UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
INSTITUTE OF COMPUTER SCIENCE

Evari Koppel

Software Test Management Tool Evaluation Framework

Master's thesis (30 EAP)

Supervisor: Raimundas Matulevičius

Author: ..... "....." May 2012
Supervisor:..... "....." May 2012
Approved for defence
Professor:..... "....." May 2012

TARTU 2012

Abstract

Software testing has proven its value for software development increasingly over the last decade. With the recognition of the benefits of software testing, several software test management tools (TMT) have emerged on the market. Although there exist different approaches, there is no method for a systematic TMT assessment.

This is a problem because to our knowledge, evaluating TMT is rather a subjective task, heavily depending on the evaluators' opinions rather than based on the objective approach. The same problem applies when test managers are asked to evaluate whether their currently used TMT meets the company's expectations.

In order to understand the importance and neccessity of TMT evaluation we perform a literature study on software testing processes and existing TMT market studies. Then we map together the identified test activities and test artifacts. The results help us formulate and design an online questionnaire and perform a TMT survey within the Estonian IT companies.

Based on the survey results, a framework for evaluating TMT software is created. Such a framework could potentially help companies to measure the TMT suitability to company's goals and to decrease subjectivity of the TMT assessment. The framework also provides test and project managers the understanding whether their current TMTs meet the company's expectations. We validate the framework with a case study performed among Quality Assurance specialists to collect information on the framework usability.

Possibilities for future work based on this thesis are numerous. The framework can be made into an application for ease of use and wider distribution. Expanding the research onto other European countries is another viable choice. Also expanding the TMT requirements based on new trends in testing can be taken into consideration. In conclusion, we believe this thesis contributes to the testing community with a practical TMT evaluation method.

Table of Contents

List of Figures and Tables

List of Tables

Chapter 1 – Introduction

Software development is a process which can be considered rather new compared to other manufacturing areas such as the production of automobiles, processing of food, production of every-day accessories. As such the quality assurance procedures and tools for implementing are still making *baby steps*. Until 1980's the main focus of testing was to find errors in the developed software (Myers, 1979). The concept changed during the next 20 years and starting from 21st century software testing focus is to prevent errors being released to end users as it has proven that the cost of resolving issues in the project's early stage is cheaper than once it has reached the customer (Boehm *et al*, 2001). This can be achieved by using automated and manual software testing. To meet such need several new Test Management Tools (TMT) have surfaced, both free-ware and commercial versions.

Choosing test management tool to support software development is a tricky activity. Test management tool reviews and their correspondence to companies' requirements are mostly performed by the tool vendors. In our thesis we search for the answers to the questions:

1. *what are the requirements when selecting a TMT;*
2. *how to evaluate whether the TMT meets the expectations.*

We are interested whether there are different expectations to the TMTs and if there are, then what is driving such differences. Once we know these requirements, we will use them and formulate a process, which will assist choosing appropriate tool.

The scope of the study covers the quality assurance from manual testing perspective and a survey is carried out in Estonian IT companies. We will analyze the theory behind software testing methodologies, carry out a case study among existing TMTs and with the combined information, we will perform a survey among IT companies to find out what are the expectations for a TMT.

The thesis contributes by providing a TMT evaluation framework which is developed based on Estonian IT companies' expectations to TMT requirements. To test framework validity a case study is carried out to ensure that its usability meets the quality assurance specialists' expectations. In addition we have found that there are 7 testing activities and artifacts identified that are equally important and required TMT features by all Estonian software development companies. However there are additional features seen as expected, but this depends on the company characteristics.

Related work. Software testing dates back to the 1950's when it was hard to distinguish testing from debugging[1]. From late 1960's first code coverage monitoring programs were created, which could be considered the first computer aided TMTs. Gradually over the decades more complex approach was adapted ranging from waterfall testing models to recent day agile models (described in Chapter 2). Starting from the late 1990s, more and more computer aided programs have been introduced to assist with the software quality assurance (QA) processes such as Robert Poston's specification-based test generation tool, Rational Robot and SQA TeamTest. Around year 2000-2005 new commercial versions of test management tools emerged, offering the support of managing larger project testing. Since then there has been a rapid growth of new products.

To our knowledge no official academic research has been made on the topic of test management tools with the emphasis on the QA processes.

Yang *et al* (2006) reports on a *survey for the coverage-based testing tools*. However their focus is on tools offering automation support. They investigated 17 tools which were analyzed

[1] Retrieved May 11, 2012, from http://extremesoftwaretesting.com/Info/SoftwareTestingHistory.html

in depth for coverage-based testing. In coverage based testing techniques, test suites are selected to cover some structural aspects of the model with respect to given coverage criteria (Gaston, 2005). They found that each tool had some unique features tailored to its application domains. In their study they excluded all tools that did not support test automation, thus covering a different aspect of test management than is this thesis. The outcome of their research was a table assisting in choosing a software test tool based on the programming language used by the company.

Elsewhere *Garousi* evaluates testing tools used in the Northern American IT market (Garousi, 2009). There the focus is to find test management tools for application in university courses, thus providing his students an opportunity to use the tools applied in IT industry. The research evaluated the students feedback on the tools used. While the paper does not result any concrete evaluation form for choosing a testing tool, the conclusion suggests further research in the field on testing tools.

This thesis was inspired by discussions and conversations between software QA specialists. Whenever the topic came to test management tools, there were heated arguments which tool fits for use and which tools are simply good because of their published reviews. Indeed, TMT reviews are often biased on marketing or receive positive feedback since the brand has good reputation (McGlohon, 2010). Also there does not seem to be a reliable source for getting unified review of a tool – some reviews rate the product in 0-5 star system (with 5 as highest), others in 1-10 point scale (with ten being highest). Some websites offer only feedbacks for the products (either bad, good, helpful). A product might receive 4 stars by one system and 7 points in another – while both are measurable values, it is not clear what the weighs of both results are. Relying on the evaluation of specialist is also subjective, since they belong to different testing schools. For example Bach (2011) promotes context-driven testing while strongly criticizes any test certification. Thus relying only on existing evaluation, such as (Bach, 2011) would already exclude some potentially acceptable tools.

We have not found anywhere an evaluation form which would provide any academic approach of the testing processes. With this thesis and the performed survey we will deliver a framework for choosing a TMT with the focus on the testing processes.

Structure of the thesis. Our thesis is arranged into three parts, each consisting of chapters. We start with Chapter 1, introducing the paper, the research question and mentioning other related works. Part 1 consists of three chapters: (i) Chapter 2 presents the theory on software testing and introduces the testing activities and artifacts mentioned in literature; (ii) Chapter 3 describes the existing TMTs and the supported activities and artifacts. Later in the paper we use these to compile the survey; (iii) Chapter 4 lists existing tool evaluation processes and explains why we create our evaluation framework.

Part 2 is composed of two chapters: (i) Chapter 5 describes the survey we carried out among Estonian IT companies. This section presents the questionnaire, respondents' background and the survey results; and (ii) Chapter 6 applies the findings from previous chapters and formulates them into TMT evaluation framework.

Part 3 contains a single chapter where the validation of the framework is described.

Part 4 summarizes the paper by providing conclusions from the paper and presents possibilities for future work.

There are 5 appendixes attached to the paper: (A) the online questionnaire; (B) survey results; (C) TMT product diagrams; (D) guideline for using TMT evaluation framework; and (E) research paper submitted to conference.

Part 1 - Theory and State of the Art

Chapter 2 – Theoretical Approach to Software Quality Assurance

In this chapter we explain what is software testing and examine how software testing is addressed in various software development models. We will identify and list the activities and artifacts introduced in the software development models. The results of this chapter support us in surveying the test management tools currently available and establishing the expected features of a test management tool.

2.1 What Is Software Testing?

During the evolution of software engineering there have been many definitions of what testing is. It is often perceived as a 'magic bullet' (Myrvold, 2011) that will solve the problem of finding errors after product has been delivered to the customer. However testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions (Kaner *et al*, 1999).

Testing aims to execute a software-intensive system or parts of the system in a controlled environment and under controlled circumstances in order to detect deviations from the specification and to check whether the system satisfies the defined acceptance criteria (Pohl, 2010). By this definition it addresses only test execution and is not concerned about other software life cycle. Another definition is following: software testing is one critical element of software quality assurance (SQA) that aims at determining the quality of the system and its related models (Keyes, 2003). Here quality of the system can mean different things to different stakeholders. For example, for the software engineer quality represents the system's correspondence to requirements; while for the end-user it also means the usability of the system. Hence software testing should cover both internal and external expectations or in other words it is a part of software quality assurance. SQA is a formal approach to software development, automated regression testing, configuration management, versioning, profiling and release control with the goal of zero defects (Britannica, 2003).

In software testing, *the terminology* can vary depending on certification (i.e. International Software Testing Qualification Board (ISTQB), Quality Assurance Institute (QAI), International Institute for Software Testing (IIST)) used in the organization. Current thesis refers to the terminology used in the ISTQB certification where applicable.

Software testing life cycle is part of the Software Development Life Cycle (SDLC). It defines a set of stages outlining what test activities to perform and when to conduct them. These stages are planning, analysis, design, construction, testing, final testing and implementation, post implementation (Keyes, 2003). The testing can be divided into functional and non-functional testing. Just like other processes, SQA can be based on different software testing models which are described in Section 2.2. Depending on the company and the working culture, they can vary. However, independently of the model, almost all of them contain similar testing levels.

The main functional testing levels during the development process (see Section 2.3) include component testing (also referred as unit testing), integration testing and system testing (Pohl, 2010). Two other levels can be identified based on the objective: there are regression testing and acceptance testing.

Besides functional testing, there exists also non-functional testing. According to Keyes (2003), in contrast to functional testing, which establishes the correct operation of the software, non-functional testing verifies that the software functions as expected even when it receives invalid or unexpected inputs. Non-functional testing will be described in Section 2.4.

2.2 Software Quality Assurance Models

Software development processes are complex by their nature. Tucker says (2004) that the activities involved in the processes are intellectually demanding and may require significant creativity on the part of the process participants. SQA activities are related to SDLC, thus, it is imperative to know which software development model is being used.

2.2.1 Waterfall Model

In the 1950's and 1960's, software development was, mostly, an informal process, based on informal[2,3] requirements. However, as the software systems grew both in size and complexity, more errors and failures were introduced ranging from inadequate performance to unmaintainability. These failures led to more structured approach of the development life cycle. In 1970 the waterfall model was introduced (see Figure 1). This model consists of a set of stages (Tucker, 2004):

1. *Requirements definition*. The services that system must provide and its operational constraints that are defined;
2. *System and software design*. The overall structure of the system is designed and software subsystems are identified. Depending on the organization, the design may be fairly abstract or developed in detail. Structured design methods may be used to develop the software design;
3. *Implementation and unit testing*. The modules making up the system are individually developed in some programming language and tested;
4. *Integration and verification*. The system modules are integrated into a complete system and this is tested as a whole;
5. *Operation and maintenance*. The software is delivered to the customer and put into use. During its lifetime, it is modified to meet changing requirements and to repair errors discovered in use.

The waterfall model of the software process has been incorporated to many process standards such as the U.S. Defense standard, MIL-STD-2167A[4].

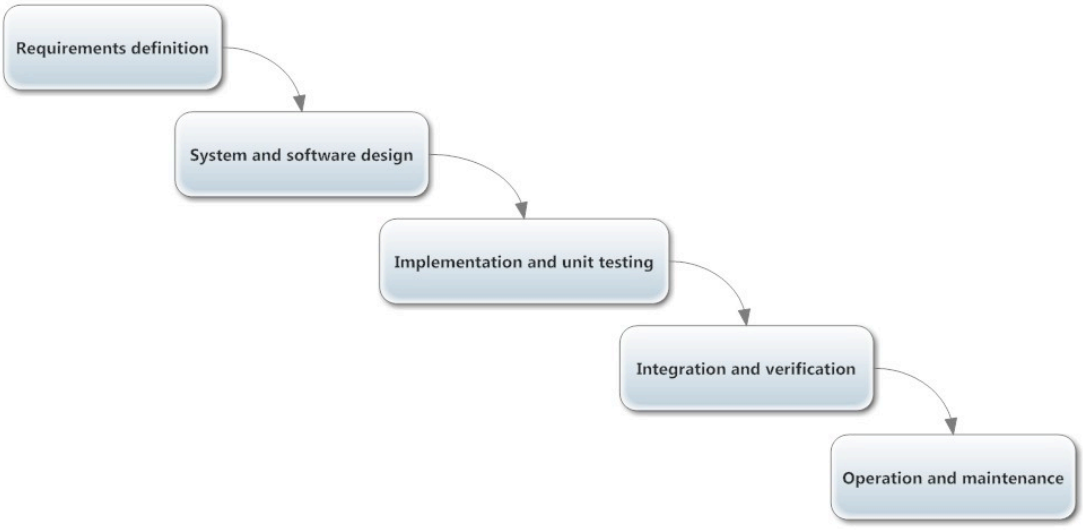

Figure 1. Waterfall-model (adapted from Tucker, 2004)

[2] The requirements for the software system were not fully documented or were communicated verbally.
[3] Retrieved May 11, 2012, from http://extremesoftwaretesting.com/Info/SoftwareTestingHistory.html
[4] Retrieved May 11, 2012, from http://en.wikipedia.org/wiki/MIL-STD-2167

Within the Waterfall model the following QA activities exist:

- Test scenarios creation – the activity is performed during the system and software design stage. The test scenarios are devised based on the created specifications and are not subject to major changes in later stages;
- Test execution – performed during the integration and verification stage. Test execution is based on the test scenarios;
- Test results reporting, including defect reporting – the activity of reporting the results of test execution. The reporting is done during integration and verification stage.

The involved artifacts are test scenarios, test cases, test sets, test reports and defect records. While the model itself does not imply on it, the test scenarios cover the software requirements and thus, are virtually linked to them.

2.2.2 V-model

The V-model represents a software development process, which is derived from the waterfall model. It maintains the development stages of the waterfall model but links the specific validation activities and validation plans with stages in the specification and design process (Tucker, 2004). In Figure 2 the horizontal axes represent time or project completeness (from left to right) and vertical axes describes the level of abstraction (coarsest-grain abstraction uppermost).

The V-model is document-based with one or more artifacts produced at each stage. This makes the project visible to management. Also it enables the validation of the requirements and specifications when they are created, thus, enabling detection of discrepancies and gaps in the requirements.

Although the weakness of the model is difficulties to cope with change in mid-process, the model will probably remain in use for large systems engineering projects since it allows process management, supports „offshore“ software engineering and is familiar to engineers from all disciplines.

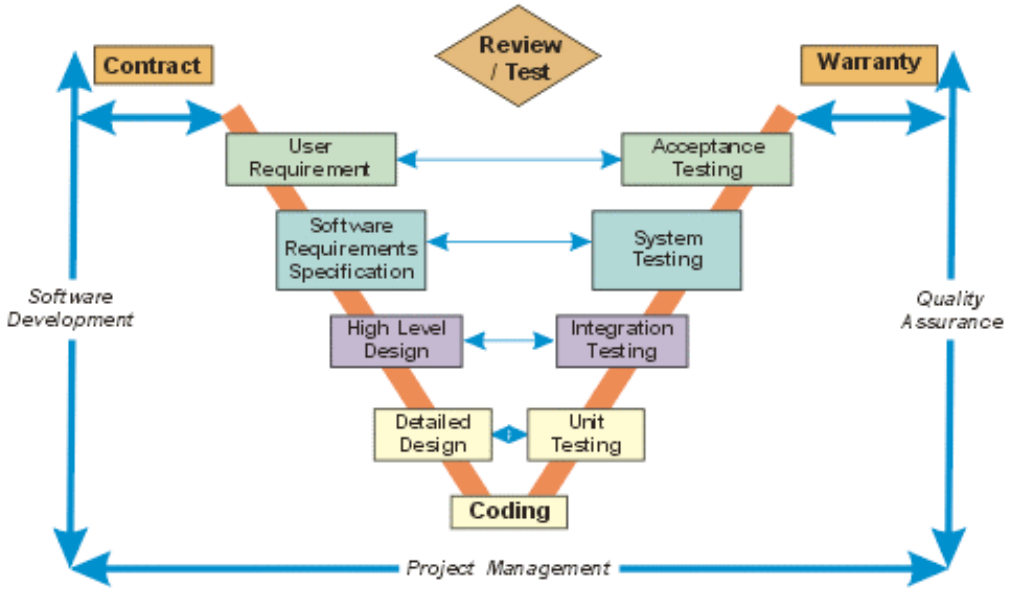

Figure 2. V-model[5]

[5] Retrieved May 11, 2012, from http://stormshadowsoftware.co.uk/Testing_Information.html

V-model from QA perspective contains several activities:

- User requirement analysis – by performing the analysis the test team receives inputs and requirements for acceptance testing. In the figure this is represented by the link between User Requirements and Acceptance Testing;
- Software requirements specification (SRS) inspection to verify completeness, consistency, feasibility and testability. By performing the SRS inspection, the test team receives the requirements and test scenarios for System Testing;
- Test plan creation based on the design – the test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications. On the figure this is shown as the connection between the High Level Design and Integration Testing;
- Preparation and division of test scenarios to integration, system and acceptance testing– in the V-model the software lifecycle moves left-to-right. Testing activities follow the same order and all planned test scenarios are prepared for each stage;
- Test design creation based on the test plans – user requirement analysis and SRS inspection provide the inputs for software test plan. The High Level and Detailed Design provide additional inputs for test design. As the V-model life cycle progresses from left to right, this implies that the test design is done after the test plans;
- Test execution – the test execution is performed in all of the stages listed on the right side of the V-model. The testing is performed during Integration, System and Acceptance testing;
- Fault reporting – the fault reporting in V-model happens at all stages. This is represented by the horizontal connections of different stages. The fault reporting applies when test team is doing requirements and design reviews and inspections, and also when they are executing tests;
- Test results reporting – the act of reporting the outcome of test execution. The reporting is performed during the Unit, Integration, System and Acceptance testing.

As mentioned above, the V-model offers greater visibility to the management and encourages tighter co-operation between the development and QA departments as they both are involved in the review of software requirements and design artifacts. In addition to the artifacts created in waterfall model, test plans are prepared. Test planning is concerned with setting out standards for the testing process rather than describing product tests (Sommerville, 1993). Keyes (2003) describes test plan consisting of system description, testing strategy, testing resources, testing metrics, testing artifacts and testing schedule. From these definitions the QA activities gain a new process – assignment of an activity to a QA resource.

Compared to the Waterfall model, traceability from requirements to design to test cases and to defects is better visible in the V-model, thus enabling to track requirements coverage and quality. The traceability is one of the key aspects provided by the V-model and will be addressed in our thesis.

2.2.3 Spiral Model

While the Waterfall and V-models are specification driven, the spiral model (Boehm, 1988) is iterative. The spiral model views the software development process as a spiral from initial conception to final system deployment (Tucker, 2004). It is also risk driven where project risks are identified and resolved before progress is made to the next stage.

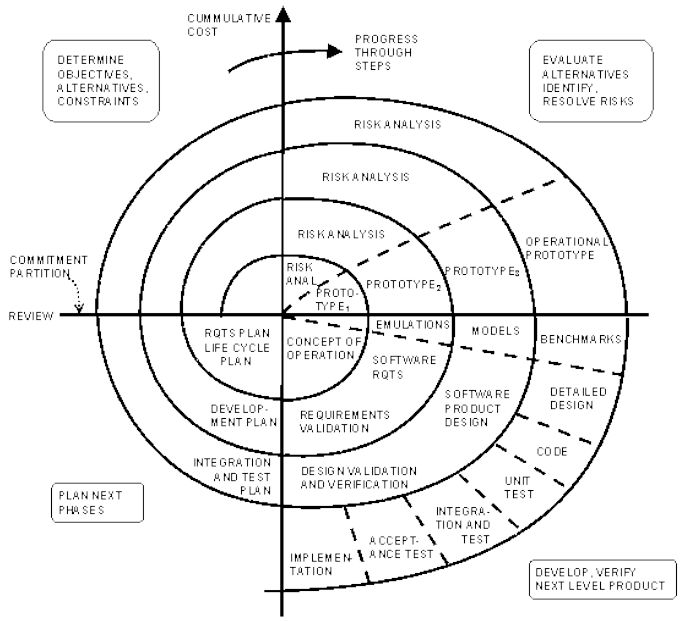

Figure 3. Spiral model (adapted from Tucker, 2004)

The stages in the spiral model iteration can be generalized as follows:

1. An objective setting stage, where the objectives of the iteration and the development constraints are established;
2. A risk-analysis stage, where the project risks are assessed against these objectives and where risk-resolution activities such as prototyping are carried out;
3. A development stage, which may involve design, coding and validation activities;
4. A planning stage, where plans for the next iteration of the spiral are drawn up.

This model is reasonable to use in projects where business goals are unstable but the architecture must be realized well enough to provide high loading and stress ability.

The spiral model does not introduce any new QA activities but rather brings onto the picture a repetitive pattern – the testing stages are repeated several times with the increased scope. From QA perspective it requires grouping all test cases into sets that will be re-executed iteratively per the number of required test runs.

2.2.4 Agile Model

Recently there has been a surge in the popularity of the agile development cycles. While there are many variations, they are based on similar principals. The twelve agile principles (Beck *et al*, 2001) are:

1. Customer satisfaction by rapid delivery of useful software;
2. Welcome changing requirements, even late in development;
3. Working software is delivered frequently (weeks rather than months);
4. Working software is the principal measure of progress;
5. Sustainable development, able to maintain a constant pace;
6. Close, daily co-operation between business people and developers;
7. Face-to-face conversation is the best form of communication (co-location);
8. Projects are built around motivated individuals, who should be trusted;
9. Continuous attention to technical excellence and good design;
10. Simplicity;
11. Self-organizing teams;
12. Regular adaptation to changing circumstances.

In contrast to the specification driven models (i.e. Waterfall and V-model), the agile models rely on face-to-face communication within the project. The agile project is developed in small iterations by a team of 5-9 people, each iteration usually lasting for two to four weeks. The planning and documenting is done by stakeholders when required. Agile development sets the focus to working software as the primary metrics for progress. This method is considered best suited for smaller projects which have frequently changing requirements. It can still be applied to larger extent where several teams would collaborate together.

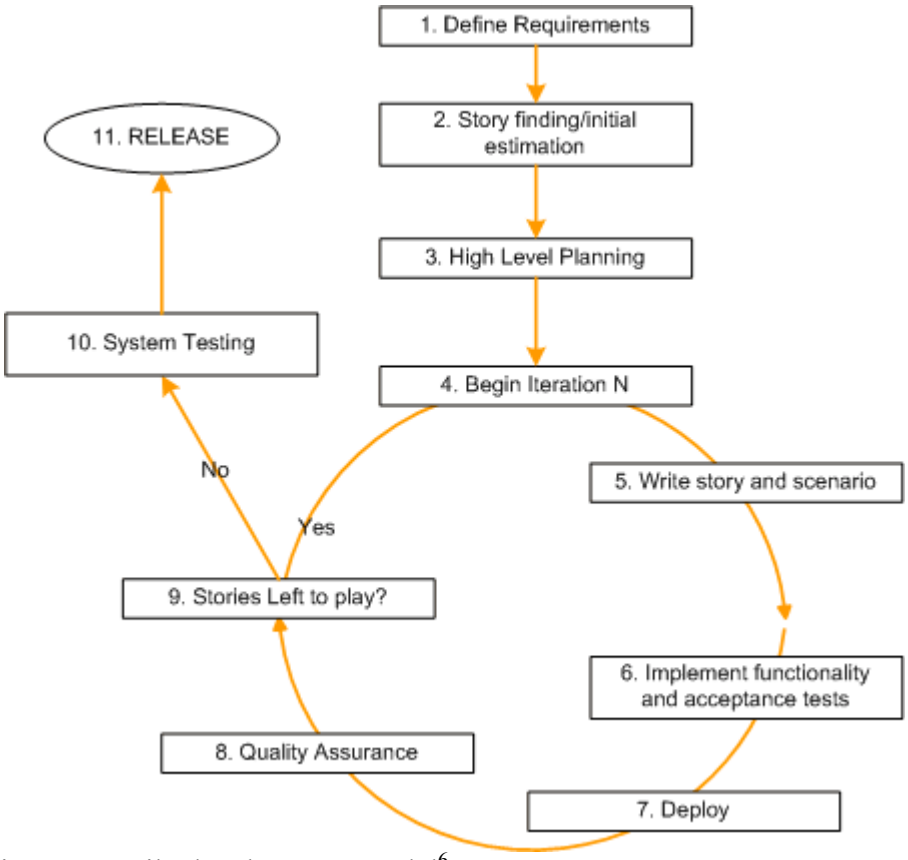

Figure 4. Agile development model[6]

Agile model introduces a new approach to planning the QA processes. The tests are planned and executed iteratively. Faster releasing of the product also means that the test management must support versioning to track, which tests were performed for given release.

2.2.5 Other Software Development Models

The list of software development models does not end here and can be extended further. Other approaches include chaos model (Raccoon, 1995), dual V-model (Clark 2009), extreme programming (Grenning, 2002) and scrum (Schwaber *et al*, 2011). However, majority of QA activities and related test artifacts have been in our overview for Waterfall, V-, Spiral and Agile models.

[6] Retrieved February 12, 2012, from http://davidhayden.com/blog/dave/archive/2005/02/14/828.aspx

2.3 Testing Levels

Testing is often grouped by the level and purpose of the tests. Typically these are separated to unit, integration, system, regression and acceptance testing.

Unit testing (a.k.a. component testing) treats each component as a stand-alone entity which does not need other components during the testing process (Sommerville, 1993). These tests are typically run by developers as they work with the code. The tests are performed on a developed module and often it verifies whether the function performs as expected in a set of conditions, including some corner cases. The input for this level is the code. The developer will commit the source code into the repository if the unit tests pass.

Integration testing seeks to verify the interfaces between components against a software design. The objective is to detect failures in the interactions between the components or subsystems (Pohl, 2010). An example would be web-application interface testing where some form is required to store new information in a database. The V-model describes that the integration testing inputs are coming from the detailed design. Based on this artifact, the test cases are created and run. These activities would be performed by QA test engineers (a.k.a. testers). Depending on the project expectations, the QA team may or may not produce test reports based on the achieved results. The QA test engineer reports the found defects to the project management (usually through a defect management system).

System testing investigates how a complete, integrated system complies with its requirements (IEEE, 1990). While integration testing has a focus on just a couple of modules, the system tests encompass all of the system. The test cases are designed based on the software requirement specification. Test team roles and activities are the QA test engineer who runs the tests, QA analyst who devises the test scenarios and QA team lead that will create a report based on the results. For system testing test-harnesses and automated test scripts may be used if applicable.

Regression testing ensures that the addition of new functionality and/or removal of program faults do not negatively impact the correctness of the program under test (Tucker, 2004). The set of test cases required to be run in this stage is usually determined by the QA team lead or QA analyst. Typically re-usable test cases are stored in a test library and used as required for the regression testing. Depending on the project stage, the depth of the regression testing can range from extensive (if major part of functionality was changed or affected) to minor (if only positive case scenarios require re-testing). Depending on the project automated testing can be used in this stage.

Acceptance testing (a.k.a. user acceptance testing) aims at checking whether the services on which the client and customer previously agreed are provided (Pohl, 2010). It is one of the final stages of a project and often occurs before a client or customer accepts the new system. Users of the system perform these tests, which are derived from the client's contracts or the user requirements specification. It is preferred that the end users perform the tests or in some cases the QA team assists the user by providing an acceptance test design. The focus of the acceptance testing is to test the business processes. The results of these tests indicate to the customer how well the system would perform in production.

2.4 Non-functional Tests

Non-functional testing addresses the qualities of the software that are not determined by the functional behavior of the product. These factors include but are not limited to performance, stability, security, compatibility and usability. Typically the performance, stability and compatibility tests are performed in the earlier stages of project. The usability testing is

carried out on the last stages of testing when the software is stable and mature, followed by regression tests.

Performance testing is conducted to evaluate the compliance of a system or component with specified performance requirements (IEEE, 1990). These tests are run by using some test-harness or by performing a mass-test (e.g. using hundreds of concurrent users or measuring the average response time of a script). Performance tests can also serve to validate and verify other quality attributes of the system, such as scalability, reliability and resource usage. The performance is measured against key performance indicators (KPI) that are defined per project requirements. The KPIs can be average response time, number of concurrent players while server load meets pre-defined values, system up-time with pre-defined network activity or other condition. The output of these tests is a report, typically offering recommendations for the planned system usage.

Load (stability) testing checks to see if the software can continuously function well in or above an acceptable period. Example scenarios involve assigning thousands of concurrent users to a system to determine the breaking point. As with performance testing, load testing is usually performed by using some test-harness.

Security testing is a process to determine that software protects data and maintains functionality as intended[7]. The criteria for the validations are:

- confidentiality which targets to verify that the information exchanged by the system and user can not be intercepted by malign third parties;
- authentication which confirms the identity of a person or the system;
- authorization, the process of determining whether the requester is allowed to perform the actions;
- non-repudiation which addresses any disputes whether an action took place. For example it includes time-stamping any interchange of authentication.

Compatibility testing is testing conducted on the application to evaluate the application's compatibility with the computing environment. It is a process where the same product is tested against different systems, e.g. different operating systems or web-browsers. The process in most cases is automated as the number of variations against which systems to test is usually large.

Usability is a complex of aspects like ease of use, ease of learning, ease of recall after a period of not using a system, affection, help, likeability, etc (Vliet, 2000). *The usability testing* addresses all of these aspects. While the listed attributes are hard to measure and are subjective to the test executor, evaluating them is typically done with the aid of a checklist:

- Use a simple and natural dialog;
- Speak the user's language;
- Be consistent;
- Give good error messages;
- Etc.

2.5 Test Automation

Typically manual testing is performed for software projects. But with growing maturity of the product, the scope of regression tests also grows. The greater the number of tests to run, the more time or resources is required to be spent. One of the solutions to remedy this problem is automation of tests. *Test automation* means use of software to control the

[7] Retrieved May 11, 2012, from http://en.wikipedia.org/wiki/Security_testing

execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions (Kolawa, 2007).

Test automation can be used in unit testing and in regression testing. This is typically performed by QA teams as they maintain the test libraries with regression test cases.

2.6 Summary of the Testing Literature

In this chapter we have examined different software development models (i.e. Waterfall, V-model, Spiral and Agile models) and their testing levels. Our purpose was to identify QA activities and artifacts, which could be required in a test management tool.

Table 1. QA activities (N/A – not available)

Approaches Software development stages	Waterfall	V-model	Spiral-model	Agile model
User requirements' definition	N/A	▪ Requirements inspection	▪ Requirements inspection	▪ Requirements inspection
Software Requirements Specification	N/A	▪ Test planning ▪ QA resource allocation	▪ Test planning ▪ QA resource allocation	N/A
Software design	N/A	▪ Test design creation	▪ Test design creation	▪ Test design creation
Integration and verification	▪ Test planning ▪ Test execution ▪ Reporting	▪ Test execution ▪ Reporting	▪ Iterative cycles ▪ Test execution ▪ Reporting	▪ Iterative cycles ▪ Test execution ▪ Reporting
Maintenance and operation	N/A	N/A	▪ Post launch defect fix testing	▪ Post launch defect fix testing

Our observations are summarized in Table 1. The content of the table identifies QA activities used in each particular combination. The requirements' inspection activities performed in the user requirement gathering and evaluating stage are present in all except the waterfall model. The same could be said for the test design creation which is performed in software design stage. Test planning and resource allocation, which is performed in parallel with software requirements specification, is carried out only in V and Spiral model. In the Agile model, this is done only if required by stakeholder. Test execution is carried out in all of the models during the integration and verification stage, within the Spiral and Agile model it is done iteratively. The only models that describe QA activities after the launch of the software are Spiral and Agile, where it is referred as iterative cycles for improving the existing product (Tucker, 2004).

The QA activities result in artifacts, summarized in Table 2. Similar to the previous table, on the horizontal axes software development models are listed. Software development stages are brought out vertically.

Table 2. QA artifacts

Approaches Software development stages	Waterfall	V-model	Spiral-model	Agile model
User requirements' definition	N/A	▪ Requirements review report	▪ Requirements review report	▪ Requirements review report
Software Requirement Specification	N/A	▪ Test Plan	▪ Test Plan	N/A
Software design	N/A	▪ Test Design	▪ Test Design	▪ Test Design
Integration and verification	▪ Test Plan ▪ Test Design ▪ Test report ▪ Defect ▪ Defect report	▪ Test report ▪ Defect ▪ Defect report	▪ Test report ▪ Defect ▪ Defect report	▪ Test report ▪ Defect ▪ Defect report
Maintenance and operation	N/A	N/A	▪ Test report	▪ Test report

In the user requirements definition stage the result is a review report which is present in all except the waterfall model. In the same models in the software design stage, test design is created. As expected, the test plans are created during the software requirement specification stage though waterfall and agile models have a slight variance. In agile model, the test plans are created only when stakeholders require it. In case of the waterfall model, all artifacts are created in the integration and verification stage. The artifacts produced during the verification stage for all models are test reports, defects and defect reports. Last stage of the software development cycle, maintenance stage, brings new entities only with spiral and agile models as they represent the iterative nature of providing new versions to existing software.

Comparing the Tables 1 and 2, we see a direct correlation that for every activity there is at least one artifact. For example the output of test design creation is test design (a.k.a. test cases), reporting activity provides reports. There exists also the opposite relation – if there exists an artifact, then it was produced from an activity.

The summary and comparison of the software QA activities and artifacts in different software life cycles will provide basis for evaluating various supporting test management tools. In the following chapter we will have a look on the current market support.

Chapter 3 – Market Support for Quality Assurance Activities

In this chapter we will examine tools that offer test management capabilities. Firstly we will make an overview of their functionalities. Thus we will consider how well these tools are suited to support testing activities and produce test artifacts that were surveyed in Chapter 2.

3.1 Overview of Tool Support for Software Quality Assurance

There are different tools[8] which *all* claim to support *all* activities of the QA team. Analyzing *all* of them would not be prudent, thus, based on the recommendations from tool users[9] we select 4 tools for this overview: TestLink, TestRail, HP Quality Centre and QA Complete.

3.1.1 TestLink

TestLink is a test management tool that is under the GPL license, which is free-ware. The tool is developed and maintained by open community consisting of testers and other people holding QA management positions in various companies. The tool has web-based interfaces developed in PHP and background database MySQL.

TestLink supports a typical software project life cycle by allowing to create requirements, test plans, test cases and grouping them by release. The activities covered with the TestLink are test planning, maintaining references between requirements and test plans, test execution and reporting of the testing progress. There exists also support for generating reports both for test results and test coverage. While the basic activities are covered, the tool lacks a lot of flexibility and does not support integration with some of the more common defect management tools.

3.1.2 TestRail

TestRail is web-based application developed by Gurock Software. The group is providing support by releasing new versions with improved functionality on bi-yearly bases.

TestRail has full support for the test design and execution stages. It also includes such useful tools as aligning QA activities with project milestones and integration with various independent defect management tools (e.g. Jira, Bugzilla, Mantis etc). Dashboards reflecting current test progress are also one of the strongest features of the tool. However, TestRail is more focused on managing the test execution stages, thus it lacks the support for requirements and test plan management.

3.1.3 HP Quality Centre

Quality Centre (QC) is web-based test management tool by Hewlett-Packard. The client is usable only by Windows and Internet Explorer users which differentiates itself from other test management products.

QC contains four major section that support QA activities: Requirements for software requirement tracking, Test Plan for test design creation, Test Lab for test execution and Defects for defect management. Any artifact created can be linked to the various sections, thus supporting traceability. QC comes with various possibilities of generating reports to reflect number of planned test cases, test execution progress, defect burn-down by severity

[8] Retrieved May 11, 2012, from http://testingsoftware.blogspot.com/2007/08/list-of-test-case-management-tools.html

[9] Retrieved May 11, 2012, from http://www.softwaretestingclub.com/forum/topics/looking-for-a-new-test-case

and customized reports. Despite its numerous features, the tool has some short-comings: with larger projects the performance of the tool degrades. Also with the release of new Internet Explorer versions and new Windows OS, compatibility issues arise.

3.1.4 QA Complete

QA Complete is web-based test management tool by SmartBear Software. The TMT is providing their software for many of the international companies, among them Sony, Skype, Google and Adobe.

QA Complete supports creating and tracking requirement coverage, test design and execution with built-in defect management system. The defect management can also be integrated with external tools. QA Complete comes with automated testing tool and can collaborate with AutomatedQA, TestComplete and HP QuickTest Pro. Unfortunately the test planning activities were not supported by the tool.

3.2 Sample Project Template

Each of the TMTs has both its strengths and shortcomings. To compare and analyze them, we need to apply the same criteria for measurement. To achieve this, a project template is applied and all QA stages and activities are implemented within it. Next we will introduce the template we used.

The sample project requires new type of online casino games to be integrated to an online poker client. The current client already supports some casino games, but the new games will come with new functionality. This requirement is coming from a customer who has created a business requirement document which in turn will be analyzed and a software requirement specification will be created. Furthermore, detailed design document will be provided and eventually the functionality will be coded. From the QA processes this will be supported by software test plan, test design and finally test execution. During the test execution stages, defects need to be reported and at the end test report will be created based on the statistics of the execution stage. The QA manager will require reporting testing progress during the project, so reports for test coverage of the requirement, planned number of tests, progress and report containing number of tests executed per stage must be supported.

The project is for an existing product, thus there will be different stages of QA activities: functional testing, regression testing and acceptance testing. The TMT should support dividing the activities per stages. Grouping the tests per versions is also required since the regression testing will cover the functionality of previous product version.

The new development might have an impact on the performance of the existing client, thus non-functional tests (e.g. performance, security, usability etc) are also required.

It is possible to make the project more complicated but current level on complexity is sufficient for the thesis and to make the comparison of existing TMTs.

3.3 Existing Support for Testing Activities

Testing activities identified in Chapter 2 can be identified in the listed TMTs. Table 3 summarizes testing activities supported by overviewed tools.

The centre part of all tools is test case creation and management. A lot of thought has been put into how to link the tests to requirements thus providing the linkage support for requirement coverage analysis. Another central part of the tools is the test execution activity. All except freeware tool (TestLink) offer defect reporting capabilities, QA Complete and HP QC even support integration with external defect management tools.

Table 3. TMT support for QA activities

TMT Software QA stages	Test Link	TestRail	HP Quality Centre	QA Complete
Requirements' analysis	▪ Creation of requirements ▪ Linkage of requirements to test plans ▪ Requirements' review	N/A	▪ Creation of requirements ▪ Requirement division per versions ▪ Linkage of requirements to test plans ▪ Requirements' review	▪ Creation of requirements ▪ Requirement division per versions ▪ Linkage of requirements to test cases ▪ Requirements' review
Test plan preparation	▪ Creation of test plans (suites of test-cases) ▪ Linkage of test plans with requirements and test cases ▪ Not possible to use previously created test cases (except export/ import)	▪ Setting of project milestones for QA activities	▪ Test cases are managed as library and structured into folders ▪ Creation of test-set folder structure (simulates test planning)	N/A
Test design stage	▪ Creation of test cases ▪ Linkage of test cases with test plans	▪ Creation of test suits (cases) ▪ Linkage of test cases with external requirements (via hyperlinks) ▪ Searching for existing test design and suites from previous versions	▪ Creation of test cases ▪ Linkage of test cases with requirements ▪ Test case reviews ▪ Test run preparations and resource assignment ▪ Searching for existing test design from previous versions	▪ Creation of test cases ▪ Linkage of test cases with requirements ▪ Test case reviews ▪ Test run preparations and resource assignment ▪ Searching for existing test design from previous versions
Test execution	▪ Test execution of created test cases ▪ Test execution is possible only on created builds	▪ Test execution of created test runs ▪ Defect reporting	▪ Test execution of created test runs ▪ Defect reporting ▪ Automated testing is supported by QTP ▪ Grouping of tests per testing stage	▪ Test execution of created test runs ▪ Defect reporting ▪ Grouping of tests per testing stage
Revision and reporting	▪ No defect management ▪ Test results driven reporting ▪ Test coverage driven reporting	▪ Test coverage driven reporting ▪ Built-in defect management system	▪ Requirement coverage driven reporting ▪ Test results driven reporting ▪ Built-in defect management system	▪ Requirement coverage driven reporting ▪ Test results driven reporting ▪ Built-in defect management system

The last stage in software QA is revision and reporting. This is supported by all tools, but it is offered differently. TestLink and TestRail offer test coverage based reporting, i.e. how many of the planned test cases have been executed. TestLink, HP QC and QA Complete offer also test results driven reporting which shows the testing results grouped by test run status (i.e. Failed, Passed, Blocked, No run, etc). HP QC and QA Complete support additionally requirement coverage reports which present to what extent a requirement has been tested.

3.4 Existing Support for Testing Artifacts

Testing artifacts identified in Chapter 2 can be identified in the listed TMTs. Following Table 4 summarizes test artifacts produced by the tools.

Table 4. TMT support for QA artifacts

TMT Software QA stages	Test Link	TestRail	HP Quality Centre	QA Complete
Requirements' analysis	▪ Software requirements	N/A	▪ Software requirements ▪ Requirement coverage report	▪ Software requirements ▪ Requirement coverage report
Test plan preparation	▪ Test suites (grouped sets of test cases)	▪ Milestones for QA activities	▪ Test case library ▪ Test case folders representing different test stages	N/A
Test design stage	▪ Test cases	▪ Test cases ▪ Test suites with assigned resources (a set of test cases prepared for execution)	▪ Test cases ▪ Test runs with assigned resources (a set of test cases prepared for execution)	▪ Test cases ▪ Test runs with assigned resources (a set of test cases prepared for execution) ▪ Test case change history
Test execution	▪ Test run with status	▪ Test run with status ▪ Defects	▪ Test run with status ▪ Defects	▪ Test run with status ▪ Defects
Revision and reporting	▪ Test report	▪ Test report	▪ Test report ▪ Printable test report ▪ Generated reports based on test execution progress	▪ Test report ▪ Generated reports based on test execution progress

The biggest effort with all tools has been put to test design artifacts: test cases. All products support various activities with the test cases, starting with searching and ending with editing them. Formulating the test cases into a test run is handled differently but in the essence, it works in a similar manner for all tools. QA Complete takes test case management one step further by providing also history of the changes done to the test cases. Finally for the last stage, reporting, the format and content of the test report is different among the four TMTs. Despite these differences, all of them support reports.

Analysis of Tables 3 and 4 reveals that for any activity which is supported by TMT, there is an output in the form of an artifact. Similarly we observe that if there exists an artifact in a

TMT, then it was produced by an activity of the TMT. We will use this correlation later in our survey and try to confirm that similar connection can be identified in the test managers' expectations for a TMT.

3.5 Summary of Existing Test Management Tools

The market offers different test management tools. There are freeware tools (e.g. TestLink, Bugzilla) but mostly, TMTs are commercial products (e.g. TestRail, QA Complete, HP Quality Centre). The support to various activities and needed artifacts differs from vendor to vendor. When comparing the solutions offered to the QA activities described in software development models, then there are gaps. Yet still, the tools are being used by IT oriented companies[10,11,12], thus we believe not all QA activities and artifacts listed in Chapter 2 are required for all companies.

Chapters 2 and 3 have provided insights into both theoretical and practical part what a test management tool should support and what is offered in practice. The short list of analyzed test management tools can question the completeness and reliability of the results; however this is mitigated by broad spectrum of their coverage. If we look at all the features of the analyzed TMTs, then all activities and artifacts of QA processes from Chapter 2 are listed. This information will be the basis for a research that will target to form test management tool acceptance criteria.

[10] Retrieved May 11, 2012, from http://www.getzephyr.com/customers/zephyr_testimonials.php
[11] Retrieved May 11, 2012, from http://smartbear.com/customers/
[12] Retrieved May 11, 2012, from http://www.techvalidate.com/portals/hp-quality-center-110

Chapter 4 – Overview of Existing Tool Evaluation Approaches

This chapter gives an overview of existing approaches to tool evaluation. We will describe the general approach (Finkelstein *et al*, 1996) to tool evaluation and introduce commercial off-the-shelf (COTS) tool evaluation approaches (e.g. OTSO, WinWin). Finally, we will provide the motivation why a TMT evaluation framework is needed and how it will be benefitial to testers.

4.1 General Tool Selection Process

General tool selection process (Finkelstein *et al*, 1996), shown on Figure 5, consists of 4 steps: user requirements specification, understanding of the available tools, evaluation of the tool compatibility with the set requirements, and selection of the „best“ tool (Matulevičius, 2009). *Requirements specification* is based on the area of expertise knowledge and on any existing tools. When new requirements are identified, they are revised based on third party reports, tool descriptions provided by vendors, and personal testing experience. *Understanding of the available tools* involves taking into account the tool capabilities and mapping this to users' activities from the users' experience and vocabulary. Some of the identified tool features might be irrelevant for particular user, requirements not complied with the tool, or requirements achieved through the use of several tool features.

During *the assessment of the tool compatibility* the user has to measure the level of meeting the requirements by different tools. This requires some comparison between tools and requirements specification. The matching is typically based on functional and non-functional requirements, and considers relationships between different requirements. *Selection of the "best" available tool* depends on if the requirements are satisfied and also on the importance of these requirements. It might happen that requirements are not satisfied by any of the tools. Then the user reviews the requirements and iterates the selection or reorganizes his existing practices in order to fit the "best" tool.

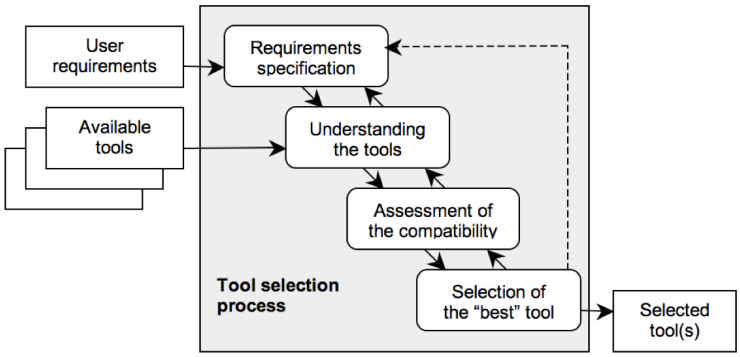

Figure 5. General tool selection process (adapted from Matulevičius, 2009)

This process is general enough for the analysis of the COTS tool selection approaches listed in section 4.2, performed by Matulevičius (2009). It supports the construction of prioritized criteria in the requirements specification phase. It manages measurement plans, user preferences, evaluation uncertainties and knowledge aggregation techniques (in the phases of *Understanding of the available tools* and *Assessment of the tool compatibility*). It helps evaluating and supports decision making in the *selection of the "best" available tools*. Finally, it supports flexibility and consistency. Users can iterate all phases as necessary and take into consideration different viewpoints when doing the evaluation.

4.2 Reviewing Commercial Off-The-Shelf Tool Evaluation Approaches

Matulevičius (2009) performed a study on 15 different COTS and requirements engineering (RE) tool evaluation approaches. We present the summarized study results in Table 5 where the different phases, activities, and steps performed during the COTS and/or RE-tool selection process are described.

Matulevičius' survey shows that, all COTS and/or RE-tool evaluation approaches closely follow the general tool selection process. However, one of the important findings of this review is the necessity to expand the tool selection process with additional selection phases. The listed evaluation approaches lack the support for the development of evaluation plan and selection process evaluation.

4.3 Motivation for Our Evaluation Framework

Firstly, the general tool selection process does not define *how* the requirements should be identified. It only states, they *should be* identified. This leaves room for subjectivity. The COTS tool evaluation processes don't explicitly give guidelines for TMT evaluation based on company characteristics with consideration to expectations for the tool from similar competing companies.

Secondly, the described evaluation approaches (e.g. PORE, STACE, PECA) do not provide a measurable value for the evaluation results. There are suggestions to select the tool that „best fit" or to „make a decision". The rules for making these calls are left free to interprete.

This thesis develops a TMT evaluation framework which addresses the following problems:

1. subjectivity of the evaluation;
2. evaluation processes do not consider IT companies' expectations to a TMT;
3. clear measurable value for the evaluation.

In the next chapters we gather Estonian IT companies' expectations to a TMT and create product diagrams based on company characteristics. Finally, we create a TMT evaluation framework which gives a clear measurable value of how the tool meets the users' company expectations.

Table 5. COTS and/or RE-tool evaluation approaches (adapted from Matulevičius, 2009)

ID	Approach	Preactivity	Requirements specification	Understanding the tools	Assessment of the compatibility	Selection of the "best" tool	Post-activity
1	**OTSO (Kontijo, 1996)**	-	Evaluation criteria definition	Search for COTS alternatives; Screen COTS alternatives	Evaluate COTS alternatives	Analyze the results	Deploy selected COTS; Assess the evaluation process
2	**IusWare (Morisio *et al*, 1997)**	Problem formulation	Design evaluation model		Apply evaluation criteria		-
			Define set of attributes, criteria, scale and rules	Choose tools for evaluation	Measure attributes and transform them into a comparable scale; Apply the aggregation techniques		-
3	**PRISM (Lichota *et al*, 1997)**	-	Sustainability criteria	Product identification	Product screening; Product stand alone application; Product integration test	-	-
					Field testing		
4	**CISD (Tran *et al*, 1997)**	-	Product identification		Product Evaluation	-	-
					Product integration / enhancement		
5	**PORE (Maiden *et al*, 1998)**	Start	Acquire Information	-	Analyze information	Make decision; Select product	Stop
6	**Stace (Kunda, 2003)**	-	Select COTS product; Define social-technical criteria for COTS components	Search for available products in market	Evaluate and select "best" COTS products		Report and store in the database
7	**CAP (Ochs *et al*, 2000)**	System design process	Initialization	Execution			System design/ Specification and/or supply process
			Reuse				
8	**CRE (Alves *et al*, 2001)**	-	Identification; Description	Evaluation	Acceptance	-	-
9	**CARE (Chung *et al*, 2004)**	-	Define goals	Match goals, Rank components		Select a set of COTS candidates	-
			Negotiate changes to COTS and system under development				
10	**PECA (Commella-Dorda *et al*, 2001)**	Plan the evaluation	Establish criteria	-	Collect data	Analyze data	-
11	**CEP (Phillips *et al*, 2002)**	Scope evaluation effort	Define evaluation criteria	Search and screen candidate components	Evaluate component alternatives	Analyze evaluation results	Preserve evaluation data
12	**Story-board (Gregor *et al*, 2002)**	Develop a plan	Requirements identification; Development of use cases	Identification of COTS (assess functional requirements and APIs)			
				Deployment of storyboard (assess according to the working scenarios)			
13	**Quality Model-based Selection (Carvallo *et al*, 2005)**	-	Quality model construction; Definition of requirements	Evaluation of products		Selection of products	-
14	**WinWin (Boehm *et al*, 2003)**	-	Identify objectives, constraints and priorities	Identify alternatives (candidate COTS products)	Asses candidate COTS products	If multiple COTS cover objectives, coordinate coding effort; Tailor COTS products	Productize transition application
			Adjust constraints and practices				
15	**R-TEA (Matulevičius *et al*, 2006)**	-	Preparation of a requirements specification	Selection of the business parties	Investigation of functional, process and product requirements	Decision about the RE-tools	-

Part 2 - Data Collection and Processing

Chapter 5 – Survey of the Testing Activities Within Software Development Companies

In this chapter we will report on the survey done within the Estonian software development companies. First we introduce the survey objectives, research questions, and research method. Then we present the questionnaire, survey environment, and target audience. Next we discuss data collection process, result analysis method, and threats to validity. Finally the chapter completes with results and discussions.

5.1 Objective and Goal of the Survey

The goal of the survey is to understand software testers' needs for the TMT and, based on that, to develop an evaluation framework for the TMT acquisition. The survey questionnaire addresses four major questions:

1. Which software development model the TMT should support;
2. Which TMT functions are important for a software developing company;
3. Which TMT artifacts are important for a software developing company;
4. How are the TMT functions and artifacts related to the characteristics of a software developing company?

The first question investigates correlations between development models and different expectations to the TMT. This hypothesis is derived from the different emphasis on the testing artifacts identified in Table 2. The second and third questions address the importance of functions and artifacts identified in Chapters 2 and 3. The answer to these questions contributes to the evaluation framework. The fourth question provides the background of the responder's company and is used to find any correlations between company characteristics and the previous responses.

5.2 Research Questions and Research Method

The research method consisting both of exploratory and descriptive analysis, is shown in Figures 6a and 6b.

During the *exploratory research* we formulate research questions, define survey constructs, and validate the constructed questionnaire. We have defined our questionnaire according to the review of the literature on software quality assurance described in Chapter 2 and according to the tool survey given in Chapter 3.

The literature review was done on the software development models. The output of the activity was a matrix of artifacts and activities produced in the software quality assurance. We have looked into four different TMTs. By using a sample software project, we identified the artifacts and activities supported by each of them. This allowed us to create another matrix containing the current TMT capabilities. With the combined information from the two data matrixes, we have formulated a web-based questionnaire consisting of 57 questions.

To validate the questionnaire we have contacted three QA specialists who were asked to evaluate the questionnaire and provide us a constructive feedback. Their responses revealed that the questionnaire should not be long, better grouped and the terminology used required explanation. Also questions falling to similar category were advised to be presented in a table format. As a result, the final survey contains 31 questions, as illustrated in Appendix A.

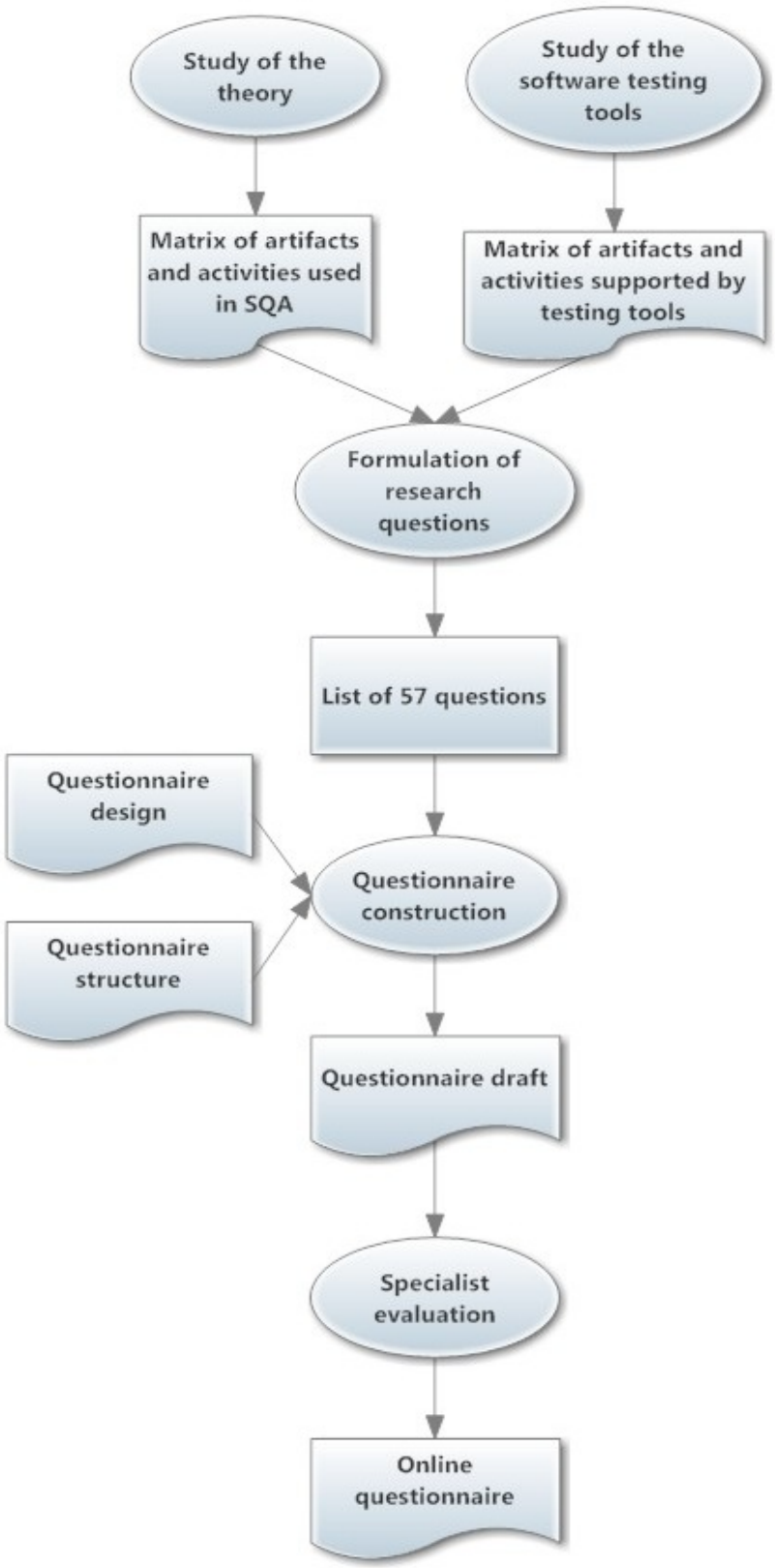

Figure 6a. Exploratory research

During the *descriptive research* we have collected data about the TMTs, analyzed it and reported the results. Detailed description of each activity will be presented in this chapter.

The list of survey respondents was composed by combining both Estonian[13] and Lithuanian[14] IT companies. The participants were, first, contacted by phone and their willingness to participate was confirmed. Then the URL to questionnaire was delivered to them by email. After a 6 week response period, all survey results were gathered.

The survey results were analyzed using visualizing the trends and determining any outliers (Wohlin *et al*, 2000). For each outlier we analyzed whether data reduction can be applied based on the characteristics of the other responses. Finally the results are interpreted and used to form the evaluation criteria for the TMT assessment framework.

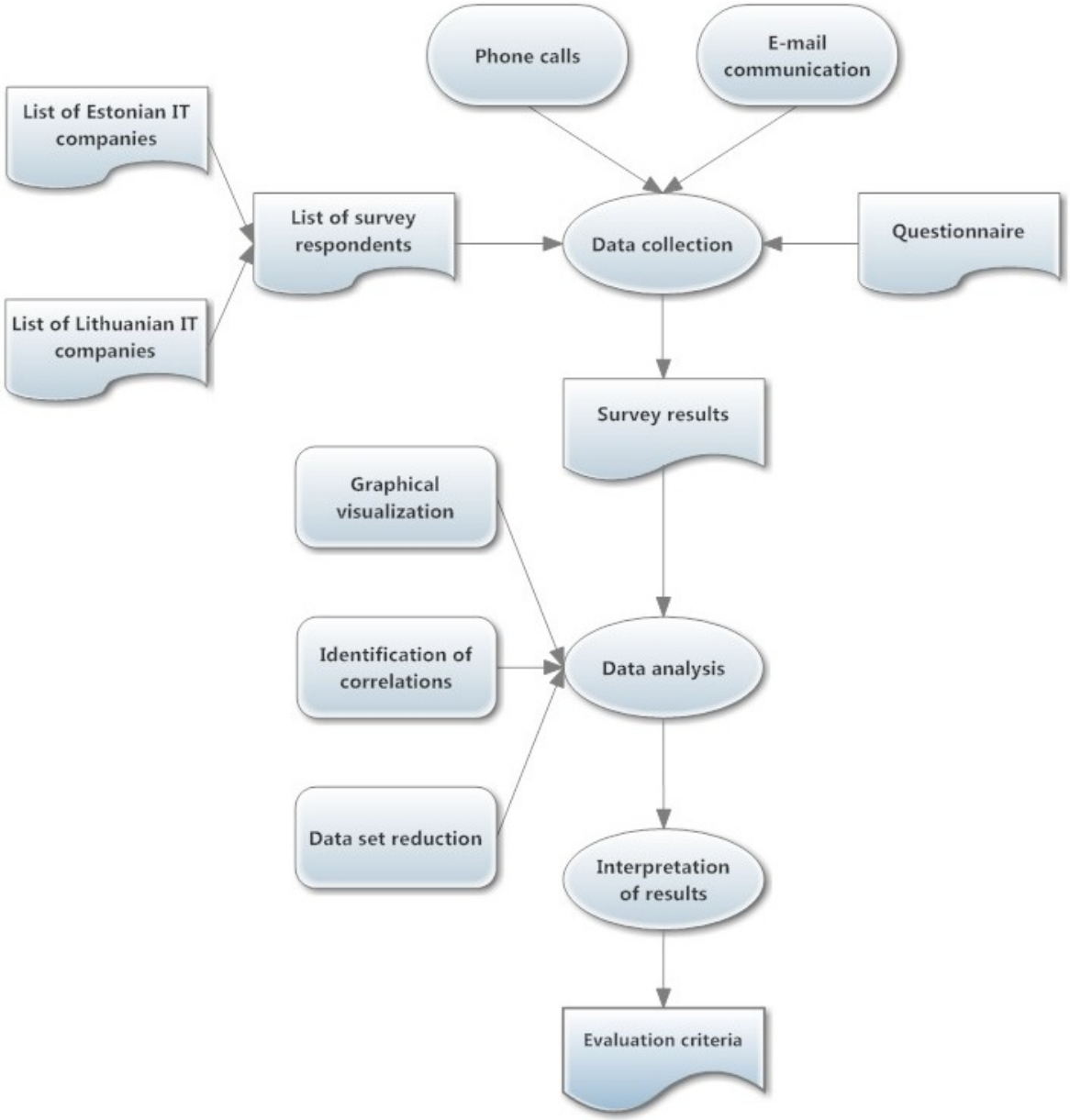

Figure 6b. Descriptive research

[13] Retrieved May 11, 2012, from http://www.itl.ee/?op=body&id=33

[14] Retrieved May 11, 2012, from http://infobalt.lt/sl/index_en.php?t=members

5.3 Questionnaire

The questionnaire, attached as Appendix A, begins with an *introduction* section describing the purpose and goal of the research. It also provides the expected duration for responding. The last part of the introduction states the target audience and also gives information where to find explanations for the used terminology. The TMT and software quality assurance definitions provided for the survey maintain the unified understanding.

Through-out the survey, the evaluation scheme from '0' to '5' is introduced ('0' – I don't know, '5' – Completely agree). This approach is used so that users would be left with a neutral answer ('3' – Hard to say) while still having the opportunity to completely agree or disagree with the provided statements.

The most difficult problem during surveying is to motive the respondents to answer an unsolicited questionnaire. People are motivated if they can see that the study results are likely to be useful to them. For that reason the questionnaire is accompanied with several key pieces of information: what the study purpose is, why each individual's participation is important and how confidentiality will be preserved. Each respondent is given an opportunity to provide their contacts so that the summarized survey results could be shared with them.

The first part of the questionnaire asks how important the TMT support for software development models is. 4 models are given: Waterfall, V-Model, Spiral and Agile. The fifth option is a free field in case for some other model.

The second part enquires about the various testing activities (such as test planning, design, execution, defect reporting). The respondents are also provided with an open-field answer in case not all activities are listed. The questions and answers are presented in table format, with the „I don't know" option separated in the last column. The terminology has definitions which could be seen by moving the mouse over the text and reading the tool-tip.

Next, in *part three***,** the survey questions consider TMT expected support for artifacts produced in software quality assurance. The same format as in the second part for the questionnaire is used.

The last part targets the information about the respondents and their organization. This data is used to find any correlations between similar company characteristics and their responses. The information includes the TMT the organization is currently using, employee number in organization and in QA department, and information (i.e. domestic or international) on market type. This section also contains open comments field for the respondents to provide feedback in free form regarding the survey.

5.4 Environment Description

The survey was carried out mainly in Estonian IT companies which provide software development services and products. For future research, similar survey could be performed in other European countries. To compare the results with neighboring region IT companies, the same survey was sent to Lithuanian IT companies.

Estonian ICT industry[15], based on 2010 statistics, consists of 1748 companies. In 2010 the number of the companies of ICT sector increased 6,26% in comparison with 2009.

In 2010 the number of employees working in such companies was 15 585.

Small companies (having less than 9 employees) dominated (89 percent) in ICT sector. About 16,7 percent of all the employees of this sector worked in small companies. Middle-sized and big companies with 50 and more employees made 1,5 percent of ICT companies; 67,4 percent of all the employees of this sector worked in such companies.

[15] Retrieved May 11, 2012, from http://pub.stat.ee

In 2010 the income of this sector's companies made 220,39 million EUR.

Lithuanian ICT industry[16], based on 2008 statistics, is the biggest in Baltic States. In 2008 there were 1777 companies in ICT sector. In 2008 the number of the companies of ICT sector increased 9% in comparison with 2007 and made 2,8 percent of all nonfinancial companies.

In 2008 the number of employees working in such companies was 24 913. In comparison with 2007 this number significantly increased by 7,1 percent.

Small companies (having less than 9 employees) dominated (82 percent) in ICT sector. About 18 percent of all the employees of this sector worked in small companies. Middle-sized and big companies with 50 and more employees made 5 percent of ICT companies; 64 percent of all the employees of this sector worked in such companies.

In 2007 the income of this sector's companies made 2205,17 million EUR.

5.5 Data Collection Process

A self-administered data collection method (Dillman, 2000) was used. The questionnaire was sent out to 30 Estonian and 55 Lithuanian testers at software companies.

In most IT companies the TMT is used on daily basis by the quality assurance teams. Informal inquiries and discussions with testing specialists confirm that more than 80% of the working hours are spent by using TMT in some form. Since the choice for the tool falls for the person responsible for software quality, we decided to target these people. The test engineers (also known as testers) sometimes lack the deeper knowledge of the software development processes. This was the main reason for excluding them from the target group. When companies were contacted, they were asked to forward the questionnaire to either QA manager, QA team leads or in the absence of the position, to project managers who were responsible for project quality.

First, the *respondents from Estonia* were contacted by phone. The purpose of the study was explained to all individual participants. To motive participants, we have promised to send them study results. If the participant agreed to take part in the study, the questionnaire URL was sent to him or her by e-mail. Majority of the contacted IT companies replied positively and agreed to participate in the survey.

For *Lithuanian respondents*, the first step was skipped and questionnaire URL was sent directly by e-mail. The responses from Lithuanian IT companies were scarce so they were contacted several times via e-mails. In the end, personal contacts from Lithuania were used to receive results for the survey. There were only 4 responses from Lithuanian companies. Their results are presented independently and are not considered when we create the TMT requirements framework.

During the data collection a certain trend could be noticed: establishing direct contact over phone or in person with the respondents resulted in greater chance of receiving their feedback.

5.6 Data Analysis Method

To analyze data collected during the survey we have applied descriptive statistics method (Wohlin *et al*, 2000). The data is grouped similarly as the questionnaire into 4 sections: TMT expected compatibility with different software development models, TMT supported functionality, TMT supported artifacts, and respondent background.

The survey responses were submitted electronically. After receiving last answers, the survey was closed and all existing data was imported into tabular format (Appendix B). This representation was chosen since it is considered to be best (Good *et al*, 2008) when the

[16] Retrieved May 11, 2012, from http://infobalt.lt/sl/index_en.php?t=lietuva

questions can have three to five possible outcomes. The matrix columns list survey questions and each respondent, and their answers, are written on each row. The question for respondent contact email was left out of the matrix, as it has no added value to the data analysis. This results in a 15x30 cell matrix that lists all responses from Estonian IT companies.

The tabulated form is further processed and each section of questions is presented in a histogram. All histograms were created by excluding non-answered responses or where the respondent did not know the answer. Since there were only a few of these, we do not consider them to invalidate our research. For each data set, we also calculated the mean, median, mode and standard deviation to provide further insight to the responses (Noether, 1990).

Finally the trends based on company background are analyzed. The results are visualized again on histograms, grouped by company size, QA team size, used development method and the target market of the company products.

5.7 Threats to Validity of the Survey

There are very few studies that are completely unbiased. With our survey we strive to exclude threats to validity as well as possible within the scope of the thesis. Good *et al* (2008) reports that the bias is caused from sample selection, study conduct, and results interpretation. In this section we will analyze what are the limitations posing threat to the validity.

The first and probably the biggest threat, is the number of participants in the survey. 15 respondents is a small subset of all IT companies in Estonia. However the initial 30 companies contacted for the survey all belonged to the top software development companies operating in Estonia. By focusing on the quality of the respondents and not on the quantity, we have mitigated some of the risk.

Another aspect which should be mentioned is that the survey was carried out among Estonian companies. Applying the results of the survey outside Estonia could have different effect. Neyer *et al* (2008) finds that the work-culture in different countries has differences. For example in Nordic countries (Norway, Sweden, Finland) the work-culture differs from Southern European countries (Greece, Bulgaria, Italy).

In all surveys there is always the risk of human error with submitting responses. While it is not completely avoidable, we mitigate the risk by determining the outlying answers and excluding them from the data set.

Finally, a threat to the validity of the survey comes from the interpretation of the results. Good *et al* (2008) suggests that collaboration between the statistician and the domain expert is essential if all sources of bias are to be detected and corrected. Accordingly, we perform a case study on the survey results and necessary corrections are applied to the interpretation.

5.8 Results

The questionnaire was available for the participants to respond for a period of 6 weeks. When the active period was over, the data was gathered and imported to tabular format. The complete list of responses can be seen from Appendix B.

The first three sections of the questionnaire asked the respondents to answer how much they agree to each of the statements. The mapping between their answers and the figures on the graphs below are following:

- 5 – Completely agree;
- 4 – Rather agree;
- 3 – Hard to say;
- 2 – Rather disagree;
- 1 – Completely disagree.

The respondents provided their evaluation of how important each attribute is for TMT. To process the data and use it for the framework, we set a threshold of 4,4. The number means, that more than every third respondent should „completely agree“ that TMT should support the attribute while no-one responds with lower than „rather agree“. This figure will be later used to highlight the most important attributes of TMT for particular focus group and it will be used in the TMT evaluation framework in Chapter 6.

5.8.1 Results Over All Respondents

The first section of the questionnaire asked which software development process TMT should support. Figure 7 illustrates the mean results based on all respondents from Estonia. The Agile model is perceived as a software development model that TMT should definitely support. The same could be said for the Spiral model. However the mean for V and Waterfall models indicates lower support. However the mod clearly indicates, that the most frequent answer was „Completely agree“ (Table 6). This will be further analyzed by grouping the respondents based on their background.

Table 6. TMT should be compatible with following software development models

	Waterfall model	V-model	Spiral models	Agile models
Mean	3,57	3,83	4	4,79
Standard deviation	1,84	1,94	1,78	1,3
Mod	5	4	4	5
Median	4	4	4	5

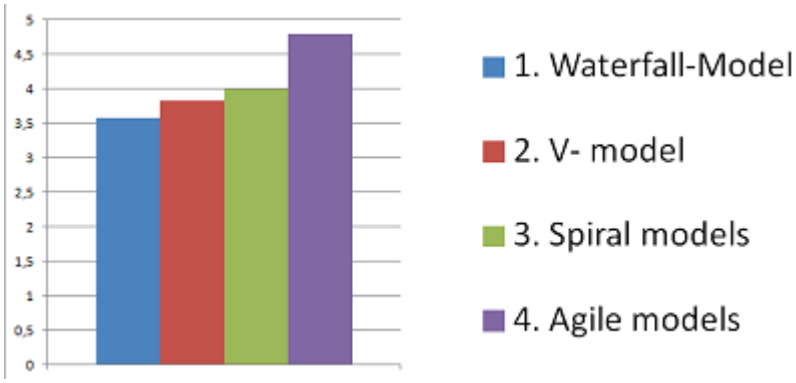

Figure 7. TMT should be compatible with following software development models (mean)

The second section of the questionnaire asked which testing functions should be implemented within the TMT. Interesting aspect that is worth highlighting is that the most valued aspects were test planning, reporting and traceability. While almost all functions were seen as required, two of them received noticeably lower values – software requirements specification review and defect reporting. These findings will be examined when data is analyzed based on grouping of respondent background. The statistics for the whole target group can be found from Table 7. Figure 8 shows the results based on the mean.

Table 7. Which testing functions should be implemented within the TMT?

	SRS re-view	Test planning	Test design	Test execu-tion	Requi-rement coverage reporting	Test execution progress reporting	Test execution results reporting	Defect reporting	Trace-ability
Mean	3,400	4,667	4	4,067	3,929	4,533	4,533	3,467	4,500
Standard deviation	1,242	0,488	1,134	1,163	0,997	0,834	0,834	1,246	1,373
Mod	2	5	5	5	5	5	5	3	5
Median	3	5	4	4	4	5	5	3	5

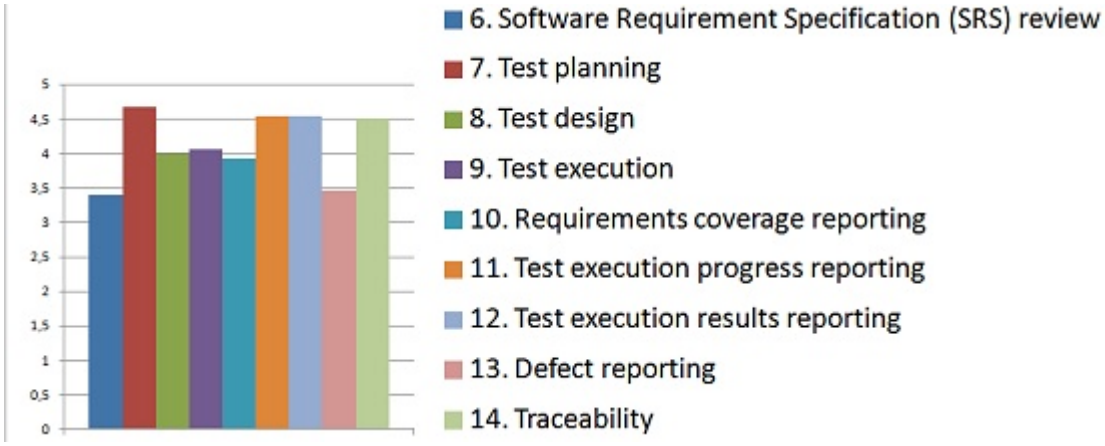

Figure 8. Which testing functions should be implemented within the TMT? (mean)

The third section of the questionnaire asked which artifacts the TMT should support. In Chapters 2 and 3 we already found a connection between the activities and the artifacts from them. The survey results repeat the same pattern – test planning is seen as highly required as software test plan, reporting activities and reports are of similar importance. However an interesting deviation exists – while traceability between requirements, test plan, test cases and test results is seen as important, the SRS itself is not seen as required for a TMT.

As above, the statistics and graph presenting the mean values for the whole target group can be found from Table 8 and Figure 9.

Table 8. Which artifacts TMT should support?

	SRS	Software test plan	Test case	Software test library	Test suite	Defects	Reports	Test library per release
Mean	3,857	4,467	4,533	4,071	4,143	3,600	4,067	4,154
Standard deviation	1,027	1,125	1,060	1,265	1,457	1,056	1,100	1,231
Mod	4	5	5	4	5	4	5	4
Median	4	5	5	4	4	4	4	4

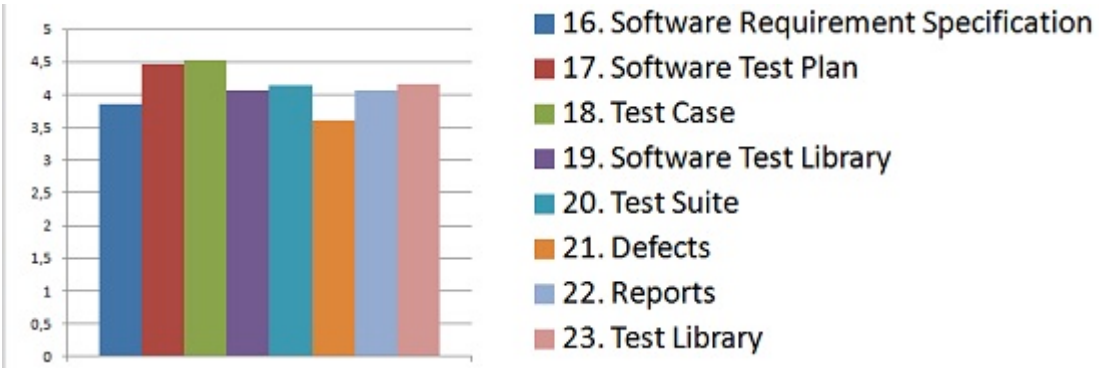

Figure 9. Which artifacts TMT should support? (mean)

The last section of the questionnaire gathered the respondents' background information. On the whole, it gives a good overview of the target group characteristics by showing the distribution of the respondents.

When analyzing the survey results, the respondents were almost split in 3, with employees in 3 ranges (Figure 10). The only exception was one respondent, where the company size was between 11 and 25 employees. Figure 11 shows similar equal distribution in the number of QA personnel, with a small deviation in the smaller QA teams. Among the results, there were no companies, who produce software only for Estonia. There is strong orientation towards exporting the software. But in the globalizing world, this is only expected.

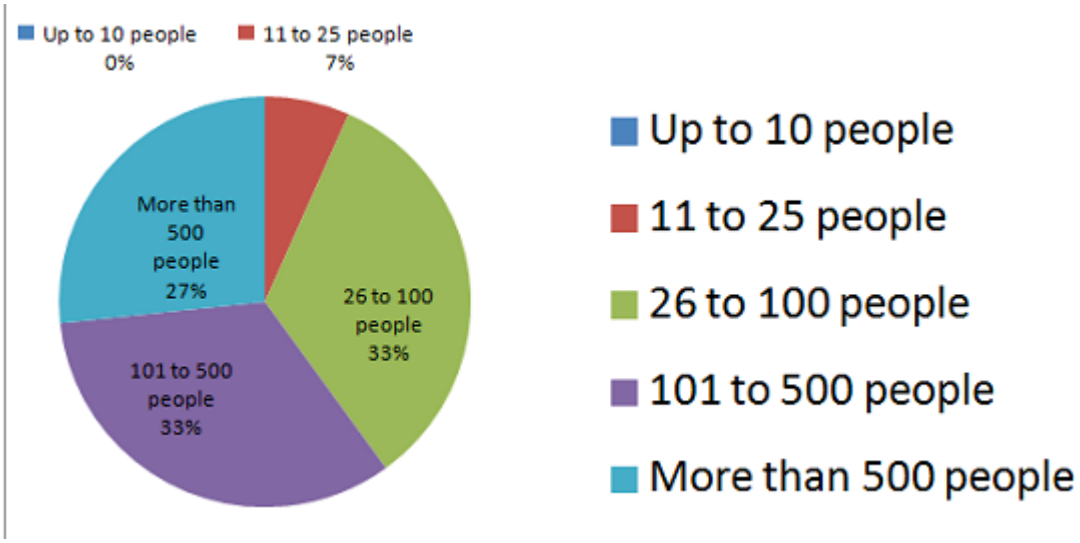

Figure 10. Size of the respondent company (number of employees)

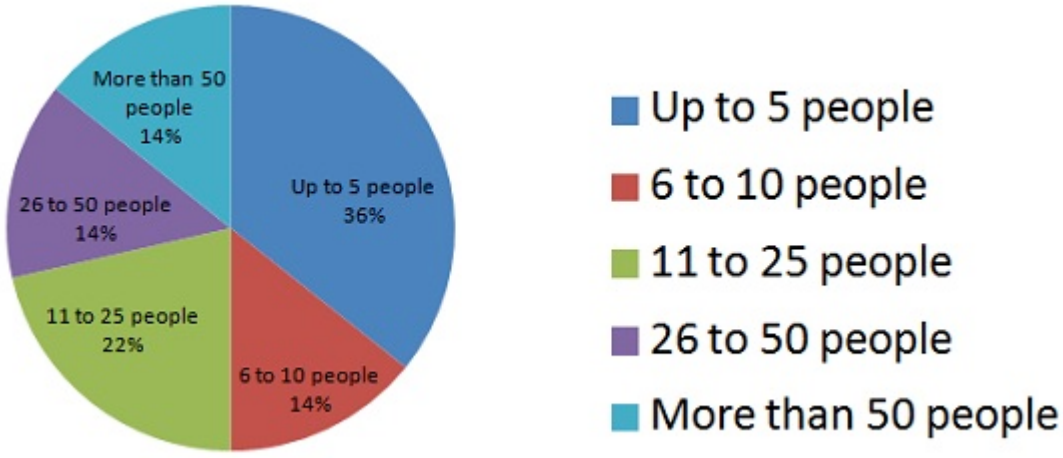

Figure 11. Size of the respondent company QA personnel

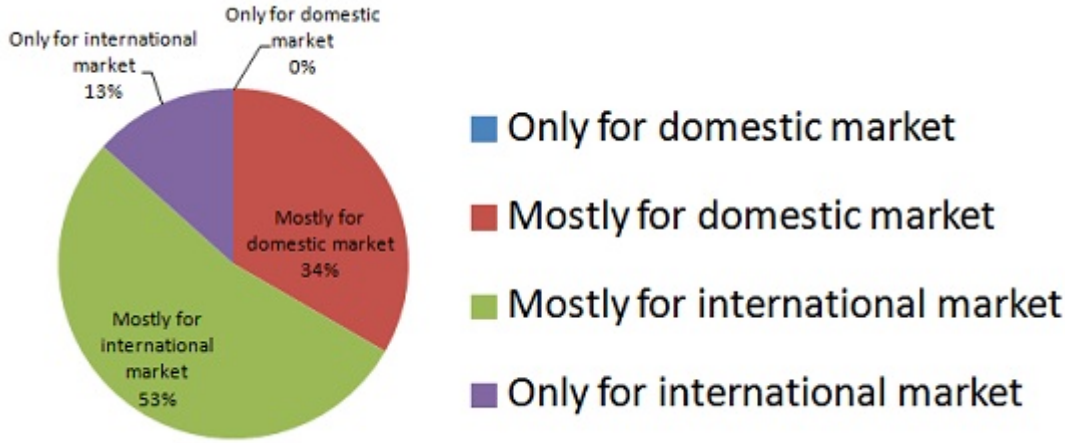

Figure 12. Market selection for the company's product

When looking at the software development models used in the companies, two larger groups could be observed. *Firstly*, there were five respondents who were using only one model. Four stated it as agile and one was Kanban. Kanban is a method for developing software products and processes with an emphasis on just-in-time delivery while not overloading the software developers[17]. As this also follows the agile method, we group this with the agile development methods. *Secondly*, the rest of the respondents stated, they are using a mixed model with Agile being in it. Leaving one respondent aside, who did not provide answer to the question, we have two groups representing 36% and 64% of the respondents.

[17] Retrieved May 11, 2012, from http://en.wikipedia.org/wiki/Kanban_(development)

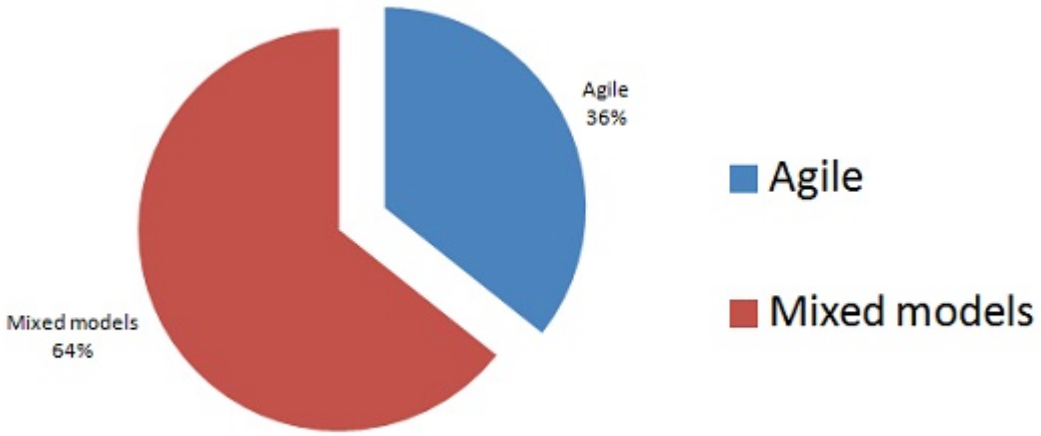

Figure 13. Development model used in the company

5.8.2 Results Based on Respondent Development Model

Analyzing the results of respondents who were using *mixed development models*, we found that in most questions, the responses do not differ significantly from the overall group. One of the differences noted (and expected) was that such companies expect the TMT to support Waterfall and V-models which can be seen when comparing Table 6 and Table 9. There are 5 attributes hitting the set 4,4 threshold. We observe that if the test planning activity reached the threshold, the software test plan was also marked as important. This confirms the relation which we showed from Chapters 2 and 3 that each activity produces an artifact and if there exists an artifact, then it was produced from an activity.

Table 9a. Survey results grouped by mixed development models (answers from 1 to 12)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12
MEAN	4,3333	4,2857	4,2500	4,6667	3,3333	4,6667	3,7778	3,6667	4,2500	4,4444	4,4444
MOD	5	5	4	5	3	5	5	5	5	5	5
MEDIAN	5	4	4	5	3	5	4	4	4,5	5	5

Table 9b. Survey results grouped by mixed development models (answers from 13 to 23)

	Question 13	Question 14	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	3,8889	4,3333	3,7500	4,4444	4,3333	4,2500	4,0000	3,6667	4,1111	4,2500
MOD	5	5	4	5	5	4	5	5	5	4
MEDIAN	4	5	4	5	5	4	4	4	5	4

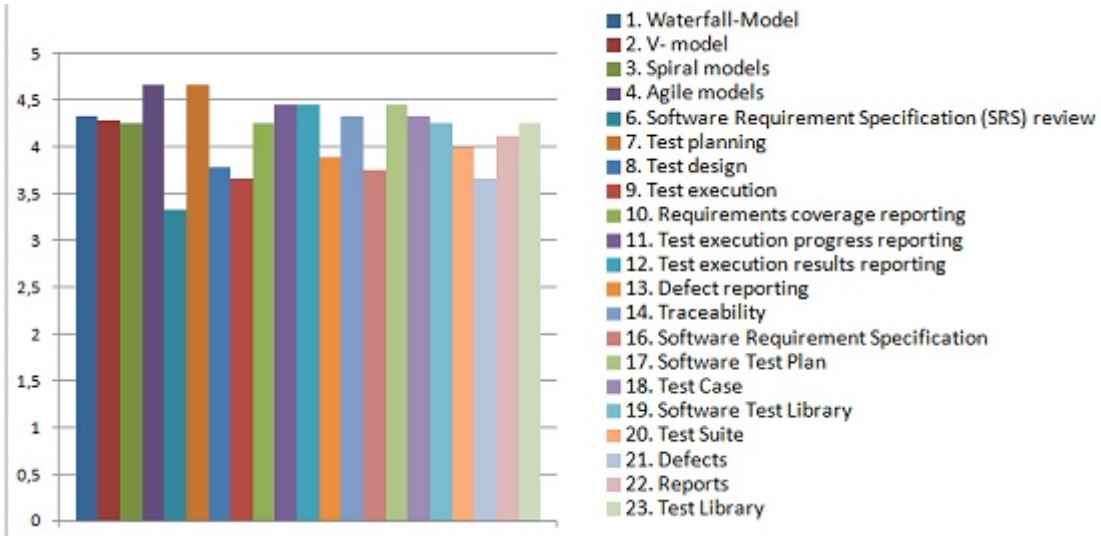

Figure 14. Result for respondents using mixed development models

Respectively when processing the data by companies with *agile development models*, the opposite trend is observed. For the question which development model should be supported, Agile models found absolute support while the others (and especially Waterfall) received less attention. From the TMT functions there were no major deviations except for the defect reporting which according to this group should be of lesser importance (mean of 2,6 compared to the overall 3,467). The third section of the questionnaire did not reveal any other differences. The summarized results can be found from Tables 10a and 10b.

In total there were 10 attributes hitting the set 4,4 threshold. The test planning, design and execution functions received higher support and respectively, the artifacts of these activities also hit the threshold level.

Table 10a. Survey results grouped by agile development models (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	2,2	3,2	3,5	5	3,2	4,6	4,4	4,8	3,4	4,8	4,8	2,6	4,8
MOD	1	4	4	5	2	5	4	5	3	5	5	3	5
MEDIAN	2	4	3	5	2	5	4	5	3	5	5	3	5

Table 10b. Survey results grouped by agile development models (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	3,8	4,4	4,8	4	4,6	3,4	4,2	4
MOD	4	5	5	4	5	4	4	4
MEDIAN	4	5	5	4	5	4	4	4

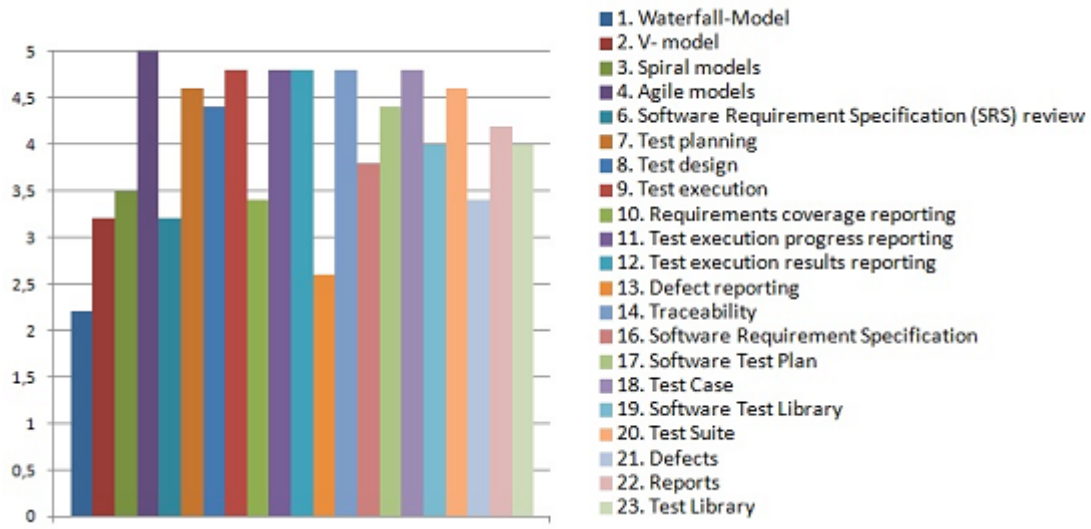

Figure 15. Result for respondents using agile development models

5.8.3 Results Based on Respondent Company Target Market

The overall statistics revealed that there are no companies developing software only for domestic market. This means that the expectations for the TMT should mostly be the same, since the target end-users are all to some extent non-Estonians.

However the results show interesting findings. Companies targeting *mostly the domestic market* show stronger expectation for the TMT to support Waterfall and V-models. Analysis indicates that this relates to the companies also using a mixed development model with Waterfall and V-model aspects. While second section of the questionnaire is similar to the general statistics, the third part demonstrates that such companies expect TMT to support more test plan and test cases (Table 11b questions 17 and 18 compared to Table 8).

9 attributes of a TMT reached the threshold. Compared to the overall statistics there are additionally mentioned the support for V-model and requirements coverage reporting.

Table 11a. Survey results grouped by respondents targeting domestic market (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	4,00	4,50	4,25	4,80	3,40	4,60	4,00	3,60	4,40	4,80	4,80	3,80	4,60
MOD	5	4	5	5	3	5	3	3	5	5	5	5	5
MEDIAN	5	4	4	5	3	5	4	3	5	5	5	4	5

Table 11b. Survey results grouped by respondents targeting domestic market (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	3,80	4,80	4,80	4,00	4,00	3,60	4,00	4,25
MOD	5	5	5	4	5	4	4	4
MEDIAN	4	5	5	4	4	4	4	4

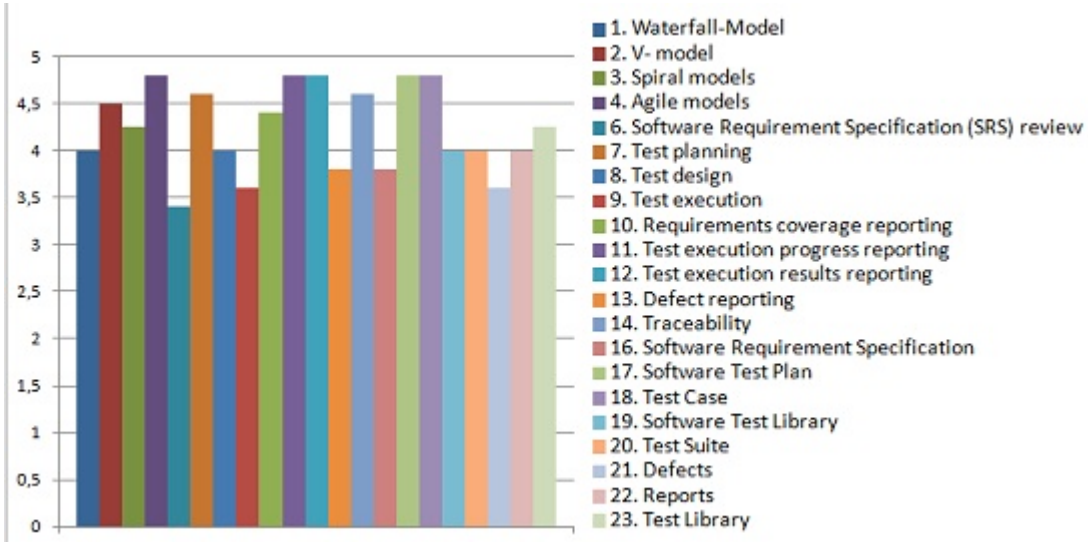

Figure 16. Result for respondents targeting mostly domestic market

The respondents whose companies were *mostly targeting international markets*, showed similar results as the overall group (Tables 12a and 12b). There were minor deviations but nothing remarkable. One additional required TMT attribute is the support for test execution.

Table 12a. Survey results grouped by respondents targeting international market (answers from 1 to 12)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12
MEAN	4,0000	4,1667	3,8571	4,8571	3,2500	4,7500	4,2500	4,6250	3,8571	4,6250	4,6250
MOD	5	4	4	5	2	5	4	5	4	5	5
MEDIAN	4	4	4	5	3	5	4	5	4	5	5

Table 12b. Survey results grouped by respondents targeting international market (answers from 13 to 23)

	Question 13	Question 14	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	3,3750	4,5714	4,0000	4,7500	4,7500	4,2500	4,3750	3,7500	4,3750	4,2857
MOD	3	5	4	5	5	4	5	4	5	4
MEDIAN	3	5	4	5	5	4	5	4	5	4

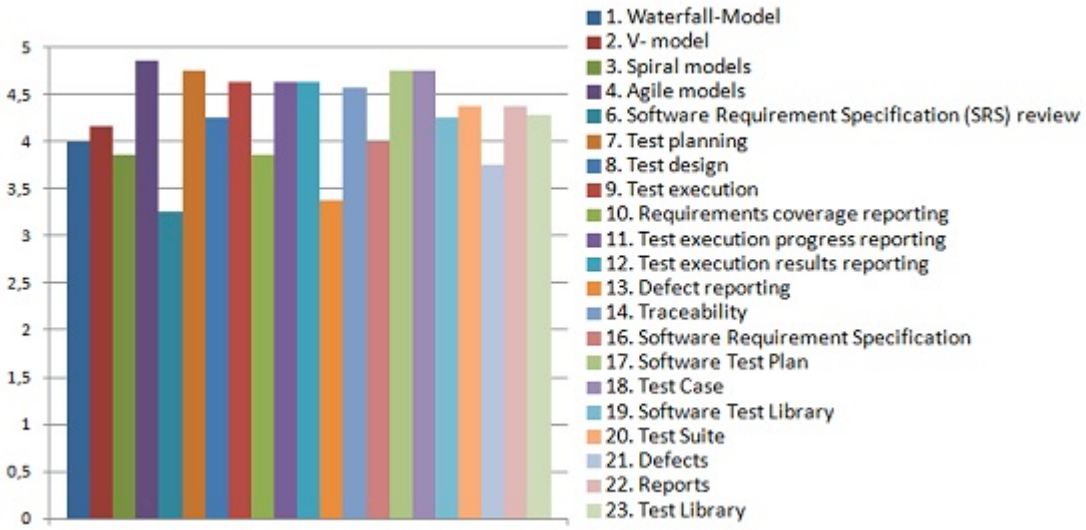

Figure 17. Result for respondents targeting mostly international market

Companies targeting only international market form a group of two. This subset is too small to draw any conclusions because their responses to the survey were of different extremes. For example, one respondent evaluated test plan and test cases as completely unimportant, while the other as completely important. What they have in common, is that they both are using agile development models and thus do not expect TMT to support other models (they gave significantly lower values for the Waterfall and V-models).

5.8.4 Results Based on Respondent Company Size

There was only one respondent who belonged to a *company with 11-25 people*. From the result, we can identify the correlation we saw earlier: the activities of test planning, test design and test execution are related to the artifacts of software test plan, test case and test suite. Respondents who identified these activities as required, found that the corresponding artifacts are also being important. Out of all the 21 listed attributes, 13 have been marked as completely required for a TMT.

Table 13a. Survey results grouped by company size, 11-25 people (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
RESULT	5	4	4	5	4	5	5	5		5	5	3	5

Table 13b. Survey results grouped by company size, 11-25 people (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
RESULT	4	5	5	5	5	4	5	4

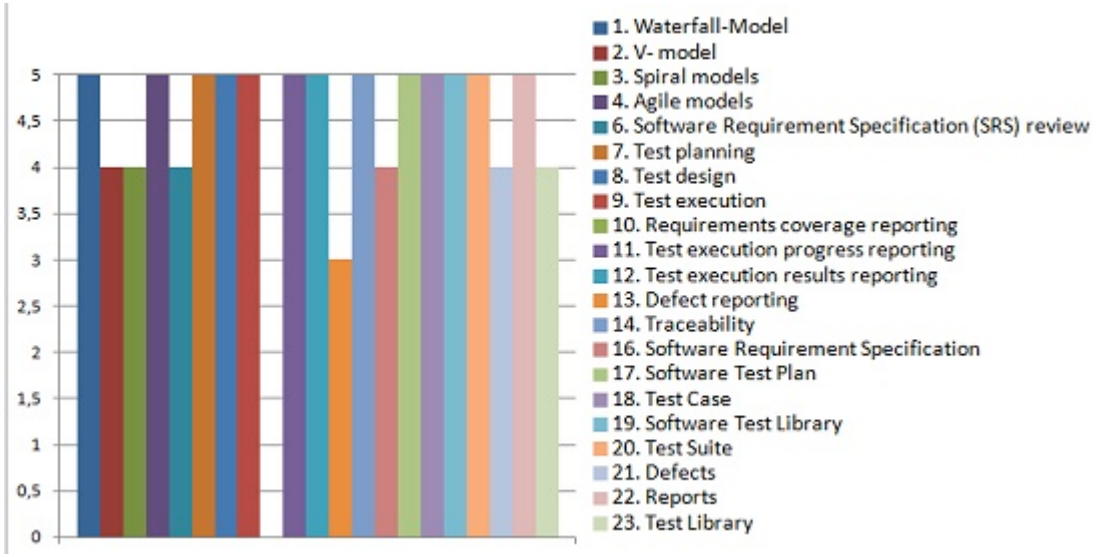

Figure 18. Result for respondents company size (11-25 people)

Running the same analysis for *companies with 26-100 employees* introduced significant deviations. For several responses, the median was around 3.5-3.6, while mod was 5. Closer inspection revealed outlying data – a respondent using session based test management was causing this. For this particular analysis, the outlying data is excluded.

This group revealed more interesting results: the TMT expected focus has shifted from the activities to the artifacts. While agile methods and test planning are still required, the reporting and traceability are no longer of high importance. On the other hand, this is one of the few groups that reported SRS to be required for the TMT. In total, there are 5 attributes expected from the TMT, which makes this one of the most focused respondent group.

Table 14a. Survey results grouped by company size, 26-100 people (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	3,66	3,00	4,00	5,00	4,00	4,75	4,25	4,00	4,25	4,25	4,25	4,25	4,00
MOD	#N/A	3	4	5	5	5	4	4	5	4	4	5	#N/A
MEDIAN	3	3	2	5	4,5	5	4	4	4,5	4	4	4,5	3,5

Table 14b. Survey results grouped by company size, 26-100 people (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	4,66	4,50	4,75	3,50	3,33	3,75	4,00	4,00
MOD	5	5	5	4	3	3	5	4
MEDIAN	5	5	5	3,5	3	3,5	4	4

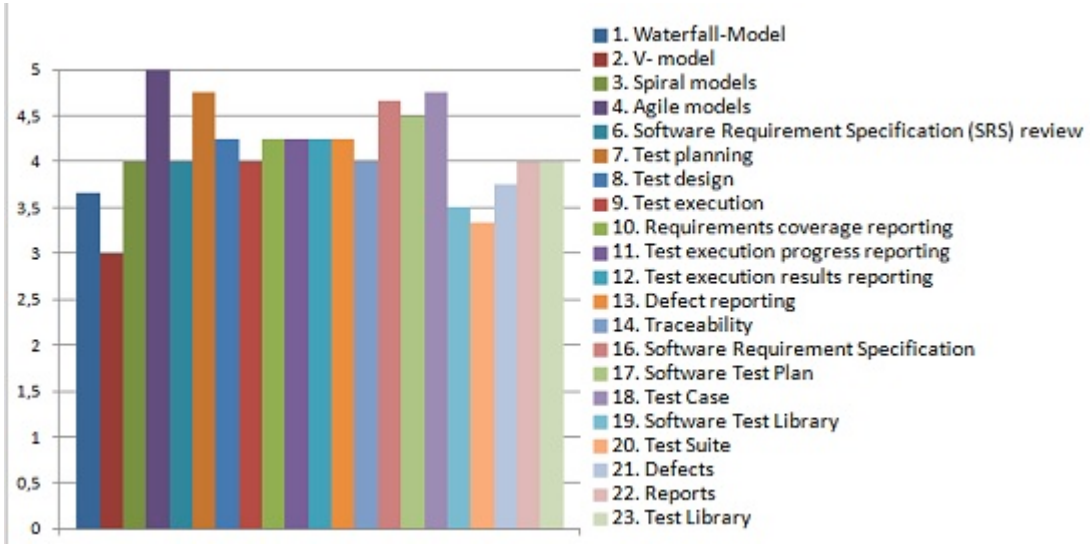

Figure 19. Result for respondents company size (26-100 people)

The next group, *companies with 101-500 employees* shows the same correlations identified earlier. However there is again a balance in the expectations for the TMT support to functions and artifacts. With the increased number of employees, the need for support of reporting is also seen. In addition to the 7 attributes of the general results which exceeded the threshold, 2 new have emerged – test execution and reports.

Table 15a. Survey results grouped by company size, 101-500 people (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	2,80	3,60	3,75	5,00	3,40	4,60	4,20	4,60	3,40	5,00	5,00	3,00	4,80
MOD	1	4	3	5	5	5	4	5	3	5	5	#N/A	5
MEDIAN	3	4	3	5	3	5	4	5	3	5	5	3	5

Table 15b. Survey results grouped by company size, 101-500 people (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	3,60	4,80	5,00	4,00	4,20	3,80	4,60	4,00
MOD	4	5	5	4	5	4	5	4
MEDIAN	4	5	5	4	5	4	5	4

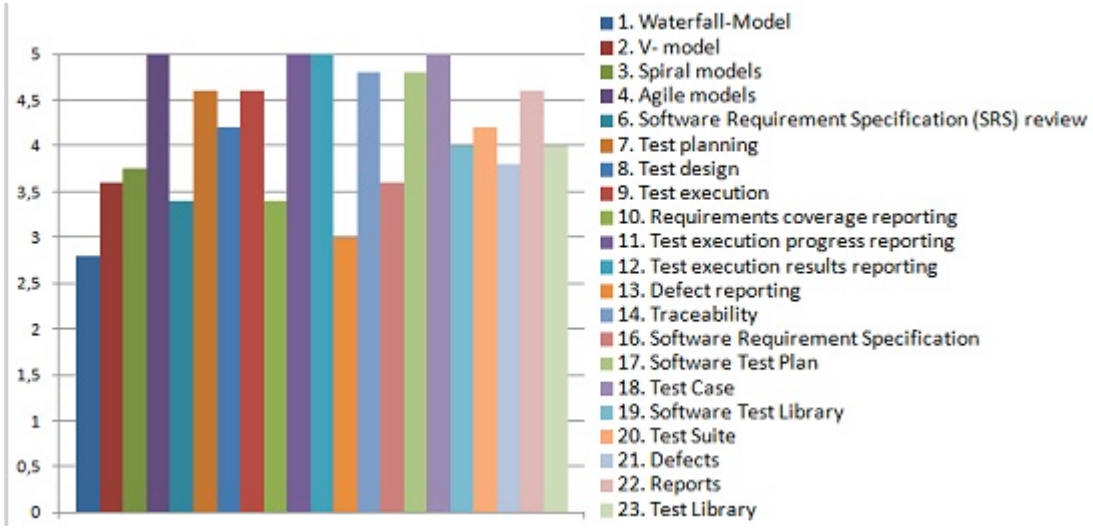

Figure 20. Result for respondents company size (101-500 people)

Final group, *companies with more than 500 employees* reveals remarkable data. All of the companies are using a mix of development models and they have the highest expectations to all of the TMT attributes. In total, there are 13 questions receiving a mean value of more than 4,4 with 9 of these having it as high as 4.75 out of 5. These attributes of the TMT are seen for the biggest companies as most crucial of them all.

Table 16a. Survey results grouped by company size, over 500 people (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	4,75	4,75	4,25	4,50	2,75	4,75	4,00	4,00	4,50	4,75	4,75	3,25	4,75
MOD	5	5	5	5	2	5	5	5	4	5	5	3	5
MEDIAN	5	5	4,5	4,5	2,5	5	4	4	4,5	5	5	3	5

Table 16b. Survey results grouped by company size, over 500 people (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	3,75	4,75	4,50	4,50	4,75	3,50	3,75	4,75
MOD	4	5	5	5	5	#N/A	4	5
MEDIAN	4	5	4,5	4,5	5	3,5	4	5

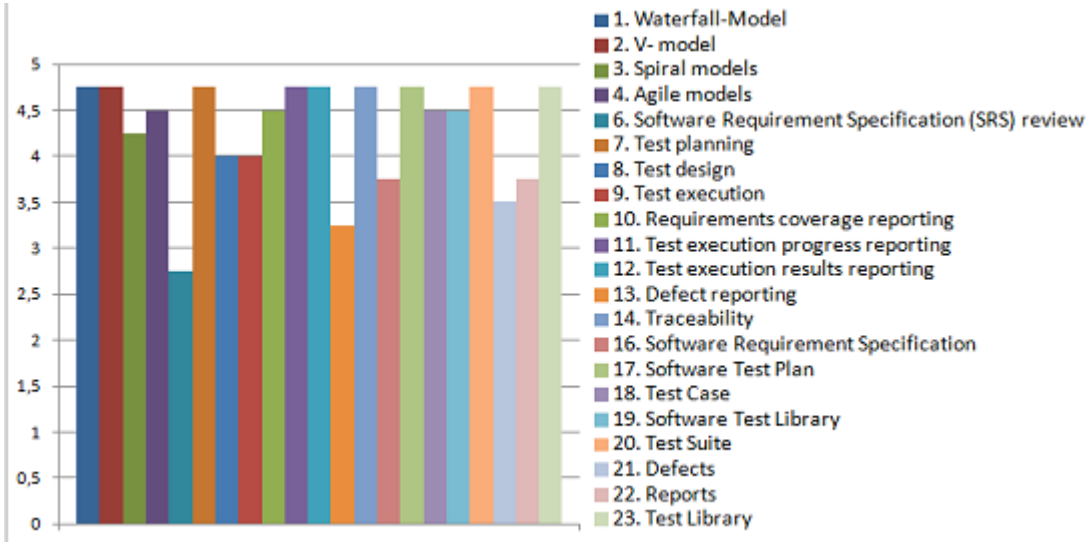

Figure 21. Result for respondents company size (more than 500 people)

5.8.5 Results Based on Respondent Company Quality Assurance Personnel

The first group is formed of *companies with up to 5 people in QA*. This group returned initial results where only two attributes exceeded the 4,4 threshold. Since this is extremely low, further investigation revealed, that one set of responses contained mostly out-lying data. The company using session based testing provided lower results for all questions than the rest from this group. Thus their responses are excluded. The re-visited results are presented in Tables 17a and 17b and on Figure 22.

There were surprisingly few expected attributes of a TMT exceeding the threshold. The most interesting is that in this group, the SRS and requirements coverage reporting have exceeded the 4,4 line.

Table 17a. Survey results grouped by company QA, up to 5 people (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	4,33	4,00	4,00	4,66	4,25	4,75	4,00	3,75	4,50	4,25	4,25	4,25	3,66
MOD	4	0	4	5	5	5	4	4	4	4	4	5	4
MEDIAN	4	0	2	4,5	4,5	5	4	4	4,5	4	4	4,5	3,5

Table 17b. Survey results grouped by company QA, up to 5 people (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	4,66	4,75	4,75	3,50	3,33	4,00	4,25	4,00
MOD	5	5	5	4	3	4	5	4
MEDIAN	5	5	5	3,5	3	4	4,5	4

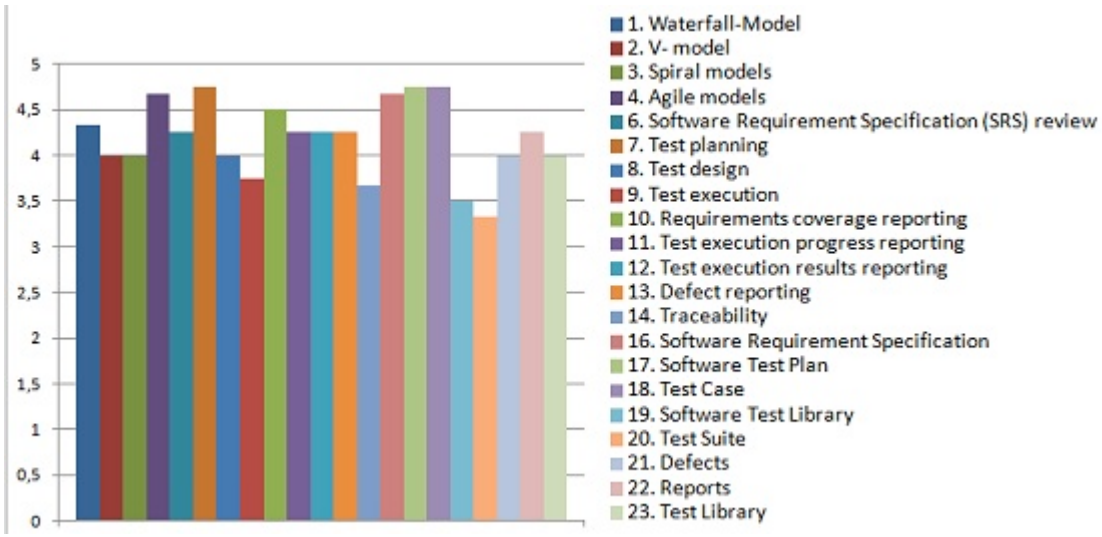

Figure 22. Result for respondents company QA (up to 5 people)

Grouping the responses by *companies with 6-10 people in QA* provided only two sets of results. Examining the respondents, we found that the number of employees of these companies ranged between 11 to 100 people, thus the results are similar to Tables 13 and with minor differences to Tables 14. Interestingly, here the correlation between test planning and software test plan does not come out. We believe this is due to the small set of results that forms this group.

Table 18a. Survey results grouped by company QA, 6-10 people (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	3,50	3,50	4,00	5,00	3,00	4,50	4,50	4,50	3,00	4,50	4,50	3,00	5,00
MOD	#N/A	#N/A	4	5	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	3	5
MEDIAN	3,5	3,5	4	5	3	4,5	4,5	4,5	3	4,5	4,5	3	5

Table 18b. Survey results grouped by company QA, 6-10 people (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	4,00	4,00	4,50	4,50	4,50	3,50	4,00	4,00
MOD	4	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	4
MEDIAN	4	4	4,5	4,5	4,5	3,5	4	4

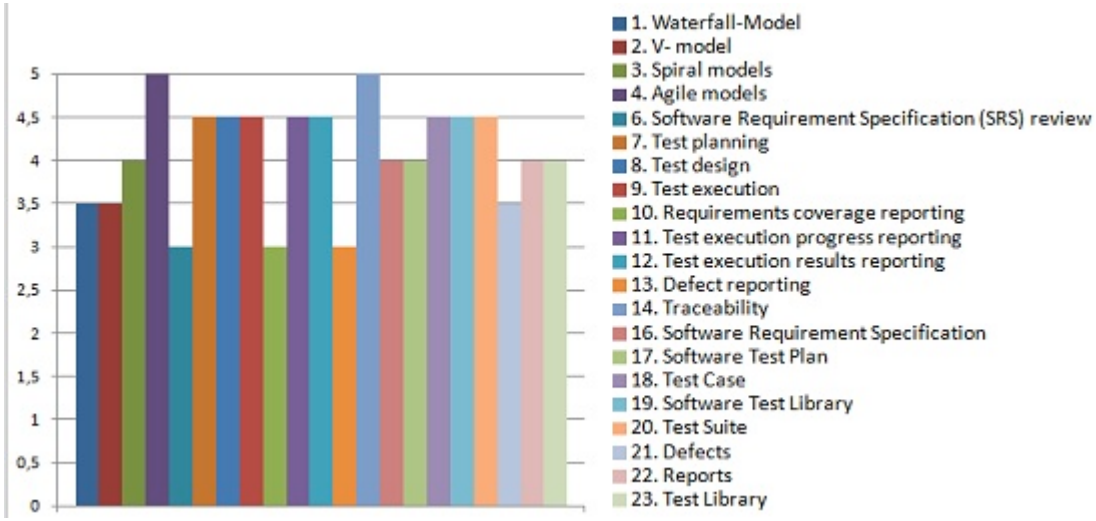

Figure 23. Result for respondents company QA (6-10 people)

The next group is formed of *companies with 11-25 people in QA*. These respondents belonged to companies of 101-500 employees and to more than 500 people company. The expectations for the TMT are similar to those shown in Tables 15 and 16. Reporting and traceability is marked to be of high importance for the TMT. However the artifacts of these activities, the reports, do not hit the set threshold.

Table 19a. Survey results grouped by company QA, 11-25 people (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	3,00	3,33	3,50	4,66	3,66	5,00	4,33	5,00	3,33	5,00	5,00	2,00	4,66
MOD	#N/A	#N/A	#N/A	5	#N/A	5	5	5	#N/A	5	5	#N/A	5
MEDIAN	3	4	3	5	4	5	5	5	3	5	5	2	5

Table 19b. Survey results grouped by company QA, 11-25 people (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	3,66	4,66	4,66	3,66	4,66	3,00	4,33	4,33
MOD	#N/A	5	5	4	5	#N/A	4	4
MEDIAN	4	5	5	4	5	3	4	4

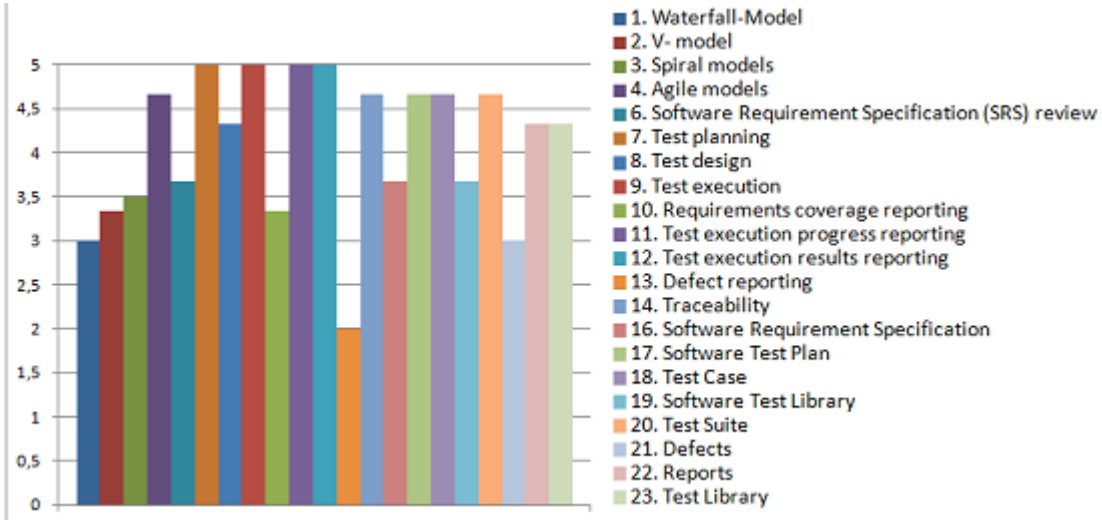

Figure 24. Result for respondents company QA (11-25 people)

Analysis of *companies with 26-50 people in QA* reveals that for this group reporting is of great importance. This also includes defect reporting and defects, which has not reached the threshold in any other group. Also worthwhile is to mention that this group sees V-model support as a required feature of a TMT which is related to the fact, that the companies are using mixed development models.

Table 20a. Survey results grouped by company QA, 26-50 people (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	3,00	4,50	4,00	5,00	4,00	4,50	3,50	4,00	4,00	5,00	5,00	4,50	5,00
MOD	#N/A	#N/A	#N/A	5	4	#N/A	#N/A	#N/A	#N/A	5	5	#N/A	5
MEDIAN	3	4,5	4	5	4	4,5	3,5	4	4	5	5	4,5	5

Table 20b. Survey results grouped by company QA, 26-50 people (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	4,00	5,00	5,00	4,00	3,50	4,50	4,50	4,00
MOD	#N/A	5	5	#N/A	#N/A	#N/A	#N/A	#N/A
MEDIAN	4	5	5	2	3,5	4,5	4,5	2

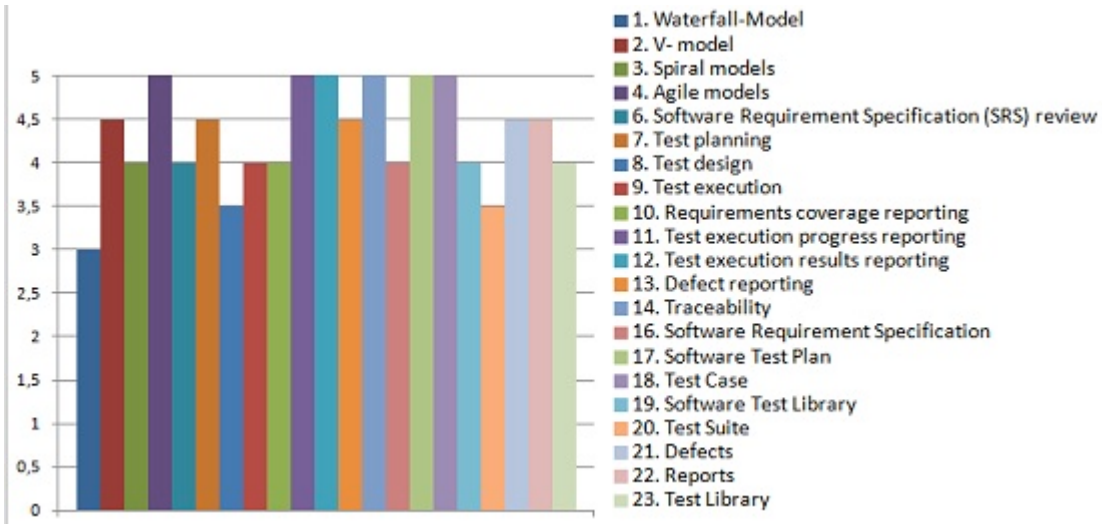

Figure 25. Result for respondents company QA (26-50 people)

The final group, *companies with more than 50 people in QA***,** is made of companies with more than 500 employees. As such, they are expecting the TMT to support all development methods. The only areas with not high expectations are SRS and defects (with related reporting). Surprisingly test execution did not reach the threshold, but this could be due to the small volume of responses.

Table 21a. Survey results grouped by company QA, over 50 people (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	5,00	5,00	5,00	5,00	2,00	5,00	5,00	4,00	4,50	5,00	5,00	3,50	5,00
MOD	5	5	5	5	2	5	5	#N/A	#N/A	5	5	#N/A	5
MEDIAN	5	5	5	5	2	5	5	4	4,5	5	5	3,5	5

Table 21b. Survey results grouped by company QA, over 50 people (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	3,00	5,00	5,00	5,00	5,00	3,50	3,50	5,00
MOD	#N/A	5	5	5	5	#N/A	#N/A	5
MEDIAN	3	5	5	5	5	3,5	3,5	5

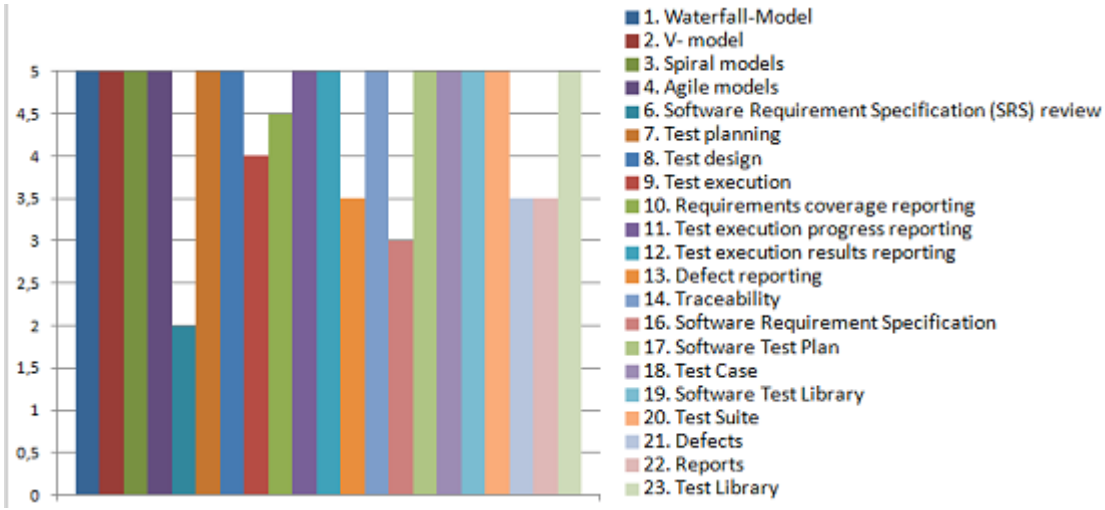

Figure 26. Result for respondents company QA (more than 50 people)

5.8.6 Results From Lithuanian Software Development Companies

The *results from Lithuania* are presented separately since there were only 4 responses to the survey. Three of the companies were using agile development models and one was using V-model. All companies had up to 10 QA personnel and total size of 101-500 employees. The results are summarized in Tables 22a, 22b and Figure 27.

Table 22a. Survey results from Lithuanian companies (answers from 1 to 14)

	Question 1	Question 2	Question 3	Question 4	Question 6	Question 7	Question 8	Question 9	Question 10	Question 11	Question 12	Question 13	Question 14
MEAN	3,75	4,25	3,75	4,50	3,00	4,00	4,00	4,75	4,75	5,00	4,75	4,75	4,75
MOD	4	5	4	4	3	5	5	5	5	5	5	5	5
MEDIAN	4,0	4,5	4,0	4,5	3,0	4,5	5,0	5,0	5,0	5,0	5,0	5,0	5,0

Table 22b. Survey results from Lithuanian companies (answers from 16 to 23)

	Question 16	Question 17	Question 18	Question 19	Question 20	Question 21	Question 22	Question 23
MEAN	3,50	4,25	4,50	4,50	4,50	4,75	4,75	5,00
MOD	3	5	5	5	5	5	5	5
MEDIAN	3,5	4,5	5,0	5,0	5,0	5,0	5,0	5,0

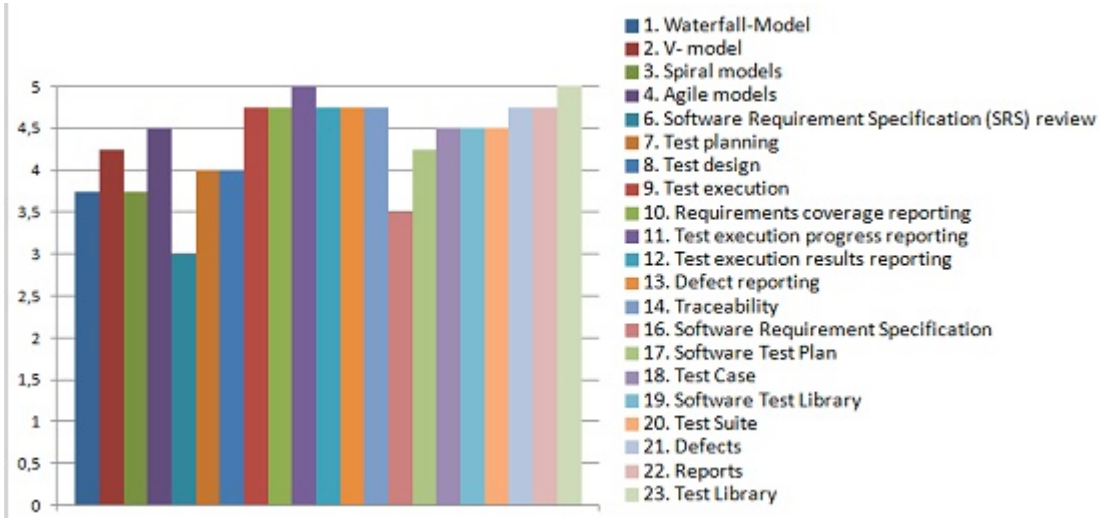

Figure 27. Survey results from Lithuanian companies

5.9 Interpretation and Comparison to Related Work

The survey was performed to answer the question, what are software tester's needs for the TMT and, based on that, to develop an evaluation framework for the TMT acquisition. Analysis of the responses has provided following insights:

- Firstly, it has confirmed that there is *dependency between the activities and artifacts* of a TMT. Test planning and software test plan are related – if respondents expected TMT to support one attribute, then the other also received higher results. The same can be said about software requirements specification review and software requirements specification, test design and test case, test execution and test suite. However the direct link between reporting and reports could not be confirmed, since only 3 respondent groups ranked reports high enough to reach the set 4,4 threshold.
- Secondly, *7 TMT attributes met the threshold of 4,4* when looking at the aggregated results: agile models, test planning, test execution progress and results reporting, traceability, software test plan and test case. When analyzing the responses, the agile models and test planning could be seen in all groups meeting the threshold. The software test plan and test case did not reach the threshold only once. The specific set of requirements depending on the company characteristics will be presented in Chapter 6 where TMT requirements framework is constructed.
- *Comparing the results with the related work*, Yang *et al* (2006) reports, that the coverage-based testing tools were tailored for specific application domain. We can observe similar expectations for the manual TMTs – depending on the company and the tools used for development, the requirements for TMT vary. Although Garousi (2009) did not contribute with a framework for choosing a TMT, he did state that further research on the matter is required. We find the same. Similar survey should be carried out on a wider target audience, possibly in the whole Baltics.

5.10 Summary of the Survey Results

This concludes the fifth chapter of the thesis which covers the survey to find out what are the expectations to a TMT. An electronic questionnaire was composed from the theoretical study of aspects of software quality assurance and from the capabilities of existing

commercial test management tools. The survey was carried out among Estonian and Lithuanian IT companies, although most responses came from Estonian companies.

The results of the survey were analyzed and found correlations are summarized in section 5.9. A threshold level for the responses was set as 4,4 - meaning, that more than every third respondent should „completely agree“ that TMT should support the attribute while no-one responded with lower than „rather agree“.

The responses were analyzed by calculating the mean, mod and median values, describing them in tabular form, and visualizing in histograms. The statistical figures were calculated first for the whole survey respondents and later for separate sets grouped by the respondent company background. 7 of the TMT attributes exceeded the 4,4 threshold. These were the support for agile models, test planning, test execution progress and results reporting, traceability, software test plan and test case.

We use the results from this chapter to create the TMT evaluation framework in Chapter 6. The set of TMT attributes exceeding the thresholds for particular company characteristics will contribute by allowing tailored requirements for TMT procurement.

Chapter 6 – The Test Management Tool Evaluation Framework

This chapter introduces the TMT evaluation framework. First we explain why the evaluation framework is required. We then introduce the identified requirements of TMTs and provide their descriptions. Next the features are incorporated into feature diagram. We then introduce how TMT product diagrams are created and illustrate it with two examples. Finally, the TMT evaluation framework is presented and we give guidelines for its application.

6.1 Purpose of the Framework

Our survey provides us with the Estonian IT companies' expectations for TMTs. From their background info we find that majority of them were using different TMTs, some use in-house developed tools. The evaluation framework created in this paper, provides companies a base-line for making a decision when choosing which TMT to procure. The framework also assists in evaluating existing TMTs by giving a clear measurable value of how the tool meets the users' company expectations.

6.2 Test Management Tool Requirements

In Chapter 5 we asked the respondents of the survey to evaluate how important different aspects for a test management tool are. The evaluation framework uses these aspects as requirements for TMTs.

The *first set of requirements* is the capability to support different development models. The differences of the models are explained in Chapter 2. The models and requirements for TMT under observation are:

- Waterfall model;
- V-Model;
- Spiral models;
- Agile models.

The *second set of requirements* is the functions a TMT should be capable to support. From the earlier analysis we have identified that following are expected functions for TMTs:

- Software requirement specification review – the ability to review and provide feedback to the SRS document;
- Test preparations:
  - Test planning – the process of setting the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. It also includes resource allocation and defining any dependencies for the testing activities;
  - Test design – act of creating and writing test cases and suites for testing software;
- Test execution – the process of running the tests;
- Traceability – the ability to trace the linkage between requirements, test plans, test cases, test suites and defects;
- Reporting of:
  - Requirement coverage – the possibility to provide reports showing the requirement coverage by test cases or executed test suites;
  - Test execution progress – the ability to report how many test suites have been run;
  - Test execution results – the TMT support for reporting the results of test suites;
  - Defect – the ability to use the TMT for notifying managers, developers or other project stake-holders of found problems in the software.

The *third and final set of requirements* is the artifacts a TMT should support. The artifacts are listed below with their meanings:

- Software requirement specification – a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software;
- Test preparation artifacts:
  - Software test plan – an artifact documenting the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements;
  - Test case – a test script detailing the test data, steps and expected results of the system behavior;
- Repositories of:
  - Software test library – a repository containing all created test cases and test suites. The repository is often extended to other TMT artifacts as well;
  - Software test library per release – a repository containing all created test cases and test suites grouped by product releases. This repository is more often used in spiral and agile software development models;
- Test execution artifacts:
  - Test suite – a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviors;
  - Defect – an error, flaw, mistake, failure, or fault in a computer program which is recorded by the tester. TMT should support creation and managing this artifact;
- Reports – various documents covering different aspects of the software testing activities. The TMT support for creating and storing the reports has been identified as a requirement.

6.3 Test Management Tool Feature Diagram

The online questionnaire addresses the TMT required features and asks the respondents to evaluate how highly they expect the features to be supported. In Chapter 5 we summarized the results and set a threshold of 4,4, meaning, that more than every third respondent should „completely agree“ that TMT should support the attribute while no-one responds with lower than „rather agree“. This threshold is used to identify the required features of a TMT – any feature reaching or exceeding the set level is considered important.

Next we take the identified requirements and map them to TMT feature diagram. We use the mean values of all respondents from the questionnaire (Tables 6-8) to find which features reached the 4,4 threshold. These are set as mandatory and all the rest as optional. This way we get 7 mandatory and 14 optional features for a general TMT (Figure 28). The 7 mandatory features are:

- Support for agile models;
- Test planning;
- Traceability between different artifacts;
- Reporting of test execution progress;
- Reporting of test execution results;
- Software test plan artifact;
- Test case.

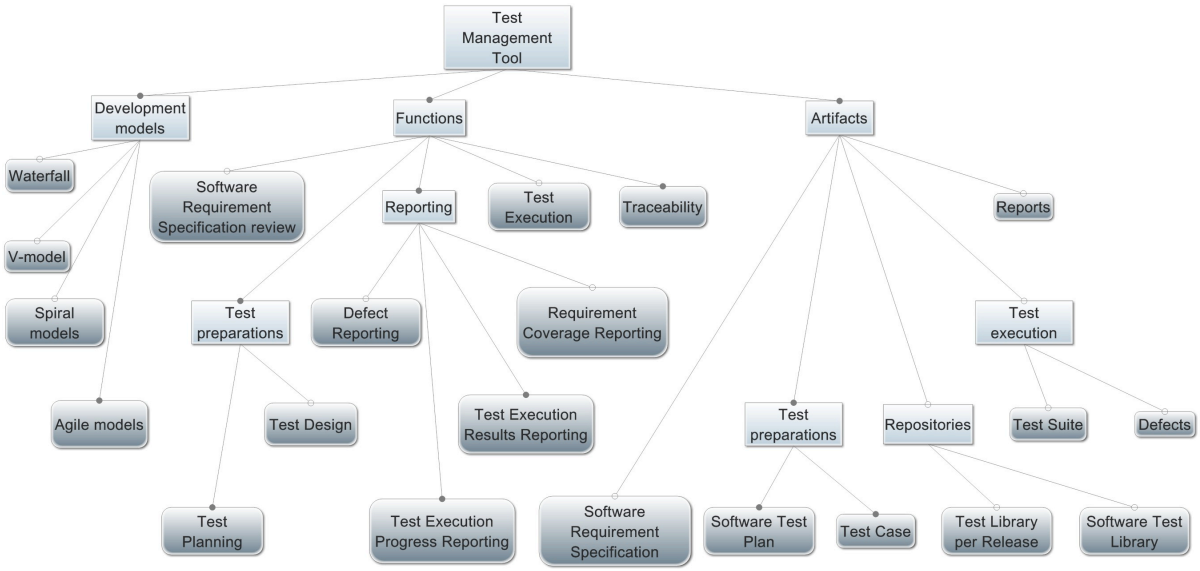

Figure 28. General TMT feature diagram (FD)

6.4 Test Management Tool Product Diagram

The online questionnaire results from Chapter 5 are grouped by the different characteristics of company backgrounds. There were four aspects:

- The software development model used in the company;
- The target market for company products;
- The company size measured in number of employees;
- The company QA department size measured in number of employees.

While the general TMT feature diagram informs which features are mandatory for a TMT in general, the results from the survey grouped by these aspects provide additional requirements specific for each group. We use the 4,4 threshold to identify the optional features of TMT per specific company characteristics when analyzing the results. In sections 5.8.2 till 5.8.5 the survey data is analyzed and the optional features are revealed. We use here the term ’optional feature’ since this is used in the product diagram literature. However from testing perspective, these features are also required. In order to get a specific product diagram, we merge the mandatory features from the general TMT FD and the features reaching the set threshold. This way we get 13 different product diagrams (Appendix C).

We bring now two examples how these product diagrams are created. From Tables 10a and 10b we find features which reach the set 4.4 threshold. These are the optional features for a TMT for a company which is using agile software development models. We take the general TMT feature diagram mandatory features and overlap it with the additional features. This results in getting TMT product diagram for companies with agile models (Figure 29).

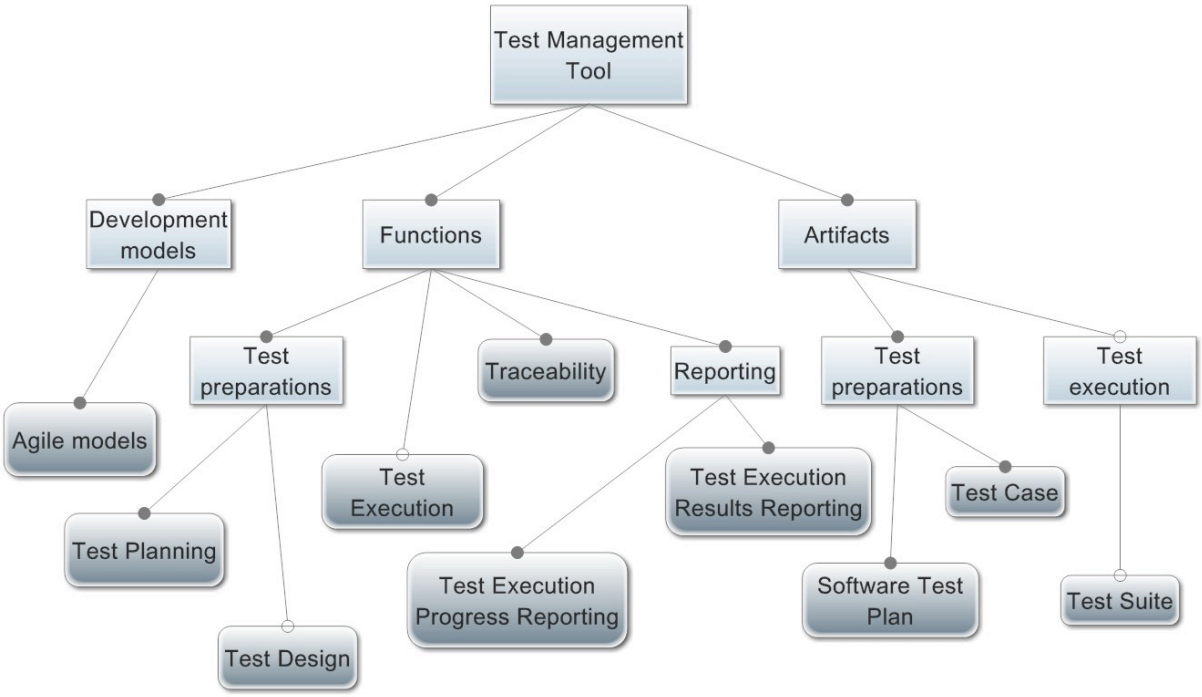

Figure 29. TMT product diagram for companies with agile models.

In similar way, we get a product diagram for companies with 26-50 QA people. We take the results from Tables 20a and 20b and find the features reaching the threshold. Applying these as optional features to the mandatory features from the general TMT FD, we get another TMT product diagram for companies where QA size is 26-50 people (Figure 30).

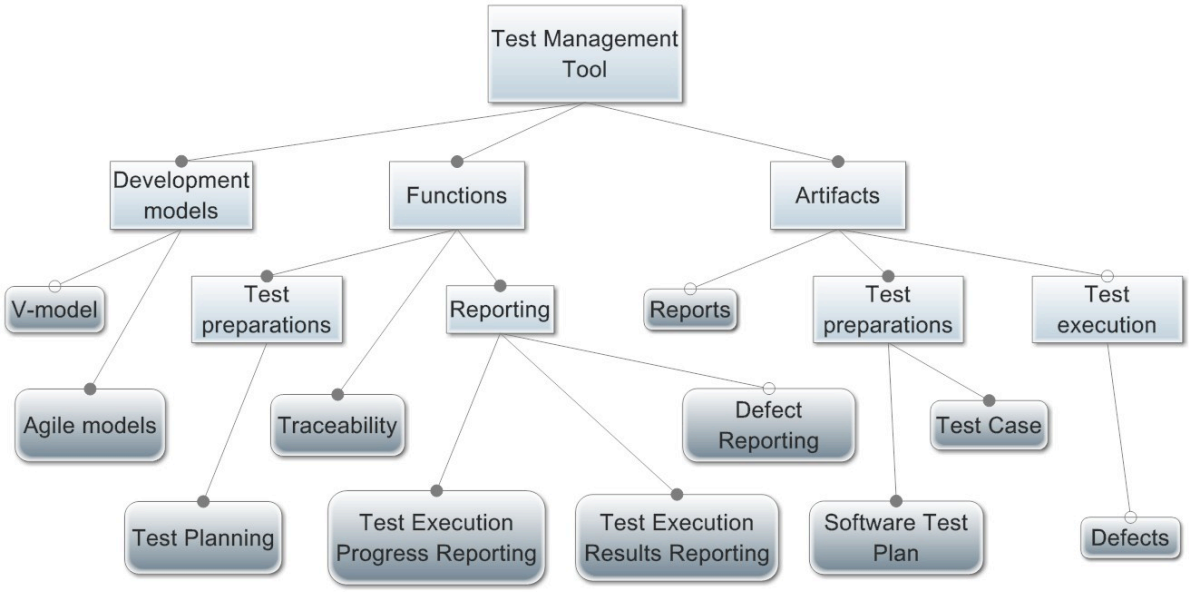

Figure 30. TMT product diagram for companies with 26-50 QA people.

6.5 Test Management Tool Evaluation Framework and Guidelines for Using

The set of TMT product diagrams listed in Appendix C provides us with powerful tools which are the basis for the TMT evaluation framework. By overlapping several individual product diagrams, we get a TMT product diagram for a specific company. Using the product

diagram as input for the framework and performing the evaluation, the output will be a measurement how well a TMT meets the company's expectations.

There are several possible options for evaluating TMTs. One of them is counting the number of steps/clicks the evaluator has to make before the TMT activity or artifact is complete. Another option is to measure the time it takes to create an artifact or perform an activity. The usability of the TMT is yet next measurement. However in this case, the evaluation should be carried out by several specialists in order to mitigate the subjectivity of the evaluation. Regardless of the method chosen, the TMT evaluation framework expects the evaluator to rate TMT features on a scale of 0 to 3. To illustrate it, we will now provide the guidelines for performing an evaluation with the framework (Figure 31).

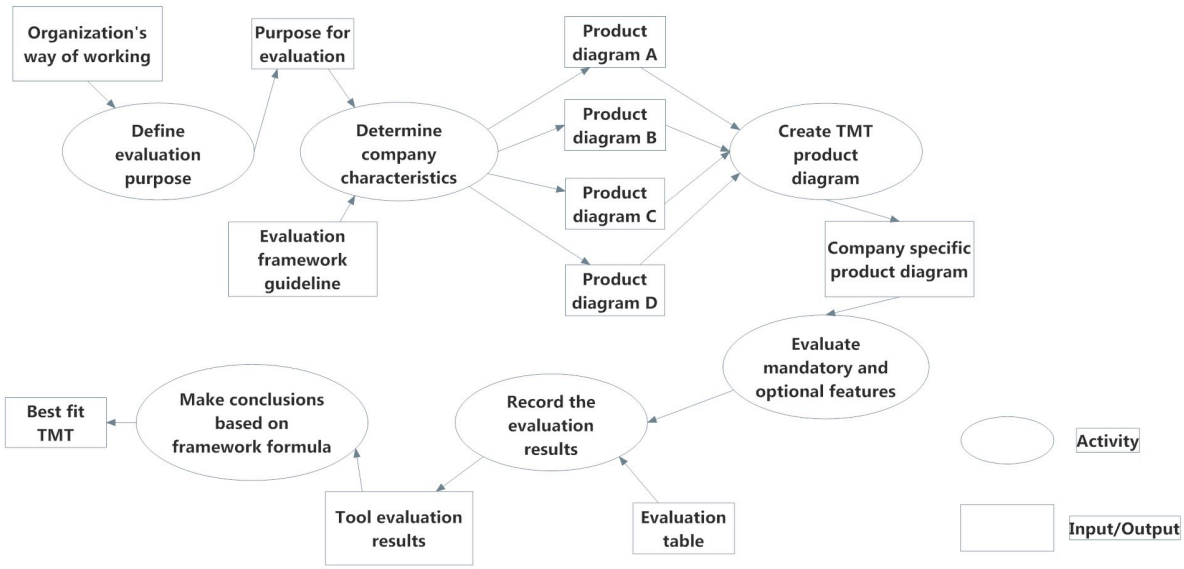

Figure 31. TMT evaluation framework processes

First step – define evaluation purpose. The evaluator defines the evaluation purpose – either to understand the effectiveness of the existing TMT or to perform evaluation for procuring new TMT. There can be further reasons. Depending on the purpose, the set of TMTs to evaluate will vary and longer time should be planned for this activity.

Second step – determine company characteristics. The environment where the evaluation is carried out is identified. This is done by determining the company four characteristics: the company size, QA personnel size, the used development model, and the target market for company's products. Based on these aspects, the evaluator takes the specific product diagrams from Appendix C. If the information about the company is un-known, then the general TMT feature diagram should be used. For example, if the company had up to 10 QA persons *and* the company size were up to 100 persons *and* the company used only agile development model *and* developed mostly for domestic market, then following product diagrams should be used: Appendix C Figures 1, 3, 6 and 10.

Third step – create TMT product diagram. The next step is to overlap the product diagrams. Overlapping means taking all features from one product diagram and adding them to the next diagram, but not duplicating any of them. Doing this completes a new company specific product diagram. For example the overlapping of appendix C Figures 1 and 3 results in the following product diagram (Figure 32).

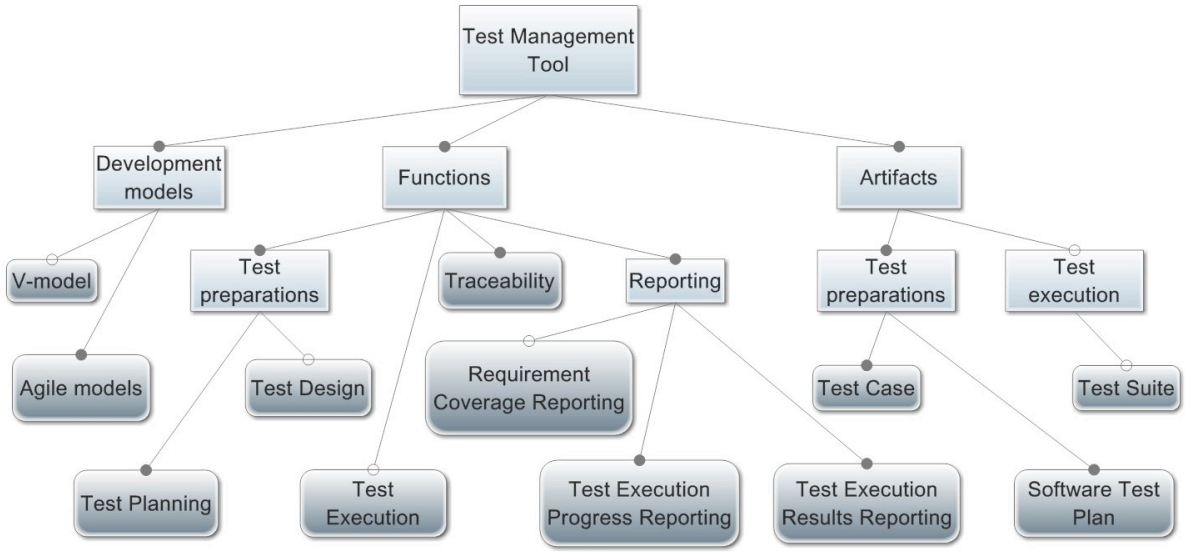

Figure 32. Company specific TMT product diagram – agile and mostly domestic market

Applying all four product diagrams returns still 7 mandatory features but in addition, there are also optional[18] features. In our example, the overlapping results are shown on a product diagram, Figure 33. The seven optional features of the TMT are:

- (support for) V-model;
- Test design;
- Test execution;
- Requirement coverage reporting;
- Software requirement specification;
- Software test library;
- Test suite.

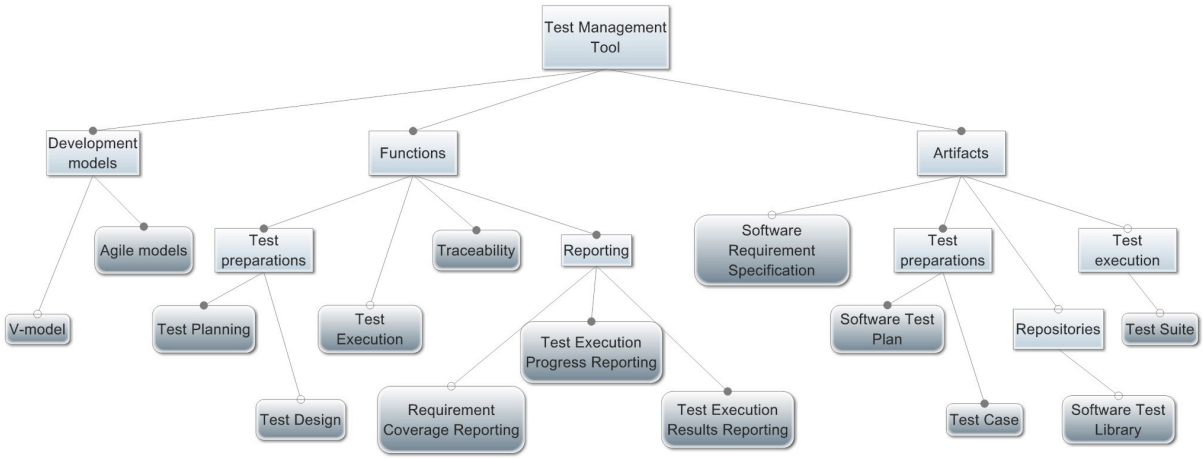

Figure 33. Company specific TMT product diagram

[18] We will use here the term 'optional feature' since this term is used in the product diagram literature. However when adapting product diagram, the optional features might also be evaluated because they are required by organization.

Fourth step – evaluate mandatory and optional features. The evaluator measures how well the listed features are supported by the TMT. As described earlier, there are multiple options. Regardless of the chosen method, each mandatory and optional feature would get a value between 0 and 3. The meaning of the values is:

- 0 – does not meet the expectations;
- 1 – mostly does not meet the expectations;
- 2 – mostly meets the expectations;
- 3 – completely meets the expectations.

Fifth step – record the evaluation results. The evaluator records the evaluation results. In the evaluation framework we use tabular format. The first column lists all TMT mandatory and optional features, the second column holds the value given to the feature. Finally, the last row totals all the results. Table 23 in our example provides arbitrary values as reference how to fill it.

Table 23. TMT mandatory and optional features evaluation table

TMT feature	**Value**
V-model	1
Agile models	3
Test planning	1
...	...
Test suite	2
Total	7

Sixth step – conclusions based on framework formula. The final step is to draw the conclusions. In order to understand to which degree the TMT meets the company's expectations, we use formula: $p= \frac{\sum_1^n result_i}{n*3} * 100\%$ where $result_i$ is the value a feature received, n is the total number of features identified from the company TMT product diagram. The number 3 is the highest value which can be given. Following these provided guidelines, the evaluator receives a result showing the percentage of how much the tool meets the expectations of the company.

6.6 Summary of Test Management Tool Evaluation Framework

This concludes Chapter 6 which introduced the framework we created from the analysis and interpretation of the survey results. The framework helps QA specialists to evaluate a TMT and based on the results to choose the tool which most meets the company's expectations.

In this chapter we have presented the general TMT feature diagram which lists the seven mandatory features of any TMT. We also demonstrated how based on the company characteristics product diagrams are created. Finally, we presented the framework and gave guidelines how to use it. By applying the framework the evaluator will receive a result for the tool showing to what extent the tool meets the company's expectations.

In the next chapter we present the case study performed in order to test the TMT evaluation framework. The case study focus is set on the framework usability and the case study is carried out among QA specialists.

Part 3 - Validation

Chapter 7 – Testing Framework Usability

This chapter describes the validation of the TMT evaluation framework. Firstly, we introduce why we tested the framework and what we expect to receive as the results from the case study. Secondly, we describe the participants and give the reasons why they were selected. We continue and report how we carried out the study and summarize the interviews. Finally, we provide the results of the case study.

7.1 Introduction to Testing the Framework

In Chapter 6 we described the TMT evaluation framework. The framework provides test managers with the means to evaluate a tool while removing large portion of subjectivity from the process. The test manager can now see which features TMTs should support for a company with certain set of characteristics.

To confirm the usability of the TMT evaluation framework, we carried out a case study among five QA specialists. The purpose of the study was to understand whether the tool is fit for use and to find out what should be done to improve the framework's usability.

7.2 Participant Selection

Five QA specialists were contacted and asked to evaluate the TMTs their companies are using. The limitation of the participants to five persons was derived from the relatively small number of Estonian IT companies. Secondly, we wanted to carry out a small proof-of-concept test, not to make a full research on the matter.

The participants were selected from four companies, in order to confirm how the company specific product diagram would be perceived. All testers were working in different QA departments which resulted in getting different evaluation metrics. Compared to the online questionnaire, the participants of the case study did not have to be test managers. Since the TMTs are used by testers, the feedback for the framework was gathered from people working with the tools. In our case, three respondents were test engineers and two were test managers.

The *first respondent's* company creates software mostly for international market *and* it has more than 50 QA engineers *and* it has more than 500 employees *and* they are using mixed development models. Based on this information, the product diagram for the company can be seen on Figure 34. The TMT the respondent evaluated was HP Quality Centre.

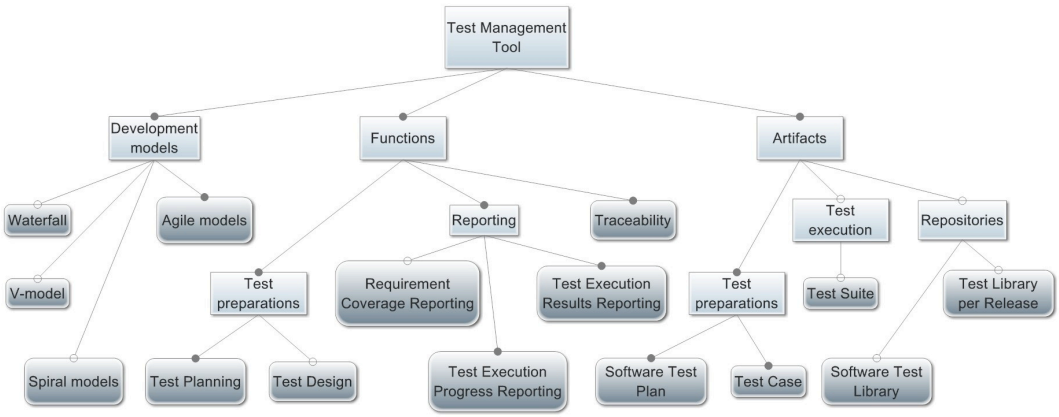

Figure 34. Respondent #1 company specific product diagram.

The *second respondent's* company creates software only for international market *and* it has 6-10 QA engineers *and* it has 51-100 employees *and* they are using only agile development model. Based on this information, the product diagram for the company can be seen on Figure 35. The respondent's company uses a mixture of MS Office applications for TMT.

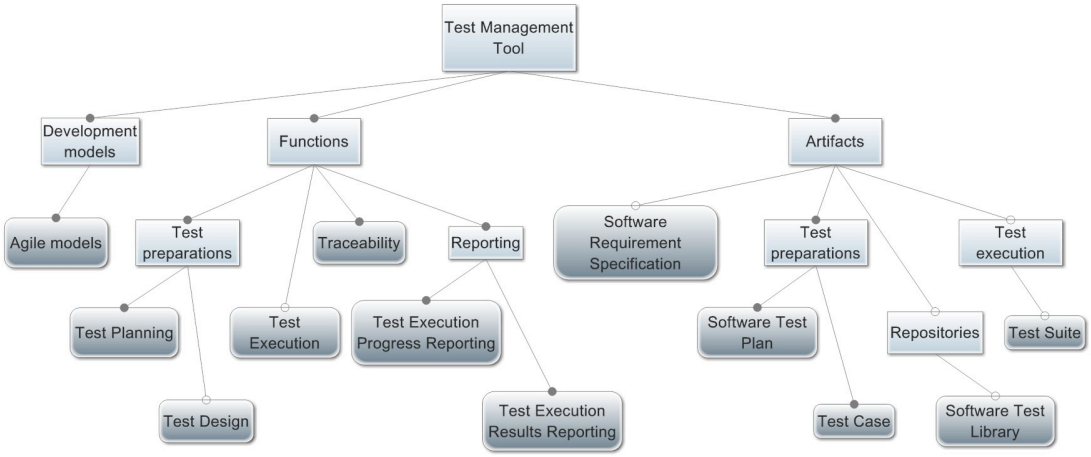

Figure 35. Respondent #2 company specific product diagram.

The *third respondent's* company creates software mostly for domestic market *and* it has 26-50 QA engineers *and* it has 101-500 employees *and* they are using only agile development model. Based on this information, the product diagram for the company can be seen on Figure 36. The TMT the respondent evaluated was Confluence.

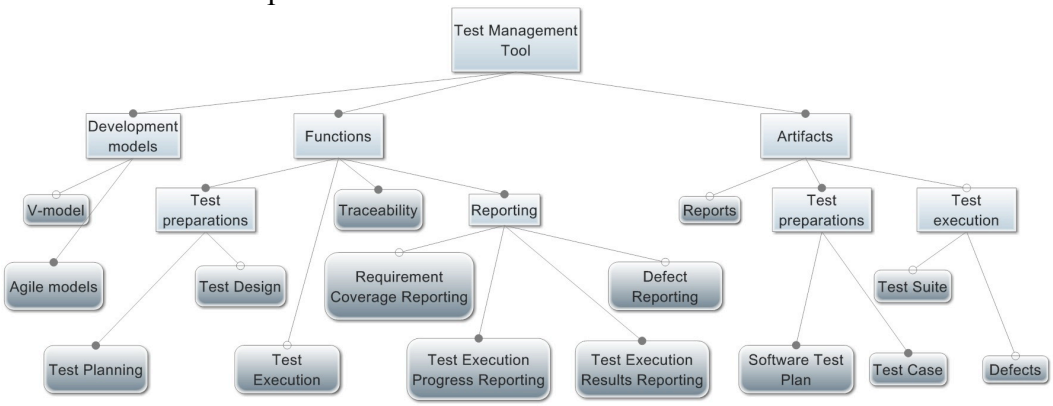

Figure 36. Respondent #3 company specific product diagram.

The *fourth respondent's* company creates software for international market only *and* it has up to 10 QA engineers *and* it has 26-100 employees *and* they are using only agile development model. Based on this information, the product diagram for the company can be seen on Figure 37. The TMT the respondent evaluated was a custom in-house developed tool.

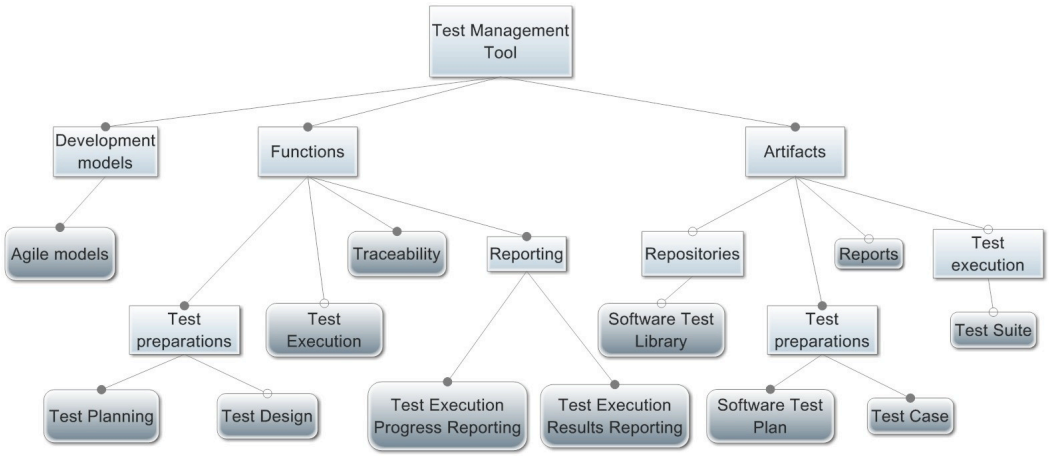

Figure 37. Respondent #4 company specific product diagram.

The *fifth respondent's* company creates software mostly for domestic market *and* it has 11-25 QA engineers *and* it has 26-100 employees *and* they are using only agile development model. Based on this information, the product diagram for the company can be seen on Figure 38. The respondent's company uses an in-house developed TMT.

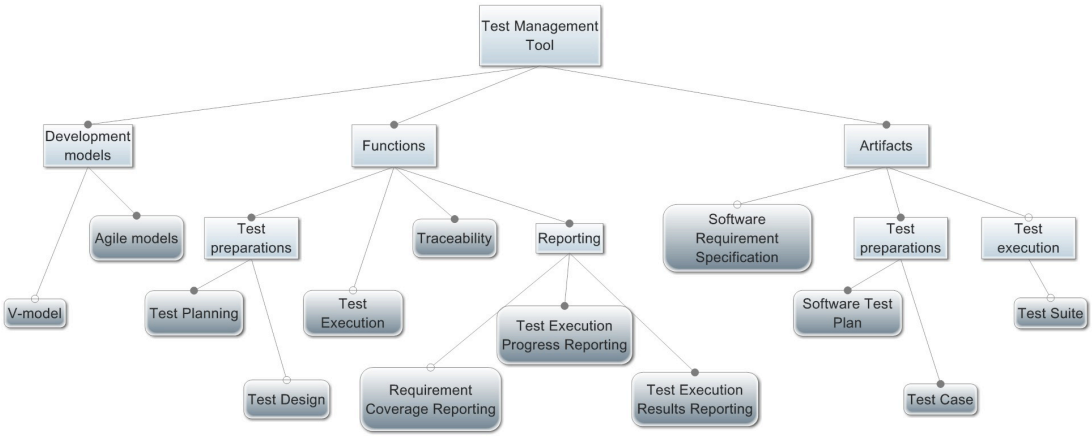

Figure 38. Respondent #5 company specific product diagram.

7.3 Evaluation Framework Usability Interviews

The case study gathered responses to the questionnaire addressing the usability of the TMT evaluation framework. We used personal approach and performed interviews with the respondents. This provided closer feedback and allowed us to ask additional specifying questions when the answers were vague or superficial.

Firstly, we introduced to the participants the purpose of the framework. We explained that the product diagrams are created based on theoretical studies and market research which was later confirmed by performing online survey among Estonian IT companies. The survey results were analyzed, seven mandatory features for all TMT were identified and optional features depending on company characteristics for TMT were determined.

Secondly, the participants were given a guideline for using the framework (Appendix D) and the product diagrams (Appendix C). We asked them to first read the guideline and then to evaluate their company's TMTs by applying the evaluation framework. Additional information was provided when questions regarding the framework were raised.

After the evaluation with the framework, the respondents were requested to respond to the short questionnaire. We were interested in five aspects:

- How easy is the framework to learn?
- How efficient is it for frequent TMT evaluation?
- How easy is it to remember the six steps of the framework?
- How satisfied are You with the framework?
- How easy it is to understand the benefits of the framework?

The first two interviews revealed that the guideline requires a change. We improved the framework guideline by adding more explanations and examples, thus making it easier to understand. After that, we proceeded with next respondents.

The interviews with the respondents lasted on an average an hour and were carried out both in English and Estonian. Three of the interviews were recorded on tape while two respondents asked us only to make footnotes. The goal of the interviews was to understand whether improvements should be made to the framework and to get feedback on the usability.

7.4 Results of the Case Study

Each of the case study participants was asked to give feedback on the framework usability and to rate it on a scale of 1 (lowest) to 5 (highest). We provide the results in Table 24. The first response about the ease of learning is lower than the others, since improvements were made to the framework guideline based on the interviews. The rest of the survey does not have outstanding differences.

Table 24. TMT evaluation framework usability

	Respondent #1	Respondent #2	Respondent #3	Respondent #4	Respondent #5
How easy is the framework to learn?	3	4	4	4	4
How efficient is it for frequent TMT evaluation?	3	4	5	4	5
How easy is it to remember the six steps of the framework?	5	2	4	5	5
How satisfied are You with the framework?	2	4	4	4	4
How easy it is to understand the benefits of the framework?	2	5	5	4	4

How easy is the framework to learn? After applying improvements to the guideline, all respondents considered the framework easy to learn. People understood the workflow how to use the framework. They also implied that the six required steps were clear to use. The hardest step for all respondents was the third step – creating TMT product diagram. For future work, 3 of the participants recommended creating the evaluation framework as an application which would automatically create the TMT product diagram based on the respondent company characteristics.

How efficient is it for frequent TMT evaluation? The respondents understood that for frequent use, the evaluator would only have to repeat steps 4-6. The hardest step (according to

them), third step, would be done once, since the company does not change. As such, most respondents found that the framework is rather efficient for frequent use.

How easy is it to remember the six steps of the framework? Most of the interviewees told the steps are easy to remember. They said the steps are logical and quickly followed. One respondent did suggest shortening the step names, however to keep the framework easy to learn, we did not make the change.

How satisfied are You with the framework? This question turned out to be the hardest to answer. The participants had never used a framework for evaluating a TMT; it was new experience for them. While they did not say they are not satisfied with the framework, they were also reluctant to confirm, that it met their expectations. There was one exception – one of the test managers believed, that evaluating a TMT should be done by company employee and not based on a framework, since „the employee knows what is required by the company".

How easy it is to understand the benefits of the framework? All respondents understood clearly the benefits of the framework – mitigation of the subjectivity of evaluation by using an evaluation framework based on structured approach. Similar to the previous question, there was one outstanding respondent who strongly believed that the framework would not be beneficial for his company. Despite the outlying result, majority of the interviewees agreed that the benefits are rather easy to understand.

Finally, respondents were asked to bring out the best aspect of the framework. Three interviewees told that they got a clear number representing how much the tool met with the company expectations. The other two agreed that the framework is excellent for frequent use and saves time.

7.5 Threats to Validity

We have applied the guidelines (e.g. personal interviews, objective questions, addressing potential risks to validity) suggested by Good *et al* (2008) to minimize the threats to the validity of our case study. However there are still few which should be paid attention to when reviewing the results.

The first and probably the biggest threat, is the *number of participants* in the case study. We asked 5 testers to evaluate our framework. The number of the participants was kept low due to the scope of this thesis. For future work, further analysis should be carried out by including more respondents to the evaluation framework's usability case study.

Another aspect which should be mentioned is that the framework validation focused only on the usability and did not address the *completeness* of the TMT evaluation framework. To address this risk additional research should be carried out to confirm if all required TMT features have been included to the evaluation framework.

Finally, a threat to the validity of the case study comes from not confirming the *correctness* of the evaluation framework. We have not investigated if the framework will produce the same results for different respondent groups who evaluate the same TMT with the TMT evaluation framework. Our focus was only on the framework usability and thus, the correctness is subject for future work.

7.6 Summary of the Evaluation Framework Testing

We carried out a case study to investigate the usability of the TMT evaluation framework. The study involved 5 practitioners and they were asked to evaluate their company's TMTs using the TMT framework. Each respondent evaluated different tools.

The results confirm that the framework is easy to learn, efficient for frequent use and fit for purpose. There was one respondent, who was doubtful of the tools suitability for the task,

especially the list of TMT requirements. He believed subjective evaluation of tools would meet company's expectations better. However the TMT evaluation framework relies on the current Estonian IT companies' expectations, thus, mitigating the subjectivity of TMT evaluation at least in this geographical area. In conclusion, the strongest aspects of the TMT evaluation framework are that using it frequently is efficient and it gives clear measurable value for the TMT.

Part 4 - Conclusions

Chapter 8 – Conclusions and Future Work

In this thesis we have developed the framework to evaluate test management tools. We achieved this by conducting a review of testing literature and existing TMTs. The review results contribute to our online questionnaire. The survey results in the creation of the TMT evaluation framework. We have validated the usability of the TMT evaluation framework in an interview study among quality assurance specialists. This chapter presents our conclusions.

8.1 Conclusions

The research questions of this work are (i) what are the requirements for TMT; and (ii) how to evaluate if the tool meets the set expectations. During the work we have made the following observations:

- *Software development models used in IT companies affect the different sets of testing activities and artifacts used by QA teams.* Our study on the theory of testing reveals that software development models have different expectations to software testing. From model to model, there are some same activities and artifacts, but as can be seen from Chapter 2 Tables 1 and 2, there are variations.
- *There exists at least one artifact for each testing activity. If there is a testing artifact then it was produced from an activity.* This correlation is used to construct the online questionnaire to understand Estonian IT companies' expectations for TMTs.
- *Choosing TMT for procurement from existing tools on the market requires thorough evaluation.* Analysis in Chapter 3 on existing TMTs revealed that they support most but not all testing activities and artifacts. When choosing one of the tools for procurement one must know the specifics of these tools and also the expectations to the tool. To understand the expectations, we used the results from Tables 3 and 4 in the online questionnaire.
- *Seven TMT features are considered as required by every software development company.* The survey results reveals that there are seven features which are seen as required for TMTs: (i) support for agile models; (ii) test planning; (iii) traceability between different artifacts; (iv) reporting of test execution progress; (v) reporting of test execution results; and (vi) software test plan artifact; and (vii) software test cases.
- *Company characteristics impact the expectations to TMT.* Although the 7 required features are seen as mandatory, other TMT features are characterized depending on the company's needs. The selection of the TMT is influenced by (i) software development model used in the company; (ii) target market for the company; (iii) company personnel size; and (iv) size of the QA in the company.
- *The TMT framework mitigates the subjectivity of the evaluation by suggesting a structured approach.* The TMT selection consists of 6 steps: (i) define evaluation purpose; (ii) determine company characteristics; (iii) create TMT product diagram; (iv) evaluate mandatory and optional features; (v) record the evaluation results; and (vi) make conclusions based on framework formula.

Finally, we have analyzed usability of the TMT evaluation framework. We have asked a number of QA specialists to apply the framework for their currently used TMTs. We concluded that: (i) the framework is easy to learn; (ii) efficient for frequent use; (iii) fit for purpose; and (iv) the framework provides measurable value for TMT.

8.2 Future Work

In this thesis we have presented the TMT evaluation framework which is focused on the manual testing aspects. We hope that this research will inspire more QA specialists to pursue further investigations in the field of software testing and similar study would be made for tools supporting test-automation.

The major future work includes extending the research to more countries and to analyze how the requirements for TMTs differ between countries and regions. It is also important to understand if and how company characteristics affect the TMT expectations. And finally, the list of new features which TMTs could potentially support requires further analysis since the software testing is still continuously evolving.

Kokkuvõte

Tarkvara Testide Haldamissüsteemi Hindamisraamistik

Magistritöö

Evari Koppel

Tarkvara testimine on korduvalt tõestanud oma olulisust tarkvara arenduse juures viimase kümnendi jooksul. Tarkvara testimise tunnustuse kasvuga on esile kerkinud paljud elektroonilised testide haldamissüsteemid (THS). Kuigi nende hindamiseks on mitmeid võimalusi, pole me siiski leidnud selleks ühtselt aktsepteeritud meetodit.

Me usume, et see on probleem, mida tuleks uurida, sest THS hindamine on sageli subjektiivne, sõltudes pigem hindaja arvamusest kui objektiivsest lähenemisest. Sama mure on ka kvaliteedikontrolli meeskondade juhtidel, kui neil palutakse hinnata, kas THS, mida neil kasutatakse, vastab ettevõtte vajadustele.

Mõistmaks THS hindamise olulisust, uurisime me testimisprotsesside alast kirjandust ning analüüsisime hetkel olemasolevaid rakendusi. Seejärel kaardistasime tuvastatud testimisprotsessid ning nende väljundid. Läbi viidud analüüsi tulemusena saadud andmete põhjal koostasime veebiküsitluse ning saatsime Eesti IT-firmadele.

Uuringu tulemuste põhjal koostasime me THS hindamisraamistiku, mis aitab ettevõtetel mõõta, kas ostetav THS on joondatud firma eesmärkidega, ning vähendab hinnangu andmisel subjektiivsust. Meie raamistik võimaldab testimis- ning projektijuhtidel mõista, kas nende ettevõttes kasutusel olev rakendus vastab firma ootustele. Veendumaks loodud hindamisprotsessi kasutatavuses, viisime kvaliteedikontrolli spetsialistide seas läbi täiendava uuringu, mis kinnitas meie ootusi.

Meie lõputöö edasi arendamiseks on mitmeid võimalusi. Raamistikust võib luua veebirakenduse, et seda oleks kergem kasutada või laiemalt levitada. Samuti tuleks uurimust laiendada, kaasates ning analüüsides teiste euroopa riikide IT-firmade THS nõudeid. Kindlasti ei saa mainimata jätta, et THS nõudeid tuleks aja möödudes täiendada vastavalt uutele trendidele kvaliteedikontrollis. Lõpetuseks me usume, et käesoleva lõputöö tulemus, THS hindamisraamistik, on praktiline ning vajalik panus tarkvara kvaliteedikontrolli kogukonnale.

Bibliography

Alves C & Castro J (2001), *CRE: A Systematic Method for COTS Component Selection.* Proceedings of the XV Brasilian Symposium on Software Engineering ©2001

Bach J (2011). *Avoding my curse on tool vendors.* Retrieved May 04, 2012, from http://www.satisfice.com/blog/archives/596

Bach J (2011). *Show #6 with James Bach (part1).* Retrieved May 04, 2012, from http://www.testcast.net/shows/mp3/Testcast-006-with-James-Bach-part-1.mp3

Beck *et al* (2001), *Principles behind the Agile Manifesto.* Retrieved May 04, 2012, from http://www.agilemanifesto.org/principles.html

Boehm B (1986), *A Spiral Model of Software Development and Enhancement.* ACM SIGSOFT Software Engineering Notes, Volume 11 Issue 4. Publisher: ACM New York, NY, USA ©1986

Boehm B & Basili V.R (2001). *Software defect reduction top 10 list*. IEEE Computer. Retrieved May 04, 2012, from http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/82.78.pdf

Boehm B, Port D, Yang Y, Bhuta J & Abts C (2003), *Composable Process Elements for Developing COTS-based Applications.* Proceedings of the 2003 International Symposium on Empirical Software Engineering ©2003

Britannica (2003), *Merriam-Webster's Collegiate Dictionary, 11th Ed.* Publisher: Merriam-Webster, Inc. ©2003

Carvallo J. P, Franch X & Quer C (2005), *A Quality Model for Requirements Management Tools.* Requirements Engineering for Sociotechnical Systems. Publisher: Idea Group, Inc ©2005

Chung L, Cooper K & Courtney S (2004), *COTS-Aware Requirements Engineering: The CARE Process.* Procedings of the 2nd International Workshop on Requirements and COTS Components ©2004

Clark J.O (2009), *System of Systems Engineering and Family of Systems Engineering from a Standards, V-Model, Dual V-Model, and DoD Perspective.* Retrieved May 04, 2012, from http://www.incose.org/hra/past_events/INCOSEHRAChapterPresentation_SoSEandFoSE_JOC_090826.pdf

Commella-Dorda S, Dean J.C, Morris E & Oberndorf P (2001), *A Process for COTS Software Product Evaluation.* Proceedings of the 1st International Conference on COTS-Based Software Systems ©2001

Dillman D. A (2000), *Mail and Internet Surveys: The Tailored Design Method, 2nd Ed.* Publisher: Wiley Publishing ©2000

Finkelstein A, Spanoudakis G & Ryan M (1996), *Software Package Requirements and Procurement*. Proceedings of the 8th International Workshop on Software Specification and Desing, IEEE Computer Society ©1996

Garousi V (2009), *Choosing the Right Testing Tools and Systems Under Test (SUTs) for Practical Exercises in Testing Education*, Presented in the 8th Workshop on Teaching Software Testing (WTST), Melbourne, Florida ©2009

Gaston C & Seifert D (2005), *Evaluating Coverage Based Testing.* Retrieved May 04, 2012, from http://lfm.iti.kit.edu/download/EvaluatingCoverageBasedTesting2005.pdf

Good P.I & Hardin J.W (2008), *Common Errors in Statistics, 2nd Ed.* Publisher: Wiley-Interscience New York, NY, USA ©2008

Grenning J.W (2002), *XP and Embedded Systems development.* Retrieved May 04, 2012, from http://narkeshare.googlecode.com/files/EmbeddedXp.pdf

Gregor S, Hutson J & Oresky C (2002), *Storyboard Process to Assist in Requirements Verification and Adaptation to Capabilities Inherent in COTS.* Proceedings of the 1st International Conference on COTSBased Software Systems ©2002

IEEE (1990), *IEEE Standard Glossary of Software Engineering Terminology*. Retrieved May 04 2012, from http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf

Kaner C, Falk J & Nguyen H.Q (1999), *Testing Computer Software, 2nd Ed*. Publisher: John Wiley & Sons, Inc. New York, NY, USA ©1999

Keyes J (2003), *Software Engineering Handbook*. Publisher: Auerbach ©2003

Kolawa A & Huizinga D (2007), *Automated Defect Prevention: Best Practices in Software Management.* Publisher: Wiley-IEEE Computer Society Press ©2007

Kontio J (1996), *A Case Study in Applying a Systematic Method for COTS Selection,* Proceedings of the 18th, International Conference on Software Engineering ©1996

Kunda D (2003) *STACE: Social Technical Approach to COTS Software Selection.* Component-Based Software Quality - Methods and Techniques ©2003

Lichota R. W & Vesprini R. L (1997), *PRISM: Product Examination Process for Component Based Development.* Proceedings of the 5th International Symposium on Assessment of Software Tools ©1997

Maiden N. A & Ncube C (1998), *Acquiring COTS Software Selection Requirements.* IEEE Software, Volume 15 Issue 2 ©1998

Matulevičius R (2009), *Aggregated Process For Evaluating Requirements Engineering Tools. Published.* Published at 15th International Conference on Information and Software Technologies ©2009

Matulevičius R & Sindre G (2006), *Requirements Engineering Tool Evaluation Approach.* In A. G. Nilsson, R.Gustas, G. W. Wojtkowski, W. Wojtkowski, S. Wrycza, and J. Zupancic (eds) *Advances in Information Systems Development: Bridging the Gap between Academia and Industry.* Publisher: Springer-Verlag New York, Inc. Secaucus, NJ, USA ©2006

McGlohon M, Glance N & Reiter Z (2010), *Star Quality: Aggregating Reviews to Rank Products and Merchants*, Presented in the Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media ©2010

Morisio M & Tsoukias A (1997), *IusWare: a Methodology for the Evaluation and Selection of Software Products.* IEE Proceedings on Software Engineering ©1997

Myers G (1979), *Art of Software Testing.* Publisher: John Wiley & Sons, Inc. New York, NY, USA ©1979

Myrvold A (2011). *11 software testing myths.* Retrieved May 04, 2012, from http://testapprentice.com/2011/02/21/software-testing-myths/

Neyer A.K & Harzing A.W (2008), *The Impact of Culture on Interactions: Five Lessons Learned from the European Commission.* Accepted for European Management Journal ©2008

Noether G.E (1990), *Introduction to Statistics The Nonparametric Way.* Publisher: Springer Publishing Company, Incorporated ©1990

Ochs M. A, Pfahl G, Chrobok-Diening G & Nothhelfer-Kolb B (2000), *A COTS Acquisition Process: Definition and Application Experience.* Proceedings of the 11th ESCOM Conference ©2000

Phillips B. C & Polen S.M (2002), *Add Decision Analysis to your COTS Selection Process.* Technical report,Software Technology Support Center ©2002

Pohl K (2010), *Requirements Engineering: Fundamentals, Principles, and Techniques.* Publisher: Springer Publishing Company, Incorporated ©2010

Raccoon L.B.S (1995), *The chaos model and the chaos cycle*, ACM SIGSOFT Software Engineering Notes, Volume 20, Issue 1. Publisher: ACM New York, NY, USA

Schwaber K & Sutherland J (2011), *The Scrum Guide*. Retrieved May 04, 2012, from http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf

Sommerville I (1993), *Software engineering, 4th Ed.* Publisher: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©1993

Tran V & Liou D (1997), *A Procurement-centric Model for Engineering Component-based Software Systems.* Proceedings of the 5th International Symposium on Assessment of Software Tools. Publisher: IEEE Computer Society Washington, DC, USA ©1997

Tucker A.B (2004), *Computer Science Handbook, 2nd Ed.* Publisher: Chapman & Hall/CRC ©2004

Vliet H (2000), *Software engineering Principles and Practice, 2nd Ed.* Publisher: John Wiley & Sons, Inc. New York, NY, USA ©2000

Wohlin C *et al* (2000), *Experimentation in Software Engineering: An Introduction.* Publisher: Kluwer Academic Publishers Norwell, MA, USA ©2000

Yang Q, Li J.J & Weiss D.M (2006), *A Survey of Coverage-Based Testing Tools*, *The Computer Journal Advance Access.* Proceedings of the 2006 international workshop on Automation of software test. Publisher: ACM New York, NY, USA ©2006

Appendix A – Online Survey

http://evari.sauropol.com/mcs-thesis-survey/

In Appendix A we present the online questionnaire which was sent to IT companies. The questionnaire consists of the following sections: (i) introduction; (ii) first part for software development processes; (iii) second part for TMT functions; (iv) third part for TMT artifacts; and (v) fourth part for the respondent company's background.

Evaluation for Test Management Tool

Introduction

Thank you for taking the time to respond to the survey!

The survey is a part of the master thesis at the University of Tartu. The topic is about Test Management Tools. The major goal of this work is to create an instrument, which would help to assess and aquire a suitable Test Management Tool.

To fill in this questionaire it takes approximately 15 minutes. All of the respondents will be granted with the access to the survey results. We will also provide you with the masters thesis report once it is presented and defended at the university.

The questionaire consists of 4 parts:

- Software development processes;
- Functions of a Test Management Tool;
- Artefacts of a Test Management Tool;
- Respondent's background.

The target audience for the survey is QA managers/team leads or project managers. The terminology used in the survey is explained with tooltips when moving cursor over the text.

Thank you in advance for your participation.

Sincerely,
Evari Koppel

I Software Development Processes

A software development process, also known as a software development life cycle (SDLC), is a structure imposed on the development of a software product. There are several models, each describing approaches to a variety of tasks or activities that take place during the process.

A Test Management Tool is a means for managing the software testing process. It can be a specifically designed aplication or it could be a system of many tools.

TMT should be compatible with:	Completely agree	Rather agree	Hard to say	Rather disagree	Completely disagree	I don't know
1. Waterfall model	○	○	○	○	○	○
2. V-model	○	○	○	○	○	○
3. Spiral-models	○	○	○	○	○	○
4. Agile models	○	○	○	○	○	○
5. [Optional] TMT should support a model which is not listed above (please enter as free text).						

II Functions of a Test Management Tool

Which testing functions should be implemented within the TMT?

	Completely agree	Rather agree	Hard to say	Rather disagree	Completely disagree	I don't know
6. Software Requirement Specification (SRS) review	○	○	○	○	○	○
7. Test planning	○	○	○	○	○	○
8. Test design	○	○	○	○	○	○
9. Test execution (software testing)	○	○	○	○	○	○
10. Requirements coverage reporting	○	○	○	○	○	○
11. Test execution progress reporting	○	○	○	○	○	○
12. Test execution results reporting	○	○	○	○	○	○
13. Defect reporting	○	○	○	○	○	○
14. Traceability between SRS, Test cases, Test execution and Defects.	○	○	○	○	○	○
15. [Optional] The activities a TMT should support should also include (please list the activities in free text).						

III Artefacts of a Test Management Tool

Which artefacts should support TMT?

	Completely agree	Rather agree	Hard to say	Rather disagree	Completely disagree	I don't know
16. Software Requirement Specification	○	○	○	○	○	○
17. Software Test Plan	○	○	○	○	○	○
18. Test Case	○	○	○	○	○	○
19. Software Test Library	○	○	○	○	○	○
20. Test Suite (Test Run)	○	○	○	○	○	○
21. Defects (bugs)	○	○	○	○	○	○
22. Reports	○	○	○	○	○	○
23. Test Library per Releases	○	○	○	○	○	○
24. [Optional] Some artefacts were not listed (please enter in free text).						

IV Respondent's background

These questions gather information to determine correlations between groups of respondents.

This section contains only optional questions.

25. Which Software Development Model does your company follow?				
Waterfall model	V-model	Spiral model	Agile model	Some other model (please specify)
☐	☐	☐	☐	

26. Which Test Management Tool(s) are you using?				
☐ In-house developed tool	☐ Test Link	☐ Test Rail	☐ QAComplete	☐ Quality Center
☐ Zephyr	☐ JIRA	☐ Bugzilla	☐ Office tools (MS Office, OpenOffice)	
Other (please specify)				

27. What is the size of your company (number of employees in all departments)?				
○ Up to 10 people	○ 11 to 25 people	○ 26 to 100 people	○ 101 to 500 people	○ More than 500 people

28. What is the size of your company QA personel (number of people)?				
○ Up to 5 people	○ 6 to 10 people	○ 11 to 25 people	○ 26 to 50 people	○ More than 50 people

29. Does your company produce software for international or domestic projects?			
○ Only for domestic market	○ Mostly for domestic market	○ Mostly for international market	○ Only for international market

30. Please provide your email address if you wish to receive the survey results:

31. Do you wish to add anything related to the Test Management Tools and the current survey?

Appendix B – Online Questionnaire Results

We present the online questionnaire results in Appendix B in the raw format. The survey questions have been listed horizontally with each row representing the answers from one respondent. Since none of the respondents provided their email, we have excluded this row from Appendix B. To keep the anonymity of the respondents, they have been listed in the table as R1..19.

	TMT should be compatible with ...					Which testing functions should be implemented within the TMT?										Which artefacts should support TMT?								Respondent's background					
	1. Waterfall-Model	2. V- model	3. Spiral models	4. Agile models	5. Not listed above	6. Software Requirement Specification (SRS) review	7. Test planning	8. Test design	9. Test execution	10. Requirements coverage reporting	11. Test execution progress reporting	12. Test execution results reporting	13. Defect reporting	14. Traceability	15. The activities a TMT should support should also include	16. Software Requirement Specification	17. Software Test Plan	18. Test Case	19. Software Test Library	20. Test Suite	21. Defects	22. Reports	23. Test Library	24. Some artefacts were not listed	25. Which Software Development Model does your company follow?	26. Which Test Management Tool(s) are you using?	27. What is the size of your company?	28. What is the size of your company QA personel?	29. Does your company produce software for international or domestic projects?	30. Do you wish to add anything related to the Test Management Tools and the current survey?
R1	5	5	5	5		2	5	5	5	4	5	5	5	5		4	5	5	5	5	5	5	5		Mitmete erinevate mudelite "segu"	qc; jira	5	5	3	
R2	1	2	4	4		3	4	1	1	3	2	2	4	3	Session Based Test Management	3	1	1	4	3	2	2	3		Probably some weird combination of spiral and agile. It's actually very difficult to define it.	inHouse	3	1	4	
R3	4	4	4	4		3	4	3	3	4	4	4	3	4		4	4	4	4	4	4	4	4	even if those listed above are created outside TMT it should be possible to link/trace them with TMT	Waterfall; V-model; Spiral; Agile	inHouse; QC; Jira; bugzilla; Office Tools; std UNIX + FOSS tools	5	1	2	
R4	4	0	4	5		5	5	4	4	5	4	4	5	3			5	5	4	3	5	5	4		Waterfall; Agile	Office Tools; Mantis	3	1	3	

R5	2	3	4	5		2	4	4	4	3	4	4	3	5	I'd love to see support for Session based testing and other not so formal testing approaches. Eg the option to create test charters instead of full test suites and to take notes on testing.	4	3	4	4	4	3	3	4		Agile	inHouse; QC; Jira	3	2	3	
R6	5	5	3	4		4	5	3	5	5	5	5	3	5	1.test execution per release would be helpful also 2.It would be great if it's possible to get overview about occupied QA personal, with reports or smth like this. It helps to plan QA people between different tasks.	5	5	4	4	5	3	4	5		Waterfall; Agile	QC; Jira; Office Tools; some testplans are also created in Confluence, easier to maintain it, but it's difficult to track coverage %	5	3	3	
R7	5	5	5	5		2	5	5	3	5	5	5	2	5		2	5	5	5	5	2	2	5		Waterfall; V-Model; Agile	inHouse; TestLink; QC; Jira; bugzilla; Office Tools	5	5	2	

R8	5	5	5	5		3	4	3	3	3	5	5	5	5		3	5	5	0	2	5	5	0		Waterfall; Spiral; Agile	inHouse; Jira; bugzilla; Office Tools; many different open source test management tools, bug trackers etc as is required by the customer or is suitable for the project	4	4	2	
R9	5	4	4	5		4	5	5	5		5	5	3	5		4	5	5	5	5	4	5	4		Waterfall; Agile	TestLink; Jira; Office Tools; trac; mantis	2	2	3	
R10	1	4	3	5		5	5	4	5	5	5	5	4	5		5	5	5	4	5	4	4	4		Agile	inHouse; Office Tools; Jira	4	4	2	
R11	3	4	4	5		2	5	5	5	2	5	5	1	4		2	5	5	4	5	2	5	4		Agile	inHouse; TestLink; Jira	4	3	3	
R12	4	4	3	5		2	4	4	5	4	5	5	3	5		4	5	5	5	5	4	5	4		Agile	Jira; inHouse	4		3	
R13	1	1	0	5		5	5	5	5	3	5	5	2	5		4	4	5	3	4	4	4	4		ganban	Jira; TestTrack; inHouse	4	3	4	
R14	0	0	0	0		5	5	4	4	4	4	4	4	0		5	5	5	3	3	4	3				bugzilla; Office Tools	3	1	3	
R15	5	0	0	5		4	5	5	4	5	5	5	5	4	Client feedback element - during a project we have clients tests and ask clients for feedback as well.	5	5	5	3	0	3	5	4		Waterfall; Agile	inHouse	3	1	2	

R16	4	4	4	4		3	5	5	5	5	5	5	5	5		3	5	5	5	5	5	5	5		Waterfall; Agile	TestLink; Jira	3	2	1	
R17	4	3	2	4		2	2	1	4	4	5	4	4	4		4	4	3	3	3	4	4			Waterfall; Agile	No tools	3	1	2	
R18	5	5	5	5		4	5	5	5	5	5	5	5	5		3	5	5	5	5	5	5	5		Agile	inHouse; Office Tools	2	1	3	
R19	2	5	4	5		3	4	5	5	5	5	5	5	5		4	3	5	5	5	5	5	5		V-model	Team Foundation Server (TFS); MANTIS	3	1	2	

Appendix C – Test Management Tool Requirements Framework Product Diagrams

Appendix C lists 13 TMT product diagrams which are specific per companies' characteristics. These product diagrams are used in the TMT evaluation framework to form company specific TMT product diagram.

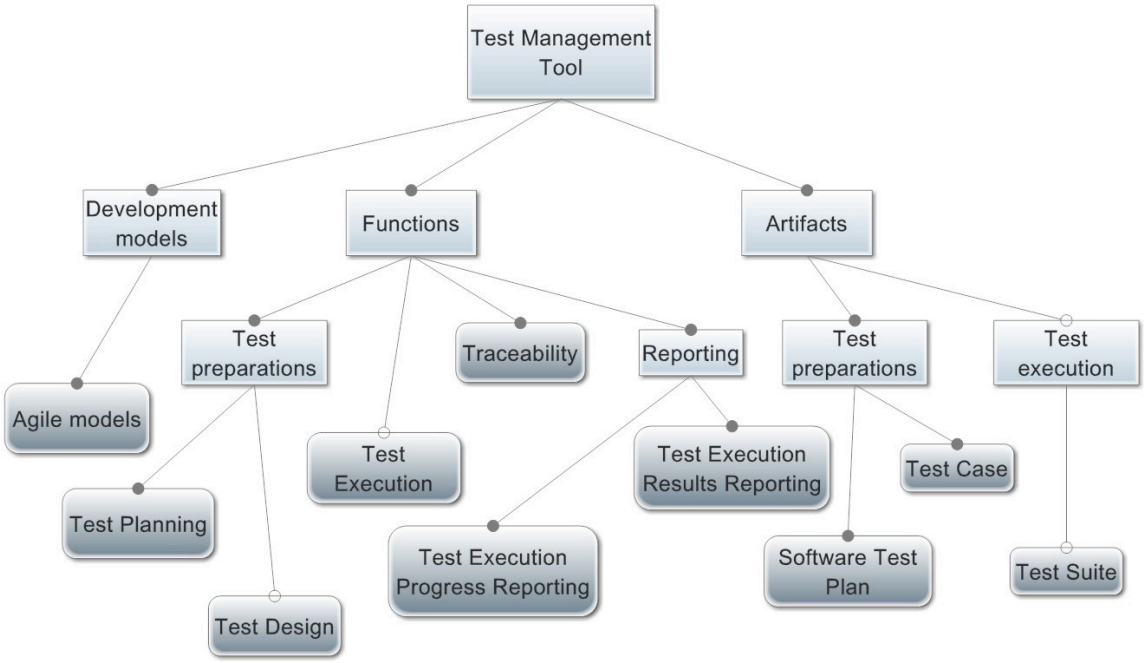

Figure 1. PD for agile models

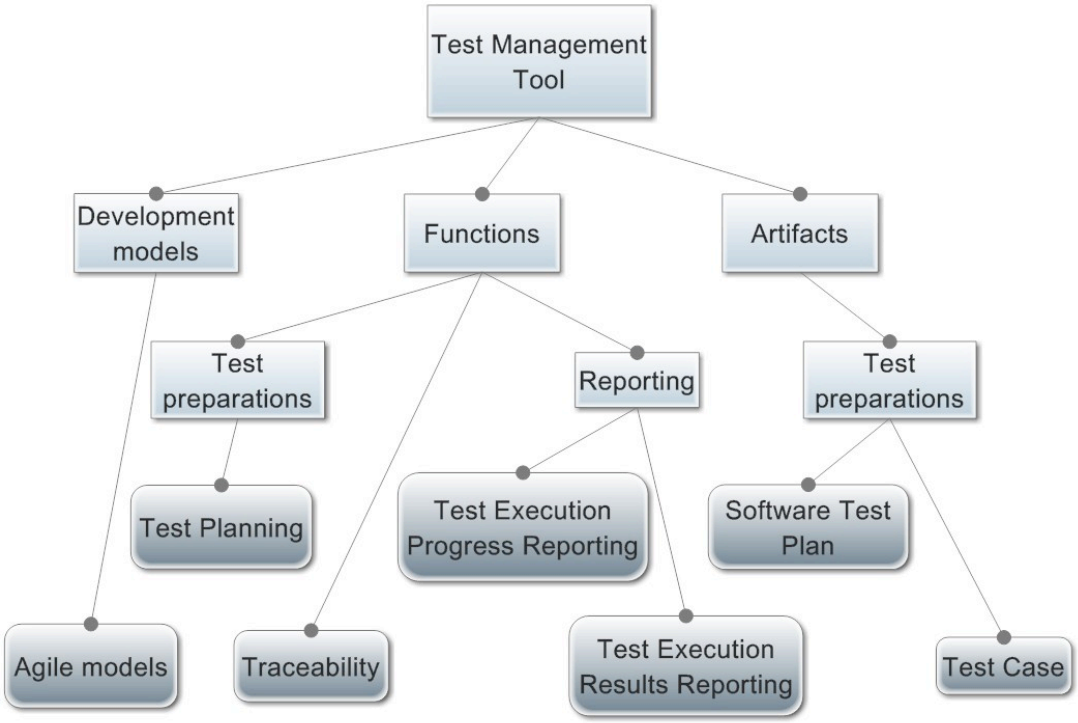

Figure 2. PD for mixed models

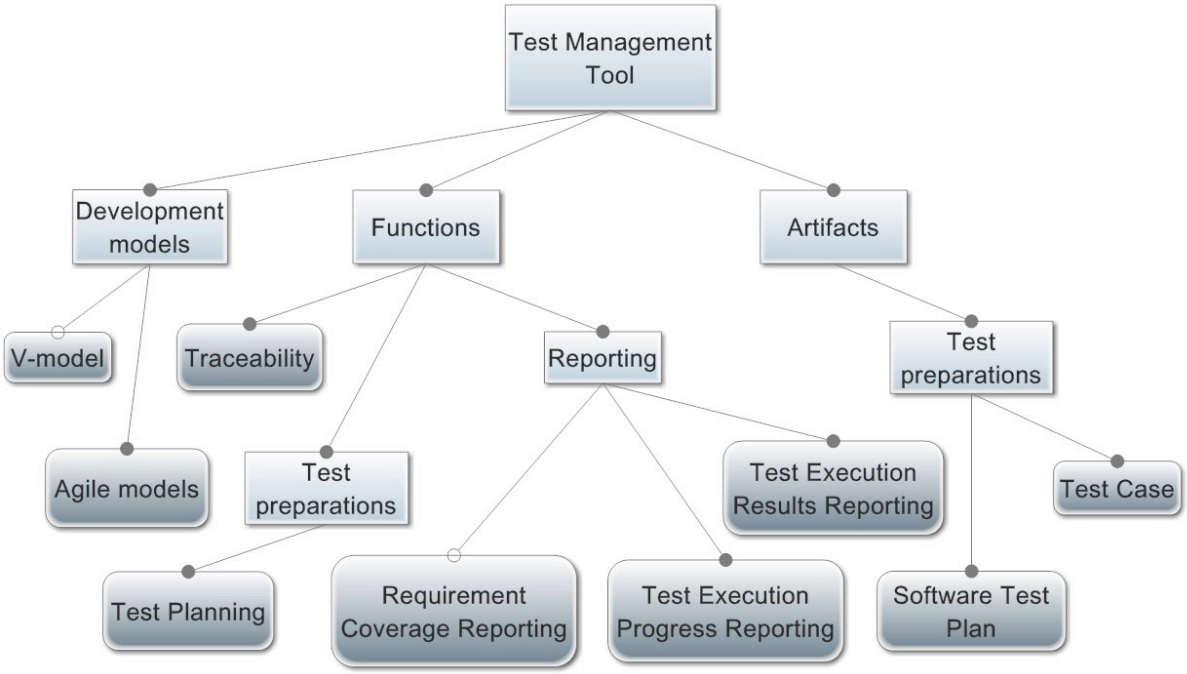

Figure 3. PD for mostly domestic market

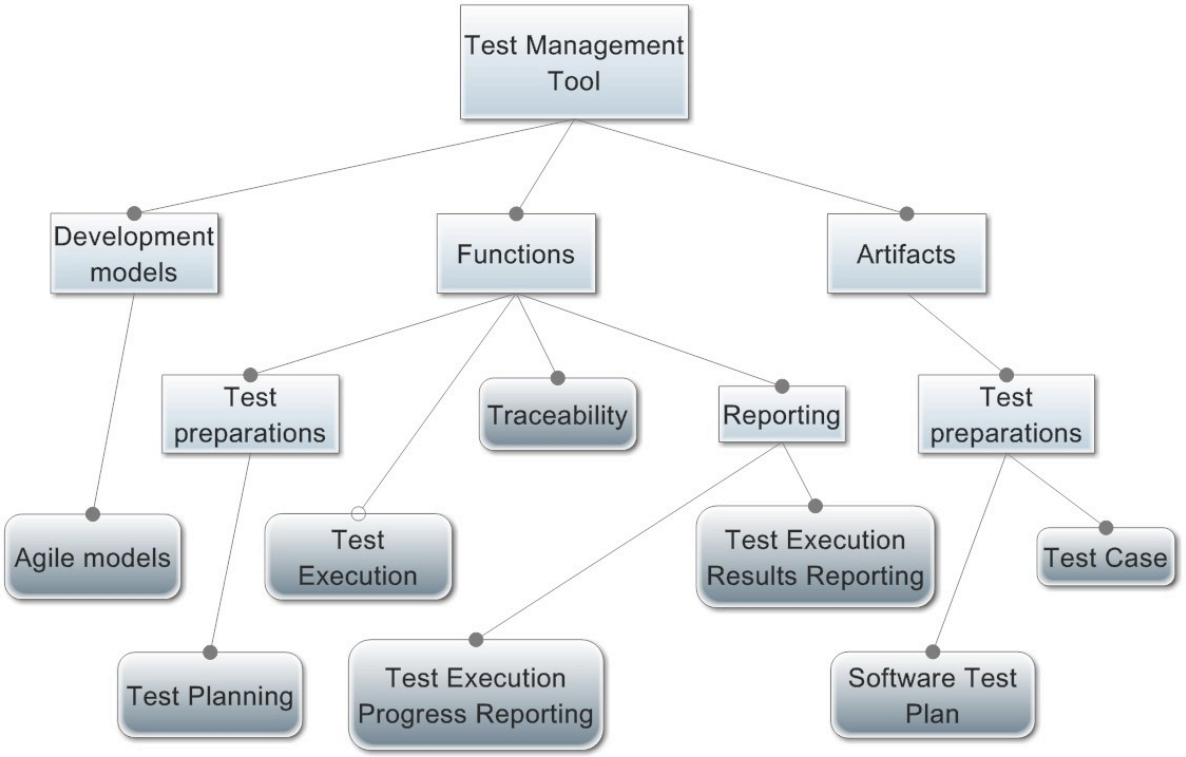

Figure 4. PD for mostly international market

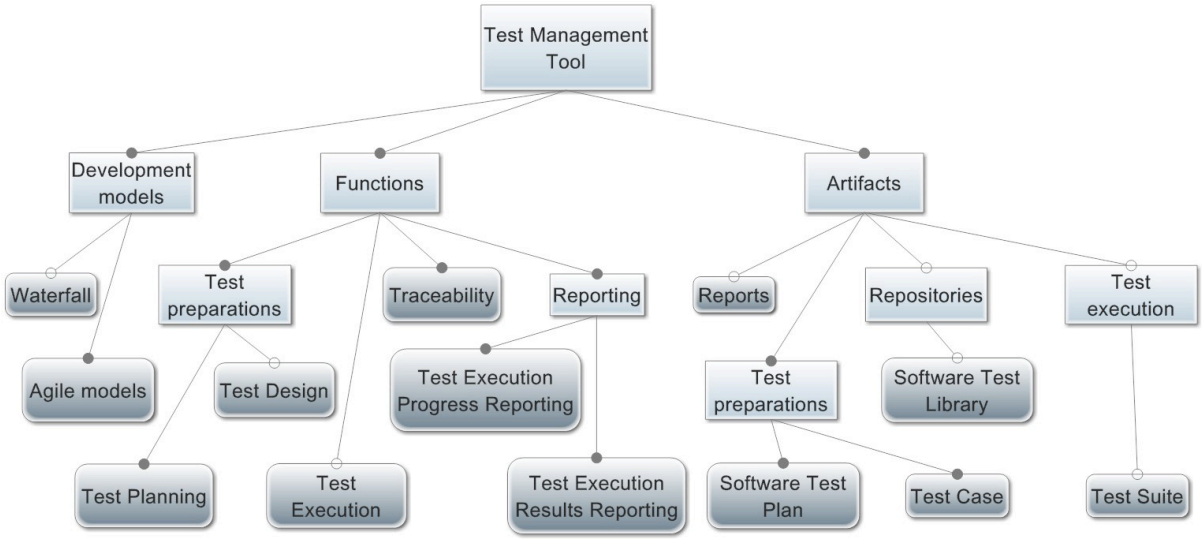

Figure 5. PD for compny size up to 25 people

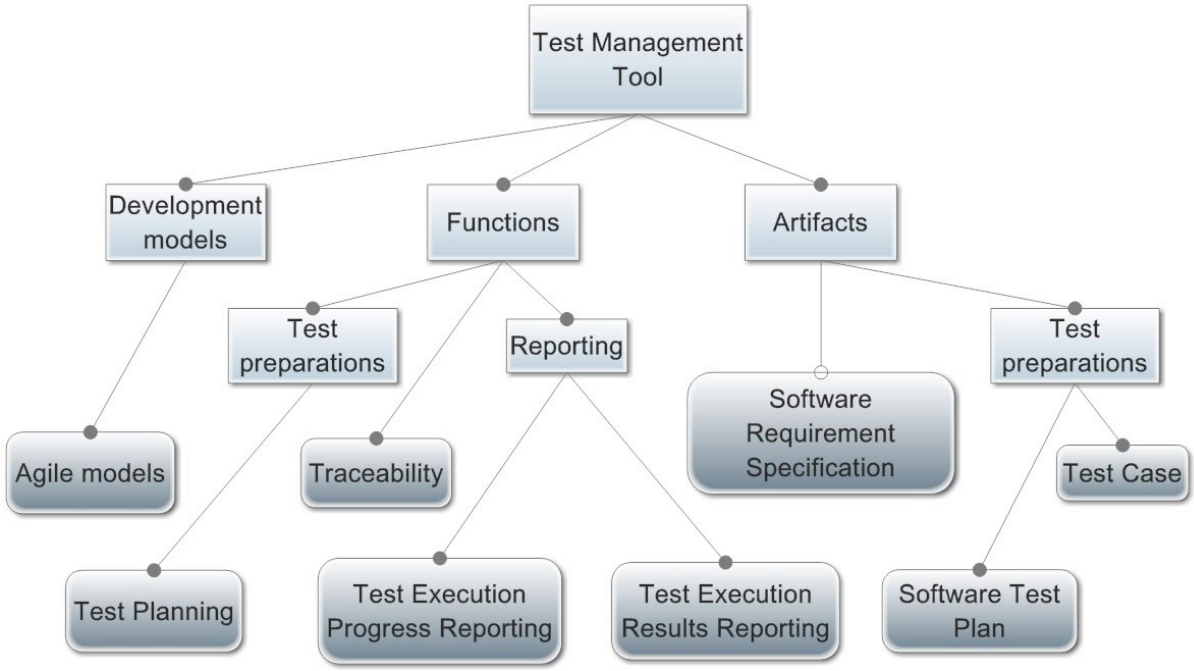

Figure 6. PD for compny size up to 100 people

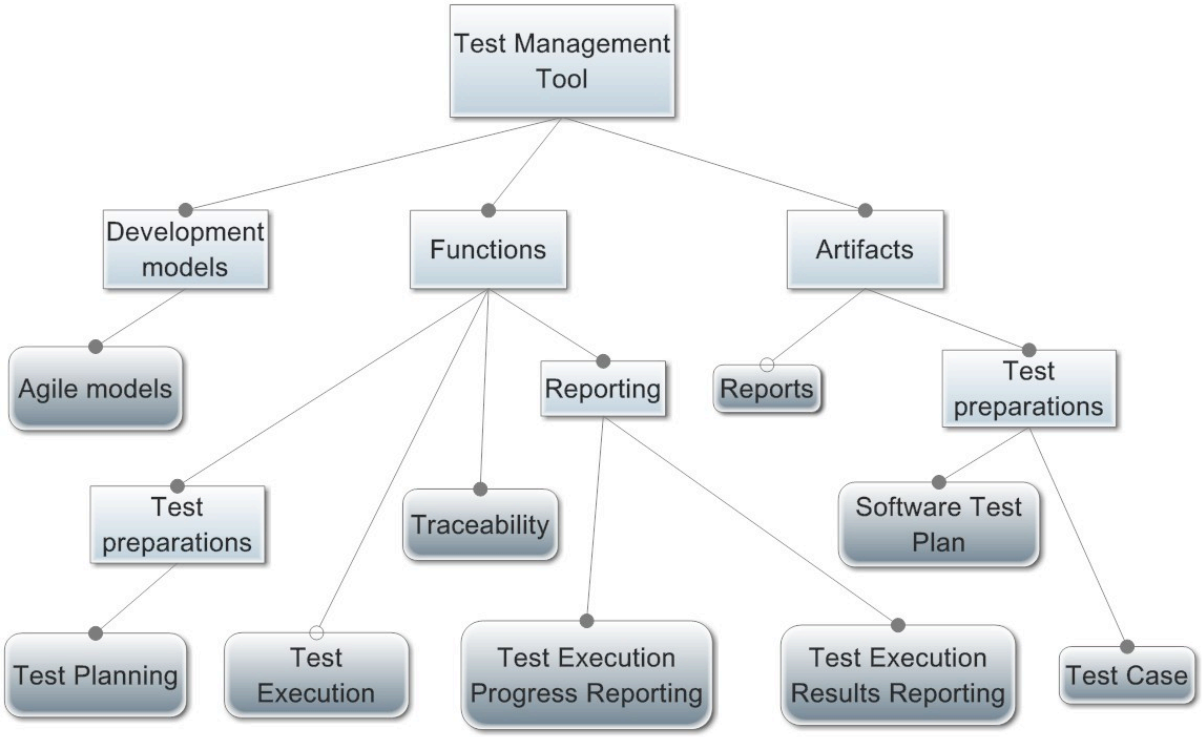

Figure 7. PD for compny size up to 500 people

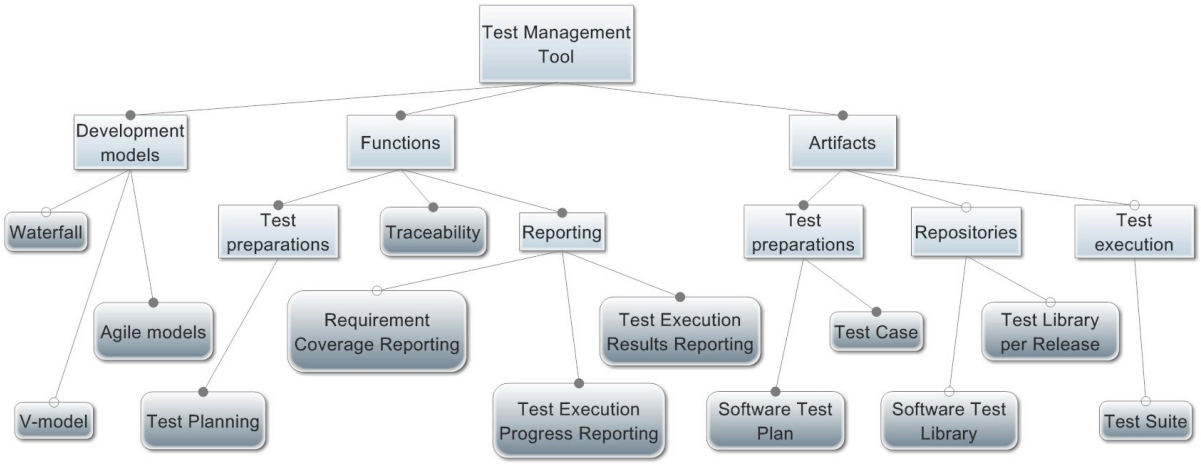

Figure 8. PD for compny size more than 500 people

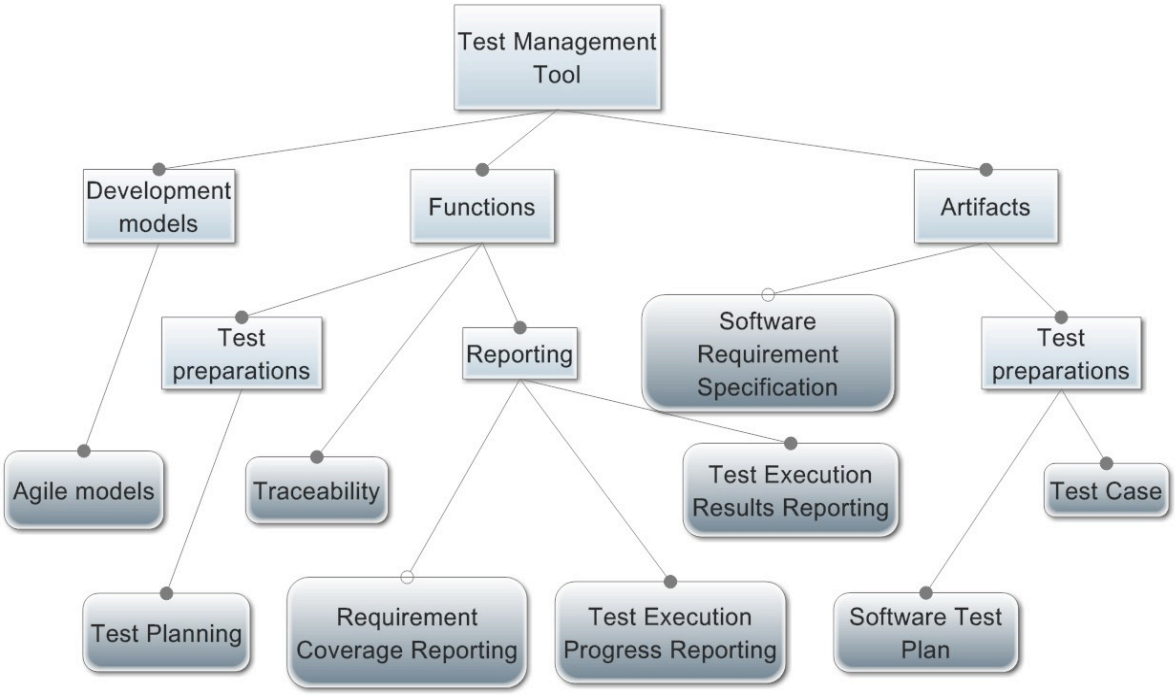

Figure 9. PD for QA size up to 5 people

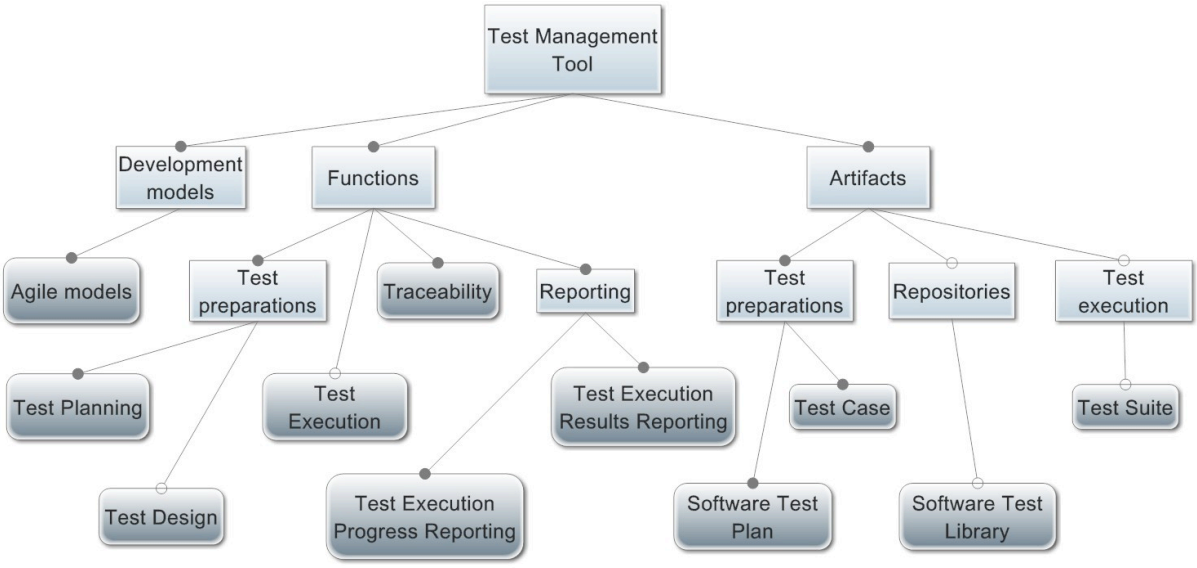

Figure 10. PD for QA size up to 10 people

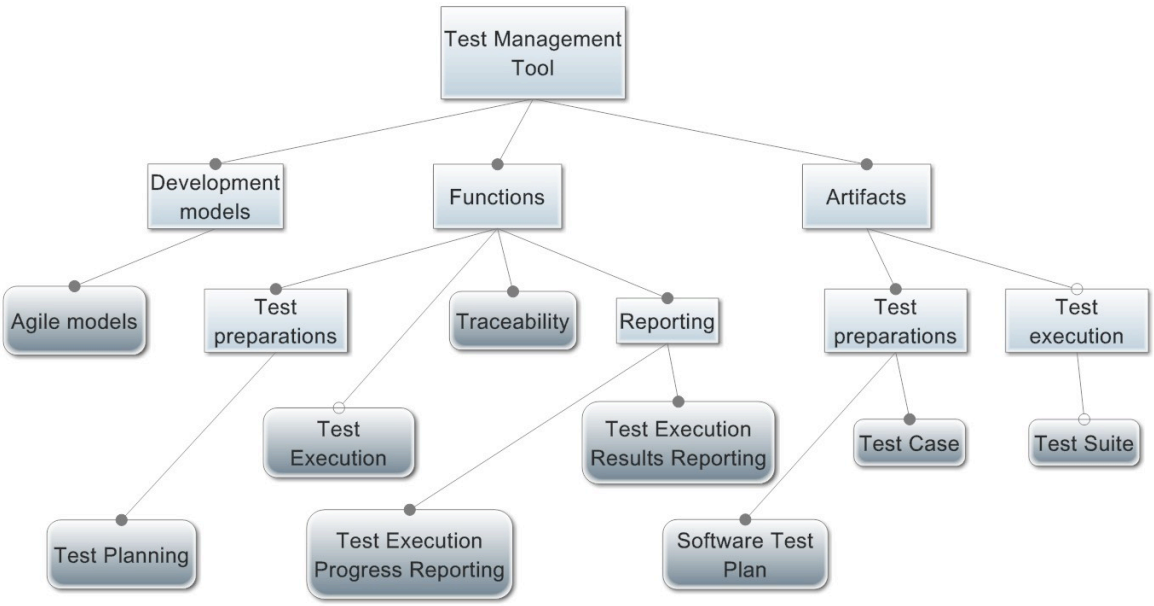

Figure 11. PD for QA size up to 25 people

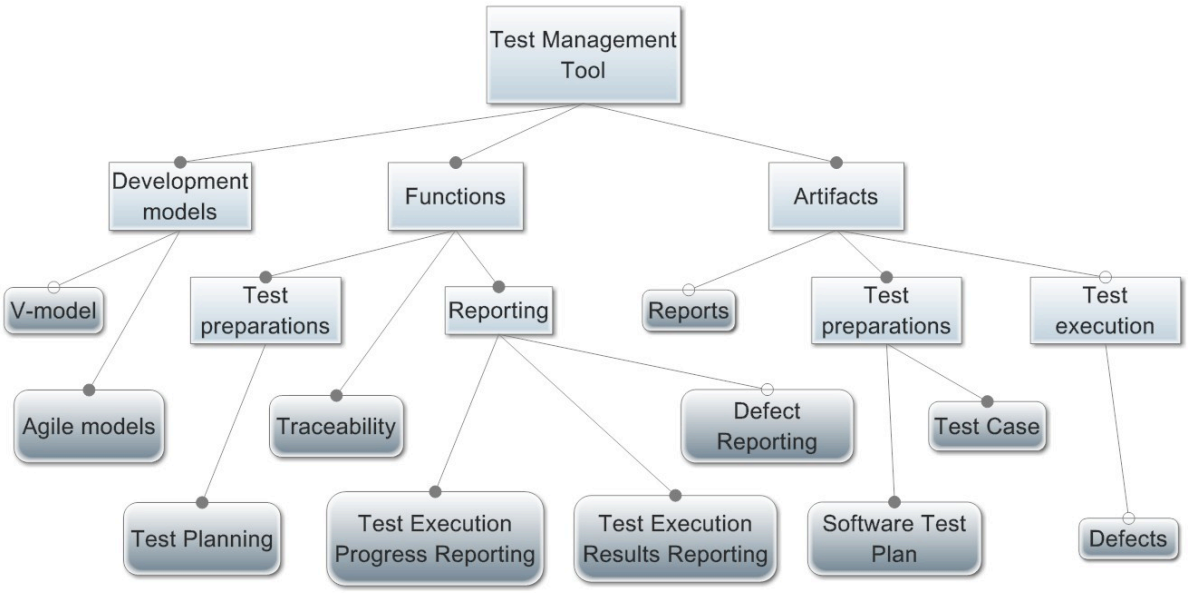

Figure 12. PD for QA size up to 50 people

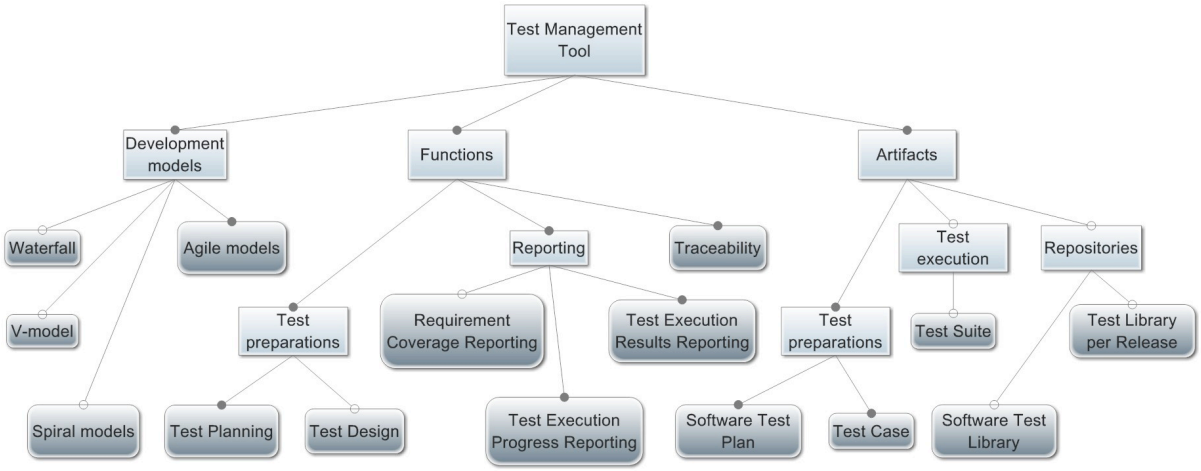

Figure 13. PD for QA size more than 50 people

Appendix D – Guideline for Using the Test Management Tool Evaluation Framework

The test management tool evaluation framework has been created to provide Estonian software developing companies' test managers and engineers with the means to give objective evaluations. Personal evaluations which are done based on best practices, company knowledge and the person's experience, are still subjective and highly dependent on the opinion of a single person. The TMT evaluation framework is created based on academic study, market research and online survey carried out among Estonian IT companies, thus applying the general expectations to test management tools depending on the company characteristics. Our approach has mitigated the subjectivness of an evaluation.

When speaking about evaluating test management tools, there are several possible options. One of them is counting the number of steps/clicks the evaluator has to make before the test management tool activity or artifact is complete. Another option is to measure the time it takes to create an artifact or perform an activity. The usability of the test management tool is yet next measurement. However in this case, the evaluation should be carried out by several specialists in order to mitigate the subjectivity of the evaluation. Regardless of the method chosen, the framework expects the evaluator to rate test management tool features on a scale of 0 to 3. To illustrate it, we will now provide the guidelines for performing an evaluation with the framework (Figure 1).

In the framework there are six steps the user should follow. When using the framework repeatedly, the first three steps can be skipped, since the company characteristics and the company specific product diagram will not change. The six steps of the framework are:

- define evaluation purpose;
- determine company characteristics;
- create TMT product diagram;
- evaluate mandatory and optional features;
- record the evaluation results
- make conclusions based on framework formula.

First step – define evaluation purpose. The evaluator should define the evaluation purpose – either to understand the effectiveness of the existing test management tool (TMT) or to perform evaluation for procuring new TMT. There could be even further reasons. Depending on the purpose, the set of TMTs to evaluate will vary and longer time should be planned for this activity.

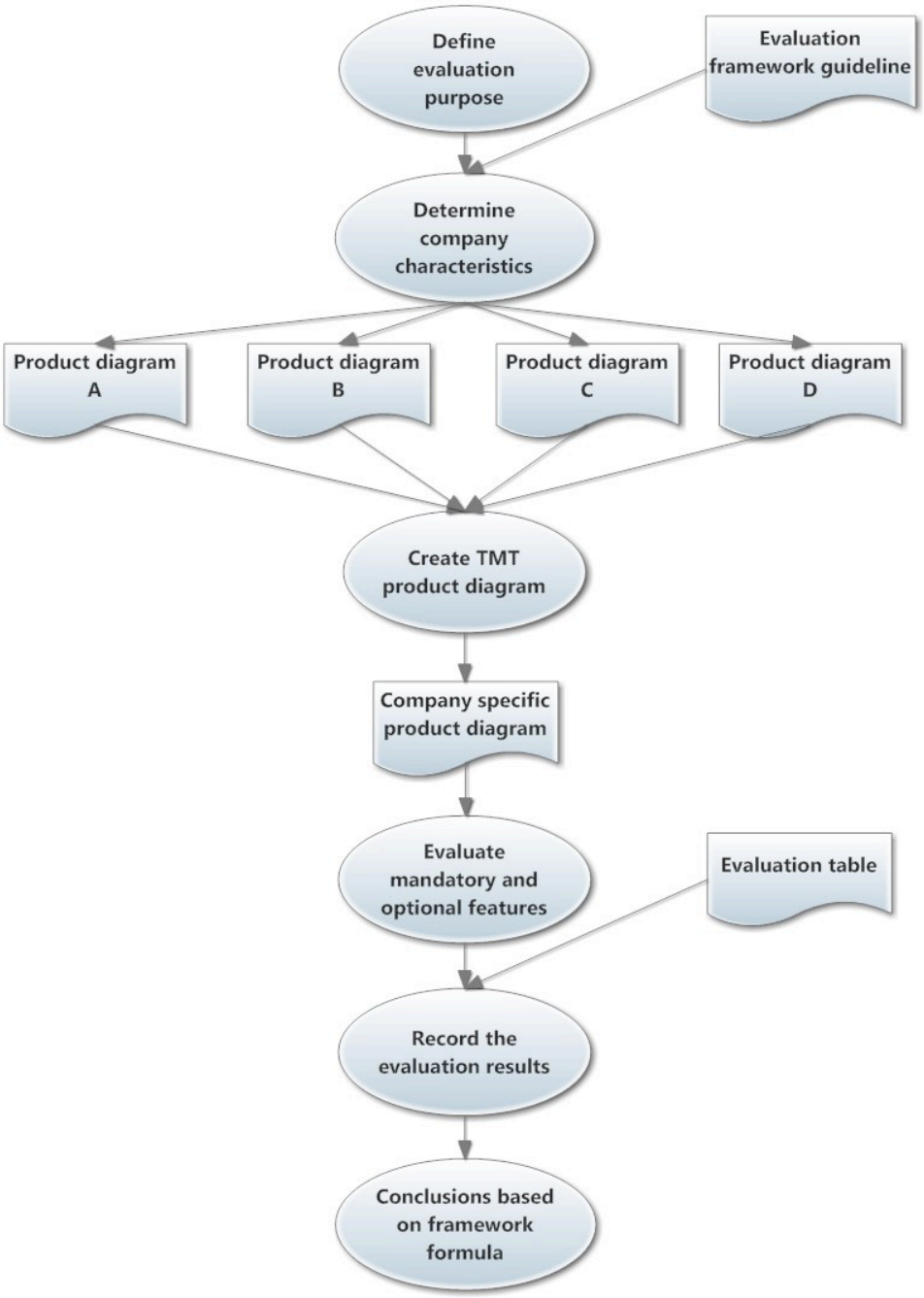

Figure 1. TMT evaluation framework processes

Second step – determine company characteristics. The environment should be identified where the evaluation is carried out. This is done by determining the company four characteristics: the company size, QA personnel size, the used development model and the target market for company's products. Based on these, the person can take the specific product diagrams from Appendix C. If the information about the company is un-known, then the general TMT feature diagram should be used. For example, if the company had up to 10 QA persons *and* the company size were up to 100 persons *and* the company used only agile development model *and* developed mostly for domestic market, then following product diagrams should be used: Appendix C Figures 1, 3, 6 and 10.

Third step – create TMT product diagram. The next step is to overlap the product diagrams. Overlapping means in other words taking all features from one product diagram and adding then to the next diagram but not duplicating any of them. By doing this, we will receive a new company specific product diagram. For example the overlapping of Appendix C Figures 1 and 3 would result in the following product diagram.

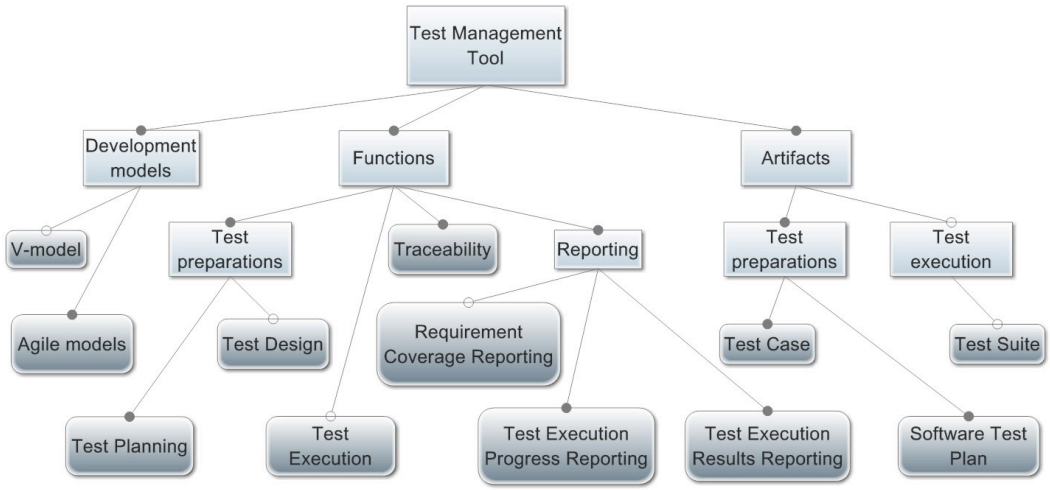

Figure 2. Company specific TMT product diagram – agile and mostly domestic market

After applying all four product diagrams, there are still 7 mandatory features but in addition, there are also optional[19] features. In our example, the overlapping results are shown on a product diagram, Figure 3. The seven optional features of a TMT are:

- (support for) V-model;
- Test design;
- Test execution;
- Requirement coverage reporting;
- Software requirement specification;
- Software test library;
- Test suite.

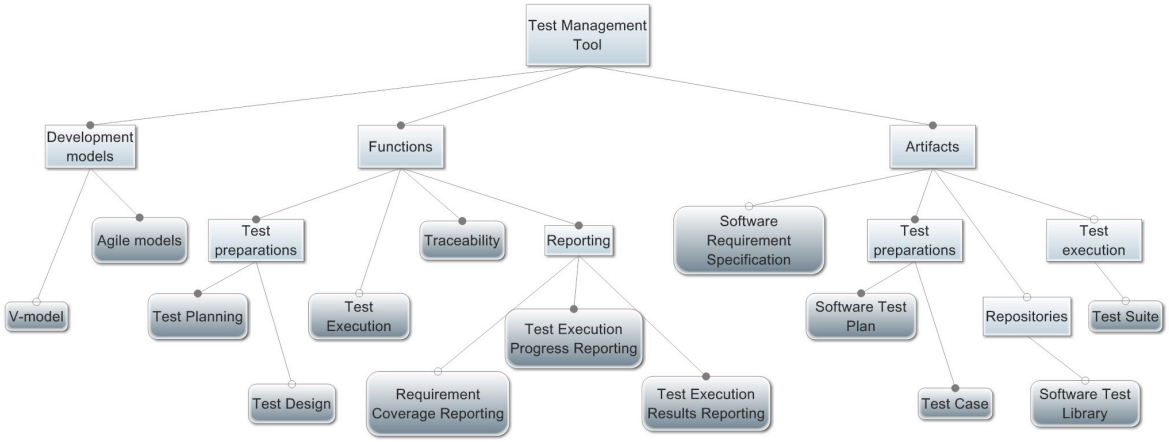

Figure 3. Company specific TMT product diagram

[19] We will use here the term 'optional feature' since the product diagrams are using this nomenclature. However from testing perspective, these features are also required.

Fourth step – evaluate mandatory and optional features. The evaluator would have to measure how well the listed features are supported by the TMT. As we described earlier, there are several options. Regardless of the chosen method, each mandatory and optional feature would get a value between 0 and 3. The meaning of the values is:

- 0 – does not meet the expectations;
- 1 – mostly does not meet the expectations;
- 2 – mostly meets the expectations;
- 3 – completely meets the expectations.

Fifth step – record the evaluation results. In the TMT evaluation framework we use tabular format. The first column lists all TMT mandatory and optional features, the second column holds the value given to the feature. Finally, the last row totals all the results. The Table 1 in our example provides arbitrary values as reference how to fill it.

Table 1. TMT mandatory and optional features evaluation table

TMT feature	**Value**
V-model	1
Agile models	3
Test planning	1
...	...
Test suite	2
Total	7

Sixth step – conclusions based on framework formula. The final step is to draw the conclusions. In order to understand to which degree the TMT meets the company expectations we use formula: $p= \frac{\sum_1^n result_i}{n*3} *100\%$ where result i is the value a feature received, n is the total number of features identified from the company TMT product diagram. The number 3 is the highest value which can be given. Following these provided guidelines, the evaluator will receive result showing the percentage of how much the tool meets the expectations of the company.

Definitions

The models and requirements for TMT under observation are:

- Waterfall model;
- V-Model;
- Spiral models;
- Agile models.

The second set of requirements is the functions a TMT should be capable to support. From the earlier analysis we have identified that these are the most expected functions for TMT:

- Software requirement specification review – the ability to review and provide feedback to the SRS document;
- Test preparations:
  - Test planning – the process of setting the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. It also includes resource allocation and defining any dependencies for the testing activities;
  - Test design – the act of creating and writing test cases and test suites for testing software;
- Test execution – the process of running the tests;
- Traceability – the ability to trace the linkage between requirements, test plans, test cases, test suites and defects;
- Reporting of:
  - Requirement coverage – the possibility to provide reports showing the requirement coverage by test cases or executed test suites;
  - Test execution progress – the ability to report how many test suites have been run;
  - Test execution results – the TMT support for reporting the results of particular test suites;
  - Defect – the ability to use the TMT for notifying managers, developers or other project stake-holders of found problems in the software.

The third and final set of requirements is the artifacts a TMT should support. We will list them below and provide their meanings:

- Software requirement specification – a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software;
- Test preparation artifacts:
  - Software test plan – an artifact documenting the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements;
  - Test case – a test script detailing the steps and expected results of the system behavior;
- Repositories of:
  - Software test library – a repository containing all created test cases and test suites. The repository is often extended to other TMT artifacts as well;
  - Software test library per release – a repository containing all created test cases and test suites grouped by product releases. This repository is more often used in spiral and agile software development models;
- Test execution artifacts:

- Test suite – a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviors;
- Defect – an error, flaw, mistake, failure, or fault in a computer program which is recorded by the tester. The TMT should support creation and managing this artifact;

- Reports – various documents covering different aspects of the software testing activities. The TMT support for creating and storing the reports has been identified as a requirement.

Evaluation Table[20]

TMT feature	**Value**
Supported development models	
Waterfall model	
V-model	
Spiral models	
Agile models	
Supported TMT activities	
Software requirement specification review	
Test planning	
Test design	
Test execution	
Traceability	
Requirement coverage reporting	
Test execution progress reporting	
Test execution results reporting	
Defects reporting	
Supported TMT artifacts	
Software requirement specification	
Software test plan	
Test case	
Software test library	
Software test library per release	
Test suite	
Defects	
Reports	
Total	

[20] Fill in only the features shown on the company specific TMT product diagram

Appendix E – Research Paper Submitted to Conference

Koppel E & Matulevičius R (2012), *An Evaluation Framework for Software Test Management Tools.* Submitted ©2012

An Evaluation Framework for Software Test Management Tools

Evari KOPPEL and Raimundas MATULEVIČIUS
Faculty of Mathematics and Computer Science, University of Tartu,
Liivi 2, 50409 Tartu, Estonia
evari@ut.ee, rma@ut.ee

Abstract. Software testing has proven its value for software development increasingly over the last decade. With the recognition of the benefits of software testing, several software test management tools (TMT) have emerged on the market. Although there exist different approaches, there is no method for a systematic TMT assessment. This is a problem because to our knowledge, evaluating TMT is rather a subjective task, heavily depending on the evaluators' opinions rather than based on the objective approach. The same problem applies when test managers are asked to evaluate whether their currently used TMT meets the company's expectations. In this paper based on the survey performed among Estonian testing practitioners, we deliver a TMT evaluation framework. The paper applies structured approach by performing a literature study on software testing processes, existing TMT market research, and mapping together the identified test activities and test artifacts. The results help formulate and design the online questionnaire and perform a TMT survey in the Estonian IT companies. Based on this survey results, a framework for evaluating TMT software is created. Such a framework could potentially help companies to measure the TMT suitability to company's goals and to decrease subjectivity of the TMT assessment. The framework also provides test and project managers the understanding whether their current TMTs meet the company's expectations.

Keywords. Test Management Tools, Quality Assurance, Testing.

Introduction

As software quality assurance procedures and tools are evolving, one of the means that could potentially help with software testing is the computerized test management tools (TMTs). Although there exist different free-ware and commercial TMTs [10], choosing a proper TMT to support testing activities is rather a complex task. In this paper we address this issue by suggesting a *framework* to assist software testers when acquiring their *best* TMT. Such a TMT could potentially support software testers when reviewing software project requirements, developing test cases, and generating test reports depending on the contextual/organizational settings. Our proposal is developed from the empirical survey executed in the Estonian ICT organizations by investigating TMT expectations and requirements within such companies.

The paper is structured as follows. In Section 1 we give a background on the testing activities, artifacts and existing TMTs. Section 2 describes our survey within the Estonian ICT organizations. Section 3 proposes the TMT evaluation framework and in Section 4 we describe how 5 software testers were interviewed to learn about the

framework usability. Finally, in Section 5 we conclude our paper of envision some future work.

1. Theory model

1.1. What is Software Testing?

Literature gives different definitions for a term *testing*. It is often perceived as a 'magic bullet' [11] that will solve the problem of finding errors after product has been delivered to the customer. *Testing* aims to execute a software-intensive system or parts of the system in a controlled environment and under controlled circumstances in order to detect deviations from the specification, and to check whether the system satisfies the defined acceptance criteria [1]. Software testing should cover both internal and external expectations or in other words it is a part of software quality assurance (QA). QA is a formal approach for automated regression testing, configuration management, versioning, profiling, and controlling releases hypothetically leading to zero defects [2].

Software testing life cycle is part of the Software Development Life Cycle (SDLC). It defines a set of stages outlining what test activities to perform and when to conduct them. Typically these stages are planning, analysis, design, construction, testing, final testing and implementation, post implementation [3]. The main functional testing levels during the development process include component testing (also referred as unit testing), integration testing and system testing [1]. Two other levels can be identified based on the objective: there are regression testing and acceptance testing.

1.2. Activities and Artifacts Produced in QA

Software development is a process that can follow different models. It is not unilaterally defined and it may vary from company to company. In this paper we have examined *waterfall*, *V-model*, *spiral* and *agile* software development models and their testing levels. Our purpose was to identify QA activities and artifacts, which could be required in a test management tool. Our observations are summarized in Table 1.

On the horizontal axes we list the selected software development models. Software development stages are brought out vertically. The content of the table identifies QA activities and artifacts produced by them in each particular combination. Out of *activities*, the requirements' inspection is performed in the user requirement gathering and evaluating stage. This is present in all except the waterfall model. The same can be said for the test design creation, which is performed in software design stage. Test planning and resource allocation, which is performed in parallel with software requirements specification, is carried out only in V and Spiral model. In the Agile model, this is done only if required by stakeholder. Test execution is carried out in all of the models during the integration and verification stage, within the Spiral and Agile model it is done iteratively. The only models that describe QA activities after the launch of the software are Spiral and Agile, where it is referred as iterative cycles for improving the existing product [4].

Next we examine the testing *artifacts*. In the user requirements definition stage the result is a review report, which is present in all except the waterfall model. In the same models in the software design stage, test design is created. The test plans are created during the software requirement specification stage though waterfall and agile models

have a slight variance. In agile model, the test plans are created only when stakeholders require it. In case of the waterfall model, all artifacts are created in the integration and verification stage. The artifacts produced during the verification stage for all models are test reports, defects and defect reports. Last stage of the software development cycle, maintenance stage, brings new entities only with spiral and agile models. These models represent the iterative nature of providing new versions to existing software.

Table 1. QA activities and artifacts (N/A – not available)

Approaches / Soft. Dev. stages	Waterfall	V-model	Spiral-model	Agile model
User requirements definition	N/A	Requirements inspection	Requirements inspection	Requirements inspection
	N/A	Requirements review report	Requirements review report	Requirements review report
Software Requirements Specification	N/A	Test planning	Test planning, QA resource allocation	N/A
	N/A	Test Plan	Test Plan	N/A
Software design	N/A	Test design creation	Test design creation	Test design creation
	N/A	Test Design	Test Design	Test Design
Integration and verification	Test planning, Test execution, Reporting, Test Plan	Test execution, Test report	Iterative cycles, Test execution, Reporting	Iterative cycles, Test execution, Reporting
	Test Design, Test report Defect, Defect report	Defect, Defect report	Test report, Defect, Defect report	Test report, Defect, Defect report
Maintenance and operation	N/A	N/A	Post launch defect fix testing	Post launch defect fix testing
	N/A	N/A	Test report	Test report

This comparison of the QA activities and artifacts in different software life cycles provides us the basis for evaluating various supporting commercial TMTs. In the following section we will have a look on the current market support.

1.3. Survey of Existing Tools

There are different TMTs [10], which (as their vendors claim) support each and every QA activity. Analyzing all TMTs in this survey would not be prudent, thus, based on the recommendations of TMTs users [12] we select 4 tools: TestLink, TestRail, HP Quality Centre and QA Complete. Their support to various activities and needed artifacts differs from vendor to vendor. When comparing the solutions offered to the QA activities described in software development models, then there are gaps. Yet still, the tools are being used by IT oriented companies and, thus, there is more than meets the eye. In Tables 2 and 3 we provide our findings of the support TMTs offer.

To summarize, although we have analyzed relative small amount of TMTs, we think that our observation is quite reliable since we identify rather broad coverage of testing activities by the TMT's functionality. Regarding functionality of the individual tool, however, we could also see several limitations to support one or another test activity. This observation motivated us to understand what the criteria for tool acceptance in a company are.

Table 2. TMT support for QA activities (N/A – not available)

TMTs / Soft. QA stages	Test Link	TestRail	HP Quality Centre	QAComplete
Requirements analysis	Creation of requirements, Traceability, Requirements' review	N/A	Creation of requirements, Requirement division per versions, Traceability, Requirements' review	Creation of requirements, Requirement division per versions, Traceability, Requirements' review
Test plan preparation	Creation of test plans, Traceability	Setting of project milestones for QA activities	Test cases are managed as library, Creation of test-set folders/test planning	N/A
Test design stage	Creation of test cases, Traceability	Creation of test suits (cases), Traceability, Searching for existing test design and suites from previous versions	Creation of test cases, Traceability, Test case reviews, Test run preparations and resource assignment, Searching for existing test design from previous versions	Creation of test cases, Traceability, Test case reviews, Test run preparations and resource assignment, Searching for existing test design from previous versions
Test execution	Test execution	Test execution Defect reporting	Test execution, Defect reporting, Grouping of tests per testing stage	Test execution, Defect reporting, Grouping of tests per testing stage
Revision and reporting	Test results driven reporting, Test coverage driven reporting	Test coverage driven reporting	Requirement coverage driven reporting, Test results driven reporting	Requirement coverage driven reporting, Test results driven reporting

Table 3. TMT support for QA artifacts (N/A – not available)

TMT / Soft. QA stages	Test Link	TestRail	HP Quality Centre	QAComplete
Requirements analysis	Software require-ments	N/A	Software requirements, Requirement coverage report	Software requirements, Requirement coverage report
Test plan preparation	Test suites	Milestones for QA activities	Test case library, Test case folders per stages	N/A
Test design stage	Test cases	Test cases, Test suites with assigned resources	Test cases, Test runs with assigned resources	Test cases, Test runs with assigned resources, Test case change history
Test execution	Test run with status	Test run with status, Defects	Test run with status Defects	Test run with status, Defects
Revision and reporting	Test report	Test report	Test report, Printable test report, Test execution progress reports	Test report, Test execution progress reports

1.4. Motivation for an Evaluation Framework

Selecting and acquiring a proper TMT is complex task. Typically evaluators need to assess tools for procurement and consider how tools could support different test activities and contribute with test artifacts. A potential help to assess the TMTs could be received from the general *tool selection approaches* (see extensive survey in [5]). Although being useful for the general commercial of-the-shelf products, these selection approaches still remain abstract when one needs to evaluate the TMTs.

Thus, in this paper we specifically target the TMT domain and develop a TMT evaluation framework based on the results received from the empirical study of the Estonian ICT companies.

2. Survey

The goal of the survey is to understand software tester's needs for the TMT and, based on the received results, to develop the evaluation framework for the TMT acquisition. The survey includes the following four research questions:

- Q1: Which software development model the TMT should support?
- Q2: Which TMT functions are important for a software developing company?
- Q3: Which TMT artifacts are important for a software developing company?
- Q4: How are the TMT functions and artifacts related to the characteristics of a software developing company?

2.1. Survey Participants and Questionnaire

The survey respondents were 15 Estonian ICT companies. The participants were, first, contacted by phone and their willingness to participate was confirmed. Then the questionnaire URL was sent to them by email giving a four-week response time.

The survey questionnaire was defined according to the review of the software development models (Table 1) and existing TMTs (Tables 2 and 3). By combining these results we have developed a web-based questionnaire consisting of 31 questions.

The first part of the questionnaire asks how important the TMT support for software development models is. Four models are given: Waterfall, V-Model, Spiral and Agile. A fifth option is a free field in case for some other model.

The second part enquires about the various testing activities (such as test planning, design, execution, defect reporting). The respondents are also provided with an open-field answer in case not all activities are listed. The term definitions are also provided.

In the *third* part, the questionnaire inquires about the TMT expected support for testing artifacts.

The last part targets the information about the respondents and their organization. These data are used to find any correlations between similar company characteristics and their responses. The information includes the TMT the organization is currently using, employee number in organization and in QA department, information (*i.e.*, domestic or international) on market type. This section also contains open comments field for the respondents to provide feedback in free form regarding the survey.

2.2. Data Analysis Method

The survey results, firstly, were analyzed using visualizing the trends and determining any outliers [6]. For each outlier we analyzed whether data reduction can be applied based on the characteristics of the other responses. Finally the results were interpreted and used to form the evaluation criteria for the TMT assessment framework.

Then to analyze data collected during the survey we have applied descriptive statistics method [6]. The data is grouped similarly as the questionnaire into 4 sections: TMT expected compatibility with different software development models, TMT supported functionality, TMT supported artifacts, and respondent background. All collected data are presented using tabular format since such presentation is considered to be best [7] when the questions can have three to five possible outcomes.

The matrix columns list survey questions, and each respondent and their answers are written on each row. For each data set, we also calculated the mean, median, mode

and standard deviation to provide further insight to the responses [8]. In this paper, we focus on the results of mean and median values.

Finally the trends based on company background are analyzed. The results are visualized again on histograms, grouped by company size, QA team size, used development method and the target market of the company products.

2.3. Generic Results

To process the questionnaire results and determine the expected requirements of TMTs, we set a threshold of 4.4. The number means, that more than every third respondent should „completely agree" that TMT should support the attribute while no-one responds with lower than „rather agree". This figure is later used to highlight the most important attributes of TMT for particular focus group and it is used in the TMT procurement framework.

The *first section* of the questionnaire asked which software development process TMT should support (Table 4). The Agile model is perceived as a model that TMT should definitely support. The lower mean for V and Waterfall models indicates lesser expectations for these models.

Table 4. TMT should be compatible with following software development models

	Waterfall model	**V-model**	**Spiral models**	**Agile models**
Mean	3,57	3,83	4	4,79
Standard deviation	1,84	1,94	1,78	1,3
Mod	5	4	4	5
Median	4	4	4	5

The *second section* of the questionnaire asked which testing functions should be implemented within the TMT. Interesting aspect that is worth highlighting is that the most valued aspects were test planning, reporting and traceability. Two TMT functions received noticeably lower values – software requirements specification review and defect reporting. The statistics for the whole target group can be found from Table 5.

Table 5. Which testing functions should be implemented within the TMT?

	SRS review	**Test planning**	**Test design**	**Test execution**	**Requirement coverage reporting**	**Test execution progress reporting**	**Test execution results reporting**	**Defect reporting**	**Traceability**
Mean	3,400	4,667	4	4,067	3,929	4,533	4,533	3,467	4,500
Standard deviation	1,242	0,488	1,134	1,163	0,997	0,834	0,834	1,246	1,373
Mod	2	5	5	5	5	5	5	3	5
Median	3	5	4	4	4	5	5	3	5

The *third section* of the questionnaire asked which artifacts should support TMT (Table 6). In background study we already found a connection between the activities and the artifacts from them. The survey results repeat the same pattern – test planning is seen as highly required, thus so is software test plan. Reporting activities and reports are of similar importance. However an interesting deviation exists – while traceability

between requirements, test plan, test cases and test results is seen as important, the SRS itself is not seen as required for a TMT.

Table 6. Which artifacts should support TMT?

	SRS	Software test plan	Test case	Software test library	Test suite	Defects	Reports	Test library per release
Mean	3,857	4,467	4,533	4,071	4,143	3,600	4,067	4,154
Standard deviation	1,027	1,125	1,060	1,265	1,457	1,056	1,100	1,231
Mod	4	5	5	4	5	4	5	4
Median	4	5	5	4	4	4	4	4

The *last section* of the questionnaire gathered the respondents' background information. On the whole, it gives a good overview of the target group characteristics by showing the distribution of the respondents.

When looking at the software development models used in the companies, two larger groups could be observed. Firstly, five respondent companies were using only agile models. The rest of the respondents stated, they are using mixed models with Agile being part of it. Thus the respondents are split into two groups representing 33% and 66% of the participants.

Another dissection of the results is based on companies' target market. There were no companies, who produce software only for Estonia. 13% developed software mostly for domestic, 53% mostly for international, and 34% only for international market.

When analyzing the survey results based on respondent company size, the respondents were almost split in 3, with employees in three ranges (Table 7). The only exception was one respondent, where the company size was between 11 and 25 employees. Table 8 shows similar equal distribution in the number of QA personnel, with a small deviation in the smaller QA teams.

Table 7. Size of the respondents' company

11-25 people	26-100 people	101-500 people	More than 500 people
1	5	5	4

Table 8. Number of QA personnel

Up to 5 people	5-10 people	11-25 people	26-50 people	More than 50 people
5	2	3	3	2

2.4. Specialized Results

Grouping the test results based on the respondent company characteristics (the last section of the questionnaire) provides us with another set of requirements for TMTs. In this paper, we focus on a target company as an example. One of the characteristics of this company is that it had 11-25 people in QA.

Analysis of the respondents belonging to this group reveals there are 9 features of TMT exceeding our set threshold: questions 4, 7, 9, 11, 12, 14, 17, 18 and 20. The mapping of the questions and to the features can be seen from Table 9 and Figure 1. Out of these 9, seven features were already identified as mandatory TMT features from the generic analysis. So the detailed data processing has revealed 2 new features (Test execution and Test suit), which are required features for companies with 11-25 people

in QA. By applying the same method for all companies' backgrounds, we identify all mandatory features dependant on company characteristics.

Table 9. Results grouped by company QA, namely for companies of 11-25 people (the Q meaning could be captured from Figure 1, where legend on the right gives the name for the analyzed property)

	Q1	Q2	Q3	Q4	Q6	Q7	Q8	Q9	Q10	Q11	Q12
MEAN	3,00	3,33	3,50	4,66	3,66	5,00	4,33	5,00	3,33	5,00	5,00
MEDIAN	3	4	3	5	4	5	5	5	3	5	5
	Q13	**Q14**	**Q16**	**Q17**	**Q18**	**Q19**	**Q20**	**Q21**	**Q22**	**Q23**	
MEAN	2,00	4,66	3,66	4,66	4,66	3,66	4,66	3,00	4,33	4,33	
MEDIAN	2	5	4	5	5	4	5	3	4	4	

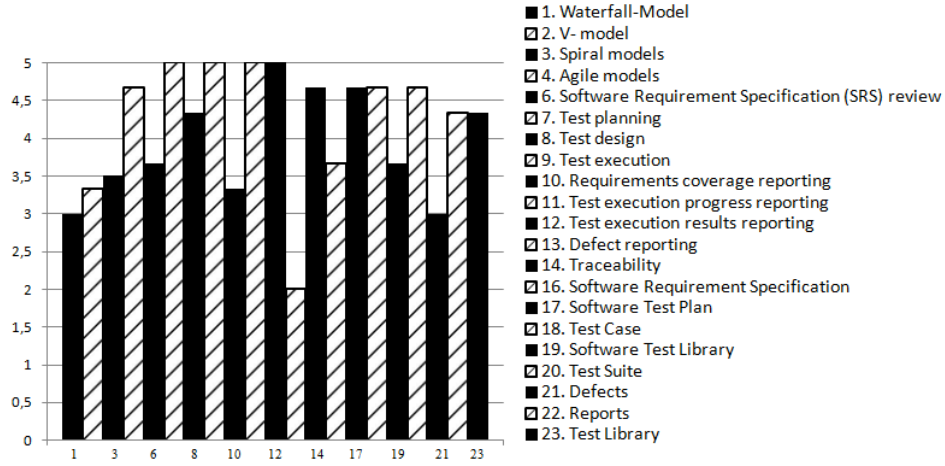

Figure 1. Result for respondents company QA (11-25 people)

3. Framework

The survey we performed provides us with the Estonian IT companies' expectations for TMTs. From the background info we find that majority of them are using different TMTs, some use in-house developed tools. The TMT evaluation framework proposed in this section, illustrates the rationale for making a decision when choosing a TMT. The framework also assists in evaluating existing TMTs by giving a clear measurable value of how the tool meets the users' company expectations.

3.1. TMT Requirements and Feature Diagram

We take the identified requirements from our questionnaire and map them to TMT feature diagram [9]. We use the mean values of all respondents from the questionnaire to find which features reached the 4.4 threshold. These are set as mandatory and all the rest as optional. This way we receive seven mandatory and fourteen optional features for a general TMT.

The *mandatory features* are: (*i*) support for agile models; (*ii*) test planning; (*iii*) traceability between different artifacts; (*iv*) reporting of test execution progress; (*v*) reporting of test execution results; (*vi*) software test plan artifact; and (*vii*) test case.

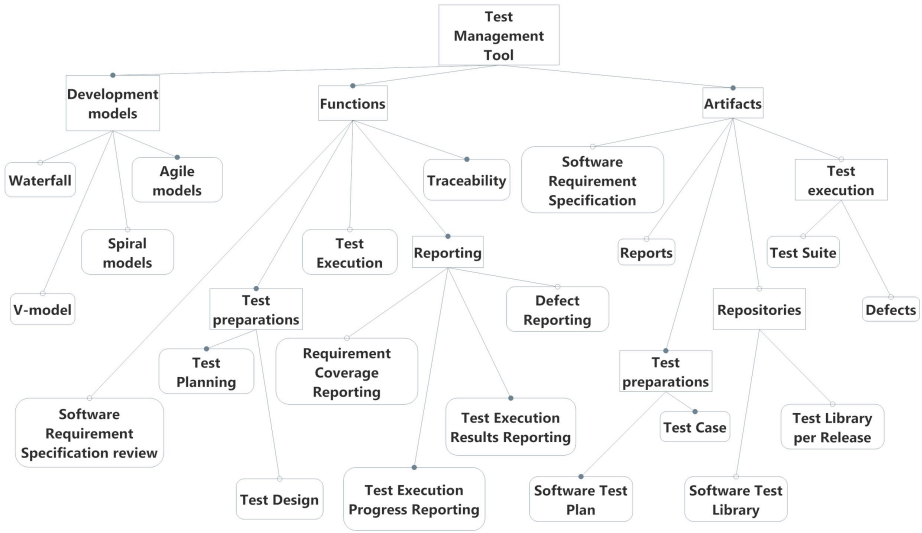

Figure 2. General TMT feature diagram (FD)

3.2. TMT Product Diagram

The online questionnaire results are grouped by the different characteristics of company backgrounds. There were four aspects:

- The software development model used in the company;
- The target market for company products;
- The company size measured in number of employees;
- The company QA department size measured in number of employees.

While the general TMT feature diagram informs which features are mandatory for a TMT in general, the results from the survey grouped by these aspects provide additional requirements specific for each group. We use the 4.4 threshold to identify the optional features of TMT per specific company characteristics when analyzing the results. We use here the term ’optional feature’ since the product diagrams are using this nomenclature. However from testing perspective, these features are also required. In order to get a specific product diagram, we merge the mandatory features from the general TMT FD and the features reaching the set threshold.

In Figure 3 we show an example of the product diagram. For instance from Table 9 we find features, which reach the set 4.4 threshold. These are the optional features for a TMT for a company with 11-25 QA personnel. We take the general TMT feature diagram mandatory features and overlap it with the additional features. Applying this approach to other company characteristics and overlapping the product diagrams, we get the company-targeted TMT product diagrams.

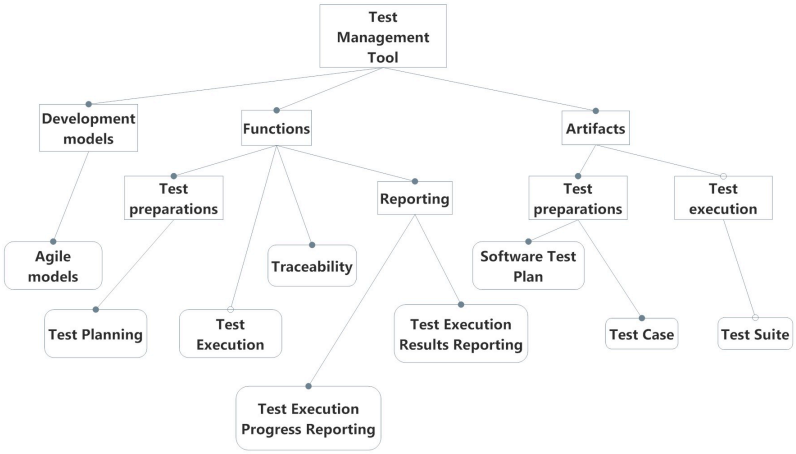

Figure 3. TMT product diagram for companies with 11-25 QA personnel

3.3. TMT Evaluation Framework and Guidelines for Application

There are several possible options for evaluating TMTs. One of them is counting the number of steps/clicks the evaluator has to make before the TMT activity or artifact is complete. Another option is to measure the time it takes to create an artifact or perform an activity. In this paper we investigate the TMT evaluation framework regarding its usability. But firstly to support framework usability we defined a six-step framework application process (see Figure 4).

1. Define evaluation purpose. The evaluator defines the evaluation purpose – either to understand the effectiveness of the existing TMT or to perform evaluation for procuring new TMT.

2. Determine company characteristics. This requires determining the company's characteristics, like company size, QA personnel size, the used development model, and target market for company's products. Based on these aspects, the evaluator takes the specific product diagrams. If the information about the company is un-known, then the general TMT feature diagram could be used.

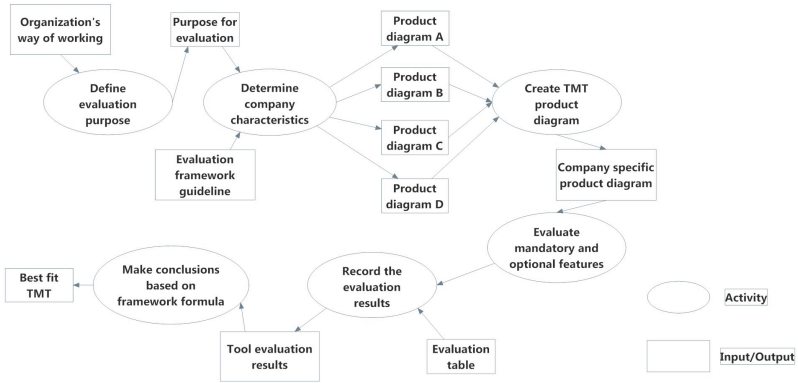

Figure 4. TMT evaluation framework processes

3. Create TMT product diagram. The next step is to overlap the product diagrams. Overlapping means taking all features from one product diagram and adding them to the next diagram, but not duplicating any of them. This completes a new company-specific product diagram. For example the overlapping of specific product diagrams for company which creates software mostly for domestic market *and* it has 11-25 QA engineers *and* it has 26-100 employees *and* they are using only agile development model is shown on Figure 5.

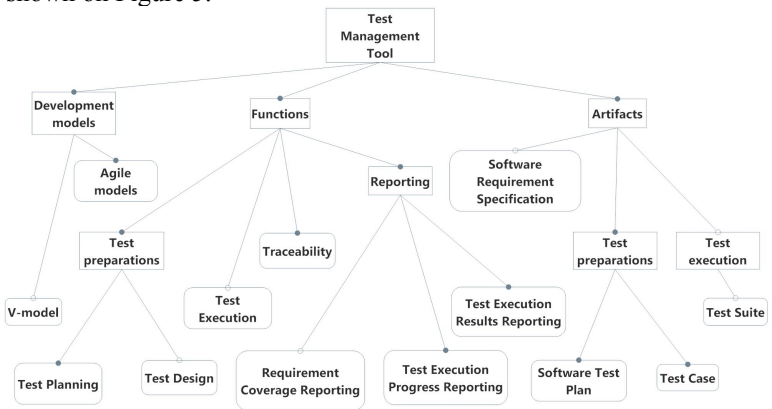

Figure 5. Company-specific TMT product diagram

Applying all four product diagrams returns the previously identified seven mandatory features. In addition, there are also new features. The six optional features of the TMT are: (*i*) support for V-model; (*ii*) test design; (*iii*) test execution; (*iv*) requirement coverage reporting; (*v*) software requirement specification; (*vi*) test suite.

4. Evaluate mandatory and optional features. The evaluator measures how well the listed features are supported by the TMT. Regardless of the chosen method, each mandatory and optional feature would get a value between 0 (does not meet the expectations) and 3 (meets the expectations to the full extent).

5. Record the evaluation results. The evaluator records the evaluation results. In our framework we use tabular format. The first column lists all TMT mandatory and optional features, the second column holds the value given to the feature. Table 10 in our example provides arbitrary values as reference how to fill it.

Table 10. TMT mandatory and optional features evaluation table

TMT feature	Value
V-model	1
Agile models	3
Test planning	1
...	...
Test suite	2
Total	7

6. Conclude based on framework formula. The final step is to draw the conclusions. In order to understand to which degree the TMT meets the company's expectations, we use formula: $p=\frac{\sum_1^n result_i}{n*3}*100\%$ where $result_i$ is the value a feature received, n is the total number of features identified from the company TMT product diagram. The number 3 is the highest value, which can be given. Following these provided guidelines,

the evaluator receives a result showing the percentage of how much the tool meets the expectations of the company.

4. Validation

4.1. Purpose of Validation

To confirm the usability of our framework, we carried out a case study among five QA specialists. The purpose of the study was to understand whether the framework fits tester expectations and to learn about framework's usability.

4.2. Participant Selection and Evaluation Framework Usability Interviews

Five QA specialists were contacted and asked to evaluate the TMTs their companies are using. The limit of the participants to five persons was derived from the relatively small number of Estonian IT companies. Secondly, we wanted to carry out a small proof-of-concept test, not to make a full research on the matter. We used personal approach and performed interviews with the respondents. This provided closer feedback and allowed us to ask additional specifying questions.

Firstly, we introduced to the participants the purpose of the framework. Secondly, the participants were given guidelines (Section 3.3) for using the framework and the product diagrams. After the TMT assessment, the participants were requested to answer the respond to the short questionnaire on the following concerns:

- How easy is the TMT evaluation framework to *learn*?
- How efficient is it for *frequent* TMT evaluation?
- How easy is it to *remember* the six steps of the framework?
- How *satisfied* are they with the framework?
- How easy it is to *understand the benefits* of the framework?

The goal of the interviews was to understand whether improvements should be made to the framework based on its usability.

4.3. Results

Each of the case study participants was asked to give feedback on the framework usability and to rate it on a scale of 1 (lowest) to 5 (highest). We provide the results of each respondent (R1-R5) in Table 11.

Table 11. TMT evaluation framework usability

	R1	R2	R3	R4	R5
How easy is the framework to learn?	3	4	4	4	4
How efficient is it for frequent TMT evaluation?	3	4	5	4	5
How easy is it to remember the six steps of the framework?	5	2	4	5	5
How satisfied are You with the framework?	2	4	4	4	4
How easy it is to understand the benefits of the framework?	2	5	5	4	4

How easy is the framework to learn? After applying improvements to the guideline, all respondents considered the framework easy to learn. People understood the workflow how the framework should be applied. They also implied that the six required steps were rather clear. The most difficult step for all respondents was the third step – creating TMT product diagram.

How efficient is it for frequent TMT evaluation? The respondents understood that for frequent use, the evaluator would only have to repeat steps 4-6, and the 1-3 steps would be performed once, since the company does not change frequently. Most respondents found that the framework is rather efficient for frequent use.

How easy is it to remember the six steps of the framework? Most of the interviewees responded that guidance steps are easy to remember. They said the steps are logical and quickly followed. One respondent did suggest shortening the step names, however to keep the framework easy to learn, we did not make the change.

How satisfied are You with the framework? This question turned out to be the hardest to answer. The participants had never used a framework for evaluating a TMT; it was new experience for them. While they did not say they are not satisfied with the framework, they were also reluctant to confirm, that it met their expectations. There was one exception to this– one of the test managers believed, that evaluating a TMT should be done by company employee and not based on a framework, since „the employee knows what is required by the company".

How easy it is to understand the benefits of the framework? All respondents understood clearly the benefits of the framework – mitigation of the subjectivity of evaluation by using an evaluation framework based on structured approach. Similar to the previous question, there was one outstanding respondent who strongly believed that the framework would not be beneficial for his company. Despite the outlying result, majority of the interviewees agreed that the benefits are rather easy to understand.

Finally, respondents were asked to bring out the best aspect of the framework. Three interviewees told that they got a clear number representing how much the tool met with the company expectations. The other two agreed that the framework is excellent for frequent use and saves time.

5. Threats to validity

There are very few studies that are completely unbiased. With the current research we strive to exclude any threats to validity as well as possible within the scope of the paper. In [7] Good *et al* report that the bias is typically caused from sample selection, study conduct, and results interpretation. Regarding our study we observe the following:

- The research target is the geographical area, but not the domain. The survey was carried out among Estonian companies;
- The sample size of respondents is relatively low. 15 respondents is a small subset of all IT companies in Estonia. The threat is compensated by a relatively high response rate (50%) and the companies belonging to the top software development companies operating in Estonia;
- Threat with the self-administered questionnaire is to state the questions in a comprehensive way [5]. To achieve a common understandability the definitions were provided;
- The risk of human error with submitting responses. We mitigate the risk by determining the outlying answers and excluding them from the data set;

Threat to the validity of the survey comes from the interpretation of the results. Following Good *et al*'s suggestions [7], we have done collaboration between the statistician and the domain expert, which is essential if all sources of bias are to be detected and corrected.

6. Conclusion and future work

In this paper we have developed a framework to evaluate test management tools. Such a framework could provides test and project managers the understanding whether their current TMTs meet the company's expectations and goals or to decrease subjectivity during new TMT assessment acquisition. We have validated usability of the TMT evaluation framework in an interview study among quality assurance specialists. The interview results confirm that the TMT evaluation framework is:

- *easy to learn*. The case study results in Table 11 show that the evaluation method is easy to remember. The framework users found the framework steps clear (see Table 11). Its application results in a measurable value for TMT assessment;
- *efficient for frequent use*. Applying the framework for several TMTs only requires the user to define the product diagram once, thus saving the evaluator the effort of re-defining expectations for every evaluation (see Table 11).
- *fit for purpose*. Our framework has been created based on the current Estonian ICT companies' expectations for TMTs. The framework has been tested by industrial software testers and their feedback has been taken into account to understand framework's usability. This potentially the subjectivity of single person TMT evaluation.

As for the future work we plan to expand the scope including insights from other countries (not only Estonia). Also expanding the TMT requirements based on new trends in testing could be taken into consideration.

References

[1] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques,* Springer Publishing Company, 2010.

[2] Britannica, *Merriam-Webster's Collegiate Dictionary, 11th Ed*, Merriam-Webster Inc., 2003.

[3] J. Keyes, *Software Engineering Handbook*, Auerbach Publications (CRC Press), 2003.

[4] A.B.Tucker, *Computer Science Handbook, 2nd Ed*, Chapman and Hall/CRC, 2004.

[5] R. Matulevičius, *Aggregated Process For Evaluating Requirements Engineering Tools*, IT 2009, Kaunas.

[6] C. Wohlin et al, *Experimentation in Software Engineering: An Introduction*, Springer Publishing Company, 2000.

[7] P.I. Good et al, *Common Errors in Statistics, 3rd Ed*, Wiley-Blackwell, 2009.

[8] G.E. Noether, *Introduction to Statistics The Nonparametric Way*, Springer Publishing Company, 1991.

[9] A. Deursen et al, *Domain-Specific Language Design Requires Feature Descriptions*, Journal of Computing and Information Technolog, 2001.

[10] List of Test Case Management Tools URL: http://www.softwaretestingclub.com/forum/topics/looking-for-a-new-test-case. Last retrieved 21.March, 2012

[11] A. Myrvold, *11 software testing myths*, Alan Myrvold's Software testing blog. URL: http://testapprentice.com/2011/02/21/software-testing-myths/. Last retrieve 21.March, 2012

[12] A Community for Software Testers, Software Testing Club, URL: http://www.softwaretestingclub.com/forum/topics/looking-for-a-new-test-case, Last retrieve 21. March, 2012