Abstract

Testing takes a large share of software development efforts, and is hence of interest when seeking improvements. Several test process improvement frameworks exist, but they are extensive and much too large for smaller organisations to be effective. This paper presents a minimal test practice framework (MTPF) that allows the incremental introduction of appropriate practices at the appropriate time in rapidly expanding organisations. The process for introducing the practice framework tries to minimise resistance to change by maximising the involvement of the entire organisation in the improvement effort and ensuring that changes are made in small steps with a low threshold for each step. The created practice framework and introduction method are evaluated at one company by applying the framework for a one-year period. 12 local software development companies also evaluate the produced framework in a survey.

Keywords: Software testing, Process improvement, Small organisations, Test practice framework

1 Introduction

Testing is an activity in software development known to consume a lot of effort. Hence it is also a prime candidate for change in order to improve the effectiveness and efficiency of the software development. There are many improvement frameworks, specifically focused on testing, for example the Testability Maturity Model (TMM) [1], the Test Improvement Model (TIM) [2] and the Test Process Improvement model (TPI) [3]. All of these suffer from being too extensive for small and medium-sized enterprises (SME) where the resources needed are simply not available. In particular taking the first improvement step is a huge effort for an SME. This is especially important in the light of the fact that SMEs tend not to consider documented processes as a key asset, but rather rely on the knowledge and skills of their staff [4].
In order to meet the needs of such companies, this paper proposes a structured framework of test practices and an introduction method, taking a minimalist approach. The context is an emerging software development organisation, which is expected to grow significantly in the near future and the current practice is of an ad-hoc nature [4]. The approach focuses on following the growth of the company while minimising the resources spent introducing the practices. Another concern that is treated in the approach is the need for continuous adaptation of the practices in order for them to stay current, effective and useful. The approach also attempts to reduce the resistance to change and increase involvement of the entire organisation by frequent feedback and adaptation opportunities.

The method used to create this framework is based on observational qualitative research performed at a small and emerging software company, Scalado in Lund, Sweden. Empirical research is accepted and promoted within software engineering. It is considered that a combination of qualitative and quantitative approaches is the most beneficial as it is very difficult to isolate the observed phenomena from their contexts to achieve control over variables. For this kind of research, aiming at observing a change at a small company, flexible, deductive design studies are most suitable [5]. By starting with a blank sheet of paper, the test process needs in a small and growing software company are investigated. Required practices are documented in a minimal test practice framework (MTPF). Other considerations taken in the creation of the practice framework are avoiding negative effects of viewpoints and process distortion [6,7,8] by involving the whole organisation in the test practices and their improvement.

The framework is evaluated by actual use within the same company and observations noted during the first year of use. Furthermore, the transferability of the framework is evaluated in a survey among twelve software companies based in the region. It is concluded that the framework is useful to small and emerging companies as a first step in their test process improvement.

The paper is structured as follows. In Section 2, other work on test improvement models and frameworks is presented and related to the minimal test practice framework. Section 3 presents the framework, along with the introduction method. Section 4 presents the procedures applied to derive the framework. In Section 5, the validation through use of the framework is presented, and in Section 6, the survey validation is presented. An overall validity analysis is provided in Section 7 and finally, Section 8 summarises the work and outlines further work.
2 Related Work

Testing in software companies is by no means an unexplored area in software engineering research. This section presents some relevant related work to the work presented in this paper.

2.1 Test Practice Frameworks

The Capability Maturity Model (CMM) [9] has been one of the most successful and influential software process improvement models to date. The areas of CMM mostly related to testing are found in the *Software Product Engineering* Key Process Area (KPA). Further testing practices are found in the *Training Program, Technology Change Management* and *Process Change Management* KPA:s. Common for the testing practices in the CMM is a high level of formalism, making them difficult to introduce in a small organisation. The CMM states that testing should be performed on four levels, unit, integration, system, and acceptance tests. Regression tests are required to verify changes. The test plan should be reviewed before completion and test process standards should be organised together with the software process standards. Testers require proper training and a separate system test group is responsible for performing independent testing. Test processes need to be continuously improved. Hence the CMM is on a more strategic level, and there is a need for more practice-oriented support. To facilitate the transition from ad-hoc to a more mature approach, the MTPF is designed to comply with the CMM KPA:s, if the organisation decides to take a CMM improvement approach later on when this is appropriate [10].

The Testability Maturity Model (TMM) [1] is modelled after the CMM and has six Key Support Areas (KSA). These are: *test friendly infrastructure, test-aware project planning, test-friendly product information, test-aware software design, testware and test environment design*. The TMM however has only two improvement steps from the initial level in contrast to the CMM’s four. This makes the levels difficult to reach. The small steps of the MTPF proposed in this paper are intended to provide an easier path to follow.

The Test Improvement Model (TIM) [2] incorporates the notion of levels from the CMM and TMM and it is composed of five key areas: *Organisation, Planning and Tracking, Test Cases, Testware, and Reviews*. These areas are very similar to the areas in the framework proposed in this paper. A key difference, however, is that the effort needed for the initial introduction of the first phase is much higher in the TIM case, compared to the MTPF low-effort approach.

The Test Process Improvement model (TPI) [3] is composed of 20 Key Areas classified into levels of maturity in a test maturity matrix. To ensure correct classification of the maturity level of a Key Area a number of checkpoints are
assigned to each level. In addition a number of improvement suggestions are provided for each level. The model has evolved largely from knowledge and experience gained in the field of administrative automation. The sheer size and complexity of the TPI model makes introduction into a small organisation very difficult. Just identifying appropriate parts of the model and extracting and adapting these to a small organisation would take more resources than a small organisation is willing to spend on test process improvement. The completeness of the model however implies that most of the practices identified as important in the MTPF presented in this paper are included in the TPI model.

2.2 Introduction Methods

In order to effectively introduce a test practice framework there is a need for a suitable introduction method. This strategy of separating practices and introduction method can be found in CMM [9], where the IDEAL improvement process defines the introduction steps [11]. Plan Do Check Act (PDCA) or Deming circles [12] is another well-known approach which aims at the same goal. TPI uses a similar approach.

An introduction method has been developed, supporting the MTPF. To ensure that the practices are suitable for use within the targeted context, the introduction method includes a pre-analysis step aiming to adapt the general knowledge from the framework to the specialised situation at the company.

2.3 Research methodology

It is well accepted in software engineering research to use empirical methods, although the practical use of empirical methods still is limited [13]. Empirical research methods comprise experiments, case studies and surveys. The former are quite well described and supported by guidelines [14, 15, 16], while the latter still rely on guidelines from social science research [5, 17], except for quantitative approaches [16, 18].

Quantitative studies or fixed design studies, e.g. experiments, are suited for use in assessing technical issues, like comparison of two object model representations or two inspection methods easily extracted from their context in the software organisation and controlled in an experimental environment. When extending the study scope to comprise a complete project or an organisation, qualitative studies or flexible design studies are better suited to capture the wide variety of aspects involved in such a situation. By their critics, qualitative studies are considered less scientific than quantitative ones [5]. The design and analysis methodology is less precise in nature, but applying a well structured and rigorously reported qualitative design methodology, the study may capture aspects of the studied situation that
cannot be captured in figures.

In this study, an approach to participant observation in identifying improvement needs in a small development organisation is applied initially. Qualitative analysis is used to prioritise the needs. The outcome is documented in terms of a draft framework. Secondly, the draft framework is validated in two steps, using semi-structured interviews within the studied company. Thirdly, the framework and its transferability to other contexts is validated in a survey comprising a set of other companies.

3 The Minimal Test Practice Framework

This section presents the Minimal Test Practice Framework in its final version, validated both by introduction of phase 1 at Scalado and by the survey of local software companies.

3.1 Overview

The MTPF defines the kind of practices that are needed in small and emerging software companies. It is structured in five categories, and is levelled in three phases.

The five categories correspond to areas in testing and test organisation:

- *Problem and experience reporting*. The systematic reporting and tracking of defects found in the product and experiences gained in the projects.
- *Roles and organisation issues*. The organisational structure of test functions and the roles determined within the organisation for test purposes.
- *Verification and validation*. Activities conducted to verify and validate the product.
- *Test Planning*. Planning of the tests.
- *Test Administration*. Administration of the testing environment.

The categories are derived from the observed needs of the small organisation studied and are validated by actual application within the organisation as well as in a survey in several different companies. The details of how the categories are derived are presented in Section 4, and their validation is presented in Sections 6 and 7.

The phases correspond to the growth of the company. The first phase includes practices that are suitable for a company with approximately 10 employees in development. The second phase includes practices that are suitable for
A company with approximately 20 employees in development. The third phase includes practices that are suitable for a company with 30+ employees in development. The idea is that as the organisation grows the new practices solve the new issues created in the new, larger organisation. It is possible that a fourth phase might be appropriate for extremely large development teams but it is believed that such a step preferably involves a more rigid framework such as TPI [3]. The number of people involved in development described in the framework is intended as a guide to using the framework, not to dictate the exact point of transition from one phase to another. This decision must be left to the company itself as it is based on too many variables to be defined in a framework of this kind.

The framework is summarized in Table 1. The table shows the categories from left to right and the phases from bottom to top. In this section an extensive description of each area is provided.

<table>
<thead>
<tr>
<th>Category</th>
<th>Phase 1 (~10)</th>
<th>Phase 2 (~20)</th>
<th>Phase 3 (30+)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem &amp; Experience Reporting</td>
<td>Define Syntax</td>
<td>Create System</td>
<td>Maintain &amp; Evolve System</td>
</tr>
<tr>
<td>Roles &amp; Organisation</td>
<td>Define Roles</td>
<td>Define Teams</td>
<td>Define Teams</td>
</tr>
<tr>
<td>Verification &amp; Validation</td>
<td>Use Checklists</td>
<td>Perform Inspections</td>
<td>Perform Inspections</td>
</tr>
<tr>
<td>Test Administration</td>
<td>Basic Administartion of Test Environment</td>
<td>Test Cases</td>
<td>Risk Management</td>
</tr>
<tr>
<td>Test Planning</td>
<td>Test Plan</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Overview of MTPF

Along with the practice framework, an introduction method has been derived, which is inspired by the IDEAL process. A significant difference is that the method proposed here enables a feedback and revision step before actual introduction of practices in order to increase the involvement of the practitioners.

The introduction method consists of five steps, prepare phase, introduce phase, revise phase, perform, and evaluate. These steps are illustrated in the flowchart shown in Figure 1. In the figure the dashed rectangle indicates the normal operating mode of the organisation. Here the practices are continuously evaluated (1) and revised if needed (2). As soon as the need for a new phase is identified (3) a phase preparation step is initiated. In this phase everything required for the introduction of the coming phase is prepared. This includes training material, database systems and...
other preparations for introducing the phase practices.

The prepared practices are presented to the employees in the second step. This is done so that feedback is easily given from all parts of the organisation, for instance in a seminar form. Before the actual performing of the new practice employee feedback is used to revise the practices, ensuring that the practices are really tailored to the context in which they will be used.

![Figure 1 Overview of Introduction Method](image)

### 3.2 Test Practices

In this section each test category is presented phase-by-phase and described in detail.

#### 3.2.1 Problem and Experience Reporting

*Phase 1 - Define Syntax.* A common problem recording syntax is introduced in the company. This will provide developers and testers with a common language to describe and record problems in the product.

*Phase 2 - Create System.* A problem reporting and storage system is introduced. This system should use the common syntax created during phase 1. The syntax should have stabilised by now and be accepted throughout the company. Routines are established for gathering, documenting, storing and using experiences gathered during the course of each project. This could be done as an extension of the problem reporting system created in this phase.

*Phase 3 – Maintain and Evolve System.* It is ensured that the system from phase 2 can handle recording defects in the new state of the company.

#### 3.2.2 Roles and Organisation

*Phase 1 – Define Responsibility.* Testing responsibility is made clear within the company. The responsibilities include: developing a test plan for each new project, administrate the test environment, administrate the problem
reporting, update checklists, continuously assess the testing practices and monitor the need for the next phase.

**Phase 2 – Define Roles.** A test manager is appointed and the developer roles are assigned. Several people at the company now perform testing. Responsibilities for testers include: administrate walkthroughs, administrate test case development, support the problem reporting system and administrate experience reporting.

**Phase 3 - Define Teams.** An independent test team is created and separated from development. This requires that the practices from the previous phases are well introduced and accepted by the personnel. Without, for example, a well-functioning problem reporting system that is used by the personnel, a division of the test team and the development team will be inefficient. The roles within the test team are clearly defined. Some testers will, for example, concentrate on system testing whereas others will concentrate on security and performance testing. This enables the test team to perform a more thorough testing of the software. It also gives the chance for the testers to become experts in a specific type of software testing.

### 3.2.3 Verification and Validation

**Phase 1- Use Checklists.** Checklists are created for the most important tasks such as GUI testing and platform testing. Initial checklists should be written prior to a new project. If checklists already exist, they should be reviewed and updated to fit the needs of the new project.

**Phase 2 – Perform Walkthroughs.** Walkthroughs should be performed before the software is ready for execution, ideally in the design phase. The walkthrough team mainly consist of developers and designers. One should assign one tester to the walkthrough team, so that comments can be made from a testing viewpoint.

**Phase 3 – Perform Inspections.** Inspections will more often replace walkthroughs. As the organisation becomes larger, a more formal way of performing reviews is required. Inspections are more formal than walkthroughs as they require the participants to prepare in advance. A tester should be assigned to the inspection team as to the walkthrough teams. Routines and responsibilities for inspections are established.

### 3.2.4 Test Administration

**Phase 1 – Basic Administration of Test Environment.** The test environment must always be available for testing when necessary. Basic administrative activities are: Organise a testing environment for each project and make this available, update the testing environment, and document the test environment and how it is used.

**Phase 2 – Test Cases.** Test cases are derived to assure that the most common situations and actions are tested. The
development of these cases takes a lot of time and several testers will need to work with this. Test scenarios are
developed that include several test cases. The test cases will be used when testing a system and will especially be
helpful when performing regression testing, as the actions are determined beforehand. The advantage of using test
cases is that the test team can work in parallel with the development. Instead of describing how to reproduce the
failure and how the failure is made manifest, the tester only needs to describe the effect of the fault and the test case
that produces the failure. Tests defined by test cases cannot, however replace all testing performed on the product.
Some testing must be performed in a more ad-hoc nature, but it must still be documented.

**Phase 3 – Risk Management.** Risk management aims to identify problem areas at the beginning of the project and
avoid or reduce the impact of issues within these areas. To be able to adapt risk management, a well-organized
problem database must be in place as well as experience reporting.

### 3.2.5 Test Planning

**Phase 1 – Test Plan.** A test plan is created in order to collect all test related planning in one document. The IEEE
Standard for Software Test Documentation [19] is an example of a standard that can be used as a template for this
purpose. An important part of the test plan is the creation of milestones for the project.

**Phase 2 and 3 – Coordinate SQA.** Quality assurance routines should be established. This will ensure that software
will not be released before it reaches a certain level of quality and will enable the sales department to give a
measurement of the quality of the software to their customers. This will increase the trustworthiness of the products
and the whole organisation.

### 3.3 Summary

This minimal test practice framework is a lightweight approach to meet the needs of a small but emerging software
development organisation. Focusing on these categories and introducing the relevant practices implies a very cost
effective improvement approach utilising well proven general practices as well as the possibility to adapt these to the
special local situation at the target company.

### 4 Derivation of the Framework

The MTPF was originally developed in a case study in a small and emerging company in Sweden, Scalado. The test
practices to be introduced at Scalado were identified using an observational empirical study [5, 17, 20]. This section
contains a summary of the procedures for deriving the test practice framework, which is intended to strengthen its validity. An overview of the work performed in order to identify the test practices is provided in Figure 2.

The first step was to observe the current situation and provide background information. The result of this step was a draft set of practices (version 0), identified through observations and interviews. The second step was to introduce the draft to the developers and other employees that were involved in the future use of the practices, providing feedback on the framework. The test practices were then revised according to the feedback produced and were further developed and prepared for actual introduction in the company (version 1). Experiences from three months of application were fed back and integrated in the framework (version 2). Then a survey of 12 software companies validated the framework, which again was adjusted based on the validation feedback (version 3), which is the version presented in Section 3.

The Initial observations and Feedback steps are presented below, while the Application step is presented in Section 5 and the Survey validation in Section 6.

---

**Figure 2 Overview of Framework Construction**

The Initial observations and Feedback steps are presented below, while the Application step is presented in Section 5 and the Survey validation in Section 6.
4.1 Company Background

Scalado was founded in the year 2000 and has around 25 employees today. Scalado develops tools for creation, presentation and maintenance of image-based digital information on the Internet, mobile Internet and hand-held devices. The company has grown rapidly and is expected to continue to expand in the future, although due to the economic recession during the last couple of years, Scalado has been forced to cut down some personnel.

Testing has previously been performed largely ad hoc but as the company is in an expanding phase, the need for a more thorough test method has increased lately. Scalado has not yet come to the point where projects are running over budget by far more than expected, but management and all project members are certain that this situation will arise if testing practices are not structured.

Scalado has on occasion attempted partial adaptation of fragments of the Extreme Programming (XP) [20, 21] development process such as pair programming. The development can despite this be most appropriately described as completely ad hoc. Testing is not adopted according to XP at all. Instead a separate test person is performing system test, as the customer is not available at site.

Integration testing is performed by using CPP Unit [22], a test-framework that integrates units to an existing system. The tester performs system testing by using the application and reports problems found. A test environment is built with virtual machines for easier administration of test. Acceptance testing is performed shortly before release.

4.2 Research Aims

In order to more systematically record the current situation, an observational study was launched. The research aims of this study are:

1. Identify a set of testing practices appropriate for introduction in an emerging, rapidly expanding, development organisation.
2. To determine how these practices be organised into a series of phases where each phase follows naturally from the last and is appropriate for a larger number of developers.

The aims are investigated using qualitative, observational research techniques, both participatory and non-participatory, and interviews [5, 14, 17, ] at Scalado.
4.3 Data Collection

The data collection in this study is performed using different means:

1. Observation of testing
2. Open-ended interviews
3. Observation at design meetings
4. Participant observation
5. Semi-structured feedback interviews

The units of analysis in the study are defined as the following three units: The current development process, the current method of testing and improvement suggestions to the existing test method. These units of analysis are summarized in Table 2.

<table>
<thead>
<tr>
<th>Unit of analysis</th>
<th>Subjects</th>
<th>Timing</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development process</td>
<td>Developers, project manager</td>
<td>Beginning of the data collection</td>
<td>Collect data to define an overview of the development process. Define the interfaces between development and testing. Seek parameters in the development that can affect software testing negatively or positively. Understand the type of product being developed.</td>
</tr>
<tr>
<td>The existing test method</td>
<td>Testers, project manager, developers</td>
<td>Beginning, parallel to defining the overview of the development process</td>
<td>Collect data to define the overview of the existing test method. Scale-up the overview and collect data to define practices for framework.</td>
</tr>
<tr>
<td>Suggestions about change to the existing test method</td>
<td>Testers, project manager and developers</td>
<td>Entering criteria: Development process is defined, existing test method is defined and enough theoretical knowledge to suggest recognized solutions to common testing problems</td>
<td>Collect data about positive and negative areas in the existing test method. Present the found bottlenecks and collect comments. Present solutions to the bottlenecks and collect comments. Space for the staffs’ own solutions to negative areas in the existing test method. Collect attitudes about process change. Collect data about how critical testing is for the product quality.</td>
</tr>
</tbody>
</table>

Table 2 Units of Analysis

The goal of the first data collection occasion was to observe how testing was conducted before any change was proposed. A tester performed testing on a new patch during the observations. After the observation, the tester was asked questions about certain actions during the testing. Based in this initial observation a set of questions were prepared for the second data collection occasion, open-ended interviews.

The open-ended interviews consisted of questions to all six members of one project in the same area: how new
systems are designed initially and how testing is performed today. The questions were slightly differently formulated depending on the role of the interviewee; developers, testers and a project manager. The open-ended interviews generated a clearer picture of the software development process; especially how development interfaced with testing.

The third data collection occasion consisted of attending design meetings and mostly verified and clarified the results of the open-ended interviews.

The fourth data collection occasion consisted of participatory observation while testing a new patch. Observations concerning practical difficulty in software testing were collected. Also, ideas from developers concerning improvements of the existing test method were gathered.

The fifth data collection occasion was intended to provide feedback on the results from the analysis of the data collected on the previous occasions. The collection and its results are therefore described in Section 4.6.

4.4 Analysis of Observations and Interviews

The analysis discovered some bottlenecks in the existing test approach. The open-ended interviews and the attendance at the design meetings revealed that testing was not planned for in advance. This resulted in deadlines overrunning and project cost increases. Also, testing was assigned very little time and resources and was performed too late in the project.

The participant observation, while testing a patch, revealed that the test environment was badly organized which lead to inefficiency as the test environment had to be reorganized before the testing could start. The participant observation generated ideas about attributes needed for the problem reporting syntax such as including the machine message as an item.

Most of the problems are of a managerial nature rather than technical. The improvement proposals from the interviewees on important areas for successful testing are:

- Using checklists
- Developing a test plan
- Using a problem reporting system or problem reporting syntax
Furthermore, the interviewees thought that *a time plan with milestones* would be the most important part of a test plan for effective testing. When it comes to checklists, they thought that it is most important *to develop a checklist prior to walkthroughs*. However, when they saw examples of checklists, they thought that the *GUI checklist* would be most useful. The questions about the test environment identified the need for someone to take the responsibility for the test environment and for the test environment to be realistic. However, the interviewees also pointed out that it is important that the test environment is available. Other issues that distinguish a good test environment are that:

- It is specified how the test environment is built
- The test environment is fast
- Enough time is assigned to test if the test environment is slow

The existing problem reporting syntax was inefficient and unstructured according to the interviewees. They agreed that an interactive problem reporting system is the best way of administrating the problem reporting. Some form of improved common syntax provides the second best alternative and the worst alternative is the existing problem reporting syntax. Important items of a problem reporting syntax are according to the interviewees:

- Instructions on how to reproduce the problem
- Priority to problems
- Search function among the problem reports
- A minimum of fields to fill in – simplicity
- A field to assign responsibility for each problem

All interviewees agreed that it is beneficial to have a common coding standard. An informal coding standard did exist but everyone did not use it. There are different opinions about the importance of walkthroughs and their efficiency. However, most interviewees thought walkthroughs are more efficient for the purpose of learning from others experience than for finding faults in the product.
4.5 **Test Practices Structure**

Based upon the analysis of the observations the most important areas identified to introduce as early as possible are:

- A problem reporting system
- Checklists
- A test plan
- Defined responsibility for test

It is also concluded that walkthroughs should not be introduced until the organization becomes larger and that a coding standard should be introduced but is not top priority for effective testing.

This ended up in a draft of the MTPF composed of five areas, as defined in Section 3:

- *Problem and Experience Reporting*
- *Roles and Organisation*
- *Verification and Validation*
- *Test Administration*
- *Test Planning*

4.6 **Feedback**

The proposed practice framework was shown to the employees at Scalado during an open interview and their comments were recorded.

The interviewees generally thought that the draft of the test method would meet the needs at Scalado. The problem reporting system, checklists, a test plan and responsibility for test are considered to be important parameters for effective testing. Walkthroughs, the test environment and a code standard are less important parameters.

The MTPF was reworked regarding wordings and other minor issues based on the feedback.
5 Application of the Framework

The next step in the development of the MTPF was applying it at Scalado. Observations were mainly done in the first three-month period and after one year of operation. As the company comprised about 10 developers, the first phase of the framework was mostly in focus.

The first month of application was dedicated solely to the introduction of each practice of the test method. The next two months, the introduced practices were used and evaluated by performing software testing in three different projects. Each project is of a different type as described below:

- The first project involved testing a patch of an existing product.
- The second project involved the reengineering of a sub-system of an existing product. The project started and ended during a period of three months.
- The third project involved testing a totally new product

In this section the lessons learned from the application of phase 1 within each test practice category are summarised. The general structure of the framework is considered sufficient, while the application experience adds to the detailed level of the framework. It is important to remember that the framework would be applied differently in a different case. This level is to be tailored for each organisation, but the examples are reported for illustration purposes.

5.1 Problem Reporting

The application identifies Define Syntax as an important feature of the problem reporting. The problem reporting syntax had to fulfil the following requirements:

- Include a set of fields that together describe a problem in any project
- It must follow the workflow of the tester
- It must be easy to understand for the developer and give motivation to correct the problem
- A problem report must have a unique number
- The syntax must allow grouping of problems
A set of fields was developed that fulfilled the above requirements and suited the development at Scalado.

It is highly important that the syntax is defined to allow flexibility. If the syntax is defined using an advanced database system (i.e. entering phase 2 directly) it will probably be hard to change the syntax even though it has shown to be ineffective. To avoid this scenario, the strategy at Scalado was to define a syntax using Microsoft Excel as reporting tool and then evaluate the result after using it for a while. The syntax probably needs to go through another (and a larger) project before it can be thoroughly evaluated. It is, however, already observed that some adjustments must be made before creating a problem reporting system.

Another level of defining the problem reporting syntax is of a linguistic character. Experiences at Scalado announce that it is important to discuss some basic linguistic definitions prior to testing a project. The long-term solution on the linguistic difficulty is probably to let the syntax develop as a consequence of striving towards constancy when writing problem reports.

5.2 Roles and Organisation

The responsibility for software testing was assigned to one person to avoid the problems experienced earlier, when testing was performed by the support manager, and no test team existed.

During the period of introduction, the test manager did the software test planning and performed software testing in the three projects mentioned above. The introduction became flexible since the person who experienced the practical difficulties with software testing was also the one who introduced the practices.

The result of defining the responsibility of software testing has affected the structure and organisation of testing at Scalado in a positive way. Not only is testing more structured but the marketing department can also more accurately plan the release date for each product.

The development department has also gained from the defined responsibilities as they are working with a person who is specialised in software testing. The developer does not need to think as much about the overall quality of the developed software at system level and can instead concentrate on the details when correcting faults at unit level. The testers’ role is to find faults indicated by product failures, distribute problem reports of these to the developers, perform regression testing and communicate the status of the project to the project manager. Currently hired consultants perform the tests. It is the aim of the company to have a permanent test responsible team that can be complemented by consultants when needed.
5.3 Verification and validation

The MTPF introduces checklists as a verification and validation practice in phase 1.

Prior to the platform testing, checklists describing the different configurations to be tested upon, were developed. This type of checklist was very useful as it can easily be transformed into a status report telling which platforms that are supported and which are not.

In the second and third project the sections Features to be tested and Pass/Fail criteria of the test plan came to be the basis of a functionality checklist, see Section 5.5. This type of checklist enables the tester to get an overall picture of the functions tested. In the second project this concept was further developed. By developing a test case description, which included the features to be tested and the pass criteria for each feature, regression testing could be performed faster.

A GUI checklist was developed but there has not been any need for this sort of checklist in the three projects.

5.4 Test Administration

The existing test environment at Scalado was very fragile. In some situations it was hard to decide if the test environment or the software under test caused a failure. The test manager and the system administrator started to reorganize the test environment by reinstalling its components and naming well defined configurations. After installing a configuration it was tested by using it for a while to confirm that it represented a stable test environment. This generated a more reliable test environment.

The following lessons were learnt:

- The test environment becomes a bottleneck since parallel testing cannot be performed.
- After setting up a test environment, the test machine itself must be tested.

This practice was, surprisingly, one of the hardest to introduce. Test environments appear to be taken for granted by developers and scarce testing hardware resources are often taken over by development, considered more critical. Alternative test environments are currently being investigated at Scalado where the test environment consists of three computers, two test computers and one administrator computer.
5.5 Test Planning

The Test Plan is the first practice of test planning. A test plan has been written prior to the testing in each project using parts of the IEEE Software Test Plan [19] as a template. The sections included in the test plan are somewhat different between the three projects but the main sections are:

1. Features to be tested
2. Pass/Fail Criteria
3. Schedule
4. Responsibilities

An important experience from planning the three projects is that the sequence of how testing is performed may determine the final software quality. By performing systematic functionality testing on a stable system before starting the platform testing, the time spent in relation to the given software quality will increase. As the most troublesome functional faults are corrected before platform testing is performed, the number of failures that may occur in the platform testing decreases and therefore the fault causing the failure on a specific platform can be more thoroughly investigated.

The second project differed from the others, as it was the only project that started and ended during the three months period. This enabled the test manager to start the planning of the project well before the testing phase had begun. The flexibility of the small organization made it possible to change the original project plan and start the test phase approximately two weeks before it was originally planned to begin. By starting the testing while the last functionality was being implemented, higher software quality was gained than in other projects. The conclusion of this experience is that to be able to plan successful testing, the test manager must dare to make use of the flexibility that exists to a higher degree in small organisations.

The definition of the responsibility also resulted in testing not being overlooked when calculating the total time of the project. Testing was, however, still dependent on the development being completed in time. If development overruns its deadline, the time available for testing will decrease, as small organisations are often completely dependent on one product with short time to market.
5.6 Phase 2

Currently only a web-based defect reporting system has been introduced from phase two. Other practices from phase two are intended to be introduced once the organisation starts to grow again.

5.7 Quantitative indicators of application

Quantitative measures were taken from two similar projects at Scalado, one before the application of the framework (Project A) and the other after the application of phase 1 of the framework (Project B). It is of course impossible to find identical projects as these are always of differing size and nature, but the two that have been used are as similar as is possible. The projects are releases of image zooming software for use in web applications, both developed with around five developers. The data has been extracted from fault reports and time reporting from the projects and is presented in Table 3.

<table>
<thead>
<tr>
<th></th>
<th>Project A</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td>% of time spent testing</td>
<td>5%</td>
<td>20%</td>
</tr>
<tr>
<td>Number of documented faults</td>
<td>20</td>
<td>400</td>
</tr>
<tr>
<td>Number of documented corrected faults</td>
<td>0</td>
<td>300</td>
</tr>
</tbody>
</table>

**Table 3: Quantitative indicators of the application of the framework**

Before the introduction of the framework none of the projects were writing or using test cases. After the introduction projects with more than 6 weeks duration wrote and used test cases. Before the introduction of the framework no projects used test planning. After the introduction projects with more than 6 weeks duration always use test planning. Projects under 6 weeks duration still use ad-hoc testing.

Project A’s indicators are the result of the code and fix style of ad-hoc working prior to the framework. The product is in development almost until it is released, there is a low number of documented faults in total, most faults are not documented at all. The indicators for Project B demonstrate a clear increase in the time spent testing the product. The increase in the number of documented faults is due to the fact that faults are now actually being documented in a structured fashion according to the framework as well the larger portion of time is spent in testing. The product
quality is perceived higher in Project B due to the more effective fault removal of the framework, i.e. the larger number of faults does not indicate a lower quality product in this case.

5.8 Summary of application

Based on the application, it can be concluded that the first phase of the MTPF is feasible and meets acceptance among practitioners. The light-weight approach is appealing to them and the focus on involvement in the introduction method, creates acceptance as it begins with the needs of the organization. The quantitative measures used indicate that the framework has significantly changed the way of working at the company.

6 Survey Validation of the Framework

To investigate the transferability of the MTPF, it was validated using a survey with representatives from twelve regional software engineering companies. Size wise, these span from one-man consulting firms to 100+ developer development units. These obviously have different expectations of a testing framework.

The economic situation at the time of the survey, summer 2003, deserves to be mentioned. The recession has affected all of the companies and, as was seen in the survey, none are growing rapidly. It would have been very interesting to perform the validation in an economic climate similar to that of the late nineties, with many companies growing very rapidly.

6.1 Method

The survey methodology [5] was chosen in order to be able to quickly reach many subject companies, as the transferability of the MTPF was considered the most important issue. Representatives of companies involved in a regional software process improvement network, SPIN-Syd\(^1\), were used as subjects. The framework was presented to the representatives first in an oral presentation and then in the form of a written summary. After the initial presentation the survey forms were handed out and the subjects were given as much time as they needed to complete it. During the survey the subjects had access to the written summary and were also free to ask questions.

\(^1\) www.spin-syd.org
The first section of the survey gathered general information about the company. The information requested was the number of developers in the company, the age of the company, the current rate of change in the company size and the relationship the company had to testing methodologies. The relationship was determined by classifying the companies into three groups:

(i) Companies that have gotten further than the concepts involved in the framework. Here the primary interest is whether the framework would have been useful in attaining their current level.

(ii) Companies that are currently using a similar method to the current framework. Here the primary interest is how well the methodologies match.

(iii) Companies that have not focused very much on testing. Here the primary interest is if the framework would be a suitable tool to begin with.

All of the companies involved in the study can therefore contribute with useful information regarding the MTPF from their unique perspective.

In the second part of the survey the subjects were asked to look at the framework stage by stage and comment on each area with respect to the situation at their company. Thirdly the subjects were asked explicitly if they thought anything useful to their company was missing in the framework. The final survey sections regarded the subjects’ perception of how the framework introduction method would perform in their respective company. An overview of the survey is shown in Table 4.

<table>
<thead>
<tr>
<th>Survey section</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Subject classification, Size, age, growth rate and relation to test frameworks</td>
</tr>
<tr>
<td>2</td>
<td>Framework evaluation stage by stage</td>
</tr>
<tr>
<td>3</td>
<td>Completion of the framework</td>
</tr>
<tr>
<td>4</td>
<td>Perceived introduction method performance in the subject’s own organization</td>
</tr>
</tbody>
</table>

**Table 4 Overview of Survey Instrument**

### 6.2 Survey Results

The subjects largely filled out the survey form satisfactorily, although three did not complete section 2. All of these were however positive concerning the appropriateness of the survey to their specific company.
The results of the first section are summarised in Table 5. 

- **Size** denotes the number of software developers in the company.
- **Age** denotes the number of years since the company’s foundation.
- **Growth** denotes the rate of change of the company’s size, on a scale of - - strong reduction, - reduction, sq status quo, + growing, + + strongly growing.
- **Relation** denotes the company’s relationship to working with testing. 0 not worked much with this, = are currently using a similar approach, + are more advanced than the issues discussed here.

Here it can be seen that the range of companies spans from 1 to 100+ in number of developers and from start-ups to relatively mature companies, for software development that is. Most of the companies are currently not growing. The four that are growing seem evenly distributed across both size and age. A trend can be seen in the relation to working with testing. Larger companies appear to have come further than medium and smaller companies.

<table>
<thead>
<tr>
<th>ID#</th>
<th>1</th>
<th>2</th>
<th>6</th>
<th>7</th>
<th>12</th>
<th>11</th>
<th>3</th>
<th>8</th>
<th>5</th>
<th>4</th>
<th>10</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>1</td>
<td>1-2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>70</td>
<td>95</td>
<td>100+</td>
</tr>
<tr>
<td>Age</td>
<td>0.9</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2+</td>
<td>15</td>
<td>&gt;10</td>
<td>15</td>
<td>old</td>
<td>40</td>
<td>~20</td>
</tr>
<tr>
<td>Growth</td>
<td>sq</td>
<td>sq</td>
<td>+</td>
<td>sq</td>
<td>sq</td>
<td>+</td>
<td>sq</td>
<td>+</td>
<td>sq</td>
<td>+</td>
<td>sq</td>
<td>+</td>
</tr>
<tr>
<td>Relation</td>
<td>0</td>
<td>=</td>
<td>0</td>
<td>+</td>
<td>=</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>=</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Table 5 Classification Results**

Smaller companies explicitly verified that simple defect tracking in a file is adequate at this stage. Also clear from most of the subjects is that responsibility benefits from being defined clearly as early as possible.

Generally it can be seen that the larger the company the more of the parts of the framework are in place, in the staged order prescribed.

Many companies mentioned automatic testing as a completion of the framework. This is probably due to the large amount of attention given lately to eXtreme Programming (XP) [21] including its view on testing. This could, if found useful by the organisation, easily be integrated into the first stage in the MTPF under test administration. Several companies also mentioned requirements engineering and the connection to testing and that this should be integrated in some way into the framework. The point is very relevant and should not be disregarded by any company doing software development. Including requirements focus however, would make the MTPF more a complete software process description, which it is not intended to be.

All subjects except one responded positively to whether the introduction method might perform well in their company. This exception expressed a need to vary the ambition level of verification and validation from project to project depending on the customer and product being developed.
One subject mentioned patterns as a good method for experience gathering and setting syntax early. Another subject suggested emptying the framework prior to the introduction of each phase in order to focus on that which is most relevant to the current situation. A few subjects responded in a discussion after the survey that the logic of what testing activity was placed in each area was a little different to the thinking at their company. As a result of this, minor revisions were made in the positioning of activities in the MTPF.

One of the largest companies reported being stuck with an outdated problem-reporting syntax in a reporting system. This demonstrates the importance of working out a good syntax prior to locking it in a system, but it is of course impossible to be completely insured against future needs for syntax changes.

6.3 Survey Conclusions

In general the MTPF appears suitable, or would have been suitable earlier, to most of the represented companies. As a result only minor structural revisions were performed after the validation. Suggestions for improvements were noted but are apparently not relevant to more than a single company in the group. The flexibility of the introduction method would allow companies with special needs to accommodate these.

A general observation noted during the survey is that one of the most important needs for the companies seems to be a need for structure in the improvement work rather than extremely detailed specific practices. This supports the flexible framework approach used.

7 Validity of the Studies

This paper presents a minimal test practice framework intended for small and emerging software development organisations. An important and relevant question to raise is how valid the MTPF is, firstly, regarding the validity of the results for Scalado, and secondly for transferring the results to other companies in similar situations. The first issue is addressed by reporting measures taken in the MTPF derivation, and the second issue is addressed by a survey among potential users of the framework. The validity issues regarding the derivation as such are referred to as credibility, confirmability and reliability, while the issues on transferring the results are referred to as transferability [5].

The MTPF was derived during a period of three months, and then internally validated during another three months. This persistent observation reduces the threat that the characteristics of the organisation are not known to the
researchers. The many sources of information in this study allow a high degree of triangulation, i.e. having multiple sources for the information, not least the peer debriefing which provides feedback opportunity increases both the confirmability and credibility in the study. Reliability was addressed by rigorously researching the theory behind observational research and interviews [5, 14, 17] and careful preparation before each data collection round.

Transferability is a central point in the creation of the framework. If there is no transferability the created framework is only applicable for the presented situation at Scalado. In order to investigate the transferability, a survey was launched with twelve local software development companies. This survey is presented in section 6, while the validity of the survey as such is analysed below, and reported in Table 6.

<table>
<thead>
<tr>
<th>Threat</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjects misunderstanding the framework and introduction method.</td>
<td>Publication of an early version of the framework to increase clarity, presentation of framework and introduction method before survey in both oral and written forms, expert support during the survey completion.</td>
</tr>
<tr>
<td>Subjects misunderstanding the survey questions and their intent.</td>
<td>Review of survey instrument prior to use, expert support during the survey completion.</td>
</tr>
<tr>
<td>Subjects do not care enough to give the right answers.</td>
<td>Promise feedback of results from survey to aid the company’s process improvement work.</td>
</tr>
<tr>
<td>The subjects represent the most relevant people to answer the survey.</td>
<td>The subjects were asked to recommend more suitable people to pass on the survey to if they were not the most suitable. None did.</td>
</tr>
<tr>
<td>The subjects are unwilling to give information that makes their organisation or themselves look bad, or not advanced.</td>
<td>None, but the SPIN-Syd forum for companies is very open and sincere so this should not affect significantly.</td>
</tr>
<tr>
<td>The researchers misinterpret the intent of the subjects’ survey answers.</td>
<td>Feedback both of the results to the subjects and of the subjects’ comments on the results to the researchers.</td>
</tr>
</tbody>
</table>

Table 6. Survey validity measures

Given these threats and the measures taken to avoid them it can be concluded that validity of the survey is adequate. Further, the results of the survey indicate that the transferability of the MTPF is good.

8 Summary and Future Work

In this paper a minimal test practice framework and a corresponding introduction method suitable for introduction in a small, rapidly expanding development company have been created. The first phase of the MTPF and the introduction
method have been evaluated by introducing them in the subject company. A significant contribution is the low threshold for introducing each phase compared to other test processes and practice frameworks. The framework is derived and validated using rigorous and well-established research methodologies, which secures the validity of the framework.

The natural continuation of the work presented in this paper is to continue introducing the MTPF at Scalado and to continue to observe the results. A second natural continuation of the work is to try to introduce the framework in other different organisations in a similar situation. It will be interesting to see how the framework is adapted in order to suit new circumstances.

9 Acknowledgements

This work was partly funded by The Swedish Agency for Innovation Systems (VINNOVA), under a grant for the Center for Applied Software Research at Lund University (LUCAS). The authors would like to thank Scalado for their extensive cooperation and support for the project and SPIN-Syd for providing a stimulating forum for software engineering practice and research in the south of Sweden and cooperation with the survey.

10 References


