Software testing in a small company

by Johan van Zyl

Let me begin by setting the record straight: I am not a tester and have never received any systematic training on testing. Thus, you might be asking yourself: how and why has this guy conducted a project on software testing? In this article, I will attempt to explain how I came about software testing, and I will make some interesting revelations about software testing in small companies, not only in South Africa, but around the world.

As part of my master's degree in information technology (M.IT), I had to do a research project. My fellow students and I were given free reins to choose any topic that interested us. I found this to be more difficult than choosing from a predetermined list of topics, but I was determined to research something that was relevant to my daily work.

In his book *Outliers – The Story of Success*, Malcolm Gladwell[1] states that it takes about 10 000 hours of practice to become an expert in something. Apparently, even Mozart's first works were not that great, and it was only after he reached this magical number of hours that his true genius came to light. I work for a small consulting and development company in South Africa, which focuses on integration and custom software development. I shall call it Company X.

One thing that always bothered me about Company X was the poor quality of the software that was produced. It seemed to me as if customers were never really satisfied with the product they received. You might think that many other factors, apart from testing, influenced that, and you would be right. The company uses the Waterfall Model because the management has not bought into the whole idea of agile development methods. Projects are late, over budget and of a questionable quality.

This led me and my project supervisor to believe that we could make a positive change by assessing the current testing process, or rather the lack of a suitable process, and then recommending an improved software testing process on the basis of these findings. My aim was to help Company X to establish a reasonable testing process based on well-known "best practices" and principles from scientific software engineering literature.

1 Gladwell, M. 2008. *Outliers – The Story of Success*. Little Brown and Company.

Small software companies (SSCs) with between one and 50 employees face many challenges in their pursuit of creating quality software. The first challenge is that they do not believe that the same processes and methods used by large software companies are applicable to them. Possible reasons for not adopting these methods are cost, lack of resources, time and complexity. The second challenge facing small software companies is that their processes and organisational structures are informal and often lead to a chaotic environment. This can be attributed to the fact that small companies must focus on time-to-market to survive and often neglect the more resource-intensive formal processes. The third challenge is a lack of skills and experience. Small software companies can often not afford to employ experienced software developers. The fourth challenge is that despite the many software process improvement (SPI) programs out there, such as CMM, CMMI and ISO/IEC15504, small software companies are either not aware of them or the software engineering practices implemented by these SPI programs are not focused on the needs and problems facing small companies.

You might think that these challenges are faced only by the local software market, but the literature suggests otherwise. A study conducted on the German software industry indicated that only 5% of all German software houses had a CMM maturity level of two or higher[2]. The other 95% had a CMM level of one.

My first goal was to get to know the testing process better. I wanted to understand how the software developers felt about testing, what they knew about software testing, and how the company perceived software

2 Broy, M. & Rombach, D. 2002. Software Engineering – Wurzeln, Stand und Perspektiven. *Informatik Spektrum*, Vol. 25, No.6, pp. 438-451, Springer Verlag.

testing. The answers to these questions were provided by way of a questionnaire, which formed the basis for further investigation. Once that was accomplished, my second goal was to use the abundance of software engineering literature available and provide a basic testing process[3].

Concrete problems in Company X

The first step in reaching these goals was to determine the problem areas in and around the testing process at Company X. These problems are by no means a South African phenomenon, but can be seen in many small software companies around the world.

The first problem was the lack of software testing knowledge. All the developers employed by Company X are university graduates. These developers were all employed without industry experience. I found that 75% of the developers did not know how to write a unit test. Software testing is currently not an individual module in the computer science or informatics curricula at many universities or technical colleges. Moreover, the management of the company did not provide the developers with an internal test training programme or external training opportunities. This led to projects that served as a training ground for these developers to build their skills. In turn, this resulted in missed deadlines and software fraught with defects. This is not a sustainable business model, as frustrated and unsatisfied customers will not return for future products. Furthermore, developers did not use any testing tools or frameworks such as Xunit, mainly because they did not know enough about testing and its importance in creating software of a high quality.

The last problem identified with regard to the knowledge and skills of the software developers was that they only performed dynamic testing[4], that is, testing that involves executable code. As is now known, static testing is just as important to detect defects early in the software process. This can be achieved by having regular code reviews, as well as reviewing the quality of requirements and design specifications.

The second part of my questionnaire was meant to yield some insight into the perceived commitment of the company towards testing. I wanted to determine how serious Company X was about providing the quality they promised in their mission statement. Here I quickly ascertained that the developers did not know what was expected of them in terms of testing. Typically, a company-wide testing policy will provide the employees with high-level testing objectives and principles that can be expanded further in a testing strategy per project. The testing strategy typically outlines the different testing levels, such as unit, integration, system and user-acceptance testing, as well as the activities to be performed at the different levels. There was also a lack of test documentation standards or templates that could be used during projects, such as testing plans.

Another problem that was identified was the lack of proper testing environments. Limited hardware was available and developers mainly tested the programs on their notebooks using virtual machines.

The main issue that this presented was that the production environment could never be properly replicated in a testing environment. This could lead to nasty surprises when it came to deployment during the project.

The current testing process in Company X is depicted in Figure 1. I used the software process engineering metamodel (SPEM) notation[5]. The workflow depicted in Figure 1 begins with the requirement specifications. Its work product serves as input into the implementation of features. A developer would take a requirement and implement it. Once it has been implemented, the developer would run the code to verify that the requirement is met. Should a defect exist in the code, the developer would attempt to fix the defect and run the code again. This code-and-fix approach would continue until all the features had been implemented and no apparent defects existed. The code would be packaged and released to the customer, who would often perform the most rigorous testing.

Test process improvement models

The test maturity model integration (TMMi) and the critical testing processes (CTP) are test process improvement models selected as the baseline for the newly suggested test process. The TMMi Foundation[6], a non-profit organisation created by test industry experts, developed the TMMi. The reason for choosing the TMMi is that it is fast gaining recognition as an industry standard, as was set out by its developers. It is based on the CMMi and the test maturity model (TMM). The CTP was developed by Rex Black[7] and was born out of 20 years' experience in the software testing industry. The CTP model contains 12 so-called "critical processes" that can be applied in any software organisation. The processes are presented as "lightweight checklists" and not as bureaucratic regulations.

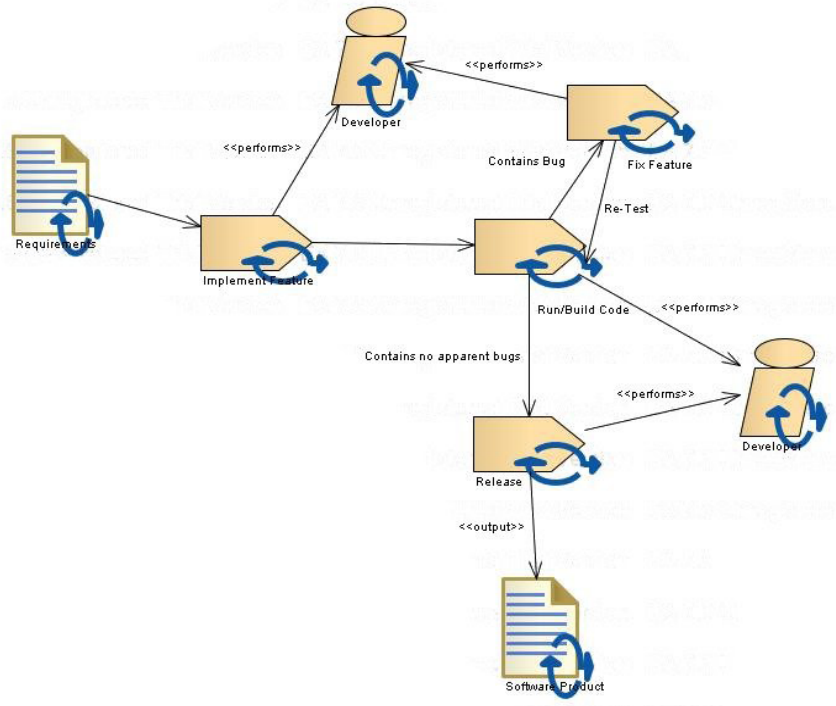
3 Van Zyl, J.M. 2010. Software Testing in a Small Company. Technical Report, University of Pretoria. http://ssfm.cs.up.ac.za/TR-JvZ-2010.pdf

4 ISTQB Standard Glossary of Terms used in Software Testing: International Software Qualifications. Board Report, v2.0, 2007, http://www.istqb.org/downloads/glossary-1.1.pdf

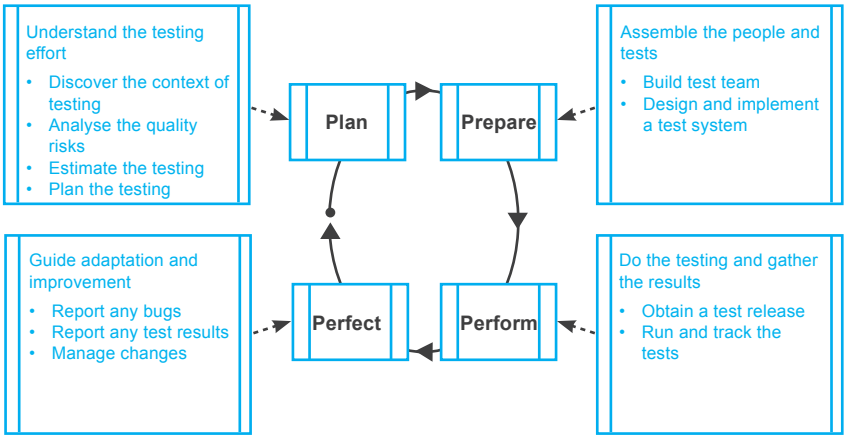
5 SPEM: Software Process Engineering Metamodel, volume 2; Object Management Group, 2008, http://www.omg.org/cgi-bin/doc?formal/08-04-01.pdf

6 TMMi Foundation: Test Maturity Model Integration (TMMi), Version 2, http://www.tmmifoundation.org/html/tmmiref.html

7 Black, R. 2004. *Critical testing processes – plan, prepare, perform, perfect*. Addison Wesley Longman Publ.



1. The current testing process in Company X.



2. Rex Black's critical testing processes.

Critical testing processes

The CTP comprises four phases. The first phase is concerned with planning the test effort. This includes a quality risk analysis, estimating the test effort and developing a plan for the test project. The deliverable of this phase is a test plan. The Prepare Phase commences upon completion of the Plan Phase. This phase includes the building of a test team, which either means employing new testers or training existing employees. According to Black, "only a team composed of qualified test professionals can succeed with the critical testing processes."

As you may have noticed, Company X does not employ a single dedicated tester. All test-related activities are performed by the software developers. The Prepare Phase is also concerned with building a test system that includes the creation of test suites to cover the risks identified in the Plan Phase, selecting the appropriate testing techniques based on the test conditions, and creating the test environment. The third phase of the CTP is the Perform Phase. The purpose of this phase is to execute the test cases developed and to record the test results. The final phase of the CTP is the Perfect Phase. The purpose of this phase is to adapt and improve the test process. This phase includes the bug reporting process, communication of the test results to relevant stakeholders, and the refinement of the applicable test process itself.

Test maturity model integration

The TMMi is a "heavyweight" model (compared to the more agile or "lightweight" CTP in which processes can be implemented as needed). It is structured more strongly and is very comprehensive. It follows the same staged approach as its CMMi uncle.

There are five levels of maturity, each consisting of different process areas. These process areas are made up of specific and generic goals that consist of specific and generic practices. The practices in each process area must be implemented before the maturity at that level can be attained. I only focused on TMMi levels two and three, as it can take up to two years to reach a level two maturity from the lowest level one.

These two levels include process areas such as test planning, test policy and strategy, test monitoring and control, test design and execution, and test organisation. Clear guidelines are provided on what is expected in each process area. Company X is currently at a TMMi level one. At this level, the test process is chaotic and undefined. There is no difference between testing and debugging.

Testing is mostly performed in an ad hoc manner after coding has been completed. I was shocked to see the similarities between the TMMi's description of a level one company and Company X. My goal was to elevate it to at least a level two maturity.

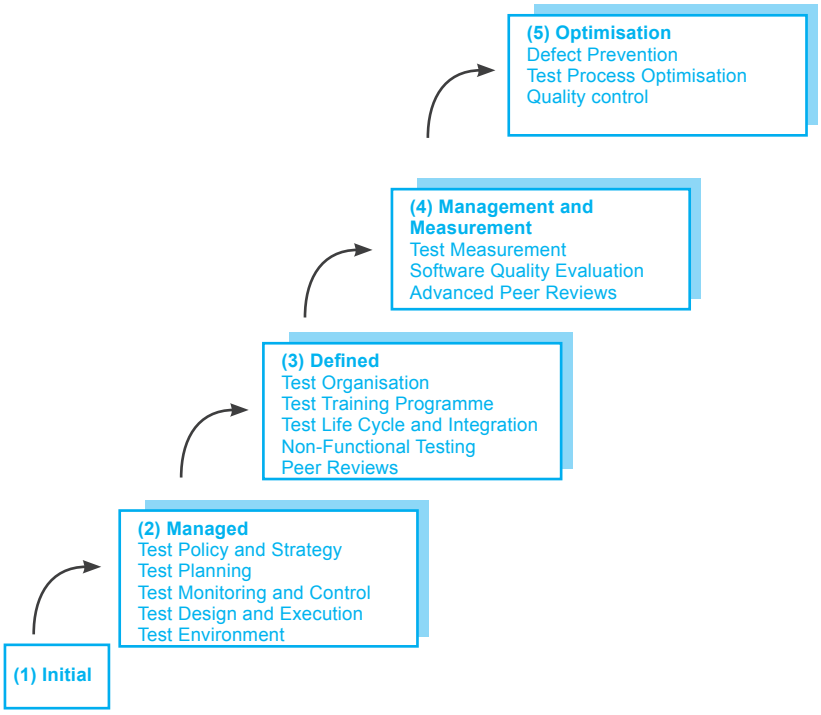
Based on the suggestions found in the literature and fruitful discussions with test industry experts, I devised a basic test process for Company X. This high-level overview of my suggestions improved the test process.

Improved test process

As my objective was to improve only the test process and not the entire software development process, I opted to incorporate the V-model[8] into the new test process for Company X. The V-model of software testing suits the Waterfall Model of software development and can be integrated without disrupting the core of the workflow of the entire development process. The V-model also allows for the early integration of the test activities into the software development process. The likeness of the new test process for Company X to the abstract V-model can be seen in Figure 4.

This diagram is also shown in SPEM notation and deserves further explanation. The icons that resemble envelopes represent the deliverables at different stages of the process. The icon that resembles a paper document at the top left is the test strategy, which serves as input to the test process. The left leg of the V consists of the software development phases, coupled with the process areas from the TMMi. It is clear that the Test Planning Phase follows the Requirement Phase and that a master test plan is produced as a deliverable of this phase.

8 Forsberg, K. & Mooz, H. 1998. System Engineering for Faster, Cheaper, Better. Technical Report, Center of Systems Management. Available online via Citeseer and Google Docs.



3. Rex Black's TMMi maturity levels.

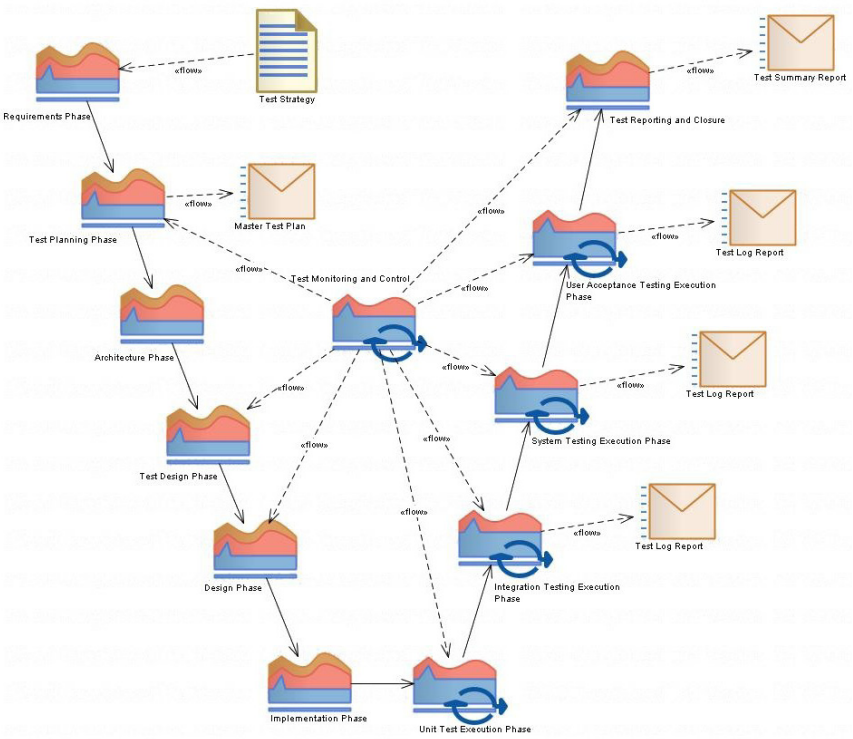
Test design is performed after the Architecture Phase. The whole process is continuously monitored and controlled, as can be seen from the icon in the centre of the process model. This is the monitoring and control process area adapted from the TMMi recommendations. The aim of this process area is to measure the test process with the use of metrics such as code coverage or the number of open defects. The reviewing of documentation and program code also forms part of this phase and helps to ensure that quality is achieved at the outset of the project.

The test execution phases (also adapted from TMMi) follow the Implementation Phase of the software development process. This includes the normal testing phases from the V-model (unit, integration, system and user-acceptance testing). Each phase is concluded with the creation of a test log to decide whether the entry and exit criteria of the respective levels have been met, and to track the progress of the entire test process.

The process is concluded with the test reporting and closure process area, where a test summary report is distributed to all stakeholders, the test environment is backed up for possible future use, and a test project "post-mortem" is conducted to determine where the process diverged from the test plan and how to improve the test process during the next project.

Looking back at this research project, I would have liked to have had a more agile process. This was not possible due to the current Waterfall process being followed in Company X. I can happily report that an agile approach is now being implemented, which will emphasise test-driven development (TDD). However, a recent paper[9] published on the benefits of TDD reported some remarkable results. It was found that TDD improved the internal code quality in some cases, but increased unwanted code complexity in others.

9 Siniaalto, M. & Abrahamsson, P. (2008). Does Test-Driven Development improve the Program Code? Alarming Results from a Comparative Case Study. Lecture Notes in Computer Science, Vol. 5082, pp. 143-156, Springer Verlag.



→ *4. Improved test process for Company X.*

The benefits of TDD were not as self-evident as previously expected and the ultimate verdict is still out.
I believe that there must be stronger collaboration between industry and tertiary institutions to deliver the right skills needed to the local IT market.
It came as a surprise to find that small companies play a big role in the global software economy, but many of them suffer from problems similar to those experienced by Company X.

Acknowledgements

This contribution is a slightly modified reprint of an article that appeared in the *Test Focus*, edited by Wayne Mallinson. Written permission was kindly given by the magazine, which is gratefully acknowledged.

I would also like to thank the people from the Special Interest Group in Software Testing (SIGiST) of the Computer Society of South Africa (CSSA) and the Johannesburg Centre for Software Engineering (JCSE) for the opportunity to present my findings at one of their public seminars in Johannesburg.A further thank you to my supervisor at the University of Pretoria, Stefan Gruner, for his support during my M.IT dissertation project.

Johan van Zyl *is a software development manager at a small company in Pretoria. He loves the mental challenge that programming holds and dreams to be a software architect when he grows up. He relaxes by riding mountain bikes over weekends and has a rather expensive fixation with collecting books on software engineering.*
Last, but not least, he is enrolled for PhD studies in computer science (software engineering) at the University of Pretoria.