products developed by companies at this level are mostly too slow to work with, unstable, or do not fulfil their requirements. According to the TMMi [2] this is due to a lack of focus on quality and risks.
- Projects are usually late, over budget, and quality is not at the expected level. With regards to testing at this level, it is usually performed without tools, proper resources or well-educated staff.

By looking at these characteristics of a typical TMMi level 1 company, it is clear from the interviews conducted with the developers of company X that it currently finds itself at this level. The aim of this research is to assist company X to reach level 2 or 3 of the TMMi.

3.3.2 Level 2: Managed

Unlike level 1, there are key process areas at TMMi level 2. According to the TMMi [2], the following process areas form part of level 2: Test Policy and Strategy, Test Planning, Test Monitoring and Control, Test Design and Execution and Test Environment.

3.3.2.1 Test Policy and Strategy

The first step to test process improvement according to the TMMi [2] framework is the creation of a clearly defined test policy. The purpose of the test policy is to define at a high level the organisation's approach, objectives and principles regarding testing [2]. The test policy provides a common perspective on testing between all stakeholders in the organisation that proves essential to test process improvement [2].

A test strategy is defined based on the test policy of the organisation. The test strategy provides generic test requirements that can be applied to all projects in the organisation. It serves as the starting point for testing activities in all test projects.

3.3.2.2 Test Planning

The test planning process should start at the initiation of the test effort. The test planning includes implementation and identification of required resources and activities to meet the objectives set in the test strategy [2]. It is important to understand the objectives and goals of the customers and stakeholders. The risks of the project need to be clearly identified and prioritized. The test policy and test strategy serve as input to the test planning process [2].

3.3.2.3 Test Monitoring and Control

The activities in the test plan must be monitored continuously throughout the test process to ensure that they are proceeding according to the test plan. This forms the foundation of test monitoring and control. Test control measures the actual progress against the test plan and takes corrective action where needed.

3.3.2.4 Test Design and Execution

The purpose of the test design and execution process area of the TMMi [2] level 2 is to create test design specifications using test design techniques. A structured test execution process is performed and test incidents are managed to closure.

3.3.2.5 Test Environment

The test environment process area is the final process area at level 2 of the TMMi [2]. This process area is concerned with the establishment, management and control of a proper test environment, including test data, in which tests can be executed in a manageable and repeatable process. The models discussed in this chapter will also be used to develop a matrix of the comparisons between the CTP and TMMi. These models will be used to assess the current test process maturity of company X.

3.3.3 Level 3: Defined

As opposed to TMMI level 2 where testing is a phase that follows coding, testing at TMMi level

3 is integrated into the development life cycle. According to the TMMi [2] a major difference between TMMi level 2 and TMMi level 3 is the scope of standards, processes descriptions and procedures. Testing standards include the IEEE829 (The Test Documentation standard) [31], as well as BS7925-2 (British Standard for Component Testing) [32]. The test process areas at TMMi level 3 according to the TMMi [2] are discussed below: Test Organisation, Test Training Program, Test Life Cycle and Integration, Non-Functional Testing and Peer Reviews.

3.3.3.1 Test Organisation

According to the TMMi [2], the purpose of a 'Test Organisation' is to establish a group of trained and skilled individuals that are responsible for testing. A test organisation is characterized as having a commitment to improved testing and higher-quality software. There must be an independent test group. This group must have a clearly defined position in the organisational hierarchy and the test group must not be part of the development group. The test group must also have its own management and reporting structure that is independent from development. The individuals in the test group are seen as specialists that have a clearly defined career path and training program in the organisation.

3.3.3.2 Test Training Program

The 'Test Training Program' process area is concerned with meeting the strategic business objectives of the organisation and meeting the training needs of test group. According to the TMMi [2], the 'Test Training Program' is closely related to the 'Test Organisation' process area. One of the main objectives of the Test Training Program is to support the Test Organisation by training test specialists and stakeholders involved. The TMMi [2] argues that the following diverse set of skills is needed for testing at TMMi level 3: test principles, test techniques, test management, test tools, domain knowledge, IT knowledge, system engineering, software development, and interpersonal skills. Once a test training program has been implemented according to the needs of the people involved in testing, the effectiveness of the training program must be evaluated.

3.3.3.3 Test Life Cycle and Integration

According to the TMMi [2] a standard test process must be defined, documented and maintained. The standard test life cycle forms part of the test assets that ensure reproducible test process performance across projects. The test process asset library contains a collection of items such as the description of test processes, descriptions of test life cycle models, and supporting test tools. These assets are used to support organisational learning by providing a shared knowledge base of best practices and lessons learnt in the past. The TMMi [2] suggests that the test life cycle model “define the phases, activities, and deliverables for the various test levels.” The test life cycle must also be aligned with the development life cycle and integrated already from the requirements phase of the project. The early involvement of testing includes planning at the requirements specification phase, and unit test planning during the design phase.

3.3.3.4 Non-Functional Testing

Non-functional testing is concerned with testing the non-functional requirements of a software product. According to the TMMi [2] non-functional requirements are very important for

customer satisfaction. The ISTQB [24] defines a non-functional requirement as “a requirement that does not relate to functionality, but to attributes such as reliability, efficiency, usability, maintainability and portability. ” This process area thus places emphasis on developing a capability in the test organisation for non-functional testing. The non-functional test techniques address software quality characteristics such as performance, usability and reliability. The TMMi recommends the following non-functional test techniques: operational profiles for reliability, and load, stress, and volume testing for efficiency. The first step to non-functional testing is to perform a non-functional product risk assessment. This will assist in determining what needs to be tested, to which degree it must be tested, and how it must be tested. Various test techniques are used to derive test conditions and subsequently test cases are developed. These test cases are translated into manual or automated test scripts that are run during the execution phase of testing.

3.3.3.5 Peer Reviews

The TMMi [2] suggests the implementation of peer reviews as the final process area at TMMi level 3. Peer reviews are used to detect problems in work products as well as to identify changes that are needed in work products. The TMMi [2] recommends that between two and seven engineers take part in a review. The type of work products that can be reviewed include: requirements specification, design, source code, test design, and a user manual. The reviews can range from informal reviews, to more formal walkthroughs and inspections.

3.4 Discussion and Comparison of TMMi and CTP

The most notable difference between the TMMi and the CTP is that the CTP does not have levels of maturity defined. Black [1, 33] suggests that any one of the 12 critical processes can be implemented in no specific order. He further recommends that an organisation should implement the processes that make most sense to their needs. The TMMi follows a staged approach where each lower level must be implemented before a higher level of maturity can be attempted. Thus implementing a certain level necessarily implies that the lower levels have also been implemented. The CTP uses check-lists with steps defined in each process, whereas the TMMi has specific and generic goals, each made up of specific and generic practices. The TMMi has an assessment method whereby organisations can have their test processes assessed by the TMMi Foundation. This assessment can lead to a formal accreditation with the TMMi Foundation. There is no assessment method for the CTP. The CTP was developed by a single test practitioner whereas the TMMi receives input from a whole body of knowledge in the form of the TMMi Foundation. Both the TMMi and the CTP present clear guidelines on the procedures and activities that should be implemented in the test process. The TMMi presents a much more comprehensive and in-depth model of the test process through its use of process areas. The TMMi also aims to be the industry standard for test process improvement and assessment. By comparing the TMMi and CTP with each other, we have developed a matrix that depicts the TMMi maturity of a test process that would have been implemented by CTP. The CTP would not reach TMMi level 2 as it does not implement all the process areas at this maturity level. Although some of the processes of the CTP implement some of the TMMi level 3 process areas, the TMMi level 3 maturity cannot be attained if level 2 is not reached. The mapping of the CTP against the TMMi is depicted in Tables 1 and 2 below. Table 1 maps the CTP against the process areas of TMMi level 2, whereas Table 2 maps the CTP against the process areas of TMMi level

3. Each process area of the TMMi in level 2 and level 3 was mapped against a single process of the CTP. The TMMi is much more structured and comprehensive in terms of its practices and goals for each process area. An X was placed in a cell where there was at least one step prescribed in the CTP process that correlated with a goal or practice of the TMMi for a specific process area. A cell where a '--' is present indicates that no steps in the specific CTP process correlates with any of the goals or practices of the TMMi process area.

TMMi / **CTP**	Test Policy and Strategy	Test Planning	Test Monitoring and Control	Test Design and Execution	Test Environment
Understand Testing Effort	--	--	--	--	--
Discover Context of Testing	--	--	--	--	--
Analyse Quality Risks	--	--	X	--	--
Estimate Testing	--	X	--	--	--
Plan Testing	X (Only Test Strategy)	X	--	--	--
Build Test Team	--	X	--	--	--
Design and Implement Test System	--	--	--	X	X
Test Release	--	--	--	--	--
Run and Track Tests	--	--	X	X	--
Manage Bugs	--	--	X	--	--
Report Results	--	--	--	--	--
Manage Changes	--	--	--	--	--

Table 1: Mapping of CTP to TMMi Level 2

The following table indicates the mapping of the CTP against the TMMi level 3.

TMMi / **CTP**	Test Organisation	Test Training Program	Test Life Cycle and Integration	Non-Functional Testing	Peer Reviews
Understand Testing Effort	--	--	--	--	--
Discover Context of Testing	--	--	X	--	--
Analyse Quality Risks	--	--	--	--	--
Estimate Testing	--	--	--	--	--
Plan Testing	--	--	--	--	X
Build Test Team	X	X	--	--	--
Design and Implement Test System	--	--	--	--	--
Test Release	--	--	--	--	--
Run and Track Tests	--	--	--	X	--
Manage Bugs	--	--	--	--	--
Report Results	--	--	--	--	--
Manage Changes	--	--	--	--	--

Table 2: Mapping of CTP to TMMi Level 3

3.5 Discussion

This chapter discussed two prevalent test process improvement models that shall be used as a basis for improving the test process of company X. The CTP and TMMi provide best practices and guidelines for test process improvement. Upon comparing the two models, it is clear that the TMMi provides a much more comprehensive test process than the CTP. Although the TMMi proposes more specific test goals and practices, these goals are clearly defined and easy to follow. By assessing the current test process of company X, it was determined that company X is still at TMMi level 1. The lack of test process structure, planning and commitment to testing in company X clearly represents the description of a company at level 1 of the TMMi. Except for the Test Environment process area, company X does not implement any of the process areas of TMMi level 2. Test environments are only available on some projects of company X and these environments do not always fully represent the production environment. The following chapter will discuss the findings of the software engineering literature to provide solutions for the problems identified in chapter 2.

4 Literature

The current test process and test-related activities in company X have been assessed and analysed in chapter 2 through a questionnaire and semi-structured interviews. This provided valuable insights into concrete problems that are experienced in the company. In this chapter we will look at the software engineering literature to match the concrete problems identified in chapter 2 and attempt to make typical suggestions on how to improve the current situation. Each problem will be discussed along with the related literature.

4.1 Lack of in-depth tertiary testing education

All the developers at company X, except for one, completed their tertiary education in Computer Science (CS) related fields. The developers commented that little or no software testing principles were taught. According to [33] and [34] only a small portion of undergraduate CS curricula is allocated to testing. They recommend that more testing should be taught at tertiary level. [35] agrees that university training on software testing is not good enough. A study conducted by [36] indicated that 27 out of 65 participating organisations reported that less than 20% of their test team received formal university training in software testing. [36] concluded that there is either a lack of practical skills delivered to tertiary students in software testing courses or a lack of short software testing courses at university. [33] recommends that more testing be taught and that there be more communication between industry practitioners and academics to ensure that undergraduate curricula provide students with a solid foundation in software testing. A study by Broy and Rombach [16] indicated that only 22% of all German software houses cooperate with research institutions to improve their software process.
These papers are all hinting at long term solutions to a widely recognized problem. In the immediate situation in company X, however they are no directly applicable. Nevertheless, we recommend that company X would send staff to extra courses.

4.2 Company X does not provide opportunity for test training or certifications

A study conducted by [36] indicated that 47 out of 65 organisations provided opportunities for their testing staff to be trained in software testing. Among the organisations investigated in this study, 37 out of 65 organisations preferred commercial training courses as opposed to 25 organisations that preferred internal training courses. Another study [37] conducted in Canada indicated that 31% of the respondents received rigorous training on software testing. [38] advises that small to medium enterprises (SME) should invest in test training courses for developers or testers. The International Software Testing Qualification Board (ISTQB) provides professional software testing certifications in almost 40 countries across the world, including South Africa. The ISTQB syllabi are made up of three levels: Foundation, Advanced and Expert. The Foundation certification syllabus provides a solid introduction to software testing and is well suited for people entering the field of software testing. The Advanced certification syllabus provides a mid-level certification to practitioners with at least 5 years experience. [39] suggests that prospective testers do not even have to attend expensive training, but can gain free access to the ISTQB syllabi from the ISTQB website and only write the exams at accredited testing centres.
The South African Software Testing Qualification Board (SASTQB) was established in 2008 as

the 40th National Board member of the ISTQB. There are at least three accredited ISTQB training providers in South Africa and we recommend that all the developers from company X attain the Foundation and Advanced level certification within the next two years. In addition to obtaining test-related certifications, we also recommend that developers get certified with IEEE as professional software engineers. Various levels of certification are available, ranging from the Certified Software Development Associate (CSDA) to the more experienced Certified Software Development Professional (CSDP). These certifications will verify that developers understand fundamental software engineering principles [40].

4.3 No unit testing performed by developers

The IEEE 610 [41] defines unit testing as "testing of individual hardware or software units or groups of related units". In Object Oriented programming, a unit may represent a single method in a class or the whole class.
None of the developers in company X write unit tests for their code. According to [23, 42-43] developers typically write unit tests as opposed to testers. They also reiterate the importance of unit testing and state that it is the most efficient test to conduct due to its low granularity and high code coverage. A study [42] conducted of the developers' testing process between 6 software organisations revealed that even having a defined test process still did not provide developers with a "concrete method for conducting unit and unit integration testing."
According to [44] unit testing is seen by many as an essential phase in software testing. [43-44] further state that unit testing allows individual units to be inspected that can reveal faults early on that would otherwise be difficult to find in system testing. [44] also argues that unit testing is not performed well and sometimes not performed at all due to cost implications. [38] recommends the following techniques to improve unit testing by developers:

- Force developers to perform unit testing be emphasizing that unit testing is their responsibility.
- Motivate developers by having regular meetings and seminars to share information.
- Give developers the freedom to perform unit testing as they feel convenient. Allow them to experiment with different test tools and technologies available.
- Provide developers the necessary resources. This includes time specifically allocated to research of testing methods and actual testing.

The IEEE 1008 [45] provides a standard for an integrated approach to structured and documented software unit testing.
We recommend that developers at company X be responsible for all unit test related activities. This practice will be emphasized in company X's test policy discussed in § 7 below. The training that the developers will undergo as well as the information provided by IEEE 1008 will provide developers with the necessary knowledge to perform unit testing.

4.4 All testing currently performed is manual

Company X does not use any test tools to assist in the automation of test tasks. In a study by [36] it was found that 44 out of 65 organisations used test automation tools. Interestingly, 30 of those

44 organisation thought that the use of automated test tools proved beneficial. According to [36] the biggest factors against the use of test tools according to their study were cost, time and difficulty of use. This view is also supported by [46] who states that test tools can be very expensive and involve a lot of effort from the testers. He suggests that organisations may consider using test tools when they have mature test practices or have a large number of tests. Organisations may do very well in software testing without the use of test tools [46]. [47] suggests that using test tools could pose a problem for organisations having the following characteristics:

- Lack of testing process.
- Ad-hoc testing.
- Organisational culture that does not support and embrace testing.

These three issues are very evident in company X at the current moment as there is no structured testing process. Most testing activities are ad-hoc, and the company has not yet embraced a culture of software testing.

Although testing tools and automated testing lightens the workload by eliminating repetitive manual tasks, the difficulties identified by the literature suggest that manual testing will be sufficient for company X for the time being. We thus recommend that all testing at company X be manual until the test process in company X is well established. Only once the test process becomes very labour intensive and time consuming will a test automation strategy be considered.

4.5 No review of project documentation or code

The only form of testing currently being conducted in company X is dynamic testing. The ISTQB defines dynamic testing as "testing that involves the execution of the software of a component or system". This implies that code must already be available before testing can commence. Static testing on the other hand does not include the execution of code, but refers to activities such as code reviews. The ISTQB [24] defines static testing as the "testing of a component or system at specification or implementation level without execution of that software, e.g. reviews or static code analysis". According to a study [48] conducted on the software test maturity of the Korean Defence Industry indicated that static test techniques were identified as having a low maturity on the Test Process Improvement [27] test maturity matrix. A study conducted by [42] of the test process in six software organisations revealed that only two organisations conducted code reviews against company coding standards. Only two organisations conducted peer reviews and only three organisations conducted inspections. Although the sample size of this study was small, it indicates that static testing techniques might be a lacking in many software organisations.

[42] recommends that organisations encourage the implementation of static testing alongside dynamic testing. [43] lists reviews of functional requirements and design as one two the key factors that indicate an organisation's test maturity. He advises that functional requirements reviews improve the correctness of requirements before the software is designed as well as reducing the risk of rework once flawed requirements are implemented. In terms of design reviews [43] argues that conducting reviews of the design can help eliminate flaws in the design before coding commences. In a recent study conducted by [49] between 10 software organisations to determine which factors impede the improvement of testing processes, the establishment of review techniques ranging from informal peer reviews to formal code inspections was suggested as a solution to improve the test process. [50] argues that reviewed

documents can result in a 65% deduction in defects that could otherwise have been carried forward to later stages of the development process.
In light of the problems identified with static testing techniques and the benefits that it provides, we recommend that company X implement three reviews throughout its software process. The first review will be conducted during the requirements phase to ensure that all specified requirements are correct and testable. The second review will be conducted after the design phase to ensure that the design does not contain flaws that will propagate to the coding phase. The third review will be a code review held during the coding phase to ensure that developers are coding according to company coding standards and to identify possible weaknesses or bad coding practices.

4.6 Testing only commences once coding is complete

Company X currently employs the sequential Waterfall model [17] for software development where the testing phase is near the end of the process. This is in stark contrast to newer software engineering literature findings. A study conducted by [51] on testing in emerging software organisations, much like company X, revealed that testing was being performed too late in projects. [52] argues that testing has a life cycle of its own and must run concurrently with the software development process. They too suggest that testing must start already during the requirements phase of a project. This view is also emphasized by [53] and [26] that between 60% and 80% of the testing effort should be completed before code is executed. This clearly implies that testing cannot occur only at the end of the software development life cycle. [50] suggest that testers be involved in reviewing requirements documents as well as design documentation. According to [50, 54] defects found early in the software life cycle cost much less to resolve than in the later stages of the process.
As suggested in § 5, we recommend that various reviews be conducted before coding commences. We recommend that testing be integrated into the software process of company X from the requirements phase and follow right through to the end of the software process. This suggested solution will also be emphasized in the test policy of company X (discussed in § 7) so as to make all developers and other project members aware of this practice.

4.7 There is no guidance for developers or testers on what or how to test

Many of the developers in company X indicated in the semi-structured interviews that they do not know how to perform testing, or what is expected of them in terms of testing. The developers also indicated that the company does not provide them with any guidelines of what is expected in projects in terms of software testing. According to [30] a test policy answers questions such as "Why do we test?", "How important is quality?", and "Are we time-driven or quality-driven?". [30] suggests that the test policy forms a critical basis for commitment to all test-related activities in an organisation. [55] suggests that an explicitly defined test policy provides a "framework for planning, organisation and execution of all test activities within a company". He warns that even an implicitly defined test policy or complete lack of a test policy can have the following adverse effects:

- Testing is not consistent over different departments. This will cause the testing approach to be defined individually on projects and the results will not be comparable between projects.
- The test policy will not be recognized and adhered to by the people involved.
- Managers will struggle to properly track the project's test status and performance as results will differ based on the testing approach followed.

The test policy will become a fundamental part of the test process and software process of company X. We suggest that the test policy be one of the first improvements to be implemented in company X as it provides the foundation for all test-related activities and procedures. The test policy will also emphasize the importance of testing and software quality as viewed by company X. This will create an awareness and sense of responsibility among employees, especially developers, that testing is viewed as a critical part of the software process.

4.8 No properly managed and maintained test environments

The test environment available at company X is currently not adequate to support the different software projects. Developers mostly use virtual machines running on their notebook computers to perform testing. Due to the high costs of hardware and software, small organisations do not have the resources to manage and maintain dedicated test environments [56]. In a study [42] conducted between six organisations on their implemented test process, one of the difficulties listed was that of creating test environments that closely represent production environments. [43] developed a set of twenty questions to determine the testing maturity of an organisation. The first question in his questionnaire enquires whether an organisation consistently creates an environment for development and testing that is separate from the production environment. He argues that the benefit of such a separate testing environment allows organisations to avoid making changes to production environments on the fly, as well as to develop and test software completely before it is deployed in a production environment.
Although a test environment is an important part of software testing as identified by the literature, it is not always possible to implement a pristine test environment on every project. The main reason for this is cost. In light of the current financial situation and recession in South Africa, an investment in a test environment is not possible at company X. The recommendation we make is that hardware currently available at company X is properly utilized. This will entail the enforcement of strict guidelines and management of the test environment. The test environment will be closely monitored to ensure that it is not abused during the test process. Due to the limited hardware available, virtual machines will be used to provide the operating system platforms and software. We recommend that the virtual machines hosting the software also be properly managed and that a library of software and operating system configurations be established. This library will provide quick access, configuration and deployment of the test environments.

4.9 No proper test documentation or templates available

Currently, little or no test documentation is created in company X. The few test plans that have been created, were not done according to an industry standard such as IEEE829 [31]. A study

[14] conducted in Finland regarding software process improvement in small companies also indicated that there were problems with software testing, especially with the creation of test-related documentation. The study did not elaborate on the reasons for this problem. [57] assessed the software process of a small company and found that no written test plans existed. According to them, this lack of documentation caused difficulty in reproducing the defects in order to fix them. [57] also noted that once these defects were fixed, the problem and solutions were never recorded at all. This caused the development team to investigate similar problems from scratch without having access to past documentation for guidance. The IEEE 829 provides a standard for software test documentation that can be used as templates in test projects.
We recommend that company X create a set of test documentation templates that will be available on each project. These templates will be created from the IEEE 829 standard [31]. This will include templates for a Master Test Plan (MTP), Level Test Case (LTP), Level Test Log (LTL), Anomaly Report (AR) and a Master Test Report (MTR). The templates will be stored in the Document Management System of company X where all employees have access to them. The use of proper test documentation will also be emphasized in the test policy of company X.

4.10 No structured testing approach defined

The test process on projects at company X is mostly ad-hoc and is left to the developers to decide how and what to test. A test strategy is a "high-level description of the test levels to be performed and the testing within those levels for an organisation" [24]. According to [58] a test strategy thus provides an organisation with responsibilities within each test level and also advises the organisation on how to choose test methods and coverage criteria in each phase. There is no such test strategy in company X to guide developers on how to perform testing. Interestingly, from the study conducted by [58] only 4 out of the 12 studied organisations used explicitly defined test strategies. Three organisations had implicitly defined strategies that formed part of some standard regulating the testing aspects of a product. Another test process maturity study [42] conducted in 2007 between six software organisations indicated that there was a definite lack of testing guidelines. This study argued that developers did not know what to test or how to test and provides an ideal opportunity for implementation of a test strategy. According to [30] the test strategy is based on the organisation-wide test policy. The strategy must be aligned with the organisation's software development process as various test activities and procedures are defined for each phase of the software development process. These activities typically include user acceptance testing planning at the requirements phase, system testing planning at the architecture phase, integration testing at the design phase, and unit test design and execution during the implementation phase.
We recommend that a test strategy be developed for company X. This strategy will form the foundation for test-related activities on each software project. It will provide developers, testers and management with practical guidelines on how testing will be implemented on software projects. The test strategy will be based on the test policy of company X as discussed in § 7. It is important that the test strategy be one of the first recommendations implemented as it forms the basis from which all test activities and procedures will developed.

4.11 Not enough time on a software project is allocated to testing

The developers at company X indicated that they spend less than 10% of the project on software testing. Testing easily can exceed half of the project's total effort [23, 59]. [51] found that testing activities were allocated too little time and resources in a project. [42] found that developers do not pay enough attention to testing, mainly due to a lack of time.
Based on these findings, we recommend that time be explicitly allocated to testing during a project. This must be reflected in the project documentation such as the project plan so that all stakeholders are aware of the time needed for testing. The 'lack of time' indicated by the developers can also be addressed by prioritizing testing. More complex code is likely to contain more defects and must receive priority above other code. Even though more time might not be available, at least the most important code will be properly tested. See § 18 for a further discussion of this problem.

4.12 No test-related activities are measured to track progress

No test-related data is stored in company X. This makes the measurement of test progress difficult or near impossible. According to [52] "metrics are measurements, collections of data about project activities, resources and deliverables." [52] can be used to assist in the estimation of projects, measure project progress and performance, as well as quantifying project attributes. [52] recommends that the following test metrics be tracked in every project:

- Number of test cases
- Number of test cases executed
- Number of test cases passed
- Number of test cases failed
- Number of test cases under investigation
- Number of test cases blocked
- Number of test cases re-executed

A study conducted by [36] indicated that only 38 out of 65 of the organizations investigated used test metrics. 21 out of the 38 organizations agreed that metrics improved the quality of the software developed. [60] also advises that metrics are an important part of the test process as it provides valuable insight into the progress of test projects. Company X does not collect any form of metrics and therefore cannot make estimations on future software projects. [60] also argues that metrics provide a basis for test estimation and helps to identify process improvement opportunities. [30] warns against selecting too many metrics when embarking on a test process improvement initiative as this could demand too much effort to monitor.
We recommend that company X implement a database to keep track of all project history. Data from projects can be collected from the test documentation discussed in § 9. Many of the metrics mentioned above can automatically be tracked by most defect tracking systems available today. This will be discussed further in § 17. The number of test metrics can be increased as the test process of company X matures.

4.13 No proper planning of the test process

The approach to testing in company X is not planned at all. Developers mostly start testing

without any plan of how, when or what needs to be tested. [51] performed a study that analysed the current testing practices of a small software company. The results of the study indicated that no upfront planning of the testing process was conducted. This is the same situation as in company X where no test planning is performed. According to [51] this resulted in deadlines being overrun and increases in project costs. [25] suggest that test planning is an essential component of a test process as it ensures that the process is repeatable, defined, and managed. [3] also support this view and state that test planning contributes significantly to improve the test process.
We suggest that each project in company X incorporates a test planning phase. The planning outcome of the phase will be a Master Test Plan (MTP) as discussed in § 9. The MTP provides an overall view of the test planning and test management of the project. The planning phase will be discussed in more detail in chapter 5 where the improved test process will be discussed. The following test planning activities will be implemented in company X according to the IEE 829 [31]:

- Selecting the different parts of the project's test effort.
- Setting objectives for each constituent part of the test process identified.
- Identifying the risks, assumptions and standards for each part.
- Identify the different levels of test and their associated tasks, as well as the documentation requirements for each level.

See § 18 for a further discussion of this problem.

4.14 No test cases are designed, created or executed

The IEEE610 [41] defines a test case as a "set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement".
The developers in company X do not design or execute test cases. This is mainly due to a lack of testing knowledge and guidance from a test policy and test strategy. In a study conducted by [58] in 2006 between 12 software organisations, 9 out of the 12 organisations did not enforce the use of test case selection methods. It was left to developers and testers to decide which test cases were selected. [49] argued that the absence of clearly defined test cases in a test process impedes the improvement of the test process. They recommend that templates be available that provides guidelines on the most common steps to create a test case. A study conducted by [42] to investigate the test processes followed by developers between six organisations revealed that only 50% of the organisations created test cases for their code. None of the organisations in this study developed any test cases before coding. Test Driven Development (TDD) is a software practice where unit tests are written before code is implemented. The code for the unit test is gradually added until the test passes.
[61] found that code developed using TDD had superior code quality compared to code developed using traditional approaches like the waterfall-like model used in company X. This same study revealed that 78% of the professional programmers interviewed believed that TDD improved their productivity.
In light of the findings above we recommend that company X implement a TDD approach. This will involve additional training needed by developers, but can be conducted in-house without incurring any costs. The TDD approach will become an integral part of both the software process

and test process and will be emphasized in the test policy. The recommendation for company X will force developers to write test cases throughout the implementation phase of the software process.

4.15 There is no system to keep track of defects

There is currently no defect tracking system implemented at company X to help keep track of defects detected or solved. Once a developer detects a defect, he will either try to remember the defect or scribble it on a piece of paper. [43] suggests that a defect tracking system allows an organisation to assign ownership to defects and track the defects until they are resolved. According to [43] can also indicate the status of the test effort by reporting on the number of defects found, number of defects resolved, as well as the outstanding defects. These metrics correspond with those suggested by [51] as discussed in problem 12 above.
We recommend that company X implement a defect tracking system such as Bugzilla or Track. These defect tracking systems are open source software freely available on the Internet. The Anomaly Report template from IEEE 829 discussed in § 9 above can be incorporated into the defect tracking system to ensure that all necessary defect data is captured. The defect tracking system will automatically store some of the test metrics discussed in § 12 that can be used to track test process progress.

4.16 No dedicated test team

Company X does not currently employ any dedicated software testers and all test-related activities are performed by developers. According to [62] small companies have limited resources and therefore do not invest in separate quality assurance staff. A software process improvement study [9] conducted in a small company also indicated that not enough resources were available to establish separate quality assurance group. This coincides with the view of [56]. In another study [42] conducted in 2007 around the testing processes of six organisations, five out of the six organisations had independent test teams. The study did not specify the size of the organisation, but did indicate that some of the organisations did have up to twenty developers. Company X currently has only eight developers. The ratio between developers and testers indicated that the organisations had between one to four developers per tester. [51] found that having a dedicated test role in projects allowed for testing to become more structured as well as allowing developers to focus more on fault finding at the unit level than at the system level of testing. A survey conducted by [36] in Australia regarding software testing practices indicated that 44 out of 65 organisations investigated had an independent test team. [38] recommends that small to medium sized enterprises (SME) employ at least one person that is dedicated to testing. If this is not possible, [18] recommends that some developers in the team be assigned the role of tester. This would mean that the same developer not tests his own code but that there exists some form of independence between development and testing.
The best possible solution for company X is to employ dedicated testers. Taking into account the current financial situation in South Africa, it will probably not be possible to employ testers in company X in the near future. Nevertheless, we recommend that company X employ at least two testers, when possible, that will function independently from the development team. This will provide company X with a ratio of 1 tester to 4 developers which will be sufficient for the

current workload and size of projects. Our long term vision for company X is that these dedicated testers form part of a separate quality assurance group that will eventually meet the requirements of the Test Organisation process area of the TMMi level 3.

4.17 Content of software project documentation not sufficient

Test Basis is defined by the ISTQB [24] as "all documents from which the requirements of a component or system can be inferred. The documentation on which test cases are based." [60] argues that insufficient test cases could be attributed to the low quality of the test basis. The developers of company X recommended that better requirements specifications be created so as to improve the testing process. Although the requirements specification is strictly speaking a software development process component and not a test process component, it has a direct effect on the quality of the test effort. According to [62] small organisations do typically not have clearly defined and stable requirements on software projects. In a study [14] conducted in 2007 regarding software process improvement (SPI) in a small companies in Finland, one of the problems identified was that of requirements documents that were not good enough. It was found that structured documents such as requirements documents were created on projects, but their contents and depth were deficient. A possible reason for the haphazard requirements documentation might be attributed to a lack of research in requirements engineering for small companies. According to [8] there is not one single paper published in the history of Requirements Engineering conferences that addresses the requirements processes of small companies. They further argue that this may be due to mistaken assumptions that small companies are no different than their larger counterparts or that small companies do not present noteworthy research challenges. Although not referring specifically to requirements engineering or testing, [12] also emphasize that hardly any publications exist that provide software engineering solutions specifically for small companies.
Based on the literature findings in this section, it is evident that the requirements specification forms an important part of the test process. Although the creation of requirements is not explicitly a test process activity, we recommend that company X improve their current requirements elicitation and specification processes. The implementation of requirements reviews discussed in § 5 will improve the quality of the test basis in company X.

4.18 No identification of critical bottleneck components for testing priority assignment

The developers in company X indicated that they do not know what to test and that they do not have enough time to spend on test-related activities. These problems were discussed in § 9 and § 13. Although more time may not be available to the developers, they must to prioritize their testing activities to test the most complex code or code most likely to contain defects.
In order to address this, [63] recommend static code analysis and visualisation techniques. Static analysis is a fault-detection technique that aims to evaluate a system or component based on its form, structure, content, or documentation, where the component or system's code is not executed [42]. Tools are available to automatically perform static analysis and we refer to these tools as Automatic Static Analysis (ASA). ASA tools automate the detection of defects and poor code constructs by parsing the source code and searching for certain patterns in the code [64].

According to [63] a typical pipeline for a static analysis tool includes the following components:

- Parser – Analyses the raw source code and produces and low-level representation thereof. The source code is usually represented as a syntax tree.
- Query engine – The query engine checks the source code for certain facts by scanning the syntax tree for specific patterns. These patterns include showing variables that are used before they are initialized, code modularity, and cyclomatic complexity to name a few. Cyclomatic complexity is the measure of the complexity of a software module. A module is defined as a single method or function in a programming language that has a single entry and exit point [65].
- Presentation engine – The queries can be represented visually to show the patterns uncovered with the query engine.

[63] integrated their own query engine with SolidFX, an existing Interactive Reverse-engineering Environment (IRE). This IRE has the same look as modern Integrated Development Environments (IDE) such as Eclipse or Microsoft Visual Studio. They applied their query engine together with SolidFX and Call-i-Grapher to a number of industrial C and C++ projects.
Their study revealed the following interesting patterns:

- Complexity hot-spots based on metrics such as McCabe's cyclomatic complexity and lines of code commented.
- The modularity of a system and its constituent modules based on its call graphs.
- The maintainability of a system based on specified metrics such as lines of code, lines of commented code, and cyclomatic complexity.

McCabe [65] argues that cyclomatic complexity should be limited because complex software is harder to maintain, harder to test, more difficult to comprehend and more likely to cause errors.

Based on the literature findings in this section, we recommend that static code analysis with visualisation be implemented in company X. This practice will be implemented during and after the coding phase of the software process. Static analysis during the coding phase can reveal defects in the code that can be corrected immediately, while static analysis after the coding phase can reveal complex hot-spots that are likely to contain more errors. Developers or testers can use static analysis to prioritize their testing activities by testing the most complex code first. Since most projects in company X are Java-based, we recommend FindBugs and SonarJ as static analysis and visualisation tools.

4.19 Summary

This chapter discussed the concrete problems that were identified in chapter 2 in the test process of company X through the use of relevant software engineering literature. To address these problems, various solutions were identified that could improve the test process of company X. This chapter emphasized the fact that company X is not unique in its shortcomings and deficiencies in software testing. From the items discussed in this chapter, not all are currently relevant to company X. The lack of tertiary testing education is not an issue that can be resolved in the short term. There must be closer collaboration between industry and academia to close the gap in software testing skills. Although automated testing is a recommended practice for software testing, it was also mentioned that only organisations with stable test processes should

embark on an automation strategy. We therefore recommend that all testing at company X be conducted manually until the test process reaches a more mature level. A dedicated test team that functions independently from development is recommended by most practitioners and academics. Due to the current recession in South Africa as well as a lack of buy-in from management, developers will have to be responsible for all testing activities.

The literature also provided valuable recommendations that are feasible for company X and we recommend that these practices be implemented with the improved test process discussed in chapter 5. Due to a lack of a dedicated test team, developers will have to improve their knowledge and understanding of testing. We recommend that all developers become ISTQB certified at the Foundation level and later attain the Advanced and Expert level certifications. Developers will be responsible for writing unit tests, using a TDD approach. This will force developers to write their own test cases. These practices will be emphasized in the test policy, along with the other test practices and guidelines. A test strategy for company X will be developed based on the test policy, to provide the developers with a high-level test approach. The test approach will incorporate a planning phase where a Master Test Plan will be created for each project according to the IEEE 829. We also suggest that company X establish a database with test metrics that will keep track of project history. This will be a valuable tool to track the improvement of the test process in company X. A defect tracking system will be used to capture and track defects. Most modern defect tracking systems also keep track of test metrics as discussed in § 12.

This chapter highlighted the shortcomings in the different parts of software test processes in small organisations, but there still lacks a standard generic test process that can be implemented by company X. This research indicates a definite need to create a basic, structured, and clearly defined test process. Chapter 5 shall suggest an improved test process, consisting of all the practices and improvements identified in this chapter.

5 Workflow Improvement

In chapter 4 we discussed the problems in the software testing process of company X and suggested solutions for each. The problems that were identified focused on the smaller

components that are part of the test process. These smaller components still do not constitute a complete process and must be placed into a context of a complete test process from start to finish. We will analyse the current workflow of company X's test process and identify the weak spots. Based on the suggestions in chapter 4, a new test process workflow will be presented that addresses the complete test process.

The Software Process Engineering Metamodel (SPEM) diagram depicted in Figure X below indicates the problem areas of the current workflow. SPEM is a metamodeling language created by the Object Management Group that can be used to define software engineering processes [22].

The problem area is indicated with an arrow and a rectangle box containing the number of the problem as identified in chapters 2 and 4.

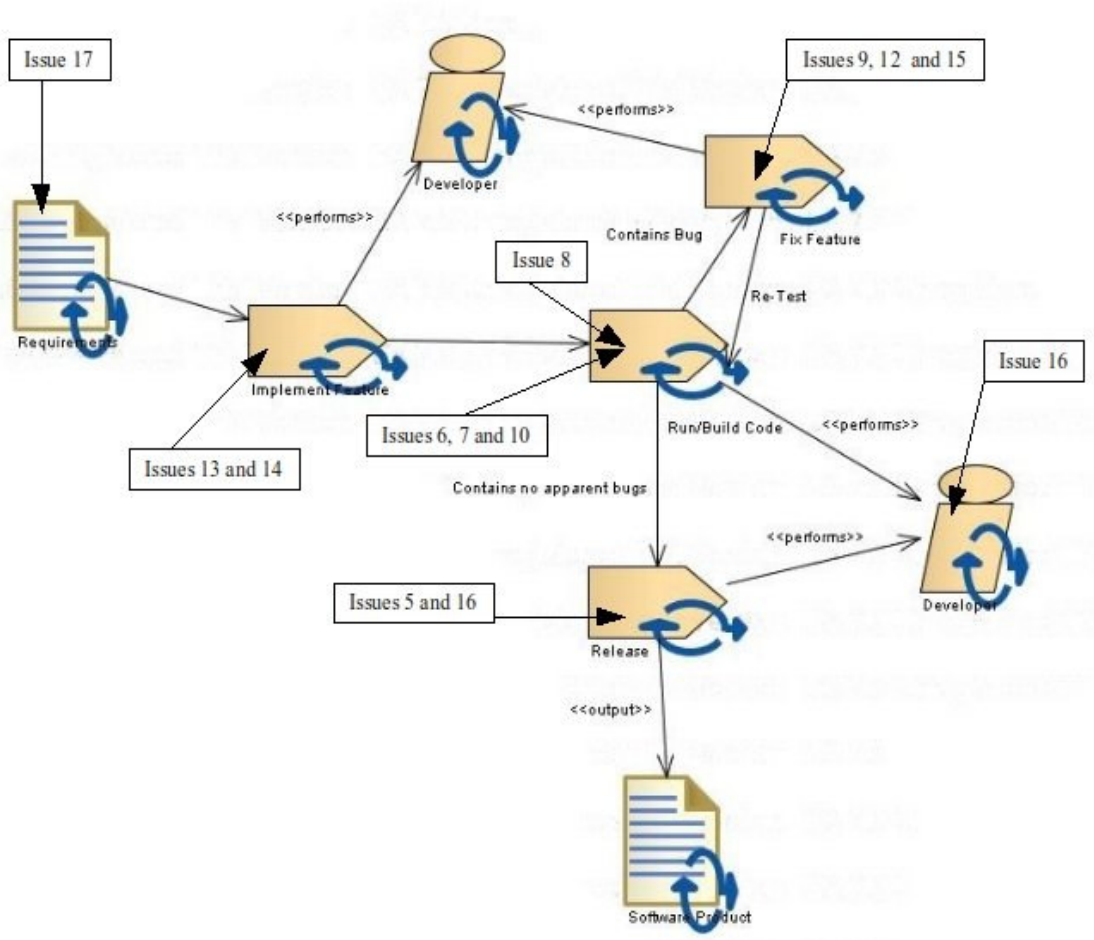


Figure 10: Current Workflow Indicating Problem Issues

The document-like icons in the figure represent work products of the process. The work product can either be used as input to the process such as the requirements specification, or as output in the case of a software product that is created. The rectangle figures with a sharp edge pointing to the right represent tasks in the process. In Figure 10, the 'Implement Feature' is an example of

such a task. The remaining human-like icon represents the roles in the process. The only role represented in Figure 10 is the Developer.

5.1 Description of Current Workflow and Problems Identified

The complete software process is not depicted in the workflow diagram. The problems that were identified stretch further than what can be seen in the current workflow process. These issues will be depicted in the improved workflow discussed in section 5.2 below.

5.1.1 Implement Feature

i. Issue 17

The requirements specification is the only documentation that is used to guide the test process in terms of what needs to be tested. This document is mostly not complete and is not updated when new requirements are identified by customers. As the requirements specification is a critical component of the test basis, there needs to be an improvement in the elicitation and ongoing management of this document.

ii. Issue 5

Project documentation such as the requirements specification, architecture and design specification are not reviewed to ensure that requirements are testable, and design is correct. Code is also not reviewed once it is written. There is thus no control over the quality and correctness of the code.

iii. Issues 3 and 14

Developers start implementing code before any test cases are designed. No unit tests are written by developers and production code is mostly littered with debugging statements.

5.1.2 Build and Run Code

iv. Issues 6, 7, and 10

This step in the software process is where the first testing activity takes place. This is problematic as no proper planning is performed to determine how and what needs to be tested. Apart from the lack of test planning, there is also no guidance for developers on what or how to test in the form of a test strategy. Company X does not have a test policy where developers and testers know what is expected from them in terms of software quality and testing.

v. Issue 8

Once a feature is completed and ready to be tested, it is not moved into a properly managed and configured test environment. The code stays mostly on the notebook computer of the developer where it is sometimes tested on a virtual machine also hosted on the developer's notebook computer.

vi. Issue 9, 12 and 15

The developer that implemented the feature then proceeds to test it by executing the code and monitoring debug statements or program flow. When a defect is detected, the developer would either proceed to immediately try and resolve the defect or scribble it on a piece of paper to resolve it later. There is thus no test documentation such as defect reports that clearly describes the defect and how to reproduce it. The project manager or development manager is also not always aware of these defects as they are not entered into a defect tracking system. Some defects remain unresolved as developers sometimes lose track of identified defects. This lack of measurement makes it difficult to track the progress of the test process as there are no metrics to indicate the number of defects found and number of defects still unresolved. As no test cases are written (as indicated in issue 14) there is no way to determine how much testing is still needed. Developers will continue testing until they cannot "find" any more defects.

vii. ***Issues 11, 13 and 18***

Developers indicated in the survey that not enough time is allocated to testing (issue 11). Once they start testing they also indicated that they do not know what to test or where to start. Therefore testing is rushed and there is no prioritization (issue 18) of the features to be tested. The lack of proper test planning (issue 13) causes this haphazard approach to testing.

viii. ***Issue 16***

Due to the lack of resources in company X, especially a team of dedicated testers, all testing activities are performed by the developers. Although unit testing is traditionally the task of the developer, integration and system testing should be conducted by professional testers.

5.1.3 Fix Defect

ix. ***Issue 15***

As already indicated in section vi above, there is no defect tracking system. Once defects are resolved, they must also be marked as resolved in a defect tracking system so that the overall test progress can be measured.

5.1.4 Release Product

x. ***Issues 5 and 16***

This is the final stage of the software process in company X. Code is packaged and prepared for deployment at a customer without going through a formal code review or inspection. The quality of the code has thus not undergone any form of peer review to identify improvements or weak spots. The software produced by the developers does not pass through any form of quality assurance or testing by professional test staff. All code is written by one or more developers, "tested" by the same developers, and from there it is deployed at the customer.

5.1.5 Discussion of Current Workflow

The current workflow does not include all problems identified as the workflow itself is not structured or clearly defined. Some of the problems identified come into play early in the software process and is therefore not depicted in the current workflow as it only occurs at the end of the software process. There are no phases or deliverables defined for the current workflow. The current workflow is clearly insufficient as it is very isolated and not integrated into the software development process.

5.2 Improved Workflow

The improved test process workflow for company X will be discussed next. This workflow consists of various phases that are described in chronological order. Each phase consists of specific activities. The test policy and test strategy will not be phases of the project, but merely used as input to the test process. However, their importance to the test process cannot be overemphasized and therefore we have added them to this discussion.

5.2.1 Test Policy

The results of the survey indicated that developers do not know what is expected of them in terms of testing or software quality. There is not a single mission statement or vision for software quality and testing by company X. The test policy is a company-wide policy that will guide company X in its test improvement process. It will establish the foundation from which all test-related activities will be derived from. The test policy will clearly state company X's commitment to quality and provide a high-level organisational approach to software testing. The test policy provides the foundation for the development of a test strategy. The test policy will only be defined once and remain a working document that evolves with the company's test process. The test policy will address problem issue 7. Based on the test policy, we will define a test strategy.

5.2.2 Test Strategy

As indicated through the interviews, the developers do not know how or what to test. We recommend that a test strategy be developed that is based on the test policy. According to the TMMi [2], the test strategy should be based on the test policy and address the following issues:

- A test model (for example the V-Model) to integrate the test process into the software life cycle.
- Test levels that define the different levels and components to be tested at each level. These levels include unit testing, integration testing, system testing and user acceptance testing.
- The test case design techniques to be used at each test level. This includes black box and white box test design techniques.
- The type of tests to be used at each level such as black box and white box testing.
- Exit criteria at each test level. According to Craig and Jaskiel [66] exit criteria is the “metrics specifying the conditions that must be met in order to promote a software

product to the next stage or level". An example of such a metric is code coverage. Code coverage indicates the percentage of code that is covered by a test case.

- Standards that must be adhered to such as the IEEE829 [31] and coding conventions.
- Description of test environment in which testing will be conducted.

Company X is using the Waterfall model as software process and we therefore recommend that the V-Model be implemented alongside the software process. The V-Model will complement the Waterfall model as it follows the same structure and flow of software process phases. We suggest that the test process of company X implement all 4 levels of the V-Model as mentioned above. We recommend that all the above-mentioned issues be addressed in company X's test strategy. The test strategy in company X will be defined as an organisation-wide test strategy that can be used as a guideline on all projects. A test strategy can also be defined per project, but due to limited resources and test management staff, as well as the similar nature of software projects in company X, we recommend that a single test strategy be applied to all projects. The test strategy will be the first point of reference for starting the test process on a project. The test strategy will address problem issue 10 identified in chapter 4.

As the test policy and test strategy is not defined per project, they are not strictly seen as being part of the test process. Nevertheless, their importance to the test process cannot be overemphasized. The following phases will be included in the improved test process of company X and become part of each project. The first of these phases is the test planning phase.

5.2.3 Test Planning

No upfront planning of the test process is currently conducted in company X. This usually results in an ad-hoc test approach that is rushed and without any form of structure. Many developers complained that not enough time is allocated to testing. The amount of time that is actually spent on testing was reported as being less than 10%. We recommend that the test planning phase commence at the start of the software process, along with the requirements phase of the project. This will ensure that everyone in the project team knows what test activities need to be performed and where in the test process these activities are to be performed. The test planning phase of company X will include the following activities as recommended by the TMMi [2]:

- Product risk assessment. The risk assessment will identify the critical areas of the product to be tested. After the risks have been identified, they have to be prioritized. The outcome of the risk assessment is a product risk list.
- Based on the risk assessment, we will define the test approach to be used in the project. The first step in defining the test approach will be to identify the items and features that need to be tested. We will use the software requirements specification from the requirements phase of the project and risk list from the product risk assessment to define these items and features. The test approach will also include the test design techniques that will be used on the project. The test design techniques will be selected based on the risks identified. Different test design techniques are used, depending on the type of product and risks involved. A system that manipulates financial data has higher risks involved and needs a different test approach than a web application that merely provides company information. The test coverage for the different features must also be specified.

Test coverage will be influenced by the test design technique used. Using the same example as above, a financial system will have to be tested with test design techniques that provide higher test coverage than a non-mission critical system such as a company website. We recommend that the test planning be aligned with the test strategy of the company X and that the test approach be reviewed with all stakeholders.

- Entry and exit criteria need to be defined to determine when testing can start at a certain test level and when testing can continue to the next test level respectively. An example of an exit criterion is that 80% unit testing coverage be attained before integration testing can start. An example of an entry criterion is that there are no outstanding defects left from integration testing and system testing can therefore commence.
- We also recommend that suspension and resumption criteria be defined as part of the test approach. These criteria determine when part of the test process or the whole test process must be stopped. For example, if the test environment is not representative enough of the production environment or too many critical defects are discovered during integration testing, the test effort can be stopped to ensure that these issues are resolved before testing can be resumed.
- The test effort must be estimated to determine the resources that will be needed as well as how long the test process will take. We recommend that company X develop a work breakdown structure (WBS) in each project. Along with the WBS, we recommend that the test life cycle phases be defined for the project. The life cycle phases include test planning, test design, test execution etc. The effort needed for each life cycle phase must be estimated along with the different milestones and deliverables for each phase. The test environment requirements must also be planned for and additional costs and time to implement it must be estimated.
- The final activity of the test planning phase will be the creation of a test plan. As suggested earlier, we recommend that company X use the Master Test Plan (MTP) template from the IEEE829 as a reference. The test plan will contain the results from all the test planning activities above as well as the test staff that will be involved. Testers that do not have the skills needed to perform the testing needed must be trained in-house to acquire these skills.
- One of the problems identified during the investigation of the test process in company X was that of poor requirements specifications (See issues 5 and 17). By reviewing the requirements specification during the test planning phase, we can ensure that all requirements are testable and consistent.

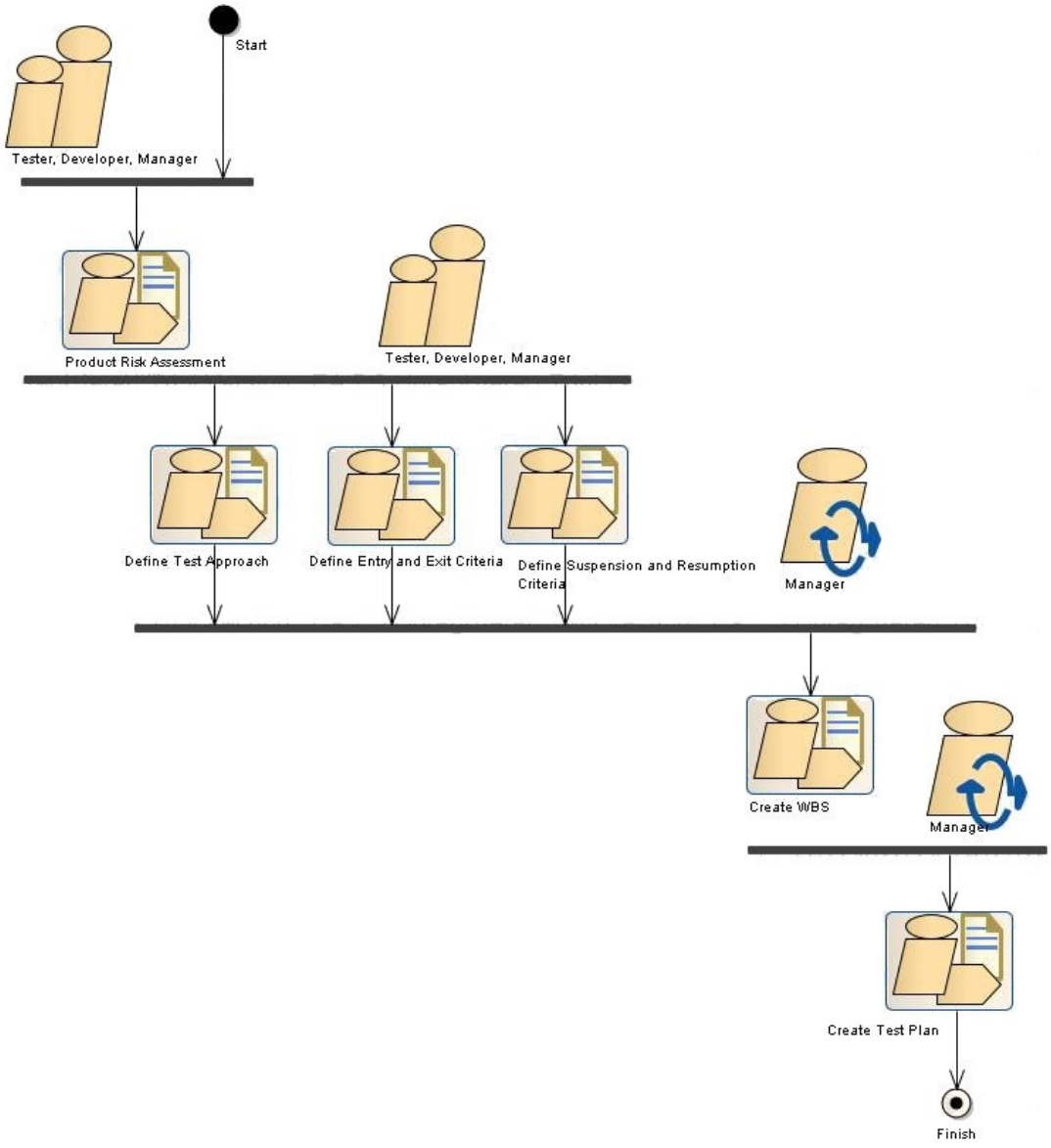


Figure 11: Test Planning Phase of Improved Workflow

Figure 11 depicts an activity diagram of test planning phase of the improved workflow process in SPEM notation. Test planning starts with a product risk assessment with involvement from the developer, tester and manager. Upon completion of the product risk assessment, a test approach is defined, along with entry and exit criteria, as well as suspension and resumption criteria. The manager proceeds to create a work breakdown structure (WBS) of all testing activities to be performed during the project. The deliverable of the test planning phase is a test plan according the IEEE829 standard.

This concludes the test planning phase. The test planning phase will start alongside the requirements phase of the software process nears completion. The next phase of the test process that we recommend for company X is the test design phase.

5.2.4 Test Design

The design phase of the test process occurs after the design phase in the software process. We recommend that the following steps be taken in the test design phase:

- The test design techniques identified in the test planning phase will be used to create test conditions and test cases. A test condition is defined by the ISTQB [24] as an "item or component that could be verified by one or more test cases". The test conditions are used to derive the test cases which are defined as a set of inputs, execution conditions and a set of expected results by the IEEE829 [31].
- The aim of the test design phase is to design test cases that will achieve the largest possible test coverage and to meet the coverage goals set in the test planning phase. We recommend that white-box and black-box test design techniques be selected for the test levels that will follow during the execution phase. Black-box test design techniques include equivalence partitioning, boundary value analysis and state transition testing to name a few. White-box test design techniques include statement testing, branch or decision testing, and condition testing. We agree with the TMMi [2] and recommend that at least one test design technique be used per test level.
- Once all test cases have been identified and prioritized, a requirements traceability matrix must be created according to IEEE829 [31]. This document will be used to trace test conditions and test cases back to the requirements. It is important that everyone knows which requirements will be verified by which test conditions and this will be addressed through the requirements traceability matrix.
- Based on the test cases, a list of test procedures must be developed. The IEEE829 defines a test procedure as "detailed instructions for the setup, execution, and evaluation of results for a given test case".
- We then recommend that test data be created that will be used throughout the test execution phase. After test data are created, we recommend that the test environment be implemented. Due to the problem identified in issue 8, we recommend that company X implement a test environment library. This library will store preconfigured virtual machines that will contain the various operating system platforms. We suggest that this library be carefully managed so as to be easily available for each test project. Developers in company X are spending too much time configuring test environments with every project and no preconfigured virtual machines currently exist for this purpose.
- A final step in the test design phase is to review the design specification created during the software process design phase. This will allow the test team to verify the consistency of the design with the requirements specification and ensure testability of all components.
- The deliverable of the test design phase is a test design specification created according to the IEEE829. This document will capture the output from all the activities above.

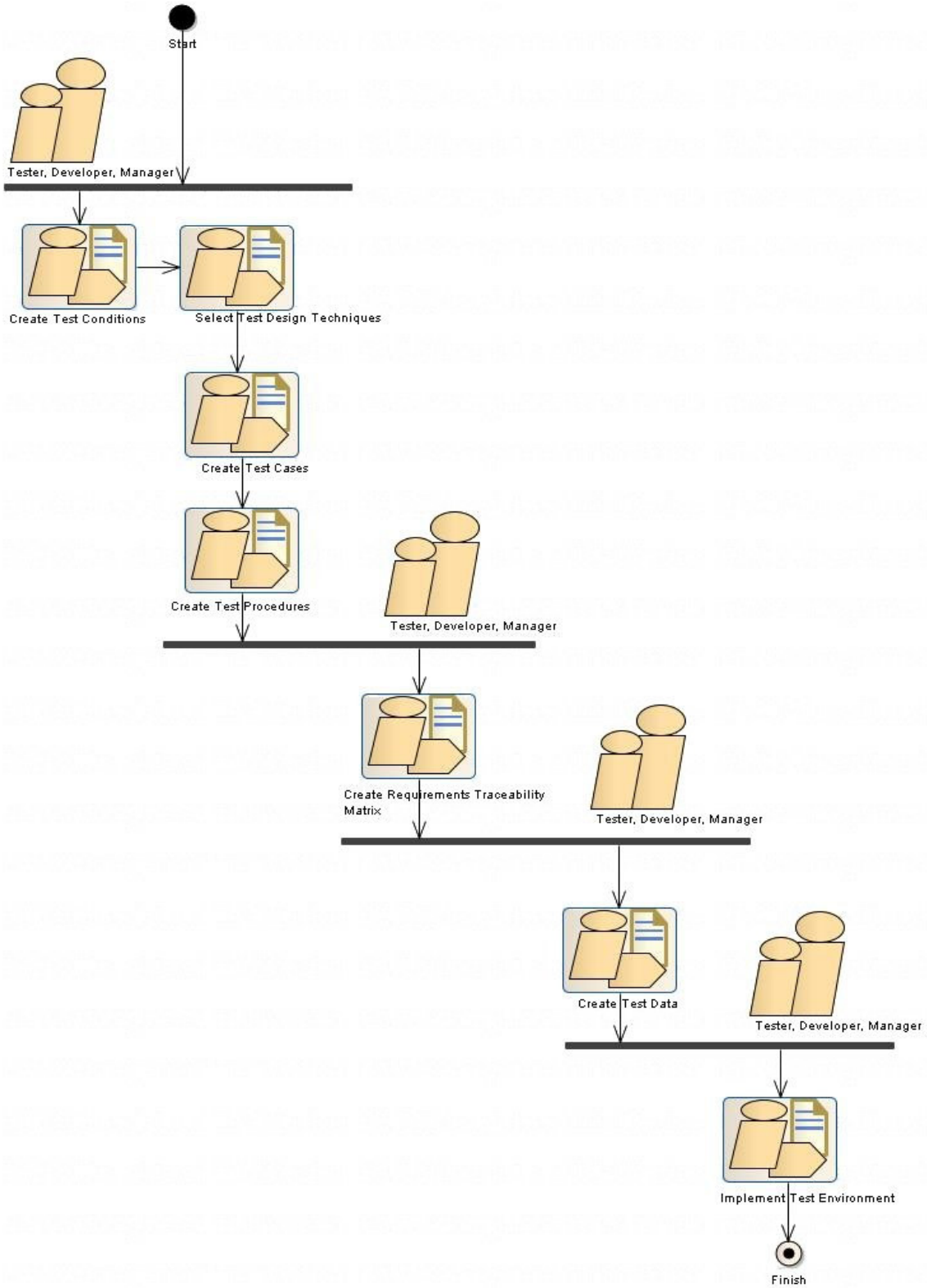


Figure 12: Test Design Phase of Improved Workflow Process

Figure 12 depicts the activity diagram of the test design phase of the improved workflow process in SPEM notation. The process starts with the creation of test conditions by the manager, developer, and tester. Based on the test conditions, test case design techniques are selected to create test cases that will achieve the desired test coverage. Test procedures are created from the test cases that contain detailed instructions on how to execute the test case. The three same roles then proceed to create a traceability matrix. This ensures that each test case is tied to a requirement to ensure that all requirements have corresponding test cases. This step is followed by the creation of test data along with the implementation of the test environment. The deliverable of the test design phase is a test design specification according to the IEEE829 [31] standard.

This concludes the test design phase of the project. The next step in the test process is the test execution phase.

5.2.5 Test Execution

The test execution phase is where code is executed to test whether it performs as expected. We suggest that company X include the following activities in the test execution phase:

- Before the test execution actually starts, an intake test should be performed to determine whether the system is ready to be tested and whether the test environment is properly configured. The validity of the test data will also be determined by the intake test. Should the intake test point to problems with certain test cases or data, these issues should be revised before continuing with the test execution.
- The first level of test execution will be at the unit level. As stated in problem issue 3, developers in company X do not write any unit tests. We have addressed this problem by recommending the implementation of Test Driven Development (TDD). Developers will thus be responsible for writing unit tests as they implement their code. The unit test cases designed and developed during the test design phase will be executed by the developers during the unit test execution phase. Should any bugs be encountered that cannot be resolved immediately, developers should log the bugs in the defect tracking system. The implementation of a defect tracking system in company X will address issue 15.
- The integration, system and user acceptance level testing will follow the unit testing level. Issue 16 identified a need for company X to have dedicated testers. Therefore we recommend that company X evaluate the possibility of employing a tester in the near future. The dedicated tester will be responsible for performing testing at the integration and system levels.
- With a dedicated tester as part of the test team, we recommended in issue 18 that code visualisation and static analysis techniques be used to determine which components are most complex and will most likely to contain more defects. Using these techniques, developers and testers can prioritize their testing tasks to test the most important code first.
- Defects that are not resolved during each test level must be analysed to determine their severity and impact on the system. Defects that can be resolved should be returned to development.

- Once new features have been implemented or defects corrected by developers, regression testing must be performed to ensure that no new defects have been introduced. The iteration of these develop-test-fix cycles must continue until the exit criteria for each level is reached. Amman and Offutt [67] suggests that regression testing is an essential part of the test process as small changes to one part of a system often cause many problems in many other parts of the system. They also argue that regression testing must be automated using some test automation tool.
- The final level of testing is user acceptance testing where the internal users or the actual end-users of the system are given the opportunity to test the system. Most defects should have been resolved at this stage of the test process. Nevertheless, all defects found by user should also be logged in the defect tracking system. Once user-acceptance testing has reached the exit criteria defined in the test plan, test execution can be concluded.

The test execution phase is depicted in the Figure 14 on the page 51. The test execution phase starts with an intake test to determine whether the test environment and test data is ready for testing. Developers with create unit tests using TDD as code is being implemented. Developers will resolve defects that are found during unit testing or resolve them if possible. Regression testing will be performed as new components are developed to ensure that no new defects are introduced. Upon completion of unit testing, the tester will perform integration testing of the components. Defects found by the tester will be logged in the defect tracking system and returned to development. System testing will follow where all components will be tested together as a complete system. The tester will once again log any defects found in the defect tracking system. Transition from one level to the next will be determined by the exit criteria set the in the test planning phase. The final test activity for the execution phase is user acceptance testing. This will allow the users of the system to perform testing. Any defects found will also be logged in the defect tracking system and be returned to development. Once the exit criteria of the user acceptance phase have been satisfied, testing can be stopped.

5.2.6 Test Monitoring and Control

The test monitoring and control phase of the test process will continue throughout the test process from the test planning phase to the end of the test execution phase. The test process will be monitored at each of the defined phases. The following activities are recommended for this phase of the process:

- One of the issues (issue 12) identified in company X was that there were no metrics. No test activities are currently measured and there is also no historic test data to track test process progress or base predictions on. We therefore recommend that a database of metrics be established and continually updated during test monitoring and control phase of the test process.
- It is important that the goals set in the test plan are continuously measured against the

actual progress of the test process to ensure that deviations from the project plan are managed and corrective action is taken. This will include the creation of test logs as recommended by the IEE829 [31]. After testing at each level is completed and the defined exit criteria of each level have been reached, we recommend that a test log be created per level. The test log will contain information such as the specific test cases executed, number of test cases executed, number of test cases failed, number of test cases passed etc.

- Another test activity that should be performed throughout the software process is the review of documentation and code. We recommend that regular code inspections be held to ensure that developers are using sound coding practices and are adhering to coding standards. The review of project documentation and code addresses issue 5 and will ensure that defects are detected early and do not propagate to the later stages of the test process.

The final phase of the proposed test process for company X will include reporting and closure.

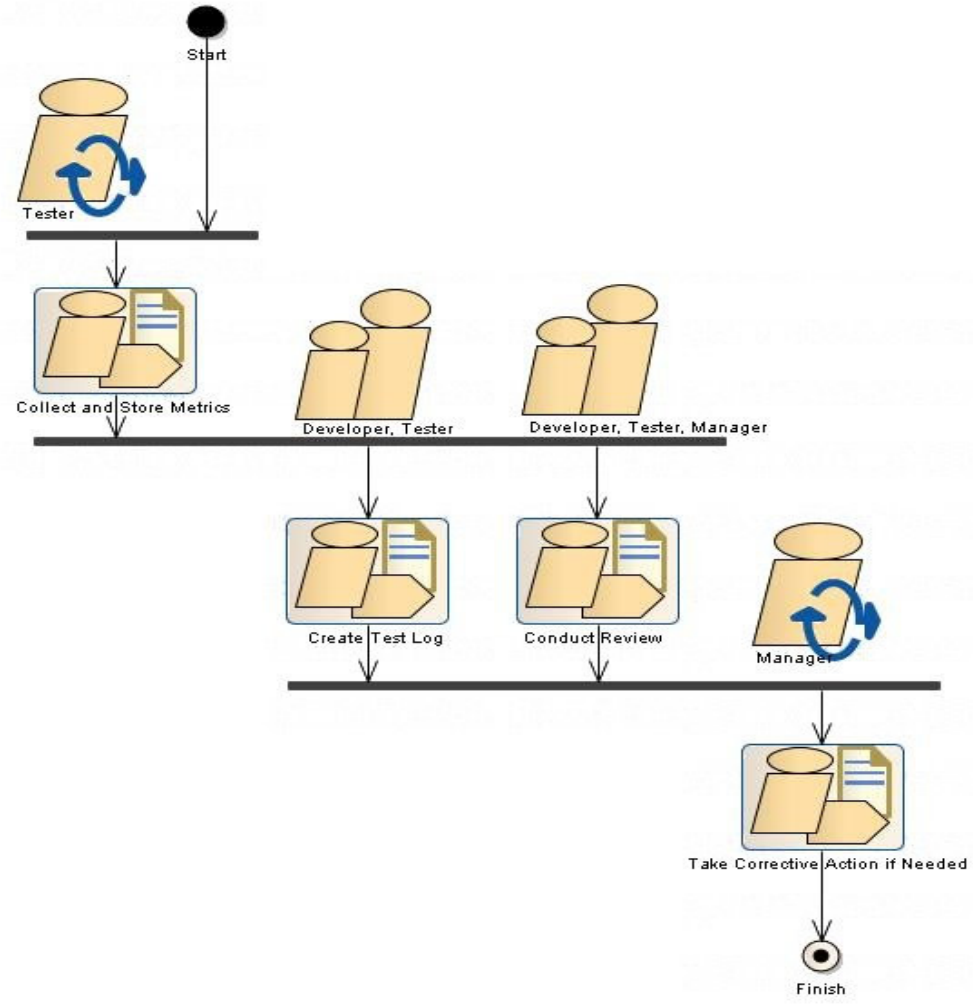


Figure 13: The Monitoring and Control Phase of Improved Workflow

Figure 13 above depicts the test monitoring and control phase of the improved workflow. This phase will be a continuous phase runs throughout the test process. During test monitoring and control metrics will be collected to determine the progress of the test process, for example test coverage metrics will be used to determine whether the coverage for a certain test level have been satisfied. Documentation and code will be reviewed during the process to ensure that a certain level of quality is maintained, for example whether the requirements and design documentation are consistent, or whether developers are adhering to company coding standards. Upon completion of each test level, a test log will be created to communicate the progress of the test process. Corrective action must be taken should the test process deviate from the test plan.

5.2.7 Test Reporting and Closure

Test reporting will play an important role in the test process to convey results to stakeholders as well as tracking test progress. We recommend that the following activities form part of this phase:

- Apart from the test reporting in the form of test logs and defect reports discussed in the test monitoring and control phase and test execution phase, a test summary report will be produced to convey the overall results of the test project. We recommend that this test summary report be based on an industry standard as can be found in the IEEE829 [31]. The test summary report will include items such as a summary of testing activities performed, the results of the executed test cases, outstanding defects and their severity or impact on the system, as well as a summary of the metrics collected during the test process.
- We recommend that the test report be distributed to all stakeholders at the end of the project. In the test closure phase we recommend that all test artefacts or testware such as test data, test documentation, and collected test metrics be safely stored in a database for future use.
- The test environment should also be cleaned up and backed up for future use. The testware will provide valuable data for future test projects and this historic data can be used to track test process improvement over time.
- As a final activity of the test process, we recommend that company X conduct a post-mortem meeting of the test project to identify weak spots in the test process as well as possible room for improvement.

This concludes the test reporting and closure phase. We suggest that all test project documentation be stored in a configuration management system on completion of the project.

Figure 15 depicts the test reporting and closure phase in SPEM notation.

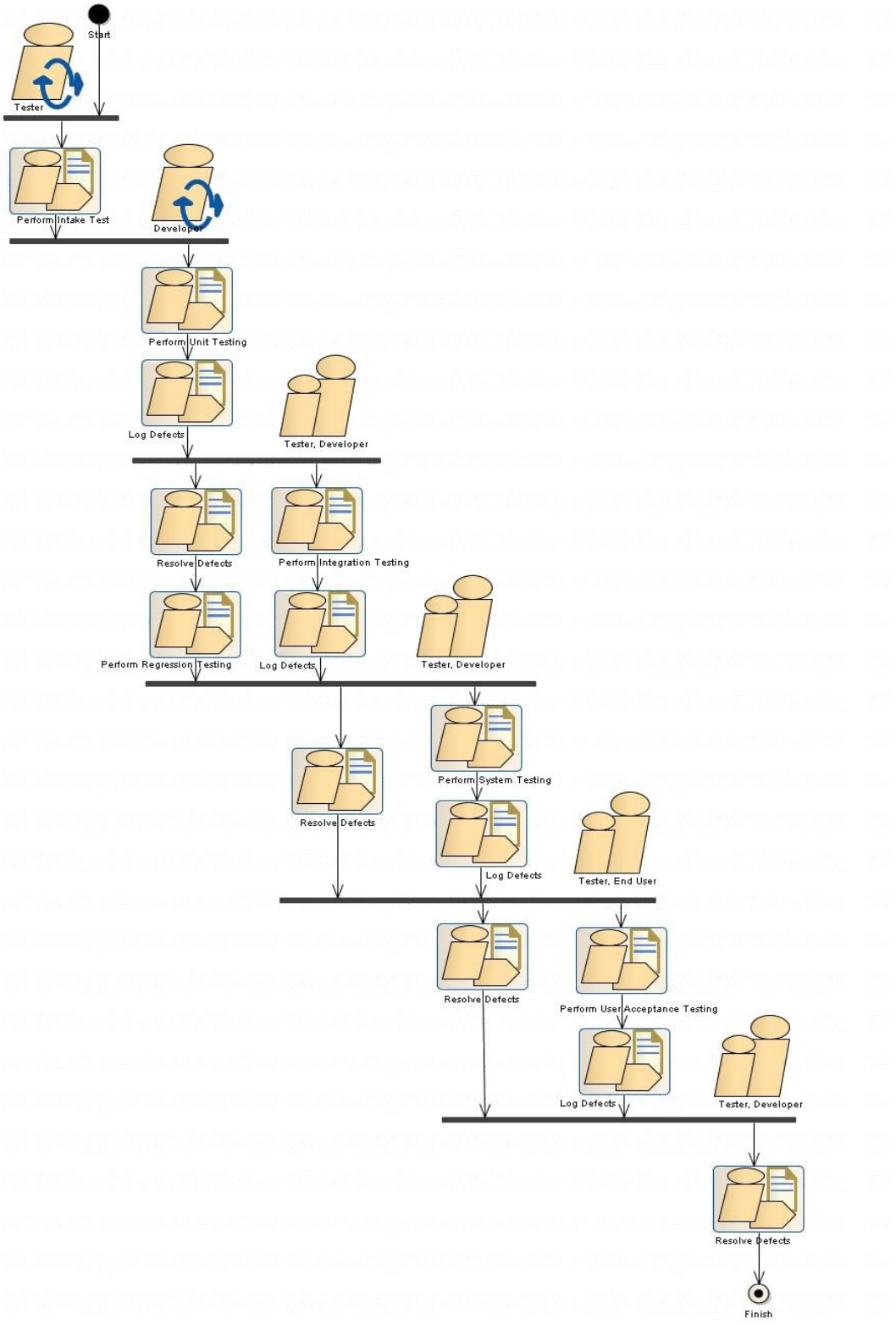


Figure 14: Test Execution Phase of the Improved Workflow

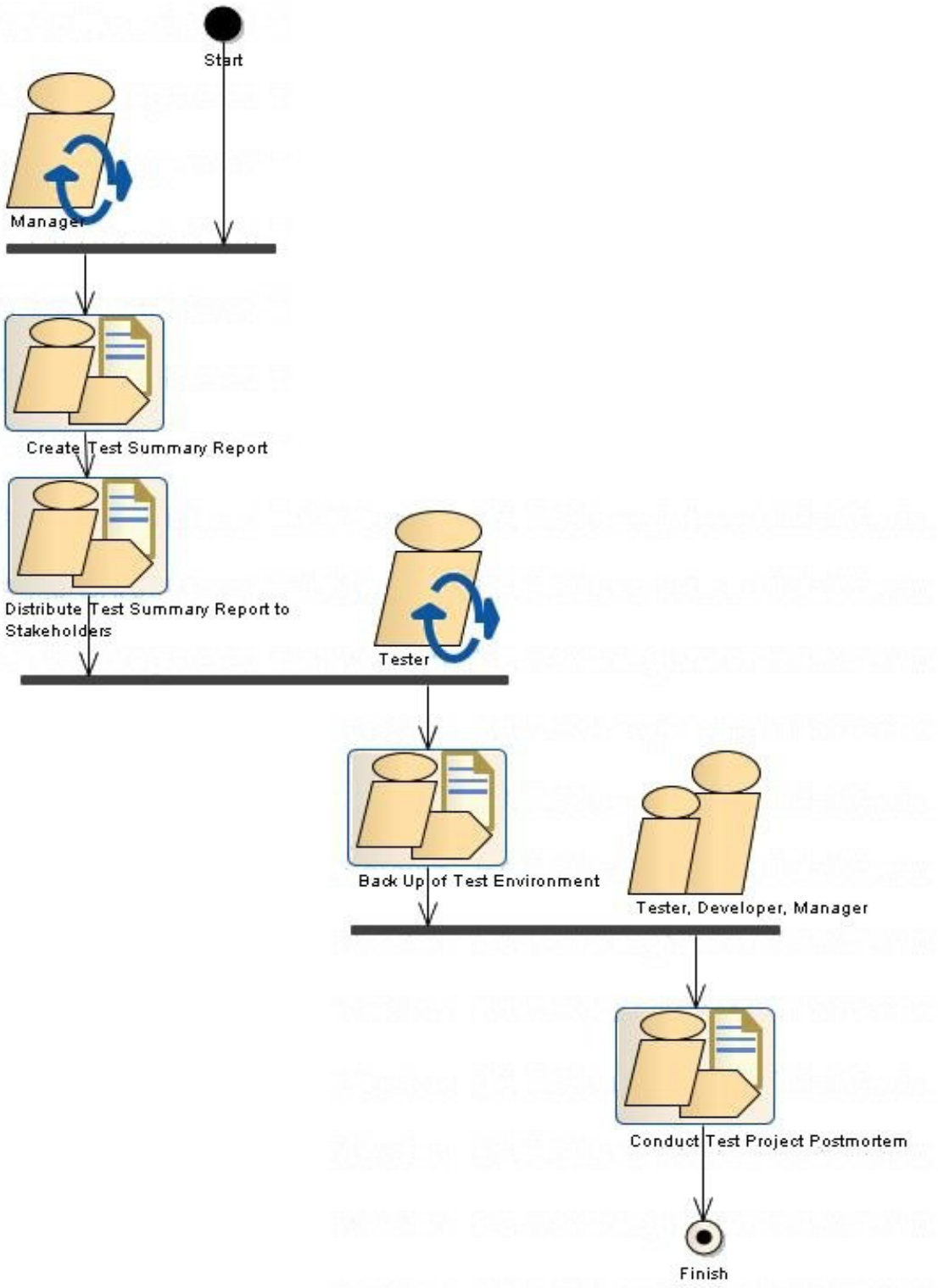


Figure 15: Test Reporting and Closure Phase of the Improved Workflow

The test reporting and closure phase depicted in Figure 15 above is the final phase of the test process. A test summary report is created by management to convey the overall test results of the project. This report is distributed to all the other stakeholders. The test environment together with the test data must be backed up in case further testing is required. The test project ends with a project post-mortem meeting to identify the weak spots in the test process as well as areas where improvement may be necessary.

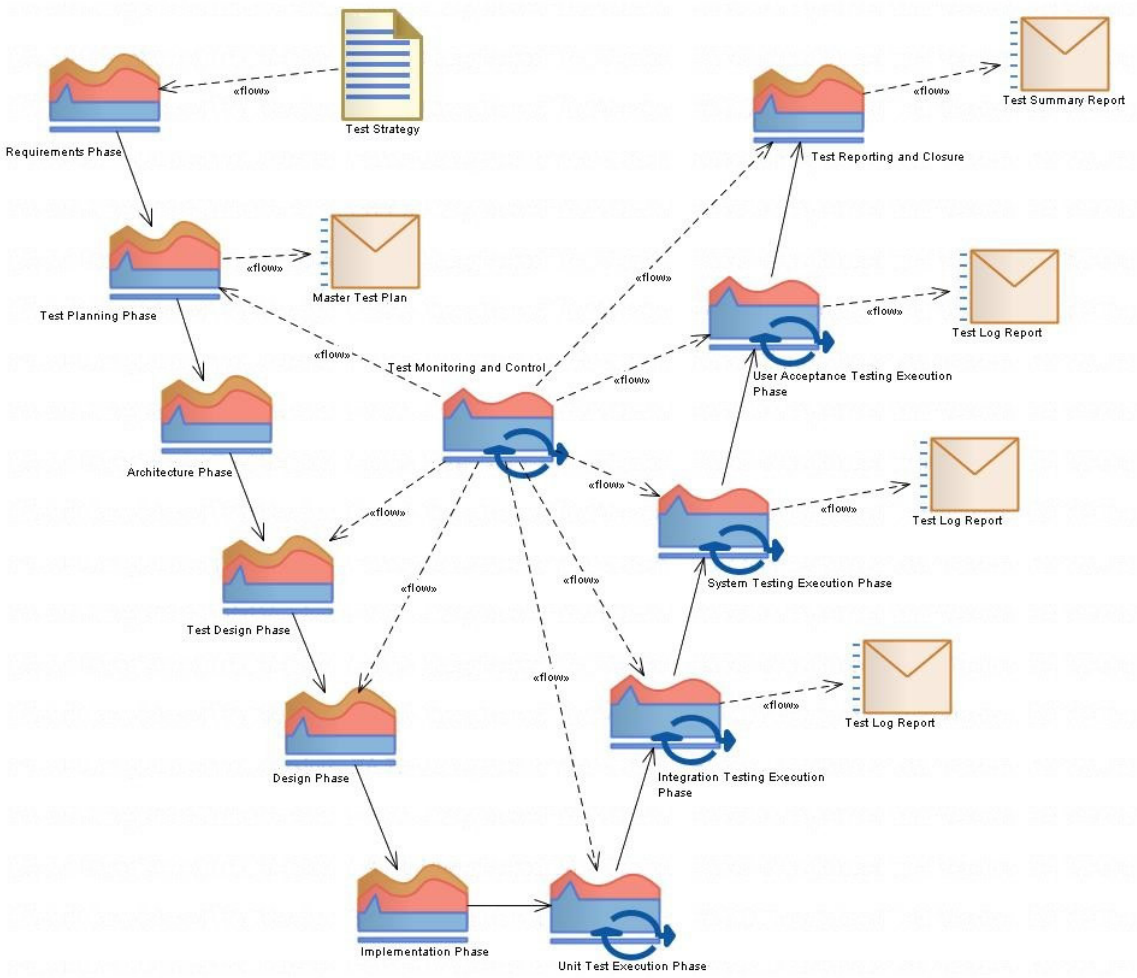


Figure 16: Improved Workflow Process of Company X

Figure 16 depicts the high-level flow of the improved workflow process for company X. The blue and red icons depict the different phases of the project. This diagram also illustrates the use of the V-Model where testing activities are integrated into the software process. The icons resembling envelopes represent the deliverables of the different phases. The icon resembling a document represents the test strategy that serves as input to the test process. The blue circular arrows on the right of the diagram indicate iterative phases of the process. This is where testing must be repeated after defects have been resolved and until the required exit criteria have been satisfied.

5.3 Discussion

In this chapter we assessed the current test workflow in company X and recommended solutions to the problem areas identified. The current workflow is clearly insufficient and only spans a very short area of the entire software process. There is no structure to the current workflow and developers mostly perform testing in an ad-hoc and chaotic manner. Therefore we suggested an improved workflow that defines a complete and structured process. This proposed workflow is now tightly integrated into the software process of company X. We have also created clearly defined activities and deliverables for each phase of the test process. Many of the suggested solutions for the improved workflow were derived from the process areas of the TMMi model. Therefore we would like assess the TMMi maturity of the improved workflow to determine how it compares to the old workflow maturity level.

6 Assessment

In order to determine how effective the suggested improved workflow for company X can be, we must assess it according a chosen standard, here the TMMi standard. In this chapter we will assess the improvements suggested in chapters 4 and 5 and assign a TMMi maturity level to the improved workflow. As discussed in chapter 3, each maturity level of the TMMi is made up of process areas. The process areas in turn consist of specific goals that are supported by specific practices. Generic goals are also specified for each process area and they can be found in more than on process area. Generic goals are supported by generic practices. The TMMi is a very comprehensive test maturity model with strict requirements regarding the achievement of process areas and we suggest that company X first institutionalize the basic practices specified in the improved workflow before attempting to satisfy all specific and generic goals and practices of the TMMi. The main reason for this decision is that company X does not have any dedicated test resources such as testers and a test manager. Embarking on such a test process improvement program will need dedicated staff to ensure its success. For purposes of this assessment we will assume a process area as satisfied if at least one of the specific goals in that process area is implemented. We will award an 'X' to a specific goal of a process area in the assessment tables below if more than one of the specific practices of that specific goal has been addressed by our improved workflow. Although this would not suffice for formal accreditation with the TMMi Foundation, it provides a valuable indication as to the possible maturity of the improved workflow.

The following section lists all the process areas of the TMMi level 2 and level 3 and discusses the test activities of the improved workflow of company X in each.

6.1 TMMi Level 2

a.) Test Policy and Strategy

This TMMi level 2 process area requires that an organisation establish a test policy and test strategy. The test policy must include the goals of testing and be distributed to stakeholders. A test strategy must also be defined and distributed to stakeholders. The last specific goal of this process area is to establish test performance indicators such as number of defects found, test effort and cost, and defect detection percentage.

For company X we recommended that both a test strategy and test policy be created as the starting point for their test process improvement program. The planning phase contains test effort estimation as well as the definition of exit and entry criteria. The test monitoring and control phase includes metrics that will be collected during the test process. In light of these recommended improvements we can safely say that this process area is achieved by the improved workflow.

b.) Test Planning

The Test Planning process area requires that a product risk assessment be performed where risks are categorized and prioritized. Once completed, a test approach must be established. This includes identifying the items and features to be tested, definition of entry and exit criteria as well as definition of suspension and resumption criteria. The test approach is defined next and it includes the selection of test design techniques and assignment of test coverage for each item to be tested. Establishing test estimates should include a WBS,

definition of the test life cycle, and estimation of test costs. The last two specific goals of this process area include the creation of a test plan and obtaining commitment to the test plan.

In the improved workflow we implemented a test planning phase that starts off with a risk assessment, followed by the definition of the test approach where the test design techniques are defined. Once test design techniques have been identified, test coverage is defined for each item or feature to be tested. We recommended that entry and exit criteria be defined to determine when testing can proceed to the next test level. Along with the entry and exit criteria, we also suggested that suspension and resumption criteria be defined to determine when then test effort needs to be suspended temporarily. A WBS is created to determine the test effort, along with the test life cycle phases. Our recommended test planning phase concludes with the creation of a master test plan. The test plan must be reviewed by stakeholders and all stakeholders must commit to the plan.

Based on these recommendations, we want to implement all 5 specific goals of this process area. Therefore we can ascertain that this process area of the TMMi is satisfied in our improved workflow.

c.) Test Monitoring and Control

Test monitoring and control involves monitoring the progress and quality of the test process and implementing corrective action when the process deviates from the test plan. This process area of the TMMi requires that all components specified in the test plan be constantly monitored such as test project risks, test environments, test incidents, exit criteria etc. Should these components not proceed according to the test plan, the issues must be analysed and corrective action must be taken.

For the improved workflow we have defined a monitoring and control phase that spans the entire life cycle of the test process. We have emphasized the importance of collecting metrics during the process. Reviews of project documentation and code will improve quality and reduce propagation of defects to later test levels. Test logs were recommended at the end of each test level to indicate the progress of the test effort.

The TMMi Test Monitoring and Control process area contains 17 specific practices. We have selected only a few practices to add to our improved workflow due to two reasons. The first is that company X is only starting its test process improvement program and it needs to be simple and relatively easy to implement. We do not want to introduce too much complexity into the process at this early stage, but rather introduce new practices over time. The second reason for limiting our monitoring and control phase to basic practices is that company X does not have enough human resources to manage the implementation of these practices. Based on these suggestions, we can still ascertain that the improved workflow meets the requirements of this process area as at least two of the specific practices are met.

d.) Test Design and Execution

The Test Design component of this process area involves the design of test cases using test design techniques, the creation of test data, execution of the test cases, and managing test incidents to closure. More specifically the design component of this process area is made up of 8 specific practices. These include identification and prioritization of test conditions

and test cases, identification of necessary test data, maintaining traceability with requirements. Once these components have been identified, test procedures are developed along with the creation of test data.

The Test Execution component of this process area requires that an intake test be performed, test cases are executed, test incidents are reported and the creation of a test log. Unresolved test incidents must be tracked throughout the process and managed to closure.

We have recommended that a test design phase be incorporated into the improved workflow process. We based our recommendations for the design phase almost entirely on the TMMi specification and have therefore implemented most of the specific goals of the design component.

With the development of the test execution phase, we have also followed the TMMi specification closely as most of the specific goals for test execution are critical to the test process. The only specific goal of the test execution component that we cannot implement is the management of test incidents by a configuration control board as company X does not currently have such a board. We recommend that this goal can be sufficiently managed by the project team along with involvement from the customer. Based on these findings, we can ascertain that the improved workflow implements most of the specific goals of this process area. Therefore we conclude that this process area is satisfied by the improved workflow.

e.) Test Environment

The final process area of the TMMi level 2 involves the design, implementation and management of the test environment. The first specific goal for this process area includes the development and analysis of the test environment requirements. The second specific goal involves the implementation of the test environment, creation of test data, and execution of an intake test. This process area concludes with the management and control of the test environment by carefully managing access and usage of the system.

One of the problems identified with the current test process of company X was the availability, management and use of test environments. We have addressed this issue in our general recommendations by stating that a test environment library be created to store the different preconfigured virtual machines. As part of the suggested improved workflow, we included test environment requirements elicitation during the test planning phase. The test design phase includes the implementation of the test environment. Our suggested test execution phase starts with an intake test and all testing will be performed within carefully managed test environments.

During the problem analysis of the test process in company X, we have realized the importance of proper test environments. Therefore we have emphasized this in the improved workflow. Based on the improvements made in the suggested workflow, we can state that at least two of the specific goals of this process area of the TMMi are satisfied.

6.2 Summary of TMMi Level 2 Assessment

With the development of the improved workflow, we have carefully followed the recommendations made by the TMMi level 2. We attempted to include all of the process areas

into our new workflow and this was achieved. However, the TMMi is a very comprehensive model with many specific and generic goals, in turn containing even more specific and generic practices. We have selected, in our opinion, the most important goals and practices from each process area. It is currently not possible to implement all the components in each process area as that would just add to the already heavy laden improvement program. We suggest that more goals and practices be added at later stages as the test process in company X becomes stable and more mature.

TMMi Level 2 Process Areas	Specific Goals	Company X Improved Workflow
2.1 Test Policy and Strategy	SG1: Establish a Test Policy	X
	SG2: Establish a Test Strategy	X
	SG3: Establish Test Performance Indicators	X
2.2 Test Planning	SG1: Perform a Product Risk Assessment	X
	SG2: Establish a Test Approach	X
	SG3: Establish Test Estimates	X
	SG4: Develop a Test Plan	X
	SG4: Obtain Commitment to the Test Plan	X
2.3 Test Monitoring and Control	SG1: Monitor Test Progress against Test Plan	X
	SG2: Monitor Product Quality against Plan and Expectations	X
	SG3: Manage Corrective Action to Closure	--
2.4 Test Design and Execution	SG1: Perform Test Analysis and Design using Test Design Techniques	X
	SG2: Perform Test Implementation	X
	SG3: Perform Test Execution	X
	SG4: Manage Test Incidents to Closure	X
2.5 Test Environment	SG1: Develop Test Environment Requirements	X
	SG2: Perform Test Environment Implementation	X
	SG3: Manage and Control Test Environments	X

Table 3: Assessment of improved workflow against TMMi Level 2

6.3 TMMi Level 3

The process areas for TMMi level 3 is discussed below along with the assessment of the improved worfklow in each process area. For this research we mainly focused on the TMMi level 2 as level 3 was only recently published by the TMMi Foundation.

a.) Test Organisation

This process are requires that an organisation commits to better software testing and higher-quality software. According to the TMMi [2] the Test Organisation's main purpose is to “identify and organize a group of highly skilled people that is responsible for

testing". The specific goals of this process area include the establishment of test functions for test specialists, establishing test career paths, implementing test process improvements, and deploying organisational test processes.

For company X we recommended that management of company X must emphasize the importance of testing and quality software through the test policy. We have also recommended that two dedicated testers be employed. This process area requires that there be a dedicated and independent test group. This is not currently possible in company X due to the limited resources. Based on the limited improvements suggested for company X in terms of this process area, we cannot currently satisfy the specific goals of this process area.

b.) Test Training Program

The main purpose of this process area is to establish a test training program within an organisation that facilitates the development of knowledge and skills to perform effective and efficient testing. Some of the specific goals include the establishment of test training needs, establishment of an organisational test training plan as well as establishing a test training capability. Once these goals have been met, the test training must be delivered, records of training conducted must be kept, and the effectiveness of the training must be assessed.

We have identified the lack of test knowledge and skills as one of the main impedances of the current test process in company X. Therefore we recommended that developers and managers must attain the ISTQB test certifications to provide them with the necessary test knowledge and skills. The TMMi process area requires that internal training be established, but there is currently not enough internal test knowledge in company X to start such a program. We thus recommend that external training such as the ISTQB training be attended before an internal training program can be established. This process area cannot be satisfied with the suggested improved workflow in company X.

c.) Test Life Cycle and Integration

The purpose of test life cycle and integration is to establish a structured test process with a defined life cycle that is integrated and aligned with the software development life cycle. The three specific goals for this process area includes the establishment of organisational test process assets, integration of the test life cycle with the development life cycle, and establishing a master test plan.

We developed and recommended a standard test process for company X with defined life cycle phases. Each life cycle phase consists of clearly defined activities and deliverables. The improved workflow is integrated with the current Waterfall development model of company X and will run alongside this model from early on in the development process. We have also suggested that the test planning phase of the improved workflow create a master test plan as a deliverable.

Based on the recommendations that were made in the improved workflow, we can ascertain that this process area is satisfied.

d.) Non-functional Testing

The purpose of non-functional testing process area is to improve the non-functional testing capability of an organisation. This includes performing a non-functional product risk assessment, establishing a non-functional test approach, analysing and designing non-functional tests cases, and performing the execution of these non-functional test cases.

For the purpose of this research we have decided to not implement any of the specific goals of this process area for company X. Although we suggest that non-functional requirements such as performance, reliability and maintainability be addressed, we recommend that these issues are not formally addressed in the test process as it would over-complicate the process. We suggest that non-functional testing be addressed once company X has fully reached a TMMi level 2 maturity. Based on these assumptions, we cannot satisfy this process area with our improved workflow.

e.) Peer Reviews

The main purpose of the Peer Review process area is to verify that work products meet their specified requirements and to eliminate defects early in the test process. The specific goals of this process area include the establishment of a peer review approach followed by an implementation of this approach.

One of the problem issues identified in the current test process of company X is that requirements specifications are not complete. The requirements specification plays an important role in the current test process of company X as it forms part of the test basis from which tests are created. Therefore we have recommended that all project documentation such as the requirements specification, architecture and design specifications be reviewed by the test team and developers to ensure their correctness. We have also implemented code reviews to ensure that developers keep to company coding standards, as well as to identify good and bad coding practices. Defects can also be identified during code reviews.

Based on the recommendations we have made in the improved workflow regarding reviews, we can determine that this process area is satisfied.

6.4 Summary TMMi Level 3 Assessment

Although the TMMi requires that all process areas of a lower level be satisfied before process areas at higher levels should be implemented, we incorporated two of the five process areas in our improved workflow. The integration of the test process into the software development life cycle is important as more time can be invested into testing. One of the problems identified in company X is that testing is rushed and developers spend too little time on testing activities. Integration of the test process into the software development life cycle addresses these two problem issues. Two other problem issues identified in the current workflow of company X is that the requirements specification is not complete and leads to sub-optimal testing. Lack of code

reviews in the current workflow allowed for poor coding practices to creep into the development process. The suggestion for the implementation of TDD will allow for test cases to be reviewed as they will now be part of the code base. Peer reviews are viewed by us as a critical component of the test process as it eliminates potential defects early in the test process. Reviews also improve the quality of the test basis. By including these process areas of TMMi level 3 in our improved workflow, we have eliminated many of the current problems in company X's test process.

TMMi Level 3 Process Areas	Specific Goals	Company X Improved Workflow
3.1 Test Organisation	SG1: Establish a Test Organisation	--
	SG2: Establish Test Functions for the Test Specialists	--
	SG3: Establish Test Career Paths	--
	SG4: Determine, Plan and Implement Test Process Improvements	--
	SG5: Deploy Organisational Test Processes and Incorporate Lessons Learned	--
3.2 Test Training Program	SG1: Establish an Organisational Test Training Capability	--
	SG2: Provide Necessary Test Training	--
3.3 Test Life Cycle and Integration	SG1: Establish Organisational Test Process Assets	--
	SG2: Integrate the Test Life Cycle with the Development Models	X
	SG3: Establish a Master Test Plan	X
3.4 Non-functional Testing	SG1: Perform a Non-Functional Product Risk Assessment	--
	SG2: Establish a Non-Functional Test Approach	--
	SG3: Perform Non-Functional Test Analysis and Design	--
	SG4: Perform Non-Functional Test Implementation	--
	SG5: Perform Non-Functional Test Execution	--
3.5 Peer Reviews	SG1: Establish a Peer Review Approach	X
	SG2: Perform Peer Reviews	X

Table 4: Assessment of improved workflow against TMMi level 3

6.5 Discussion

In this chapter we assessed the improved workflow against the process areas of the TMMi Levels 2 and 3. We found that the TMMi is very comprehensive and demands that many practices be implemented in each process area before that process area can be satisfied. We have decided that not all these practices and goals in each process area will be beneficial to the suggested workflow of company X at this current moment in time. The main reason for this decision is that the implementation of all these practices and goals is a mammoth task that cannot be handled by the

current resources available in company X. We therefore recommend that an independent team be established in company X to take on this task. This will be necessary should company X decide to submit their test process for official accreditation with the TMMi Foundation. For purposes of this research, we recommend that the suggested workflow be implemented as is to first establish a structured test process. Once the test process has matured and is fully integrated into the activities of the software development division of company X, do we recommend that the outstanding specific goals and practices be implemented.

We have now completed the assessment of the improved workflow, but we have to compare this to the current workflow of company X. Implementing the improved workflow in company X will be a huge undertaking. We have to estimate the effort and resources that will be needed to implement this process in company X. These issues will be addressed in the next chapter.

7 Discussion

In this chapter we will compare *hypothetically* the various test process components that were analysed during this research. First we will compare the old workflow of company X with the newly suggested workflow. Then we will analyse the TMMi maturity level of the old workflow in company X and in comparison with the new workflow. We have now suggested an improved workflow process for company X, but we have to estimate what the effort would be to implement this solution in their organisation. We would like to provide company X with practical guidelines on how to undertake this test process improvement program and make a success of it. For an *empirical* comparison of the performance of company X before and after our suggested improvements, see chapter 8.

7.1 Comparison of Workflow

We created the old test process workflow from results that were gathered by the survey. Developers were asked to model the test process as they performed it. The old workflow had no defined structure to it. Although company X describes rigorous testing as part of their software project proposal documentation, there is no evidence of this implementation. There were no distinct phases with activities specifically defined that make up the whole test process. In the old workflow, "testing" was merely a side-effect of the main purpose: to get the software to run without any observable errors. Therefore we can state that the old workflow in company X was more of a debug process than a test process. The old workflow was very one-dimensional as the entire process was performed by a single person: the developer that developed the product. No test planning is performed upfront and all test activities are conducted on an ad-hoc basis. The old workflow did not produce any data or deliverables and therefore we could not analyse historic test data from previous projects. One of the issues that stood out for us was the lack of testing knowledge by the software developers. It was clear from the results of the survey that most of the developers are not familiar with the fundamental concepts of testing, for example how to create a simple test case. Nevertheless, most developers have received a tertiary education. Defects that are encountered during projects are never tracked in a defect tracking system and therefore propagate into the final product. This leads to extensive rework of products once delivered to the customer. Most of the testing occurred on the desktop or notebook computers of the developers. There is no test environment that closely represents the production environment. The old test process occurred at the very end of the software process. In the case of a project running over schedule, the time initially allocated to testing was usually shortened as it was not deemed critical to the success of project. The result was low quality software fraught with defects that had to be resolved once the product was deployed at the customer. This is not a sustainable software development model.

After performing a rigorous study of the software engineering literature and insightful discussions with industry test practitioners, we devised a new software testing process for company X. The suggested workflow is structured and clearly defines the activities and outcomes of each phase in the test process. As opposed to the old workflow, the new workflow is integrated into the software process. Test planning starts early in the software process alongside the requirements phase. As part of the test planning phase we have incorporated metrics into the process. These metrics will allow company X to track the progress of each test project and provide them with valuable information to make decisions on when to advance to the next test level. The newly suggested workflow includes test levels taken from the V-Model that separate

the different types of testing during the project. These levels include the unit, integration, system, and user acceptance testing levels. We also suggested that company X employ dedicated testers and possibly a test manager. Although this might not be feasible for company X at this stage, this suggestion will be critical to the company's test process in the long run. Defect tracking will also be a critical component to the suggested process and the implementation of a defect tracking system was suggested. As there are no proper implementation of test environments where the integration, system and user acceptance testing is performed, we suggested that a test environment software library be established. This library will store preconfigured virtual machines that can be deployed in a test environment. The use of preconfigured machines with little or no modification necessary will reduce the time necessary to create a test environment from scratch. As part of the test process improvement program, we suggested that developers receive test training to improve their test knowledge. There are at least three ISTQB training providers in South Africa that deliver this service.

The table below compares the components of the test process as they are implemented in the old workflow and the suggested workflow.

Components of Test Process	Old Workflow	Suggested Workflow
Structure	No defined structure. Ad-hoc approach.	Clearly defined with distinct phases.
Test activities	No defined activities.	Each phase has clearly defined activities.
Documentation	No use of standard templates. No review of project documentation.	Makes use of IEEE829. Documentation is reviewed at various phases of test process.
Defect tracking	Little or none informal defect tracking.	Implementation of defect tracking system.
Metrics	No metrics defined or collected.	Defined set of test metrics. Use of metrics database for project history.
Test deliverables	Only software product as result of testing.	Each phase defines deliverables such as test summary report, test plan, test log.
Test cases	No creation of test cases.	Creation of test cases using black-box and white-box test design techniques.
Test policy and strategy	Does not exist.	Suggests creation of test policy and strategy as a critical component of test process.
Test levels	No levels distinguished.	Unit, integration, system, and user acceptance testing levels.

Roles involved in test process	Only developer.	Suggests involvement of management, developers, and dedicated testers.
Test environment	Not properly managed or configured.	Use of test environment software library with preconfigured virtual machines.
Test training	Very limited testing knowledge of developers. No provision made for training.	Suggests rigorous test training and certification with ISTQB for all developers.

Table 5: Comparison of Old Workflow and Improved Workflow

7.2 Comparison of TMMi Maturity Level

In chapter 2 we analysed the old workflow and assessed its TMMi maturity level. We determined that company X would only reach a TMMi level 1 maturity. The description of a TMMi level 1 test process in the TMMi standard epitomises company X. Because the similarities between the standard's description and company X are so disturbingly accurate, we have included some of these statements here: "*At TMMi level 1, testing is a chaotic, undefined process and is often considered as part of debugging. The organisation does not provide a stable environment to support the processes. Tests are developed in an ad-hoc way after coding is completed. The objective of testing at this level is to show that the software runs without major failures. Products are delivered without adequate visibility regarding quality and risks. In the field, the product does often not fulfil its needs, is not stable, or is too slow to work with. Within testing there is a lack of resources, tools and well-educated staff. At TMMi level 1 there are no defined process areas. Also products tend not to be released on time, budgets are overrun and quality is not according to expectations.*" [2]

In terms of TMMi level 2, the old workflow did not satisfy any of the process areas.

We closely followed the recommendations of the TMMi level 2 for the improved workflow. Each process area at level 2 was carefully analysed along with other software engineering literature to determine its importance and feasibility for company X. We decided not to implement all the generic goals and practices for each process are as this would not be feasible for a company with a basically "non-existent" process. There are simply too many requirements by the TMMi for a company that is just starting to establish a test process. For this reason we did not take into account all the specific and generic practices in this assessment. Nevertheless, for the specific goals and practices that we did implement we can safely ascertain that our improved workflow achieved a TMMi level 2 maturity. In order to be officially accredited by the TMMi Foundation on level 2 maturity, we would have to implement the generic goals and practices of each process area of level 2.

7.3 Feasibility and Estimated Effort for Implementation of Improved Workflow

Based on the outcomes of this research we would like to provide company X with practical guidelines on how to implement the improved workflow. This will not be an easy task and [30] suggest that it could take such a test process improvement project up to two years to reach TMMi level 2. Unfortunately company X has no established test practices that can be improved upon or built on and therefore we estimate that this effort will take about two years to complete. [30] suggest the following steps for specifically implementing the TMMi in an organisation: initiation, planning, implementation, and deployment.

i. Initiation

The initiation phase is used to determine the current maturity level of the organisation. Based on the results, recommendations must be made to management. Fortunately this research has already analysed the current test process and has already made recommendations. We therefore suggest that the results of this research be presented to the management of company X. As part of the initiation phase, common goals must be set by the organisation on what they want to achieve with this improvement project. These goals can include predictability, higher productivity, efficiency and effectiveness. [30]

ii. Planning

The planning phases involve the creation of a project team that will be responsible for the improvement project. A common problem seen by [30] is that organisations do not invest enough resources and attention into an improvement project. Each of the recommendations is assigned to a workgroup that implements and deploys the recommendations. We recommend that company X establish a steering committee with a test project improvement manager. Specific time and resources must be allocated to the workgroup so that this project is separate from day-to-day work activities. Proper communication during this project is important as many changes will be introduced. All employees of company X need to know which changes are going to be implemented and why these changes are necessary. [30] recommends creating a periodic newsletter that communicates progress of the project. This will assist with the buy-in from employees into the project. The use of an external consultant to take ownership of the project is recommended by [30] to guide the whole process. We do not currently foresee this recommendation as feasible in company X due to the cost implications. However, the creation of an internal project team to manage this process improvement project will be critical to its success. [30] recommends that the processes which the test process depend on must also be taken into consideration. As the suggested test process is tightly integrated into the software development process, this process must also be changed to accommodate the changes in the test process. We have seen through this research that the software development process will be affected by the suggested test process. Therefore we suggest that a separate improvement project be established for the software development process. This project can be managed by the current software development manager and no additional resources will be needed. We also recommend that the newly suggested test process be documented thoroughly and be made available to all stakeholders in company X.

iii. Implementation

This phase of the improvement project sees the implementation of the recommendations made in this research. We agree with [30] that the test policy and strategy are the first and most important changes to be implemented. This will provide direction on the whole improvement project. We then recommend that each process area of TMMi level 2 and 3 that was satisfied with the suggested test process be implemented. We also recommend that the structure of the recommended documentation be created. All the templates from the IEEE829 should be adapted to the needs of company X and stored in a document management system. A document management system is already in use in company X and therefore this practice should not present any difficulties.

iv. Deployment

According to [30] this part of the improvement project is the most difficult. All the documentation and process guidelines are futile if employees do not adhere to them. We therefore recommend that company X deploy the suggested test process on an internal software project first. This will allow management to carefully monitor the progress of the test project without running the risk of possible failure with a customer-facing project. Only once the basic practices of the test process have been implemented in small steps and the test process is producing the desired outcomes should it be implemented in customer-facing projects. We estimate that this pilot implementation could take between six and eight months.

In this chapter we compared the old workflow with the suggested workflow. We also assessed the maturity levels of the two processes and determined that the suggested process could potentially attain a TMMi level 2 accreditation. The purpose of this research was to answer the main research question: How can we improve the current software testing process for a small company such as Company X? Although we have answered this question through this research, we would like to see the suggested process implemented in company X and observe its practicality. Therefore we also suggested how to approach a process improvement project. By following a carefully managed process with management buy-in, it was estimated that such a project could take up to two years to reach a TMMi level 2 maturity. In chapter eight we will discuss future work.

8 Future Work

This research answered all the questions from the problem statement in chapter 1, but there are certain aspects surrounding this research that were omitted. We did not investigate the test process from the perspective of the management of company X. All the results were based on the findings from the interviews with the eight developers. A recent discussion with the CEO of company X regarding this research proved very positive. The CEO requested that the suggested test process be implemented in company X as soon as possible. This indicates that there is an awareness of the current problems being experienced with the software process, more specifically the test process.

Another perspective that was not taken into account in this research was that of the customer. We did not determine the customer's view of the products they purchased and their perceived quality thereof.

This research only focused on one small company that might not be representative of the test processes of other small software companies. We cannot conclude from this research whether software testing is a general problem facing small software companies.

The author's future work will include the implementation of the suggested test process in company X as discussed in chapter 7. The ideas postulated as part of this research must be empirically evaluated in order to determine the feasibility of the suggested process. Before we embark on such a process, we intend on performing empirical measurements of quantitative parameters including:

- **Developer disgruntlement**

  Developers of company X are frustrated with the current process as there is usually significant rework to be done after delivery of erroneous software. This measurement would attempt to determine how satisfied developers are with the current test process? We shall then measure this variable after the implementation of the process to determine whether the morale and attitude of developers has improved towards the process.

- **Defect rate**

  There is no measurement of the number of defects found in the software. The current defect rate would need to be determined in order for us to know whether there is any improvement after the implementation of the suggested process.

- **Customer satisfaction**

  As mentioned above, the customer perspective was not taken into account with this research. We would need to determine how satisfied customers are with the quality of the current software. Once the suggested process has been implemented, we can measure this variable again to determine the change in customer satisfaction.

Empirical measurement of these factors can provide us with valuable information as to the effectiveness of the suggested process. As many of the components of the suggested process were taken from the TMMi, it could also indicate the suitability of this standard for a small company. Upon completion of the implementation of this process improvement project, we would like to submit the suggested test process to the TMMi Foundation for a formal accreditation and assessment. Obtaining a TMMi accreditation could be beneficial to company X

and South African software companies in general as it indicates a certain level of software quality and software engineering ability.

Acknowledgements

I would like to thank Stefan Gruner from the University of Pretoria for his insight, direction and continuous support throughout this project. Thanks also to the managers and software developers at Company X for their participation in this project.

Bibliography

1. Black R. Critical Testing Processes: Plan, Prepare, Perform, Perfect: Addison-Wesley; 2003.
2. Van Veenendaal E. Test Maturity Model integrated (TMMi); Version 2.0: TMMi Foundation, Available from http://www.tmmifoundation.org/downloads/tmmi/TMMi%20 Framework.pdf; 2009.
3. Gelperin D, Hetzel B. The growth of software testing. Commun ACM, 31(6):687-695, 1988.
4. Kautz K, Hansen HW, Thaysen K, editors. Applying and adjusting a software process improvement model in practice: the use of the IDEAL model in a small software enterprise. ICSE '00: Proceedings of the 22nd international conference on Software engineering, New York, NY, USA: ACM; pp. 626-633; 2000.
5. Von Wangenheim CG, Varkoi T, Salviano CF. Standard based software process assessments in small companies. Software Process Improvement and Practice, 11(3):329-35, 2006.
6. McCaffery F, Taylor PS, Coleman G. Adept: A Unified Assessment Method for Small Software Companies. Software, IEEE, 24(1):24-31, 2007.
7. Cater-Steel A. Process improvement in four small software companies. Proceedings of the 13th Australian Software Engineering Conference (ASWEC); Los Alamitos, CA, USA; IEEE Computer Society; pp. 262–272; 2001.
8. Aranda J, Easterbrook S, Wilson G, editors. Requirements in the wild: How small companies do it; Proceedings of 15th IEEE International Requirements Engineering Conference (RE'07); pp. 39-48; 2007.
9. Batista J, De Figueiredo AD. SPI in a very small team: a case with CMM. Software Process: Improvement and Practice. 2000;5(4):243-50.
10. Von Wangenheim CG, Anacleto A, Salviano CF. Helping small companies assess software processes. Software, IEEE. 2006 Jan.-Feb.;23(1):91-8.
11. Pino FJ, Garcia Felix, Piattini M, editors. Key processes to start software process improvement in small companies. SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing, New York, NY, USA: ACM; pp. 509-516; 2009.
12. Richardson I, Gresse von Wangenheim C. Guest Editors' Introduction: Why are Small Software Organizations Different? Software, IEEE, Jan.-Feb.;24(1):18-22, 2007.
13. Von Wangenheim CG, Weber S, Hauck JCR, Trentin G. Experiences on establishing software processes in small companies. Information and Software Technology, 48(9):890 - 900; 2007.
14. Savolainen P, Sihvonen H, Ahonen JJ. SPI with Lightweight Software Process Modeling in a Small Software Company. Springer Berlin / Heidelberg, pp. 71-81; 2007
15. Grunbacher P, editor. A software assessment process for small software enterprises. EUROMICRO 97 'New Frontiers of Information Technology', Proceedings of the 23rd EUROMICRO Conference; pp. 123-128; 1997.
16. Broy M, Rombach D. Software Engineering: Wurzeln, Stand und Perspektiven. Informatik Spektrum, December 16; pp. 438-451; 2006.
17. Royce WW. Managing the development of large software systems: concepts and techniques. ICSE '87: Proceedings of the 9th international conference on Software Engineering, Los Alamitos, CA, USA: IEEE Computer Society Press; pp. 328-338; 1987.
18. Pyhäjärvi M, Rautiainen K, Itkonen J, editors. Increasing understanding of the modern testing

perspective in software product development projects. Proceedings of the 36th IEEE International Conference on System Sciences; pp. 250-259; 2002.

19. A Test Process for all Lifecycles; Technical Paper; Available online from http://www. applabs.com/uploads/app_whitepaper_test_process_for_all_lifecycles_1v011.pdf; 2008
20. Olivier MS. Information Technology Research: A practical guide for Computer Science and Informatics. Second Edition ed. Pretoria: Van Schaik Publishers, 2004.
21. Hofstee E. Constructing a Good Dissertation: A Practical Guide to Finish a Masters, MBA or PHD on Schedule; Johannesburg: EPE, 2006.
22. SPEM. Software Process Engineering Metamodel version 2.0; Object Management Group; Available online from http://www.omg.org/cgi-bin/doc?formal/08-04-01.pdf; 2008.
23. Juristo N, Moreno AM, Strigel W. Software Testing Practices in Industry. IEEE Software; pp. 19-21; 2006.
24. ISTQB Standard Glossary of Terms used in Software Testing: International Software Qualifications Board; Report No. v2.0; Available from www.istqb.org/downloads/glossary-1.1.pdf; 2007.
25. Burnstein I, Suwanassart T, Carlson R, editors. Developing a Testing Maturity Model for software test process evaluation and improvement. International Test Conference; pp. 581-589; 1996.
26. Van Veenendaal E, Pol M. A Test Management Approach for Structured Testing. Achieving Software Product Quality; Online collection without page numbers; 1997.
27. Koomen T, Pol M. Test Process Improvement, A practical step-by-step guide to structured testing: Addison-Wesley, 1999.
28. Ericson T, Subotic A, Ursing S. TIM - A Test Improvement Model. Software Verification and Reliability, 7(4):229-246, 1998.
29. Myers G. The Art of Software Testing. New York: John Wiley & Sons, Inc., 2004.
30. Van Veenendaal E, Hendriks R. Van de Laar J., Bouwers B. Test Process Improvement using TMMi. Testing Experience: The Magazine for Professional Testers, 3:21-25, 2008.
31. IEEE Standard for Software and System Test Documentation. IEEE Std 829; Available online from http://0ieeexplore.ieee.org.innopac.up.ac.za/stamp/stamp.jsp?tp=&arnumber= 4578383&isnumber=4578382; 2008.
32. British Computer Society Specialist Interest Group in Software Testing. Software Component Testing Standard; BS7925-2; Available from http://www.testingstandards.co.uk /Component%20Testing.pdf 1998.
33. Shepard T, Lamb M, Kelly D. More testing should be taught. Commun ACM, 44(6):103--8, 2001.
34. Edwards SH. Improving student performance by evaluating how well students test their own programs. J Educ Resour Comput, 3(3):1, 2003.
35. Glass RL, Collard R, Bertolino A, Bach J, Kaner C. Software testing and industry needs. Software, IEEE, 23(4):55-7, 2006.
36. Ng SP, Murnane T, Reed K, Grant D, Chen TY. A preliminary survey on software testing practices in Australia. Proceedings of the 2004 Australian Software Engineering Conference; pp. 116-125; 2004.
37. Geras AM, Smith MR, Miller J. A survey of software testing practices in alberta. Canadian Journal of Electrical and Computer Engineering, 29(3):183-91, 2004.

38. Vos TEJ, Sánches J.S., Mannise M. Size does matter in process improvement. Testing Experience: The Magazine for Professional Testers, 3:91-5, 2008.

39. Black R. ISTQB Certification: Why You Need It and How to Get It. Testing Experience: The Magazine for Professional Testers, 1:26-30, 2008.

40. IEEE, IEEE Computer Society, 2009, Available from: http://www.computer.org, Accessed 13 October 2009.

41. IEEE standard glossary of software engineering terminology. IEEE Std 610.12-1999; Available online from http://0-ieeexplore.ieee.org.innopac.up.ac.za/stamp/stamp.jsp?tp=&arnumber=159342&isnumber=4148; 1990.

42. Kajko-Mattsson M, Björnsson T, editors. Outlining developers' testing process model; 33rd EUROMICRO Conference on 28-31 Aug 2007; pp. 263-270; 2007.

43. Mogyorodi GE. Let's Play Twenty Questions: Tell me about your organization's quality assurance and testing. Crosstalk: The Journal of Defense Software Engineering; Available online from http://www.stsc.hill.af.mil/crosstalk/2003/03/mogyorodi1.html; Without page numbers; 2003.

44. Bertolino A. Software Testing Research: Achievements, Challenges, Dreams. FOSE '07, Future of Software Engineering, Washington, DC, USA: IEEE Computer Society; pp.85-103 2007.

45. IEEE standard for software unit testing. ANSI/IEEE Std 1008-1987; Available online from http://0-ieeexplore.ieee.org.innopac.up.ac.za/stamp/stamp.jsp?tp=&arnumber=27763&is number=1092; 1987.

46. Farooq A, Dumke RR. Evaluation Approaches in Software Testing; Technical Report, FIN-05-2008, Faculty of Computer Science, University of Magdeburg; Available online from http://www.cs.uni-magdeburg.de/fin_media/downloads/forschung/preprints/2008/ TechReport5.pdf; 2008.

47. Lewis WE. Software Testing and Continuous Quality Improvement. Second Edition ed: Auerbach Publications, 2004.

48. Park J, Ryu H, Choi H-J, Ryu D-K. A survey on software test maturity in Korean defense industry. ISEC '08: Proceedings of the 1st conference on India software engineering conference, New York, NY, USA: ACM; pp. 149-150; 2008.

49. García J, De Amescua A, Velasco M, Sanz A. Ten factors that impede improvement of verification and validation processes in software intensive organizations. Software Process Improvement and Practice, 13(4):335-43, 2008.

50. Maes D, Mertens S. 10 tips for successful testing! Testing Experience: The Magazine for Professional Testers, 3:52-3, 2008.

51. Karlström D, Runeson P, Nordén S. A minimal test practice framework for emerging software organizations. Software Testing Verification and Reliability, 15(3):145-66, 2005.

52. Lazic L, Mastorakis N. Cost effective software test metrics. WSEAS Transactions on Computers, 7(6):599-619, 2008.

53. Mallinson W. Testing: A Growing Opportunity. Test Focus; Available online from http://www.testfocus.co.za/featurearticles/Jan2002.htm; No page numbers available; 2002.

54. Graham D, Van Veenedaal E, Evans I, Black R. Foundations of Software Testing: ISTQB Certification: International Thomson Business Press, 2008.

55. Pinkster-O'Riordain I. Test Policy: Gaining Control on IT Quality and Processes. 2008 IEEE International Conference on Software Testing Verification and Validation Workshop, ICSTW'08,

art. no. 4567030, pp. 338-342, 2008

56. Martin K, Hoffman B. An Open Source Approach to Developing Software in a Small Organization. Software, IEEE. 24(1):46-53, 2007

57. Otoya S, Cerpa N. An experience: a small software company attempting to improve its process. Software Technology and Engineering Practice Conference, STEP '99; pp. 153-160; 1999.

58. Grindal M, Offutt J, Mellin J. On the Testing Maturity of Software Producing Organizations. Testing: Academic and Industrial Conference - Practice And Research Techniques, TAIC PART Proceedings; pp.171-180; 2006.

59. Brooks F. The Mythical Man-Month. Reading, MA: Addison-Wesley. 1975.

60. Blokland K. A universal management and monitoring process for testing. 2008 IEEE International Conference on Software Testing Verification and Validation Workshop, ICSTW'08, art. no. 4567025, pp. 315-321; 2008.

61. George B, Williams L. An initial investigation of test driven development in industry. SAC '03: Proceedings of the 2003 ACM symposium on Applied computing. New York, NY, USA: ACM; pp. 1135-1139; 2003.

62. Fayad ME, Laitinen M, Ward RP. Thinking objectively: software engineering in the small. Commun ACM, 43(3):115-118, 2000.

63. Telea A, Byelas H. Querying Large C and C++ Code Bases: The Open Approach. in Colloquium and Festschrift at the occasion of the 60th birthday of Derrick Kourie (Computer Science), Windy Brow, South Afica, 28 June 2008. Available online from http://www.cs.up.ac.za/cs/sgruner/Festschrift/; 2008

64. Zheng J, Williams L, Nagappan N, Snipes W, Hudepohl JP, Vouk AM. On the Value of Static Analysis for Fault Detection in Software. IEEE Transactions on Software Engeineering. 32(4):240-253, 2006.

65. Watson AH, McCable TJ. Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric. Gaithersburg, MD: National Institute of Standards and Technology; NIST Special Publication 500-235; Available online from http://www.mccabe.com/pdf/nist235r.pdf; 1996.

66. Craig RD, Jaskiel SP. Systematic Software Testing. Boston: Artech House Publishers, 2002.

67. Amman P, Offutt J. Introduction to Software Testing. Cambridge: Cambridge University Press, 2008.